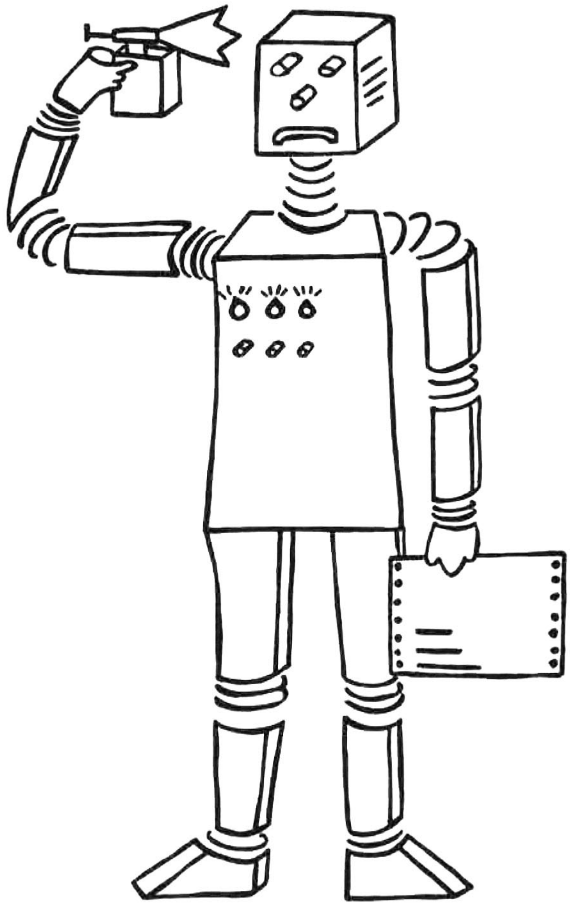
SEKI-REPORT

Artificial
Intelligence
Laboratories

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



# Computational Aspects of an Order-Sorted Logic with Term Declarations

Manfred Schmidt-Schauss

July 1988 SEKI Report SR-88-10

# Computational Aspects of an Order-Sorted Logic with Term Declarations

Vom Fachbereich Informatik
der Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation

von

Dipl.-Math. Manfred Schmidt-Schauß

Berichterstatter: Prof. Dr. Jörg H. Siekmann
Prof. Dr. Jean-Pierre Jouannaud
Dekan: Prof. Dr. Theo Härder
Tag der wissenschaftlichen Aussprache: 26.4.1988

D 386

# Abstract.

In this thesis I investigate the logical foundations of a very general order-sorted logic. This sorted logic extends usual first order logic by a partially ordered set of sorts, such that every term is of a particular sort or type, in addition there is a mechanism to define the sort of terms using term declarations. Syntax and semantics of this order-sorted logic with declarations are defined in a natural way.

Unification in order-sorted logics with term declarations is undecidable and infinitary, i.e., minimal complete sets of unifiers may be infinite. However, under the restriction that declarations are only of the form $f:S_1\times\ldots\times S_n \rightarrow S$ and that the signature is regular, unification is decidable and minimal complete sets of unifiers exist and are always finite. Furthermore there exists a signature of this form such that unification is NP-complete.

If there is no equality predicate in the logic we use resolution and factoring as inference rules, where the unification algorithm is adapted to the sort-structure. The corresponding calculus is refutation complete.

If there is an equality predicate and all equational literals are in unit clauses, we use a special E-unification algorithm and show that under some restrictions such an algorithm can be constructed from an unsorted unification algorithm by postprocessing the set of unifiers.

If arbitrary equations are admissible, we use paramodulation as additional inference rule or replace resolution by the E-resolution rule.

An algorithm for transforming unary predicates into sorts is presented. It is shown that this algorithm is correct and complete under sensible restrictions. Usually, the algorithm may require exponential time, however, in the special case of Horn clauses the algorithm can be performed in polynomial time.

We also investigate term rewriting systems in an order-sorted logic and extend the confluence criterion that is based on critical pairs by critical sort relations.

# Zusammenfassung.


In dieser Arbeit untersuche ich die logischen Grundlagen einer sehr allgemeinen ordnungssortierten Logik. Diese sortierte Logik erweitert die übliche Logik erster Stufe um eine partiell geordnete Menge von Sorten, so daß jeder Term eine bestimmte Sorte (Typ) hat. Zusätzlich gibt es einen Mechanismus zum Definieren von Termsorten mittels Termdeklarationen. Syntax und Semantik dieser sortierten Logik werden auf natürliche Weise definiert.

Unifikation in ordnungssortierten Logiken mit Termdeklarationen ist unentscheidbar und infinitär, d.h., minimale und vollständige Mengen von Unifikatoren können unendlich sein. Unter der Einschränkung, daß Deklarationen nur von der Form $f{:}S_1 \times \ldots \times S_n \rightarrow S$ sein dürfen und die Signatur regulär ist, erhält man daß Unifikation entscheidbar ist und daß minimale Mengen von Unifikatoren immer endlich sind. Weiterhin gibt es eine solche Signatur, in der Unifikation NP-vollständig ist.

Wenn kein Gleichheitsprädikat in der Logik ist, kann man Resolution und Faktorisierung als Ableitungsregeln benutzen, wobei der Unifikationsalgorithmus an die Sortenstruktur angepasst ist. Der zugehörige Kalkül ist widerspruchsvollständig.

Wenn ein Gleichheitsprädikat vorhanden ist und alle Gleichungen in Unitklauseln vorkommen, kan man einen speziellen E-Unifikationsalgorithmus benutzen. Wir zeigen, daß man unter gewissen Bedingungen einen Algorithmus aus einem unsortierten Unifikationsalgorithmus und einer Nachbearbeitung der Menge der Unifikatoren konstruieren kann.

Wenn beliebige Gleichungen erlaubt sind, benutzt man Paramodulation als zusätzliche Ableitungsregel oder man ersetzt Resolution durch die E-Resolution.

Es wird ein Algorithmus zum Transformieren einstelliger Prädikate in Sorten vorgestellt. Von diesem Algorithmus wird gezeigt daß er unter gewissen sinnvollen Einschränkungen korrekt und vollständig ist. Der Algorithmus hat normalerweise exponentielle Zeitkomplexität, aber im Spezailfall von Hornklauseln kann der Algorithmus in polynomialer Zeit ausgeführt werden.

Wir untersuchen auch Termersetzungssysteme in einer ordnungssortierten Logik und erweitern das auf kritischen Paaren beruhende Konfluenzkriterium um kritische Sortenrelationen.

## Acknowledgements.

I would like to thank my supervisor Jörg Siekmann. He introduced me into the field of Artificial Intelligence and Automated Deduction. His enthusiasm, guidance and critics were indispensible for writing down this thesis. I thank him for his final revision of this thesis.

Hans Jürgen Ohlbach's contributions are manifold. He poses the 'sort-generation'-problem and thus caused me to investigate sorted logics. His experience helped in many cases to recognize foolish ideas and to avoid dead ends and black holes.

Alexander Herold introduced me into the field of Unification and we had a lot of fruitful discussions concerning unification and subsumption.

I acknowledge discussions with Gert Smolka concerning sorted algebras and semantics and for explaining me the ideas of the order-sorted algebra approach of Goguen and Meseguer.

Jean-Pierre Jouannaud carefully read a preliminary version of the part on unification with term declarations.His hints and ideas contributed to the present form of this thesis.

I am particularily grateful to Hans-Jürgen Bürckert for his thorough reading of a draft of this thesis and for the time he spent in many discussions.

I am grateful to Norbert Eisinger for reading a draft of this thesis.

Finally, I thank my wife Marlies for her patience during finishing this thesis.

## Statement:

This thesis evolves from the previous papers on sorted calculi [Sch85a, Sch85b] and unification of sorted terms under term declarations [Sch85d]. The results of paragraph IV.3 are published in a modified form in [Sch86a]. Part VI is a revision of the report on mechanical sort generation [Sch85c]. The appendix will be published in [BHS87].

# Contents:

# Introduction.

## Motivation.

The investigation of logical calculi suitable for an implementation on the computer and the development of methods for the reduction of search spaces are essential tasks in the field of Automated Deduction. The distinction of objects into different classes, called sorts, for example points, lines and planes in geometry, and the exploitation of this information in the search for a proof is a very promising technique for many problems (such as Schubert´s steamroller [Wa83, Wa85, St86]). The proposed techniques of using sort information have the additional advantage that they can be combined with most other known methods in use for the reduction of search spaces, such as the standard search strategies [Lo78, CL73], the building in of equational theories [Pl72] or techniques for the building in of arbitrary theories [St86].

First order logic is often used to describe facts or relations that hold in some domain D. Given some facts that hold in D the deduction methods of first order logic can be used to deduce new facts that are true for the domain D. In the standard first order predicate calculus the knowledge that some objects in D are of a certain type or belong to a particular subset is expressible only using a unary predicate and also there are no restricted quantifiers. For example, the variable x in the formula: $\forall x\ \ Nat(x) \Rightarrow x \geq 0$ ranges over all possible objects. This has the undesired effect that formulae like $Nat(c) \Rightarrow c \geq 0$ can be deduced for all objects c, even if $c \geq 0$ does not make sense, for example if c is a list. The essential idea in a many-sorted logic is to distinguish different sorts of objects and to restrict the scope of variables to a particular sort. For example, after introducing the sort (or type) NAT, the formula above reads $(\forall x{:}NAT\ x \geq 0)$. In this formula the variable x ranges only over objects of sort NAT.

Using this idea as a starting point for a modification of the syntax and deduction in first order logic, several other concepts and extensions arise naturally:

One may need a set S of sorts that is partially ordered.

If we consider a term t as a function with input from our object domain D and a value in D, where the inputs have to obey the sort of variables, then in general the value produced by t does not range over the whole set D, but over a smaller subset. This range of values should be syntactically reflected and hence to every term t a sort should be assigned. Since we have functions in our logic and hence there are compound terms, there is the need for a method to compute the sort of terms. Usually, this is done by specifying functions with declarations like $f:S_1 \times \ldots \times S_n \rightarrow S$, where $S_i$ are sort names. Hence the sort of terms is usually computed from the range sort S of the top level function symbol. An equivalent method to specify the sort of terms is to use term declarations of the form $f(x_{S1},\ldots,x_{Sn})$: S. As a generalization we allow term declarations of the form t:S, where t is an arbitrary term. This is a very general method to specify the sort of terms.

In addition we need the concept of a well-sorted substitution that substitutes only admissable terms for sorted variables. For example, we may have the sort-structure INT $\sqsupset$ NAT, the variable $x_{NAT}$ with sort NAT as above and now the substitutions are to replace $x_{NAT}$ only by terms of sort equal or less than NAT.

A further concept is that of the sorted domain of a predicate, i.e., a predicate accepts only certain combinations of sorted arguments, otherwise the expression is ill-sorted.

We shall call a logic with these ingredients an **order-sorted logic** in order to emphasize that subsorts are permitted and we shall reserve the word many-sorted for logics that use unrelated sorts. Note that some authors use many-sorted logic also for logics with subsorts.

The following specification of even numbers is an example for term declarations:

EVEN $\sqsubset$ NAT, 0:EVEN, s:NAT$\rightarrow$NAT,

$s(s(x_{EVEN}))$: EVEN.

This gives recursively the terms of type EVEN: 0, s(s(0)), s(s(s(s(0))),... , which correspond to the even numbers 0,2,4,... .

Using the above specification of even numbers we can exemplify the use of well-sorted substitutions and sorted unification:

Consider the two statements $\forall x_{EVEN}\ P(x_{EVEN})$ and $\forall y_{EVEN}\ \neg P(s(s(y_{EVEN})))$. These two formulae are contradictory, since the well-sorted substitution $\{x_{EVEN} \leftarrow s(s(y_{EVEN}))\}$ gives an obvious contradiction. However, the two formulae $\forall x_{EVEN}\ P(x_{EVEN})$ and $\forall y_{EVEN}\ \neg P(s(y_{EVEN}))$ are not contradictory. The necessary substitution $\{x_{EVEN} \leftarrow s(y_{EVEN})\}$ is not well-sorted, since the term $s(y_{EVEN})$ is not of sort EVEN but of sort NAT.

If we again slightly change the above example, we see how unification has to be extended: Consider the two formulae $\forall x_{EVEN}\ P(x_{EVEN})$ and $\forall y_{NAT}\ \neg P(s(s(y_{NAT})))$. The substitution $\{x_{EVEN} \leftarrow s(s(y_{NAT}))\}$ is not the right one, since it is not well-sorted. So unification has to try to make it well-sorted. A substitution which makes the two formulae contradictory is $\{x_{EVEN} \leftarrow s(s(z_{EVEN})), y_{NAT} \leftarrow z_{EVEN}\}$, that is the variable $y_{NAT}$ is weakened to sort EVEN by substituting $z_{EVEN}$. This example shows that usual unification has to be extended by a weakening step.

Sorts also provide a means for combining many inferences into one formula. Consider for example the following Horn-clause variant of the above problem:

$EVEN(0)$,

$\forall x\ EVEN(x) \Rightarrow EVEN(s(s(x)))$,

$\forall x\ EVEN(x) \Rightarrow P(x)$

The query $\exists y\ EVEN(y) \wedge P(y)$ would produce an infinite number of answers $y = 0$, $y = s(s(0))$, ... , $s^{2*n}(0)$,... .

A sorted formulation of this problem is

$0{:}EVEN$,

$s(s(x_{EVEN}))$: EVEN

$\forall x_{EVEN}\ P(x_{EVEN})$

The corresponding sorted query $?P(y_{EVEN})$ would produce only one answer, namely $y_{EVEN} = x_{EVEN}$, which has the meaning that all terms of sort EVEN are allowed as answers.

The next step in order to obtain a more powerful deduction calculus for a wider range of well-sorted formulae is to have equality as a distinct predicate. The semantic aspect of such a logic with equality and sorts is relatively straightforward, but is not as intuitive as it is without equations. For example there may be a gap between the syntactic sort and the semantic sort of objects: if there is a sort structure and an equational theory, which for some reasons allows the deduction of $s = t$ for every two terms (i.e., it is inconsistent), then every model has exactly one element and all semantical sort domains are equal, whereas the syntactical sorts are all different.

The computational aspects of a logic with equality and sorts causes even more difficulties. If paramodulation is extended in the natural way, then it may be possible to infer ill-sorted formulae. If for example the unrelated sorts A and B are in the signature, and also there are constants a:A and b:B, a predicate P, which accepts only terms of sort A, then let the formulas be $a = b$ and $P(a)$. A replacement of a by b (i.e. by paramodulation) gives the ill-sorted formula $P(b)$. There are more complex and more natural sets of formulae with no obvious way of how to avoid the deduction of such ill-sorted formulae. For example if there is an injectivity clause of the form $\forall x,y{:}\ \ x = y \vee f(x) \neq f(y)$, then paramodulating with the equation $x = y$ is a potential source for plenty of such ill-sorted paramodulants. In this thesis we will present several approaches to solve this problem.

Of course sorts can be encoded using unary predicates and the sorted part of the signature can then be interpreted as a set of (Horn-) clauses, that allows to deduce the sort of a term. This translation process yields for every sorted clause set an equivalent unsorted one, which is called the relativized clause set [Ob62, Sch 38]. The converse problem whether a unary predicate can be interpreted as a sort, or how to encode a certain problem with a sorted specification, is more difficult and is the subject of chapter VI. In particular it would be useful to have a translation process, such that an Automated Deduction System can find equivalent representations by itself and decides which of these representations is more appropriate for the search process.

**Related Work.**

The use of sorts or types in logic dates back to J. Herbrand [Her30, Her71]. His completeness proof of the sorted calculus was not correct, however, as pointed out by A. Schmidt [Sch38]. The completeness of a calculus for a many-sorted logic with function symbols is proved correctly in [Sch38, Sch51]. All these logics are somehow restricted: the many-sorted logic considered by H. Wang and P. Gilmore [Wan52, Gi58] has no function symbols and all the many-sorted logics in [Her30, Sch38, Wan52, Gi58] do not make use of subsorts. The extension investigated by T. Hailperin [Hai57] allows the restriction of the quantification of a variable by arbitrary formulae. This seems to be too general an extension for deduction systems, since in this calculus one needs the full power of first order calculus to infer if a formula is well-sorted.

The most interesting formulation of many-sorted logics for our purposes is that of A. Oberschelp [Ob62]. He describes several different many-sorted logics. In his S-logic function symbols, multiple assignments for functions and subsorts are allowed. He gives a clear Tarski-type semantics, which is the same as ours. To our knowledge he was the first to introduce a notion of order-sorted algebra. His Σ-logic uses a relation on variables and terms to specify the sort of a term, which is similar to the R-systems in this thesis. However, term declarations are not allowed in the Σ-logic. All these classical sorted logics had no notion of unification or of a most general inference.

Sorts were recognized as an important tool for Artificial Intelligence and Automated Deduction by P. Hayes [Hay71], who allows unrelated sorts and multiple sort range assignments per function symbol.
More recently, sorted logics were investigated as useful tools for Automated Deduction by Ch. Walther and A. Cohn [Wa83, Co83]. Ch. Walther [Wa83] developed a calculus based on resolution and paramodulation, which allows subsorts and equations, but only one declaration per function symbol. His paper was the first to combine resolution and sorts using a sorted unification algorithm. The completeness proofs in [Wa83] are obtained by a transformation of the classical completeness proofs and the semantics given there are defined via relativizations. Ch. Walther demonstrated with many examples, including the now well-known Steamroller example [Wa85], that his logic is a powerful technique for avoiding redundancies in the search

for a proof.

A. Cohn [Co83] considers a more general calculus which allows multiple function declarations per function. His logic is more expressive than Walther´s, since some unit clauses may be built into the logic (polymorphic predicates), however, there are no equations in his logic. His evaluation rule competes with unit deductions as in PROLOG [CM81] or with the terminator algorithm described by G. Antoniou and H.J. Ohlbach [AO83]. Cohn´s logic has the advantage of small initial clause sets, but the drawback of more deduction rules.

The many-sorted logic of K.Irani and D. Shin [IS85] has a dynamic sort-structure, but it may be too heavy a machinery for most practical purposes, since one can think of the sort structure as virtually fixed and hence use some standard many-sorted logic, and let the program generate the sorts only if needed.

Our approach to a many-sorted logic follows the lines of [Ob62] and [Wa83]. A characteristic of this approach is that once the signature is given, all terms have a fixed sort. For some applications this may be a disadvantage, for example the situation where one knows that A is a person, but one does not know whether A is male or female, is not expressible in this logic. In other words the sort of a term is computable given the signature, but not deducible from some given statements. This is clearly a restriction, but it allows for fast algorithms to compute the sort of a term.

An approach that is very close to ours is that of G. Smolka [Sm86], who employs order-sorted algebra in the development of an order-sorted Horn-logic. Further work with similar semantics is carried out by W.W. Wadge [Wad82], who gives in fact a semantics for specifications that allow declarations. Our semantics is also similar to the semantics in J.A. Goguen and J. Meseguer [GM85a], but their notion of homomorphisms as a family of mappings is based on the many-sorted approach and seems to be not the optimal one.

In the field of algebraic specifications [EM85] the use of sorts is a common technique, however, usually no subsorts are admitted and just one declaration per function is allowed. This was extended to subsorts and term declarations by J.A. Goguen [Gg78, GM85a], who introduced the notion 'order-sorted algebra' to indicate that subsorts are permitted. Sorts are mainly used in this field in order to give the semantics of specifications in the form of initial algebras and to support an appropriate handling of errors [GM85a, Go83].

Most Programming Languages use type systems for different purposes, such as type checking at compile time, error detection, modularization of programs and for efficient programming (cf. [HLS72, Ml78, Ml84, BB86, Go83, Go86, Sn86, SH85, Tu85] These languages are designed such that there is either no or at least only a small amount of type handling at run time. Many-sorted unification is used in a type-checking system described by G. Snelting [Sn86, SH85, BS86]. In the specification languages OBJ2 [FGJM85] and EQLOG [GM85b] the handling of sorts is done at run time and there is also the need for sorted equational unification in order to have an appropriate operational semantics.

The combination of equational deduction and sorts for term rewriting systems was investigated by R.J. Cunningham and A.J.J. Dick [CD83], G. Huet and D. Oppen [HO80] and by J.A. Goguen, J.-P. Jouannaud and J. Meseguer [GJM85]. The system in [CD83] is unfortunately inconsistent without additional restrictions. A translation of order-sorted term rewriting to many-sorted term-rewriting is described in [GJM85].

Order-sorted deduction and narrowing are considered by G. Smolka, W. Nutt, J.A. Goguen and J. Meseguer [SNMG87] and order-sorted unification also in [MGS87]. A notion of 'meta'-variables and domains which converge to a sorted logic is given by H. Kirchner [HKi87] in order to handle term rewriting systems with an infinite number of rules.

Unification under sorts originated with the papers of Ch. Walther [Wa83, Wa84]. The handling of sort-arrays [Co83a, Co83b] is also a type of sorted unification. In [CD83] a sorted unification algorithm is used and it is recognized that a complete and minimal set of unifiers may be finite for elementary signatures, but a proof for the correctness of this algorithm is not given. Unification for polymorphic signatures is proved to be of unification type finitary by the author in [Sch85].

The extension of many-sorted logic by term declarations was proposed by J.A. Goguen [Gg78] and term declarations were later called sort-constraints [GM85a]. These sort-constraints are more general than term declarations, but this generality necessitates the use of deductions to obtain the sort of a term. In fact the sort of a term may be undecidable. Our term-declarations correspond to unconditioned sort-constraints. Other work using term declarations is described in [Go83, Wad82].

## Overview.

In this thesis we investigate order-sorted logic and its computational part. The logic allows subsorts, term declarations and equations but provides only a fixed sort of a term. The general aim of this work that motivated the design of our logic is to identify those computations with sorts that can be done efficiently. A further guideline was that the resulting logic should be intuitive and simple. In general we concentrate on finite sets of sorts, although most of the results hold also for an infinite set of sorts. All computability and efficieny considerations are made only for finitely many sorts, we do not consider deductions with empty sorts (cf. [GM81, GM85]). The logic is constructed such that all connnectives and quantifiers of first order logic can be used and a formula in this logic has the familiar shape, besides the fact that instantiation into variables is now restricted.

Although we prefer to use resolution and paramodulation-based calculi, most classical calculi and all types of refutation calculi (for example [Ro65, RW69, And81, Bib81b, Ri78] can be adapted to this sorted logic. Also equational logic and term rewriting systems [HO80, Hu80, Bu87] can be adapted.

In part I we give an account of the foundations of this logic, its algebraic treatment and a semantics based on a type of order-sorted algebra, called $\Sigma$-algebra, which is conceptually closer to [Wad82] than to [GM85a]. We extend the equational logic and Birkhoff's Theorem to sorted term algebras. Note that the straightforward solution is impossible, since it would mean the deduction of ill-sorted terms, whereas our solution allows only well-sorted terms in the deduction process. The same problem arises for term rewriting systems and in order to solve this problem, some new concepts are needed, which are described in I.12 and II.3. In paragraph I.13 we work out the rule-based approach to unification which first appeared in A. Martelli and U. Montanari's paper [MM82] and was used for an equational unification procedure in [CKi84, CKi85, Cki87, MGS87]. This approach has advantages over the usual extensions of the Robinson approach [Ro65], since the basic unification operations and the control strategy are separated. The last paragraph gives a comparison between different views of unification as a process of solving equations.

In part II we show that the distinction between well-sorted and ill-sorted formulae is not an essential one. The satisfiability of a formula does not change, if we modify the signature and consider all ill-sorted expressions as well-sorted. This justifies our assumption in the following that we can ignore the problem of the deduction of ill-sorted formulae and can always assume that all formulae are well-sorted. However, the restriction remains that only well-sorted substitutions and instantiations are to be used. Paragraph 3 gives a general condition for a term rewriting system to be compatible and canonical. It also contains a completion procedure for ground term rewriting systems. In paragraphs 4 and 8 we give several equivalent formulations for a sorted signature with term declarations, including an infinite set of term declarations. In §6 we investigate the properties of deduction-closedness, congruence-closedness and sort-preservation, which arise in combining sorts and equations and we shall give criteria to check these conditions, given the signature and the axioms of an equational theory. In §7 we investigate conservative transformations of the signature. This is a preparatory work for the sort generation process in Part VI. In paragraphs 10 and 11 we consider different encodings of sorted logic into first order logic and show the important Herbrand Theorem also in the context of sorts and equations.

Part III and IV of this thesis are devoted to unification algorithms, where part III gives results on unification of sorted terms without equational theories. We show that unification in elementary regular signatures is decidable and finitary and that in general unification is undecidable and may be infinitary, but minimal sets of unifiers do always exist and are recursively enumerable. Furthermore we investigate the complexity of unification for different types of signatures.

In the case of an equational theory, we give a rule-based complete sorted unification procedure. Here the problem of functionally reflexive axioms arises and we give an example that in general they are needed for sorted equational unification. In the unsorted case they are not needed: the paramodulation-based algorithm of P. Padawitz [Pa86] works without them and the algorithm of J.H.Gallier and W.Snyder [GS87] needs functional reflexive axioms only for the special case to eliminate occur-check failures.

If more restrictions are given like sort-preservation and congruence-closedness, then a unification algorithm can be generated from an unsorted one and a weakening procedure as postprocessor. We show in IV.3 that this is a complete unification procedure and that in the case of elementary signatures the algorithm is well-behaved. We demonstrate how to use this combination algorithm for AC and ACI function symbols.

In part V we show that a resolution-based calculus with a sorted unification algorithm is a complete refutation procedure. We demonstrate that resolution together with paramodulation provides a refutation-complete calculus for equality if the functionally reflexive axioms are in the clause set. The proof method uses Herbrand`s Theorem and the usual lifting-arguments. We give an example that the functionally reflexive axioms are necessary in general, even in the case of elementary signatures. An alternative to paramodulation is J. Morris` E-resolution [Mo69]. We propose to use it in combination with rigid E-unification as defined by J.H.Gallier, S. Raatz and W.Snyder [GRS87] for deductions of equational matings [And81]. In paragraph 6 we extend M. Stickel's theory-resolution [St85] to a sorted signature.

Part VI is a description of an algorithm that manipulates clause sets and signatures in order to obtain a clause set with respect to a more sorted signature. The idea is to have a relatively fast transformation algorithm and to make the deduction on the transformed clause set, where more sort-information is given and hence the search space is smaller than in the original clause set. We prove that this procedure is correct and give conditions for it to be complete. We adapt the algorithm to sets of Horn clauses, such that it can be used to transform logic programs into more sorted versions. For the case of Horn clauses this algorithm is shown to be of polynomial time complexity.

# Part I
# Foundations

Overview: In this part we develop the frame-work for an order-sorted logic with equality, its syntax, semantics and its computational aspects. We define order-sorted signatures and show that the term algebras provide free and initial algebras with respect to our notion of a signature. A notion of $\Sigma$-model for arbitrary clause sets is presented.

The rest of this part is devoted to the consequences of the combination of sorts with equational theories for unification problems and for term rewriting systems.

## 1. Preliminaries.

We use the usual set theoretical symbols $\in$, $\subseteq$, $\cap$, $\cup$ for the membership relation, subset relation, intersection and union of sets and abbreviate $A_1 \cup \ldots \cup A_n$ by $\cup\{A_i | i = 1,\ldots,n\}$. The set difference is denoted by A–B and the powerset of a set A is denoted as $\mathcal{P}(A)$. The n-fold direct product of a set A is denoted by $A^n$ and the empty set is denoted by $\emptyset$. For partial functions f: A $\rightarrow$B we denote the domain of f, i.e. the subset of A where f is defined, by $\mathcal{D}(f)$. A function f: A $\rightarrow$B with $\mathcal{D}(f) = A$ is called a total function. By $\mathbb{N}$ we denote the set of natural numbers, including zero.

A reflexive and transitive relation $\leq$ on a set A is called a **quasi-ordering**. A quasi-ordering $\leq$ naturally generates an equivalence relation $\equiv$, such that $a \equiv b \Leftrightarrow (a \leq b$ and $b \leq a)$. The **equivalence-class** in A with respect to $\equiv$ is denoted as $[a]_{\equiv}$. We use $a < b$ to denote that $a \leq b$, but not $a \equiv b$. The notations $\geq$ and $>$ have their obvious meaning. A subset B of A is called a **lower segment** if it is downward closed, i.e., for $a \in A$ and $b \in B$: $a \leq b$ implies $a \in B$. Accordingly we define an **upper segment.** Note that unions and intersections of upper segments (lower segments) are again upper segments (lower segments).

An element 'a' is **minimal (maximal)** in A, iff for all $b \in A$: $b \leq a$ implies $a \leq b$ (iff for all $b \in A$: $b \geq a$ implies $a \geq b$). With $[-\infty,a]$ we denote the lower segment of all elements that are less than or equal to a; similarily we define $[a,\infty]$. A quasi-ordering is **linear** (or a chain), iff $a \leq b$ or $b \leq a$ for all elements. It is **well-founded**, iff every chain has a minimal element. An antisymmetric quasi-ordering is called a **partial ordering**.

Let U be an upper segment of the quasi-ordered set A. A **complete** subset cU (or a generating subset) of U with respect to A has the property: $\forall u \in U\ \exists\, v \in cU$: $v \leq u$. The set of all complete subsets of U is sometimes denoted by C(U). A **base** B for an upper segment U of A is a complete set of representatives of minimal elements of U. A base is also called a minimal, complete subset. As notation we have $\mu U$ for a special base B and M(U) for the set of all bases. Not every upper segment U has a base, but if it has one the base is unique, that is two bases $B_1$ and $B_2$ of U are equivalent, in the sense that there exists a bijection $\psi$: $B_1 \to B_2$ such that $\psi(b) \equiv b$ for all $b \in B_1$. This is almost trivial for quasi-orderings and was first proved for minimal sets of unifiers by Fages and Huet [FH83]. The cardinality of a base of an upper segment U is an invariant of U.

In a partially ordered set A a **greatest lower bound** (g.l.b.) for two elements a,b is a unique element $g \in A$ with $g \leq a,b$, such that for every $c \leq a,b$ we have $c \leq g$. Dually we can define a **least upper bound** (l.u.b) for two elements in A. This definition can be lifted to finite subsets of A. A partially ordered set A in which for all elements a,b their g.l.b. and l.u.b. exists is a **lattice** with operators glb(a,b) and lub(a,b). We say a partially ordered set A is a **semilattice**, iff for all elements $a,b \in A$, a least upper bound lub(a,b) exists. For a finite set A an equivalent property is that i) for all elements $a,b \in A$ that have a common lower bound c ($c \leq a,b$), a greatest lower bound exists for a,b and ii) that A has a maximal element.

**Multisets** are like sets, but allow multiple occurrences of identical elements. The operations on sets are adapted to multisets, for example $M - \{a\}$ means to delete one occurrence of a in M. If we have a well-founded partial ordering on the elements of a multiset M, then we can construct recursively a well-founded multiset-ordering [DM79, De87] on multisets as follows: $M > N$, if for some $a \in M$ and $b_i \in N, i = 1,\ldots,n$ : $a > b_i$, $i = 1,\ldots,n$ and $M - \{a\} > N - \{b_1,\ldots,b_n\}$.

## 2. Symbols, Terms and Substitutions.

In the following we will use the bar $^{-}$ to indicate that objects are unsorted, in particular for unsorted signatures, since we later define sorted signatures as composed of unsorted ones plus additional symbols and properties.

An **unsorted signature** $\bar{\Sigma}$ consists of the three pairwise disjoint sets of symbols.

- $\mathbf{F}_{\bar{\Sigma}}$ the set of **function symbols**. Elements are denoted by f,g,h.
- $\mathbf{V}_{\bar{\Sigma}}$ the countably infinite set of **variable symbols**. Elements are denoted by x,y,z.
- $\mathbf{P}_{\bar{\Sigma}}$ the set of **predicate symbols**. Elements are denoted by P,Q.

Every function symbol f has a nonnegative arity and every predicate symbol P has a positive arity denoted by arity(f) and arity(P), repectively. In the following the suffix $_{\bar{\Sigma}}$ is often omitted, but we always assume that such an unsorted signature is explicitly given.

The set of function (predicate) symbols of arity n is denoted by $\mathbf{F}_n$ $(\mathbf{P}_n)$. Function symbols of arity 0 are called **constant symbols**, the set of constants is **C**. The **equality symbol** '=' is a distinguished binary predicate, usually written infix. This is the only predicate symbol we assume to have a fixed arity. Note that we do not textually distinguish between the use of '=' as a symbol and its use as a meta-symbol, but '=' will be used only if the meaning is clear from the context.

A **term** is either a variable or a string $f(s_1,\ldots,s_n)$, $n$ = arity(f) , where f is a function symbol and $s_i$, $i = 1,\ldots,n$ are terms. The set of all terms is denoted by **T**. (in the notation of [HO82] the set of terms is denoted by $\mathbf{T}(\mathbf{F}_{\bar{\Sigma}}, \mathbf{V}_{\bar{\Sigma}})$). Terms can also be seen as finite labelled trees as in [Hu76]. We shall use the letters p,q,r,s,t,u,v,w for terms. Let **V**(t) denote the variables occurring in term t, i.e., $\mathbf{V}(t) := \{t\}$, if t is a variable and $\mathbf{V}(t) := \cup\mathbf{V}(t_i\}$, if $t = f(t_1,\ldots,t_n)$. The top-symbol of a term $t = f(t_1,\ldots,t_n)$ is the symbol f , denoted as f = hd(t). The terms $t_i$ are called the **immediate subterms** of t. A term t in which every variable occurs at most once is called **linear**. An **atom** is a string of the form $P(s_1,\ldots,s_n)$, n = arity(P), where $s_i$, $i = 1,\ldots,n$ are terms and P is a predicate symbol. we shall use the symbol **A** for the set of all atoms. A **literal** is a signed atom, i.e., a string of the form +A or -A, where A is an atom. The minus sign has the meaning of logical negation. We use the convention, that if L denotes a literal, then -L denotes the literal with opposite sign, i.e. if L = -A, the -L denotes the literal +A. The set of all literals is denoted by **L**. A **clause** is a finite set of literals, including the empty clause. A clause is interpreted as the disjunction of its literals, where the whole clause is universally quantified over all variables occurring in it. A clause set denoted as CS is a set of clauses. A clause set stands for the conjunction of its clauses. A **Horn clause** is a clause with at most one positive literal (also called a **definite clause**), a **logic program** is a set of Horn clause, where every clause has exactly one positive literal, a fact is a clause with exactly one positive literal and no negative literals and a **query** is a clause without positive literals.

We use the operator **V**(.) also for literals, atoms and clauses and moreover for sets of objects with its obvious meaning. An object t with **V**(t) = Ø is called **ground**. The set of all ground terms is denoted as $\mathbf{T}_{gr}$ and is called the **Herbrand-universe** in the field of Automated Deduction [Lo78, CL73]. The set of all ground atoms is accordingly called the **Herbrand-base**.

In order to select subterms of a given term t (or atom, or literal) we use occurrences [Hu80]. An **occurrence** (or a position) is a word over ℕ. Let Λ denote the empty word. Then we define the set of occurrences D(t) of ta term as follows: i) the empty word Λ is in

D(t) ii) i.$\pi$ is in D(t) iff $t = f(t_1,\ldots,t_n)$ and $\pi$ is in $D(t_i)$. We say two occurrences $\pi$ and $\nu$ are **independent**, if they neither $\pi$ is a prefix of $\nu$ nor $\nu$ a prefix of $\pi$. The **depth** of a term t (or atom, or literal) denoted by depth(t) is defined as the maximal length of an occurrence in D(t). The **size** of a term is the number of symbols in it, or equivalently the number of occurrences in D(t). The set of nonvariable occurrences of a term t is denoted as O(t). The subterm at occurrence $\pi$ is denoted as $t\backslash\pi$ and the term constructed from t by replacing the subterm at occurrence $\pi$ by s is denoted by $t[\pi \leftarrow s]$. The set of subterms of a term t, denoted as subterms(t) is the set $\{s \in T \mid s = t\backslash\pi$ for some $\pi \in D(t)\}$. A set T' of terms is called **subterm-closed**, iff for every $t \in T'$ we have also subterms(t) $\subseteq$ T'.

The set of terms **T** can be turned into an **algebra** [Gr79] by defining for every symbol $f \in$ **F** an operator $f_{\mathbf{T}}$ , such that $f_{\mathbf{T}}(t_1,\ldots,t_n) := f(t_1,\ldots,t_n)$. A **homomorphism** $\varphi{:}\mathbf{T} \rightarrow \mathbf{T}$ is a mapping such that $\varphi(f_{\mathbf{T}}(t_1,\ldots,t_n)) = f_{\mathbf{T}}(\varphi t_1,\ldots,\varphi t_n))$, which is equivalent to $\varphi(f(t_1,\ldots,t_n)) = f(\varphi t_1,\ldots,\varphi t_n))$. A homomorphims $\varphi{:}\mathbf{T} \rightarrow \mathbf{T}$ is also called **endomorphism**. The set **T** is in fact the free term algebra (over **V**) and the set of ground terms $\mathbf{T}_{gr}$ is the initial algebra.

A **substitution** $\sigma$ is an endomorphism $\sigma$: **T** $\rightarrow$ **T**, such that the set $\{x \in \mathbf{V} \mid \sigma(x) \neq x\}$ is finite. The set of all substitutions is denoted as SUB. The empty or identical substitution is denoted by 'Id'. Since every substitution is uniquely determined by its action on variables, it can be represented as a finite set of variable-term pairs $\{x_1 \leftarrow s_1,\ldots,x_m \leftarrow s_m\}$. The single pairs $x_i \leftarrow s$ are called **components** or **bindings**.

Let DOM($\sigma$) denote the set $\{x \mid \sigma(x) \neq x\}$, COD($\sigma$) := $\sigma$DOM($\sigma$) and I($\sigma$) := **V**(COD($\sigma$)). Two substitutions $\sigma,\tau$ are equal, if $\sigma x = \tau x$ for all variables. If $\sigma x = \tau x$ for all variables $x \in$ W, we say $\tau$ and $\sigma$ are equal modulo W and denote this by $\sigma = \tau$ [W].

The effect of applying a substitution $\sigma$ to a term t can also be obtained as the result of simultaneously replacing all variables $x \in$ **V**(t) by the term $\sigma x$. The composition $\sigma \circ \tau$ of two substitutions $\sigma$ and $\tau$ is again a substitution and is usually abbreviated as $\sigma\tau$. The composition can be computed for substitutions with given representations: If $\sigma = \{x_1 \leftarrow s_1,\ldots,x_n \leftarrow s_n\}$ and $\tau = \{y_1 \leftarrow t_1,\ldots,y_m \leftarrow t_m\}$ then $\sigma\tau = \{y_1 \leftarrow \sigma t_1,\ldots,y_m \leftarrow \sigma t_m\} \cup \{x_i \leftarrow s_i \mid x_i \in \mathrm{DOM}(\sigma) - \mathrm{DOM}(\tau)\}$.

A substitution $\sigma$ is called **ground**, iff I($\sigma$) = $\emptyset$. With $\sigma_{|W}$ we denote the restriction of the substitution $\sigma$ to the set of variables W, i.e., $\sigma_{|W}x = \sigma x$ for $x \in W$ and $\sigma_{|W}x = x$ otherwise. For a set of substitutions U and a set of variables W we denote with $U_{|W}$ the set $\{\sigma_{|W} \mid \sigma \in W\}$.

We extend the application of substitutions to atoms by $\sigma(P(s_1,\ldots,s_n)) := P(\sigma s_1,\ldots,\sigma s_n)$ and similarly for literals and clauses.

A substitution $\sigma$ is called **idempotent**, iff $\sigma\sigma = \sigma$. It is a well-known fact that a substitution $\sigma$ is idempotent iff DOM($\sigma$) $\cap$ I($\sigma$) = $\emptyset$ [He83, Ed85]. An idempotent substitution $\sigma = \{x_1 \leftarrow s_1,\ldots,x_n \leftarrow s_n\}$ can be decomposed into its components:

$\sigma = \{x_1 \leftarrow s_1,\ldots,x_n \leftarrow s_n\} = \{x_1 \leftarrow s_1\}\{x_2 \leftarrow s_2\}\ldots\{x_n \leftarrow s_n\}$.

A **renaming** is a substitution $\rho \in$ SUB that is injective on DOM($\rho$) and whose COD($\rho$) consists of variables. If $\rho = \{x_1 \leftarrow y_1,\ldots,x_n \leftarrow y_n\}$ is a renaming, then let $\rho^- := \{y_1 \leftarrow x_1,\ldots,y_n \leftarrow x_n\}$ denote **the converse of** $\rho$. As a technical lemma we have:

**2.1 Lemma.** Let $\rho$ be a renaming. Then:

| | |
|---|---|
| i) $\rho^-$ is a renaming | ii) DOM($\rho$) = COD($\rho^-$) |
| iii) $\rho^-\rho = \rho^-$ | iv) $\rho\,\rho^- = \rho$ |
| v) DOM($\rho^-$) = COD($\rho$) | vi) $(\rho^-)^- = \rho$ |
| vii) $\rho^- \circ \rho = \mathrm{Id}$ [DOM($\rho$)] ■ | |

A more detailed account on substitutions is given in chapter 10.

## 3. Sorted Signatures.

In this paragraph we define sorted signatures, for which we need an additional set of symbols:

- $S_\Sigma$ is the nonempty set of **sort symbols**. Elements are denoted by R,S.

A **term declaration** is a pair (t,S) usually denoted as t:S, where $t \in$ **T**–**V** and S is a sort symbol. If t is of the form $f(x_1,\ldots,x_n)$, where the $x_i$ are different variables, then we say t:S is a **function declaration**, if it is of the form c:S we call it a **constant declaration** and otherwise it is a **proper term declaration.** A **subsort declaration** has the form $R \sqsubseteq S$, where R and S are sorts. A **predicate declaration** is of the form P: $S_1 \times \ldots \times S_n$, where the $S_i$ are sorts.

**3.1 Definition.** A **sorted signature** $\Sigma$ consist of

i) an unsorted signature $\overline{\Sigma}$,

ii) a set $S_\Sigma$ of sorts

iii) a function **S**: $\mathbf{V}_{\overline{\Sigma}} \to$ **S**, such that for every sort $S \in S_\Sigma$, there exist countably infinitely many variables $x \in \mathbf{V}_{\overline{\Sigma}}$ with **S**(x) = S,

iv) a set of term declarations, subsort declarations and predicate declarations. ■

We assume that the equality predicate $=_\Sigma$ is in $\mathbf{P}_\Sigma$ and that for all sorts R,S the predicate declaration $=_\Sigma$: R x S is also in $\Sigma$.

The effect of the function $\mathbf{S}$: $\mathbf{V}_\Sigma \to \mathbf{S}$ is to partition the set of variables $\mathbf{V}$ into the sets $\mathbf{V}_S$ of variables of sort S. We abbreviate function declarations $f(x_1,\ldots,x_n)$: S as $f{:}S_1 \times\ldots\times S_n \to S$, where $S_i$ is the sort of the variable $x_i$.

Generally, it is sufficient for the presentation of a signature to write down only the term declarations, subsort declarations and predicate declarations together with some information concerning the sort of the variables occurring in term declarations.

For algebraic specifications term-declarations appear in [Gg78,Wad82, Go83].

We use $\mathbf{F}_\Sigma$ for the set of function symbols, $\mathbf{P}_\Sigma$ for the set of predicate symbols and $\mathbf{V}_\Sigma$ for the set of variable symbols in $\Sigma$. The set of term-declarations and of subsort-declarations is denoted as $\mathbf{TD}_\Sigma$ and $\mathbf{SD}_\Sigma$, respectively.

Let $\sqsubseteq_\Sigma$ be the quasi-ordering on $\mathbf{S}_\Sigma$ defined by the reflexive and transitive closure of the subsort declarations. We say the sorts R,S are **equivalent**, iff $R \sqsubseteq_\Sigma S$ and $R \sqsupseteq_\Sigma S$. To emphasize that some objects belong to the signature $\Sigma$, we prefix it by $\Sigma$–. The symbols $\sqsubset_\Sigma$, $\sqsubseteq_\Sigma$, $\sqsupset_\Sigma$, $\sqsupseteq_\Sigma$ are used with their usual meaning.

In the description of a signature or in examples we indicate that a variable x or a constant c has sort S by x:S, c:S or $x_S$, $c_S$.

We say a signature is **finite**, if its description is finite, i.e., the set of sorts, function symbols, predicate symbols, subsort declarations and term declarations is finite.

The definition of well-sorted terms and substitutions requires some preparation and we will carry out these definitions and corresponding lemmas in the next paragraph.

**Remarks.**

i) If all term declarations are function declarations, then the signature as defined in this paper corresponds to the standard definition described in the literature, (cf. [Ob62, HO80, CD83, Co83, Wa84, Sch85a, Sm86].

ii) If all sorts are equivalent, then a sorted signature is equivalent to an unsorted one.

A signature is **one-sorted** , iff it has just one sort. A signature is **many-sorted**, if it has more than one sort symbol and there are no subsort declarations. A signature is **order-sorted**, if it has more than one sort symbol and there are subsort declarations. A signature is **linear**, iff all terms in term declarations are linear. A signature, where all term-declarations are function-declarations is called an **elementary** signature. We call a signature **simple**, iff it is elementary and has exactly one function declaration for every function symbol.

With this terminology, the signatures in [HO80, Wa84] are simple ones, whereas the signatures in [Ob62, CD83, Co83, Sch85a, Sm86] are nonsimple, but elementary.

**Remark.** The notions of a one-sorted signature and an unsorted signatures are not equivalent, since in the unsorted case all terms are well-sorted, whereas in the one-sorted case there may be a difference between ill- and well-sorted terms.

In the following we use the bar $\bar{}$ also as an operator that assigns to every signature $\Sigma$ its unsorted subsignature, that is given a sorted signature $\Sigma$, we use the signature $\bar{\Sigma}$ to provide us with all unsorted objects of $\Sigma$. For example we denote with $\mathbf{T}_{\bar{\Sigma}}$ the set of all unsorted terms of $\Sigma$. The set of ill-sorted terms is the difference $\mathbf{T}_{\Sigma} - \mathbf{T}_{\bar{\Sigma}}$. The set of all substitutions $\sigma: \mathbf{T}_{\bar{\Sigma}} \to \mathbf{T}_{\bar{\Sigma}}$ is denoted as $SUB_{\bar{\Sigma}}$.

## 4. Well-sorted Terms and Substitutions.

First we shall give some introductory examples in order to provide the reader with some intuition on term declarations.

**4.1 Example.** We give a specification of the even numbers as a subset of the natural numbers:

EVEN ⊏ NAT ; 0:EVEN ; s: NAT → NAT

$s(s(x_{EVEN}))$: EVEN. ■

This definition works as a specification of even numbers, if we have only the set of ground terms. In the corresponding term algebra (with term declarations) we consider terms of the form s(s(t)) to be of sort EVEN if t is a term of sort EVEN. Hence the set of ground terms of sort EVEN is exactly $\{0, s(s(0)), s(s(s(s(0)))), \ldots, s^{2i}(0), \ldots\}$. In order to obtain the same semantical result with usual many-sorted specifications there are two options: 1) The sort ODD has to be specified and the corresponding function declarations s:ODD → EVEN and s:EVEN → ODD must be in the signature. 2) A new function times2: NAT → EVEN has to be added together with the equations times2(0) = 0 and times2(s(x)) = s(s(times2(x)).

As a further (more complex) example we specify addition on natural numbers with the sorts EVEN and NAT.

**4.2 Example.**

| | |
|---|---|
| EVEN ⊏ NAT; | 0:EVEN |
| s: NAT → NAT ; | +: NAT x NAT → NAT , EVEN x EVEN → EVEN |

$s(s(x_{EVEN}))$: EVEN.
$y_{NAT} + y_{NAT}$: EVEN;
$y_{NAT} + 0 = y_{NAT}$; $y_{NAT} + s(z_{NAT}) = s(y_{NAT} + z_{NAT})$. ■

This example is an (initial algebra) specification of the addition of natural numbers together with even numbers. However, the initial algebra and the full term algebra (modulo equations) do not exhibit the same behaviour: the term $(x + y) + (y + x)$ is of sort NAT in the term algebra, but represents always an even number in the initial algebra (the set of ground terms).

The definition of the sort of a term is straightforward. The problem is that it is not possible to define well-sorted terms without well-sorted substitutions. As we shall see later in paragraph 5, the following construction is appropriate for free and initial term algebras.

The set of Σ-terms of sort S, $\mathbf{T}_{\Sigma,S}$ is defined as follows:

**4.3 Definition:** $\mathbf{T}_{\Sigma,S}$ is (recursively) constructed by the following three rules:

i) $x \in \mathbf{T}_{\Sigma,S}$ If $S(x) \sqsubseteq S$.
ii) $t \in \mathbf{T}_{\Sigma,S}$ If $t{:}R \in \Sigma$ and $R \sqsubseteq S$.
iii) $\{x \leftarrow r\}t \in \mathbf{T}_{\Sigma,S}$ If $t \in \mathbf{T}_{\Sigma,S}$, $r \in \mathbf{T}_{\Sigma,R}$ and $x \in \mathbf{V}_\Sigma$ such that $R \sqsubseteq S(x)$. ■

That is, we start with the information given in the signature and the sort of the terms. A new term t' of sort S is constructed from a term t of sort S by replacing one of t's variables simultaneously by a term of sort less than or equal to the sort of this variable.

In order to illustrate this definition we consider Example 4.2 (without equations). Rule ii) implies that $y_{NAT} + y_{NAT} \in \mathbf{T}_{\Sigma,EVEN}$, hence by rule iii) $s(0) + s(0) \in \mathbf{T}_{\Sigma,EVEN}$, but the term $s(0) + 0$ is not a member of $\mathbf{T}_{\Sigma,EVEN}$, as expected. However, all three terms are in $\mathbf{T}_{\Sigma,NAT}$.

A first trivial consequence of the above construction is :

**4.4 Lemma.**

i) For all sorts $R,S \in \mathbf{S}_\Sigma$: $R \sqsubset S$ implies $\mathbf{T}_{\Sigma,R} \subseteq \mathbf{T}_{\Sigma,S}$.
ii) For variables we have: $x \in \mathbf{T}_{\Sigma,S} \Leftrightarrow S(x) \sqsubseteq S$. ■

We define $\mathbf{T}_\Sigma$, the set of all Σ-terms (or well-sorted terms) as the union $\cup\{\mathbf{T}_{\Sigma,S} \mid S \in \mathbf{S}_\Sigma\}$. We denote the set of well-sorted ground terms of sort S by $\mathbf{T}_{\Sigma,S,gr}$ and the set of all well-sorted ground terms by $\mathbf{T}_{\Sigma,gr}$. The sort of a term t is defined as the set $\mathbf{S}_\Sigma(t) := \{S \in \mathbf{S}_\Sigma \mid t \in \mathbf{T}_{\Sigma,S}\}$. For Σ-variables x we have $\mathbf{S}_\Sigma(x) = \{S \in \mathbf{S}_\Sigma \mid S(x) \sqsubseteq S\}$. Obviously for every term $t \in \mathbf{T}_\Sigma$ the set $\mathbf{S}_\Sigma(t)$ is a nonempty upper segment in $\mathbf{S}_\Sigma$ and for every variable x

$\in V_\Sigma$ the set $S_\Sigma(x)$ has a unique least element, namely $S(x)$. We show in paragraph 5 that the function $S_\Sigma(t)$ is computable for finite signatures. The set of well-sorted atoms $A_\Sigma$ is defined as the set of expressions $P(t_1,\ldots,t_n)$, such that there exists a predicate declaration $P:S_1\times\ldots\times S_n$ and $t_i \in T_{\Sigma,Si}$. The set of ground $\Sigma$-atoms is denoted by $A_{\Sigma,gr}$.

We say a term declaration t:S is **redundant**, iff t is of sort S with respect to the remaining term declarations. If we add in example 4.1 the term declaration s(s(0)): EVEN to the signature, then the sort of terms is not changed. In the new signature, this is a redundant term-declaration, since s(s(0)) is of sort EVEN with respect to the old signature. In general we assume that there are no redundant term-declarations. We may change the definition of an elementary signature to be a signature, where all proper term declarations are redundant.

We say $\Sigma$ (or $T_\Sigma$ ) is **subterm-closed**, iff each subterm $s_i$ of every well-sorted term $f(s_1,\ldots,s_n)$ is also a well-sorted term.

**4.5 Example.** An example for a non subterm-closed signature is $\Sigma := \{f(a):S\}$. We have $F_\Sigma = \{f,a\}$, $S_\Sigma = \{S\}$ and $T_\Sigma = \{f(a)\} \cup V_S$. This means $a \notin T_\Sigma$, hence $\Sigma$ is not subterm-closed. ■

A computationally desirable property of a signature is regularity: A signature is **regular**, iff $\sqsubseteq_\Sigma$ is a partial ordering and for every term t the set $S_\Sigma(t)$ has a unique least sort. The same notion is called preregular in [GM85a]. In [Sm86] regular is used with the same meaning and a characterization is given for elementary signatures.

In this case of regular signatures we denote this unique sort by $LS_\Sigma(t)$ and call it the (unique) 'sort of a term'. Note that the relation $S_\Sigma(t) \subseteq S_\Sigma(s)$ is equivalent to $LS_\Sigma(t) \sqsupseteq_\Sigma LS_\Sigma(s)$. By abuse of notation we sometimes write $LS_\Sigma(x)$ for the sort of a variable x, even if the signature is not regular; obviously $LS_\Sigma(x) = S(x)$. If $\sqsubseteq_\Sigma$ is a well-founded, linear partial ordering, then the signature is trivially regular. Furthermore simple signatures are always regular.

We call a signature **polymorphic**, if it is regular, elementary and the signature has a top-sort TOP.

**4.6 Definition.** The set of **well-sorted substitutions** (or $\Sigma$-substitutions) $SUB_\Sigma$ is defined as follows:

$SUB_\Sigma := \{\sigma \in SUB_{\bar{\Sigma}} \mid S_\Sigma(\sigma x) \supseteq S_\Sigma(x)\}$. ■

The intuitive meaning is that substitutions weaken the sorts, i.e. $\sigma x$ has a smaller or equal sort than x for all variables x. For regular signatures we can reformulate this definition as $SUB_\Sigma := \{\sigma \in SUB_{\bar\Sigma} \mid LS_\Sigma(\sigma x) \sqsubseteq LS_\Sigma(x)\}$. Obviously we have $SUB_\Sigma \subseteq SUB_{\bar\Sigma}$ and the identity substitution $Id_\Sigma$ is in $SUB_\Sigma$.

An immediate consequence of the definition of well-sorted substitutions is that the set of codomain terms is well-sorted.

The (injective) operator 'bar' $\bar{\ }$ (considered as a mapping on terms and substitutions) behaves like a forgetful functor, i.e. $\bar{\ } : \Sigma \to \bar\Sigma$, $\bar{\ } : \mathbf{T}_\Sigma \to \mathbf{T}_{\bar\Sigma}$ and $\bar{\ } : SUB_\Sigma \to SUB_{\bar\Sigma}$ is injective. We have $\overline{\sigma\tau} = \bar\sigma \circ \bar\tau$ for two well-sorted substitutions $\sigma,\tau$ and $\overline{\sigma t} = \bar\sigma(\bar t)$ for a well-sorted substitution $\sigma$ and a well-sorted term t. This observation is helpful and justifies the lifting of various lemmas from the unsorted to the sorted case.

A **$\Sigma$-renaming** is a sort-preserving renaming $\rho \in SUB_\Sigma$, i.e., it satisfies in addition $S_\Sigma(\rho x) = S_\Sigma(x)$.

Well-sorted substitutions are compatible with the sort-structure on $\mathbf{T}_\Sigma$, i.e., well-sorted substitutions map $\mathbf{T}_{\Sigma,S}$ into $\mathbf{T}_{\Sigma,S}$ for all sorts S:

**4.7 Proposition:** For all well-sorted terms $t \in \mathbf{T}_{\Sigma,S}$ and all well-sorted sustitutions $\sigma \in SUB_\Sigma$ we have $\sigma t \in \mathbf{T}_{\Sigma,S}$.

**Proof:**

Let $t = f(t_1,\dots,t_n) \in \mathbf{T}_{\Sigma,S}$ and let $\sigma = \{x_1 \leftarrow s_1,\dots,x_m \leftarrow s_m\} \in SUB_\Sigma$. If $\sigma$ is an idempotent $\Sigma$-substitution, then $\sigma = \{x_1 \leftarrow s_1\} \circ \dots \circ \{x_m \leftarrow s_m\}$. Repeated application of Definition 4.3 implies $\sigma t \in \mathbf{T}_{\Sigma,S}$.

To prove the general case let $\rho \in SUB_\Sigma$ be an idempotent $\Sigma$-renaming with $DOM(\rho) = I(\sigma)$ such that $COD(\rho)$ consists of variables not occurring in t or some $s_i$. Let $\rho^-$ denote the converse of $\rho$. Then $\sigma t = \rho^-\rho\sigma t$ by Lemma 2.1. The substitution $\rho\sigma$ is idempotent, hence we can decompose the substitution $\rho\sigma$ into a product as follows: $\{x_1 \leftarrow \rho s_1,\dots,x_m \leftarrow \rho s_m\} = \{x_1 \leftarrow \rho s_1\} \circ \dots \circ \{x_m \leftarrow \rho s_m\}$. Every component $\{x_m \leftarrow \rho s_m\}$ is in $SUB_\Sigma$, since $s_m \in \mathbf{T}_{\Sigma,R}$ implies $\rho s_m \in \mathbf{T}_{\Sigma,R}$ for every sort R. Hence $\rho\sigma t \in \mathbf{T}_{\Sigma,S}$ by repeated application of Definition 4.3 iii). Now we conclude from $\sigma t = \rho^-\rho\sigma t$ that $\sigma t \in \mathbf{T}_{\Sigma,S}$. ■

**4.8 Corollary**. The composition $\sigma\tau$ of two well-sorted substitutions is again well-sorted, i.e., $SUB_\Sigma$ is a monoid.

**Proof**. Let $\tau = \{x_1 \leftarrow s_1,\dots,x_m \leftarrow s_m\} \in SUB_\Sigma$ and let $\sigma \in SUB_\Sigma$. Consider the composition $\sigma\tau$. In order to show that the composition $\sigma\tau$ is well-sorted we have to show

$\sigma\tau x \in \mathbf{T}_{\Sigma,S(x)}$ for all $\Sigma$-variables x. By Proposition 4.7 we have $\sigma\tau x_i = \sigma s_i \in \mathbf{T}_{\Sigma,S(x)}$ for all $x_i \in DOM(\sigma)$. For all other variables x we have either $\sigma\tau x = x$ or $\sigma\tau x = \sigma x$. Hence $\sigma\tau$ is well-sorted. ■

The next proposition shows that the set of all well-sorted terms can be generated by applying well-sorted substitutions to terms in term-declarations.

**4.9 Proposition.** For every sort $S \in \mathbf{S}_\Sigma$ and every nonvariable term $s \in \mathbf{T}_{\Sigma,S}$ there exists a term declaration $t{:}R \in \Sigma$ with $R \sqsubseteq S$ and a substitution $\sigma \in SUB_\Sigma$ such that $\sigma t = s$.
**Proof.** Follows by structural induction using Definition 4.3 and Corollary 4.8. ■

In elementary signatures the replacement of subterms behaves similar to the application of substitutions:

**4.10 Lemma.** Let $\Sigma$ be an elementary signature. Then

i) $\mathbf{S}_\Sigma(f(t_1,\ldots,t_n))$ depends only on f and the sorts of subterms $t_i$.

ii) For all $s,t \in \mathbf{T}_\Sigma$ and all $\pi \in D(t)$:

$$\mathbf{S}_\Sigma(s) \subseteq \mathbf{S}_\Sigma(t\backslash\pi) \Rightarrow t[\pi \leftarrow s] \in \mathbf{T}_\Sigma \text{ and } \mathbf{S}_\Sigma(t[\pi \leftarrow s]) \subseteq \mathbf{S}_\Sigma(t).$$

**Proof.** Shown by induction. ■

We show in part II. Example 6.14 that the condtion 4.10 i) is in general not sufficient to characterize elementary signatures.

Most signatures considered in the Automated Deduction literature are regular and elementary (cf. [GM85, Wa83, CD85, Co83, Sch85a]. Algebraic specifications based on a many-sorted signature always use elementary signatures. The signatures in [Wad82, Sch85b, Go83, Gg78, GM85a] are not elementary. An example for a non elementary signature is example 4.1 above.

In our terminology A. Oberschelp [Ob62] investigated elementary signatures, which may be regular or not. G. Huet and D.C. Oppen [HO80] have as basis many-sorted and simple signatures. A. Cohn [Co83] considers elementary signatures, but has no equations and a different definition of clauses and deductions. Our approach is in fact an extension of Ch. Walther´s [Wa83], who considered simple signatures.

We always assume the following conditions:

**4.11 Assumptions:**

i) Signatures are subterm-closed.

ii) For every $S \in \mathbf{S}_\Sigma$, there exists a ground term $t_{gr} \in \mathbf{T}_{\Sigma,gr}$ with $S \in \mathbf{S}_\Sigma(t)$. ■

The first assumption appears to be natural, since it does not make sense to allow ill-formed subexpressions in well-formed expressions. Furthermore non subterm-closed signatures would cause technical problems, for example structural induction would not be possible.

The second assumption is necessary to ensure that sorts are not empty. It is possible to make deductions with empty sorts, for example J.A. Goguen and J. Meseguer [GM81] permit empty sorts. Their idea is to do as if sorts are nonempty and to collect all these nonemptyness assumptions during a deduction. The deduced sentences are then of the form: If the sorts $S_1,\ldots,S_n$ are not empty, then F holds.

However, from our point of view, this is more a problem at a meta-level and should not be confused with the pure sorted calculus. For example the proof of the nonemptyness of sorts could be carried out in a preprocessing step and afterwards the deduction system would have a solid basis.

Automated deduction systems based on resolution usually work with the tacit assumption that sorts are nonempty, since otherwise the combination of resolution and order-sorted unification becomes unsound. For example if S is an empty sort, the two statements P(x:S) and $\neg$P(x:S) are not unsatisfiable.

Note that Assumptions 4.11 ii) implies that every well-sorted term t and every well-sorted atom A has a well-sorted ground instance, i.e. there exists a $\Sigma$-substitution $\sigma$, such that $\sigma t$ is a ground term ($\sigma A$ is a ground atom). For every $\Sigma$–algebra A the assumption 4.11 ii) implies that for every sort $S \in S_\Sigma$, the set $S_A$ is not empty (Corollary 6.5).

Further assumptions like finiteness of the set of sort $S_\Sigma$ are made explicit when they are needed. For finite signatures the above assumptions are decidable properties: ( (i) is proved in Proposition 4.9).

**4.12 Lemma.** For a finite signature it is decidable if 4.11 ii) is satisfied.

**Proof.** We can compute the nonempty sorts by a simple fixed-point iteration (using Definition 4.3). ■

## 5. Order-sorted Matching

**5.1 Definition.** Let $s,t \in T_\Sigma$. Then

i) $s \geq_\Sigma t$ iff there exists $\sigma \in SUB_\Sigma$ such that $s = \sigma t$.

In this case we call $\sigma$ an **instantiating substitution of t to s** and s a **$\Sigma$-instance** of t.

ii) $s \equiv_\Sigma t$ iff $s \leq_\Sigma t$ and $s \geq_\Sigma t$. ■

Sometimes $\sigma$ is called a matcher of t to s, however, this should be reserved for the case $DOM(\sigma) \cap \mathbf{V}(s) = \emptyset$.

The relation $\leq_\Sigma$ is a quasi-ordering on well-sorted terms and $\equiv_\Sigma$ is an equivalence relation.

We extend the instance relation of terms to well-sorted substitutions.

**5.2 Definition.** Let $W \subseteq V$ and $\sigma,\tau \in SUB_\Sigma$.

i) $\sigma = \tau \ [W]$ , iff $\sigma x = \tau x$ for all $x \in W$.

ii) $\sigma \geq_\Sigma \tau \ [W]$ , iff there exists a $\lambda \in SUB_\Sigma$ with $\sigma = \lambda\tau \ [W]$.
In this case we call $\lambda$ an **instantiating substitution** of $\tau$ to $\sigma$ and $\sigma$ a $\Sigma$-instance of $\tau$ modulo W.

iii) $\sigma \equiv_\Sigma \tau \ [W]$ , iff $\sigma \leq_\Sigma t \ [W]$ and $\sigma \geq_\Sigma \tau \ [W]$.

Obviously the relation $\leq_\Sigma[W]$ is a quasi-ordering on well-sorted substitutions and the relation $\equiv_\Sigma[W]$ is an equivalence relation. If W is the set of all $\Sigma$-variables, then we write $\leq_\Sigma$ instead of $\leq_\Sigma[V_\Sigma]$.

The computation of instantiating substitutions for unsorted first order terms (often called the Robinson case) is well-known [Ro65, Hu76, FH83]. In particular the following holds: If there exists a substitution $\sigma$ with $s = \sigma t$, then there exists a unique (effectively computable) substitution $\tau$, such that $s = \tau t$ and $DOM(\tau) = \mathbf{V}(t)$. We have $\tau = \sigma \ [\mathbf{V}(t)]$. If t is not a variable, then $depth(s) > depth(\tau x)$ for every variable $x \in \mathbf{V}(t)$. The same holds for the instance problem of substitutions.

The proof of the following lemma gives a recursive algorithm for the computation of the sort of a term.

**5.3 Proposition.** For a finite signature $\Sigma$ the sort $S_\Sigma(t)$ is effectively computable for all terms t.

**Proof.** The proof is by induction on the term depth:

Let $s \in \mathbf{T}_\Sigma$.

As a basis for induction we have to compute the sort of s if $depth(s) = 0$. But this is a trivial computation: either s is a variable or s is a constant and then we have to examine at most finitiely many term declarations.

If $depth(s) > 0$, then for every declaration $t_i:S_i$ we can compute the (unsorted) Robinson matcher $\sigma_i$ with $\sigma_i t_i = s$. For every term $r \in COD(\sigma_i)$ we have $depth(r) < depth(s)$, since variables are forbidden as terms in term declarations. To check the well-sortedness of $\sigma_i$

requires to compute the sort of all terms in the codomain of $\sigma_i$. The condition to check is: $\sigma_i x \in \mathbf{T}_{\Sigma,S(x)}$ for every $x \in \mathbf{V}(s)$. This is decidable by induction hypothesis. ■

**5.4 Corollary.** Let $\Sigma$ be a finite signature. Then

i) For two well-sorted terms s,t, $s \leq_\Sigma t$ is decidable. Furthermore an instantiating substitution $\mu$ with $\mu s = t$ and $\mathrm{DOM}(\mu) \subseteq \mathbf{V}(s)$ is unique, if it exists.

ii) For two well-sorted substitutions $\sigma,\tau$ it is decidable whether $\sigma \leq_\Sigma \tau$ [W] (for a set of variables W). Furthermore an instantiating substitution $\mu$ with $\mathrm{DOM}(\mu) \subseteq \mathbf{V}(\sigma W)$ and $\mu\sigma = \tau$ [W] is unique, if it exists. ■

A consequence of Proposition 5.3 is that for finite signatures the subterm-closedness is decidable:

**5.5 Lemma.** For a finite signature $\Sigma$, it is decidable if it is subterm-closed.

**Proof.** Assume there is a well-sorted term $s = f(s_1,\ldots,s_n)$ with an ill-sorted subterm $s_j$. Proposition 4.9 yields that there exist a declaration t:S and a substitution $\sigma$, such that $\sigma t = s$. This means that t has an ill-sorted subterm. Hence the procedure for testing subterm-closedness may work as follows: Compute the sort of all subterms of terms in declarations. If all subterms are well-sorted, then $\Sigma$ is subterm-closed, otherwise it is not subterm-closed. This check is finite, since the signature is finite and the sort of a term is computable in finite signatures by Proposition 5.3. ■

**5.6 Corollary.** In finite signatures it is decidable, whether a term-declaration t:S is redundant. ■

**5.7 Proposition.** In finite, elementary signatures it is decidable whether a set of sorts is the sort $\mathbf{S}_\Sigma(t)$ of some term t.

**Proof.** A fixed-point iteration using Proposition 4.9 and Lemma 4.10 gives a terminating algorithm to determine all sets possible as the sort of a term. ■

**5.8 Corollary.** For finite, elementary signatures it is decidable, whether they are regular. ■

In III.6.5 ff. we give a method to check regularity of signatures.

For every function symbol f we collect the term declarations with terms starting with f and choose the maximal ones with respect to $\leq_\Sigma$. We define **mgterms**(f,S) (most general terms) to be a set of representatives of $\leq_\Sigma$-minimal terms in $\{t \mid t{:}S' \in \Sigma \text{ with } S' \sqsubseteq S\}$. By Corollary 5.4 this set is effectively computable. The terms in mgterms(f, S) are said to be **basic**, iff they are of the form $f(x_1,\ldots,x_n)$ where all $x_i$ are different variables.

Let us have a look at the time complexity of sort-computation. In the trivial case of simple signatures, we can compute the sort of a term by inspecting its toplevel function symbol.

In signatures with multiple function declarations, a recursive algorithm which does not store the result of computing may behave exponentially:

Consider the term $( s_1 * ( s_2 * (\ldots * s_n) \ldots)$ and assume there are two function declarations for $*$. Then the sort-computation of $s_1$ is performed 2 times, the sort computation of $s_2$ is performed 4 times and the sort computation of $s_n$ is performed $2^n$ times.
However, if the results of computing sorts is stored then sort-computation is quasi-linear, i.e. of time complexitiy less than $O(n^{1+\varepsilon})$ for all $\varepsilon > 0$:

5.9 **Proposition.** Sort-computation in finite signatures has quasi-linear time complexity.
**Proof.** Let t be a $\Sigma$-term. We can assume that we proceed by first computing the sort of subterms of t, i.e., we first compute the sort of subterms of depth 1, then the sort of subterms of depth 2 and so on. Obviously the number of subterms of t is linear in the size of t. Since the signature is finite, all operations connected with the sorts and term declarations require constant time, for example subsort-checking or matching a term-declaration against an arbitrary term. Due to Proposition 4.9 the operation to be performed is matching a term declaration and subsort checking. Hence sort-computation is quasi-linear. ■

## 6. Algebras and Homomorphisms.

As a prerequisite for the definition of a $\Sigma$-algebra we introduce the more general technical notion of $\Sigma$-quasi-algebras, which is an extension of the notion of partial algebras [Gr79, BR87] by denotations for sort symbols:

Let $\Sigma$ be a signature. A **$\Sigma$-quasi-algebra** $\mathcal{A}$ consists of a carrier set A, a partial function $f_{\mathcal{A}}: A^{\text{arity}(f)} \to A$ (with domain $\mathcal{D}(f_{\mathcal{A}})$ ) for every function symbol f in $\Sigma$, a set $S_{\mathcal{A}} \subseteq A$ for every sort S, such that the carrier A is the union of denotations for sort symbols in $\Sigma$ , i.e., $A = \bigcup\{S_{\mathcal{A}} \mid S \in S_\Sigma\}$.

Let $\mathcal{A}$ be a $\Sigma$-quasi-algebra. We say a partial mapping $\varphi: \mathbf{V}_\Sigma \to A$ is a **partial $\Sigma$-assignment**, iff $\varphi(x) \in S(x)_{\mathcal{A}}$ for every $\Sigma$-variable $x \in \mathcal{D}(\varphi)$. If $\varphi$ is a total funtion, we call it a **Σ-assignment**. The **homomorphic extension** $\varphi_h$ of a (partial) Σ-assignment $\varphi: \mathbf{V}_\Sigma \to A$ on $\mathbf{T}_\Sigma$ is defined as a (partial) function $\varphi_h: \mathbf{T}_\Sigma \to A$ as follows:

i) $\varphi_h(x) := \varphi(x)$ for all Σ-variables $x \in \mathcal{D}(\varphi)$ and

ii) for every $f(s_1,\ldots,s_n) \in \mathbf{T}_\Sigma$ :

if $s_i \in \mathcal{D}(\varphi_h)$ for $i = 1,\ldots,n$ and $(\varphi_h s_1,\ldots,\varphi_h s_n) \in \mathcal{D}(f_\mathcal{A})$

then $f(s_1,\ldots,s_n) \in \mathcal{D}(\varphi_h)$ and $\varphi_h(f(s_1,\ldots,s_n)) := f_\mathcal{A}(\varphi_h s_1,\ldots,\varphi_h s_n)$.

This definition makes sense, since we assume that signatures are subterm-closed.
The reason for introducing partial Σ-assignments is that sorts may be empty in Σ-algebras and if one denotation for a sort is empty in a Σ-algebra $\mathcal{A}$, then there exists no total Σ-assignment. However, as it will turn out below, Assumption 4.11 implies that in Σ-algebras denotations for sorts are always nonempty.

**6.1 Definition.** Let Σ be a signature. Then a **Σ-algebra** $\mathcal{A}$ is defined as a Σ-quasi-algebra $\mathcal{A}$ that satisfies the following additional conditions:

i) If $R \sqsubset S$ is in Σ, then $R_\mathcal{A} \subseteq S_\mathcal{A}$

ii) For all term-declarations $t{:}S \in \Sigma$ and for every partial Σ-assignment $\varphi: \mathbf{V}_\Sigma \to A$ with $V(t) \subseteq \mathcal{D}(\varphi)$: $t \in \mathcal{D}(\varphi_h)$ and $\varphi_h(t) \in S_\mathcal{A}$. ■

Note that the second condition has strong implications for the domain of functions $f_\mathcal{A}$ on $\mathcal{A}$.
In the following we do not distinguish between an algebra $\mathcal{A}$ and its carrier A and we denote both with A.

**6.2 Definition.** Let Σ be a signature and let A and B be Σ-algebras. A **Σ-homomorphism** is a mapping $\varphi: A \to B$ such that:

i) $\varphi(S_A) \subseteq S_B$ for all $S \in \mathbf{S}_\Sigma$.

ii) $\varphi(\mathcal{D}(f_A)) \subseteq \mathcal{D}(f_B)$ for all $f \in \mathbf{F}_\Sigma$.

iii) If $(a_1,\ldots,a_n) \in \mathcal{D}(f_A)$ then $\varphi(f_A(a_1,\ldots,a_n)) = f_B(\varphi a_1,\ldots,\varphi a_n)$.

Obviously, the composition of two Σ-homomorphisms is again a Σ-homomorphism. A Σ-homomorphism $\varphi: A \to A$ is called a **Σ-endomorphism**. A bijective Σ-homomorphism $\varphi: A \to B$ is called a **Σ-isomorphism**, if the inverse mapping is again a Σ-homomorphism. In this case we say A and B are **isomorphic** as Σ-algebras.
Note that for every Σ-algebra the identity $Id_A$ is a Σ-endomorphism of A.

We also need the notion of a **partial Σ-homomorphism**. This is defined as a partial mapping $\varphi: A \to B$, such that Definition 6.2 i) and ii) are satisfied on $\mathcal{D}(\varphi)$ and instead of iii) we have :

iii)' If $(a_1,\ldots,a_n) \in \mathcal{D}(f_A)$ and $a_i \in \mathcal{D}(\varphi)$ then $f_A(a_1,\ldots,a_n) \in \mathcal{D}(\varphi)$ and $\varphi(f_A(a_1,\ldots,a_n) = f_B(\varphi a_1,\ldots,\varphi a_n)$.

The term algebra of well-sorted terms is a $\Sigma$-algebra with carrier set $\mathbf{T}_\Sigma$ if we define:

i) $S_{T\Sigma} := \mathbf{T}_{\Sigma,S}$ for every sort $S \in \mathbf{S}_\Sigma$.

ii) $\mathcal{D}(f_{T\Sigma}) := \{(s_1,\ldots,s_n) \mid f(s_1,\ldots,s_n) \in \mathbf{T}_\Sigma\}$.

iii) $f_{T\Sigma}(s_1,\ldots,s_n) := f(s_1,\ldots,s_n)$.

Since we have assumed that $\Sigma$ is subterm-closed, this is a $\Sigma$-algebra,by Proposition 4.7 and by Lemma 4.4. The set of $\Sigma$-endomorphism of $\mathbf{T}_\Sigma$ that move only finitely many variables is exactly the set of well-sorted substitutions, i.e., $SUB_\Sigma = \{\varphi: \mathbf{T}_\Sigma \to \mathbf{T}_\Sigma \mid \varphi$ is a $\Sigma$-homomorphism and $DOM(\varphi)$ is finite$\}$. Note that the set $\mathbf{T}_{\Sigma,gr}$ is also a $\Sigma$-algebra according to these definitions.

Now we show that the $\Sigma$-algebra $\mathbf{T}_\Sigma$ is the free algebra of type $\Sigma$ and that the ground term algebra $\mathbf{T}_{\Sigma,gr}$ is the initial algebra of type $\Sigma$:

**6.3 Proposition.** Let A be a $\Sigma$-algebra. Then the homomorphic extension $\varphi_h$ of every partial $\Sigma$-assignment $\varphi: \mathbf{V}_\Sigma \to A$ is a partial $\Sigma$-homomorphism with domain $\mathcal{D}(\varphi_h) = \{t \in \mathbf{T}_\Sigma \mid V(t) \subseteq \mathcal{D}(\varphi)\}$. Furthermore $\varphi_h$ is a $\Sigma$-homomorphism for a total $\Sigma$-assignment $\varphi$.

**Proof.** We show by structural induction according to Definition 4.3. that $\varphi_h$ is a partial $\Sigma$-homomorphism. Definition 6.1 serves as an induction basis for our proof.

First we show that Definition 6.2 i) holds for $\varphi_h$:

Let $t \in \mathbf{T}_{\Sigma,S}$ , $r \in \mathbf{T}_{\Sigma,R}$ and let x be a variable in t such that $R \sqsubseteq S(x)$. By Definition 4.3 iii) we have $\{x \leftarrow r\}t \in \mathbf{T}_{\Sigma,S}$. Let $\varphi : \mathbf{V}_\Sigma \to A$ be a $\Sigma$-assignment.

We have to show that $\mathbf{V}(\{x \leftarrow r\}t) \subseteq \mathcal{D}(\varphi)$ implies $\{x \leftarrow r\}t \in \mathcal{D}(\varphi_h)$ and $\varphi_h\{x \leftarrow r\}t \in S_A$.

If $V(r) \nsubseteq \mathcal{D}(\varphi)$, there is nothing to show, since then $\mathbf{V}(\{x \leftarrow r\}t) \nsubseteq \mathcal{D}(\varphi)$.

Hence we can assume that $\mathbf{V}(r) \subseteq \mathcal{D}(\varphi)$ and $\mathbf{V}(t)-\{x\} \subseteq \mathcal{D}(\varphi)$.

By induction hypothesis we have $r \in \mathcal{D}(\varphi_h)$ and $\varphi_h r \in R_A$, hence we can define the $\Sigma$-assignment $\psi : \mathbf{V}_\Sigma \to A$ as follows: $\psi y := \varphi y$ for all variables $y \in \mathbf{V}(t)-\{x\}$ and $\psi x := \varphi_h r$. This is a $\Sigma$-assignment, since $R \sqsubseteq S(x)$. Again by induction hypothesis we have $\psi_h t \in S_A$, since $\mathcal{D}(\psi) = \mathbf{V}(t)$. Now $\psi_h t = \varphi_h\{x \leftarrow r\}t$ implies $\{x \leftarrow r\}t \in \mathcal{D}(\varphi_h)$ and $\varphi_h\{x \leftarrow r\}t \in S_A$. ❑

Parts ii) and iii)' of the definition of a partial $\Sigma$-homomorphism are easy to see:
(ii) is equivalent to the claim that for $f(s_1,\ldots,s_n) \in \mathcal{D}(\varphi_h)$ the function $f_B$ is defined for the arguments $(\varphi_h s_1,\ldots, \varphi_h s_n)$.
(iii)' follows from the above and from the definition of homomorphic extensions. ■

**6.4 Corollary.** The $\Sigma$-algebra $\mathbf{T}_{\Sigma,gr}$ is the initial algebra of type $\Sigma$:
**Proof.** Application of Proposition 6.3 to the empty $\Sigma$-assignment yields for every $\Sigma$-algebra A a unique $\Sigma$-homomorphism $\varphi$: $\mathbf{T}_{\Sigma,gr} \to A$. ■

**6.5 Corollary.** For every $\Sigma$–algebra A and for every sort $S \in \mathbf{S}_\Sigma$, the set $S_A$ is nonempty.
**Proof.** Follows from Assumption 4.11 ii) together with the initiality of the ground term algebra as proved in Corollary 6.4.

In the following we do not distinguish between a $\Sigma$-assignment $\varphi$ and its extension $\varphi_h$ and denote both by $\varphi$.

**Remark.** It may be possible to extend this machinery to non subterm-closed signatures, but there we have the problem that $\mathbf{T}_{\Sigma,gr}$ is not the initial algebra, since operations have to be defined on ill-sorted terms.

## 7. $\Sigma$-congruences.

In this chapter we define and develop some properties of $\Sigma$-congruences for later use in the context of equational theories as well as for semantic issues. Most definitions are straightforward generalizations of the unsorted and order-sorted case, nevertheless, they should be made precise.

We define $SUB_\Sigma$-invariant $\Sigma$-congruences in the usual way as follows:

**7.1 Definition.** Let A be a $\Sigma$-algebra. Then the binary relation $\equiv$ on A is a **$\Sigma$-congruence**, iff the following conditions hold:

i) The relation $\equiv$ is an equivalence relation on A.
ii) For every $f \in \mathbf{F}_{\Sigma,n}$, and for all element $a_i, b_i \in A$:
if $a_i \equiv b_i$ for $i = 1,\ldots,n$, and $(a_1,\ldots,a_n), (b_1,\ldots,b_n) \in \mathcal{D}(f_A)$ then $f_A(a_1,\ldots,a_n) \equiv f_A(b_1,\ldots,b_n)$.

Furthermore, we call a congruence **strong** [Gr79], iff $a_i \equiv b_i$ for $i = 1,\ldots,n$, and

$(a_1,\dots,a_n) \in \mathcal{D}(f_A)$ implies that also $(b_1,\dots,b_n) \in \mathcal{D}_A(f)$. ■

**7.2 Definition.** Let A be a $\Sigma$-algebra. Then the $\Sigma$-congruence $\equiv$ on A is called fully invariant [BS81] iff for every $\Sigma$-endomorphism $\varphi$ of A we have:

$$a \equiv b \;\Rightarrow \varphi(a) \equiv \varphi(b). \;■$$

Fully invariant congruences on the free term-algebra $\mathbf{T}_\Sigma$ are also called $SUB_\Sigma$-invariant, since in this case it is sufficient to use $\Sigma$-substitutions instead of all $\Sigma$-endomorphisms.
An important example for fully invariant congruences are equational theories. In this thesis we are mainly interested in fully invariant $\Sigma$-congruences.

An example for a strong congruence is syntactical equality of terms.
An instructive example for a congruence that is not strong is:

**7.3 Example.** Let $\Sigma$ be a signature with $\Sigma := \{B \sqsubset A,\ b{:}B,\ f{:}B\mathrm{x}B \to A,\ h{:}B \to B\}$. Assume that f is idempotent, i.e. we have the defining equation $f(x, x) = x$. Let $\equiv$ be a fully invariant $\Sigma$-congruence on $\mathbf{T}_\Sigma$ generated by this equations. (In paragraph 9 equational theories are treated in more detail)
The two terms b and f(b b) are congruent, but have a different sort. The constant b is of sort B and the term f(b, b) is of sort A. Hence h(b) is a well-sorted term, whereas h(f(b, b)) is not. This means the congruence is not strong in the sense of Definition 7.1. ■

**7.4. Definition.** Let A be a $\Sigma$-algebra and let $\equiv$ be a $\Sigma$-congruence on A. Then we define the **quotient $\Sigma$-algebra** as the the factor $A/\equiv$ (the quotient of A modulo $\equiv$) as follows:

i) $S_{A/\equiv} := \{a/\equiv \mid a \in S_A\}$ for all $S \in \mathbf{S}_\Sigma$.
ii) $\mathcal{D}(f_{A/\equiv}) := \mathcal{D}(f_A)/\equiv$.
iii) For all $(a_1/\equiv, \dots, a_n/\equiv) \in \mathcal{D}(f_{A/\equiv})$ we have $f_{A/\equiv}(a_1/\equiv, \dots, a_n/\equiv) := f_A(a_1,\dots,a_n)/\equiv$.

It is not difficult to see that $A/\equiv$ is a $\Sigma$-algebra:

**7.5 Proposition.** Let A be a $\Sigma$-algebra and let $\equiv$ be a $\Sigma$-congruence. Then $A/\equiv$ is a $\Sigma$-algebra and the (canonical) mapping $\gamma\colon A \to A/\equiv$ with $\gamma(a) := a/\equiv$ is a $\Sigma$-homomorphism.

**Proof.** The well-definedness of $f_{A/\equiv}$ follows from Definition 7.1 ii), hence $A/\equiv$ is a $\Sigma$-quasi-algebra. We prove the requirements of Definition 6.1:

i) Let $R,S \in \mathbf{S}_\Sigma$ with $R \sqsubseteq S$. Then $R_A \subseteq S_A$, since A is a $\Sigma$-algebra. Hence the relation $\{a/\equiv \mid a \in R_A\} \subseteq \{a/\equiv \mid a \in S_A\}$ holds, which means $R_{A/\equiv} \subseteq S_{A/\equiv}$.
ii) Let t:S be a term declaration and let $\varphi_\equiv\colon \mathbf{V}_\Sigma \to A/\equiv$ be a (partial) $\Sigma$-assignment with

$\mathcal{D}(\varphi_{\equiv}) \subseteq V(t)$. By Definition 7.4 there exists a partial $\Sigma$-assignment. $\varphi: V_\Sigma \to A$ with $\varphi(x)/\equiv\ = \varphi_{\equiv}(x)$ and $\mathcal{D}(\varphi) = \mathcal{D}(\varphi_{\equiv})$. By an easy induction argument we see that $\varphi(t)/\equiv\ = \varphi_{\equiv}(t)$. Definition 6.1 ii) implies that $\varphi(t) \in S_A$, hence $\varphi(t)/\equiv\ = \varphi_{\equiv}(t) \in S_{A/\equiv}$. ❑

In order to show that $\gamma$ is a $\Sigma$-homomorphism, we have to check the three conditions of Definition 6.2. The first two, namely $\gamma(S_A) \subseteq S_{A/\equiv}$ for all $S \in S_\Sigma$ and $\gamma(\mathcal{D}(f_A)) \subseteq \mathcal{D}(f_{A/\equiv})$ for all $f \in F_\Sigma$, are trivially satisfied. The third condition follows directly from Definition 7.4 iii). ■

There is as usual a strong connection between $\Sigma$-congruences and $\Sigma$-homomorphisms: For two $\Sigma$-algebras A, B and a $\Sigma$-homomorphism $\varphi : A \to B$ let the relation $\equiv_\varphi$ (the **kernel** of $\varphi$) on A be defined as $a_1 \equiv_\varphi a_2$ iff $\varphi(a_1) = \varphi(a_2)$ for all $a_1, a_2 \in A$.

**7.6 Proposition.** Let $\varphi : A \to B$ be a $\Sigma$-homomorphism of two $\Sigma$-algebras A, B.

i) The kernel of a $\Sigma$-homomorphism is a $\Sigma$-congruence.

ii) $\varphi$ is a $\Sigma$-isomorphism, iff $\varphi$ is bijective, $\varphi(S_A) = S_B$ for every $\Sigma$-sort S and $\varphi(\mathcal{D}(f_A)) = \mathcal{D}(f_B)$ for every $\Sigma$-function symbol f.

iii) If $\varphi$ is surjective, $\varphi(S_A) = S_B$ for every $\Sigma$-sort S and $\varphi(\mathcal{D}(f_A)) = \mathcal{D}(f_B)$ for every $\Sigma$-function symbol f, then $A/\equiv_\varphi$ is $\Sigma$-isomorphic with B.

**Proof.** The proof is straightforward .■

Note that part iii) of the above proposition may be false for a surjective homomorphism $\varphi$ in case the other conditions are not satisfied.

For a $SUB_\Sigma$-invariant $\Sigma$-congruence $\equiv$ on the free term-algebra all endomorphisms of a factor $\mathbf{T}_\Sigma/\equiv$ can be computed from endomorphisms of $\mathbf{T}_\Sigma$:

**7.7 Proposition.** Let A be a $\Sigma$-algebra and let $\equiv$ be a fully invariant $\Sigma$-congruence on A. Then

i) For every endomorphism $\varphi: A \to A$ the mapping $\varphi_{\equiv}: A/\equiv\ \to A/\equiv$ defined as $\varphi_{\equiv}(a/\equiv) := \varphi(a)/\equiv$ is a $\Sigma$-endomorphism on $A/\equiv$. Furthermore $\varphi(a) \equiv \psi(a)$ for all $a \in A$ implies $\varphi_{\equiv} = \psi_{\equiv}$.

ii) For every endomorphism $\psi: \mathbf{T}_\Sigma/\equiv\ \to \mathbf{T}_\Sigma/\equiv$ there exists an endomorphism $\varphi: \mathbf{T}_\Sigma \to \mathbf{T}_\Sigma$ such that $\varphi_{\equiv} = \psi$.

**Proof.** i) Let $\varphi: A \to A$ be an endomorphism. Let $\varphi_{\equiv}: A/\equiv\ \to A/\equiv$ be defined as

$\varphi_{\equiv}(a/\equiv) := \varphi(a)/\equiv$. Then the full invariance implies well-definedness of $\varphi_{\equiv}$. It is an easy task to verify the remaining conditions for a $\Sigma$-homomorphism.

ii) Let $A := \mathbf{T}_\Sigma$ and let $\psi: A/\equiv \to A/\equiv$ be an endomorphism of $A/\equiv$. Then we obtain a $\Sigma$-homomorphism $\varphi: A \to A$ as follows:
Let $\gamma: A \to A/\equiv$ be the canonical $\Sigma$-homomorphism. Then $\psi\gamma : A \to A/\equiv$ is a $\Sigma$-homomorphism.
Let $x \in \mathbf{V}_\Sigma$ be a variable and let $S := \mathbf{LS}_\Sigma(x)$. We have $\psi\gamma x \in S_{A/\equiv}$ and there exists a term $t_x \in S_A$ with $t_x/\equiv = \psi\gamma x$. Now $\varphi$ defined as $\varphi x := t_x$ for all $x \in \mathbf{V}_\Sigma$ is a (total) $\Sigma$-assignment. Obviously we have $\varphi_{\equiv} = \psi$. ∎

## 8. Specifications, Structures and Models.

This paragraph on specifications and models is restricted to clause sets.
In part II.12 we consider an extension to full first order predicate logic, i.e., including the quantifiers $\forall$ and $\exists$.

Usually, the notion specification is only used if some fixed model is to be specified. We use this term also in the general case of arbitrary clause sets.

**8.1 Definition.** A **$\Sigma$-specification** is a pair $\mathcal{S} = (\Sigma, CS)$, where $\Sigma$ is a signature and CS is a well-sorted clause set. We assume that every clause set CS contains the reflexive axioms $x_S = x_S$ for every sort S. ∎

**8.2 Definition.** A **$\Sigma$-quasi-structure** $\mathcal{A}$ is a $\Sigma$-quasi-algebra which has additional denotations $P_A$ for every predicate symbol $P \in \mathbf{P}_\Sigma$, such that

i) $P_A$ is a relation with $P_A \subseteq A^{\mathrm{arity}(P)}$

ii) $=_A$ is the identity on A, i.e., $=_A \; = \; \{(a,a) \mid a \in A\}$.

We say a $\Sigma$-quasi-structure $\mathcal{A}$ is a **$\Sigma$-structure**, iff the underlying $\Sigma$-quasi-algebra is a $\Sigma$-algebra. ∎

Note that Definition 8.2 ii) enforces a particular interpretation of the equality symbol. The only possible interpretation of '=' in structures will be to denote identity.

The notion of a $\Sigma$-quasi-structure is later needed for conservative transformations in II.7.

We do not introduce the notion of the 'domain of a predicate', since it obscures the intuition and complicates proofs. Instead we always assume that the domain is the whole set $A^{\mathrm{arity}(P)}$. A drawback of this omission is the lack of a semantical correspondence of the predicate declarations.

We can extend all notions for algebras to structures: We state those extensions explicitely that deal with atoms and predicates:
A **$\Sigma$-homomorphism** (of $\Sigma$-structures) $\varphi$: $\mathcal{A} \to \mathcal{B}$ is a $\Sigma$-homomorphism of the underlying $\Sigma$-algebras satisfying in addition $(a_1,\ldots,a_n) \in P_A \Rightarrow (\varphi a_1,\ldots,\varphi a_n) \in P_B$.
We can turn $\mathbf{T}_\Sigma$ into a $\Sigma$-structure by adding the definitions $P_{T\Sigma} := \emptyset$ (if P is not the equality symbol). This is in fact the free $\Sigma$-structure.
A **$\Sigma$-congruence** $\equiv$ (of $\Sigma$-structures) on A is a **$\Sigma$-congruence** (of $\Sigma$-algebras) satisfying in addition: if $a_i \equiv b_i$ for $i = 1,\ldots,n$, then $(a_1,\ldots,a_n) \in P_A$ implies $(b_1,\ldots,b_n) \in P_A$.
In a similar way as for $\Sigma$-algebras we have quotients modulo a congruence and all properties are as usual.

Now we can define $\Sigma$-interpretations and $\Sigma$-models for a $\Sigma$-specification $\mathcal{S}$.
Let $\mathcal{S} = (\Sigma,CS)$ be a specification:

A **$\Sigma$-interpretation** $I = (\mathcal{M},\Phi)$ for CS is a $\Sigma$-structure $\mathcal{M}$ together with a $\Sigma$-homomorphism $\Phi$: $\mathbf{T}_\Sigma \to \mathcal{M}$.

Since $\mathbf{T}_\Sigma$ is the free $\Sigma$-strcuture, it suffices to specify a $\Sigma$-assignment $\Phi$: $\mathbf{V}_\Sigma \to \mathcal{M}$.
We say an interpretation $I = (\mathcal{M},\Phi)$ **satisfies** a $\Sigma$-atom $P(t_1,\ldots,t_n) \in \mathbf{A}_\Sigma$, iff $(\Phi t_1,\ldots, \Phi t_n) \in P_{\mathcal{M}}$. Alternativiely, we may say $P(t_1,\ldots,t_n)$ is valid in I. As an extension, we say I satisfies a positive literal +A iff it satisfies the atom A. Furthermore we say I satisfies a negative literal -A iff it does not satisfy the atom A. An interpretation I satisfies a clause C iff some literal in C is valid in I. Note that no interpretation satisfies the empty clause. An interpretation I satisfies a clause set CS, iff it satisfies every clause $C \in CS$.

**8.3 Definition.** A **$\Sigma$-model** $\mathcal{M}$ for a clause set CS is structure $\mathcal{M}$, such that for every $\Sigma$-assignment $\Phi$: $\mathbf{V}_\Sigma \to \mathcal{M}$, the interpretation $(\mathcal{M},\Phi)$ satisfies the clause set CS.
We say a clause set CS is **satisfiable** (**unsatisfiable**), iff there exists some (no) model $\mathcal{M}$ for CS.
Furthermore we say a clause C is a consequence of the clause set CS, iff for every model $\mathcal{M}$ of CS, $\mathcal{M}$ is also a model for C. ■

We give an example for $\Sigma$-models, which shows in particular that equations in specficiations can have strong implications on the sort-structure of the models.

**8.4 Example.**

i) Let $\Sigma := \{B \sqsubset A, C \sqsubset A, b{:}B, c{:}C, g_B{:}A \rightarrow B, g_C{:}A \rightarrow C\}$ and let

$CS := \{ \{x_{1,B} \neq x_{2,C}\}, \{g_B(x_{3,B}) = x_{3,B}\}, \{g_C(x_{3,C}) = x_{3,C}\},$
$\{x_{4,A} = g_B(x_{4,A}), x_{4,A} = g_C(x_{4,A})\} \}.$

These equations in $CS_1$ enforce that in every model M the set $A_M$ is the disjoint union of $B_M$ and $C_M$, i.e., we have $A_M = B_M \cup C_M$ and $B_M \cap C_M = \emptyset$. The clause set CS has a $\Sigma$-model $M = \{b,c\}$ with $A_M = \{b,c\}$, $B_M = \{b\}$ and $C_M = \{c\}$ together with the operations $g_{B,M}(b) = g_{B,M}(c) = b$ and $g_{C,M}(b) = g_{C,M}(c) = c$. Note that $\Sigma_1$ is regular and satisfies conditions 4.4 . ❑

ii) Without equations it is only possible to enforce disjointness of sorts. A clause set that enforces the disjointness of two sorts A and B is $CS := \{ \{P(x_A)\}, \{-P(x_B)\} \}$

For technical reasons one can view an interpretation also as a set of true literals. The corresponding Herbrand interpretations (H-interpretations) or Herbrand models (H-models) are defined as sets of well-sorted ground literals.

**8.5 Definition.** An **H$\Sigma$-interpretation** is a set M of literals (with meaning: set of true or valid literals) satisfying the following conditions:

i) For every well-sorted ground literal L either L or -L is in M.
ii) $t=t \in M$ for every well-sorted ground $\Sigma$–term t (reflexivity).
iii) If $s=t$ is in M, then $t=s$ is in M (symmetry).
iv) If $s_1 = s_2$, $s_2 = s_3 \in M$, then $s_1 = s_3 \in M$ (transitivity).
v) If $s_i = t_i \in M$ and $f(s_1,\ldots,s_n), f(t_1,\ldots,t_n) \in \mathbf{T}_\Sigma$, then $f(s_1,\ldots,s_n)=f(t_1,\ldots,t_n) \in M$.
vi) If $s_i = t_i \in M$ and the literal $\pm P(s_1,\ldots,s_n) \in M$ then $\pm P(t_1,\ldots,t_n) \in M$, provided $\pm P(t_1,\ldots,t_n)$ is well-sorted. ❑

An H$\Sigma$-interpretation M satisfies a clause C iff for every ground instance $\sigma C$ the intersection of M with $\sigma C$ is not empty.

An H-interpretation M is called a **H$\Sigma$-model** of a clause set CS, iff it satisfies every clause $C \in CS$. ■

We show that the notion of satisfiability defined by models and H-models is equivalent. This justifies to use the appropriate definition for completeness proofs for deduction systems. Furthermore the next theorem is a sorted version of the Löwenheim-Skolem theorem, that every satisfiable set of formulae has a model over a countable carrier.

**8.6 Theorem.** Let $\mathcal{S} = (\Sigma, CS)$ be a specification. Then $\mathcal{S}$ has a $\Sigma$-model iff it has a H$\Sigma$-model.

**Proof.** "$\Rightarrow$":

Let $\mathcal{M}$ be a $\Sigma$-model of $\mathcal{S}$. We define M to be the set of all well-sorted ground literals that are satisfied by $\mathcal{M}$. This makes sense, since $\mathbf{T}_{\Sigma,gr}$ is the initial algebra. Let $\gamma$: $\mathbf{T}_{\Sigma,gr} \rightarrow \mathcal{M}$ be the canonical $\Sigma$-homomorphism. We show that M is a H$\Sigma$-model:

i) follows from the definition of M

ii)-vi) are trivial consequences of the initiality of $\mathbf{T}_{\Sigma,gr}$ and the interpretation of the equality symbol in $\mathcal{M}$.

It remains to show that all clauses are satisfied by the H$\Sigma$-model M. Let C be a clause and let $\sigma$ be a well-sorted ground substitution. Then $\gamma\sigma$: $\mathbf{T}_{\Sigma,gr} \rightarrow \mathcal{M}$ is a $\Sigma$-homomorphism. Hence there exists a literal L in C, such that L is satisfied by the $\Sigma$-interpretation $(\mathcal{M},\gamma\sigma)$. Hence $\sigma$L is satisfied and by definition in M. ❑

"$\Leftarrow$": Let M be a H$\Sigma$-model of $\mathcal{S}$. We define a $\Sigma$-model $\mathcal{M}$ as a quotient algebra of $\mathbf{T}_{\Sigma,gr}$. Let $\equiv$ be the following relation on $\mathbf{T}_{\Sigma,gr}$: $s \equiv t :\Leftrightarrow s = t \in M$. Conditions 8.5 ii)-v) imply that $\equiv$ is a $\Sigma$-congruence on $\mathbf{T}_{\Sigma,gr}$. It is even $SUB_\Sigma$-invariant, since all terms in $\mathbf{T}_{\Sigma,gr}$ are ground. We define $\mathcal{M} := \mathbf{T}_{\Sigma,gr}/\equiv$. We define the relations $P_{\mathcal{M}} := \{P(t_1/\equiv,\ldots, t_n/\equiv) \mid P(t_1,\ldots, t_n) \in M$.

Condition 8.5 vi) implies that the definition of $P_{\mathcal{M}}$ is well-defined. Obviously $\mathcal{M}$ is a structure according to Definition 8.2.

To show that every clause C is satisfied by $\mathcal{M}$ is trivial, since $\Sigma$-assignments correspond to ground substitutions. ■

**8.7 Corollary.** For every clause set CS that has a $\Sigma$-model, there exists a $\Sigma$-model with carrier $\mathbf{T}_{\Sigma,gr}/\equiv$ where $\equiv$ is a $SUB_\Sigma$-invariant $\Sigma$-congruence on $\mathbf{T}_{\Sigma,gr}$. Furthermore if no equational literals are in the clause set then there exists a $\Sigma$-model with carrier $\mathbf{T}_{\Sigma,gr}$. ■

## 9. Equational Theories , Birkhoff´s Theorem.

A **$\Sigma$-equation** is a pair of $\Sigma$-terms, written as $s = t$. An **axiomatization** (or a **specification**) of an **equational theory** is a pair $\mathcal{E} = (\Sigma, E)$ where E is a set of equations (or the set of axioms, or the presentation). We say a $\Sigma$-algebra A **satisfies** an equation $s = t$, written $A \models s = t$, iff $\varphi s = \varphi t$ for every $\Sigma$-assignment $\varphi: T_\Sigma \rightarrow A$. A $\Sigma$-algebra A satisfies a set E of equations (or A is a **$\Sigma$-model** for E), if it satisfies every equation in E. We denote this by $A \models E$. An equation s=t is a **consequence** of a set of identies E, iff s=t is satisfied by every $\Sigma$-model of E. We define the equational theory $T(\mathcal{E})$ to be the set of all consequences of E. Two axiomatizations $\mathcal{E}_1$ and $\mathcal{E}_2$ are **equivalent,** iff their sets of consequences are the same, i.e., if $T(\mathcal{E}_1) = T(\mathcal{E}_2)$. Note that there may exist different axiomatizations of the same

equational theory. We say an equational theory $T(\mathcal{E})$ is **finitely presented**, iff its set of axioms E is finite.
From now on we will use the notation $\mathcal{E}$ instead of $T(\mathcal{E})$ for an equational theory.

**9.1 Definition.** We give a derivation system for order-sorted equational theories. We denote the deduction relation by $\vdash$ :

i) $\vdash$ t=t for every $t \in \mathbf{T}_\Sigma$.
ii) $\{s=t\} \vdash t=s$
iii) $\{r=s, s=t\} \vdash r=t$.
iv) If $f(s_1,\ldots,s_n)$ and $f(t_1,\ldots,t_n)$ are well-sorted, then
$\{s_1 = t_1,\ldots, s_n = t_n\} \vdash f(s_1,\ldots,s_n)= f(t_1,\ldots,t_n)$.
v) $\{s=t\} \vdash \sigma s=\sigma t$ for every well-sorted substitution $\sigma$. ■

We write $\mathcal{E} \vdash s=t$ if there exists a finite proof of s=t starting with equations from E using the rules (i) - (v).
The following completeness theorem is the well-known Birkhoff-Theorem extended to the order-sorted case.

**9.2 Theorem.** $\mathcal{E} \models s=t$ iff $\mathcal{E} \vdash s=t$ for all well-sorted terms s,t and all sets of axioms E.
**Proof.** i) $\mathcal{E} \vdash s=t \Rightarrow \mathcal{E} \models s=t$ :

The proof is by induction on the length of a deduction. We show that if A is a model of the equations on the left hand side of the rules then A is also a model of the derived equation. For rules (i)-(iv) this can easily be verified. To prove the soundness of rule (v), let A be a model of s=t, let $\sigma$ be a $\Sigma$-substitution and let $\varphi$: $\mathbf{V}_\Sigma \rightarrow A$ be a $\Sigma$-assignment. Then $\varphi\sigma$ is also a $\Sigma$-assignment, hence $(\varphi\sigma)s = (\varphi\sigma)t$ and consequently $\varphi(\sigma s) = \varphi(\sigma t)$.

ii) $\mathcal{E} \models s=t \Rightarrow \mathcal{E} \vdash s=t$:

The relation $\equiv$ on $\mathbf{T}_\Sigma$ defined as $s \equiv t$, iff $\mathcal{E} \vdash s=t$, is a $\Sigma$-congruence on $\mathbf{T}_\Sigma$. It is also $SUB_\Sigma$-invariant, since the restriction of a $\Sigma$-endomorphism of $\mathbf{T}_\Sigma$ on a finite set of variables is a $\Sigma$-substitution.

We show that $\mathbf{T}_\Sigma / \equiv$ is a model of $\mathcal{E}$:

Let $\varphi_\equiv$: $\mathbf{V}_\Sigma \rightarrow \mathbf{T}_\Sigma / \equiv$ be a $\Sigma$-assignment and let s=t be an identity from $\mathcal{E}$. Then there exists a $\Sigma$-assignment $\varphi$: $\mathbf{V}_\Sigma \rightarrow \mathbf{T}_\Sigma$ with $\varphi(x)/\equiv\ = \varphi_\equiv(x)$ for all $\Sigma$-variables x. Since $s \equiv t$ and $\equiv$ is $SUB_\Sigma$-invariant, we get $\varphi s \equiv \varphi t$. This means $\varphi s/\equiv\ = \varphi t/\equiv$ , hence by Proposition 7.7 we obtain $\varphi_\equiv(s) = \varphi_\equiv(t)$.

Now we are ready, since an identity $s_0 = t_0$ that is not derivable from $\mathcal{E}$ yields different elements $s_0/\equiv$ and $t_0/\equiv$ , hence $\mathbf{T}_\Sigma / \equiv$ is not a model for $s_0 = t_0$ . ■

As usual we abbreviate $\mathcal{E} \vdash s=t$ as $s =_{\mathcal{E}} t$ or $s =_{\Sigma,E} t$ for $\Sigma$-terms s and t. We have the following fact:

**9.3 Propopsition.** The relation $=_{\Sigma,E}$ is the least $SUB_{\Sigma}$-invariant $\Sigma$-congruence on $\mathbf{T}_{\Sigma}$, such that for all $s=t \in E$ the relation $s =_{\Sigma,E} t$ holds. ■

The quotient algebra $\mathbf{T}_{\Sigma,gr} / =_{\Sigma,E}$ is the **standard model** for the equational theory E. It is the initial model in the variety of all models of $\mathcal{E}$. The quotient algebra $\mathbf{T}_{\Sigma} / =_{\Sigma,E}$ is the free algebra in the variety of all models of $\mathcal{E}$. If $\mathcal{E} = \emptyset$, then $=_{\emptyset}$ is the syntactical equality of terms.

An equational theory $\mathcal{E}$ is **consistent** iff it has a model consisting of more than one element, i.e., there are two terms that are not $=_{\Sigma,E}$ -equal, otherwise we call $\mathcal{E}$ **inconsistent.** Note that a theory is inconsistent, iff the equations $x = y$ are derivable for all $\Sigma$-variables x,y. Nevertheless, for a consistent theory an equation $x = y$ may be derivable for some sorted variables x,y (even with different sorts). This is an appropriate way to encode sorts that consist exactly of one element, such as the sort ZERO in the integers, which has 0 as its unique element.

We extend E-equality to well-sorted substitutions by defining:

$\sigma =_{\Sigma,E} \tau$ , iff $\sigma x =_{\Sigma,E} \tau x$ for all variables x.

If we are only interested in the behaviour on a set V of variables, we write

$\sigma =_{\Sigma,E} \tau$ [V], iff $\sigma x =_{\Sigma,E} \tau x$ for all variables $x \in V$.

If the set of axioms is empty, i.e., there are no defining equations, then we may abbreviate $=_{\Sigma,\emptyset}$ as $=_{\Sigma}$.

Since $=_{\Sigma,E}$ is a $SUB_{\Sigma}$-invariant congruence we have by Proposition 3.10 that $\sigma =_{\Sigma,E} \tau$ and $s =_{\Sigma,E} t$ implies that $\sigma s =_{\Sigma,E} \tau t$. This can be strengthened to

**9.4 Lemma.** If $s =_{\Sigma,E} t$ and $\sigma =_{\Sigma,E} \tau$ $[V(s) \cap V(t)]$, then $\sigma s =_{\Sigma,E} \sigma t$.
**Proof.** see [He87]. ■

An equational theory $\mathcal{E}$ is called **deduction-closed,** iff $s_1 =_{\Sigma,E} t_1, \ldots, s_n =_{\Sigma,E} t_n$ and $f(s_1,\ldots,s_n) \in \mathbf{T}_{\Sigma}$ imply that $f(t_1,\ldots,t_n)$ is also well-sorted (i.e., iff the congruence $=_{\Sigma,E}$ is a strong congruence). Obviously an equational theory $\mathcal{E}$ is deduction-closed, iff the replacement of equals for equals does not produce ill-sorted terms from well-sorted ones.
An equational theory $\mathcal{E}$ is called **sort-preserving,** iff for all relations $s =_{\Sigma,E} t$ we have also $\mathbf{S}_{\Sigma}(s) = \mathbf{S}_{\Sigma}(t)$. This implies that sort-preserving theories are also deduction-closed.

In general it is undecidable whether an equational theory is deduction-closed or sort-preserving (see paragraph II.6). However, for elementary signatures the deduction-closedness is decidable (cf. Proposition II.6.7).

We distinguish different classes of equational theories: A theory $\mathcal{E}$ is **regular,** iff $s =_{\Sigma,E} t$ implies $\mathbf{V}(s) = \mathbf{V}(t)$. Obviously a theory is regular, iff every equation in its axiomatization has this property. A theory is **collapse-free,** iff $t =_{\Sigma,E} x$ implies that t is the variable x itself. Again it can be decided by looking at the axioms whether a theory is collapse-free or not. A theory is **finite,** iff every equivalence class w.r.t $=_{\Sigma,E}$ is finite. A theory is **simple**, iff $s =_{\Sigma,E} t$ implies that s is not a proper subterm of t [BHS86]. A theory is **$\Omega$-free**, iff for every function symbol f the equations $f(s_1,\ldots,s_n) =_{\Sigma,E} f(t_1,\ldots,t_n)$ imply $s_i =_{\Sigma,E} t_i$ for all i. It is undecidable whether equational theories are finite, simple or $\Omega$-free [BHS86].

The word problem of an equational theory is the problem to decide whether $s =_{\Sigma,E} t$ holds for given $\Sigma$-terms s,t . In general the word-problem is undecidable [Ta79, Mc76]
However in (unsorted) finite equational theories the word problem is always decidable. In order-sorted, finite equational theories the word problem is decidable, if they are deduction-closed. In paragraph IV.3 we take a closer look at finite theories.

## 10. Substitutions.

We introduce some notation and technicalities that are needed in later proofs. Almost all notions, lemmas and proofs are straightforward extensions of the unsorted case by using the operator $\bar{}$ for lifting results of the unsorted case to the order-sorted case, as e.g. in [He83, Ed85, Hu76] .

Idempotent substitutions (i.e., $\sigma$ satisfies $\sigma\sigma = \sigma$) are an important subset of all substitutions. The crucial property of idempotent substitutions is that their domain and codomain have disjoint sets of variables, i.e., $\mathrm{DOM}(\sigma) \cap \mathrm{I}(\sigma) = \emptyset$. Since these two properties are equivalent, we often say a substitution is idempotent and mean $\mathrm{DOM}(\sigma) \cap \mathrm{I}(\sigma) = \emptyset$. A disadvantage is that the composition of idempotent substitutions may not be idempotent, hence the subset of idempotent substitutions is insufficient as a theoretical basis.

There is a sufficient criterion for a product of idempotent substitutions to be idempotent:
**10.1 Lemma.** [He83]: Let $\sigma,\tau$ be idempotent $\Sigma$-substitutions with $\mathrm{DOM}(\tau) \cap \mathrm{I}(\sigma) = \emptyset$.
Then $\sigma\circ\tau$ is idempotent. ■

Two $\Sigma$-substitutions $\sigma,\tau$ with $(DOM(\sigma) \cup I(\sigma)) \cap (DOM(\tau) \cup I(\tau)) = \emptyset$ are permutable, i.e., $\sigma \circ \tau = \tau \circ \sigma$.

For two $\Sigma$-substitutions $\sigma$ and $\tau$ with $\sigma = \tau\ [DOM(\sigma) \cup DOM(\tau)]$, we can define their **union**, denoted by $\sigma \cup \tau$, as the substitution with $DOM(\sigma \cup \tau) = DOM(\sigma) \cup DOM(\tau)$, $\sigma \cup \tau = \sigma\ [DOM(\sigma)]$ and $\sigma \cup \tau = \tau\ [DOM(\tau)]$.

Let us recall the definition of a $\Sigma$-renaming: A substitution $\rho \in SUB_\Sigma$ is called a **$\Sigma$-renaming**, iff $\rho$ maps variables into variables, $\rho$ is injective on $DOM(\rho)$, and $S(x) = S(\sigma x)$ for all $x \in \mathbf{V}_\Sigma$. Note that $\Sigma$-renamings may be not idempotent. For every $\Sigma$-renaming $\rho = \{x_1 \leftarrow y_1,\ldots, x_n \leftarrow y_n\}$ a converse $\rho^-$ is defined as $\rho^- := \{y_1 \leftarrow x_1,\ldots, y_n \leftarrow x_n\}$. A substitution $\rho \in SUB_\Sigma$ is called a **$\Sigma$-permutation,** iff $\rho$ is a bijective $\Sigma$-renaming. It follows from this definition that a $\Sigma$-permutation $\rho$ has an inverse $\rho^-$ with $\rho\rho^- = \rho^-\rho = Id_\Sigma$. Hence the set of all permutations is a group together with $Id_\Sigma$ and the composition of substitutions $(\circ)$. Obviously restrictions of $\Sigma$-permutations are $\Sigma$-renamings. Furthermore every $\Sigma$-renaming is a restriction of some $\Sigma$-permutation.

There are enough (idempotent) renamings to rename every finite set V of variables, since we have assumed that for every sort there are infinitely many variables.

We summarize the properties of $\rho^-$ in a Lemma (cf. 2.1):

**10.2 Lemma.** Let $\rho$ be a $\Sigma$-renaming. Then:

i) $\rho^-$ is a $\Sigma$-renaming ii) $DOM(\rho) = COD(\rho^-)$

iii) $DOM(\rho^-) = COD(\rho)$ iv) $(\rho^-)^- = \rho$

v) $\rho^- \circ \rho = Id_\Sigma\ [DOM(\rho)]$

vi) If $\rho$ is idempotent, then $\rho \cup \rho^-$ is a $\Sigma$-permutation.

vii) If $\rho$ is a permuation, then $\rho\rho^- = \rho^-\rho = Id_\Sigma$ ■

**10.3 Proposition.**

i) Let $s,t \in \mathbf{T}_\Sigma$. Then $s \equiv_\Sigma t \Leftrightarrow$ there exists a $\Sigma$-permutation $\xi$ with $\xi s = t$.

ii) Let $\sigma,\tau \in SUB_\Sigma$. Then $\sigma \equiv_\Sigma \tau\ [W] \Leftrightarrow$ there exists a $\Sigma$-permutation $\xi$ with $\xi\sigma = \tau\ [W]$.

**Proof.** For the unsorted case, see for example [Hu76], we have that $\lambda_1\sigma = \tau\ [W]$ and $\lambda_2\sigma = \tau\ [W]$ implies that $\lambda_1 = \lambda_2\ [V(\sigma W)]$. Furthermore there exists an unsorted renaming $\rho$ with $\rho\sigma = \tau\ [W]$. Hence $\rho_{|V(\sigma W)}$ is well-sorted and a $\Sigma$-renaming. ■

Let $U \subseteq SUB_\Sigma$ be a set of substitutions and let $W \subseteq Z \subseteq V$. Then we say U is **based on W away from Z,** iff for all substitutions $\sigma$ in U we have $DOM(\sigma) = W$ and $I(\sigma) \cap Z = \emptyset$.

**10.4 Lemma.** Let $W \subseteq V$ and let $\tau \in SUB_\Sigma$. Then for every idempotent $\Sigma$-renaming $\rho$ with $DOM(\rho) \supseteq V(\tau W)$ : $\tau \equiv_\Sigma \rho \circ \tau\ [W]$.

Proof. Follows from Lemma 10.2. ■

The next proposition is trivial for the unsorted case and in the order-sorted case it is a consequence of the finiteness of the set of sorts $\mathbf{S}_\Sigma$.

**10.5 Proposition.** Let $\Sigma$ be a finite signature. Let W be a finite set of variables and let n be a natural number.

i) The set $\{t \in \mathbf{T}_\Sigma \mid depth(t) \le n\}$ contains a finite number of $\equiv_\Sigma$ - congruence classes

ii) The set $\{\sigma \in \Sigma \mid depth(\sigma) \le n\}$ contains a finite number of $\equiv_\Sigma [W]$- congruence classes.

**Proof.** i) In the unsorted case we have: $\{t \in \mathbf{T}_\Sigma \mid depth(t) \le n\}$ contains a finite number of $\equiv_{\bar{\Sigma}}$ - congruence classes. Furthermore if $s \equiv_\Sigma t$, then $s \equiv_{\bar{\Sigma}} t$. Terms s,t with $s \equiv_{\bar{\Sigma}} t$ have the same occurrences. The $\equiv_\Sigma$ - congruence class of a term s is determined by its $\equiv_{\bar{\Sigma}}$ - congruence class and by the sort of its variables. There are only a finite number of possibilities for different sorts of variables, hence a $\equiv_{\bar{\Sigma}}$ - congruence class is partitioned into a finite number of $\equiv_\Sigma$ - congruence classes.

ii) The proof is trivially extended to vectors of finite length and hence to substitutions. ■

We note some observations on noncyclic substitutions that are needed later on.

**10.6 Definition.** A variable $x_1$ is **strongly cyclic** for a substitution $\sigma$, iff there are variables $x_i$, $i = 2,\ldots,n$ such that $x_i \in \mathbf{V}(\sigma x_{i-1})$, $i = 2,\ldots,n$ , $x_1 \in \mathbf{V}(\sigma x_n)$ and $\sigma^{n-1} x_1 \neq x_1$.

It is **weakly cyclic**, iff there are variables $x_i$, $i = 2,\ldots,n$ such that $x_i \in \mathbf{V}(\sigma x_{i-1})$, $i = 2,\ldots,n$ and $x_1 \in \mathbf{V}(\sigma x_n)$

**10.7 Lemma.** i) If x is strongly cyclic in $\sigma$ then x is also weakly cyclic in $\sigma$.

ii) If there is no weakly cyclic variable in $\sigma$, then $\sigma^m x = x$ for some $m > 0$ implies $\sigma x = x$, i.e. $DOM(\sigma^n) = DOM(\sigma)$ for all n

**10.8 Example.**

i) Idempotent substitutions have no cyclic variables

ii) The substitution $\sigma := \{x \leftarrow f(x)\}$ has the strongly cyclic variable x and $\sigma^m = \{x \leftarrow f^m(x)\}$.

iii) The substitution $\sigma := \{x \leftarrow f(y),\ y \leftarrow z\}$ has no cyclic variable and

$\sigma^m = \{x \leftarrow f(z), y \leftarrow z\}$ for all $m \geq 2$

iii) The substitution $\sigma := \{x \leftarrow f(y), y \leftarrow z, z \leftarrow y\}$ has no strongly cyclic variable, but y and z are weakly cyclic. If we compute the powers of $\sigma$, we obtain $\sigma^2 = \{x \leftarrow f(z)\}$ and $\sigma^3 := \{x \leftarrow f(y), y \leftarrow z, z \leftarrow y\} = \sigma$.

**10.9 Lemma.** Let $\sigma$ be a substitution without strongly cyclic variables. Then there exist natural numbers $m,k > 0$ such that $\sigma^m = \sigma^{m+k}$.

**Proof.** We have $DOM(\sigma^n) \subseteq DOM(\sigma)$ and $I(\sigma^n) \subseteq I(\sigma)$.

If the depth of terms in $COD(\sigma^n)$ is bounded, then not all $\sigma^n$ can be different, since there are at most finitely many terms of bounded depth and with a fixed set of symbols.

Hence in this case there exist natural numbers $m,k > 0$ such that $\sigma^m = \sigma^{m+k}$.

If the depth is unbounded, then there exists a variable $x_0$, such that depth $(\sigma^n x_0)$ is not bounded.

This means depth $(\sigma^n(\sigma x_0))$ is not bounded, hence there exists some variable $x_1 \in V(\sigma x_0)$, such that depth $(\sigma^n x_1)$ is not bounded. In this manner we can construct an infinite chain $x_0, x_1, \ldots$, such that $x_i \in V(\sigma x_{i-1})$. Since there are only finitely many variables, there exists a variable that occurs twice in the chain. Without loss of generality we can assume that $x_0$ occurs twice and $x_0 = x_n$.

If all terms $\sigma x_i$ are variables for $i = 1,\ldots,n$, then the depth of $\sigma x_0$ is bounded, hence there exists a variable $x_j$ such that $\sigma x_j$ is not a variable. Hence $x_j$ is a strongly cyclic variable in $\sigma$. ■

**10.10 Lemma.** Let $\sigma$ be a substitution without strongly cyclic variables. Then there exists a number n, such that $\sigma^n$ is idempotent. Furthermore if $\sigma^n$ and $\sigma^m$ are idempotent powers of $\sigma$, then $\sigma^n = \sigma^m$.

**Proof.** Using the last lemma we see that there exist $k,m > 0$ such that $\sigma^m = \sigma^{m+k}$. With $n = km$ we obtain $\sigma^{km} \sigma^{km} = \sigma^{km}$ by applying $\sigma^m = \sigma^{m+k}$ repeatedly.

If $\sigma^n$ and $\sigma^m$ are idempotent we compute $\sigma^{mn}$ in two ways: If we use the idempotency of $\sigma^m$, then we obtain $\sigma^{mn} = \sigma^m$. From the idempotency of $\sigma^n$ we obtain $\sigma^{mn} = \sigma^n$, hence $\sigma^n = \sigma^m$. ■

The converse of Lemma 10.10 holds:

**10.11 Lemma.** Let $\sigma$ be a substitution such that $\sigma^m$ is idempotent for some $m > 0$. Then $\sigma$ contains no strongly cyclic variables.

**Proof.** Obviously the depths of terms in $COD(\sigma^n)$ are bounded. Suppose $\sigma$ contains a strongly cyclic variable, then there is a power $\sigma^k$ of $\sigma$ such that there is a variable x with $\sigma^k x$ is a nonvariable term and $x \in V(\sigma^k x)$, hence the depth of $\sigma^{kl} x$ for $l \geq 1$ is unbounded, which is a contradiction. ■

**10.12 Definition.** For a substitution $\sigma$ without strongly cyclic variables we define the **idempotent closure** $\sigma^*$ as the least power $\sigma^n$ that is idempotent. ■

The above lemmas show that the idempotent closure of a substitution can be defined as any idempotent power of $\sigma$.

**10.13 Lemma.** Let $\sigma$ be a substitution that has no weakly cyclic variables. Then there exists a natural number $m > 0$ such that $\sigma^m = \sigma^{m+1}$.

**Proof.** By Lemma 10.10 we have that there exists a number $m > 0$ such that $\sigma^m$ is idempotent. Furthermore Lemma 10.7 shows that $DOM(\sigma^m) = DOM(\sigma)$. Since $DOM(\sigma^m) \cap I(\sigma) = \emptyset$, we have $\sigma\sigma^m = \sigma^m$. ■

## 11. Theory-Unification and Theory-Matching.

Let $\mathcal{E} = (\Sigma, E)$ be an axiomatization of an equational theory.
The subsumption relation for two terms $s,t \in \mathbf{T}_\Sigma$ is defined as follows:

$$s \geq_{\Sigma,E} t \ :\Leftrightarrow\ \exists\, \lambda \in SUB_\Sigma \text{ with } s =_{\Sigma,E} \lambda t.$$

In this case we say t is more general than s or s is an E-instance of t. Obviously the relation $\geq_\Sigma$ is a quasi-ordering on $\mathbf{T}_\Sigma$.
Note that sometimes the reversed ordering is used, (cf. [Si84, Si86, Sz82, Sch85]).
The corresponding equivalence relation is denoted as $\equiv_{\Sigma,E}$ i.e.,

$$s \equiv_{\Sigma,E} t \quad \text{iff} \quad s \geq_{\Sigma,E} t \text{ and } t \geq_{\Sigma,E} s$$

We extend the subsumption relation to substitutions:
Let $\sigma,\tau \in SUB_\Sigma$ and let $V \subseteq \mathbf{V}_\Sigma$. Then

$$\sigma \geq_{\Sigma,E} \tau\ [V] \quad :\Leftrightarrow\ \exists\, \lambda \in SUB_\Sigma \text{ with } \sigma =_{\Sigma,E} \lambda\tau\ [V].$$

In this case we say $\tau$ subsumes $\sigma$ modulo V or $\tau$ is more general than $\sigma$ wrt V. Obviously the relation $\geq_{\Sigma,E}[V]$ is a quasi-ordering. The corresponding equivalence relation is denoted by $\equiv_{\Sigma,E}$ i.e.,

$$\sigma \equiv_{\Sigma,E} \tau \ [V] \ \text{iff} \ \ \sigma \geq_{\Sigma,E} \tau \ [V] \ \text{and} \ \tau \geq_{\Sigma,E} \sigma \ [V] .$$

Note that $\equiv_{\Sigma,\emptyset}$ [V] and $\leq_{\Sigma,\emptyset}$ [V] may be abbreviated as $\equiv_{\Sigma}$ [V] and $\leq_{\Sigma}$ [V] , respectively (cf. paragraph 5). If V is the set of all $\Sigma$-variables, then we will omit the set of variables in the notation of the subsumption of terms and substitutions.

Given an equational theory $\mathcal{E} = (\Sigma,E)$, an **E-unification problem** is a finite set of equations denoted as $\Gamma = \langle s_i = t_i \mid i = 1,\ldots,n\rangle_{\mathcal{E}}$. Instead of an E-unification problem we sometimes speak of a system of equations to be solved. We say a well-sorted substitution $\sigma$ is an **E-unifier** of $\Gamma$ (or an E-solution of $\Gamma$) iff $\sigma s_i =_{\Sigma,E} \sigma t_i$ for all $s_i = t_i \in \Gamma_E$ . The set of all E-unifiers of the system $\Gamma$ is denoted by $U_{\Sigma,E}(\Gamma)$. Obviously the set $U_{\Sigma,E}(\Gamma)$ is a left ideal in the set of all well-sorted substitutions, i.e. $SUB_{\Sigma} \circ U_{\Sigma,E}(\Gamma) = U_{\Sigma,E}(\Gamma)$ or equivalently every instance of an E-unifier is also an E-unifier. The set $U_{\Sigma,E}(\Gamma)$ is recursively enumerable (even for an infinitely presented equational theory) by a simple dovetailing argument.
However, for most purposes it is not necessary to compute the whole set of E-unifiers, but a smaller subset from which we can obtain every solution by instantiation.

We say a set $cU \subseteq SUB_{\Sigma}$ is a **complete** set of E-unifiers for $\Gamma_E$, iff the following conditions hold:

i) $cU \subseteq U_{\Sigma,E}(\Gamma)$ (correctness)
ii) $\forall \theta \in U_{\Sigma,E}(\Gamma)\ \exists \sigma \in cU\colon\ \theta \geq_{\Sigma,E} \sigma\ [V(\Gamma)]$. (completeness)

The set of all complete sets is denoted by $CU_{\Sigma,E}(\Gamma)$. We may use the notation $cU_{\Sigma,E}(\Gamma)$ for a special complete set of E-unifiers.
Furthermore a complete set cU of E-unifiers is called **minimal** or a set of **most general E-unifiers** (or set of '**mgus**'), iff in addition

iii) $\forall \sigma,\tau \in cU\colon \sigma \geq_{\Sigma,E} \tau\ [V(\Gamma)] \Rightarrow \ \sigma = \tau$. (minimality)

All minimal sets of E-unifers are collected in the set $MU_{\Sigma,E}(\Gamma)$. We may denote a special set of mgus as $\mu U_{\Sigma,E}(\Gamma)$, if it is clear from the context, which particular set we mean. In general there are infinitely many different sets of mgus for some system of equations $\Gamma$, but they are all equivalent, in the sense: if $\mu U_1, \mu U_2 \in MU_{\Sigma,E}(\Gamma)$ then there is a bijection $\alpha\colon \mu U_1 \rightarrow \mu U_2$ with $\alpha(\sigma) \equiv_{\Sigma,E} \sigma\ [V(\Gamma)]$ for all $\sigma \in \mu U_1$. This was proved by [Hu76, FH86], and is a trivial result for the equivalence for bases of upper segments in a quasi-ordering.
Unfortunately, a minimal set of mgu's does not always exist, the first example for such a

theory was given in [FH86]. Recently it was shown that the theory of associativity and idempotence is also an example for a theory where in some cases a set of mgu's does not exist [Ba86, Sch86].

Depending on the cardinality of the sets of most general unifiers we can classify equational theories according to the following hierarchy [Si75, Sz82, Si86, Si87]:

A theory $\mathcal{E}$ is **unitary unifying** (or is of **unification type 1** or $\mathcal{E} \in \mathcal{U}_1$)
iff $\mu U_{\Sigma,E}(\Gamma)$ exists and $|\mu U_{\Sigma,E}(\Gamma)| \leq 1$ for all equation systems $\Gamma$.
A theory $\mathcal{E}$ is **finitary unifying** (of **unification type** $\omega$ or $\mathcal{E} \in \mathcal{U}_\omega$)
iff $\mu U_{\Sigma,E}(\Gamma)$ exists and $|\mu U_{\Sigma,E}(\Gamma)| < \infty$ for all equation systems $\Gamma$.
A theory $\mathcal{E}$ is **infinitary unifying** (of **unification type** $\infty$ or $\mathcal{E} \in \mathcal{U}_\infty$)
iff $\mu U_{\Sigma,E}(\Gamma)$ exists for all equation systems $\Gamma$ and $|\mu U_{\Sigma,E}(\Gamma)| = \infty$ for some equation system $\Gamma$.
A theory $\mathcal{E}$ is **nullary unifying** ( of **unification type 0** or $\mathcal{E} \in \mathcal{U}_0$)
iff $\mu U_{\Sigma,E}(\Gamma)$ does not exist for some equation system $\Gamma$.

We use as abbreviation $\mathcal{U} := \mathcal{U}_1 \cup \mathcal{U}_\omega \cup \mathcal{U}_\infty$ and also say that theories $\mathcal{E} \in \mathcal{U}$ are **unification based**. The unification type of a theory is undecidable (cf. [BHS86]). The subclass of unitary or finitary theories where the sets of unifiers are always effectively computable is denoted as $\mathcal{U}_{1,\text{eff}}$ or $\mathcal{U}_{\omega,\text{eff}}$
Usually the unification type is defined using a single equation. But in general the problem is to unify lists of terms. The crucial point is that the definition of the unification type via a single equation and via a system of equations is not equivalent. An example is given in the appendix showing that there exists a theory of (single-equation) unification type $\infty$, that is nullary unifying with respect to equation systems. Hence our definition here is more adequate for describing the unification behaviour of theories. Furthermore, the result of [BS86] that there does not exist a finitary theory with an upper bound on the cardinality of minimal unifier sets, provided there is at least one free function symbol with more than one argument in the signature, is true without any restrictions, if our definition of unification type is used.

However, for the case of unitary and finitary theories, the two definitions are equivalent (cf. [He86] for a proof in the unsorted case). The same is true if the signature contains at least one free function symbol of arity greater than 1.

By Lemma 10.3, we can always find a minimal (or a complete) set of idempotent E-unifiers by renaming their codomain, hence it is not a restriction to assume that all unifiers in a minimal set of E-unifiers are idempotent.

We will also consider one-sided unification problems or **matching** problems. We denote an equation system as a matching problem as follows:

$\Delta := \langle s_i \ll t_i \mid i = 1,\ldots,n\rangle_E$

To solve the matching problem $\Delta$ means to find well-sorted substitutions $\sigma$ with $DOM(\sigma) \cap V(t_1,\ldots,t_n) = \emptyset$ and $\sigma s_i =_{\Sigma,E} t_i$ for all $i = 1,\ldots,n$. In this case we call $\sigma$ an **E-matcher**. The set of all E-matchers is denoted as $M_{\Sigma,E}(\Delta)$. Note that the set of all E-matchers is a left ideal in the monoid of all substitutions $\sigma$ with $DOM(\sigma) \cap V(t_1,\ldots,t_n) = \emptyset$. Similar as for unification we define minimal and complete sets of matchers. We use the relation $\leq_{\Sigma,E}[V(\Delta)]$ for comparing matchers. This is equivalent instantiate only with substitutions $\sigma$ with $DOM(\sigma) \cap V(t_1,\ldots,t_n) = \emptyset$.

This definition of matching is not less general than the problem of one-sided unification, since the sets $\{\sigma \in SUB_\Sigma \mid \sigma s_i =_{\Sigma,E} t_i$ for all $i = 1,\ldots,n\}$ and $(M_{\Sigma,E}\langle \rho s_i \ll t_i \mid i = 1,\ldots,n\rangle) \circ \rho$ (where $\rho$ is an appropriate $\Sigma$–renaming) are equivalent with respect to $=_{\Sigma,E}[V(\Delta)]$.

Analogous to the unification hierarchy we classify the equational theories into **unitary matching** ($\mathcal{E} \in \mathcal{M}_1$), **finitary matching** ($\mathcal{E} \in \mathcal{M}_\omega$), **infinitary matching** ($\mathcal{E} \in \mathcal{M}_\infty$), and **nullary matching** ($\mathcal{E} \in \mathcal{M}_0$) theories.

It is undecidable where a theory resides in the matching hierarchy [BHS86].

This definition implies that in regular theories every matcher is minimal [Sz82] . Hence we have that regular equational theories are not in $\mathcal{M}_0$. In [Sz82] it is shown that in the unsorted case the $\Omega$-free theories are exactly the regular and unitary matching theories. In paragraph IV.2 we consider the connection between unitary matching and $\Omega$-free theories for the sorted case .

A further problem tackled in this thesis is the problem of **weakening** [Wa83], that is, given a (non well-sorted) substitution $\tau$, find a well-sorted substitutions $\sigma \in SUB_\Sigma$ such that $\sigma\tau$ is well-sorted. We denote such problems simply as:

$\langle \tau \in SUB_\Sigma \rangle$.

We denote the set of solutions as $W_\Sigma\langle \tau \in SUB_\Sigma\rangle$ or simply as $W_\Sigma\langle \tau \rangle$.

We will consider also weakening problems for terms t, either denoted $W_\Sigma\langle t \in \mathbf{T}_\Sigma\rangle$ or $W_\Sigma\langle t \in \mathbf{T}_{\Sigma,S}\rangle$ or $W_\Sigma\langle t \sqsubseteq S\rangle$. The problem is to find the well-sorted substitutions $\sigma$ with $\sigma t \in \mathbf{T}_\Sigma$ or $\sigma t \in \mathbf{T}_{\Sigma,S}$ for some sort S.

Again we consider minimal and complete subsets of weakenings, denoted as $cW_\Sigma$ and $\mu W_\Sigma$. If not stated otherwise, we use the quasi-ordering $\leq_\Sigma[I(\tau)]$ ( $\leq_\Sigma[V(t)]$) for comparing the weakenings.

Note that we do not consider weakening problems with respect to an equational theory.

We say a theory is simple, iff $s =_{\Sigma,E} t$ implies that s is not a proper subterm of t. In [BHS86] it is shown that in simple theories an occurs-check is possible during unification, i.e. the equation $<x = t>_E$ is unsolvable, if $x \in \mathbf{V}(t)$. Furthermore it is shown there that simplicity is an undecidable property of an equational theory.

A theory is **Noetherian,** iff there are no infinite properly descending chains of substitutions with respect to $\leq_{\Sigma,E}$ [W] for a finite set of variables W. In part IV we show that every finite equational theory is Noetherian.

## 12. Computational Logic.

There is another important derivation system, called (undirected) **demodulation** in the field of Automated Deduction [WR67], which allows to replace equals by equals. In the following we assume that a fixed equational theory $\mathcal{E} = (\Sigma,E)$ is given. We shall define demodulation for a sorted logic.

**12.1 Definition.** Let s,t be $\Sigma$-terms.

Then we can deduce t from s, denoted as

$$s \longrightarrow_{\pi,e,\sigma} t$$

iff there exists an equation e of the form $r = l$ or $l = r$ in E, a substitution $\sigma \in SUB_\Sigma$ and an occurrence $\pi \in D(s)$ such that $s\backslash\pi = \sigma l$ and $t = s[\pi \leftarrow \sigma r]$. ■

This definition includes the definition of $s \longrightarrow_{\pi,e,\sigma} t$ for ill-sorted terms. If it is necessary to distinguish between $s \longrightarrow_{\pi,e,\sigma} t$ for well-sorted terms andfor ill-sorted terms, we shall say so explicitly. The default assumption is that the relation is restricted to well-sorted terms.

A derivation is a finite sequence of such derivation steps. If a term s can be derived from a term t by a finite sequence of such steps, we denote this by $s \xrightarrow{*} t$. Obviously $\xrightarrow{*}$ is symmetric. It is not difficult to see that for the unsorted case this derivation system is equivalent to the one defined in 9.1.

We call an equational theory $\mathcal{E}$ **demodulation-complete,** iff $s =_{\Sigma,E} t \iff s \xrightarrow{*} t$ for all well-sorted terms.

In part II 2 we show that for the extended relation $\xrightarrow{*}$ on all unsorted terms, we have always $s =_{\Sigma,E} t \iff s \xrightarrow{*} t$, but the examples below demonstrate that there are equational theories that are not demodulation-complete.

We give two examples for theories that are not demodulation-complete. Note that the corresponding equational theories are not deduction-closed:

**12.2 Example.** a) Let $S_\Sigma := \{A,B,C,D\}$ and let a,b,c,d be constants of sort A,B,C,D, respectively.
Let f be a binary function with $f{:}AxB \to A$ and $f{:}CxD \to A$. Let $E := \{a=c, b=d\}$ be the set of axioms. We have $f(a,b) =_{\Sigma,E} f(c,d)$, but not $f(a,b) \xrightarrow{*}_\Sigma f(c,d)$, since the intermediate terms f(a,d) and f(c,b) are not well-sorted.

b) Let $S_\Sigma := \{A,B,C,D\}$ and let a,b,c be constants of sort A,B,C, respectively.
Let f be a unary function with $f{:}A \to D$, $f{:}C \to D$, Let $E := \{a=b, b=c\}$ be the set of axioms. We have $f(a) =_{\Sigma,E} f(c)$, but not $f(a) \xrightarrow{*} f(c)$, since the intermediate term f(b) is not well-sorted. ■

Example b) shows that even a deductive calculus that allows parallel substitution of equals for equals is not sufficient to compute the whole congruence relation $=_{\Sigma,E}$.

We give a criterion for an equational theory to be demodulation-complete:

**12.3 Proposition.** If all terms are well-sorted, i.e., $\mathbf{T}_\Sigma = \mathbf{T}_{\overline{\Sigma}}$,
Then $s =_{\Sigma,E} t \Leftrightarrow s \xrightarrow{*} t$, i.e., $\mathcal{E}$ is demodulation-complete.

**Proof.** Using Birkhoffs Theorem, it is sufficient to show that every step of the deductions system in 9.1 can be simulated by steps $s \longrightarrow_{\pi,e,\sigma} t$. The only nontrivial part is to show that $s \xrightarrow{*} t$ implies $\tau s \xrightarrow{*} \tau t$ for every Σ-substitution τ. But obviously $s \longrightarrow_{\pi,e,\sigma} t$ implies $\tau s \longrightarrow_{\pi,e,\sigma} \tau t$, since $s\backslash\pi = \sigma l$ and $t = s[\pi \leftarrow \sigma r]$ imply that $\tau\, s\backslash\pi = \tau\sigma l$ and $\tau s[\pi \leftarrow \tau\sigma r] = \tau(s[\pi \leftarrow \sigma r] = \tau t$. ■

We give more sufficient conditions for demodulation-completeness:

**12.4 Lemma.**

i) Let $\mathcal{E}$ be an equational theory. If for every well-sorted term s and for every term t with $s \xrightarrow{*} t$, the term t is well-sorted, where $\xrightarrow{*}$ is the extended relation on all un-sorted terms, then $\mathcal{E}$ is demodulation-complete.

ii) If $\mathcal{E}$ is deduction-closed, then $\mathcal{E}$ is also demodulation-complete

**Proof.**i) is trivial, since the assumption implies that it is not possible to deduce ill-sorted terms from well-sorted ones by replacement of equals by equals.
ii) trivial. ■

An important way of computing with equations is to direct the equations and to use them as 'simplification' rules. Then $\xrightarrow{*}$ is usually called rewriting or reduction.

$R = (\Sigma, \{s_1 \to t_1, \ldots, s_n \to t_n\})$ with $\mathbf{V}(s_i) \supseteq \mathbf{V}(t_i)$ is called a **term rewriting system (TRS)**.

We say a term s is R-reducible to a $\overline{\Sigma}$-term t ( $s \longrightarrow_R t$) iff $s \longrightarrow_{\pi,e,\sigma,\overline{\Sigma}} t$ for some indices $\pi,e,\sigma$, where e is an oriented equation from R. Note that the default assumption for term rewriting systems is that R-reduction is allowed to produce ill-sorted terms. That means the applicability of a simplification rule does not depend on the well-sortedness of superterms of a term.

We say a term rewriting system is **compatible,** iff for all well-sorted $\Sigma$-terms s : $s \longrightarrow_R t$ implies that t is well-sorted. This means compatible rewriting systems never reduce a well-sorted term to an ill-sorted one. In the following we assume that a term rewriting system is compatible, if not stated otherwise.

We denote the transitive and reflexive closure of $\longrightarrow_R$ by $\xrightarrow{*}_R$ and the symmetric closure of $\xrightarrow{*}_R$ on well-sorted terms by $\xleftrightarrow{*}_R$. A term is **R-irreducible** or in **R-normalform**, iff it is not further reducible.

**12.5 Lemma.** Let R be a (compatible) term rewriting system.

Then $\xleftrightarrow{*}_R$ is a $SUB_\Sigma$-invariant $\Sigma$-congruence and it is the same relation as $=_{\Sigma,E}$ on $\mathbf{T}_\Sigma$.

**Proof.** We prove only that $\xleftrightarrow{*}_R$ is a $\Sigma$-congruence. Obviously it is an equivalence relation. since $\xrightarrow{*}_R$ is $SUB_\Sigma$-invariant we can use induction to prove that $\xleftrightarrow{*}_R$ is also $SUB_\Sigma$-invariant. The congruence property follows from the compatibility of R. ■

An important property of term rewriting systems is confluence: The relation $\xrightarrow{*}_R$ (or R) is **confluent**, iff for all well-sorted terms $s, s_1, s_2$ :

$$s \xrightarrow{*}_R s_1 \text{ and } s \xrightarrow{*}_R s_2 \Rightarrow \exists t \in T_\Sigma : s_1 \xrightarrow{*}_R t \text{ and } s_2 \xrightarrow{*}_R t.$$

In a confluent term rewriting system a term t has a unique normalform, if the process of reducing t terminates. In this case the R-normalform of a term s is denoted by $\| s \|_R$. If every reduction sequence for every term is terminating, then we say R is **terminating** (or **Noetherian**). A term rewriting system is called **canonical,** iff it is confluent and terminating. A canonical term rewriting system R for an equational theory $\mathcal{E}$ provides a decision procedure for equality: To decide $s =_{\Sigma,E} t$, reduce s and t to their R-normalforms $\| s \|_R$ and $\| t \|_R$ and then compare these normalforms for syntactic equality.

For a term rewriting system R, confluence of R is equivalent to the **Church-Rosser property**, i.e., $s =_{\Sigma,E} t$ iff there exists an $r \in \mathbf{T}_\Sigma$ with $s \longrightarrow_R r$ and $t \longrightarrow_R r$. The proof is straightforward by induction on the number of $\xleftrightarrow{*}_R$ -deriviation steps.using $\xleftrightarrow{*}_R = =_{\Sigma,E}$.

For noncompatible term rewriting systems confluence and the Church-Rosser property may be not the same:

**12.6 Example.**

Let $S_\Sigma := \{A,B\}$ with $A \sqsubset B$ and let $a_1, a_2$ be constants of sort A and b be a constant of sort B. Let $f : A \to A$ be a function symbol. Now consider the rewrite system $R := \{a_1 \to b,\ a_2 \to b\}$. This term rewriting system is confluent, since the rewriting relation is deterministic. The terms $f(a_1)$ and $f(a_2)$ are equal, i.e., $f(a_1) =_{\Sigma,E} f(a_2)$, but their reduct f(b) is not well-sorted, hence the relation is not Church-Rosser. ■

A term rewriting system R is called **sort-decreasing**, iff for all $\Sigma$-terms s,t : $s \dashrightarrow_R t$ implies that $S_\Sigma(s) \subseteq S_\Sigma(t)$ (or $LS_\Sigma(s) \sqsupseteq LS_\Sigma(t)$ for regular signatures). This implies that the property holds also for the relation $\xrightarrow{*}_R$.

For sort-decreasing and canonical term rewriting systems we can lift the relation $\xrightarrow{*}_R$ to substitutions and we can use **normalized** substitutions, that means every term in the codomain is in R-normalform.

For term rewriting systems R that are not sort-decreasing it is not possible to lift the reduction to substitutions or to define the normal form of a substitution: Let $s \longrightarrow_R t$ and let $S \in S_\Sigma(s) - S_\Sigma(t)$. Then the substitution $\{x_S \leftarrow s\}$ is well-sorted, but its reduct $\{x_S \leftarrow t\}$ is not. For example, a theory axiomatized by the single equation $\{a = b\}$, where a and b have an uncomparable sort, has no sort-peserving term rewriting system.

The completion procedure of Knuth and Bendix [KB70] is a tool for computing a canonical term rewriting system for a given set of axioms. Since the existence of a canonical term rewriting system implies the decidability of the word problem, there are theories that do not admit a canonical TRS.

We show in II.3 that the confluence test for terminating term rewriting systems using critical pairs and critical sort-relations is a criterion for a sort-decreasing term rewriting system to be canonical. Furthermore if unifier sets w.r.t. the empty theory (together with sorts) are effectively computable (i.e., $(\Sigma,\emptyset)$ is of type finitary) then this test is a decision procedure. For a survey on TRS´s see [HO80, Bu85].

## 13. Manipulating and Solving Equational Systems.

The methods in this paragraph go back to J. Herbrand [Her30], A. Martelli & U. Montanari [MM82] and C. Kirchner [CKi85]. We want to employ these ideas to describe unification as a process that manipulates the original equational system $\Gamma$ by a set of rules. In [MM82, CKi85] a set of multiequations is used instead of an equations system. They consider multiequations of the form $x = y = r = s = t$, denoted by $\{x,y,r,s,t\}$. However, this can be seen as a different representation of the unification problem $\langle x = y,\ y = r, r = s, s = t\rangle$ and the structure of multiequations can be seen as an equivalence relation on the set of equations in an equation system $\Gamma$. For the sake of simplicity we consider equation systems in this paragraph, but all results are also valid for multiequations.

We assume throughout this paragraph that a signature $\Sigma$ and an equational theory $\mathcal{E}$ are given. Recall that the set of E-unifiers of an equational system $\Gamma = \langle s_i = t_i\rangle_E$ is defined as

$$U_{\Sigma,E}(\Gamma) := \{\sigma \in SUB_\Sigma \mid \sigma s_i =_{\Sigma,E} \sigma t_i \text{ for all } i\}.$$

In the following we consider transformations of an equational system $\Gamma_1$ to a system $\Gamma_2$ denoted by $\Gamma_1 \Rightarrow \Gamma_2$. We also consider chains C of such transformations. It is technically important to trace the variables that are used in such a chain C. We assume that all variables introduced by new terms do not occur elsewhere in the chain. As an abbreviation we sometimes call them 'new variables'. In order to make this precise, we assume that every system $\Gamma$ is assigned the set of already **used variables** with respect to the chain C denoted as $UV_C(\Gamma)$. Generally we omit the suffix C and assume it is implicitely given. For the starting equation system we assume that $UV(\Gamma) = \mathbf{V}(\Gamma)$. Furthermore for every transformation step $\Gamma_1 \Rightarrow \Gamma_2$ we assume that $(\mathbf{V}(\Gamma_2) - \mathbf{V}(\Gamma_1)) \cap UV(\Gamma_1) = \emptyset$. That means used variables should not be reintroduced. This is a natural restriction and it allows to compute solutions of an original system $\Gamma$ as the restriction of solutions of a final system to the set of variables in $\mathbf{V}(\Gamma)$. As a consequence we always have $\mathbf{V}(\Gamma) \subseteq UV(\Gamma)$ and $UV(\Gamma_1) \subseteq UV(\Gamma_2)$ for $\Gamma_1 \Rightarrow \Gamma_2$.

**13.1 Definition.** A transformation $\Gamma_1 \Rightarrow \Gamma_2$ is **correct**,

iff $U_{\Sigma,E}(\Gamma_1) \supseteq U_E(\Gamma_2)$.

We say a correct transformation $\Gamma_1 \Rightarrow \Gamma_2$ is **complete,**

iff additionally $U_{\Sigma,E}(\Gamma_1)_{|UV(\Gamma_1)} \subseteq U_{\Sigma,E}(\Gamma_2)_{|UV(\Gamma_1)}$, i.e., $U_{\Sigma,E}(\Gamma_1)_{|UV(\Gamma_1)} = U_{\Sigma,E}(\Gamma_2)_{|UV(\Gamma_1)}$.

We say a set of correct transformations $\Gamma \Rightarrow \Gamma_1,\ldots, \Gamma \Rightarrow \Gamma_n$ is a **complete set of alternatives,** iff $U_{\Sigma,E}(\Gamma)_{|UV(\Gamma)} = U_{\Sigma,E}(\Gamma_1)_{|UV(\Gamma)} \cup \ldots \cup U_{\Sigma,E}(\Gamma_n)_{|UV(\Gamma)}$. ■

**13.2 Lemma.**

i) If $\Gamma_1 \Rightarrow \Gamma_2$ and $\Gamma_2 \Rightarrow \Gamma_3$ are correct , then $\Gamma_1 \Rightarrow \Gamma_3$ is correct.

ii) If $\Gamma_1 \Rightarrow \Gamma_2$ and $\Gamma_2 \Rightarrow \Gamma_3$ are complete, then $\Gamma_1 \Rightarrow \Gamma_3$ is complete.

iii) If $UV(\Gamma_1) = UV(\Gamma_2)$ then:
$\Gamma_1 \Rightarrow \Gamma_2$ is complete, iff $U_{\Sigma,E}(\Gamma_1) = U_{\Sigma,E}(\Gamma_2)$.

iv) $\Gamma_1 \Rightarrow \Gamma_2$ is complete, iff for every substitution $\sigma \in U_{\Sigma,E}(\Gamma_1)$ with $DOM(\sigma) \subseteq UV(\Gamma_1)$ there exists a $\lambda$ with $DOM(\lambda) \subseteq UV(\Gamma_2) - UV(\Gamma_1)$, such that $\sigma \cup \lambda \in U_{\Sigma,E}(\Gamma_2)$.

**Proof.** i) is obvious

ii) From $U_{\Sigma,E}(\Gamma_2)_{|UV(\Gamma 2)} = U_{\Sigma,E}(\Gamma_3)_{|UV(\Gamma 2)}$ and $UV(\Gamma_1) \subseteq UV(\Gamma_2)$ it follows that $U_{\Sigma,E}(\Gamma_2)_{|UV(\Gamma 1)} = U_{\Sigma,E}(\Gamma_3)_{|UV(\Gamma 1)}$, hence $U_{\Sigma,E}(\Gamma_1)_{|UV(\Gamma 1)} = U_{\Sigma,E}(\Gamma_3)_{|UV(\Gamma 1)}$.

iii) is a consequence of i) and ii).

iv) Follows from the definition. ■

Note that 13.2 iii) does not hold in general if $UV(\Gamma_1) \neq UV(\Gamma_2)$:

**13.3 Example.** Let $\Sigma$ be a signature with one sort, let $\mathcal{E}$ be an equational theory and let a,b be two constants that are not E-equal.
Let $\Gamma_1 := \langle x = y \rangle_{\mathcal{E}}$ and $\Gamma_2 := \langle x = z, y = z \rangle_{\mathcal{E}}$. Then $U_{\Sigma,E}(\Gamma_1) = \{\sigma \in SUB_\Sigma \mid \sigma x =_{\Sigma,E} \sigma y\}$ and $U_{\Sigma,E}(\Gamma_2) = \{\sigma \in SUB_\Sigma \mid \sigma x =_{\Sigma,E} \sigma y =_{\Sigma,E} \sigma z\}$. Obviously $\Gamma_1 \Rightarrow \Gamma_2$ is complete and correct, but $U_{\Sigma,E}(\Gamma_1) \neq U_{\Sigma,E}(\Gamma_2)$, since $\{x \leftarrow a,\ y \leftarrow a,\ z \leftarrow b\}$ is in $U_{\Sigma,E}(\Gamma_1) - U_{\Sigma,E}(\Gamma_2)$. ■

**13.4 Lemma.**

i) For all $\sigma \in U_{\Sigma,E}(\Gamma)$, $\tau \in SUB_\Sigma$ and $\sigma \leq_{\Sigma,E} \tau\ [UV(\Gamma)] \Rightarrow \tau \in U_{\Sigma,E}(\Gamma)$.

ii) For all $\sigma \in U_{\Sigma,E}(\Gamma)$, $\tau \in SUB_\Sigma$ and $\sigma \equiv_{\Sigma,E} \tau\ [UV(\Gamma)] \Rightarrow \tau \in U_{\Sigma,E}(\Gamma)$.

iii) $\sigma \in U_{\Sigma,E}(\Gamma)$, $\tau \in SUB_\Sigma$ and $\sigma =_{\Sigma,E} \tau\ [UV(\Gamma)] \Rightarrow \tau \in U_{\Sigma,E}(\Gamma)$.

iv) For every $\sigma \in U_{\Sigma,E}(\Gamma)$, there exists an idempotent substitution $\tau \in SUB_\Sigma$ with $\sigma \equiv_\Sigma \tau\ [UV(\Gamma)]$ and $\tau \in U_{\Sigma,E}(\Gamma)$.

**Proof.**

i) Holds , since $\lambda\sigma =_{\Sigma,E} \tau\ [UV(\Gamma)]$ and $\sigma s =_{\Sigma,E} \sigma t$ implies $\lambda\sigma s =_{\Sigma,E} \lambda\sigma t$.

ii) Follows from i)

iii) Trivial

iv) By Lemma 10.4 there exists an idempotent substitution $\tau \in SUB_\Sigma$ with $\sigma \equiv_\Sigma \tau\ [UV(\Gamma)]$, hence by ii) we have also $\tau \in U_{\Sigma,E}(\Gamma)$. ■

This lemma shows that we can improve the completeness-criterion in Lemma 13.2 iv) to idempotent substitutions:

**Lemma 13.5** $\Gamma_1 \Rightarrow \Gamma_2$ is complete, iff for every idempotent substitution $\sigma \in U_{\Sigma,E}(\Gamma_1)$ with $DOM(\sigma) \subseteq UV(\Gamma_1)$ there exists a $\lambda$ with $DOM(\lambda) \subseteq UV(\Gamma_2) - UV(\Gamma_1)$, such that $\sigma \cup \lambda \in U_{\Sigma,E}(\Gamma_2)$. ■

The conjunction (or the union) of two equational systems $\Gamma_1$ and $\Gamma_2$ is denoted as $\Gamma_1 \& \Gamma_2$. Obviously we have $U_{\Sigma,E}(\Gamma_1 \& \Gamma_2) = U_{\Sigma,E}(\Gamma_1) \cap U_{\Sigma,E}(\Gamma_2)$. In Lemma 13.8 we show that local completeness can be lifted to a conjunction.
The following set of rules is applicable to every equation system and every equational theory.

**Demodulation Rule.**

$$s = t \,\&\, \Gamma \Rightarrow s' = t \,\&\, \Gamma$$

If $s =_{\Sigma,E} s'$ and $(\mathbf{V}(s) - \mathbf{V}(s')) \cap UV(\Gamma) = \emptyset$.

**Trivial Equation Rule.**

$$s = t \,\&\, \Gamma \Rightarrow \Gamma$$

If $s =_{\Sigma,E} t$.

**Binding Rule.**

$$x = t \,\&\, \Gamma \Rightarrow x = t \,\&\, \{x \leftarrow t\}\Gamma$$

If $\{x \leftarrow t\}$ is a well-sorted substitution.

**Internal Demodulation.**

$$s = t \,\&\, \Gamma \Rightarrow s = t \,\&\, \Gamma'$$

where $\Gamma'$ is obtained from $\Gamma$ by replacing some subterm s by t.

These rules can be used to simplify equational systems $\Gamma$. For example, it is possible to delete equations that have the solution Id. Internal demodulation has a nice application as a general simplification rule for unification problems. Consider for example the AC-unification problem $\langle xx = ya, xxc = yb\rangle_{AC}$. This problem can be transformed into $\langle xx = ya, yac = yb\rangle_{AC}$ and then by cancellation rules into $\langle xx = ya, ac = b\rangle_{AC}$, which is unsolvable.

**13.6 Proposition.**

i) The demodulation rule is complete.
ii) The trivial-equation rule is complete
iii) The binding rule is complete.
iv) The internal demodulation rule is complete.

**Proof.** i) and ii) are obviously true.

iii) Let $\sigma \in U_{\Sigma,E}(x = t \,\&\, \Gamma)$. Then $\sigma x =_{\Sigma,E} \sigma t$, hence $\sigma\{x \leftarrow t\} =_{\Sigma,E} \sigma$ . This means $\sigma \in U_{\Sigma,E}(x = t \,\&\, \{x \leftarrow t\}\Gamma)$.

To prove the converse, let $\sigma \in U_{\Sigma,E}(x = t \,\&\, \{x \leftarrow t\}\Gamma)$, then again $\sigma x =_{\Sigma,E} \sigma t$, hence $\sigma\{x \leftarrow t\} =_{\Sigma,E} \sigma$ and $\sigma \in U_{\Sigma,E}(x = t \,\&\, \Gamma)$.

iv) Let $\sigma \in U_{\Sigma,E}(s = t \,\&\, \Gamma)$ and assume that $u = v \in \Gamma$ , $u/\pi = s$ , $u' = u[\pi \leftarrow t]$ and $\Gamma'$ is obtained from $\Gamma$ by this replacement. Then $\sigma u =_{\Sigma,E} \sigma v$ and $\sigma s =_{\Sigma,E} \sigma t$ implies $\sigma u' =_{\Sigma,E} \sigma v$, hence $\sigma \in U_{\Sigma,E}(s = t \,\&\, \Gamma')$. The converse is a symmetric case. ■

In order to design transformation rules it may be helpful to know for some special cases of terms how to obtain their complete set of unifiers. The same type of problem arises in combining known unification procedures with a set of transformation rules. The idea is to replace the unified equation by the pairs $x = \sigma x$ for a unifier $\sigma$. We denote the equation system obtained from $\sigma$ in this way by $\langle\sigma\rangle_E$ or as $\langle\sigma\rangle$ for short. This result is also known as the inheritance theorem in [Oh87]. This proposition can be applied to minimal sets of unifiers and shows then that we can sequentialize the computation of minimal sets of unifiers: In order to solve $\Gamma_1 \& \Gamma_2$ we first compute a minimal set of unifiers for $\Gamma_1$, apply the obtained substitutions to $\Gamma_2$, and solve the obtained system. The conditions on variables means that the variables in $\Gamma_2$ that are not in $\Gamma_1$ should not be used in the codomain of minimal unifiers of $\Gamma_1$.

**13.7 Proposition.** Let $\Gamma_1$ and $\Gamma_2$ be two unification problems and let U be a complete set of idempotent E-unifiers for $\Gamma_1$ (modulo the set of variables $V(\Gamma_1)$ ), such that $DOM(\sigma) \subseteq V(\Gamma_1)$ and $I(\sigma) \cap UV(\Gamma_1 \& \Gamma_2) \subseteq V(\Gamma_1)$ for all $\sigma \in U$.

Then the rule:

$$\Gamma_1 \& \Gamma_2 \quad \Rightarrow \langle\sigma\rangle \,\&\, \Gamma_2 \qquad \text{for } \sigma \in U$$

provides a correct and complete set of alternatives.

**Proof.** i) Correctness: Let $\tau \in U_{\Sigma,E}(\langle\sigma\rangle \,\&\, \Gamma_2)$. Then we have $\tau x =_{\Sigma,E} \tau\sigma x \; [V(\Gamma_1)]$. Hence $\sigma \leq_{\Sigma,E} \tau \; [V(\Gamma_1)]$ , which implies $\tau \in U_{\Sigma,E}(\Gamma_1)$, hence $\tau$ is a solution of $\Gamma_1 \& \Gamma_2$.

ii) Completeness: Let $\tau \in U_{\Sigma,E}(\Gamma_1 \& \Gamma_2)$ with $DOM(\tau) \subseteq UV(\Gamma_1 \& \Gamma_2)$. Then there exists a $\sigma \in U$, such that $\sigma \leq_{\Sigma,E} \tau \; [V(\Gamma_1)]$ and hence there exists a $\lambda$ with $DOM(\lambda) \subseteq I(\sigma) \cup V(\Gamma_1)$ such that $\lambda\sigma =_{\Sigma,E} \tau \; [V(\Gamma_1)]$.

We have to show that there exists $\tau'$ with $\tau' =_{\Sigma,E} \tau \; [UV(\Gamma_1 \& \Gamma_2)]$ such that $\tau' x =_{\Sigma,E} \tau'\sigma x$ for all $x \in V(\Gamma_1)$.

Let $W := I(\sigma) - V(\Gamma_1)$ and let $\tau' := \tau \cup \lambda_{|W}$. Note that $DOM(\tau) \cap W = \emptyset$.

We have $\lambda\sigma =_{\Sigma,E} \tau' \; [\, I(\sigma) \cup V(\Gamma_1) \,]$: For $x \in V(\Gamma_1)$ this is true by assumption.

For $y \in I(\sigma) - V(\Gamma_1)$, we have $\lambda\sigma y = \lambda y$, since $\sigma$ is idempotent and $\tau' y = \lambda y$.

Now consider $\tau'\sigma x$ for all $x \in V(\Gamma_1)$. Obviously $V(\sigma x) \subseteq I(\sigma) \cup V(\Gamma_1)$. Hence $\tau'\sigma x =_{\Sigma,E} \lambda\sigma\sigma = \lambda\sigma = \tau = \tau' \; [V(\Gamma_1)]$ . ■

The idempotency of unifiers is necessary:
Consider the equation system $\{x = f(y)\}$. Then $\{x \leftarrow f(x), y \leftarrow x\}$ is a most general unifier for $\Gamma$, but the system $\{x = f(x), y = x\}$ is unsolvable.

We can localize the test for correctness and completeness of $\Gamma_1 \Rightarrow \Gamma_2$ on the parts that are different.

**13.8 Proposition.** Let $\Gamma$ be an equational system. Then
i) If $\Gamma_1 \Rightarrow \Gamma_2$ is correct, then $\Gamma\&\Gamma_1 \Rightarrow \Gamma\&\Gamma_2$ is correct.
ii) If $\Gamma_1 \Rightarrow \Gamma_2$ is complete, then $\Gamma\&\Gamma_1 \Rightarrow \Gamma\&\Gamma_2$ is complete.

**Proof.**
i) From $U_{\Sigma,E}(\Gamma_1) \supseteq U_{\Sigma,E}(\Gamma_2)$ we conclude $U_{\Sigma,E}(\Gamma) \cap U_{\Sigma,E}(\Gamma_1) \supseteq U_{\Sigma,E}(\Gamma) \cap U_{\Sigma,E}(\Gamma_2)$.
ii) Let $U_{\Sigma,E}(\Gamma_1)_{|UV(\Gamma 1)} = U_{\Sigma,E}(\Gamma_2)_{|UV(\Gamma 1)}$. Then $(U_{\Sigma,E}(\Gamma) \cap U_{\Sigma,E}(\Gamma_1))_{|UV(\Gamma 1)}$ $= (U_{\Sigma,E}(\Gamma) \cap U_{\Sigma,E}(\Gamma_2))_{|UV(\Gamma 1)}$. ■

In parts III and IV we investigate unification procedures defined by rules in a set RS that transform equational systems . The transformations specified by such rules are in general nondeterministic. We denote the corresponding transitive, reflexive relation on equational systems by $\xrightarrow{*}_{RS}$. We denote the unsolvable system with the sign ✱, i.e, we have always $U_{\Sigma,E}(✱) = \emptyset$.

**13.9 Definition.** We say a system $\Gamma$ is **solved**, iff $\Gamma = \{x_i = t_i \mid i = 1,\ldots,n\}$, all $x_i$ are distinct, $\{x_1,\ldots,x_n\} \cap \mathbf{V}(t_i) = \emptyset$ and $\mathbf{LS}_\Sigma(x_i) \in \mathbf{S}_\Sigma(t_i)$ for all $i = 1,\ldots,n$. The corresponding solution $\sigma_\Gamma$ is the substitution $\{x_i \leftarrow t_i \mid i = 1,\ldots,n\}$. ■

Note that the substitution $\sigma_\Gamma$ is always idempotent for solved systems.

**13.10 Definition.** We say a rule-system RS is a **complete unification procedure**, iff for every system $\Gamma$ and every substitution $\sigma \in U_{\Sigma,E}(\Gamma)$ there exists a system $\Delta$ with $\Gamma \xrightarrow{*}_{RS} \Delta$ and $\sigma \geq_{\Sigma,E} \sigma_\Delta\ [UV(\Gamma)]$. ■

Note that a set of rules that allows only complete transformation steps is not necessarily a complete unification procedure: For example if there are no rules at all, then every transformation is complete, but not every equation system is in solved form.

We have as a first trivial lemma that solved equation systems have the right solution and are unitary solvable.

**13.11 Lemma.** Let $\Gamma$ be a solved equational system. Then

i) $\sigma_\Gamma \in U_{\Sigma,E}(\Gamma)$.

ii) for all $\sigma \in U_{\Sigma,E}(\Gamma)$: $\sigma \geq_{\Sigma,E} \sigma_\Gamma$ [UV($\Gamma$)].

**Proof.** i) is trivial.

ii) Let $\Gamma = \{x_i = t_i \mid i = 1,\ldots,n\}$ and let $\sigma \in U_{\Sigma,E}(\Gamma)$. Then we have $\sigma x_i =_{\Sigma,E} \sigma t_i$ for all i. We show $\sigma =_{\Sigma,E} \sigma\sigma_\Gamma$ [UV($\Gamma$)]: For all i we have $\sigma\sigma_\Gamma x_i = \sigma t_i =_{\Sigma,E} \sigma x_i$. For $y \in V(t_i)$ we have $\sigma\sigma_\Gamma y = \sigma y$. ■

**13.12 Lemma.** Let $\Gamma$ be an equational system and let $\Delta$ be a solved equation system obtained by correct transformation steps .

Then we have $\sigma_\Delta \in U_{\Sigma,E}(\Gamma)$. ■

If all transformation steps in a rule system are complete, then all solutions are equivalent, that means it is sufficient to compute just one solution:

**13.13 Lemma.** Let $\Gamma$ be an equational system and let $\Delta$ be a solved equation system obtained by complete transformation steps .

Then for every $\sigma \in U_{\Sigma,E}(\Gamma)$ we have $\sigma \geq_{\Sigma,E} \sigma_\Delta$ [V($\Gamma$)]:

**Proof.** From completeness we obtain $U_{\Sigma,E}(\Gamma)_{|UV(\Gamma)} = U_{\Sigma,E}(\Delta)_{|UV(\Gamma)}$. For $\sigma \in U_{\Sigma,E}(\Gamma)$ with DOM($\sigma$) $\subseteq$ UV($\Gamma$) there exists a substitution $\lambda$ with $\sigma\cup\lambda \in U_{\Sigma,E}(\Delta)$ by Lemma 13.2 iv). Lemma 13.11 shows that $\sigma\cup\lambda \geq_{\Sigma,E} \sigma_\Delta$ [UV($\Delta$)] . Since UV($\Delta$) $\supseteq$ UV($\Gamma$) and $\sigma = \sigma\cup\lambda$ [UV($\Gamma$)] we conclude $\sigma \geq_{\Sigma,E} \sigma_\Delta$ [UV($\Gamma$)]. ■

If we start with an equation system $\Gamma$ and use only complete transformation steps, then all obtained solutions are equivalent.

**13.14 Lemma.** Let $\Gamma$ be an equational system and let $\Gamma_1$ and $\Gamma_2$ be two solved equation systems obtained by complete transformation steps .

Then $\sigma_{\Gamma 1} \equiv_{\Sigma,E} \sigma_{\Gamma 2}$ [UV($\Gamma$)]:

**Proof.** From Lemma 13.12 we obtain $\sigma_{\Gamma 1}, \sigma_{\Gamma 2} \in U_{\Sigma,E}(\Gamma)$. Lemma 13.13 shows $\sigma_{\Gamma 1} \geq_{\Sigma,E} \sigma_{\Gamma 2}$ [UV($\Gamma$)] and $\sigma_{\Gamma 2} \geq_{\Sigma,E} \sigma_{\Gamma 1}$ [UV($\Gamma$)], hence $\sigma_{\Gamma 1} \equiv_{\Sigma,E} \sigma_{\Gamma 2}$ [UV($\Gamma$)]. ■

Every equation system can be partitioned into the parts: $\Gamma = \Gamma_S \cup \Gamma_U$, where

i) $\Gamma_S$ is the solved part, that is the set of equations of the form x = t, such that $x \notin \Gamma - \{x{=}t\}$ and $\mathbf{LS}_\Sigma(x) \in \mathbf{S}_\Sigma(t)$.

ii) $\Gamma_U = \Gamma - \Gamma_S$ is the unsolved part.

We can further partition $\Gamma_U$ into $\Gamma_{QS} \cup \Gamma_{QU}$, where

i) $\Gamma_{QS}$ is the quasi-solved part, that is the set of equations of the form x = t, such that

$x \notin \Gamma-\{x=t\}$ and $\mathbf{LS}_\Sigma(t) \nsubseteq \mathbf{LS}_\Sigma(x)$.

ii) $\Gamma_{QU} = \Gamma - \Gamma_{QS}$ is the quasi-unsolved part.

In order to obtain a deterministic procedure from a nondeterministic rule system, we take sets of equational systems that represent all solutions. The set of solutions of such a set $\{\Gamma_1,\ldots, \Gamma_n\}$ is the set $U_{\Sigma,E}(\Gamma_1) \cup \ldots \cup U_{\Sigma,E}(\Gamma_n)$. The tranformation rules lift to these sets as follows:

i) If $\Gamma_1 \Rightarrow \Gamma_1'$ is a complete step, then we transform $\{\Gamma_1,\ldots, \Gamma_n\}$ into $\{\Gamma_1',\ldots, \Gamma_n\}$.

ii) If the transformations $\Gamma_1 \Rightarrow \Gamma_{11},\ldots, \Gamma_1 \Rightarrow \Gamma_{1m}$ are a complete set of alternatives, i.e. $U_{\Sigma,E}(\Gamma_1) = U_{\Sigma,E}(\Gamma_{11}) \cup \ldots \cup U_{\Sigma,E}(\Gamma_{1m})$, then we transform

$$\{\Gamma_1, \Gamma_2,\ldots, \Gamma_n\} \text{ into } \{\Gamma_{11},\ldots, \Gamma_{1m}, \Gamma_2,\ldots, \Gamma_n\}.$$

iii) A rule $\Gamma_1 \Rightarrow *$ translates into $\{\Gamma_1, \Gamma_2,\ldots, \Gamma_n\} \Rightarrow \{\Gamma_2,\ldots, \Gamma_n\}$.

## 14. Comparison of Different Appraoches to Unification.

In a deduction system equations have to be unified (or solved) in order to compute the most general unifiers for the resolution steps. Without built-in equations, this is just ordinary unification [Her30, Ro65] and with built-in equations this is called E-unification [Plo72, Si86].

In all these approaches, unification can be seen as solving equations over the free algebra of terms modulo an equational theory, the solutions are substitutions and subsumption is defined in terms of a composition of substitutions. The Herbrand-Theorem [CL73] states that for every unsatisfiable clause set there exists a finite and unsatisfiable set of ground instances of clauses. Hence a resolution-based automated deduction system (cf. Part V) remains a complete proof procedure, if instead of all unifiers only ground unifiers are used for the resolution steps. This obversation could have an impact on the unification algorithm since now only ground solutions have to be represented (instead of all solutions) and perhaps the notion of a most general unifier could be modified.

In this paragraph we compare these two methods of unification. Comparison also shows more explicitly the connection between E-unification and solving polynomial equations over integers or rationals.

In effect, this paragraph is more a justification of the usual unification definitions than their refusal. The advantages of the usual definition are that most general unifier sets remain invariant if the theory is disjointly combined with another theory. This means unification behaves context independent. This property does not hold for the definitions with respect to ground terms as we shall see. However, in the case where a model or an algebra is fixed (for example solving polynomials over rationals), the ground solution approach may be more natural.

Solving equations containing unknowns (or variables ) requires an exact specification of the signature since this determines what can be substituted for the unknowns, i.e., an exact declaration of the algebra is required. We consider three different possibilities:

i) Free algebras

ii) Initial algebras

iii) Some fixed algebra (or model of the equational theory)

A second problem is the representation of the solutions as well as the definition of subsumption, of the most general solutions and of complete sets of solutions.

An example for the free algebra solution method is Robinson's unification approach for the empty theory (cf part III).

We give some introductory examples for solving equations in the initial algebra :

**14.1 Example.**

a) Let the natural numbers be specified with constructors 0 and succ and let the problem to be solved be $\langle succ(x) = succ(y)\rangle$. In the initial algebra there are infinitely many solutions: $\{x \leftarrow 0, y \leftarrow 0\}$ , $\{x \leftarrow succ(0), y \leftarrow succ(0)\}$, ...
   As a most general solution we would take $\{x \leftarrow z, y \leftarrow z\}$, since every instantiation of a ground term for z results in a solution for the original equation.

b) If we specify the addition on natural numbers by the equations
   $x + 0 = x$
   $x + succ(y) = succ(x + y)$
   Then addition is commutative on the initial algebra, but not on the free algebra, since the terms x+y and y+x are not equal modulo this theory.
   Hence the equation $\langle x + y = y + x\rangle$ has Id as most general solution in the initial algebra, but not in the free algebra.

**14.2 Example.** An example for solving equations in an explicitely given algebra are the following linear equations over the algebra of rational numbers without division:
The solution of $\langle 3x + 4y = 0\rangle$ is $\{x \leftarrow 4z, y \leftarrow 3z\}$, where z ranges over all real numbers.
The solution of $\langle 3x = 4\rangle$ is $\{ x \leftarrow 4/3\}$. ■

The solution process for free algebras is exactly that defined in paragraph I.11. We will call this method of free solving, F-solving, and refer to these unifiers as F-solutions. Furthermore we denote E-equality and subsumption by the symbols $=_{F,\Sigma,E}$ and $\leq_{F,\Sigma,E}$, respectively.

Now we define more precisely what we mean by initial equality and initial solving of equations, where we assume a specification $\mathcal{S} = (\Sigma, \mathcal{E})$ given and an equation system $\Gamma$ that has to be solved.

**14.3 Definition.**

i) Two terms s,t are **I-equal** ($s =_{I,\Sigma,E} t$) iff for every ground substitution $\lambda$ with $DOM(\lambda) = V(s,t)$, we have $\lambda s =_{\Sigma,E} \lambda t$.

ii) An I-**solution** $\sigma$ is a substitution $\sigma$ that I-solves $\Gamma$, i.e. $\sigma s =_{I,\Sigma,E} \sigma t$ for all equations $s = t$ in $\Gamma$.

iii) We compare two I-solutions $\sigma$ and $\tau$ with a strong subsumption ordering as follows:

$\sigma \leq_{sI,\Sigma,E} \tau\, [V(\Gamma)]$ iff there exists a substitution $\lambda$ such that $\lambda\sigma =_{I,\Sigma,E} \tau\, [V(\Gamma)]$.

We say $\sigma$ **strongly I-subsumes** $\tau$

In the same way as in I.11 we can define the strong I-unification type of an equational theory.

iv) We compare two I-solutions $\sigma$ and $\tau$ with a weak subsumption ordering as follows:

$\sigma \leq_{wI,\Sigma,E} \tau\, [V(\Gamma)]$ ($\sigma$ **weakly I-subsumes** $\tau$ )

iff every ground instance $\tau_{gr}$ of $\tau$ is also a ground instance of $\sigma$ (modulo the set $V(\Gamma)$).

In the same way as in I.11 we can define the weak I-unification type of an equational theory.■

Solving equation systems with respect to a predefined algebra can be simulated by I-solving, if the signature contains the usual function symbols and additionally all elements of the algebra as constants and the (initial) equational theory contains all the equations in the multiplication table of the algebra. The equational theory may also be chosen as the theory generated from the initial algebra, i.e. that $=_{I,\Sigma,E}$ is the same relation as $=_{F,\Sigma,E}$. The problem then is that in general induction is necessary to prove the validity of equations (cf. Example 14.1) and that the generated theory has no finite axiomatization.

An obvious fact is:

**14.4 Lemma.** $\sigma \leq_{F,\Sigma,E} \tau\, [V(\Gamma)] \Rightarrow \sigma \leq_{sI,\Sigma,E} \tau\, [V(\Gamma)] \Rightarrow \sigma \leq_{wI,\Sigma,E} \tau\, [V(\Gamma)]$ ■

**14.5 Example.** The theory of free bands, (associativity and idempotency) is an example where the F-unification type differs from the weak I-unification type. We assume that the signature contains only the associative and idempotent function symbol and finitely many free constants. F-unification is of type zero [Ba86, Sch86], whereas weak I-unification is of type finitary, since finitely generated bands are finite [Ho76] and hence there are only finitely many ground I-unifiers. This means every properly (weak) descending chain of I-solutions is finite.

In the case where only one constant is present, the unification type switches from F-zero to weak I-unitary and to strong I-unitary. ■

We prove that all subsumption relations are identical if infinitely many free constants are available:

**14.6 Theorem.** If the specification contains infinitely many free constants then for all substitutions $\sigma,\tau$:

$$\sigma \leq_{F,\Sigma,E} \tau\,[W] \;\Leftrightarrow\; \sigma \leq_{wI,\Sigma,E} \tau\,[W] \;\Leftrightarrow\; \sigma \leq_{sI,\Sigma,E} \tau\,[W] \text{ where } W = V(\Gamma).$$

**Proof.** Due to Lemma 14.4 it is sufficient to prove $\sigma \leq_{wI,\Sigma,E} \tau\,[W] \Rightarrow \sigma \leq_{F,\Sigma,E} \tau\,[W]$ :
Without loss of generality we can assume that $DOM(\sigma) = DOM(\tau) \subseteq V(\Gamma)$ and that $I(\sigma) \cap I(\tau) = \emptyset$.
Let $x_1,\ldots,x_n$ be the variables in $V(\tau W)$ and let $a_1,\ldots,a_n$ be constants not occurring as subterms in the terms of $COD(\sigma) \cup COD(\tau)$. Then $\tau_{gr} := \{x_i \leftarrow a_i\}\tau$ is a ground instance of $\tau$. Let $y_1,\ldots,y_m$ be the variables in $V(\sigma W)$. There exist constants $b_j$, $j=1,\ldots,m$ such that $\{y_j \leftarrow b_j \mid j=1,\ldots,m\}\sigma =_{\Sigma,E} \{x_i \leftarrow a_i \mid i = 1,\ldots,n\}\tau\ [W]$. Since $a_i$ are free constants, the above equation remains valid, if the $a_i$ 's in $\{y_j \leftarrow b_j\}$ and $\{x_i \leftarrow a_i\}$ replaced by new variables $z_i$. Hence $\lambda\sigma =_{\Sigma,E}\{x_i \leftarrow z_i\}\tau\ [W]$, where $\lambda$ is the substitution obtained from replacing $a_i$ by $z_i$ in the codomain of $\{y_j \leftarrow b_j\}$. Applying the converse substitution $\{z_i \leftarrow x_i\}$ gives $\{z_i \leftarrow x_i\}\,\lambda\sigma =_{\Sigma,E} \{z_i \leftarrow x_i\}\{x_i \leftarrow z_i\}\tau =_{\Sigma,E} \tau\ [W]$. This immediately implies $\sigma \leq_{F,\Sigma,E} \tau\,[W]$ . ■

**14.7 Proposition.** If the specification contains infinitely many free constants then for all equation systems $\Gamma$ : $\sigma \in U_{I,\Sigma,E}(\Gamma) \Leftrightarrow \sigma \in U_{F,\Sigma,E}(\Gamma)$.

**Proof.** The proof argues similar to the proof of the above theorem: replace variables in $I(\sigma)$ by new constants.

In a special case we can generalize Theorem 14.6:

**14.8 Theorem.** If the specification contains a free constant c and a nonconstant free function symbol g, then for all substitutions $\sigma,\tau$ that do not have g or c in their codomain terms, the following holds:

$$\sigma \leq_{F,\Sigma,E} \tau\,[W] \;\Leftrightarrow\; \sigma \leq_{wI,\Sigma,E} \tau\,[W] \;\Leftrightarrow\; \sigma \leq_{sI,\Sigma,E} \tau\,[W] \text{ where } W = V(\Gamma).$$

**Proof.** The proof proceeds like the proof of Theoprem 14.7 except that instead of new constants we use ground terms built from g and c . These terms behave like infinitely many free constants, since g and c are not used in $COD(\sigma)$ and $COD(\tau)$. ■

In Theorem 14.8 it is not possible to drop the condition on $\sigma$ and $\tau$:

**14.9 Example**. Let f be a binary function symbol that is idempotent in the initial algebra, but not in the free algebra, i.e. $f(t\ t) = t$ for all gound terms t, but $f(t\ t) \neq t$ for all nonground terms t. Assume that a is the only free constant. Furthermore let g be a unary free function symbol.

Then we have $f(g(x)\ y) \leq_{sI,\Sigma,E} g(z)\ [W]$, since on the ground terms we have $\{x \leftarrow z, y \leftarrow g(z)\}f(g(x)\ y) =_{I,\Sigma,E} g(z)\ [W]$, but obviously not $f(g(x), y) \leq_{F,\Sigma,E} g(z)\ [W]$. ■

**14.10 Proposition.** If the specification contains a free constant c and a nonconstant free function symbol g then for all equation systems $\Gamma$, which do not contain g or c as symbols: $\sigma \in U_{I,\Sigma,E}(\Gamma) \Leftrightarrow \sigma \in U_{F,\Sigma,E}(\Gamma)$.

**Proof.** The proof argues similar to the proof of the Theorem 14.6. ■

Together we have the theorems:

**14.11 Theorem:** If the specification contains infinitely many free constants, then the weak I-unification type, strong I-unification type and the F-unification type of an equational theory $\mathcal{E}$ are the same. ■

**14.12 Theorem:** If the specification contains a free constant c and a nonconstant free function symbol g then the unification types of $U_{wI,\Sigma,E}(\Gamma)$, $U_{sI,\Sigma,E}(\Gamma)$ and $U_{F,\Sigma,E}(\Gamma)$ are the same for all equation systems $\Gamma$ that do not contain g or c. ■

These two theorems justify the use of free unification in Automated Deduction systems: If I-unification and I-minimization is used, then the results (the set of unifiers) depend on the context. For example if it is possible to invent new constants or to have Skolem-functions, then I-unification has no advantage over F-unification.

In the following we investigate properties of equational theories with a generic [Gr79] initial algebra.

**14.13 Definition.**: An equational theory $\mathcal{E}$ is **initial-generic**, iff for all terms s,t :

$$s =_{I,\Sigma,E} t \Leftrightarrow s =_{F,\Sigma,E} t. \quad ■$$

That means that equality of two terms can be tested on their ground instances.

$$s =_{\Sigma,E} t \Leftrightarrow \forall\lambda \in SUB_{\Sigma,gr}\ \lambda s =_{\Sigma,E} \lambda t.$$

Examples for theories that are always initial-generic are those that are generated by their initial algebras. Note, however, that in general initial-generic does not imply that a theory is generated by the initial algebra.

There are two ways to modify an equational theory in order to make it initial-generic:
i) add free constants
ii) define a new equational theory $\mathcal{E}'$ with $s =_{\Sigma,E'} t$ iff $s =_{I,\Sigma,E} t$ (consider the theory generated by the initial algebra)

**14.14 Lemma.** In initial-generic theories, I-solutions and F-solutions are the same.
**Proof.** Obvious.

**14.15 Example.** There are theories, where infinitely many free constants have to be added to make them initial-generic:
Consider the theory with one binary function symbol f, a constant 0 and let f be associative, commutative and assume the following additional equations hold:
$f(x\ 0) = 0$, $f(x\ x) = 0$.
If we write terms as strings, it is easy to see, that either a string is E-equal to 0, or it is of the form $x_1x_2...x_n$, where all $x_i$ are different.
Furthermore two nonzero strings $x_1x_2...x_n$ and $y_1y_2...y_n$ are E-equal, iff $x_1,...,x_m$ is a permutation of $y_1,...,y_n$.
The addition of a finite number k of free constants is not sufficient to make the theory initial-generic, since a ground instance of a nonzero string that has more than k variables is E-equal to zero. ■

**14.16 Example.** The empty theory is initial-generic, if there are at least two ground terms.

**14.17 Lemma.** In initial-generic theories, the notion of sI-subsumption and F-subsumption is the same, furthermore the sI-type and the F-type are the same.
**Proof.** The first statement is obvious. The second follows with Lemma 14.14 . ■

The next example demonstrates that sI-subsumption and wI-subsumption are different.
**14.18 Example.**
i) The theory INT of integers (as ring) is initial-generic:
Polynomials over the integers are equal, if all their ground instances (under the same ground substitution) are equal.
ii) wI-subsumption and F-subsumption in INT are different:
The polynomial $p := x_1^2 + x_2^2 + x_3^2 + x_4^2 - x_5^2 - x_6^2 - x_7^2 - x_8^2$ has all integers as

ground instances, since it is well-known that every integer is the sum of four squares of integers, hence it is wI-equivalent to a variable. However, the polynomial p is not F-equivalent to a variable, since there is no instantiation of p, such that the value is a variable. ■

A related problem to Example 14.18 is the open problem of the unification type of Hiberts $10^{th}$ problem, i.e. of the F-unification type of integer equation solving. For this problem it would be equally interesting to determine its wI-unification type.

We should also mention a Theorem in [Ti86], that most general F-unifiers sets are invariant if a disjoint theory is added:

**14.19 Theorem** [Ti86]. Let $E = E_1 \cup E_2$ be a disjoint combination of two equational theories. Furthermore let $\Gamma$ be an equation system that does not contain symbols from $E_2$. Then a complete set $cU_{E1}(\Gamma)$ of F-unifiers is also a complete set $cU_E(\Gamma)$ of F-unifiers with respect to the combination. ■

Example 14.5 shows that this is not true for sets of I-unifiers, since the addition of constants can be seen as a disjoint combination of theories.

# Part II.
# Various Extensions

**Overview:** This part extends the first part on foundations in several aspects:

The extension of semantics and deduction to ill-sorted terms is investigated and it is shown, that a deduction system remains correct, if it is allowed to deduce ill-sorted terms.

The combination of sorts and term rewriting systems is studied and a criterion is given for canonical term rewriting systems, which is an extension of the usual critical pair criterion by a critical sort relation criterion. A completion procedure for ground TRS is given.

We have a closer look on the properties deduction-closedness, congruence-closedness and sort-preservation and give criteria for checking them as well as results about the decidability of these properties.

Conservative transformations of signatures and specifications are studied in detail in paragraph 7.

We give different methods to construct unsorted (relativized) specifications from sorted ones.

The logic is extended to full first order predicate calculus and a method for skolemization in a sorted signature is given.

## 1. Extension to Ill-Sorted Terms.

The aim of this paragraph is to investigate extensions of sorted calculi to ill-sorted terms and atoms. If equations are absent, then the usual deduction methods do not derive ill-sorted formulae from well-sorted ones, whereas in the presence of equations a deductive system may produce ill-sorted terms by replacing equals for equals. In general such deduction steps are forbidden, since all terms have to be well-sorted. We show by semantical means that every model for some specification can be extended to a model, where ill-sorted terms have a denotation in the model. This has as consequence, that deduction with intermediate ill-sorted terms or atoms, but with the same set of well-sorted substitutions, is sound. In the next paragraph it is demonstrated that clause sets consisting only of sorted equations behave very similar to the unsorted case, if one allows unsorted terms during the deduction.

The consequence of Theorem 1.1 is that for equational deduction we can assume that all terms are well-sorted by adding a sort IT (ill-sorted terms) and leaving all other sorts of terms invariant. Furthermore the set of (well-sorted) theorems derived by equational deductions with intermediate ill-sorted terms is exactly the same as that derived by deduction with intermediate well-sorted terms.

A similar situation arises in lifting congruences in a partial algebra [Gr79] to all elements of the algebra. In the Rewrite Theorem in [Wa83] it is proved for simple signatures that well-sorted equations obtained by non well-sorted equational deductions can always be obtained by a well-sorted deduction.

Let $\mathcal{S} := (\Sigma, CS)$ be a specification. We construct the **ill-sorted extension** as follows:

Let $\Theta$ be the signature with the same function and sort symbols as $\Sigma$, but with an additional top-sort IT (ill-sorted terms). I.e., $\mathbf{S}_\Theta = \mathbf{S}_\Sigma \cup \{IT\}$, $\mathbf{F}_\Theta = \mathbf{F}_\Sigma$, $T_\Theta = T_{\bar{\Theta}}$, and the sort of $\Sigma$-terms in $\Theta$ is the same as in $\Sigma$, the sort of ill-sorted $\Sigma$-terms is IT, all atoms are well-sorted, and CS is unchanged.

This can be performed by adding the following things to a signature: a top-sort IT, the function declarations $f: IT \times \ldots \times IT \rightarrow IT$ for every nonconstant function symbol, and by replacing all sorts in predicate declarations by IT. The definiton of signature requires that there are also infinitely many variables of sort IT. It is easy to see that every $\Theta$-term t that has sort less IT is also a $\Sigma$-term. Furthermore every substitution component of a well-sorted substitution in $\Theta$ is either also well-sorted in $\Sigma$ or it is of the form $\{x \leftarrow t\}$, where x is of sort IT.

**1.1 Theorem.** Let $\mathcal{S} := (\Sigma, CS)$ be a specification and let $\Theta$ be the ill-sorted extension of $\Sigma$.

Then CS has a $\Sigma$-model iff it has a $\Theta$-model.

**Proof.** If CS has a $\Theta$-model then it obviously has a $\Sigma$-model.

Let CS have a $\Sigma$-model $\mathcal{A}$. Then we recursively construct a $\Theta$-model $\mathcal{B}$ from $\mathcal{A}$ as follows:

i) $A \subseteq B$.

ii) If $(a_1,\ldots,a_n) \notin \mathcal{D}(f_A)$, then we add the expression $f(a_1,\ldots,a_n)$ to B.

This construction gives a B, such that $\mathcal{D}(f_B) = B^n$ for all $f \in \mathbf{F}_n$.

As denotation for sorts we choose $S_B := S_A$, if $S \in \mathbf{S}_\Sigma$ and $IT_B := B$. As denotation for a function f we define $f_B(b_1,\ldots,b_n) := f_A(b_1,\ldots,b_n)$, if $(a_1,\ldots,a_n) \in \mathcal{D}(f_A)$, and $f_B(b_1,\ldots,b_n) := f(b_1,\ldots,b_n)$ otherwise. For predicates P we define $P_B := P_A$, if P is not the equality and $=_B := \{(b,b) \mid b \in B\}$.

Now let $\Phi: T_\Theta \rightarrow B$ be a $\Theta$-assignment. Then the mapping $\Phi_\Sigma: T_\Sigma \rightarrow A$ defined by $\Phi_\Sigma x := \Phi x$ for all $\Sigma$-variables x is a $\Sigma$-assignment. Furthermore $\Phi$ is the same mapping as $\Phi_\Sigma$ on the $\Sigma$-terms and $\Sigma$-atoms. Since $\mathcal{A}$ is a $\Sigma$-model, every clause from CS is satisfied by $\Phi_\Sigma$ and hence also by $\Phi$. This means $\mathcal{B}$ is a $\Theta$-model of CS. ■

## 2. Extending Congruences to Ill-Sorted Terms.

Let $T_0 \subseteq T_\Sigma$ and let $\sim$ be a binary relation on $T_0$ and let $\Psi \subseteq SUB_\Sigma$ be a monoid such that $\Psi(T_0) = T_0$. Extending Definition I.7.2 we say $\sim$ is a **$\Psi$-invariant congruence on $T_0$**, iff the following conditions are satisfied:

i) $\sim$ is an equivalence relation.

ii) For all function symbols f and all $s_i,t_i \in T_0$:

$s_i \sim t_i$ , $i=1,\ldots,n$ and $f(s_1,\ldots,s_n) \in T_0 \Rightarrow f(t_1,\ldots,t_n) \in T_0$ and $f(s_1,\ldots,s_n) \sim f(t_1,\ldots,t_n)$.

iii) $\forall \sigma \in \Psi: \forall s,t \in T_0$: $s \sim t \Rightarrow \sigma s \sim \sigma t$.

We say $\sim$ is a **$\Psi$-invariant weak congruence on $T_0$**, iff instead of ii) the following condition ii)' holds:

ii)' For all function symbols f and all $s_i,t_i \in T_0$:

if $s_i \sim t_i$ for all i and $f(s_1,\ldots,s_n)$ , $f(t_1,\ldots,t_n) \in T_0$ then $f(s_1,\ldots,s_n) \sim f(t_1,\ldots,t_n)$.

In the following we assume that the equational theory $\mathcal{E} = (\Sigma,E)$ is given.
Note that the relation $=_{\Sigma,E}$ is a $SUB_\Sigma$-invariant weak congruence on $\mathbf{T}_\Sigma$.
The relation $=_{\overline{\Sigma},E}$ is the equational theory generated by E, if all sort information is ignored.
This relation is a $SUB_{\overline{\Sigma}}$-invariant congruence on $\mathbf{T}_{\overline{\Sigma}}$.
We say the congruence $=_{\Sigma,E}$ is **congruence-closed** , iff $\forall s,t \in \mathbf{T}_\Sigma$ : $s =_{\Sigma,E} t \Leftrightarrow s =_{\overline{\Sigma},E} t$. ■

We give some examples of equational theories that are not congruence-closed or not deduction-closed

**2.1 Example.**

a) Let $\Sigma := \{A \sqsupseteq B,\ f: A \times A \to A\}$. Let $=_{\Sigma,E}$ be generated by $E := \{f(x_B, x_B) = x_B\}$. Then $=_{\Sigma,E}$ is neither congruence-closed nor sort-preserving: We have $f(x_A, x_A) =_{\overline{\Sigma},E} x_A$, but not $f(x_A, x_A) =_{\Sigma,E} x_A$ . Furthermore $\mathbf{LS}_\Sigma(f(x_B, x_B)) = A$, whereas $\mathbf{LS}_\Sigma(x_B) = B$.

b) Let $\Sigma := \{A \sqsupseteq B,\ f: A \times A \to A,\ f: B \times B \to B\}$. Let $=_{\Sigma,E}$ be generated by $E := \{f(x_B,x_B) = x_B\}$.
Then $=_{\Sigma,E}$ is sort-preserving and deduction-closed, but not congruence-closed.

c) Let $\Sigma := \{A \sqsupseteq B,\ a_1{:}A,\ a_2{:}A,\ f: B \times B \to B,\ f(a_1){:}A\}$. Let $=_{\Sigma,E}$ be generated by $E := \{a_1 = a_2\}$. Then $=_{\Sigma,E}$ is sort-preserving on the well-sorted terms, but not sort-preserving and not deduction-closed, since $f(a_1) =_{\Sigma,E} f(a_2)$ and $f(a_2)$ is not well-sorted.
■

**2.2 Proposition.** Let $\mathcal{E}$ be an equational theory. Let $\sim_{\Sigma,E}$ be the $SUB_{\Sigma}$-invariant congruence on $\mathbf{T}_{\bar{\Sigma}}$ generated by $=_{\Sigma,E}$.

Then $s \sim_{\Sigma,E} t \Leftrightarrow s =_{\Sigma,E} t$ for all well-sorted terms s,t.

**Proof.** The nontrivial direction is to show that $s \sim_{\Sigma,E} t \Rightarrow s =_{\Sigma,E} t$ for all well-sorted terms s,t. Let CS be a set of unit clauses consisting of the axioms E. Assume there are well-sorted terms $s_0,t_0$ with $s_0 \sim_{\Sigma,E} t_0$. Let $\Theta$ be the ill-sorted extension of $\Sigma$. Since the relation $=_{\Theta,E}$ and $\sim_{\Sigma,E}$ are equal, the clause set $CS \cup \{s_0 \neq t_0\}$ has no $\Theta$-model, hence by Theorem 1.1 it has no $\Sigma$-model. Hence $s_0 = t_0$ is valid in every $\Sigma$-model. Now Birkhoff's Theorem I.9.2 shows that $s =_{\Sigma,E} t$ is derivable. ■

Due to the above proposition we can extend the relation $=_{\Sigma,E}$ to all (including ill-sorted) terms, i.e. to the set $\mathbf{T}_{\bar{\Sigma}}$.

The set of terms that are related to some well-sorted term via $=_{\Sigma,E}$ is denoted by $QT(\mathcal{E})$, i.e., $QT(\mathcal{E}) := \{t \in \mathbf{T}_{\bar{\Sigma}} \mid \exists s \in \mathbf{T}_{\Sigma}\ s =_{\Sigma,E} t\}$, the set of **quasi-terms** with respect to $\mathcal{E}$. Note that the relation $=_{\Sigma,E}$ is a $SUB_{\Sigma}$-invariant congruence on $QT(\mathcal{E})$.

**2.3 Lemma.** $QT(\mathcal{E})/ =_{\Sigma,E}$ is $\Sigma$-isomorphic to $\mathbf{T}_{\Sigma} / =_{\Sigma,E}$ as a $\Sigma$-algebra.

**Proof.** Let $\gamma$: $\mathbf{T}_{\Sigma} / =_{\Sigma,E} \rightarrow QT(\mathcal{E})/ =_{\Sigma,E}$ be the mapping with $\gamma(t/=_{\Sigma,E}) = t/=_{\Sigma,E}$. Proposition 2.2 shows that this is well-defined. Obviously $\gamma$ is a bijection. We have to show that $\gamma$ and $\gamma^{-1}$ are $\Sigma$-homomorphisms, but this is again obvious since $\gamma$ is well-defined and works in some sense as identity on $\mathbf{T}_{\Sigma} / =_{\Sigma,E}$. ■

The following proposition shows that $=_{\Sigma,E}$ is demodulation-complete on the set $QT(\mathcal{E})$.

**2.4 Proposition.** Let $\mathcal{E} = (\Sigma,E)$ be the axiomatization of an equational theory. Let s,t be $\bar{\Sigma}$-terms. Assume that the (undirected) demodulation relation $\xrightarrow{*}$ is meant on ill-sorted terms as defined in I.12

Then $s =_{\Sigma,E} t \Leftrightarrow s \xrightarrow{*} t$.

**Proof.** Use Theorem I.9.2 and the ill-sorted extension $\Theta$ of $\Sigma$ as constructed in paragraph 1. ■

**2.5 Proposition.** The set $QT(\mathcal{E})$ is subterm-closed.

**Proof.** Let $s = f(s_1,\ldots,s_n) \in QT(\mathcal{E})$. Choose a shortest deduction $s \longrightarrow r_1 \ldots r_n \longrightarrow t$, where t is well-sorted an the terms $r_i$ are ill-sorted. The term t is not a vbariable or constant, since then $r_n$ must be well-sorted, hence $t = f(t_1,\ldots,t_n)$. Since the terms $r_i$ are not well-sorted, there is no reduction at toplevel. This means that for every $s_i$ we have a deduction to $t_i$, hence $s_i \in QT(\mathcal{E})$. This proves the proposition. ■

## 3. Order-Sorted Term Rewriting Systems.

In order to extend term rewriting systems to an order-sorted signature, we use [Hu80] and [HO80] as a guideline. Related work on sorted term rewriting systems is presented in [CD85, GJM85, SNMG87].

We assume that term rewriting systems are compatible, if not stated otherwise. This assumption is not critical, as shown in paragraphs 1 and 2, where it is shown that this assumption can easily be satisfied by adding a greatest sort for ill-sorted terms.

A term rewriting system R is called **weakly sort-decreasing**, iff for all $\Sigma$-terms s,t with $s \dashrightarrow_R t$, there exists a $\Sigma$-term r such that $t \overset{*}{\dashrightarrow}_R r$ and $\mathbf{S}_\Sigma(s) \subseteq \mathbf{S}_\Sigma(r)$. Obviously sort-decreasing (cf. paragraph I.12) implies weakly sort-decreasing.

A term rewriting system R is **locally confluent**, iff for all $\Sigma$-terms $r,s_1,s_2$:

$$r \longrightarrow_R s_1 \text{ and } r \longrightarrow_R s_2 \;\Rightarrow\; \exists t \in \mathbf{T}_\Sigma : s_1 \overset{*}{\longrightarrow}_R t \text{ and } s_2 \overset{*}{\longrightarrow}_R t.$$

In [Hu80] it is shown that

**3.1 Lemma.** A Noetherian relation is confluent iff it is locally confluent. ■

Now let us define **critical pairs**: We can assume without loss of generality that all rules in R are variable disjoint. Let $l_1 \to r_1$, $l_2 \to r_2 \in$ R and let $\pi \in O(l_1)$. Further let $\sigma \in \mu U_\Sigma(l_1\backslash\pi, l_2)$, then consider the term pair $(\sigma(l_1[\pi \leftarrow \sigma r_2]), \sigma r_1)$. Note that $l_1[\pi \leftarrow \sigma r_2]$ is a well-sorted term, since R is well-sorted and that in part III it will be shown that minimal sets of unifiers always exist.

The pair $(\sigma(l_1[\pi \leftarrow \sigma r_2]), \sigma r_1)$ is called a **critical pair**.

We say a critical pair (s,t) is **confluent**, iff there exists a $\Sigma$-term r with $s \overset{*}{\dashrightarrow}_R r$ and $t \overset{*}{\dashrightarrow}_R r$.

**3.2 Proposition.** Let R be weakly sort-decreasing .

Then the relation $\overset{*}{\longrightarrow}_R$ is locally confluent if every critical pair is confluent.

**Proof.** We proceed as in the proof of [Hu80]:

Assume that every critical pair is confluent. Let $s,t_1,t_2$ be $\Sigma$-terms with $s \longrightarrow_R t_1$ and $s \longrightarrow_R t_2$. There exist $\pi_1, \pi_2 \in O(s)$, $l_1 \to r_1$, $l_2 \to r_2 \in$ R and $\sigma_1, \sigma_2 \in SUB_\Sigma$ such that $\sigma_i l_i = s\backslash\pi_i$ and $t_i = s[\pi_i \leftarrow \sigma_i r_i]$ for i = 1,2.

We have two cases, according to the relative position of $\pi_1$ and $\pi_2$.

Case 1: Disjoint redeces: Then the two reductions commute.

Case 2: One redex is a prefix of the other. W.l.o.g.we can assume that $\pi_1$ is a prefix of $\pi_2$.

Let $\nu$ be an occurrence such that $\pi_1\nu = \pi_2$.

Case 2.1: $v = v_1v_2$ such that $l_1\backslash v_1 = x$ is a variable.

Then we can further reduce $t_2$ in such a way that all (appropriate) subterms $\sigma_1 x$ below $\pi_1$ are reduced in the same way as $s\backslash\pi_1.v_1$ and the sorts are decreased (i.e., that the sort of the reduct $r_x$ of $\sigma_1 x$ is smaller than the sort of x, since R is weakly sort-decreasing. Let $\sigma_1'$ be a substitution such that $\sigma_1'x := r_x$ and $\sigma_1'y := \sigma_1 y$, otherwise. Now we can apply the rewrite rule $l_2 \to r_2$ at occurrence $\pi_1$ (with well-sorted substitution $\sigma_1'$). We get the same result as a reduction of $t_1$, if we reduce appropriately subterms $\sigma_1 x$ in $t_1$ to $r_x$.

Case 2.2. $v \in O(l_1)$ and $l_1\backslash v$ is not a variable. $l_1\backslash v$ and $l_2$ are $\Sigma$-unifiable. Hence there exists some most general $\Sigma$-unifier $\sigma$ with $\sigma \leq_\Sigma \sigma_1[V(l_1)]$ and $\sigma \leq_\Sigma \sigma_2\,[V(l_2)]$. This unifier corresponds to the critical pair $(\sigma(l_1[v \leftarrow \sigma r_2]), \sigma r_1)$, which is confluent by assumption.

Hence also the terms $s_1\backslash\pi_1[v \leftarrow \sigma_2 r_2]$ and $\sigma_1 r_1$ are confluent. ■

Lemma 3.1 and Proposition 3.2 imply the following:

**3.3 Theorem**. Let R be a weakly sort-decreasing term rewriting system such that $\xrightarrow{*}_R$ is Noetherian.

Then $\xrightarrow{*}_R$ is confluent, iff every critical pair is confluent. ■

If R is a weakly sort-decreasing term rewriting system such that $\xrightarrow{*}_R$ is Noetherian and confluent, then we will call R **canonical**.

Note that for a canonical term rewriting system every term t has a unique normalform $\|t\|_R$, such that $S_\Sigma(t) \subseteq S_\Sigma(\|t\|_R)$ and in regular signatures $\mathbf{LS}_\Sigma(\|t\|_R) \sqsubseteq \mathbf{LS}_\Sigma(t)$.

The following example from [SNMG87] shows that Theorem 3.3 does not hold if the TRS is not weakly sort-decreasing:

Let $\Sigma := \{A \sqsubset B$ , a:A , b:B, f: B $\to$ B $\}$ .

The term rewriting system R := $\{f(x_A) \to x_A$ , a $\to$ b$\}$ has no critical pairs and is terminating. But it is not confluent, since f(a) $\to_R$ a and f(a) $\to_R$ f(b), but a and f(b) are not reducible.

In the following we give a criterion for sort-decreasingness of a term rewriting system in a linear signature. Without linearity of the signature one needs very strong restrictions on the term rewriting system and the signature: For example if f(x x): S is a (nonlinear) term declaration and f(t t) is a term such that t is reducible to t', then f(t t) $\longrightarrow_R$ f(t t'), but f(t t') is not a $\Sigma$-instance of f(x x). Hence there must be another declaration that shows that f(t t') is of sort S. This example shows that a term rewriting system in a nonlinear signature is in general not sort-decreasing, and if it is, the nonlinear declarations are either redundant or the nonlinear declarations and the reduction are in some sense separated.

In the following we give a criterion for compatibility and sort-decreasingness, which is more general than the condition in [SNGM87], who give a criterion for elementary, regular signatures in terms of all weakenings of all rewrite rules.

Let t:S be a term declaration, $\pi \in O(t)$, $l_i \to r_i$ be a rewrite rule and let $\sigma$ be a most general $\Sigma$-unifer of $t\backslash\pi$ and $l_i$. Then we call the pair $(\sigma t[\pi \leftarrow \sigma r_i], \mathbf{S}_\Sigma(t))$ a **critical sort relation.** We say a critical sort relation $(s, \mathbf{S}_\Sigma(t))$ is **satisfied,** if $\mathbf{S}_\Sigma(s) \supseteq \mathbf{S}_\Sigma(t)$.

**3.4 Proposition.** Let $\Sigma$ be a linear signature and let R be a (not necessarily compatible) term rewriting system. If all critical sort relations are satisfied, then R is compatible and sort-decreasing.

**Proof.** Assume by contradiction that the proposition is false. Then there exist terms $s_1$ and $s_2$ with $s_1 \longrightarrow_{\pi,e,\mu} s_2$ such that $\mathbf{S}_\Sigma(s_1) \nsubseteq \mathbf{S}_\Sigma(s_2)$, where $e = l_i \to r_i$. Without loss of generality we can assume that $s_1$ is a smallest term with this property.
Let t:S be a term declaration in $\Sigma$ such that $S \in \mathbf{S}_\Sigma(s_1) - \mathbf{S}_\Sigma(s_2)$ and $s_1$ is a $\Sigma$-instance of t, i.e., $\sigma t = s_1$. Since $S \notin \mathbf{S}_\Sigma(s_2)$, we have that $s_2$ is not a $\Sigma$-instance of t. The occurrence $\pi$ must be an occurrence in t, since otherwise due to linearity of $\Sigma$ there exists a variable $x \in \mathbf{V}(t)$ at occurrence $\nu$ in t, such that $\sigma x \longrightarrow s_2\backslash\nu$ and $\mathbf{S}_\Sigma(\sigma x) \nsubseteq \mathbf{S}_\Sigma(s_2\backslash\nu)$, as $\Sigma$ is linear. This contradicts the minimality of $s_1$.
Since $\mu\sigma(t\backslash\pi) = \mu l_i$, there exists a most general $\Sigma$-unifier $\tau$ of $t\backslash\pi$ and $l_i$ with $\tau \le \mu\ [\mathbf{V}(t,l_i]$. Since the corresponding critical sort relation is satisfied, we have $S \in \mathbf{S}_\Sigma(\tau t[\pi \leftarrow \tau r_i])$. Hence $S \in \mathbf{S}_\Sigma(s_2)$, since $s_2$ is a $\Sigma$-instance of $\tau t[\pi \leftarrow \tau r_i]$. This is the final contradiction. ■

This proposition gives for linear signatures a nice and useful criterion for a term rewriting system to be canonical:

**3.5 Corollary**. Let $\Sigma$ be a linear signature and let R be a term rewriting system.
If i) all critical pairs are confluent and
ii) all critical sort relations are satisfied and
iii) $\xrightarrow{*}_R$ is Noetherian,
then R is a canonical term-rewriting system. ■

This specializes to elementary signatures in the following way:
The critical sort relations in elementary signatures can be obtained by weakening the left hand side of every rewrite rule in all possible ways (with most general weakening substitutions) and

then check the sort of the right hand side after substituting. Hence the following is the essential step in the test for critical sort relations:

Take a sort S and a rewrite rule $l \to r$, compute the set of most general weakenings $\mu W_\Sigma(l \sqsubseteq S)$ and check $S_\Sigma(\sigma l) \subseteq S_\Sigma(\sigma r)$ for all $\sigma \in \mu W_\Sigma(l \sqsubseteq S)$.

In part III it is shown, that unification and weakening in elementary signatures is decidable and effectively finitary, hence we have:

3.6 **Corollary.** In elementary signatures it is decidable, whether a terminating term rewriting system is canonical or not. ■

Note that termination of a TRS is in general undecidable (a proof and further references can be found in [De87].

Proposition 3.4 can also be used to give criteria for a regular equational theory to be sort-preserving.

3.7 **Corollary**. Let $\Sigma$ be a linear signature and let $\mathcal{E} = (\Sigma,E)$ be a regular equational theory. Furthermore let $R_E$ be the term rewriting system consisting of all rules $s \to t$ and $t \to s$ for $s = t \in E$.

Then $\mathcal{E}$ is sort-preserving, iff $R_E$ is sort-decreasing. ■

In order to handle the general case of term rewriting systems in nonlinear signatures, we extend the definitions above.

We define a parallel reduction rule for the TSR R, written $s \Longrightarrow_R t$:
Let $\pi \in O(s)$, let $l_j \to r_j$ be a rewrite rule and let $\sigma$ be a $\Sigma$-substitution with $\sigma l_j = s\backslash\pi$. Now let $\Pi = \{\pi_1,\ldots,\pi_k\}$be the set of all occurrences of s with $s\backslash\pi_i = s\backslash\pi$. We say s reduces to t (in parallel), denoted as $s \Longrightarrow_R t$, if $t = s[\pi_1 \leftarrow \sigma r_j] \ldots[\pi_k \leftarrow \sigma r_j]$. We may also denote this reduction by $s \Longrightarrow_{j,\sigma} t$ Let $\overset{*}{\Longrightarrow}_R$ denote the transitive, reflexive closure of $\Longrightarrow_R$.
We define weak critical sort relations:
Let t:S be a declaration, let $\Pi' = \{v_1,\ldots,v_m\} \subseteq O(t)$ be a set of independent occurrences, $l_j \to r_j$ be a rewrite rule and let $\tau$ be a most general $\Sigma$-unifer of the set $\{t\backslash\pi \mid \pi \in \Pi'\} \cup \{l_j\}$.
Let $t_\tau$ be defined as the corresponding $\Longrightarrow$-reduct: $\tau t \Longrightarrow_{j,\tau} t_\tau$.
Then $(t_\tau, S_\Sigma(t))$ is a **weak critical sort relation**. We say a weak critical sort relation $(t_\tau, S_\Sigma(t))$ is **satisfied**, if $S_\Sigma(t_\tau) \supseteq S_\Sigma(t)$. For $\tau t \Longrightarrow_{j,\tau} t_\tau$ let $\Pi = \{\pi_1,\ldots,\pi_k\} \supseteq \Pi'$ be the set of occurrences involved in this reduction.

We give a criterion for weak sort-decreasingness:

**3.8 Proposition.** Let $\Sigma$ be a signature and let R be a term rewriting system.

If all weak critical sort relations are satisfied, then $\xRightarrow{*}_R$ is sort-decreasing.

Furthermore R is weakly sort-decreasing.

**Proof.** It suffices to prove that $\Longrightarrow_R$ is sort-decreasing.

Assume by contradiction that the proposition is false. Then there exist $\Sigma$-terms $s_1$ and $s_2$ with $s_1 \Longrightarrow_{j,\mu} s_2$ such that $S_\Sigma(s_1) \nsubseteq S_\Sigma(s_2)$, with the rewrite rule $l_j \to r_j$ and the substitution $\mu$. Without loss of generality we can assume that $s_1$ is a smallest term with this property.

Let t:S be a term declaration in $\Sigma$ such that $S \in S_\Sigma(s_1) - S_\Sigma(s_2)$ and $s_1$ is a $\Sigma$-instance of t, i.e., $\sigma t = s_1$. Since $S \notin S_\Sigma(s_2)$, we have that $s_2$ is not a $\Sigma$-instance of t.

By the minimality of $s_1$ the reduction on the subterms $\sigma x \Longrightarrow_{j,\mu} s_x$ is sort-decreasing for all variables $x \in V(t)$. If the reduction $\Longrightarrow_{j,\mu}$ took place only below variable occurrences of t, then $s_2$ would be a $\Sigma$-instance of t, which is not true.

Let $\Pi := \{\pi_1,\ldots,\pi_k\}$ be the occurrences in t, where the reduction of $s_1 \Longrightarrow_{j,\mu} s_2$ actually changes the term t.

Now there exists a weak critical sort relation constructed from t:S and $l_j \to r_j$ and the set of occurrences $\Pi$ with most general $\Sigma$-unifier $\tau$ such that $\tau \le \sigma \cup \mu\ [V(t,l_j)]$. We argue that the corresponding set of occurrences for the critical sort relation is exactly $\Pi$. Otherwise, the reduction $\Longrightarrow_{j,\mu}$ changes more occurrences in $s_1$, since $\tau$ is more general than $\sigma$. Let $\tau t \Longrightarrow_{j,\tau} t_\tau$

Since the weak critical sort condition is satisfied, we have $S \in S_\Sigma(t_\tau)$. The occurrences of reductions in $s_1$ are independent: Either these occurrences are in $\Pi$ or the reductions are below variable occurrences of t.

Let $\sigma'$ be the $\Sigma$-substitution defined by $\sigma' x := s_x$ (where $\sigma x \Longrightarrow_{j,\mu} s_x$ ). We have $s_2 = \sigma'\, t_\tau$

. Hence $S \in S_\Sigma(s_2)$. This is a contradiction.

The second part of the proposition holds, since $\Longrightarrow_R$ can be simulated by some steps $\dashrightarrow_R$. ■

Now we have a general criterion for canonicity:

**3.9 Theorem.** Let R be a term rewriting system.

If i) all critical pairs are confluent and

ii) all weak critical sort-relations are satisfied and

iii) $\xrightarrow{*}_R$ is Noetherian,

then R is a canonical term rewriting system. ■

**Remark**: These criteria turn into a decision algorithm for local confluence of R, if unification in $\Sigma$ is decidable, finitary and the finite unifier sets are effectively computable. Furthermore with some luck, the check for weak sort-decreasingness and for local confluence may terminate and then we know definitively whether R is or is not locally confluent.

A completion algorithm [KB70, Bu87] can be adapted to our kind of signatures, however, it is not clear which restrictions a term ordering should obey. In the case of sort-decreasing TRS, it seems to be appropriate to use a term-ordering that respects sorts, but in the general case the term ordering may be such that it does not respect the sort ordering.

We give an application of term rewriting systems in a sorted signature, which shows that in some cases it is possible to describe an infinite term rewriting system in a finite way using sorts and declarations. This example is from [HKi87, HKi85], where a concept of domains, meta-variables and meta-rules is used, which seems to converge to sorted signatures.

**3.10 Example.** Given the rule $f(g(f(x))) \rightarrow g(f(x))$, the usual completion procedure generates an infinite family of rules, all of the form $f(g^n(f(x))) \rightarrow g^n(f(x))$. The following sort-structure shows how to describe this infinite rule system in a finite way and how to prove that it is canonical:

Let $\Sigma := \{TOP \sqsupseteq A,\ f{:}TOP \rightarrow TOP,\ g{:}TOP \rightarrow TOP\ g{:}A \rightarrow A,\ g(f(x_{TOP})){:}A\}$.
The rule is $f(x_A) \rightarrow x_A$.
Obviously this system is terminating.
In order to show that this system is canonical, we have to check the conditions in 3.4.
There are two critical sort relations: one is that the sort of $f(x_A)$ is greater than or equal to A, which is true. The other, nontrivial one comes from overlapping $g(f(x_{TOP})){:}A$ by the rule $f(x_A) \rightarrow x_A$. The critical sort relation is that $g(x_A)$ should be of sort A, which is also true.
Now Proposition 3.4 states that the TRS is compatible and canonical.
It remains to show that the transformation into the sorted case is correct. The set of all terms of sort A is the following: $\{x_A$, $g(f(t))$, where t is an arbitrary term and $g(s)$, where s is a term of sort $A\}$. This means the set corresponds to the set of terms defined by the term scheme $g^n(f(x))$ for $n \geq 1$. ■

The rest of this paragraph deals with ground equations and ground term rewriting systems and is used in part V.

Related work on ground equations can be found in [NO80, Ga86] where congruence-closure methods are used to give decision algorithms for systems of ground equations.

In order to simplify arguments we assume in the rest of this paragraph that there are no

ill-sorted terms and that the signature is finite.
We define some notions for term orderings for proving termination of term rewriting systems which are consistent with [De87].

**3.11 Definition.** An ordering $<_s$ on ground terms is a simplification ordering, iff the following conditions are satisfied:
i) $<_s$ is monotonic, i.e. $s_i \leq_s t_i \Rightarrow f(s_1,\ldots,s_n) \leq_s f(t_1,\ldots,t_n)$.
ii) $<_s$ has the subterm property $t <_s f(\ldots,t,\ldots)$.

In the following we will use the ordering defined as follows:
**3.12 Definition.** Let $\Sigma$ be a finite signature. Let the ordering $<_s$ on ground terms $T_{\Sigma,gr}$ be defined as follows:
i) constants and functions are ordered by a linear ordering.
ii) If size(s) < size(t) then $s <_s t$.
iii) Terms of equal size are ordered lexicographically (as strings). ■

We use $\leq_s$, $>_s$, $\geq_s$ with the obvious meaning.

The next lemma shows that $<_s$ is a well-founded simplification ordering on ground terms.
**3.13 Lemma.** Let $<_s$ be the ordering of Definition 3.12.
i) $s \leq_s t$ implies size(s) $\leq$ size(t).
ii) $<_s$ is a well-ordering on $T_{\Sigma,gr}$.
iii) $<_s$ is monotonic.
iv) $<_s$ has the subterm property.

**Proof.**
i) Follows from the definition.
ii) Due to I.10.5 there is only a finite number of terms that are smaller than a given one Furthermore all ground terms are comparable and $<_s$ is antisymmetric.
iii) Let $s_i \leq_s t_i$ and let f be a function symbol. We have to prove that $f(s_1,\ldots,s_n) \leq_s f(t_1,\ldots,t_n)$.
If size($s_i$) < size($t_i$) for some i, then $f(s_1,\ldots,s_n) <_s f(t_1,\ldots,t_n)$ by definition.
In the case size($s_i$) = size($t_i$) for all i, we have also size($f(s_1,\ldots,s_n)$) = size($f(t_1,\ldots,t_n)$).
Now we can use the lexicographic ordering on $f(s_1,\ldots,s_n)$ and $f(t_1,\ldots,t_n)$ and obtain $f(s_1,\ldots,s_n) \leq_s f(t_1,\ldots,t_n)$.
iv) We have $f(\ldots,t,\ldots) >_s t$, since size( $f(\ldots,t,\ldots)$ ) > size(t). ■

The ordering $<_s$ can be lifted to all terms as follows: For a term t let mgrt(t) be the multi-set of maximal ground terms of t and let the multisets be ordered by the multiset-ordering induced by $<_s$. Note that this ordering is well-founded, but not total.

**3.14 Proposition.** Let $R = \{l_i \rightarrow r_i \mid i = 1,\ldots,n\}$ be a ground term rewriting system with $l_i >_s r_i$. Then:

i) R is Noetherian.

ii) If there are no critical pairs, then R is also confluent and hence canonical.

**Proof.** Let t be a term and mgrt (t) be the multi-set of maximal ground terms of t. Then every reduction step makes mgrt(t) smaller in the multiset-ordering, hence reduction terminates and so R is Noetherian.

If there are no critical pairs, then the normalform of a term t can be computed by reducing the maximal ground terms of some term t. The result is obviously independent of the sequence of reductions. ■

We want to give an nondeterministic completion procedure of Knuth-Bendix type [KB70] for constructing a canonical term rewriting system in order to solve the word-problem with respect to a set of ground equations.

**3.15 Definition.** Let $R = \{l_i \rightarrow r_i \mid i = 1,\ldots,n\}$ be a ground term rewriting system with $l_i >_s r_i$.

We use a deduction system that consists of the following rules:

Rule 1. (critical pairs)

Let $l_j \rightarrow r_j$, $l_k \rightarrow r_k$ be different rules in R such that $l_j /\pi = l_k$. Let $s_1 := r_j$ and let $s_2 := l_j[\pi \leftarrow r_k]$.

Delete $l_j \rightarrow r_j$ from R and if $s_1 >_s s_2$ then add $s_1 \rightarrow s_2$ to R, if $s_1 <_s s_2$ then add $s_2 \rightarrow s_1$ to R and if $s_1 = s_2$ then do not add $s_1 = s_2$.

Rule 2. (application of rules to other rules)

Let $l_j \rightarrow r_j$, $l_k \rightarrow r_k$ be different rules in R such that $r_j /\pi = l_k$.

Replace $l_j \rightarrow r_j$ by $l_j \rightarrow r_j[\pi \leftarrow r_k]$. ■

**3.16 Proposition.** The completion procedure in Definition 3.15 terminates, leaves the generated equational theory on $\mathbf{T}_\Sigma$ invariant and the resulting term rewriting system is canonical on $\mathbf{T}_\Sigma$.

**Proof.**

i) First we prove that the equational theory is not changed:

Rule 1: there are two cases.

Case 1. $s_1 \neq s_2$ . Then we can prove $r_j =_{\mathcal{E}} l_j[\pi \leftarrow r_k] =_{\mathcal{E}} l_j[\pi \leftarrow l_k] = l_j$

Case 2. $s_2 = s_1$ . Then we can prove $l_j = l_j[\pi \leftarrow l_k] =_{\mathcal{E}} l_j[\pi \leftarrow r_k] = r_j$ .

Rule 2: We have to prove $l_j =_{\mathcal{E}} r_j$: From $l_j =_{\mathcal{E}} r_j[\pi \leftarrow r_k] =_{\mathcal{E}} r_j[\pi \leftarrow l_k] = r_j$ we conclude $l_j =_{\mathcal{E}} r_j$.

ii) The completion terminates: If we order rewrite rules by the lexicographical ordering induced by $<_S$, then we obtain a well-founded ordering on rules. Every rule replaces a rule by a smaller rule or deletes a rule. Hence the procedure terminates.

iii) There are no critical pairs, since otherwise Rule 1 is applicable.

iv) Now Proposition 3.14 shows that R is canonical. ■

Note that the above results hold also for non sort-decreasing ground term rewriting systems.

A trivial (well-known) corollary is:

**3.17 Corollary.** The word-problem in an equational theory defined by ground axioms is decidable.

## 4. Sort-assignments.

Usually, the syntactical sort of a term is defined by specifying the sort of variables, constants and the behaviour of functions in the form f: $S_1 \times \ldots \times S_n \rightarrow S$ or even with term declarations. In this paragraph we abstract from this syntactical specification of sorts and view the sort of a term as a function on terms having the right properties. We show that the notion of a sort-assignment as defined here corresponds to the notion of a signature with (infinitely many) declarations. The notion of a sort-assignment enables us to use different descriptions of the sort of a term.

**4.1 Definition.** Let $\bar{\Sigma}$ be an unsorted signature, let $\mathbf{S}_\varphi$ be a set of sorts quasi-ordered by $\sqsubseteq_\varphi$, let $\mathbf{T}_\varphi$ be a subterm-closed set of terms and let $\varphi: \mathbf{T}_\varphi \rightarrow \mathcal{P}(\mathbf{S}_\varphi)$ be a mapping from terms into sets of sorts, such that $\varphi(t)$ is an upper segment. Let $\mathbf{V}_\varphi$ denote the set of all variables in $\mathbf{T}_\varphi$

Define $SUB_\varphi$ to be the set of all substitutions $\sigma$ satisfying $(\forall x \in \mathbf{V}_\varphi\ \varphi(\sigma x) \supseteq \varphi x)$.

Furthermore let the following conditions be satisfied:

i) For every sort $S \in \mathbf{S}_\varphi$ : $\mathbf{V}_S \subseteq \mathbf{T}_\varphi$.

ii) For every variable $x \in \mathbf{T}_\varphi$: $\varphi(x) = \mathbf{S}_\Sigma(x)$.

iii) for every sort S, there exists a ground term $t_{S,gr}$, such that $S \in \varphi(t_{S,gr})$.

iv) $\forall \sigma \in SUB_\varphi\ \forall t \in \mathbf{T}_\varphi$ : $\sigma t \in \mathbf{T}_\varphi$ and $\varphi(\sigma t) \supseteq \varphi t$.

In this case we say $\varphi$ is a **sort-assignment.**■

The next proposition shows that sort-assignments are describable by (possibly infinitely many) term declarations.

**4.2 Proposition.** Let $\bar{\Sigma}$ be an unsorted signature and let $\varphi$ be a sort-assignment.

Then there exists a signature $\Sigma$ such that $\mathbf{S}_\varphi = \mathbf{S}_\Sigma$, $\sqsubset_\varphi = \sqsubset_\Sigma$, $\mathbf{T}_\varphi = \mathbf{T}_\Sigma$ and $\varphi(t) = S_\Sigma(t)$.

**Proof.** To satisfy the conditions $\mathbf{S}_\varphi = \mathbf{S}_\Sigma$ and $\sqsubset_\varphi = \sqsubset_\Sigma$ is trivial.

We define $\Sigma$ as the set consisting of all subsort declarations $R \sqsubset_\varphi S$ and of all term declarations $\{t{:}S \mid S \in \varphi(t)\}$ for $t \in \mathbf{T}_\varphi - \mathbf{V}_\varphi$. Let $T_{\varphi,S} := \{t \in \mathbf{T}_\varphi \mid S \in \varphi(t)\}$.

We show $T_{\varphi,S} = T_{\Sigma,S}$ for all $S \in \mathbf{S}_\varphi$:

The relation $T_{\varphi,S} \subseteq T_{\Sigma,S}$ is obvious by definition of $\Sigma$.

In order to show the converse $T_{\Sigma,S} \subseteq T_{\varphi,S}$ it is sufficient to show that the sets $T_{\varphi,S}$ are closed with respect to Definition I.4.3. We check condition iii) of Definition 4.3:

Let $t \in \mathbf{T}_{\varphi,S}$, $r \in \mathbf{T}_{\varphi,R}$ and x a variable with $R \sqsubseteq_\varphi S(x)$.Then $\{x \leftarrow r\}$ is in $SUB_\varphi$, hence by condition iii) above we have $\{x \leftarrow r\}t \in \mathbf{T}_{\varphi,S}$. An immediate consequence is $SUB_\Sigma = SUB_\varphi$.

The assumptions I.4.11 on signatures are satisfied due to the preconditions of this proposition. ■

We can characterize regular signatures in a similar way, if we let $\mathbf{S}_\varphi$ be a set of sorts partially ordered by $\sqsubset_\varphi$ and replace the function $\varphi: \mathbf{T}_\varphi \rightarrow \mathcal{P}(\mathbf{S}_\varphi)$ by a function $\varphi: \mathbf{T}_\varphi \rightarrow \mathbf{S}_\varphi$ and the conditions ii) and iii) by

ii)$_R$ For every variable $x \in \mathbf{T}_\varphi$: $\varphi(x) = S(x)$.

iii)$_R$ $\forall \sigma \in SUB_\varphi\ \forall t \in \mathbf{T}_\varphi : \varphi(\sigma t) \sqsubseteq_\varphi \varphi t$.

In this case we also speak of a **least-sort-assignment.**

## 5. Another Equational Deduction System.

We now give another derivation system for equational theories. It is similar to the Birkhoff-like derivation system in I.9.1, but to derive instances of equations is only allowed for the axioms in E and not for derived equations. We use this derivation system later in part IV to prove that certain unification algorithms are complete.

Let $E = \{l_i = r_i\}$ be the set of axioms of $\mathcal{E}$.

**5.1 Definition.**

i) $\vdash_d t = t$ for every term $t \in \mathbf{T}_\Sigma$.

ii) $\{s = t\} \vdash_d t = s$.

iii) $\{r = s, s = t\} \vdash_d r = t$.

iv) If $f(s_1,\ldots,s_n)$ and $f(t_1,\ldots,t_n)$ are well-sorted, then

$\{s_1 = t_1 , \ldots , s_n = t_n\} \vdash_d \ f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)$

v) $\vdash_d \ \sigma s = \sigma t$ for every $\sigma \in SUB_\Sigma$ and every $s = t \in E$. ■

Let the relation $\overset{\infty}{\longmapsto}_d$ be defined by : $s \overset{\infty}{\longmapsto}_d t$ iff $\vdash_d s = t$.

The above deduction system computes every valid equation:

**5.2 Proposition.** Let $s,t \in \mathbf{T}_\Sigma$. Then $(\vdash_d s = t) \Leftrightarrow s =_{\Sigma,E} t$.

Proof. "$\Rightarrow$" : trivial.

"$\Leftarrow$": We show that all steps of the Birkhoff deduction system in I.9.1 can be simulated, the only missing step is rule I.9.1.v), where all well-sorted instances of equations can be deduced.

We show by induction on the length of a deduction that for all terms s, t with $\vdash_d s = t$ and all substitutions $\sigma \in SUB_\Sigma$ we also have $\vdash_d \sigma s = \sigma t$.

The base case is rule 5.1 v) for the axioms of E.

The induction step is trivial for the rules i) -iii).

Let $\vdash_d s_1 = t_1 \ \& \ldots \& \ s_n = t_n$, let $f(s_1,\ldots,s_n)$ and $f(t_1,\ldots,t_n)$ be well-sorted, let $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)$ be the newly deduced equation and let $\sigma$ be a well-sorted substitution. Then by induction hypothesis we have $\vdash_d \sigma s_1 = \sigma t_1 \ \& \ldots \& \ \sigma s_n = \sigma t_n$. Furthermore $\sigma f(s_1,\ldots,s_n)$ and $\sigma f(t_1,\ldots,t_n)$ are well-sorted terms, hence by rule 5.1 iv) we can deduce $\sigma f(s_1,\ldots,s_n) = \sigma f(t_1,\ldots,t_n)$. ■

This deduction system is more appropriate for induction proofs involved in proving completeness of unification procedures. The next lemma shows that for every equation there exists a deduction that can be arranged in a somewhat standard way:

**5.3 Lemma.** Let $s,t \in \mathbf{T}_\Sigma$ and $s =_{\Sigma,E} t$. Then there exists a chain $s = r_0, r_1,\ldots,r_m = t$ such that

i) For all i either $r_i = r_{i+1}$ is deduced by rule 5.1 v) or by rule 5.1 iv)

ii) For all appropriate i: either $r_i = r_{i+1}$ or $r_{i+1} = r_{i+2}$ is deduced by rule 5.1 v).

**Proof.** i) We obtain such a chain by unfolding in a deduction the most recent steps 5.1 ii) and 5.1 iii).

ii) Assume by contradiction that $r_i = r_{i+1}$ and $r_{i+1} = r_{i+2}$ are both deduced by step 5.1 iv) and the chain corresponds to a dedcution with a minimal number of applications of rule 5.1 iv) Then we can already deduce $r_i = r_{i+2}$ by step 5.1 iv). The new deduction thus obtained may have more applications of symmetry and transitivity, but the number of applications of rule 5.1 iv) is decreased, hence we have reached a contradiction. ■

## 6. Characterizations of Deduction-Closedness, Congruence-Closedness and Sort-Preservation.

In Part IV.3 we give a unification procedure for a class of congruence-closed and sort-preserving equational theories. In order to use this procedure it is necessary to have criteria to recognize these properties given an axiomatization of the equational theory. In this paragraph we give some characterizations of deduction-closed, congruence-closed and sort-preserving congruences by properties of the generating set of equations. We also investigate the decidability of these properties.

In this paragraph we assume that $\mathcal{E} = (\Sigma,E)$ is given, that E is symmetric and finite and that the signature is finite.

First we give a criterion for checking the congruence-closedness of an equational theory:

**6.1 Lemma.** Let $\Sigma$ be a regular, elementary signature.

If for all $s = t \in E$ and for all $\bar{\Sigma}$-renamings $\rho$: $\rho s \in \mathbf{T}_\Sigma \Rightarrow (\rho t \in \mathbf{T}_\Sigma$ and $\rho s =_{\Sigma,E} \rho t)$

Then for all $s = t \in E$ and for all $\sigma \in SUB_{\bar{\Sigma}}$:

$$\sigma s \in \mathbf{T}_\Sigma \Rightarrow (\sigma t \in \mathbf{T}_\Sigma \text{ and } \sigma s =_{\Sigma,E} \sigma t).$$

**Proof.** Let $s=t \in E$ and let $\sigma \in SUB_{\bar{\Sigma}}$ with $DOM(\sigma) = \{x_1,...,x_n\}$, such that $\sigma s \in \mathbf{T}_\Sigma$. There exist new variables $y_i$ of sort $\mathbf{LS}(\sigma x_i)$, since the terms $\sigma x_i$ are well-sorted. Let $\tau := \{y_i \leftarrow \sigma x_i \mid i = 1,\dots,n\}$ and let $\rho := \{x_i \leftarrow y_i \mid i = 1,\dots,n\}$. Then $\rho$ is an idempotent $\bar{\Sigma}$-renaming and $\tau \in SUB_\Sigma$. Furthermore $\rho s \in \mathbf{T}_\Sigma$, since $\Sigma$ is elementary and $\sigma s \in \mathbf{T}_\Sigma$. The precondition now yields $\rho t \in \mathbf{T}_\Sigma$ and $\rho s =_{\Sigma,E} \rho t$.
Since $=_{\Sigma,E}$ is $SUB_\Sigma$-invariant we have $\tau\rho t \in \mathbf{T}_\Sigma$ and $\tau\rho s =_{\Sigma,E} \tau\rho t$ which in turn implies $\sigma t \in \mathbf{T}_\Sigma$ and $\sigma s =_{\Sigma,E} \sigma t$, since $\sigma = \tau\rho\,[x_1,...,x_n]$. ∎

Now we can give some criteria for congruence-closedness. The third criterion for regular, elementary signatures is decidable and easy to test.

**6.2 Proposition.**

i) If for all generating equations $s = t \in E$:
$\forall \sigma \in SUB_{\bar{\Sigma}}$: $\sigma s \in \mathbf{T}_\Sigma \Rightarrow \sigma t \in \mathbf{T}_\Sigma$ and $\sigma s =_{\Sigma,E} \sigma t$,
Then $=_{\Sigma,E}$ is congruence-closed.

ii) If $\Sigma$ is regular and elementary and for all generating equations $s = t \in E$:
For all $\bar{\Sigma}$-renamings $\rho$: $\rho s \in \mathbf{T}_\Sigma \Rightarrow (\rho t \in \mathbf{T}_\Sigma$ and $\rho s =_{\Sigma,E} \rho t)$.
Then $=_{\Sigma,E}$ is congruence-closed.

iii) If $\Sigma$ is regular and elementary and for all generating equations $s = t \in E$:

For all $\bar{\Sigma}$-renamings $\rho$: $\rho s \in \mathbf{T}_\Sigma \Rightarrow \rho \in SUB_\Sigma$

Then $=_{\Sigma,E}$ is congruence-closed.

**Proof.** We prove only i), since Lemma 6.1 and part i) immediately imply the second part. The third part follows from part ii), since $=_{\Sigma,E}$ is $\Sigma$-invariant.

i) The following assertion is proved by induction on the length of a deduction (I.9.1) of an equation:

For all $s =_{\bar{\Sigma},E} t$:

(†) $\forall \sigma \in SUB_{\bar{\Sigma}}$: $\sigma s \in \mathbf{T}_\Sigma \Rightarrow \sigma t \in \mathbf{T}_\Sigma$ and $\sigma s =_{\Sigma,E} \sigma t$.

Base case. For $s = t \in E$, which is the precondition of this proposition, here is nothing to prove.

Induction step.

i) New equations introduced by reflexivity or symmetry have the property (†).

ii) Let $t_1 =_{\bar{\Sigma},E} t_2$, $t_2 =_{\bar{\Sigma},E} t_3$ be the old equations and let $t_1 =_{\bar{\Sigma},E} t_3$ be the new one, introduced by transitivity.

Let $\sigma \in SUB_{\bar{\Sigma}}$ such that $\sigma t_1 \in \mathbf{T}_\Sigma$. Then by induction hypothesis, $\sigma t_2 \in \mathbf{T}_\Sigma$ and $\sigma t_1 =_{\Sigma,E} \sigma t_2$. Now again by induction hypothesis we have $\sigma t_3 \in \mathbf{T}_\Sigma$ and $\sigma t_2 =_{\Sigma,E} \sigma t_3$.

Transitivity yields $\sigma t_1 =_{\Sigma,E} \sigma t_3$.

iii) Let $s_i =_{\bar{\Sigma},E} t_i$ be given and let $f(s_1,...,s_n) =_{\bar{\Sigma},E} f(t_1,...,t_n)$ be the new equation.

Let $\sigma \in SUB_{\bar{\Sigma}}$ such that $\sigma f(s_1,...,s_n) \in \mathbf{T}_\Sigma$.

Then for all i we have $\sigma s_i \in \mathbf{T}_\Sigma$, since $\mathbf{T}_\Sigma$ is subterm-closed. The induction hypothesis implies $\sigma t_i \in \mathbf{T}_\Sigma$ and $\sigma s_i =_{\Sigma,E} \sigma t_i$. Since $=_{\Sigma,E}$ is a congruence and since $f(\sigma s_1,...,\sigma s_n) \in \mathbf{T}_\Sigma$, we conclude $f(\sigma s_1,...,\sigma s_n) =_{\Sigma,E} f(\sigma t_1,...,\sigma t_n)$

iv) Let $s =_{\bar{\Sigma},E} t$, $\tau \in SUB_{\bar{\Sigma}}$ and let $\tau s =_{\bar{\Sigma},E} \tau t$ be the new equation.

Let $\sigma \in SUB_{\bar{\Sigma}}$ such that $\sigma\tau s \in \mathbf{T}_\Sigma$.

Then by induction hypothesis, we have $\sigma\tau t \in \mathbf{T}_\Sigma$ and $\sigma\tau s =_{\Sigma,E} \sigma\tau t$, since $\sigma\tau \in SUB_{\bar{\Sigma}}$. ■

In general it is undecidable whether a congruence is congruence-closed:

**6.3 Proposition.** It is undecidable (even for regular and elementary signatures) whether a congruence is congruence-closed.

**Proof.** We show that decidability of congruence-closedness would imply the decidability of the word-problem (for ground terms) in finitely presented semi-groups:

Let $\Sigma$ be a signature which has only one sort A. Let $\mathcal{E}$ be a finitely presented semi-group and let s,t be two $\Sigma$–ground terms. We add the new sort $B \sqsubset A$ and the new ternary function symbol $f: B \times B \times A \rightarrow A$. Let $\Sigma'$ be the new signature. Note that all nonvariable $\Sigma$–terms have sort A and that $\Sigma'$ is regular and elementary. Let $E' := E \cup$

$\{f(x_B,y_B,s) = x_B,\ f(x_B,y_B,t) = y_B\}$.

It is easy to see $x_B =_{\Sigma',E'} y_B$ iff s and t are E-equal: If s and t are E-equal, then obviously $x_B =_{\Sigma',E'} y_B$. If s and t are not E-equal, then for every variable $x_B$ of sort B its E-equivalence-class is exactly $\{x_B\} \cup \{f(x_B, z_B, s') \mid s' =_{\Sigma,E} s$ and $z_B$ a variable of sort B$\}$ $\cup \{f(z'_B, x_B, t') \mid t' =_{\Sigma,E} t$ and $z'_B$ a variable of sort B$\}$.

$=_{\Sigma',E'}$ is congruence-closed $\Leftrightarrow$ $s =_{\Sigma,E} t$:

If s and t are not E-equal, then $=_{\Sigma',E'}$ is congruence-closed, since the application of a new equation is a dead end: the unsorted equivalence class of a term r not containing f of sort A does not contain well-sorted term with an occurrence of f. If s and t are E-equal, then we have $x_B =_{\Sigma',E'} y_B$ and all terms are in the relation $=_{\bar{\Sigma}',E'}$. Hence $=_{\Sigma',E'}$ is not congruence-closed.

Hence congruence-closedness is undecidable, since the word-problem for ground terms in finitely presented semi-groups is also undecidable [Ta79]. ■

Now we investigate the property deduction-closedness.

Note that sort-preservation implies deduction-closedness.

**6.4 Lemma.** Let the following condition be satisfied:

$\forall s_i,t_i \in \mathbf{T}_\Sigma$: $s_i =_{\Sigma,E} t_i$ and $f(s_1,\ldots,s_n) \in \mathbf{T}_\Sigma \Rightarrow f(t_1,\ldots,t_n) \in \mathbf{T}_\Sigma$.

Then $=_{\Sigma,E}$ is a deduction-closed congruence.

**Proof.** We have to show that for $s \in \mathbf{T}_\Sigma$, $t \in \mathbf{T}_{\bar{\Sigma}}$ and $s =_{\Sigma,E} t$ we have $t \in \mathbf{T}_\Sigma$.

Assume there is an equation $s =_{\Sigma,E} t$ with $s \in \mathbf{T}_\Sigma$, $t \in \mathbf{T}_{\bar{\Sigma}} - \mathbf{T}_\Sigma$.

We can assume that $s =_{\Sigma,E} t$ is the equation with a shortest deduction starting wit equations from E and $s \in \mathbf{T}_\Sigma$, $t \in \mathbf{T}_{\bar{\Sigma}} - \mathbf{T}_\Sigma$. This means all terms occurring in th deduction are well-sorted. Since t is not well-sorted, the equation $s =_{\Sigma,E} t$ must have bee generated in the following way: $s = f(s_1,\ldots,s_n)$ and $t = f(t_1,\ldots,t_n)$ and $s_i =_{\Sigma,E} t_i$ for all But then the precondition of this lemma shows $t = f(t_1,\ldots,t_n) \in \mathbf{T}_\Sigma$. ■

**6.5 Proposition.** Let $\Sigma$ be an elementary signature and let $=_{\Sigma,E}$ be a sort-preservin congruence.

Then $=_{\Sigma,E}$ is deduction-closed.

Proof. The requirements of Lemma 6.4 are satisfied. ■

**6.6 Proposition**. Let $=_{\Sigma,E}$ be a sort-preserving congruence and for every function symb f let the most general terms be basic terms (cf. I.5.7 ff.).

Then $=_{\Sigma,E}$ is deduction-closed.

**Proof.** We show the preconditions of Lemma 6.4: Let $s_i,t_i \in \mathbf{T}_\Sigma$ and let $s_i =_{\Sigma,E} t_i$. Then $S_\Sigma(s_i) = S_\Sigma(t_i)$ since $=_{\Sigma,E}$ is sort-preserving. By assumption there exists a term declaratio

$f(x_1,\ldots,x_n):S$ with $f(x_1,\ldots,x_n) \geq_\Sigma f(s_1,\ldots,s_n)$. Obviously we have also $f(x_1,\ldots,x_n) \geq_\Sigma f(t_1,\ldots,t_n)$, hence $f(t_1,\ldots,t_n)$ is well-sorted. ■

**6.7 Proposition.** For a regular, elementary signature $\Sigma$ it is decidable whether $=_{\Sigma,E}$ is deduction-closed.

**Proof.** Let the relation $\approx$ on $S_\Sigma$ be defined as follows: $A \approx B$, iff there exist terms $t$, $t'$ with $\mathbf{LS}_\Sigma(t) = A$, $\mathbf{LS}_\Sigma(t') = B$ and $t =_{\Sigma,E} t'$. We use the deduction-system in 5.1 to make a fixed-point iteration to determine $\approx$. For the generating relations $s_i = t_i$ in E we can comput the relation $\approx$ by checking all sorts for variables in these equations. We generate the transitive closure and then use the steps 5.1.iv). This iteration terminates and either has produced a relation $A \approx IL$ or not. Hence deduction-closedness is decidable. ■

However, decidability is endangered if the preconditions are dropped.

**6.8 Proposition.** In general it is undecidable whether a congruence is deduction-closed.

**Proof.** We show that decidability of deduction-closedness would imply the decidability of the $\Sigma$-unification problem in arbitrary signatures:

Let $\mathcal{E}$ be the empty theory and let s,t be two terms. We add the new unary function symbol f defined on all sorts in $S_\Sigma$, the new sort A, the constants a and b of sort A, and the declaration f(a):A. Let $E' := \{f(s) = a, f(t) = b\}$ and let $\Sigma'$ be the new signature.

It is easy to see $a =_{\Sigma',E'} b$ iff s and t are $\Sigma$-unifiable. The only possibility to deduce an ill-sorted term is to deduce f(b) from f(a). Hence we have that s and t are unifiable iff $\Sigma'$ is deduction-closed. Theorem III.6.1 shows now that deduction-closedness is undecidable. ■

Now we turn to the sort-preserving property of equational theories.

**6.9 Proposition.** Let $\Sigma$ be a regular, elementary signature. Let the following condition be satisfied:

For all well-sorted $\bar{\Sigma}$-renamings $\rho$ and all $s=t \in E$: $\mathbf{LS}_\Sigma(\rho s) = \mathbf{LS}_\Sigma(\rho t)$.

Then for all $\sigma \in SUB_\Sigma$ and all $s=t \in E$: $\mathbf{LS}_\Sigma(\sigma s) = \mathbf{LS}_\Sigma(\sigma t)$.

**Proof.** Let $s=t \in E$, let $\sigma \in SUB_\Sigma$ with $DOM(\sigma) = \{x_1,\ldots,x_n\}$. There exist new variables $y_i$ of sort $\mathbf{LS}_\Sigma(\sigma x_i)$. Let $\tau := \{y_i \leftarrow \sigma x_i \mid i = 1,\ldots,n\}$ and let $\rho := \{x_i \leftarrow y_i \mid i = 1,\ldots,n\}$. Obviously $\rho$ is a $\bar{\Sigma}$-renaming and $\rho,\tau$ are well-sorted substitutions. Hence $\mathbf{LS}_\Sigma(\rho s) = \mathbf{LS}_\Sigma(\rho t)$. Application of $\tau$ to the terms $\rho s$ and $\rho t$ does not change their sorts, since $\Sigma$ is elementary. From $\sigma = \tau\rho\ [x_1,\ldots,x_n]$ we conclude $\mathbf{LS}_\Sigma(\sigma s) = \mathbf{LS}_\Sigma(\sigma t)$. ■

**+6.10 Proposition.** Let $\Sigma$ be an elementary signature. Then the following two properties are equivalent:

i) For all $\sigma \in SUB_\Sigma$ and all $s{=}t \in E$: $S_\Sigma(\sigma s) = S_\Sigma(\sigma t)$.

ii) $=_{\Sigma,E}$ is sort-preserving.

**Proof.** ii) $\Rightarrow$ i) is trivial.

i) $\Rightarrow$ ii): We show by induction on the length of a deduction that

$s =_{\Sigma,E} t \Rightarrow \forall\sigma \in SUB_\Sigma : S_\Sigma(\sigma s) = S_\Sigma(\sigma t)$.

Condition i) is the base case.

Induction step:

i) Let $t_1 =_{\Sigma,E} t_3$ be deduced from $t_1 =_{\Sigma,E} t_2$ and $t_2 =_{\Sigma,E} t_3$ and let $\sigma \in SUB_\Sigma$. By induction hypothesis we have $S_\Sigma(\sigma t_1) = S_\Sigma(\sigma t_2) = S_\Sigma(\sigma t_3)$.

ii) Let $\tau s =_{\Sigma,E} \tau t$ be deduced from $s =_{\Sigma,E} t$ for $\tau \in SUB_\Sigma$. For a well-sorted substitution $\sigma$ we have $\sigma\tau \in SUB_\Sigma$, hence $S_\Sigma(\sigma\tau s) = S_\Sigma(\sigma\tau t)$ by induction hypothesis.

iii) Let $f(s_1,\ldots,s_n) =_{\Sigma,E} f(t_1,\ldots,t_n)$ be deduced from $s_i =_{\Sigma,E} t_i$. Let $\sigma \in SUB_\Sigma$. The induction hypothesis implies $S_\Sigma(\sigma s_i) = S_\Sigma(\sigma t_i)$ and since $\Sigma$ is elementary we have $S_\Sigma(\sigma f(s_1,\ldots,s_n)) = S_\Sigma(\sigma f(t_1,\ldots,t_n))$. ■

**6.11 Corollary.** Let $\Sigma$ be a regular, elementary signature. Then it is decidable, whether $=_{\Sigma,E}$ is sort-preserving.

**Proof.** Follows from Lemma 6.9 and from Proposition 6.10. The precondition of Lemma 6.9 is decidable by Proposition I.5.3, since we have to check only a finite number of $\Sigma$-renamings. ■

The above arguments can be generalized to show that for every elementary (nonregular) signature, the sort-preservation of congruences is decidable.

For nonelementary signatures it is in general undecidable whether a congruence is sort-preserving:

**6.12 Proposition.** It is undecidable whether $=_{\Sigma,E}$ is sort-preserving.

**Proof.** We show that decidability of sort-preservation would imply the decidability of the word-problem in equational theories:

Let E be an equational theory, where only the sort A is available. Let s,t be two terms. We add two new sorts B and C, two new constants b,c of sort A , the new function symbol f and the term declarations f(b):B, f(c):C to the signature. Furthermore we add the axioms $b = s$ and $c = t$ to E, giving E'. Let $=_{\Sigma,E'}$ be the new congruence. Obviously we have that $=_{\Sigma,E'}$ is sort-preserving, iff $s =_{\Sigma,E} t$. Since $s =_{\Sigma,E} t$ is undecidable, the sort-preservation is undecidable. ■

A signature is called **sort-stable,** iff $S_\Sigma(s_i) = S_\Sigma(t_i)$ for i = 1,...,n implies $S_\Sigma(f(s_1,...,s_n)) = S_\Sigma(f(t_1,...,t_n))$.
This means that $S_\Sigma$ is a function of f and $S_\Sigma(t_i)$ alone and that $S_\Sigma(f(t_1,...,t_n))$ does not depend on the structure of the subterms $t_i$ of t. By Lemma I.4.10 we have that elementary signatures are sort-stable.

We have that regular, sort-stable signatures characterize elementary signatures:

**6.13 Proposition.**

i) In a regular, sort-stable signature $\Sigma$ all term-declarations, which are not of the form $f: S_1 \times...\times S_n \to S_{n+1}$, are redundant. That means the signature is elementary.

**Proof.** i) Consider an arbitrary nonredundant term declaration $f(t_1,\dots,t_n):S$, that is not a function declaration. That means $LS_\Sigma(f(t_1,\dots,t_n)) = S$. We can replace the terms $t_i$ by variables $x_i$ with $S(x_i) = LS_\Sigma(t_i)$. Since $\Sigma$ is sort-stable and regular, we have $S \in S_\Sigma(f(x_1,\dots,x_n))$. By Proposition I.4.9 there must exist a function declaration $f:S_1\times\dots\times S_n \to S$. ■

**6.14 Example.** If the signature is not regular, then Proposition 6.13 may be false:
Let $\Sigma := \{A,B, f:A \to A,\ f:A \to B,\ g(f(x_A)): A\}$. Then $\Sigma$ is not regular, since $S_\Sigma( f(x_A)) = \{A,B\}$. However, the signature is sort-stable: Every well-sorted term t starting with f has as sort $S_\Sigma(t) = \{A,B\}$. Every well-sorted term starting with g has sort A and has the form g(f(t)). The only possibility to replace f(t) is by a term of sort {A,B}. Every such term has toplevel symbol f, hence a replacement of f(t) by f(t') gives a term of the form g(f(t')) and this term is of sort A. Now $\Sigma$ is sort-stable, but the term declaration $g(f(x_A))$: A is not redundant. ■

In the following we note some properties of substitutions that hold if restrictions are imposed on the signature or on the equational theory.

**6.15 Lemma.** Let $=_{\Sigma,E}$ be a sort-preserving congruence. Then:

i) $\forall \sigma \in SUB_\Sigma\ \forall \tau \in SUB_{\overline{\Sigma}}: \sigma =_{\Sigma,E} \tau\,[V] \Rightarrow \tau \in SUB_\Sigma$.

ii) If $=_{\Sigma,E}$ is congruence-closed, then:

$$\forall \sigma \in SUB_\Sigma\ \forall \tau \in SUB_{\overline{\Sigma}}:\ \sigma =_{\overline{\Sigma},E} \tau\,[V] \Rightarrow \tau \in SUB_\Sigma.$$

**Proof.**

i) Let $\sigma \in SUB_\Sigma$ and $\tau \in SUB_{\overline{\Sigma}}$. For all $x \in V_\Sigma$ we have $\{x \leftarrow \sigma x\} \in SUB_\Sigma$. Hence:
$S_\Sigma(\sigma x) \supseteq S_\Sigma(x) \Rightarrow$ (since $=_{\Sigma,E}$ is sort-preserving)
$S_\Sigma(\tau x) \supseteq S_\Sigma(x) \Rightarrow$
$\{x \leftarrow \tau x\} \in SUB_\Sigma$ for all $x \in V_\Sigma$. Thus $\tau \in SUB_\Sigma$.

ii) $\sigma =_{\overline{\Sigma},E} \tau\,[V] \Rightarrow \sigma =_{\Sigma,E} \tau\,[V]$, since $=_{\Sigma,E}$ is congruence-closed. Then apply i).■

**6.16 Lemma** Let $=_{\Sigma,E}$ be a sort-preserving and congruence-closed congruence.

Then: $\forall\sigma \in SUB_\Sigma\ \forall\tau \in SUB_{\bar\Sigma}:\ \tau \geq_{\Sigma,E} \sigma\,[V] \Rightarrow \tau \in SUB_\Sigma$.

**Proof.** There exists a $\lambda \in SUB_\Sigma$ such that $\tau =_{\Sigma,E} \lambda\sigma\ [V]$. Lemma 6.15 implies $\tau \in SUB_\Sigma$, since $\lambda\sigma \in SUB_\Sigma$. ■

**6.17 Example.** Let Let $=_{\Sigma,E}$ be a sort-preserving and congruence-closed congruence. Then for $\sigma,\tau \in SUB_\Sigma$ the implication $\sigma \leq_{\bar\Sigma,E} \tau\,[V] \Rightarrow \sigma \leq_{\Sigma,E} \tau\,[V]$ may be false:

It suffices to consider the empty theory $\mathcal{E}$ and $\Sigma := \{B \sqsubset A,\ C \sqsubset A\}$. Let $\sigma := \{x_A \leftarrow y_B\}$ and $\tau := \{x_A \leftarrow z_C, y_B \leftarrow z_C\}$. Then we have $\sigma \not\leq_{\Sigma,E} \tau\ [V]$, but $\{y_B \leftarrow z_C\}\sigma = \tau$, hence $\sigma \leq_{\bar\Sigma,E} \tau$. ■

**6.18 Lemma.** Let $=_{\Sigma,E}$ be a sort-preserving congruence.

Let $\rho_1, \rho_2 \in SUB_\Sigma$ be idempotent $\bar\Sigma$-renamings with $DOM(\rho_1) = DOM(\rho_2) = W$.

Then $\rho_1 \leq_{\Sigma,E} \rho_2\ [W] \Rightarrow \rho_1 \leq_\Sigma \rho_2\ [W]$.

**Proof.** There exists a $\lambda \in SUB_\Sigma$ such that $\lambda\rho_1 =_{\Sigma,E} \rho_2\,[W]$.

The substitution $\lambda' := \{\rho_1 x \leftarrow \rho_2 x \mid x \in W\}$ is well defined and satisfies:

$\lambda'\rho_1 = \rho_2\,[W]$ and $\lambda' \in SUB_\Sigma$ since $S(\rho_2 x) = S(\lambda\rho_1 x) \sqsubseteq S(\rho_1 x)$ and $\rho_1 x$ and $\rho_2 x$ are variables. Hence $\rho_1 \leq_\Sigma \rho_2\,[W]$ holds. ■

In the following we give an interesting consequence of the sort-preservation in a regular, elementary signature. In this case the equational theory can be lifted to the set of sorts. That means the set of sorts provides an algebra that satisfies the equational theory:

**6.19 Definition.** If the congruence $=_{\Sigma,E}$ is sort-preserving and the signature is regular and elementary, then we define the following theory S-$TH_{\mathcal{E}}$ on the set $S_\Sigma$.

Every declaration $f{:}S_1 \times \ldots \times S_n \rightarrow S$ is translated into $f(S_1,\ldots,S_n) = S$. ■

**6.20 Proposition.** If the congruence $=_{\Sigma,E}$ is sort-preserving, congruence-closed and the signature is regular and elementary, then the theory S-$TH_{\mathcal{E}}$ on $S_\Sigma$ has the following properties:

i) Sorts are not identified

ii) For every equation $s =_{\Sigma,E} t$ there holds a corresponding equation $s^* = t^*$ over $S_\Sigma$, where $s^*$ and $t^*$ are obtained from s,t by replacing variables and constants by their respective sorts. (Note that $s^*$ is exactly $\mathbf{LS}_\Sigma(s)$ in this theory.)

iii) This theory on $S_\Sigma$ is compatible with the subsort-ordering:

$S_i \sqsubseteq R_i$, $i=1,\ldots,n$ implies $f(S_1,\ldots,S_n) \sqsubseteq f(R_1,\ldots,R_n)$

**Proof.** i) follows from the definition, ii) follows from sort-preservation and iii) follows from regularity of $\Sigma$. ■

## 7. Conservative Transformations.

Given two specifications it is a natural question to ask if they specify the same problem or if they are in some sense equivalent. For example, the specification $\{A \sqsubset B, a{:}A, b{:}B, a=b\}$ is semantically equivalent to $\{A \sqsubset B, a{:}A, b{:}A, a=b\}$. That means they specify the same standard model although their signatures and their free term algebras are different.
In order to be able to compare such specifications, we introduce the notion of transformations, where each transformation H should be conservative, that is H transforms (un)satisfiable specifications into (un)satisfiable ones. The notion of conservative transformations will play a crucial role in proving that the sort-generation algorithm in part VI is correct. Conservative extensions of theories in the sense of [Sh67] have the embedding mapping of theories as conervative transformation. Our notion of conservative transformation of signatures corresponds to those conservative extensions.
We emphasize that in this paragraph the assumption I.4.11 i), that sorts are not empty, is important, since most of the theorems are no longer valid without it.

**7.1 Definition.** Let $\mathcal{S}_1 := (\Sigma_1, CS_1)$ and $\mathcal{S}_2 := (\Sigma_2, CS_2)$ be specifications and let $H: \mathcal{S}_1 \to \mathcal{S}_2$ be a total mapping. i.e., $H: \mathbf{S}_{\Sigma 1} \to \mathbf{S}_{\Sigma 2}$, $H: \mathbf{P}_1 \to \mathbf{P}_2$, $H: \mathbf{F}_1 \to \mathbf{F}_2$. The mapping H extends in an obvious way to term declarations, subsort declarations, atoms, literals, clauses and clause sets.
We say H is a **well-sorted transformation**, iff the following is satisfied:
i) $H: \mathbf{F}_1 \to \mathbf{F}_2$ and $H: \mathbf{P}_1 \to \mathbf{P}_2$ is an injection.
ii) $H: \mathbf{TD}_{\Sigma 1} \to \mathbf{TD}_{\Sigma 2}$ and $H: \mathbf{SD}_{\Sigma 1} \to \mathbf{SD}_{\Sigma 2}$ are total mappings.
iii) $H(CS_1) = CS_2$ ■

We may use the notion of well-sorted transformations for signatures (without specifications) as well as for specifications.

**7.2 Lemma.** For a well-sorted transformations $H: \mathcal{S}_1 \to \mathcal{S}_2$ we have
i) $\forall R,S \in \mathbf{S}_{\Sigma 1}: R \sqsubseteq_1 S \Rightarrow H(R) \sqsubseteq_2 H(S)$.
ii) $t \in \mathbf{T}_{\Sigma 1,S} \Rightarrow H(t) \in \mathbf{T}_{\Sigma 2,H(S)}$.
**Proof.** Follows from the fact that H is defined for every sort, every subsort declaration and every term declaration in $\Sigma_1$. ■

Note that Lemma 7.2 implies that the image H(A) of every well-sorted atom A is well-sorted.

**7.3 Definition.** We say a well-sorted transformation $H: \mathcal{S}_1 \to \mathcal{S}_2$ is a **conservative** transformation, iff the following holds:

$\mathcal{S}_1$ has a $\Sigma_1$-model iff $\mathcal{S}_2$ has a $\Sigma_2$-model.

Furthermore we say a well-sorted transformation of signatures is **conservative**, iff for every clause set $CS_1$, the transformation $H:(\Sigma_1,CS_1) \to (\Sigma_2,CS_2)$ is conservative. ■

Now given a $\Sigma_1$-structure $\mathcal{A}$, we investigate how to construct the $\Sigma_2$-quasi-structure $H(\mathcal{A})$. Note that (in general) $H(\mathcal{A})$ need not be a $\Sigma_2$-structure.

Let $\mathcal{A}$ be a $\Sigma_1$-structure and let $H: \mathbf{F}_1 \to \mathbf{F}_2$ be bijective. Then we define the $\Sigma_2$-quasi-structure $\mathcal{B} = H(\mathcal{A})$ as follows:

i) $B := A$, (i.e., the carriers are the same)

ii) $(H(S))_B := S_A$.

iii) $(H(f))_B := f_A$,

iii) $(H(P))_B := P_A$ for $\Sigma_1$-predicate symbols P and $(H(P))_B := \emptyset$ otherwise.

iv) If $S \in S_{\Sigma 2} - H(S_{\Sigma 1})$, then $S_B := \cup \{R_B \mid R \sqsubseteq_2 S \text{ and } R \in H(S_{\Sigma 1})\}$.

The case where $H: \mathbf{F}_1 \to \mathbf{F}_2$ is not bijective is handled separately in a proposition.

We say that the sort structure $<\mathbf{S}_{\Sigma,1}, \sqsubseteq_1>$ is **embedded** into $<\mathbf{S}_{\Sigma,2}, \sqsubseteq_2>$ with embedding H iff,

i) $H:\mathbf{S}_{\Sigma 1} \to \mathbf{S}_{\Sigma 2}$ is injective

ii) For all $R,S \in \mathbf{S}_{\Sigma 1}$ we have $R \sqsubseteq_1 S \Leftrightarrow H(R) \sqsubseteq_2 H(S)$.

iii) For every sort $S \in \mathbf{S}_{\Sigma 2}$ there exists a sort $R \in \mathbf{S}_{\Sigma 1}$ with $H(R) \sqsubseteq_2 S$.

We give a criterion for $H(\mathcal{A})$ to be a $\Sigma_2$-algebra.

**7.4 Proposition.** Let $H: \Sigma_1 \to \Sigma_2$ be a transformation:

Let $H: \mathbf{F}_1 \to \mathbf{F}_2$ and $H: \mathbf{TD}_1 \to \mathbf{TD}_2$ be bijective and let $<\mathbf{S}_{\Sigma,1}, \sqsubseteq_1>$ be embedded into $<\mathbf{S}_{\Sigma,2}, \sqsubseteq_1>$.

Then i) H is a well-sorted transformation.

ii) For every $\Sigma_1$-algebra A its image $H(A)$ is a $\Sigma_2$-algebra.

**Proof.** The transformation H is well-sorted, since all conditions of Definition 7.1 are satisfied.

Note that $H(\mathbf{T}_{\Sigma 1,S}) = \mathbf{T}_{\Sigma 2,H(S)} \cap H(\mathbf{T}_{\Sigma 1})$, since H is injective on sorts and term declarations. Furthermore the above embedding condition enforces that the nonempty sort assumption for $\Sigma_2$ is satisfied.

Let A be a $\Sigma_1$-algebra A and let H(A) be its image. First of all H(A) is a $\Sigma_2$-quasi-algebra. Furthermore it follows trivially from the above definition of H(A), that $R \sqsubseteq_1 S$ implies $R_{H(A)} \subseteq S_{H(A)}$. In order to prove condition I.6.1 ii) let H(t):H(S) be a term-declaration in $\Sigma_2$ and let $\varphi_2: \mathbf{V}_{\Sigma 2} \to H(A)$ be a partial $\Sigma_2$-assignment with $\mathbf{V}(H(t)) \subseteq \mathcal{D}(\varphi_2)$. Let

$\varphi_1$: $V_{\Sigma 1} \to H(A)$ be the partial $\Sigma_2$-assignment defined by $\varphi_1 x := \varphi_2(H(x))$. Since A is a $\Sigma_1$-algebra, we have that $\varphi_1$ is defined on t and $\varphi_1(t) \in S_A$. Since $\varphi_1(t) = \varphi_2(H(t))$ and $S_A = H(A)_{H(A)}$ the condition I.6.1 ii) is satisfied. We conclude that H(A) is a $\Sigma_2$-algebra. ■

In the following we give some useful sufficient criteria for a transformation to be conservative. The method described here will be used extensively to show that the transformations of the sort generating process in part VI are conservative transformations.

**7.5 Lemma.** Let $H:\Sigma_1 \to \Sigma_2$ be a well-sorted transformation and let A be a $\Sigma_1$-algebra such that H(A) is a $\Sigma_2$-algbera. Then the following holds:

i) For every $\Sigma_1$-homomorphism $\varphi_1 : \mathbf{T}_{\Sigma 1} \to A$ there exists a $\Sigma_2$-homomorphism $\varphi_2 : \mathbf{T}_{\Sigma 2} \to H(A)$ with $\varphi_1(t) = \varphi_2(H(t))$.

ii) For every $\Sigma_2$-homomorphism $\varphi_2 : \mathbf{T}_{\Sigma 2} \to H(A)$ there exists a $\Sigma_1$-homomorphism $\varphi_1 : \mathbf{T}_{\Sigma 1} \to A$ with $\varphi_1(t) = \varphi_2(H(t))$ for all $t \in \mathbf{T}_{\Sigma 1}$.

**Proof.**

i) Let $\varphi_1 : \mathbf{T}_{\Sigma 1} \to A$ be a $\Sigma_1$-homomorphism. Let $\varphi_2$: $\mathbf{V}_{\Sigma 2} \to \mathbf{T}_{\Sigma 2}$ be a mapping with $\varphi_2(H(x)) := \varphi_1(x)$. This is a partial $\Sigma_2$-assignment, since the denotations for sorts in $\mathbf{S}_{\Sigma 1}$ and $H(\mathbf{S}_{\Sigma 1})$ are the same. Since sorts are not empty by Corollary I.6.5 and assumption I.4.11, we can extend the partial $\Sigma_2$-assignment $\varphi_2$ to a total $\Sigma_2$-assignment. We can further extend $\varphi_2$ to a total $\Sigma_2$-homomorphism $\varphi_2$: $\mathbf{T}_{\Sigma 2} \to A$, since $\mathbf{T}_{\Sigma 2}$ is a free $\Sigma_2$-algebra. The interpretation of functions over A and H(A) is the same, hence $\varphi_2(H(t)) = \varphi_1(t)$ for all $t \in \mathbf{T}_{\Sigma 1}$.

ii) Let $\varphi_2 : \mathbf{T}_{\Sigma 2} \to A$ be a $\Sigma_2$-homomorphism. Define $\varphi_1$: $\mathbf{T}_{\Sigma 1} \to A$ by $\varphi_1(x) := \varphi_2(H(x))$ for all $x \in \mathbf{T}_{\Sigma 1}$. Similar as in the proof of part i) this is a $\Sigma_1$-assignment and can be extended to a $\Sigma_1$-homomorphism, since $\mathbf{T}_{\Sigma 1}$ is a free $\Sigma_1$-algebra. Furthermore $\varphi_2(H(t)) = \varphi_1(t)$ for all $t \in \mathbf{T}_{\Sigma 1}$, since the interpretation of functions over A and H(A) is the same. ■

**7.6 Theorem.** Let $H:\Sigma_1 \to \Sigma_2$ be a well-sorted transformation.

i) Let A be a $\Sigma_1$-model for $CS_1$ and let H(A) be a $\Sigma_2$-algebra.
Then H(A) is a $\Sigma_2$-model of $CS_2$.

ii) Let B be a $\Sigma_2$-model for $CS_2$ and let A be a $\Sigma_1$-algebra, such that H(A) = B.
Then A is a $\Sigma_2$-model of $CS_2$.

**Proof.** i) Let A be a $\Sigma_1$-model for $CS_1$ and let H(A) be a $\Sigma_2$-structure.
We show that H(A) is a $\Sigma_2$-model for $CS_2$. We have to show that all clauses are satisfiedin H(A). Let $\varphi_2$ be a $\Sigma_2$-assignment and let $H(C) \in CS_2$ be a clause. Then by Lemma 7.5 i) there exists a $\Sigma_1$-homomorphism $\varphi_1 : \mathbf{T}_{\Sigma 1} \to A$ such that $\varphi_1(t) = \varphi_2(H(t))$ for all $t \in \mathbf{T}_{\Sigma 1}$. Since C is valid under the interpretation $\varphi_1$, and $\varphi_1(C) = \varphi_2(H(C))$ we have that H(C) is

valid under the interpretation $\varphi_2$.

ii) the proof is similar using part ii) of Lemma 7.5.

As a first application we prove a corollary that we can extend the signature by adding supersorts of given sorts:

**7.7 Corollary.** Let $\mathcal{S}_1 := (\Sigma_1, CS_1)$ and $\mathcal{S}_2 := (\Sigma_2, CS_2)$ be specifications and let H: $\mathcal{S}_1 \rightarrow \mathcal{S}_2$ be a well-sorted transformation satisfying the conditions of Proposition 7.4.

Then H is a conservative transformation of signatures.

**Proof.** Follows from Proposition 7.4 and 7.6. ■

We formulate the special case that we can add a greatest sort to the signature as a corollary:

**7.8 Corollary.** Let $\mathcal{S} := (\Sigma, E)$ be an equational specification. Then we can always add a greatest sort TOP satisfying the conditions of Proposition 7.4, such that the the instance relation for $\Sigma$-substitutions does not change.

**Proof.** Follows from Corollary 7.7. The new term algebra is the old one plus variables of sort TOP. The only possible new components are of the form $\{x_{TOP} \leftarrow t\}$. Hence the new substitutions do not influence the instance relation on old ones. ■

**7.9 Proposition.** Let $\mathcal{S}_1 = (\Sigma_1, CS_1)$ and $\mathcal{S}_2 = (\Sigma_2, CS_2)$ be specifications, where $\Sigma_1$ and $\Sigma_2$ are regular signatures and $CS_1$ does not contain an equality-literal. Let H: $\Sigma_1 \rightarrow \Sigma_2$ be a well-sorted transformation which only increases the set of functions, i.e.,

H: $\mathbf{F}_1 \rightarrow \mathbf{F}_2$, H: $\mathbf{TD}_1 \rightarrow \mathbf{TD}_2$ are injective and H:$\mathbf{S}_{\Sigma 1} \rightarrow \mathbf{S}_{\Sigma 2}$ and H:$\mathbf{SD}_{\Sigma 1} \rightarrow \mathbf{SD}_{\Sigma 2}$ are bijective. Furthermore assume that all new term declarations have a toplevel function symbol from $\mathbf{F}_{\Sigma 2}$.

Then H is conservative (as transformation of specifications).

**Proof.** One direction is trivial: If $A_2$ is a $\Sigma_2$-model of $CS_2 = H(CS_1)$, then we obtain a $\Sigma_1$-model of $CS_1$ by simply forgetting the superfluous function symbols.

For the other direction we show that if $CS_2$ is $\Sigma_2$-unsatisfiable, then $CS_1$ is also $\Sigma_1$-unsatisfiable.

Due to the Herbrand-theorem 11.2 there exists an unsatisfiable, finite set $CS_{2,gr}$ of $\Sigma_2$-ground instances of $CS_2$. If there is no occurrence of a new function symbol in $CS_{2,gr}$, then $CS_{2,gr}$ serves also as an unsatisfiable, finite set $CS_{1,gr}$ of $\Sigma_1$-ground instances of $CS_2$.

Assume by contradiction that $CS_{2,gr}$ contains a minimal number of occurrences of new function symbols. Let t be a term occurring in $CS_{2,gr}$ with a new toplevel function symbol and with a maximal term depth. Since t has maximal term depth, for every occurrence of t in $CS_{2,gr}$ the function symbols above it are old ones. (Here the precondition on the toplevel function symbols of new declarations is used.) Since $\Sigma_2$ is regular, we can choose a $\Sigma_1$-term t' with $\mathbf{LS}(t') \sqsubseteq \mathbf{LS}(t)$. Replacing every occurrence of t in $CS_{2,gr}$ by t' gives a new set of ground clauses $CS'_{2,gr}$. The set $CS'_{2,gr}$ is well-sorted, since we have assumed that there are no new term declarations with an old toplevel function symbol. Furthermore $CS'_{2,gr}$ is contradictory, since it represents the same propositional clause set as $CS_{2,gr}$. This is a contradiction to the minimal choice of $CS_{2,gr}$. ■

It is not possible to drop the requirements of Proposition 7.9:

**7.10 Counterexamples.**

i) If we add declarations in $\Sigma_2$ with old toplevel function symbols, then Proposition 7.9 may be false:
Let $\Sigma_1 := \{B \sqsubset A, b{:}B, g{:}A \rightarrow A\}$ and let $CS_1 := \{\{P(x_B)\}, \{-P(g(y_A))\}\}$ . This clause set has a $\Sigma_1$-model, since $x_B$ and $g(y_A)$ are not $\Sigma_1$-unifiable.
Let $\Sigma_2 := \Sigma_1 \cup \{g(f(z_A)){:}B\}$. However, this term declaration allows to unify the terms $x_B$ and $g(y_A)$ with the unifier $\{y_A \leftarrow f(z_A), x_B \leftarrow g(f(z_A))\}$, hence the clause set $CS_1$ is contradictory with respect to $\Sigma_2$. ❑

ii) If the signature $\Sigma_2$ is not regular, then Proposition 7.9 may be false:
Let $\Sigma_1 := \{B \sqsubset A, C \sqsubset A , b{:}B, c{:}C\}$ and let $CS_1 := \{\{P(x_B)\}, \{-P(y_C)\}\}$. This clause set has a $\Sigma_1$-model. If we add the term declaration $g{:}B \rightarrow B$, $g{:}B \rightarrow C$ , i.e., $\Sigma_2 := \Sigma_1 \cup \{g{:}B \rightarrow B, g{:}B \rightarrow C\}$, then $x_B$ and $y_C$ are unifiable with unifier $\{x_B \leftarrow g(z_B), y_C \leftarrow g(z_B)\}$, hence the clause set $CS_1$ is contradictory with respect to $\Sigma_2$. ❑

iii) If there are equations in the clause set, then Proposition 7.9 may be false:
Let $\Sigma_1 := \{B \sqsubset A, C \sqsubset A , D \sqsubset A, E \sqsubset A, b{:}B, c{:}C, d{:}D, e{:}E\}$ and let $CS_1 :=$ $\{\{x_D \neq y_E\}, \{b=c\}\}$. This clause set has a $\Sigma_1$-model. If we add the term declarations $g{:}B \rightarrow D$, $g{:}C \rightarrow E$ , i.e., $\Sigma_2 := \Sigma_1 \cup \{g{:}B \rightarrow D, g{:}C \rightarrow E\}$, then we have $g(b) = g(c)$. However, g(b) is of sort D and g(c) is of sort E, hence $x_D \neq y_E$ implies $g(b) \neq g(c)$, which is a contradiction. ■

**7.11 Proposition.** Let $\Sigma$ be a signature. Then factoring out the equivalence of sorts is a conservative transformation of signatures.

**Proof.** Note that in all $\Sigma$-models M the denotation of equivalent sorts is the same, i.e. $A \sqsubseteq B$ and $B \sqsubseteq A$ implies $A_M = B_M$. The proof is straightforward and uses the same ideas as the proof of 7.6. ■

This proposition means that we can assume that the order on sorts is a partial ordering. Our next aim is to show that we can also assume that the sort-structure is a semilattice. We show how to embed an arbitrary finite partial ordering into a semilattice:

**7.12 Lemma.** Let $<S_a, \sqsubseteq_a>$ be a partial ordering on the finite set $S_1$. Then there exists a semilattice $<S_b, \sqsubseteq_b>$ such that $<S_a, \sqsubseteq_a>$ is embedded into $<S_b, \sqsubseteq_b>$.

**Proof.** We define the set $S_b$ as follows:

$S_b := \{M \neq \emptyset \mid M = [-\infty, S_1] \cap \ldots \cap [-\infty, S_k]$ for $S_1, \ldots, S_k \in S_a\}$. We allow also the empty intersection , i.e,. we assume that the whole set $S_a$ is an element of $S_b$. We define the embedding function $H: S_a \to S_b$ as $H(S) := [-\infty, S]$ . Furthermore we define the ordering $\sqsubseteq_b$ to be the subset ordering on $S_b$.

Obviously H is injective, since $[-\infty, R] = [-\infty, S]$ implies $R = S$ as $\sqsubseteq_a$ is antisymmetric.

i) $R \sqsubseteq_a S \Leftrightarrow H(R) \sqsubseteq_b H(S)$ for all $R, S \in S_a$:

Obviously $R \sqsubseteq_a S$ is equivalent to $[-\infty, R] \subseteq [-\infty, S]$ , by the definition of $S_b$.

ii) For every sort $S \in S_b$ there exists a sort $R \in S_a$ with $H(R) \sqsubseteq_b S$:

This holds, since all elements of $S_b$ are lower segments and hence for every $M \in S_b$ and every $S \in M$ we have $[-\infty, S] \sqsubseteq_b M$.

Obviously, for every $M_1, M_2 \in S_b$ we either have $M_1 \cap M_2 = \emptyset$ or $M_1 \cap M_2 \in S_b$. This means that $<S_b, \sqsubseteq_b>$ is a semilattice. ■

Note that the construction in Lemma 7.12 is optimal in the sense that a minimal number of new sorts is generated. The argument is that in an arbitrary lattice in which $<S, \sqsubseteq>$ is embedded, the intersection construction of Lemma 7.12 is also possible and shows that the semilattice constructed in Lemma 7.12 is a subsemilattice.

**7.13 Corollary.** For every finite signature $\Sigma$ the embedding of the sort structure $<S_\Sigma, \sqsubseteq>$ into the finite semilattice as constructed in Proposition 7.12 is a conservative transformation. ■

In general this result increases the efficiency of a unification procedure, since the number of unifiers can be reduced. However, in the worst case it may be possible that the number of sorts to be generated is exponential:

**7.14 Proposition.** The embedding of a sort structure $<S_\Sigma, \sqsubseteq>$ into a finite semilattice may require an exponential number of new sorts.

**Proof.** Consider the following sort structure: Let $A_i$, $i = 1, \ldots, n$ and $B_i$ , $i = 1, \ldots, n$ be sorts such that the relations are $A_i \sqsupseteq B_j$ iff $i \neq j$. The construction of Lemma 7.12 yields that every nonempty subset of $\{B_1, \ldots, B_n\}$ corresponds to a sort in the completion lattice. These are $2^n - 1$ sorts. On the other hand, the above construction gives an exponential upper bound, since $\mathcal{P}(S)$ is sufficient for a completion. ■

Corollary 7.13 justifies the assumption that sort-structures are semilattices. It has as a consequence (see part III) that the number of unifiers can be reduced by a preprocessing step, which transforms the sort-structure into a semilattice. In the case where this transformation is exponential, there are two remedies to the situation: the first is to use a logic in which sorts change dynamically [IS85] or else we assume that the sort-strucure is completed, but perform a lazy computation of the completion, i.e. we compute the needed sorts at unification time.

## 8. R-systems.

The definitions of this paragraph are only used here, only the final result will be used outside of this paragraph.

Consider the situtation, where the sets $\mathbf{T}_\Sigma$ and $SUB_\Sigma$ are given, or where we only have an algorithm for distinguishing well-sorted terms and substitutions from ill-sorted ones, but no term declarations are given. We show, that some sensible restrictions enforce that the notion of well-sortedness is generated by an order-sorted signature with term-declarations.

A similar way to define sorts starting with a relation on variables is used in the Σ-logic of A. Oberschelp [Ob62].

Throughout this paragraph we assume that an unsorted signature $\overline{\Sigma}$ and restricted sets of terms $\mathbf{T}_R \subseteq \mathbf{T}_{\overline{\Sigma}}$ and substitutions $SUB_R \subseteq SUB_{\overline{\Sigma}}$ are given.

The following conditions should hold for the (restricted) **R-system** $(\mathbf{T}_R, SUB_R)$:

**R-i)** $\mathbf{T}_R$ is subterm-closed and $\mathbf{C}_{\overline{\Sigma}}, \mathbf{V}_{\overline{\Sigma}} \subseteq \mathbf{T}_R$.

**R-ii)** $SUB_R$ is a monoid with $SUB_R \circ (\mathbf{T}_R) = \mathbf{T}_R$.

**R-iii)** $\forall W \subseteq \mathbf{V}_{\overline{\Sigma}}, \forall \sigma \in SUB_R : \sigma_{|W} \in SUB_R$.

**R-iv)** $\forall t \in \mathbf{T}_R\ \exists x \in \mathbf{V}_{\overline{\Sigma}}$: $\{x \leftarrow t\} \in SUB_R$.

**R-v)** For every variable x there exists a ground term $t_{gr}$ with $\{x \leftarrow t_{gr}\} \in SUB_R$.

We define subsumption with respect to $\mathbf{T}_R$ and $SUB_R$:

**8.1 Definition.** Let $s,t \in \mathbf{T}_{\overline{\Sigma}}$. Then

i) $s \leq_R t \ :\Leftrightarrow\ \exists \lambda \in SUB_R: \lambda s = t$.

ii) $s \equiv_R t \ :\Leftrightarrow\ s \leq_R t$ and $t \leq_R s$. ■

**8.2 Lemma.** $\leq_R$ is a quasi-ordering .

**Proof.** We have $t \leq_R t$ for all $t \in \mathbf{T}_{\overline{\Sigma}}$, since $Id \in SUB_R$.

Let $r \geq_R s \geq_R t$. Then there exist $\lambda,\sigma \in SUB_R$ with $r = \lambda s$ and $s = \sigma t$. We have $\lambda \circ \sigma \in SUB_R$ since $SUB_R$ is a monoid, hence $r = \lambda \circ \sigma t$ and $r \geq_R t$. ■

**8.3 Lemma.** For $x \in V_{\overline{\Sigma}}$ and $t \in T_{\overline{\Sigma}}$: $t \geq_R x \Leftrightarrow \{x \leftarrow t\} \in SUB_R$.

**Proof.**

"$\Rightarrow$": Let $\lambda \in SUB_R$ with $\lambda x = t$. Then $\{x \leftarrow t\} = \lambda_{|\{x\}} \in SUB_R$ by **R**-iii.

"$\Leftarrow$": trivial. ■

The last condition for an **R**-system ($T_R$, $SUB_R$) is :

R-vi) $\forall x \in V_{\overline{\Sigma}}$: the equivalence class $[x]_{\equiv R}$ is an (countably) infinite set.

**8.4 Definition.** An **R-system** consists of an unsorted signature $\overline{\Sigma}$, a set of terms $T_R$ and a set of substitutions $SUB_R$ such that condition R-i) - R-vi) are satisfied. ■

Obviously $[x]_{\equiv R} \subseteq V_{\overline{\Sigma}}$ for all variables.

The notion of **R**-systems is sensible:

**8.5 Proposition.** Signatures with term declarations generate **R**-systems.

**Proof.** The verification of every condition is straightforward.■

We define the notion of sorts in **R**-systems. Here sorts are defined as sets, but we could just as well have a sort-symbol for every sort. We use the symbol $\Sigma$ to indicate the signature to be defined.

**8.6 Definition.** The set of sorts with a partial ordering and the sort of a term is defined as follows:

i) $S_\Sigma := \{ [x]_{\equiv R} \mid x \in V_{\overline{\Sigma}} \}$

ii) The ordering on $S_\Sigma$ is: $S_1 \sqsubseteq_R S_2 :\Leftrightarrow x_1 \geq_R x_2$, where $S_1 = [x_1]_{\equiv R}$ and $S_2 = [x_2]_{\equiv R}$.

iii) For $t \in T_{\overline{\Sigma}}$: $S_\Sigma(t) := \{ [x]_{\equiv R} \mid \{x \leftarrow t\} \in SUB_R\}$. ■

**8.7 Proposition.**

i) $\sqsubseteq_R$ is a partial ordering on **S**.

ii) $\forall t \in T_{\overline{\Sigma}}$: $S_\Sigma(t) \neq \emptyset \Leftrightarrow t \in T_R$.

iii) For all $x \in V_{\overline{\Sigma}}$ and all $t \in T_R$: $S_\Sigma(x) \subseteq S_\Sigma(t) \Leftrightarrow \{x \leftarrow t\} \in SUB_R$.

**Proof.**

i) $\sqsubseteq_R$ is well-defined, since $x_1 \equiv_R x_2$, $y_1 \equiv_R y_2$ and $x_1 \leq_R y_1$ imply $x_2 \leq_R y_2$ by the transitivity of $\leq_R$. That $\sqsubseteq_R$ is a partial ordering on $S_\Sigma$ follows from the fact that $\leq_R$ is a quasi-ordering on $V_{\overline{\Sigma}}$

ii) "$\Rightarrow$": If $S_\Sigma(t) \neq \emptyset$ then $\{x \leftarrow t\} \in SUB_R$ for some variable x.

Hence by **R**-ii : $t \in T_R$.

"$\Leftarrow$": Follows from **R**-iv.

iii) $\mathbf{S}_\Sigma(x) \subseteq \mathbf{S}_\Sigma(t) \Leftrightarrow [x]_{\equiv R} \in \mathbf{S}_\Sigma(t) \Leftrightarrow \{x \leftarrow t\} \in \mathrm{SUB}_R$. ■

**8.8 Definition.** $\rho \in \mathrm{SUB}_R$ is called a **$\mathrm{SUB}_R$-renaming,** iff $\rho$ is a renaming and $\forall x \in \mathbf{V}\ \rho x \equiv_R x$.

The existence of sufficiently many $\mathrm{SUB}_R$-renamings is not obvious and has to be proved:

**8.9 Lemma.** Let $W_1 \subseteq W_2 \subseteq V_\Sigma$ be finite sets of variables.

Then there exists a $\mathrm{SUB}_R$-renaming $\rho \in \mathrm{SUB}_R$ such that $\mathrm{DOM}(\rho) = W_1$ and $I(\rho) \cap W_2 = \emptyset$.

**Proof.** Let $W_1 := \{x_1,...,x_n\}$. Since $[x_i]_{\equiv R}$ contains infinitely many variables (**R**-vi), we can choose variables $y_i \in [x_i]_{\equiv R} - W_2$, such that all $y_i$ are different.

Lemma 8.3 implies $\rho_i := \{x_i \leftarrow y_i\} \in \mathrm{SUB}_R$. We have $\rho_i \circ \rho_j = \rho_j \circ \rho_i$ for $i \neq j$ and define $\rho := \rho_1 \circ ... \circ \rho_n \in \mathrm{SUB}_R$. The result $\rho$ is the desired $\mathrm{SUB}_R$-renaming. ■

**8.10 Lemma.** Let $x_i$ be different variables and let $\{x_i \leftarrow t_i\} \in \mathrm{SUB}_R$ for $i=1,...,n$

Then $\{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\} \in \mathrm{SUB}_R$.

**Proof.** There exist $\mathrm{SUB}_R$-renamings $\rho_i$ $i=1,...,n$, such that $\mathrm{DOM}(\rho_i) = \mathbf{V}(t_i)$ and $I(\rho_i)$ consists of variables (see Lemma 8.9).

The following reasoning relies on the trick that a substitution can be made idempotent by renamings and that idempotent substitutions are equal to the composition of their components.

We have $\rho_i \circ \{x_i \leftarrow t_i\} \in \mathrm{SUB}_R$. Let $\sigma_i := \rho_i \circ \{x_i \leftarrow t_i\}_{|\{x_i\}}$. Then $\sigma_i \in \mathrm{SUB}_R$, hence $\sigma_1 \circ ... \circ \sigma_n \in \mathrm{SUB}_R$. Furthermore $\rho_1^- \circ ... \circ \rho_n^- \circ \sigma_1 \circ ... \circ \sigma_n \in \mathrm{SUB}_R$, where $\rho_i^-$ is the converse renaming of $\rho_i$.

We compute:

$\rho_1^- \circ ... \circ \rho_n^- \circ \sigma_1 \circ ... \circ \sigma_n x_i =$

$= \rho_1^- \circ ... \circ \rho_n^- \circ \sigma_i x_i$ — $\mathrm{DOM}(\sigma_i) \cap \{x_1,...,x_n\} = \{x_i\}$ and $I(\sigma_i) \cap \mathrm{DOM}(\sigma_j) = \emptyset$ for $i \neq j$.

$= \rho_1^- \circ ... \circ \rho_n^- \circ \rho_i t_i$

$= \rho_i^c \circ \rho_i t_i$ — $\mathrm{DOM}(\rho_j^-) \cap I(\rho_i) = \emptyset$ for $i \neq j$ and $\mathrm{DOM}(\rho_j^-) \cap I(\rho_i^-) = \emptyset$ for $i \neq j$.

$= t_i$

Hence $\rho_1^- \circ ... \circ \rho_n^- \circ \sigma_1 \circ ... \circ \sigma_{n\,|\{x_1,...,x_n\}} = \{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\} \in \mathrm{SUB}_R$ ■

**8.11 Theorem:** $\mathbf{S}_\Sigma$ is a sort-assignment, $\mathbf{T}_\Sigma = \mathbf{T}_R$ and $\mathrm{SUB}_\Sigma = \mathrm{SUB}_R$.

**Proof.** We check the conditions of Definition 4.1.

i) Obviously $\mathbf{S}_\Sigma$ maps terms onto upper segments in $\mathbf{S}_\Sigma$.

ii) Proposition 8.7 implies that $\mathbf{T}_\Sigma = \mathbf{T}_\mathbf{R}$.

iii) $\mathrm{SUB}_\Sigma = \mathrm{SUB}_\mathbf{R}$:

$\sigma \in \mathrm{SUB}_\mathbf{R}$

$\Leftrightarrow \forall x \in \mathbf{V}_\Sigma: \{x \leftarrow \sigma x\} \in \mathrm{SUB}_\mathbf{R}$ — Lemma 8.10 and **R**-iii)

$\Leftrightarrow \forall x \in \mathbf{V}_\Sigma: [x]_{\equiv R} \in \mathbf{S}_\Sigma(\sigma x)$ — Proposition 8.7 iii)

$\Leftrightarrow \forall x \in \mathbf{V}_\Sigma: \mathbf{S}_\Sigma(x) \subseteq \mathbf{S}_\Sigma(\sigma x)$

$\Leftrightarrow \sigma \in \mathrm{SUB}_\Sigma$.

iv) The other conditions follow directly from **R**-i) - **R**-vi).■

## 9. Sort-Preserving Congruences.

We are interested in congruences, which are sort-preserving and deduction-closed. In this paragraph we show that every sort-assignment and a congruence on terms can be conservatively transformed into a sort-assignment and sort-preserving congruence.
However, the new sort-assignment may not be effectively computable. For practical applications, the equational theory should be decidable. If the equational theory has normalforms, then the new sort for terms can be defined as the sort of their normalform, provided the normalform has a minimal sort in its equivalence class. This is particularly useful if the term rewriting system is weakly sort-decreasing and canonical.
Due to paragraph 2 we can assume that the congruence is deduction-closed.
The following theorem introduces a new sort of a term t that corresponds to the union of the sets $\mathbf{S}_\Theta(t) := \cup\{\mathbf{S}_\Sigma(s) \mid s =_{\Sigma,E} t\}$. We will refer to this notion also as an**E-semantical sort** or for short as a semantical sort of t. This sort can be seen as the sort of a term in the quotient algebra of $\mathbf{T}_\Sigma$ modulo the equational theory. However, there is the problem that with this definition a variable may not have a unique least sort. The construction in the proof is by using the abstract notion of R-systems introduced in the last paragraph.

**9.1 Theorem.** Let $=_{\Sigma,E}$ be a deduction-closed congruence. Let CS be a clause set with $E \subseteq CS$.

Then there exists a mapping $\mathbf{S}_\Theta: \mathbf{T}_\Sigma \rightarrow \mathbf{S}_\Sigma$, such that

a) $\mathbf{S}_\Theta$ is a sort-assignment.

b) $\mathbf{T}_\Theta = \mathbf{T}_\Sigma$ and $\mathrm{SUB}_\Sigma \subseteq \mathrm{SUB}_\Theta$.

c) The generated $\mathrm{SUB}_\Theta$-invariant congruence $=_{\Theta,E}$ is the same relation as $=_{\Sigma,E}$ on $\mathbf{T}_\Sigma$.

d) $=_{\Theta,E}$ ($=_{\Sigma,E}$) is sort-preserving and deduction-closed with respect to $\mathbf{S}_\Theta$.

e) CS has a $\Sigma$-model $\Leftrightarrow$ CS has a $\Theta$-model.

Furthermore there is a well-sorted transformation $H:\Sigma \rightarrow \Theta$ that is bijective on functions and predicates and terms and $S_\Theta(t) = \cup H(\{S_\Sigma(s) \mid s =_{\Sigma,E} t\})$.

**Proof.** We define a new notion of the sort of a term using Theorem 8.11. To distinguish between old and new objects we use the suffix $\Theta$ for new ones. We define $SUB_\Theta := \{\sigma \in SUB_\Sigma \mid \forall x \in DOM(\sigma)\ \exists s \in \mathbf{T}_\Sigma\text{:}\ \ s =_{\Sigma,E} \sigma x \text{ and } S_\Sigma(s) \supseteq S_\Sigma(x)\}$

a) $\forall \sigma_\Theta \in SUB_\Theta\ \exists \sigma \in SUB_\Sigma\ \sigma =_{\Sigma,E} \sigma_\Theta$ :

Let $\sigma_\Theta \in SUB_\Theta$ and let $x \in \mathbf{V}$. There exists a term $s_x \in \mathbf{T}_\Sigma$ with $s_x =_{\Sigma,E} \sigma_\Theta x$ and $S_\Sigma(s_x) \supseteq S_\Sigma(x)$. Define $\sigma x := s_x$. Then $\sigma \in SUB_\Sigma$ and $\sigma =_{\Sigma,E} \sigma_\Theta$. ❑

$(\mathbf{T}_\Sigma, SUB_\Theta)$ is an R-system:

We have by assumption that $\mathbf{T}_\Sigma$ is subterm-closed.

We show only **R**-ii), since the other conditions are trivially satisfied.

$SUB_\Theta(\mathbf{T}_\Sigma) = \mathbf{T}_\Sigma$:

Let $\sigma_\Theta \in SUB_\Theta$ and $t \in \mathbf{T}_\Sigma$. Then there exists $\sigma \in SUB_\Sigma$ with $\sigma =_{\Sigma,E} \sigma_\Theta$. We have $\sigma t =_{\Sigma,E} \sigma_\Theta t$, hence $\sigma t \in \mathbf{T}_\Sigma$.

$SUB_\Theta$ is a monoid:

Let $\sigma_\Theta, \tau_\Theta \in SUB_\Theta$ and let $x \in \mathbf{V}_\Sigma$. Let $\sigma \in SUB_\Sigma$ be a substitution with $\sigma =_{\Sigma,E} \sigma_\Theta$.

For $\tau_\Theta x$ there exists a term $t_x$ with $\tau_\Theta x =_{\Sigma,E} t_x$ and $S_\Sigma(t_x) \supseteq S_\Sigma(x)$.

We have $\sigma_\Theta \tau_\Theta x =_{\Sigma,E} \sigma_\Theta t_x =_{\Sigma,E} \sigma t_x$. Hence by Proposition I.4.7: $S_\Sigma(\sigma t_x) \supseteq S_\Sigma(t_x) \supseteq S_\Sigma(x)$. We have shown $\sigma_\Theta \tau_\Theta \in SUB_\Theta$.

Now by Theorem 8.11 there exists a sort-assignment $S_\Theta$.

b) is trivial.

c) Let $=_{\Theta,E}$ be the $SUB_\Theta$-invariant congruence on $\mathbf{T}_\Sigma$ generated by $=_{\Sigma,E}$. We show by induction that $=_{\Theta,E}$ is identical with $=_{\Sigma,E}$.

It suffices to verify that every newly generated $=_{\Theta,E}$-relation is also a $=_{\Sigma,E}$-relation.

The nontrivial part is to show that for $s =_{\Sigma,E} t$ and $\sigma_\Theta \in SUB_\Theta$ we have $\sigma s =_{\Sigma,E} \sigma t$.

Let $\sigma \in SUB_\Sigma$ be the corresponding well-sorted substitution with $\sigma =_{\Sigma,E} \sigma_\Theta$.

Then $\sigma_\Theta s =_{\Sigma,E} \sigma s =_{\Sigma,E} \sigma t =_{\Sigma,E} \sigma_\Theta t$.

d) We show that $=_{\Theta,E}$ is sort-preserving:

Let $s,t \in \mathbf{T}_\Sigma$ with $s =_{\Sigma,E} t$. To show that $S_\Theta(s) = S_\Theta(t)$ it suffices to show that $\{x \in \mathbf{V}_\Sigma \mid \{x \leftarrow s\} \in SUB_\Theta\} = \{x \in \mathbf{V}_\Sigma \mid \{x \leftarrow t\} \in SUB_\Theta\}$ by the definition of $SUB_\Theta$ and by Definition 8.6. If $\{x \leftarrow s\} \in SUB_\Theta$, there exists a term $s_x$ such that $s =_{\Sigma,E} s_x =_{\Sigma,E} t$, hence also $\{x \leftarrow t\} \in SUB_\Theta$.

e) Let the transformation $H : \Sigma \to \Theta$ be defined as follows:

H is bijective on **F**, **P** and $\mathbf{T}_\Sigma$. $H : \mathbf{S}_\Sigma \to \mathbf{S}_\Theta$ with $H(S) = \mathbf{LS}_\Theta(x)$ for some variable x with $\mathbf{LS}_\Sigma(x) = S$.

1) H is well-sorted:

i) $R \sqsubseteq S \Rightarrow H(R) \sqsubseteq_\Theta H(S)$:

Is obvious, since $\{x_S \leftarrow y_R\} \in SUB_\Sigma \Rightarrow \{x_S \leftarrow y_R\} \in SUB_\Theta$.

ii) Let $S \in \mathbf{S}_\Sigma(t)$. Then $H(S) \in \mathbf{S}_\Theta(t)$:

This holds, since $\{x_S \leftarrow t\} \in SUB_\Sigma$ implies $\{x_S \leftarrow t\} \in SUB_\Theta$.

2) Let M be a $\Sigma$-model of CS. Then H(M) is a $\Theta$-model of CS:

It suffices to show that H(M) is a $\Theta$-algebra by Theorem 7.6.

i) $H(R) \sqsubseteq_\Theta H(S)$ implies $H(R)_M \subseteq H(S)_M$:

Let $H(R) \sqsubseteq_\Theta H(S)$ and let $x_{H(R)}$, $y_{H(S)}$ be variables of $\Theta$-sort H(R) and H(S), respectively. Then either $\{y_{H(S)} \leftarrow x_{H(R)}\} \in SUB_\Sigma$ or there exists a term $t_x$ such that $x =_{\Sigma,E} t_x$ and $\{y \leftarrow t_x\} \in SUB_\Sigma$.

Let $m \in H(R)_M$. By definition of H(M) (cf. paragraph 7) there exists a variable x with $\mathbf{LS}_\Theta(x) = H(R)$ and $\{x \leftarrow m\}$ is a $\Sigma$-assignment. Let y be a variable with $\mathbf{LS}_\Theta(y) = H(S)$. If $\{y \leftarrow x\} \in SUB_\Sigma$, then $\varphi := \{y \leftarrow m\}$ is a $\Sigma$-assignment and hence $m \in H(S)_M$. In the case $\{y \leftarrow x\} \notin SUB_\Sigma$ there exists a term $t_x =_{\Sigma,E} x$, such that $\{y \leftarrow t_x\} \in SUB_\Sigma$. Since M is a $\Sigma$-model we have $\varphi t_x = m$, for every total $\Sigma$-assignment extending $\{x \leftarrow m\}$, hence $\{y \leftarrow m\}$ is a $\Sigma$-assignment and $m \in (\mathbf{LS}_\Sigma(y))_M \subseteq H(S)_M$.

ii) $H(S)_M = S_M$ and every $\Theta$-assignment is also a $\Sigma$-assignment:

We show $H(S)_M = S_M$. The second claim then follows immediately.

Let $\{x \leftarrow m\}$ be a $\Sigma$-assignment and let y be a variable with $y \equiv_\Theta x$. If $\{y \leftarrow x\} \in SUB_\Sigma$, then $\{y \leftarrow m\}$ is a $\Sigma$-assignment. If $\{y \leftarrow x\} \notin SUB_\Sigma$, then there exists a term $t_x =_{\Sigma,E} x$, such that $\{y \leftarrow t_x\} \in SUB_\Sigma$. Since M is a $\Sigma$-model we have $\varphi t_x = m$ for every total $\Sigma$-assignment extending $\{x \leftarrow m\}$, hence $\{y \leftarrow m\}$ is a $\Sigma$-assignment and $m \in (\mathbf{LS}_\Sigma(y))_M \subseteq H(S)_M$. Hence $H(S)_M = S_M$. ❑

iii) For $H(S) \in \mathbf{S}_\Theta(t)$ and all $\Theta$-assignments $\varphi_\Theta$ we have $\varphi_\Theta t \in H(S)_M$:

Let $\varphi_\Theta$ be a $\Theta$-assignment. By ii) $\varphi_\Theta$ is also a $\Sigma$-assignment. A similar argument as above shows that for every variable x with $\{x \leftarrow t\} \in SUB_\Theta$ we have $\varphi_\Theta t \in (\mathbf{LS}_\Sigma(x))_M$.

3) Let $M_\Theta$ be a $\Theta$-model of CS. We have to construct an M, such that $H(M) = M_\Theta$ and M is a $\Sigma$-algebra. We let the denotation of functions and predicates unchanged and define $S_M := H(S)_{M\Theta}$. It is a trivial task to verify all necessary conditions. ■

The semantical sort-assignment may be not regular:

**9.2 Example.** Let $\Sigma := \{B \sqsubseteq A, C \sqsubseteq A, b{:}B, c{:}C\}$ and let $E := \{b = c\}$. Then $\mathbf{S}_\Theta = \mathbf{S}_\Sigma$ and the sort of b with respect to $\Theta$ is $\mathbf{S}_\Theta(B) = \{A,B,C\}$. Since this set has no unique minimal element, the new sort-assignment is not regular. ■

Unfortunately, the construction in Theorem 9.1 may not be effective in general. A consequence is that the new sort-assignment may not be computable. It may nevertheless be of

practical use to consider a term t of sort S, if there is a term s of sort S with $s =_{\Sigma,E} t$. Theorem 9.1 shows that this is a correct method and that in Example 9.2 we can consider c to be also of sort B.
A case, where the above construction behaves well is that $\Sigma$ is regular and $\mathcal{E}$ is defined by a weakly sort-decreasing and canonical term rewriting system. Then $\Theta$ is regular, the set of sorts does not change, i.e., $S_\Sigma = S_\Theta$, and the new sort of a term is the sort of its normalform.

## 10. Relativizations.

In this paragraph we consider two different methods to transform sorted clause sets into unsorted ones in a conservative way. The first is the standard method [Ob62, Wa83] to provide a unary predicate for every sort, to add conditional literals to clauses and to add clauses that express the signature of the clause set.

The second method transforms sorts into unary functions and the sort-information into suitable equations for these unary functions.

For the special case where no equations are in the clause set and the sort-structure is a tree, there is a third method to relativize a clause set (cf. [St86]), namely to embrace every term with unary function symbols that represent the sort of the term. For example if there are the sorts $A \sqsupseteq B \sqsupseteq C$ and the term t has sort C, then we relativize (recursively) t as $f_A(f_B(f_C(t)))$. It can be shown, that unsorted resolution for the thus relativized clauses simulates sorted resolution. We do not further consider the third case.

**10.1 Definition.** Let $\mathcal{S} = (\Sigma, CS)$ be a specification. The we define the **relativized specification** $\mathcal{S}_{REL} := (\Sigma_{REL}, CS_{REL} \cup Ax_{REL})$ as follows:

i) $\Sigma_{REL} := \bar{\Sigma} \cup \mathbf{P}_{REL}$, where $\mathbf{P}_{REL}$ is a set of new unary predicate symbols $P_S$ for every sort S.

ii) $Ax_{REL}$ is the set of clauses
   a) $\{-P_R(x), P_S(x)\}$ for every relation $R \sqsubseteq S$.
   b) $\{-P_{S1}(x_1), \ldots, -P_{Sn}(x_n), P_S(t)\}$ for every term declaration $t{:}S \in \Sigma$, where $\mathbf{V}(t) = \{x_1,\ldots,x_n\}$ and $S_i = S(x_i)$.

iii) $CS_{REL}$ is the set of clauses:
   $C_{REL}$ for every $C \in CS$, where $C_{REL} := \{-P_{S1}(x_1), \ldots, -P_{Sn}(x_n)\} \cup C$, $\mathbf{V}(C) = \{x_1,\ldots,x_n\}$ and $S_i = S(x_i)$. ■

The "Sortensatz" in [Ob62, Wa83] states that a sorted clause set and its relativization have the same semantics. The same is true in signatures with term-declarations:

**10.2 Theorem.** $\mathcal{S}$ has a $\Sigma$-model, iff $\mathcal{S}_{REL}$ has a $\Sigma_{REL}$-model.

**Proof.** "$\Rightarrow$": Let M be a $\Sigma$-model of $\mathcal{S}$. In order to obtain a $\Sigma_{REL}$-model of $\mathcal{S}_{REL}$ we add the relations $P_{S,M} := S_M$, i.e., we define the predicate $P_{S,M}$ to be valid exactly on the set $S_M$. In order to be precise we have to forget about the denotation of sorts and have to define the functions $f_M$ on the whole set M. This definition can be done arbitrarily.

ii.a): The clauses $\{-P_R(x), P_S(x)\}$ are valid, since $R_M \subseteq S_M$.

ii.b): Let $\varphi$: $V_{\overline{\Sigma}} \rightarrow M$ be an assignment. If $\varphi(x_i) \notin P_{Si,M}$ for some $S_i$ then the clause is valid. If $\varphi(x_i) \in P_{Si,M}$ for all i, then $\varphi$ corresponds to a $\Sigma$-assignment, hence $\varphi(t) \in P_{S,M}$, and hence the literal $P_S(t)$ is valid.

iii): Similar to the proof of ii) a). ❑

"$\Leftarrow$": Let M be a $\Sigma_{REL}$-model of $\mathcal{S}_{REL}$. In order to obtain a $\Sigma$-model of $\mathcal{S}$ we define the denotations of a sort S as $S_M := P_{S,M}$.

The clauses ii.a) enforce that $R \sqsubseteq S$ implies $R_M \subseteq S_M$. The clauses ii.b) enforce that for term declarations t:S and $\Sigma$-assignments $\varphi$ the application of $\varphi$ to t is defined and that $\varphi t \in S_M$. The clauses C are valid in M, since a $\Sigma$-assignment $\varphi$ corresponds to a usual $\Sigma_{REL}$-assignment, which makes all literals $-P_S(x)$ false and hence the remainder of the clause $C_{REL}$, that is the clause C itself, valid. ■

We prove that the sort of a term is reflected in the relativization of clauses related to the sort of a term:

**10.3 Lemma.** Let $\Sigma$ be a signature and let $\Sigma_{REL}$ be its relativization.

Then for every well-sorted $\Sigma$-term t:

$t \in T_{\Sigma,S} \Leftrightarrow Ax_{REL} \models \{-P_{S1}(x_1),\ldots,-P_{Sn}(x_n), P_S(t)\}$,

where $V(t) = \{x_1,\ldots,x_n\}$ and $S_i = S(x_i)$.

**Proof.** The proof is similar to the proof of Proposition I.6.3.

"$\Rightarrow$": We prove this by structural induction according to Definition I.4.3. As induction basis, we have that the axiom 10.1.ii.b) is deducable for term declaration t:S and for variables x we have the tautology $\{-P_S(x), P_S(x)\}$.

In order to prove the induction step, let $t \in T_{\Sigma,S}$, $r \in T_{\Sigma,R}$ and $x_1 \in V(t)$, such that $R \sqsubseteq S(x_1)$. Let $V(t) := \{x_1,\ldots,x_n\}$ and let $V(r) := \{y_1,\ldots,y_m\}$. Furthermore let $S_i$ be the sort of $x_i$ and $R_i$ be the sort of $y_i$. The term $\{x_1 \leftarrow r\}t$ is in $T_{\Sigma,S}$ by Definition I.4.3. We have to show that $\{-P_{S1}(x_2),\ldots,-P_{Sn}(x_n), -P_{R1}(y_1),\ldots,-P_{Rm}(y_m), P_S(\{x_1 \leftarrow r\}t)\}$ holds. By the induction hypothesis we have that $\{-P_{S1}(x_1),\ldots,-P_{Sn}(x_n),P_S(t)\}$ and $\{-P_{R1}(x_1),\ldots,-P_{Rm}(y_m), P_R(r)\}$ hold in all models of $Ax_{REL}$. Let M be a model and let $\varphi$ be an assignment such that the prefix $\{-P_{S1}(x_2),\ldots,-P_{Sn}(x_n), -P_{R1}(y_1),\ldots,-P_{Rm}(y_m)\}$ is not valid in this model. Then the literal $P_R(r)$ is valid under $\varphi$. Let $\psi$ be the assignment that differs from $\varphi$ only at the variable $x_1$ and assigns $x_1$ the element $\varphi x_1$. Then the literal

$P_S(t)$ is valid under $\psi$. We have $\varphi\{x_1 \leftarrow r\}t = \psi t$, hence $P_S(\{x_1 \leftarrow r\}t)$ is valid under $\varphi$. We have proved the induction step. Hence the conclusion is true ❑

"$\Leftarrow$": The other direction follows from model-theoretic considerations.

We construct a $\Sigma_{REL}$-model for the axioms $Ax_{REL}$ as follows. Let $M := \mathbf{T}_{\bar{\Sigma}}$, furthermore define the denotations for predicates $P_{S,M} := \mathbf{T}_{\Sigma,S}$. Then M is a $\Sigma_{REL}$-model.

Now for all terms $t \in \mathbf{T}_{\bar{\Sigma}} - \mathbf{T}_{\Sigma,S}$ the clause $\{-P_{S1}(x_1),\ldots,-P_{Sn}(x_n), P_S(t)\}$, where $V(t) = \{x_1,\ldots,x_n\}$ and $S_i = S(x_i)$, is not valid using the 'identical' assignment, since M is a model. ■

**10.4 Corollary.** Let $\Sigma$ be a signature and let $\Sigma_{REL}$ be its relativization.

Then for every well-sorted ground $\Sigma$-term t:

$t \in \mathbf{T}_{\Sigma,S} \Leftrightarrow Ax_{REL} \models \{P_S(t)\}$

where $V(t) = \{x_1,\ldots,x_n\}$ and $S_i = S(x_i)$.

Another method to relativize a sorted clause set is to introduce unary function symbols $f_S$ for every sort S and to add equations to ensure the right behaviour. The transformation of a sorted term t into its unsorted version is done by embracing every variable x in t of sort $S_x$ by the sort function $f_{Sx}$ and "t has sort S" is translated into $f_S(t) = t$.

**10.5 Definition.** Let $\mathcal{S} = (\Sigma, CS)$ be a specification. The **equationally relativized specification** is defined as follows: $\mathcal{S}_{EQR} := (\Sigma_{EQR}, CS_{EQR} \cup Ax_{EQR})$ of CS.

i) $\Sigma_{EQR} := \bar{\Sigma} \cup \mathbf{F}_{EQR}$, where $\mathbf{F}_{EQR}$ is a new set of unary function symbols $f_S$ for every sort S.

ii) The relativization of terms in $\mathbf{T}_\Sigma$ is a function $\delta: \mathbf{T}_\Sigma \rightarrow \mathbf{T}_{\Sigma EQR}$, where $\delta t$ is the term obtained by replacing every variable x of sort $S_x$ in t by the term $f_{Sx}(x)$. We can extend $\delta$ in the usual way to atoms, literals, clauses and clause sets.

iii) $Ax_{EQR}$ is the set of clauses

a) $\{f_S(f_R(x)) = f_R(x)\}$ for every relation $R \sqsubseteq S$.

b) $\{f_S(\delta t) = \delta t\}$ for every term declaration $t{:}S \in \Sigma$.

iv) $CS_{EQR}$ is the clause set $\delta(CS)$. ■

We denote equality defined by the above axioms as $=_{EQR}$.

The next lemma shows one direction of the sortal behaviour of the relativization, the other direction is shown in Lemma 10.8.

**10.6 Lemma.** We have for all $t \in \mathbf{T}_\Sigma$: $t \in \mathbf{T}_{\Sigma,S} \Rightarrow f_S(\delta t) =_{EQR} \delta t$:

**Proof.** We prove this by structural induction on the generation of terms according to Definition I.4.3. The induction base is that for term declarations t:S we have the axiom

$f_S(\delta t) = \delta t$ and for variables $x \in \mathbf{T}_{\Sigma,S}$ we have $S(x) \sqsubseteq S$, hence $f_S(\delta x) = \delta x$ by the axiom $f_S(f_{S(x)}(x)) = f_{S(x)}(x)$.

In order to prove the induction step, let $t \in \mathbf{T}_{\Sigma,S}$, $r \in \mathbf{T}_{\Sigma,R}$ and $x \in \mathbf{V}_\Sigma$, such that $R \sqsubseteq S(x)$. The term $\{x \leftarrow r\}t$ is in $\mathbf{T}_{\Sigma,S}$. We have to show that $f_S(\delta(\{x \leftarrow r\}t)) =_{EQR} \delta(\{x \leftarrow r\}t)$. Application of the substitution $\{x \leftarrow \delta r\}$ to the equation $f_S(\delta t) = \delta t$ yields that the equation $f_S(\{x \leftarrow \delta r\}\delta t) =_{EQR} \{x \leftarrow \delta r\}\delta t$ holds. However, by induction hypothesis, we have $\delta(\{x \leftarrow r\}t) =_{EQR} \{x \leftarrow \delta r\}\delta t$, since $f_{S(x)}(\delta r) =_{EQR} \delta r$. ■

**10.7 Theorem.** $\mathcal{S}$ has a $\Sigma$-model, iff $\mathcal{S}_{EQR}$ has a $\Sigma_{EQR}$-model.

**Proof.** "⇒": Let M be a $\Sigma$-model of $\mathcal{S}$. In order to obtain a $\Sigma_{EQR}$-model of $\mathcal{S}_{EQR}$ we define the new unary functions $f_{S,M}$ to be the identity on $S_M$, and for an element $a \in M - S_M$ we define $f_S(a)$ to be an arbitrary element in $S_M$. This definition is possible since $S_M$ is nonempty. (In the following we refer to Definition 10.5)

i) The axioms iii.a) are valid in the new model:

$f_S(f_R(x)) = f_R(x)$ holds in the model, since $f_{R,M}(a) \in S_M$ and $f_{S,M}$ is the identity on $S_M$.

ii) The clauses 10.5 iii.b) are valid in the new model:

It suffices to show that for every term declaration t:S and for every $\Sigma_{EQR}$-assignment $\varphi$, the $\varphi(\delta t) \in S_M$. This is true, since every variable x of sort $S_x$ is embraced by the function symbol $f_{Sx}$. Hence $\varphi$ corresponds to the $\Sigma$-assignment $\varphi'$ with $\varphi'(t) = \varphi(\delta t)$, hence $\varphi(\delta t) \in S_M$.

iii) The clauses $CS_{EQR}$ are valid:

Every variable x of sort $S_x$ is embraced by the function symbol $f_{Sx}$. Hence $\varphi$ corresponds to $\Sigma$-assignment $\varphi'$ with $\varphi'(C) = \varphi(\delta C)$ for every clause C, hence $CS_{EQR}$.

"⇐": Let M be a $\Sigma_{EQR}$-model of $\mathcal{S}_{EQR}$. In order to obtain a $\Sigma$-model of $\mathcal{S}$ we define the denotation $S_M$ for every sort S as $S_M := \{a \in M \mid f_{S,M}(a) = a\}$. By Lemma 10.6 we have for every term $t \in \mathbf{T}_{\Sigma,S}$ that $f_S(\delta t) = \delta t$, hence $S_M$ is nonempty by assumption I.4.11.

From the axioms in iii.a) it follows that $R \sqsubseteq S$ implies $R_M \subseteq S_M$. In order to show condition I.6.1.ii, let $\varphi$ be a partial $\Sigma$-assignment and let t:S be a term-declaration. Obviously we have $\varphi(\delta t) = \varphi t$ and since $f_S(\delta t) =_{EQR} \delta t$, we have also $f_{S,M}\varphi(\delta t) = \varphi(\delta t)$, hence $\varphi t \in S_M$.

For every $\Sigma$-assignment $\varphi$ and every clause C we have $\varphi\ \delta(C) = \varphi\ (C)$, hence every clause is valid. ■

Now we can prove that also the converse of Lemma 10.6 holds:

**10.8 Lemma.** We have for all $t \in \mathbf{T}_\Sigma$: $t \in \mathbf{T}_{\Sigma,S} \Leftrightarrow f_S(\delta t) =_{EQR} \delta t$:

**Proof.** Lemma 10.6 shows "⇒".

The other direction follows from semantical considerations.

We construct a $\Sigma_{EQR}$-model for the empty clause-set as follows. Let $M := T_\Sigma$, furthermore define $f_{S,M}$ as the identity for terms $t \in T_{\Sigma,S}$ and for terms $t \notin T_{\Sigma,S}$ let $f_{S,M}(t)$ be an arbitrary term in $T_{\Sigma,S}$. The axioms in 10.5 iii) are satisfied due to the definition of M and Definition I.4.3. Assume by contradiction that there exists a term $t \in T_\Sigma - T_{\Sigma,S}$ with $f_S(\delta t) =_{EQR} \delta t$. We have $t \neq f_{S,M}(t)$. Let $\varphi$ be the 'identical' assignment . Then $\varphi(f_S(\delta t)) = \varphi(\delta t)$, since M is a model.

It follows that $f_{S,M}(t) = \varphi(f_S(\delta t)) = \varphi(\delta t) = t$. We have reached the contradiction $t \in T_{\Sigma,S}$. ■

## 11. Herbrand-Theorem.

Herbrand's theorem [He30] states that every unsatisfiable clause set has a finite set of ground instances that is unsatisfiable. We show that this result also applies to the sorted case. As a prerequisite we first have to relativize the equations, since the original Herbrand-Theorem is proved for the case without built-in equality and without sorts.

It is well-known [Lo78, CL73] that an unsorted clause set CS with equational literals can be transformed into a clause set CS' $\cup$ EQ-AX, where the equality predicate '=' is replaced by a new binary predicate EQ which is interpreted as any other binary predicate and EQ-AX is the set of equality axioms. We make the same process for sorted clause sets. We add the predicate declarations EQ (S, S) for every sort S to the signature. Let EQ-AX be the axioms:

i) $EQ(x_S, x_S)$ for every sort S and some variable $x_S$ of sort S.

ii) For every function symbol f and for all term declarations $f(r_1,\dots,r_n):S_1$, $f(t_1,\dots,t_n):S_2$ the clause:

$EQ(r_1, t_1) \wedge \dots \wedge EQ(r_n, t_n) \Rightarrow EQ(f(r_1,\dots,r_n), f(t_1,\dots,t_n))$

iii) For every predicate (including the new precicate EQ) and two predicate declaration $P(S_1,\dots,S_n)$ and $P(R_1,\dots,R_n)$ let $x_i$, $y_i$ be different variables of sort $S_i, R_i$, respectively. Let the clause be:

$EQ(x_1, y_1) \wedge \dots \wedge EQ(x_n, y_n) \wedge P(x_1,\dots,x_n) \Rightarrow P(y_1,\dots,y_n)$. ❑

Note that i) and iii) have as a consequence that the symmetry and transitivity of EQ holds in every model. Furthermore note that this is a finite set of equations.

In an unsorted signature the clauses ii) and iii) correspond to the clauses

$EQ(x_1, y_1) \wedge \dots \wedge EQ(x_n, y_n) \Rightarrow EQ(f(x_1,\dots,x_n), f(y_1,\dots,y_n))$ and

$EQ(x_1, y_1) \wedge \dots \wedge EQ(x_n, y_n) \wedge P(x_1,\dots,x_n) \Rightarrow P(y_1,\dots,y_n)$ .

These two clause sets are semantically equivalent:

**11.1 Proposition.** Let $\Sigma$ be a signature and $\Sigma'$ be the signature with the additional binary predicate EQ. Then CS has a $\Sigma$-model iff $CS' \cup$ EQ-AX has a $\Sigma'$-model.

**Proof.** "$\Rightarrow$": Let CS have a $\Sigma$-model M. Then the $\Sigma'$-model M' constructed by interpreting EQ in the same way as the original equality is indeed a $\Sigma'$-model, since the above clauses i) -iii) are satisfied.

"$\Leftarrow$": Let M' be a $\Sigma'$-model of CS. We can assume by I.8.7 that M' is the ground term algebra. We define the relation $\equiv$ on M' by $a_1 \equiv a_2$, iff $EQ(a_1, a_2)$ is valid.

$\equiv$ is a fully invariant congruence relation on M':

From i) and iii) it follows that $\equiv$ is an equivalence relation. Furthermore it is fully invariant.

To show that $\equiv$ is a $\Sigma$-congruence let $s_i \equiv t_i$ for $i = 1,\ldots,n$ and let f be a function symbol such that $f(s_1,\ldots,s_n)$ and $f(t_1,\ldots,t_n)$ are well-sorted. By Lemma I.4.9 there are term declarations that are more general than these terms. Hence there exists an axiom among the axioms under ii) that enforces $EQ(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n))$ to be valid, hence $f(s_1,\ldots,s_n) \equiv f(t_1,\ldots,t_n)$.

The relation $\equiv$ is fully invariant, since the only endomorphism on M' is the identity. Furthermore for every predicate P we have $a_1 \equiv b_1 \wedge \ldots \wedge a_n \equiv b_n \wedge P_{M'}(a_1,\ldots,a_n) \Rightarrow P(b_1,\ldots,b_n)$ for elements $a_i,b_i \in$ M'. Hence we can factor out the equivalence relation $\equiv$ and obtain a $\Sigma$-model of CS. ■

Now we can prove the sorted version of Herbrand's Theorem:

**11.2 Theorem.** Let $\Sigma$ be a signature.

A clause set CS is $\Sigma$-unsatisfiable iff there exists a finite set of ground $\Sigma$-instances that is unsatisfiable.

**Proof.** The proof is done in two steps. First we prove (using Theorem 10.2 'Sortensatz') that Herbrand's theorem holds for a clause set without equations. Second we use Proposition 11.1 to lift the Theorem to clauses with equations. We prove only the nontrivial direction.

i) Let CS be a $\Sigma$-unsatisfiable clause set without equations. We have to show that there exists a finite, unsatisfiable set of ground $\Sigma$-instances of CS. Theorem 10.2 implies that $CS_{REL} \cup AX_{REL}$ is $\Sigma_{REL}$-unsatisfiable. The Herbrand-Theorem for the unsorted case [CL73, Lo78] gives a finite, $\Sigma_{REL}$-unsatisfiable set of ground $\Sigma_{REL}$-instances $CS_{REL,gr} \cup AX_{REL,gr}$ of $CS_{REL} \cup AX_{REL}$. This implies that also $CS_{REL,gr} \cup AX_{REL}$ is a $\Sigma_{REL}$-unsatisfiable clause set. It may be possible that some clauses in $CS_{REL,gr}$ do not correspond to ground $\Sigma$-instances of a clause in CS. We argue that we can delete these clauses: Let $CS_{REL,ws}$ be the subset of clauses in $CS_{REL,gr}$ that correspond to ground $\Sigma$-instances of clauses in CS.

Assume by contradiction that $CS_{REL,ws} \cup AX_{REL}$ has a $\Sigma_{REL}$-model M. Let $CS_{ws}$ be the set of well-sorted clauses in $CS_{REL,ws}$, which are obtained by deleting the literals $P_S(t)$ from the clauses in $CS_{REL,ws}$. Then by Theorem 10.2 $CS_{ws}$ has a $\Sigma$-model. The proof of Theorem 10.2 and Corollary I.8.7 show that we can assume that this $\Sigma_{REL}$-model has $\mathbf{T}_{\overline{\Sigma},gr}$ as the underlying $\Sigma_{REL}$-algebra. Furthermore we can assume that the interpretation of the predicates $P_{M,S}$ is exactly $\mathbf{T}_{\Sigma,S,gr}$. Consider a clause C in $CS_{REL,gr}$, which cannot be obtained as relativization of a well-sorted instance of a clause in CS. Such a clause C has a literal $-P_S(t)$, such that t is a ground term and $t \notin \mathbf{T}_{\Sigma,S}$. Hence such clauses are valid in M. This means that we have reached the contradiction that $CS_{REL,gr} \cup AX_{REL}$ has a $\Sigma_{REL}$-model.

We conclude that the set $CS_{WS}$ is a finite $\Sigma$-unsatisfiable set of ground $\Sigma$-instances of clauses in CS. ❑

ii) Let CS be a $\Sigma$-unsatisfiable clause set with equations. Proposition 11.1 implies that set CS is $\Sigma$-unsatisfiable, iff $CS' \cup$ EQ-AX is $\Sigma'$-unsatisfiable. Part i) of this proof gives a finite, $\Sigma'$-unsatisfiable set of ground instances $CS'_{gr} \cup \text{EQ-AX}_{gr}$ of $CS' \cup$ EQ-AX. This implies that also $CS'_{gr} \cup$ EQ-AX is $\Sigma'$-unsatisfiable. Now Proposition 11.1 yields a finite $\Sigma$-unsatisfiable set of ground instances, namely $CS_{gr}$. ■

## 12. First-Order Formulas and Skolemization.

The notion 'signature with term-declarations' can be extended to first order predicate logic. That means that we can use logical connectives such as $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$, and the quantifiers $\forall$ and $\exists$. The formulas and the semantics of closed formulas (no free variables) are recursively defined as usual.

In the following definition we use the notation $\varphi\{x \leftarrow a\}$ for the assignment that is equal to $\varphi$ on all variables but x and $\varphi\{x \leftarrow a\}(x) := a$.

We give the usual recursive definition of validity (denoted by $\models$) for formulae [EF78, Sh67].:

Let M be a $\Sigma$-structure and let $\varphi{:}\mathbf{T}_\Sigma \rightarrow M$ be a partial $\Sigma$-assignment. Let F and G be formula.

$(M, \varphi) \models P(t_1,\ldots,t_n)$, iff $(\varphi t_1,\ldots,\varphi t_n) \in P_M$.

$(M, \varphi) \models t_1 = t_2$, iff $\varphi t_1 = \varphi t_2$

$(M, \varphi) \models F \wedge G$, iff $(M, \varphi) \models F$ and $(M, \varphi) \models G$

$(M, \varphi) \models \neg F$, iff not $(M, \varphi) \models F$.

| | | |
|---|---|---|
| $(M, \varphi) \models F \vee G$, | iff $(M, \varphi) \models F$ or $(M, \varphi) \models G$. |
| $(M, \varphi) \models F \Rightarrow G$, | iff $(M, \varphi) \models F$ implies $(M, \varphi) \models G$. |
| $(M, \varphi) \models F \Leftrightarrow G$, | iff $(M, \varphi) \models F$ is equivalent to $(M, \varphi) \models G$. |
| $(M, \varphi) \models \exists x_S: F$, | iff there exists an $a \in S_M$: $(M, \varphi\{x \leftarrow a\}) \models F$ |
| $(M, \varphi) \models \forall x_S: F$, | iff for all $a \in S_M$: $(M, \varphi\{x \leftarrow a\}) \models F$. |

A $\Sigma$-structure M is a **$\Sigma$-model** of a closed formula F (F has no free variables), iff $(M,\emptyset) \models F$, also denoted as $M \models F$. Note that this definition depends only on the structure M. This definition is consistent with the definition of validity of clauses and clause sets if each clause is interpreted as universally quantified over all variables occurring in it and as disjunction of its literals, wheras clause sets are conjunctions of their clauses.
The above definition works also if the same variable occurs under different quantifiers. Without loss of generality we can assume that in closed formulas every variable occurs exactly under one quantifier. If F' is the approriately renamed version of a formula F, then $M \models F'$ iff $M \models F$.

We have the same skolemization as described in [Wa83]:
A prenex formula F containing a subformula $\exists x_S$: G can be tranformed by **skolemization** steps as follows:

Let $\{x_1,\ldots,x_n\}$ be the set of variables occurring under a universal quantifier above $\exists x_S$: G.
Introduce a new n-ary function f: $S(x_1)\times\ldots\times S(x_n) \rightarrow S$.
Let G' be the formula G, where every occurrence of $x_S$ is replaced by $f(x_1,\ldots,x_n)$.
The new formula F' is then the formula F, where $\exists x_S$: G is replaced by G'.

The **skolemized formula** $F_{SK}$ of a formula F can be obtained by applying skolemization steps until all $\exists$-quantifiers have disappeared.

The skolemization is conservative:

**12.1 Proposition.** Let F be a prenex formula with respect to $\Sigma$ and let F' be the skolemized formula with respect to $\Sigma'$. Then F has a $\Sigma$-model iff F' has a $\Sigma'$-model.

**Proof.** Let M be a $\Sigma$-model of F. We can assume that there is only one skolemization step. We use the notation of the above definition. To construct a $\Sigma'$-model M' of F we have to define the function $f_M$ on M'. Let $a_i$ be elements in $S(x_i)_M$. If there exists an element $a \in S_M$, such that $(M, \{x_1 \leftarrow a_1,\ldots, x_n \leftarrow a_n, x_S \leftarrow a\}) \models \exists x_S$: G, then we define $f_M(a_1,\ldots,a_n) := a$, otherwise we define $f_M(a_1,\ldots,a_n)$ to be an arbitrary element of $S_M$. Note that $S_M$ is nonvoid.

If we have a partial $\Sigma$-assignment $\varphi = \{x_1 \leftarrow a_1,\ldots, x_n \leftarrow a_n\}$, then $(M, \varphi) \models \exists x_S$: G is equivalent to $(M', \varphi) \models G'$. Hence $M \models F$ implies $M' \models F'$.

The reverse direction uses the same techniques and is omitted. ■

Now we can use the same techniques as in the unsorted case to normalize a formula F, i.e. to transform it in a conservative way into a clause set. However, the method described here is straightforward and not very efficient in practice. There exist more efficient methods (cf. [EW83]).

The steps of such an algorithm are:

1) Eliminate $\Rightarrow$ and $\Leftrightarrow$.
2) Move the negation sign inside.
3) Skolemize
4) Move $\forall$-quantifiers outside
5) Use the associativity, commutativity and distributivity of $\wedge$ and $\vee$ to transform the formula into a conjunction of disjunctions.
6) Make clauses variable disjoint and eliminate all $\forall$-quantifiers.

Note that this algorithm has to rename variables appropriately, for instance in step 1) if copies of formulas are introduced and in step 6) where clauses have to be renamed.

In chapter VI.5 we give a method to combine this normalization algorithm with a sort-generating algorithm.

# Part III.
# Unification of Sorted Terms without Equational Theories

**Overview.** In this part the properties of unification of free order-sorted terms are investigated and rule-based unification algorithms are presented. We show that for elementary, regular signatures $\Sigma$-unification is decidable and finitary. In the general case when we have signatures with term declarations, unification is undecidable and infinitary. We also determine the unification behaviour under certain restrictions such as linearity.

Throughout this part we assume that the given signature $\Sigma$ is finite.

## 1. Minimal Unifier Sets and Minimal Weakening Sets.

**1.1 Proposition.** For every finite set W of variables, the quasi-ordering $\leq_\Sigma[W]$ is well-founded:

**Proof.** Obviously the set $S_\Sigma$ is finite. Assume there is a possibly infinite descending chain of substitutions $\sigma_1 >_\Sigma \sigma_2 >_\Sigma \sigma_3 >_\Sigma \ldots [W]$. Without loss of generality we can assume that $DOM(\sigma_i) \subseteq W$. Obviously, the depths of terms in $COD(\sigma_i)$ decrease. Due to Proposition I.10.5 the quasi-ordering $\leq[W]$ is well-founded, hence there exists an index n, such that $\sigma_m \equiv_{\overline{\Sigma}} \sigma_n [W]$ for all $m \geq n$. This means there exist $\overline{\Sigma}$-renamings $\rho_m$ with $DOM(\rho_m) = V(\sigma_m W)$, such that $\sigma_n = \rho_m \sigma_m [W]$. Due to Corollary I.5.4 the substitution $\rho_m$ is unique and hence it is well-sorted. Application of $\rho_m$ weakens the sorts of variables in $VCOD(\sigma_m)$. Since the number of sorts is finite and the number of variables in $VCOD(\sigma_m)$ is fixed for $m \geq n$, there exist different numbers $i,j \geq n$ such that $\sigma_i \equiv_\Sigma \sigma_j [W]$, hence the chain is finite. ■

An immediate consequence is:

**1.2 Corollary.** For every finite set $\Gamma$ of equations, there exists a minimal, complete set of $\Sigma$-unifiers $\mu U_\Sigma(\Gamma)$. ■

Furthermore a minimal set $\mu U_\Sigma(\Gamma)$ is recursively enumerable by the following algorithm: Using Proposition 1.1 and the decidability of syntactical equality of terms we can enumerate the set $U_\Sigma(\Gamma)$ in a sequence $\tau_i$ such that

i) for every $\equiv_\Sigma[V(\Gamma)]$-equivalence class only one representative is considered.

ii) the maximal depth of terms in $\tau_i\ (V(\Gamma))$ is increasing.

Using this enumeration we can collect a minimal set of $\Sigma$-unifiers in a set $\mu U$ by adding the next $\tau_i$ if and only if it is not an instance of a unifier already in $\mu U$. This procedure gives a minimal, complete set of unifiers, since the instance test is decidable (Corollary I.5.4)

Hence we have:

**1.3 Theorem.** For every set $\Gamma$ of equations, a minimal, complete set of $\Sigma$-unifiers for $\Gamma$ exists and is recursively enumerable. ∎

**1.4 Corollary**. Minimal and complete weakening sets $\mu W(\tau)$, $\mu W(t)$ and $\mu W(t \sqsubseteq S)$ exist and are recursively enumerable. ∎

**1.5 Proposition.** Let $\Sigma$ be elementary and regular and let $\tau$ be an ill-sorted substitution such that there exists a well-sorted substitution $\theta$ with $\theta\tau \in \mathbf{SUB}_\Sigma$.
Then there exists a finite, minimal set $\mu W(\tau)$ of weakenings, such that $\mu W(\tau)$ is effectively computable and consists of $\bar{\Sigma}$-renamings.

**Proof.** Let $\theta$ be a substitution, such that $\theta\tau$ is well-sorted. Let $\lambda$ be a substitution with $DOM(\lambda) = I(\tau)$, such that $COD(\lambda)$ consists of new variables and $\mathbf{LS}_\Sigma(\lambda x) = \mathbf{LS}_\Sigma(\theta x)$. By Lemma I.4.10 we have $\mathbf{LS}_\Sigma(\lambda\tau y) = \mathbf{LS}_\Sigma(\theta\tau y)$ for all $y \in I(\tau)$. Hence $\lambda \in W(\tau)$. Furthermore we have obviously $\lambda \leq_\Sigma \theta\ [I(\tau)]$. This means there exists a complete subset of $W(\tau)$ that consists of $\bar{\Sigma}$-renamings. Since the number of variables in $I(\tau)$ is finite and the signature is finite, it is sufficient for completeness to take a finite number of such $\bar{\Sigma}$-renamings. Since finite sets can be minimized, there exists a minimal complete subset of $W(\tau)$ consisting only of $\bar{\Sigma}$-renamings. Such a set is furthermore effectively computable, since the number of sorts is finite and matching and sort-computation are effective (cf.§I.5). ∎

## 2. A General Unification Procedure for Sorted Terms without Equational Theories.

In this paragraph we give a complete (nonterminating) unification procedure for the empty equational theory. First we give a procedure for the general case, which includes nonregular signatures. Second we give a more efficient procedure for regular signatures having a semilattice as sort-structure. Both algorithms use the ill-sorted binding-rule, which is in fact the internal paramodulation rule (cf. I.13) $x = t\ \&\ \Gamma \Rightarrow x = t\ \&\ \{x \leftarrow t\}\Gamma$ where $\{x \leftarrow t\}$ may be ill-sorted. We demonstrate that this is more efficient than the same rule using well-sorted replacements.

There are many well-known efficient unification algorithms for the unsorted case [Ba76, Hu76, PW78, MM82, KKN82]. The usual Robinson-algorithm [Ro65] with instantiation is of exponential time complexity, an improvement of [BC83] is quadratic, but it is not known whether there exists a quasi-linear algorithm with instantiation, hence the rule-based, quasi-linear unification algorithm of Martelli-Montanari type [MM82] avoids the instantiation rule,

For sorted unification, however, it is crucial to have a term fully instantiated, since otherwise the solution of problems like $x = t$ would blow up the search space. Those equations may have a lot of solutions in the sorted case in contrast to the unsorted case, where at most one most general solution exists.

The following is an nondeterministic rule system for sorted unification without any restrictions on the sort structure. For unusual notations and conventions the reader should refer paragraph I.13.

**2.1 Definition.** The set of rules GSOUP (general sorted unification procedure) is defined as follows:

VV1) $x = x \,\&\, \Gamma \quad \Rightarrow \quad \Gamma$

VV2) $x = y \quad \Rightarrow \quad y = x$
if $\mathbf{LS}_\Sigma(x) \sqsubset \mathbf{LS}_\Sigma(y)$.

VV3) $x = y \quad \Rightarrow \quad x = z \,\&\, y = z$
if $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and S is a maximal sort with $S \sqsubseteq \mathbf{LS}_\Sigma(x)$, and $S \sqsubseteq \mathbf{LS}_\Sigma(y)$ and z is a new variable of sort S.

VV4) $x = y \Rightarrow x = f(s_1,\ldots,s_n) \,\&\, y = f(t_1,\ldots,t_n) \,\&\, s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n$
if $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $f(s_1,\ldots,s_n){:}S$ is a term declaration with $S \sqsubseteq \mathbf{LS}(x)$ and $f(t_1,\ldots,t_n){:}R$ is a term declaration with $R \sqsubseteq \mathbf{LS}(y)$.

VV5) $x = y \Rightarrow *$
if $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $\mathbf{LS}_\Sigma(x)$ and $\mathbf{LS}_\Sigma(y)$ have no common subsort and there are no term declarations $f(s_1,\ldots,s_n){:}S$ with

$S \sqsubseteq \mathbf{LS}(x)$ and $f(t_1,\ldots,t_n)$:R with $R \sqsubseteq \mathbf{LS}(y)$.

VT1) $x = t \,\&\, \Gamma \quad \Rightarrow x = t \,\&\, \{x \leftarrow t\}\Gamma$
if $x \notin \mathbf{V}(t)$ and $x \in \mathbf{V}(\Gamma)$ .

VT2) $t = x \quad \Rightarrow x = t$
if t is not a variable.

VT3) $x = f(t_1,\ldots,t_n) \quad \Rightarrow x = f(s_1,\ldots,s_n) \,\&\, s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n$
if $x \notin \mathbf{V}(f(t_1,\ldots,t_n))$ , $\mathbf{LS}_\Sigma(x) \notin \mathbf{S}_\Sigma(f(t_1,\ldots,t_n))$ and $f(s_1,\ldots,s_n)$:S is a term declaration with $S \sqsubseteq \mathbf{LS}_\Sigma(x)$.

VT4) $x = f(t_1,\ldots,t_n) \quad \Rightarrow *$
if $x \notin \mathbf{V}(f(t_1,\ldots,t_n))$ , $\mathbf{LS}_\Sigma(x) \notin \mathbf{S}_\Sigma(f(t_1,\ldots,t_n))$ and there is no term declaration $f(s_1,\ldots,s_n)$:S with $S \sqsubseteq \mathbf{LS}_\Sigma(x)$

VT5) $x = t \quad \Rightarrow *$
If $x \in \mathbf{V}(t)$.

TT1) $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n) \Rightarrow s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n$ .

TT2) $s = t \quad \Rightarrow *$
If $hd(s) \neq hd(t)$. ■

Every declaration t:S taken from $\Sigma$ must be completly renamed with new variables before it is used in a unification step.

The above procedure could be enriched by the elimination rule $x = t \,\&\, \Gamma \Rightarrow \Gamma$, if x is an auxiliary variable not occurring in $\Gamma$ and $\{x \leftarrow t\}$ is well-sorted, where we say a variable is **auxiliary** if it does not appear in the original equation system to be solved.

The following lemma is easily albeit tediously shown:

**2.2 Lemma.** All steps of the above procedure are correct. ■

**2.3 Proposition.** All steps of the above procedure except the steps VV3), VV4), VT3) are complete.

**Proof.** We prove the nontrivial parts.

VV5) Let $\sigma \in \mathrm{SUB}_\Sigma$ with $\sigma x = \sigma y$. Since $\sigma$ is well-sorted, we have $\mathbf{LS}_\Sigma(x) \in \mathbf{S}_\Sigma(\sigma x)$ and

$LS_\Sigma(y) \in S_\Sigma(\sigma x)$ hence by Lemma I.4.9 there exist term declarations $f(s_1,\ldots,s_n):S$ with $S \sqsubseteq LS(x)$ and $f(r_1,\ldots,r_n):R$ with $R \sqsubseteq LS(y)$.

VT4) Let $\sigma \in SUB_\Sigma$ with $\sigma x = \sigma f(t_1,\ldots,t_n)$. Then $LS_\Sigma(x) \in S_\Sigma(\sigma f(t_1,\ldots,t_n))$. From Proposition I.4.9 it follows that there is a term declaration $s:S$ with $S \sqsubseteq LS_\Sigma(x)$.

VT5) If $x \in V(t)$, then $\langle x = t\rangle$ has no solution. ■

**2.4 Proposition.** The procedure GSOUP is a complete unification procedure.

**Proof.** Let $\sigma$ be an idempotent unifier in $U(\Gamma)$ with $DOM(\sigma) = V(\Gamma)$. We slightly change the definition of solved part and instead of the set of solved equations, that is the subset of $\Gamma$ of equations $x = t$, where $\{x \leftarrow t\}$ is well-sorted, we use the set $\Gamma_{WO}$ of worked-off equations as follows:

i) Solved equations are in $\Gamma_{WO}$.

ii) descendants of worked-off equations remain in $\Gamma_{WO}$, if VT1) is applied, even if the substitution component connected with this equation becomes ill-sorted.

Let $\Gamma_U$ be the unsolved part of $\Gamma$, i.e., the complement of $\Gamma_{WO}$ in $\Gamma$.

As well-founded complexity measure $\mu(\sigma, \Gamma)$ we use the multiset of all term depths in $\sigma(\Gamma_U)$. The idea of this proof is to show that there exists a pair $(\sigma', \Gamma')$, such that $\Gamma \Rightarrow \Gamma'$ by one step of GSOUP where $\sigma'$ is a unifier of $\Gamma'$ that is equal to $\sigma$ on old variables and extends $\sigma$ to new variables, furthermore $\mu(\sigma', \Gamma') < \mu(\sigma, \Gamma)$.

If $\mu(\sigma', \Gamma')$ is minimal, i.e., if the multiset is empty, then the set of equations is solved and we are ready. It is easy to verify that in this case all worked-off equations are indeed solved. The argument is that we can postpone the application of $\{x \leftarrow t\}$ to worked-off equations and make the application after all non-solved equations have disappeared. Then we can use VT1) only on well-sorted substitutions in an appropriate order to obtain the same solved system.

Now we show that there is always a GSOUP-step on $\Gamma$ that reduces the measure $\mu(\sigma,\Gamma)$.

First we argue that the rule VT1) does not increase the measure. This rule does not change the depths of terms in $\sigma\Gamma$, since from $\sigma x = \sigma t$ we obtain $\sigma\{x \leftarrow t\} = \sigma$, since $\sigma$ is idempotent. By definition of $\Gamma_{WO}$, equations remain in the set of worked-off equations after application of this rule.

We go through the cases for equations $s = t$ in $\Gamma_U$:

1) Case $s = t$, where neither s nor t is a variable.
   Then by step TT1 we reduce $\mu(\sigma,\Gamma)$, without changing the set of solutions.

2) Case $x = f(t_1,\ldots,t_n)$.
   Then $x \notin V(f(t_1,\ldots,t_n))$ and $LS_\Sigma(x) \notin S_\Sigma(f(t_1,\ldots,t_n))$, since $\sigma x = \sigma f(t_1,\ldots,t_n)$. By Lemma I.4.9 there exists a term declaration $f(s_1,\ldots,s_n):S$ with $S \sqsubseteq LS_\Sigma(x)$ and a

substitution $\tau$ with $\tau f(s_1,\ldots,s_n) = \sigma f(t_1,\ldots,t_n)$. We use step VT3 and VT1) to obtain a new equation system $\Gamma'$. Since the equation $x = f(s_1,\ldots,s_n)$ is solved, we have $\mu(\sigma\cup\tau,\Gamma') < \mu(\sigma,\Gamma)$, since the depth of $\sigma(f(t_1,\ldots,t_n))$ is larger than all term depths of $\sigma t_i$ and $\tau s_i$. Furthermore $\sigma \cup \tau$ is a solution of $\Gamma'$ with $\sigma \cup \tau = \sigma\ [\mathbf{V}(\Gamma)]$.

3) Case $x = y$.
Then $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$, since otherwise step VV2) reduces the complexity by shifting $x = y$ into the set of solved equations. We have $\sigma x = \sigma y$, hence $\mathbf{LS}_\Sigma(x) \in \mathbf{S}_\Sigma(\sigma x)$ and $\mathbf{LS}_\Sigma(y) \in \mathbf{S}_\Sigma(\sigma x)$.
   i) If $\sigma x$ is a variable, then $\mathbf{LS}_\Sigma(\sigma x) \sqsubseteq \mathbf{LS}_\Sigma(x)$ and $\mathbf{LS}_\Sigma(\sigma x) \sqsubseteq \mathbf{LS}_\Sigma(y)$, hence there exists a sort S with $\mathbf{LS}_\Sigma(\sigma x) \sqsubseteq S \sqsubseteq \mathbf{LS}_\Sigma(x)$ and $S \sqsubseteq \mathbf{LS}_\Sigma(y)$. We use step VV3) to obtain a new equation system $\Gamma'$. With $\tau = \{z \leftarrow \sigma x\}$ have $\mu(\sigma\cup\tau,\Gamma') < \mu(\sigma,\Gamma)$. Furthermore $\sigma \cup \tau$ is a solution of $\Gamma'$ with $\sigma \cup \tau = \sigma\ [\mathbf{V}(\Gamma)]$.
   ii) If $\sigma x$ is not a variable, then there exist term declarations $f(s_1,\ldots,s_n){:}S$ with $S \sqsubseteq \mathbf{LS}_\Sigma(x)$ and $f(r_1,\ldots,r_n){:}R$ with $R \sqsubseteq \mathbf{LS}_\Sigma(y)$ and a substitution $\tau$ with $\sigma x = \tau f(s_1,\ldots,s_n)$ and $\sigma y = \tau f(t_1,\ldots,t_n)$. With rule VV4) and VT1) we obtain an equation system $\Gamma'$. The substitution $\sigma \cup \tau$ is a solution of $\Gamma'$ with $\sigma \cup \tau = \sigma\ [\mathbf{V}(\Gamma)]$. Furthermore we have $\mu(\sigma\cup\tau,\Gamma') < \mu(\sigma,\Gamma)$. ∎

Note that the above procedure is also complete if we allow only well-sorted substitutions in rule VT1) .

## 3. Unification in Finite, Regular Signatures

We give unification rules for a complete unification procedure for a regular signature, where the sort-structure is a semi-lattice. This allows for a more efficient unification algorithm, since for example rule VV4, which requires declarations for the unification of two variables, is not necessary.

The assumption that the sort-structure is a semi-lattice is only for simplicity of rules and proofs. If the sort-structure is a partial ordering, then the new rule VV3 has to be adapted to handle a set of glb´s instead of a unique glb.

The rules given here are as in GSOUP, however the rules VV3, VV4,VV5, VT3, VT4 of GSOUP are improved.

**3.1 Definition.** The set of rules SOUP (sorted unification procedure) is defined as follows:

VV1) $\quad x = x \,\&\, \Gamma \quad \Rightarrow \quad \Gamma$

VV2) $x = y \quad \Rightarrow \quad y = x$
if $\mathbf{LS}_\Sigma(x) \sqsubset \mathbf{LS}_\Sigma(y)$.

VV3) $x = y \quad \Rightarrow \quad x = z \;\&\; y = z$
if $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $S = \mathrm{glb}(\mathbf{LS}_\Sigma(x), \mathbf{LS}_\Sigma(y))$ and z is a new variable of sort S.

VV5) $x = y \Rightarrow \ast$
if $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $\mathrm{glb}(\mathbf{LS}_\Sigma(x), \mathbf{LS}_\Sigma(y))$ does not exist.

VT1) $x = t \;\&\; \Gamma \quad \Rightarrow x = t \;\&\; \{x \leftarrow t\}\Gamma$
if $x \notin \mathbf{V}(t)$ and $x \in \mathbf{V}(\Gamma)$.

VT2) $t = x \quad \Rightarrow x = t$
if t is not a variable.

VT3) $x = f(t_1,\ldots,t_n) \quad \Rightarrow x = f(s_1,\ldots,s_n) \;\&\; s_1 = t_1 \;\&\; \ldots \;\&\; s_n = t_n$
if $x \notin \mathbf{V}(f(t_1,\ldots,t_n))$, $\mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $f(s_1,\ldots,s_n){:}S$ is a term declaration with $S \sqsubseteq \mathrm{glb}(\mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)), \mathbf{LS}_\Sigma(x))$.

VT4) $x = f(t_1,\ldots,t_n) \quad \Rightarrow \ast$
if $x \notin \mathbf{V}(f(t_1,\ldots,t_n))$, $\mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and there is no term declaration $s{:}S$ with $S \sqsubseteq \mathrm{glb}(\mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)), \mathbf{LS}_\Sigma(x))$.

VT5) $x = t \quad \Rightarrow \ast$
If $x \in \mathbf{V}(t)$.

TT1) $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n) \Rightarrow s_1 = t_1 \;\&\; \ldots \;\&\; s_n = t_n$ .

TT2) $s = t \quad \Rightarrow \ast$
If $\mathrm{hd}(s) \neq \mathrm{hd}(t)$. ■

We assume that declarations are completely renamed with new variables before used in a unification step.

**3.2 Proposition.** All steps of the above algorithm but the step VT3) are complete.

**Proof.** We prove only the nontrivial parts.

VV3) Let $\sigma \in \mathbf{SUB}_\Sigma$ with $\sigma x = \sigma y$. Since $\sigma$ is well-sorted, we have $\mathbf{LS}_\Sigma(\sigma x) \sqsubseteq \mathrm{glb}(\mathbf{LS}_\Sigma(x), \mathbf{LS}_\Sigma(y))$, hence $\sigma \cup \{z \leftarrow \sigma x\}$ is a solution of the new equations.

VV5) See the proof of VV3).

VT4) Let $\sigma \in \mathbf{SUB}_\Sigma$ with $\sigma x = \sigma f(t_1,\ldots,t_n)$. Then $\mathrm{glb}(\mathbf{LS}_\Sigma(x),\mathbf{LS}_\Sigma(f(t_1,\ldots,t_n))) \sqsupseteq \mathbf{LS}_\Sigma(\sigma f(t_1,\ldots,t_n))$. From Proposition I.4.9 it follows that there is a term declaration $f(s_1,\ldots,s_n)$: S with $S \sqsubseteq \mathrm{glb}(\mathbf{LS}_\Sigma(x),\mathbf{LS}_\Sigma(f(t_1,\ldots,t_n)))$ ). ∎

**3.3 Proposition.** The procedure SOUP is a complete unification procedure:

**Proof.** The proof of this proposition is similar to the proof of Proposition 2.4 and uses the same techniques. We mention only the differences in the induction arguments.

2) Case $x = f(t_1,\ldots,t_n)$.

Then $x \notin \mathbf{V}(f(t_1,\ldots,t_n))$ , $\mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$, since $\sigma x = \sigma f(t_1,\ldots,t_n)$. By Lemma I.4.9 there exists a term declaration $f(s_1,\ldots,s_n)$:S with $S \sqsubseteq \mathrm{glb}(\mathbf{LS}_\Sigma(f(t_1,\ldots,t_n)), \mathbf{LS}_\Sigma(x))$ and a substitution $\tau$ with $\tau f(s_1,\ldots,s_n) = \sigma f(t_1,\ldots,t_n)$. We use step VT3 and VT1) to obtain a new equation system $\Gamma'$. Since the equation $x = f(t_1,\ldots,t_n)$ is solved, we have $\mu(\sigma\cup\tau,\Gamma') < \mu(\sigma,\Gamma)$. Furthermore $\sigma \cup \tau$ is a solution with $\sigma \cup \tau = \sigma\ [V(\Gamma)]$.

3) Case $x = y$. Then

Then $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$, since otherwise step VV2 reduces the complexity.

We have $\sigma x = \sigma y$, hence $\mathbf{LS}_\Sigma(\sigma x) \sqsubseteq \mathrm{glb}(\mathbf{LS}_\Sigma(x), \mathbf{LS}_\Sigma(y))$. We use step VV3) and VT1) to obtain a new equation system $\Gamma'$. With $\tau = \{z \leftarrow \sigma x\}$ have $\mu(\sigma\cup\tau,\Gamma') < \mu(\sigma,\Gamma)$. Furthermore $\sigma \cup \tau$ is a solution with $\sigma \cup \tau = \sigma\ [V(\Gamma)]$. ∎

We demonstrate the advantages of rule VT1) over the instantiation rule with well-sorted solved equations:

**3.4 Example.** Let $\Sigma := \{B \sqsubset A,\ f(g(x_A)) : B,\ f(h(x_A)) : B,\ g{:}A \to A,\ g{:}B \to B,\ h{:}A \to A,\ h{:}B \to B\}$. Let $\Gamma := \{x_B = f(y_B),\ y_B = h(z_A)\}$.

1) We use the usual rule VT1):

Then VT1 is applicable and we get $\{x_B = f(h(z_A)),\ y_B = h(z_A)\}$. The first equation is solved, hence we proceed on the second. This gives $\{x_B = f(h(z_A)),\ y_B = h(x'_B),\ x_B' = z_A\}$ and then the final solution is $\{x_B = f(h(x_B')),\ y_B = h(x'_B),\ z_A = x_B'\}$

2) We use a rule $\mathrm{VT1}_{ws}$ that allows instantiation of $\Gamma$ only if $\{x \leftarrow t\}$ is well-sorted:

Then rule $\mathrm{VT1}_{ws}$ is not applicable, since $y_B = h(z_A)$ is not in solved form. For every equation there are two possibilities to proceed. If we proceed on the first equation with rule VT4), we obtain the two possibilities $\{x_B = f(g(x'_A)), y_B = g(x'_A), y_B = h(z_A)\}$ and $\{x_B = f(h(x'_A)), y_B = h(x'_A), y_B = h(z_A)\}$. The first equation system requires two further steps until it is recognized, that this system is unsolvable. The second requires one more step VT4) an application of $VT1_{ws}$ and then a decomposition step to obtain the solution.

A comparison of the behaviour shows that in the first approach the search space is smaller, unsolvability is earlier recognized and the path leading to a solution is shorter. It should be noted that this is also true for elementary signatures. ■

This behaviour is not accidental. We have experimented with another unification algorithm, that extends the Robinson algorithm by replacing the unification of a variable x and a term t with a procedure that first computes a complete set of weakenings of t (i.e., substitutions $\sigma$ with $\mathbf{LS}(\sigma t) \sqsubseteq \mathbf{LS}_\Sigma(x)$ } and then returns a minimal, complete set for the unification problem $<x = t>$. However, a practical comparison of these two algorithms shows that this Robinson extension is less efficient, the main reason is that computing the set of weakenings may produce many instances that are incompatible with the usual (ill-sorted) Robinson unifier of a set $\Gamma$.

Using the above rule-based procedure we can derive a deterministic Robinson-type algorithm for regular signatures, which obeys the above observation:

1) Compute a (generally ill-sorted) Robinson unifier $\sigma$ for $\Gamma$.
2) Compute a set of well-sorted instances of $\sigma$ by applying the following step repeatedly starting with PU := $\{\sigma\}$:

   Let PU be a set of substitutions.
   Take a substitution $\tau = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ from PU, a component $\{x_i \leftarrow t_i\}$ with $\mathbf{LS}_\Sigma(t_i) \not\sqsubseteq \mathbf{LS}_\Sigma(x_i)$ and a declaration $t{:}S \in \Sigma$ with $S \sqsubseteq \mathbf{LS}_\Sigma(x_i)$.
   Let $\mu$ be the Robinson unifier of t and $t_i$.
   Replace PU by PU $\cup \{\mu\tau\}$

**Remark** (for an implementation:)

i) The declaration t:S taken from $\Sigma$ must be completly renamed before every unification step.
ii) If $\sigma_i$ descends from $\sigma$ by taking the i-th component $\{x_i \leftarrow t_i\}$, then this component is "locked" for $\sigma_i$ and descendants of $\sigma_i$.
iii) Since the procedure is in general nonterminating, there is the need for some bounds, for example depth of the search tree.
iv) Efficiency can be increased if the sort-ignoring unification refuses to unify terms s,t

which have sorts with no common subsort. This test is only correct for regular signatures.

Now we consider the case of polymorphic signatures with a semilattice as sort-structure. Polymorphic signatures, a special case of finite, regular signatures, have some nice properties, for example unification is decidable and minimal unifier sets are finite. They allow a more efficient unification algorithm and are a base for the systems and calculi in [Wa84, GM85a, CD85, Sm86, Sch85a].

In [CD85] it is recognized that $\Sigma$-unification is not unitary, but a proof for finiteness is not given.

The following rules are slightly changed for polymorphic signatures and are marked with (*):

VV2*) $x = y \ \& \ \Gamma \ \Rightarrow \ y = x \ \& \ \{y \leftarrow x\}\Gamma$

if $\mathbf{LS}_\Sigma(x) \sqsubset \mathbf{LS}_\Sigma(y)$.

VV3*) $x = y \ \& \ \Gamma \ \Rightarrow \ x = z \ \& \ y = z \ \& \ \{x \leftarrow z, y \leftarrow z\}\Gamma$

if $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(y) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $S = \mathrm{glb}(\mathbf{LS}_\Sigma(x), \mathbf{LS}_\Sigma(y))$ and $z$ is a new variable of sort $S$.

VT3*) $x = f(t_1,\ldots,t_n) \ \Rightarrow \ x = f(t_1,\ldots,t_n) \ \& \ y_1 = t_1 \ \& \ \ldots \ \& \ y_n = t_n$

if $x \notin \mathbf{V}(f(t_1,\ldots,t_n))$, $\mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)) \not\sqsubseteq \mathbf{LS}_\Sigma(x)$ and $f(y_1,\ldots,y_n){:}S$ is a function declaration with $S \sqsubseteq \mathrm{glb}(\ \mathbf{LS}_\Sigma((f(t_1,\ldots,t_n)), \mathbf{LS}_\Sigma(x))$.

**3.5 Proposition.** If the signature $\Sigma$ is polymorphic and the sort-structure is a semi-lattice, then for every equation system $\Gamma$, the procedure terminates with an equation system in solved form.

**Proof.** The proof of completeness is almost the same as in the proof of Proposition 3.3.

We show that every application sequence of rules terminates:

We use the same concept of worked-off equations and unsolved equations as in Proposition 2.4. Solved equations are moved to $\Gamma_{WO}$, i.e. equations $x = t$, such that $x$ does not occur elsewhere in $\Gamma$ and $\{x \leftarrow t\}$ is well-sorted. However, equations remain in $\Gamma_{WO}$ even after applying the rule VT1), which can make the corresponding substitution ill-sorted. We assume that the rule VT3*) shifts the equation $x = f(t_1,\ldots,t_n)$ into $\Gamma_{WO}$.

The difference $\Gamma - \Gamma_{WO}$ is the set $\Gamma_U$.

If no rule is applicable, then the system is in solved form: This is true, since in the case $\Gamma_U$ is empty, all equations in $\Gamma_{WO}$ correspond to well-sorted substitutions.

For termination we use a complexity measure $\mu = (\mu_1, \mu_2, \mu_3)$, where

$\mu_1$ is the number of 'nonisolated' variables, i.e., variables in $\Gamma$ that occur not only as a left hand side of exactly one equation $x = t$ in $\Gamma$,

$\mu_2$ is the multiset of term depths of toplevel terms in $\Gamma_U$,

$\mu_3$ is the number of equations of the form $t = x$ in $\Gamma_U$, where t is not a variable.

Rule VT1) reduces the measure $\mu_1$, since only equations $x = t$ in $\Gamma_{WO}$ can be used and after application, x is isolated on a left hand side.

Rule VV2*) reduces $\mu$. Either x was not isolated before, then $\mu_1$ is reduced, or x was isolated before, then $\mu_1$ is not changed, but $\mu_2$ is reduced, since an equation is removed from $\Gamma_U$.

Rule VV3*) reduces $\mu$ since either $\mu_1$ is reduced or $\mu_2$, since an equation is removed from $\Gamma_U$.

Rule VT2) reduces the number of equations of the form $t = x$

Rule VT3*) shifts $x = f(t_1,\ldots,t_n)$ into $\Gamma_{WO}$, leaves $\mu_1$ invariant, since $y_i$ are new variables and reduces the measure $\mu_2$, since depth($f(t_1,\ldots,t_n)$) is replaced by the depths of $t_i$ and n times depth($y_i$).

Rule TT1) either reduces $\mu_1$ or leaves $\mu_1$ invariant and reduces $\mu_2$. ■

**3.6 Corollary.** $\Sigma$–unification is of type finitary for every finite, regular and elementary signature $\Sigma$. ■

The regularity condition is necessary, since in the following nonregular example with $\Sigma :=$ {a:A, a:B, f:A→A, f: B→B} the unification problem $<x_A = y_B>$ has the infinite minimal and complete set of unifiers $\{\{x_A \leftarrow f^i(a), y_B \leftarrow f^i(a)\} \mid i = 1,2,\ldots\}$, (where $f^i(a)$ means a term of the form f(...(f(a)...) with i occurrences of f).

For simple, finite order-sorted signatures in which the sort structure is a semi-lattice $\Sigma$-unification is of unification type unitary [Wa84] . In the case of a nonfinite, simple order-sorted signature Ch. Walther [Wa86] shows that the unification type is completely determined by properties of the subsort ordering.

## 4. Complexity of Unification in Elementary Signatures.

We consider in this paragraph the complexity of unification in different types of signatures. We assume that the signatures are always finite, elementary and regular.

**4.1 Proposition.** Let the sort-structure be a semi-lattice. Then

i) The number of unifiers may grow exponentially with the size of terms to be unified.

ii) The number of unifiers is at most exponential in the size of the terms.

**Proof.** i) Consider the signature with sorts {INT, POS} with INT ⊐ POS and the functions + and * with the assignments:

+: NAT x NAT → NAT, NAT x POS → POS, POS x NAT → POS,

*: NAT x NAT → NAT, POS x POS → POS,

The unification problem $(\ldots(x_{11} + x_{12})*(x_{21} + x_{22}))* \ldots *(x_{n1} + x_{n2})) = y_{POS}$, where all variables $x_{ij}$ are of sort NAT requires all subterms $x_{i1} + x_{i2}$ to have sort POS after instantiating. There are two independent solutions for every subterm $x_{i1} + x_{i2}$, namely $\{x_{i1} \leftarrow z_{i,POS}\}$ or $\{x_{i2} \leftarrow z_{i,POS}\}$, hence there are $2^n$ most general unifiers, but the term has size 2n.

ii) This can be proved by induction:

Assume as induction hypothesis that the number of unifiers is less than exp(c·size(Γ')) for some constant c and for all Γ' with a smaller size than Γ. For an equation system Γ it is possible to compute an unsorted most general unifier in quasi-linear time. Let the result be a set of multi-equations M. Note that the unification process as described in [MM82] ensures that the size of the original problem is greater than the sum of the sizes of the nonvariable terms in M. The critical part is the decomposition step that does not copy nonvariable terms.

Weakening this unifier to a well-sorted one yields for a variable-pure multi-equation one unifier and for a multi-equation $M_i$ with a nonvariable term $t_i$ less than $\exp(c \cdot size(t_i))$ unifiers. We weaken the multi-equations in the following way: we take the first multi-equation, compute the set of unifiers, apply the unifer to the whole multi-equation and afterwards add the corresponding unifier to the front of the multi-equation. As proved in Proposition 1.5, the weakenings are $\overline{\Sigma}$–renamings, hence do not increase the size of nonvariable terms. The total number of unifers in this process has as upper bound the product $\exp(c \cdot size(t_1)) \cdot \ldots \cdot \exp(c \cdot size(t_n)) = \exp(\cdot(size(t_1) + \ldots + size(t_n))$ which is smaller than $\exp(c \cdot size(\Gamma))$. Hence the number of unifiers produced is at most exponential. ■

This proves also that the computation of unifiers can be performed in exponential time, since all operations in the proof are neglectible in comparison with the exponentiality.

Now we consider the question 'is a set Γ unifiable?' and show that Σ-unification is NP-complete.

**4.2 Lemma.** $\Sigma$-unification and $\Sigma$-weakening is in NP.

**Proof.** Given an equation system $\Gamma$ and we can guess, verify and print a unifier $\theta$ in polynomial time

The same holds for weakening (cf. §I.5) ■

**4.3 Proposition.** $\Sigma$-unification is NP-complete.

**Proof.** We show that there exists a signature $\Sigma$, such that the $\Sigma$-unification is NP-hard and show this by reducing the satisfiability problem to the $\Sigma$-unification problem. It suffices to use a $\Sigma$-weakening problem, since every $\Sigma$-weakening problem has an equivalent $\Sigma$-unification problem if we use a new variable.

Let the regular signature be given as follows:

$\Sigma := \{$ BOOL $\sqsupset$ T, BOOL $\sqsupset$ F,

AND: BOOL x BOOL $\rightarrow$ BOOL, T x T $\rightarrow$ T, T x F $\rightarrow$ F, F x T $\rightarrow$ F, F x F $\rightarrow$ F,

OR: BOOL x BOOL $\rightarrow$ BOOL, T x T $\rightarrow$ T, T x F $\rightarrow$ T, F x T $\rightarrow$ T, F x F $\rightarrow$ F

NOT: BOOL $\rightarrow$ BOOL, T $\rightarrow$ F, F $\rightarrow$ T $\}$.

We use the characterization given for example [HoUl79 ].

Let $E_B$ be a boolean expression built from variables, and the connectives $\wedge$, $\vee$ and $\neg$. Then satisfiability of this expression is equivalent to the problem of finding a substitution $\sigma$ for a term $t_B$ such that $\mathbf{LS}_\Sigma(\sigma t_B) = T$. The term $t_B$ corresponding to $E_B$ is constructed by first translating $E_B$ into a boolean expression where $\wedge$ and $\vee$ are used as binary operators and then translating $\wedge$ into AND, $\vee$ into OR, $\neg$ into NOT and the variables into variables of so BOOL. ■

The interpretation of the above results is that sorted unification in polymorphic signatures is NP-complete and the explicit computation of all unifiers needs at most exponential time.

In the case of simple signatures the complexity can be improved considerably, since the computation is straightforward and the sort of compound terms is fixed and independent of the subterms. In fact unification is quasi-linear (see also [MGS87]):

**4.4 Proposition.** Unification in simple signatures is at most quasi-linear.

**Proof.** It is possible to compute in quasi-linear time an unsorted, (i.e. ignoring sort-information) solved set of multi-equations of a given $\Gamma$ [MM82].

Furthermore this unification process does not introduce new variables.

Given this solved set we do the following for every multi-equation:

1) If the multiequation M consists only of variables, we compute the set of greatest

common subsorts of all sorts of variables in M. This can be done in linear time, since the sort-structure is fixed.

2) If the multi-equation M consists of variables and a term t, we compute as in 1) the set of greatest common subsorts of all sorts of variables in M. Then we check, whether for some of these greatest common subsorts S, we have $S \sqsupseteq \mathbf{LS}_\Sigma(t)$
This is also possible in linear time.

These two actions are sufficient to give a representation of the unifiers. ■

**4.5 Example.** We give an example that the number of unifiers in a sort-structure that is not a semi-lattice may grow exponential.
Let the signature be $\{A,B \sqsupset C,D\}$.
Then the unification problem $\Gamma := \{x_{1,A} = y_{1,B}, \ldots , x_{n,A} = y_{n,B}\}$ has $2^n$ unifiers.
Hence we have the curious situation that the number of unifiers grows exponential, but it can be solved in quasi-linear time, i.e. a representation for all unifiers can be computed in quasi-linear time. ■

From the viewpoint of efficiency, the class of signatures, where always at most one unifier is necessary, is an interesting one. We have introduced this class as unification-unique in [Sch85a, Sch85b]. Unification-unique signatures are defined by two conditions: i) the sort-structure is a semi-lattice, ii) for every function symbol f and every range-sort S: the set $\{f(t_1,\ldots,t_n):R \mid R \sqsubseteq S\}$ has a unique greatest term declaration provided it is nonempty.
The same class is also investigated in [MGS87], where this class is called unitary and where it is shown that unification in this class can be performed in quasi-linear time.

## 5. Unification in Finite Signatures with Term Declarations is of Type Infinitary.

**5.1 Theorem.** There exists a finite signature in which unification is of type infinitary:
**Proof.** Let $\Sigma := \{B \sqsubset A, b:B, f(b):B, f(f(x_B)):B\}$.
The weakening problem $<f(x_B) :B>$ has infinitely many most general weakenings:
We prove by induction that all instances of $f(x_B)$ that have sort B are of the form b, f(b), f(f(b)),...:
first of all $f(x_B)$ is of sort A and f(b) is of sort B.
To prove the induction step, let t be a term of sort B with depth > 0 such that f(t) is of sort B. The term declarations show that t is of the form f(t'), where t' is of sort B. Hence by

induction $t' = f^n(b)$ for some $n \geq 0$, hence $t = f^{n+1}(b)$. ■

The proof shows a bit more than the theorem states, since in fact we constructed a linear and regular signature with unary function symbols having this property.

**5.2 Corollary.**

i) Let $\Sigma$ be a finite, linear and regular signature. Then the set $\mu U(s,t)$ may be infinite.

ii) Let $\Sigma$ be a finite, regular signature with only constants and unary function symbols. Then the set $\mu U(s,t)$ may be infinite.

## 6. Σ-Unification is Undecidable.

We reduce the decision problem for Σ-unification to the problem, whether a Turing machine (TM) accepts blank tape, a problem that is known to be undecidable (cf. [HoUl79]).

**6.1 Theorem.** Σ-unification is undecidable.

**Proof.** Given a TM M, we construct a signature $\Sigma$ and terms $s_0$, $t_0$ such that $s_0$ and $t_0$ are unifiable, iff M accepts blank tape.

We represent a configuration $\alpha q \beta$ of a TM, in which $\alpha$ is the string to the left of the head, q is the current state and $\beta$ is the string to the right of the head including the currently scanned symbol, by a term $h(r_1\ q\ r_2 \ldots)$ where $r_1$ is a term encoding the reverse of $\alpha$ and $r_2$ is a term encoding $\beta$. A string 11010... is represented as a term $g_1(g_1(g_0(g_1(g_0 \ldots (g_b(0))\ldots))))$ where $g_b(\ldots)$ represents an infinite string of blanks that is to say $g_b$ is an end-marker. There are two sorts TOP and OK with TOP ⊐ OK in the signature. The moves of M are represented as term declarations, for example if M is in state $q_1$, scanning a 1, prints 0, enters state $q_2$, and moves right, then the corresponding term declaration is $h(x\ q_1\ g_1(y)\ h(g_0(x)\ q_2\ y\ z_{OK}))$ : OK, where z is of sort OK and x,y of sort TOP.

To give a further example, if the head moves left, we may need the following three term declarations:

$h(g_0(x)\ q_1\ g_1(y)\ h(x\quad q_2\ g_0(g_0(y))\ z_{OK}))$ : OK
$h(g_1(x)\ q_1\ g_1(y)\ h(x\quad q_2\ g_1(g_0(y))\ z_{OK}))$ : OK
$h(g_b(x)\ q_1\ g_1(y)\ h(g_b(x)\ q_2\ g_b(g_0(y))\ z_{OK}))$ : OK

Let $q_S$ be the start state and $q_A$ be the accepting state. We add a term declaration for the accepting state:

$h(x\ q_A\ y\ w)$: OK, where x,y,w are variables of sort TOP.

The term $h(g_b(0)\ q_S\ g_b(0)\ z_{OK})$ represents the initial configuration of the Turing machine. Then the following holds:

M accepts blank tape iff the term $h(g_b(0)\ q_S\ g_b(0)\ z_{OK})$ has an instance of sort OK (equivalently $h(g_b(0)\ q_S\ g_b(0)\ z_{OK})$ is unifiable with a variable w of sort OK):

Proof: The equivalence follows, since such an instance has the form $h(g_b(0)\ q_S\ g_b(0)\ h(l_1\ q_1\ r_1\ h(l_2\ q_2\ r_2 \ldots)))$ and t represents the sequence $g_b(0)q_Sg_b(0)$, $l_1q_1r_1$, $l_2q_2r_2$, ...of moves of M reaching the accpeting state. ❑

Since it is undecidable whether a TM accepts blank tape, so is Σ-unification. ■

Using the concept of a universal Turing machine, a slight change in the above proof shows that there exists a signature Σ with an undecidable unification:

**6.2 Proposition.** There exists a signature Σ for which Σ−unifiability of terms is undecidable.

For nonregular signatures, we have the following undecidability results:

**6.3 Corollary.** i) It is undecidable whether two sorts have a common ground term.
ii) It is undecidable, whether two variables are unifiable.

**Proof.** Let Σ be a signature and let s and t be two Σ-terms. Then we construct a new signature Θ: Let A, B be new sorts without any subsort relation and let h be a new function symbol.

We add the declarations h(s): A and h(t): B to Σ resulting in Θ.

Were it decidable whether A and B have a common ground term, then unifiability of s and t would be decidable. This proves i). Now ii) is obvious, since unification of two variables x of sort A and y of sort B is an equivalent problem. ■

**6.4 Corollary.** The regularity of a signature is undecidable.

**Proof.** In the proof of Corollary 6.3 choose Σ as a regular signature. Then the regularity of the newly constructed Θ is equivalent to the unifiability of the terms s and t. ■

We say a unification problem Γ is **linear**, if every variable occurs at most once in Γ.

There are decidable cases of the unification problem for finite signatures.

**6.5 Proposition.** Let Σ be a finite signature.

i) If the signature is elementary, then Σ-unification is decidable.
ii) If the signature is regular, then Σ-unifiability of two different variables is decidable

iii) If the signature is linear, then linear $\Sigma$-unification problems are decidable.

iv) If all function symbols are either constants or unary functions, then $\Sigma$-unification is decidable.

**Proof.** i) Let $\Gamma$ be a set of equations. Then an unsorted most general unifier $\sigma$ of $\Gamma$ is effectively computable. To check decidability, it is sufficient due to Lemma I.4.10 and Proposition I.5.7) to check the finitely many possibilites of replacing variables in $COD(\sigma)$ by terms of an equal or smaller sort.

ii) In a regular signature, two variables are unifiable, if and only if their sorts have a common subsort.

iii) Let $\Gamma$ be a set of equations. Then an unsorted most general unifier $\sigma$ of $\Gamma$ is effectively computable. All terms in $COD(\sigma)$ are linear, variable disjoint and their depths are smaller than the maximal term depth in $\Gamma$, since $\Gamma$ is linear. Hence the check for well-sorted instances of $\sigma = \{x_1 \leftarrow t_1,\ldots,x_n \leftarrow t_n\}$ can be done by searching independently instances for the components $\{x_i \leftarrow t_i\}$ of $\sigma$. Looking for instances is done by unifying the term $t_i$ with appropriate term declarations $t{:}S$. This process again produces independent subgoals. This gives a search tree for the problem of finding an instantiation and the nodes are marked with problems $\{y \leftarrow s\}$. Since the problems are independent in different branches, we can cut a branch, if a goal has itself (in a renamed version) as a subgoal. Due to the linearity of $\Gamma$ and $\Sigma$, the depth of the terms s in the marks $\{y \leftarrow s\}$ is bounded. Hence the search tree is finite. ❑

iv) Obviously, the signature is linear, since all terms are linear. Let $n_0$ be the maximal term depth of declarations. Let $\Gamma$ be a set of equations. Similar as in iii) we can compute an unsorted idempotent most general unifier $\sigma = \{x_1 \leftarrow s_1,\ldots,x_n \leftarrow s_n\}$ of $\Gamma$. The next step is a search for well-sorted instances of $\sigma$. This creates $|I(\sigma)|$ independent search trees with subproblems (or goals) of the form $\{\{t_1, \{S_{1,1},\ldots,S_{1,k1}\}\}, \ldots, \{t_n,\{S_{n,1},\ldots,S_{n,kn}\}\}\}$, for every variable in $I((\sigma)$ one subproblem, where all $t_i$ contain the same variable. Such a subproblem means to find a substitution $\theta$ with $S_{i,j} \in S_\Sigma(\theta t_i)$ for all i,j. We argue that these search trees are finite, the main argument is that we can cut branches of this tree, if a goal has itself (in a renamed version) as a subgoal. There are two steps to expand a node in the tree:

i) If there is a term $t_i$ with $depth(t_i) > n_0$, then we take appropriate term declarations $t{:}S$ and (Robinson-) unify t and $t_i$.

   The effect of this operation is that one sort in a component $\{t_1, \{S_{1,1},\ldots,S_{1,k1}\}\}$ is removed and a problem $\{t_i', S\}$ is inserted, where $depth(t_i') < depth(t_i)$ and $V(t_i') = V(t_i)$.

ii) The same step without any conditions on the term depth of $t_i$. In this case it may be that $V(t_i') \neq V(t_i)$ and then we have to apply the substitution $\{x \leftarrow t_i'\}$ to the goal.

First we apply steps of type i) until it is no longer possible. This process terminates, hence we can reduce the goal to subgoals, where $\mathrm{depth}(t_i) \leq n_0$ for all $t_i$. However, there is only a finite number (up to renaming) of such goals, since the number of sorts is finite and the number of terms $t_i$ with the same variable and with $\mathrm{depth}(t_i) \leq n_0$ is finite. Hence the search for a well-sorted instance of $\sigma$ terminates. ■

We say a regular signature is **almost elementary**, iff the signature is linear and for every term declaration t:S and every variable $x \in V(t)$ the variable x is a direct subterm of t, i.e., in every term declaration $f(t_1,\ldots,t_n)$:S the terms $t_i$ are either ground or variables and no variable occurs twice.

In the next Proposition we show that a terminating unification algorithm for a regular, almost elementary signature is the procedure SOUP with the following modification:

Use a matching algorithm for equations $t = s_{gr}$ where $s_{gr}$ is ground, with highest priority. The operation is to replace the equation $t = s_{gr}$ by the solved system (a matcher) of $t = s_{gr}$.

**6.6 Proposition.** If the signature $\Sigma$ is regular and almost elementary, then $\Sigma$-unification is finitary and decidable.

**Proof.** We show that the algorithm SOUP terminates:

The argument is similar to the proof of Proposition 3.5.

The measure is $\mu = (\mu_1,\mu_2)$, where

$\mu_1$ is the number of variables in $\Gamma_U$, which occur not only as a left hand side of exactly one equation $x = t$ in $\Gamma_U$,

$\mu_2$ is the multiset of term depths of terms in $\Gamma_U$.

The difference is that the combination of rule VT3 with the matching rule has the following effect: either we have $y = t_i$, in which case y is a new variable, or we have the case that $s_i = t_i$ is newly introduced with a ground term $s_i$. Then matching either yields a failure or replaces this equation by solved equations.

Hence either $\mu_1$ is reduced, or $\mu_1$ is invariant , but $\mu_2$ is reduced, since $x = f(t_1,\ldots,t_n)$ is moved into the set of worked-off equations. ■

The following procedure can be used to check the regularity of finite signatures:

i) Take two term declarations s:S and r:R, such that R and S are incomparable.

ii) Compute a set $\mu U(r,s)$ under the assumption that $\Sigma$ is regular.

iii) Compute $S_\Sigma(\sigma r)$ for all $\sigma \in \mu U(r,s)$.

**6.7 Lemma.** The above procedure computes a most general term that violates regularity, i.e., without least sort.

**Proof.** Let t be a minimal term that violates regularity.

Then due to Proposition I.4.9 there exist at least two incomparable maximal sorts R,S in $S_\Sigma(\sigma t)$ and two term declarations r:R, s:S, such that t is a $\Sigma$-instance of both r and s. This means that there exists a most general $\Sigma$-unifier $\sigma$ of r and s, such that t is a $\Sigma$-instance of $\sigma r$. By minimality of t, we have that $\sigma r$ is a $\Sigma$-renaimng of t and that all terms in the codomain of $\sigma$ have a least sort, hence we can assume during unification, that regularity holds. The effect is that the unification of variables can be done as in 3.1 VV3), (see also 6.5.ii)). We have $\mathbf{S}_\Sigma(\sigma r) = \mathbf{S}_\Sigma(t)$, since $\sigma r$ and t are equivalent.

The last step iii) is needed, since it may happen that R and S are not minimal in the set $S_\Sigma(\sigma r)$. ■

**6.8 Proposition.** For almost elementary signatures regularity is decidable.

**Proof.** We use the above procedure to decide regularity. It is sufficient to show that under the regularity assumption, a minimal set of unifiers of terms in term declarations is effectively computable. This is shown in Proposition 6.6. ■

We want to mention the following open problem for finite, linear signatures:

**Open Problems:**
i) Is $\Sigma$-unification decidable for linear signatures?
ii) Is regularity of linear signatures decidable?

# Part IV
# Unification of Sorted Terms under Equational Theories.

**Overview:**

We give different methods for the unification of sorted terms provided some axioms for an equational theory are given. First we describe unification algorithms as a set of transformation rules for equational systems. Second we give an algorithm that solves unification problems by first ignoring the sort information using an unsorted algorithm and as a second step computes well-sorted instantiations.

We discuss narrowing as a universal unification algorithm for canonical term rewriting system.

**General remarks and related work.**

Research in unification problems with respect to an equational theory is an active field [Plo72,Si86, CKi85, GS87]. The problem to build equational reasoning into automated deduction procedures [WRC67, RW69, Mo69, Plo72, Di79, Bl87] is also well investigated. Our aim is to give a general unification method that works for arbitrary signatures and arbitrary equational theories. We give a rule-based procedure in the style of [CKi85, GS87].

## 1. A General Unification Algorithm For Sorted Terms under an Equational Theory.

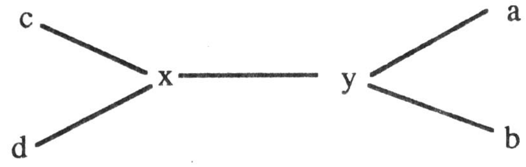
In this chapter we give a general unification procedure for arbitrary equational theories and sort-structures.

Let $\Sigma$ be a signature and let E be a symmetric axiomatization of an equational theory .

Given an equation system $\Gamma$, we may consider $\Gamma$ as a graph with the terms in $\Gamma$ as nodes and the equations in $\Gamma$ as edges. We will sometimes use the word edge as a synonym for equation in $\Gamma$.

A **path** $t_1$ — ...— $t_n$ in $\Gamma$ is a chain of equations where no $t_i$ occurs twice. A **circular path** is a path $t_1$ — ...— $t_n$ — $t_1$. A **connected component** is a set $\{t_1,\ldots,t_n\}$ of terms such that all terms $t_i$, $t_j$ are connected by a path. There is a straightforward correspondence between equation graphs and multi-equation systems: A connected component corresponds to a multi-equation, however the structure in equation graphs is richer than in multi-equations.

The following is the graph of the equation system $\{x = c, x = d, x = y, y = a, y = b\}$




Let $x_i,t_i$, $i = 1,\ldots,n$ be variable-term pairs such that $x_i$ and $t_i$ are connected and $x_i \in \mathbf{V}(t_{i-1})$ for $i = 2,\ldots,n$ and $x_n \in \mathbf{V}(t_1)$ and at least one $t_i$ is not a variable. Then we say every $x_i$ is **cyclic for** $\Gamma$. Furthermore we say the chain of pairs $(x_i,t_i)$ is a cycle and $(x_i,t_i)$ belongs to a cycle.

A connected component is in **solved form,** iff it is of the form $\{x_1,\ldots,x_n,t\}$ and $\{x_1 \leftarrow t,\ldots, x_n \leftarrow t\}$ is a well-sorted idempotent substitution. This substitution is called a **partial solution** for $\{x_1,\ldots,x_n,t\}$. An equation system $\Gamma$ is in **sequentially solved form,** iff there are no cyclic variables for $\Gamma$ and every connected component is in solved form. The corresponding **solution** $\sigma^*$ is defined as the idempotent closure of the union of all partial solutions for all connected components. In Lemma 1.6 we show that such a solution is a unifier of $\Gamma$ (cf. I.10.6 ff.)

A variable $x \in \Gamma$ is **isolated,** iff x has only one occurrence in $\Gamma$.

In the following we use sometimes the expression 'new version' for an axiom or a term declaration and mean a renamed version of an axiom or a term declaration such that they contain only new variables.

We use the following rules to transform equation systems:

**1.1 Definition.** The unification procedure GENSEQUP is defined by the following rules:

<u>Tautology</u>

Tau) If P is a circular path, then delete some edge in P.

<u>Decomposition:</u>

De) Let $s — u_1 —\ldots— u_m— t$ be a path P, where $t = f(t_1,\ldots,t_n)$, and $s = f(s_1,\ldots,s_n)$ and $u_i = f(u_{i1},\ldots,u_{in})$ or $u_i$ is a variable, where $1 \leq i \leq m$.

i) For all variables $u_i$ on the path we instantiate P with $\{u_i \leftarrow f(u_{i1},\ldots,u_{in})\}$, where $f(u_{i1},\ldots,u_{in}){:}S_i$ is a new version of a term declaration and $S_i \sqsubseteq \mathbf{LS}_\Sigma(u_i)$, and we add the equation $u_i — f(u_{i1},\ldots,u_{in})$.

ii) For every $1 \leq j \leq n$ we add the path $s_j — u_{1j} —\ldots—u_{mj} — t_j$ to the equation graph.

iii) All the edges of the instantiated original path are deleted.

Declaration introduction

Di) Let O be a connected component with $x \in O$.

Let $f(r_1,\ldots,r_n):S$ be a new version of a term declaration, such that $S \sqsubseteq \mathbf{LS}_\Sigma(x)$, and $hd(t) = f$ for all nonvariable terms $t \in O$.

We instantiate O with $\{x \leftarrow f(r_1,\ldots,r_n)\}$, and add the edge $x — f(r_1,\ldots,r_n)$.

Conditions for application:

Either

1) There is at most one nonvariable term $t \in O$ and $(x,t)$ belongs to a cycle

or 2) There is at most one nonvariable term $t \in O$ and $\mathbf{LS}_\Sigma(x) \not\in \mathbf{S}_\Sigma(t)$.

or 3) There are only variables in O and there exists a variable $y \in O$ with $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(x) \not\sqsupseteq \mathbf{LS}_\Sigma(y)$.

Mutation:

Mu) We replace the edge $s — t$ by $s — l$ and $r — t$, where $l = r$ is a new version of an axiom.

As condition for application we have:

$l = r$ is a collapse axiom or a decomposition step with l or r becomes possible.

Furthermore one of the following should hold:

$s = t$ is on a path between nonvariable terms

$s = t$ is on a path connecting a variable x and t and $(x,t)$ belongs to a cycle.

$s = t$ is on a path connecting a variable x and a nonvariable term t, with $\mathbf{LS}_\Sigma(x) \not\in \mathbf{S}_\Sigma(t)$.

$s = t$ is on a path connecting two variables x,y with $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(x) \not\sqsupseteq \mathbf{LS}_\Sigma(y)$, the connected component of x consists only of variables and $l = r$ is a collapse axiom.

Variable introduction:

Vi) Let $O = \{x_1,\ldots,x_n\}$ be a connected component consisting only of variables. Let $x,y \in O$ be variables such that $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(x) \not\sqsupseteq \mathbf{LS}_\Sigma(y)$.

Let z be a new variable, such that $\mathbf{LS}_\Sigma(z) \sqsubseteq \mathbf{LS}_\Sigma(x)$ and $\mathbf{LS}_\Sigma(z) \sqsubseteq \mathbf{LS}_\Sigma(y)$.

We instantiate $\Gamma$ with $\{x \leftarrow z, y \leftarrow z\}$, and afterwards we add the equations $x = z$ and $y = z$. ■

We following rules might also be used, but are not necessary (cf. I.13.6).

**1.2 Definition.**

Instantiation:

In) Let x and t be connected and let $\{x \leftarrow t\}$ be a well-sorted substitution.

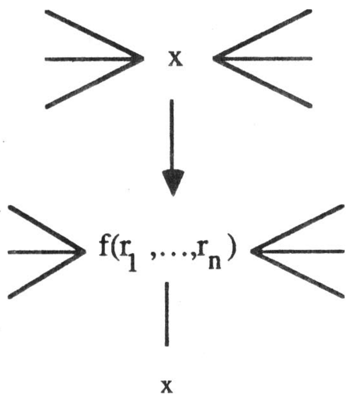
Then we apply $\{x \leftarrow t\}$ to $\Gamma$ and add the edge $x$ — $t$ to $\{x \leftarrow t\}\Gamma$.

Extended Instantiation rule:

Ex-In) Let $x_i$ and $t_i$ be connected for all $i = 1,\ldots,n$ and let $\tau := \{x_i \leftarrow t_i \mid i = 1,\ldots,n\}$ be a well-sorted substitution.

Then we apply $\tau$ to $\Gamma$ and add all the edges $x_i$ — $t_i$ to $\tau\Gamma$. ■

The instantiation of a variable x in the equation system with a term declaration $f(r_1,\ldots,r_n)$ can be illustrated by the following picture:




In order to give a more comprehensive account of the above algorithm, we describe the essential steps in the already known notation. The steps are:

i) $t_1 = t_2 \;\&\; \ldots \;\&\; t_{n-1} = t_n \;\&\; t_n = t_1 \;\Rightarrow\; t_1 = t_2 \;\&\; \ldots \;\&\; t_{n-1} = t_n$.

ii) $x = t \;\&\; \Gamma \;\Rightarrow\; x = f(r_1,\ldots,r_n) \;\&\; \{x \leftarrow f(r_1,\ldots,r_n)\}\, x = t \;\&\; \{x \leftarrow f(r_1,\ldots,r_n)\}\Gamma$

For a new version of a term declaration $f(r_1,\ldots,r_n){:}S$, such that $S \sqsubseteq \mathbf{LS}_\Sigma(x)$.

iii) $s = t \;\Rightarrow\; s = r \;\&\; l = t$ where $l = r$ is a new version of an axiom.

iv) $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n) \;\Rightarrow\; s_1 = t_1 \;\&\; \ldots \;\&\; s_n = t_n$

v) $x = x_1 \;\&\; x_1 = x_2 \;\&\; \ldots \;\&\; x_{n-1} = x_n \;\&\; x_n = y \;\&\; \Gamma \;\Rightarrow$

$x = z \;\&\; y = z \;\&\; z = x_1 \;\&\; x_1 = x_2 \;\&\; \ldots \;\&\; x_{n-1} = x_n \;\&\; x_n = z \;\&$ $\{x \leftarrow z,\, y \leftarrow z\}\Gamma$

where $\{x \leftarrow z,\, y \leftarrow z\}$ is well-sorted.

The conditions under which the steps are applied are controlled by the descriptions of the steps in 1.1. For example a decomposition step consists of some applications of ii) and some applications of iv).

**1.3 Example.**

Let $E = \{c = a, c = b\}$ and let $\Gamma = \langle x = a, x = b\rangle$ and consider the solution $\theta$ with $\theta x = c$. The procedure gives the chain $\langle x = a, x = b\rangle \Rightarrow \langle x = a, x = c, b = b\rangle \Rightarrow \langle x = a, x = c\rangle \Rightarrow \langle a = a, x = c\rangle \Rightarrow \langle x = c\rangle$. The last system of equations is solved and gives $\{x \leftarrow c\}$ as solution. The last but one system of equations does not contain $x = c$ twice, since we consider $\Gamma$ as a set.

**1.4 Lemma.** The above rule system is correct. ■

**1.5 Lemma.** The instantiation rule and the extended instantiation rule are complete.

**Proof.** Follows from I.13.6.iii). ■

**1.6 Lemma.** Let $\theta$ be a unifier of a sequentially solved equation system $\Gamma$ with solution $\sigma$. Then we have: $\sigma \leq_{\Sigma,E} \theta\ [V(M)]$

**Proof.** Follows from the completeness of the instantiation rule and from the considerations in paragraph I.10 and from Lemma I.13.11. ■

Now we address the problem of unification completeness of the procedure defined in 1.1.

Let $\theta$ be a unifier of $\Gamma$. Then according to II.5.1 there exists a proof that $\theta s =_{\Sigma,E} \theta t$ for all $s = t$ in $\Gamma$. The triple $(\Gamma,P,\theta)$ consists of an equation system $\Gamma$, a unifier $\theta$ of $\Gamma$, and a proof P that $\theta$ is a unifier of $\Gamma$. We assume that every equation $s = t$ is labelled with the proof of $\theta s =_{\Sigma,E} \theta t$. Such a proof corresponds to a chain $r_0,\ldots,r_n$, such that $\theta s = r_0$, $\theta t = r_n$ and the proof of $r_i = r_{i+1}$ is a step II.5.1 iv) or II.5.1 v), corresponding to a congruence step or an axiom step. We say s and t are connected by a congruence proof, if $\theta s = \theta t$ is proved by a congruence step. This means that the chain has at most length one and includes the case that the proof is empty, i.e. $\theta s = \theta t$. Otherwise at least one axiom step is necessary (at toplevel) to prove $\theta s =_{\Sigma,E} \theta t$. We assume that there is no sharing among proofs.

Now we prove that this procedure is a complete unification procedure. The idea of the following proof is, given a unifier $\theta$ for $\Gamma$, we use the information in $(\Gamma,P,\theta)$ to select the next step and show how to construct the resulting $(\Gamma',P',\theta')$ after application of the rules. We show that this procedure strictly decreases a well-founded complexity measure of $(\Gamma',P',\theta')$. During this procedure we extend $\theta$ to new variables but do not change it on old variables.
The second part of the proof is to show that either we can reduce the given complexity measure by some steps or the resulting equation system is in sequentially solved form.

**1.7 Theorem.** The unification procedure GENSEQUP is a complete unification procedure.

**Proof.** Let $\theta$ be a unifier of $\Gamma$ and let the triple $(\Gamma,P,\theta)$ be as described above.

We say how to construct $(\Gamma',P',\theta')$ from $(\Gamma,P,\theta)$ for every step of the above procedure. We denote these steps by the name of the original step and a star (*). Furthermore in order to show completeness of the procedure, we demonstrate that $\theta'$ is an extension of $\theta$ and a solution of $\Gamma'$. Additional restrictions for the applicability of these steps in terms of the proof P are given.

Tau*) Delete the proof corresponding to the deleted equation.

Di*) This step is made only in the case that $\theta x$ is not a variable, and either x and the nonvariable term t are connected via congruence proofs or we have case 3) of Di).
According to Lemma I.4.9 there exists a term declaration $f(r_1,\ldots,r_n)$ and a substitution $\eta$ such that $\eta f(r_1,\ldots,r_n) = \theta x$. We choose this term declaration in the step Di). We define $\theta' := \theta \cup \eta$. To show $\theta'\{x \leftarrow f(r_1,\ldots,r_n)\} = \theta\ [V(\Gamma)]$, it suffices to consider the variable x: $\theta'\{x \leftarrow f(r_1,\ldots,r_n)\}x = \theta' f(r_1,\ldots,r_n) = \eta f(r_1,\ldots,r_n) = \theta x$. Hence there is no change in the proofs of the old equations. The new equation $x = f(r_1,\ldots,r_n)$ has an empty proof, since $\theta' x = \theta' f(r_1,\ldots,r_n)$. Hence $\theta'$ is a unifier of the resulting $\Gamma'$.

De*): We decompose the connected terms $f(s_1,\ldots,s_n)$ and $f(t_1,\ldots,t_n)$ only in the case that a path exists such that all terms are connected via congruence proofs (with respect to $\theta$).
Let the path be: $s — u_1 —\ldots— u_m — t$.
If there is an instantiation step before decomposition, let $u_i$ be the instantiated variable.
The same arguments as for Di*) apply to $u_i$ and $f(u_{i1},\ldots,u_{in})$, hence we do not repeat them.
Now we can assume that all $u_i$ are nonvariable terms.
We remove all proofs of the equations in the path. The proofs of the new equations $s_i — u_{1i} —\ldots— u_{mi} — t_i$ are extracted from the proofs of the original equations.
We can choose $\theta' := \theta$ in this case.

Mu*) This step is made only in the case where the proof of $\theta s =_{\Sigma,E} \theta t$ has an axiom step at toplevel and the step Mu is applicable.
Let $s = t$ be the equation and $r_0, \ldots,r_k,r_{k+1},\ldots,r_n$ be the terms in the proof of $\theta s =_{\Sigma,E} \theta t$ and let $l = r$ be a new version of an axiom and $\eta$ be a substitution such that $\eta l = r_k$ and $\eta r = r_{k+1}$.
We define $\theta' := \theta \cup \eta$ and the proofs for the new equations we have the chain $r_0,\ldots,r_k$ and $r_{k+1},\ldots,r_n$ , respectively.

Vi*) This step is made in the case where $\theta x_i$ is a variable for all $x_i$ in a connected component O, for all $x_i,x_j \in$ O we have $\theta x_i = \theta x_j$ and O is not in solved form.
Since for all $x_i,x_j \in$ O we have $\theta x_i = \theta y_j \in \mathbf{V}$, there exists a variable z, such that $\mathbf{LS}_\Sigma(z) \sqsubseteq \mathbf{LS}_\Sigma(x_i)$ for all $x_i \in$ O and $\mathbf{LS}_\Sigma(\theta x_i) \sqsubseteq \mathbf{LS}_\Sigma(z)$. Similar as in step Di*) we define $\theta' := \theta \cup \eta$, where $\eta = \{x_i \leftarrow z\}$ and the same arguments as for Di*) apply. ❑

We define a complexity measure on $(\Gamma,P,\theta)$ as $\mu(\Gamma,P,\theta) = (\mu_1, \mu_2, \mu_3, \mu_4)$, where $\mu_1$ is the number of axiom-steps in the proof P, $\mu_2$ is the multiset of all term depths $\theta x$, where x ranges over all nonisolated variables in $\Gamma$, $\mu_3$ is the multiset of all the maximal term depths for all equations in $\Gamma$, and $\mu_4$ is the number of equations in $\Gamma$. We assume a lexicographical ordering on the 4-tuples $(\mu_1, \mu_2, \mu_3, \mu_4)$. Obviously this measure is well-founded.
We show that the measure is strictly reduced in every step:
First it is clear that the steps De*) Vi*) Di)* Tau*) do not increase the number of axiom-steps in the proof. The step Mu*) strictly decreases the number of axiom steps in $(\Gamma,P,\theta)$.
The instantiation part of De*), the rule Di*)and Vi*) strictly decrease $\mu_2$.
The decomposition part of step De*) does not change $\mu_2$, but strictly decreases $\mu_3$.
The rule Tau*) strictly decreases $\mu_4$. This means that the above process terminates. ❑

We have to show that starting with an equation system and proceeding in the above described way, we can apply steps until we reach a sequentially solved system.
Assume by contradiction we have reached an equation system $\Gamma$ that is not in sequentially solved form such that it is not possible to make any of the above *-steps.
The proof proceeds in threee steps:

i) Every connected component has at most one nonvariable term:
Otherwise we can either apply De*) or Mu*).
ii) There is no cyclic variable for $\Gamma$:
Assume there is a pair (x,t) that is connected, belongs to a cycle and t is a nonvariable term. In the case $\theta x$ and $\theta t$ are connected in $(\Gamma,P,\theta)$ by a congruence proof then Di*) is applicable. Otherwise Mu*) is applicable.
iii) Every connected component O is in solved form:
Let O be a component that is not in solved form. We have the two cases that O contains a nonvariable term or not:
    a) O contains a nonvariable term t. Then O contains also a variable x, since otherwise we can apply Tau*). Furthermore there is a variable x, such that $\mathbf{LS}_\Sigma(x) \notin \mathbf{S}_\Sigma(t)$. But then we can either apply Di*) or Mu*) depending on whether x and t are connected via a congruence proof or not.

b) O contains only variables. There are different variables $x,y \in O$ such that $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$ and $\mathbf{LS}_\Sigma(x) \not\sqsubseteq \mathbf{LS}_\Sigma(y)$, since otherwise there exists a variable in O with minimal sort in O.
If either $\theta x$ or $\theta y$ is a nonvariable term, then we can apply step Di*).
Hence $\theta x$ and $\theta y$ are variables. Since $\theta x =_{\Sigma,E} \theta y$ we either have $\theta x = \theta y$ in which case Vi*) is applicable or there exists a collapse axiom $r = l$ at toplevel in the proof of $\theta x =_{\Sigma,E} \theta y$ , in which case 3) of Mu applies.

From the above it follows that the final $\Gamma$ is sequentially solved and that the solution is more general than $\theta$. ■

The extension of the unification methods as described in [GS87] and also in [Bl87] is roughly equivalent to supplement the procedure in III.2 by application of axioms, i.e., to replace $s = t$ by $s = l$ & $r = t$, where $l = r$ is variant of an axiom and by a cycle-elimination rule. In other words, if the equation $x = t$ has to be solved and $x \in V(t)$ , then x can be replaced by a term of the form $f(y_1,\ldots,y_n)$.

This unification methods cannot be extended to sorted signatures without adding rules, which either introduce steps similar to paramodulation into variables or else make use of functionally reflexive axioms:

**1.8 Example.**

i) Simple signature:
Let $\Sigma := \{A,B \sqsubset TOP, a{:}A, b{:}B\}$ , let $E := \{a=b\}$ and let $\Gamma := \langle x_A = y_B \rangle$.
$\Gamma$ is E-unifiable, but it is necessary to use the equation $a = b$ though neither the topsymbol of $x_A$ nor $y_B$ is a or b.

ii) Nonsimple signature:
Let $\Sigma := \{A,B,C,D \sqsubset TOP, a{:}A, b{:}B, c{:}C, d{:}D, g{:}TOP \to TOP, g{:}A \to C, g{:}B \to D\}$ and let $E := \{a=b\}$ and let $\Gamma := \langle x_C = y_D \rangle$. The solution is $\{x_C \leftarrow g(a), y_D \leftarrow g(b)\}$.
This solution can only be found, if $x_C$ or $y_D$ is instantiated with a term of the form $g(z)$. ■

There are equational theories for which the above procedure can be improved by adding some nonunifiability checks. In equational theories that have a regular E-semantical sort-assignment (cf. II.9) terms s,t are unifiable only if the sorts of these terms have a common subsort. Sometimes this information can be used to cut branches of the search tree. Furthermore if in addition the sort-structure is a semilattice, then the most general unifier of two variables x,y can be chosen of the form $\{x \leftarrow z, y \leftarrow z\}$, where the sort of z is the greatest common subsort of the sorts of x and y.

## 2. Finite and $\Omega$-free Equational Theories.

Recall that an equational theory $\mathcal{E}$ is finite if the equivalence classes of $=_{\Sigma,E}$ are finite, and that $\mathcal{E}$ is $\Omega$-free, iff $f(s_1,\ldots,s_n) =_{\Sigma,E} f(t_1,\ldots,t_n)$ implies that $s_i =_{\Sigma,E} t_i$ .

First we deal with finite equational theories.
For unsorted finite equational theories, it is known that the word-problem is decidable, that matching is finitary, minimal unifier sets exist and are recursively enumerable (cf. [Sz82] ).
We show in this paragraph that these result can be lifted to the sorted case.

**2.1 Lemma.** Let $\mathcal{E}$ be a finite equational theory and assume $\dashrightarrow_E$ to be demodulation-complete. Then the word-problem for $\mathcal{E}$ is decidable.

**Proof.** Given a term s, the computation of its equivalence class is possible and terminates, hence equality of two terms is decidable. ■

It is an open problem, whether the requirement of demodulation-completeness can be omitted.

The connection between the unsorted theory and a sorted theory is as follows:

**2.2 Lemma.** i) If $=_{\overline{\Sigma},E}$ is finite, then $=_{\Sigma,E}$ is finite.

ii) The converse is false.

**Proof.** i) is trivial

ii) The following is an example for this claim: Let $\Sigma := \{A\ , B, a{:}A, f{:}A \to B\}$ and let $E := \{f(x_A) = x_A\}$. Then $=_{\Sigma,E}$ is finite, but $=_{\overline{\Sigma},E}$ is not finite, since there exists an infinite equivalence class: $\{a, f(a), f^2(a),\ldots\}$. ■

The above lemma provides an example also for the statement that $=_{\Sigma,E}$ is finite on $\mathbf{T}_{\Sigma}$, but not on $\mathbf{T}_{\overline{\Sigma}}$.

The proofs of the next lemmas and propositions in this paragraph are the proofs in [Sz82] adapted to the order-sorted case.

**2.3 Lemma.** Finite theories are regular.

**Proof.** There exist terms s,t such that $s =_{\Sigma,E} t$ and $\mathbf{V}(s) \neq \mathbf{V}(t)$. Substituting arbitrary variables we get an infinite equivalence class. ■

This proof differs slightly from the proof in [Sz82], which showed a little more: there exists an infinite equivalence classe of $=_{\Sigma,E}$ in the Herbrand-universe, if the theory is nonregular.
This is not true in general for sorted equational theories:

Let $\Sigma := \{A\ , B, a{:}A, f{:}A \rightarrow B\}$ and let $E := \{f(x_A) = f(y_A)\ \}$. Then E is not finite, but all equivalence classes in the Herbrand universe are finite, since the set of well-sorted ground terms is exactly $\{a, f(a)\}$, which is finite.
This observation suggests a different definition of finiteness in terms of the initial term algebra. This definiton in conjunction with an appropriate notion of subsumption may lead to similar results. However, we do not follow up these lines.

**2.4 Proposition.** For every finite theory $\mathcal{E}$ with demodulation-complete axiomatizations we have:

i) Matching is decidable.

ii) Most general matching sets are finite and effectively computable.

**Proof.** Let $\Gamma = \langle s_1 \ll t_1,\ldots,s_n \ll t_n\rangle_E$ be a matching problem and let U be the set of solutions, i.e. the set $\{\sigma \mid \sigma s_i =_{\Sigma,E} t_i,\ i = 1,\ldots,n$ and $DOM(\sigma) \subseteq V(s_1,\ldots,s_n) - V(t_1,\ldots,t_n)\ \}$.
Since the number of equivalence classes modulo $=_{\Sigma,E}$ is finite, there is at most a (computable) finite number of substitutions in U. ■

**2.5 Proposition.** For every finite theory $\mathcal{E}$ with demodulation-complete axiomatization and for all equation systems $\Gamma$, there exists a set of most general unifiers which is recursively enumerable.

**Proof.**

i) In order to show that minimal unifier sets exist, we show that $\mathcal{E}$ is Noetherian.
Let $\sigma_1 >_{\Sigma,E} \sigma_2 >_{\Sigma,E} \ldots$ [W] be a descending chain of substitutions for a finite set of variables W.
Then we can assume by Lemma I.10.5 that the maximal depth of terms in the codomain of $\sigma_i$ is increasing. There exist well-sorted substitutions $\lambda_i$ , $i = 1,2,\ldots$ with $\sigma_{i-1} =_{\Sigma,E} \lambda_i\sigma_i$ [W]. This is a contradiction to the finiteness of the theory $\mathcal{E}$.

ii) Since E-subsumption of substitutions is decidable and a set of all unifiers is recursively enumerable with nondecreasing maximal term depth in their codomain, it is sufficient to show that there is an algorithm that decides whether a unifier is a minimal one. The number of nonequivalent ($\equiv_\Sigma$ [W]) substitutions that are more general than a given substitution $\sigma$ is finite and effectively computable, hence minimality of unifiers is decidable. ■

Our interest in $\Omega$-free theories comes from the fact that in the unsorted case the $\Omega$-free theories are exactly the regular, unitary matching theories.
In the sorted case, this relation is true, provided some additional conditions hold:

**2.6 Example.** There are $\Omega$-free, nonregular theories:

Let $\Sigma = \{A,B, f : A \rightarrow B\}$ and let $E := \{x_A = y_A\}$. Then $\mathcal{E}$ is consistent, $\Omega$-free, but not regular.

**2.7 Proposition.** i) If $\mathcal{E}$ is $\Omega$-free, then $\mathcal{E}$ is unitary matching.

ii) If $\mathcal{E}$ is regular, unitary matching and for every function there is a maximal function declaration, then $\mathcal{E}$ is $\Omega$-free.

**Proof.**

i) Assume $\mathcal{E}$ is not unitary matching, then there exists a term t and two different idempotent substitution $\sigma,\tau$ such that $\sigma t =_{\Sigma,E} \tau t$. We can assume that t is such a term with minimal term depth. Obviously t is not a variable or constant. Hence $\Omega$-freeness implies that with $t = f(t_1,...,t_n)$ we have $\sigma t_i =_{\Sigma,E} \tau t_i$. Repeatedly applied, this gives $\sigma =_{\Sigma,E} \tau\ [V(t)]$.

ii) Assume there is a counterexample $\mathcal{E}$, that is not $\Omega$-free with the above properties. Then there exists terms $f(s_1,...,s_n)$ and $f(t_1,...,t_n)$, such that $f(s_1,...,s_n) =_{\Sigma,E} f(t_1,...,t_n)$, but for some i, we have not $s_i =_{\Sigma,E} t_i$. Now consider the matching problem $\langle f(x_1,...,x_n) \ll f(t_1,...,t_n)\rangle$, where $x_i$ are new variables and the term $f(x_1,...,x_n)$ corresponds to the maximal function declaration for f. Then there are two different matchers $\{x_i \leftarrow t_i\}$ and $\{x_i \leftarrow s_i\}$ and since in regular theories every matcher is minimal (cf I.11 ), we have reached a contradiction. ■

## 3. Unification in Sort-Preserving and Congruence-Closed Theories.

An important property of congruence-closed equational theories is that unifiers of an equation system can be computed in a particularily simple way. It is done by first computing unifiers ignoring the sort information and as a second step the sort handling is done without reference to the equational theory.

**3.1 Proposition.** Let $\mathcal{E}$ be a congruence-closed equational theory. Then

$U_E(\Gamma) \cap SUB_\Sigma = U_{\Sigma,E}(\Gamma)$

**Proof.** We prove the nontrivial direction:

Let $\sigma \in U_E(\Gamma) \cap SUB_\Sigma$. Then $\sigma s_i =_{\Sigma,E} \sigma t_i$ for all $s_i = t_i \in \Gamma$. Since $\mathcal{E}$ is congruence-closed, we have also $\sigma s_i =_{\Sigma,E} \sigma t_i$ for all $s_i = t_i \in \Gamma$, hence $\sigma \in U_{\Sigma,E}(\Gamma)$. ■

Unfortunately, this nice property is not true for not congruence-closed theories and furthermore there are in fact interesting noncongruence-closed theories. For example feature

unification [SA87] is unification in a theory that is not congruence-closed. In a sense the above property characterizes congruence-closed equational theories: Let $\mathcal{E}$ be a noncongruence-closed theory and let s,t be terms such that $s =_{\overline{\Sigma},E} t$, but not $s =_{\Sigma,E} t$. Then $Id \in U_{\overline{\Sigma},E}(\langle s = t\rangle) \cap SUB_{\Sigma}$, but $Id \notin U_{\Sigma,E}(\langle s = t\rangle)$.
Theories with collapse axioms are likely to be noncongruence-closed. For example if there are sorts $A \sqsupset B$, $f{:}A \to A$ and E contains the axiom $f(x_B) = x_B$, then for $\mathcal{E}$ to be congruence-closed it is necessary that $f(x_A) =_{\Sigma,E} x_A$ holds.

Without additional requirements there is little hope to obtain general results for congruence-closed equational theories. One requirement is that the equational theory should be sort-preserving. We will also take the requirement into account that the equational theory has a sort-decreasing term rewriting system.
In this paragraph we concentrate on the sort-preservation of equational theories.

**3.2 Assumption.** Throughout this paragraph we assume that equational theories are sort-preserving, congruence-closed and the sort-structure has one maximal sort.

Note that there is some preliminary work in paragraph II.6 that investigates criteria for sort-preservation and congruence-closedness. Furthermore some properties of substitutions in sort-preserving and congruence-closed theories are stated in II.6.15 - II.6.18.
In paragraph 4.4 there are also some examples that demonstrate the consequences of these assumptions, for example a group as a sort-preserving congruence with an underlying regular and elementary signature can only have a many-sorted sort-structure.

An advantage of sort-preserving equational theories is that equality preserves well-sortedness of substitutions, i.e. $\sigma =_{\Sigma,E} \tau$ and $\sigma \in SUB_{\Sigma}$ implies $\tau \in SUB_{\Sigma}$, which may be false in general.

We assume in the following that complete and minimal sets with respect to $\overline{\Sigma}$ are chosen such that all variables in the codomain of substitutions are of maximal sort (cf. Corollary II.7.8)

**3.3 Main Theorem.**

Let $=_{\Sigma,E}$ be an equational theory. Let W be a finite set of variables and let $U \subseteq SUB_{\overline{\Sigma}}$ be an upper segment with respect to $\leq_E[W]$. Then
$\{\omega\tau \mid \tau \in c_E(U), \omega \in \mu W_{\Sigma}(\tau)\}$ is a complete subset of $U \cap SUB_{\Sigma}$.

**Proof.** We show that this set is a correct, complete subset of $U \cap SUB_{\Sigma}$.

i) Correctness: Trivial, since U is an upper segment.

ii) Completeness: Let $\theta \in U \cap SUB_{\Sigma}$. Then there exists a substitution $\lambda \in SUB_{\overline{\Sigma}}$ with $DOM(\lambda) = V(\tau W)$ and a substitution $\tau \in c_E(U)$ with $\lambda\tau =_{\overline{\Sigma},E} \theta$ [W]. The assumption congruence-closedness implies that $\lambda\tau =_{\Sigma,E} \theta$ [W] and hence by sort-preservation $\lambda\tau x$ is well-sorted for all $x \in W$. The signature is subterm-closed, hence all terms in $COD(\lambda)$ are well-sorted. Since we have assumed that all variables in $I(\tau)$ are of maximal sort, we have $\lambda \in SUB_{\Sigma}$.

III. 1.3 implies that there exists some set $\mu W_{\Sigma}(\tau)$.

There exists an $\omega \in \mu W_{\Sigma}(\tau)$ and an $\eta \in SUB_{\Sigma}$ such that $\lambda = \eta\omega$ [ $V(\tau W)$ ].

From $\theta =_{\Sigma,E} \eta\omega\tau$ [W] it follows $\theta \geq_{\Sigma,E} \omega\tau$ [W] . ■

**3.4 Theorem.** Let the conditions of Theorem 3.3 be satisfied. If in addition $\Sigma$ is elementary and a minimal subset $\mu_{\overline{\Sigma},E}(U)$ exists, then we obtain a minimal subset of $U \cap SUB_{\Sigma}$ as the union of the minimal subsets of $\{\omega\tau \mid \omega \in \mu W_{\Sigma}(\tau)\}$ where $\tau$ ranges over the set $\mu_{\overline{\Sigma},E}(U)$.

**Proof.** For a fixed substitution $\tau \in \mu_{\overline{\Sigma},E}(U)$ the set $\mu W_{\Sigma}(\tau)$ is finite, hence the set $\{\omega\tau \mid \omega \in \mu W_{\Sigma}(\tau)\}$ is finite, thus there exists a minimal subset of $\{\omega\tau \mid \omega \in \mu W_{\Sigma}(\tau)\}$.

We show that it is sufficient to minimize the finite subsets in order to obtain a minimal subset of $U \cap SUB_{\Sigma}$.

Let $\tau_1, \tau_2 \in \mu_{\overline{\Sigma},E}(U)$ and let $\omega_1,\omega_2 \in \mu W_{\Sigma}(\tau)$ such that $\omega_1\tau_1 \leq_{\Sigma,E} \omega_2\tau_2$ [W]. Then there exists a well-sorted substitution $\lambda$ with $\lambda\omega_1\tau_1 =_{\Sigma,E} \omega_2\tau_2$ [W]. Proposition III.1.5 implies that $\omega_2$ is a renaming, hence by applying $\omega_2^{-}$ we obtain $\omega_2^{-}\lambda\omega_1\tau_1 =_{\overline{\Sigma},E} \omega_2^{-}\omega_2\tau_2 =_{\overline{\Sigma},E} \tau_2$ [W]. The minimality of $\mu_{\overline{\Sigma},E}(U)$ implies that $\tau_1 = \tau_2$. ■

In order to apply these theorems to unification problems note that the set $U_{\overline{\Sigma},E}(\Gamma)$ is always an upper segment with respect to the ordering $\leq_{\overline{\Sigma},E}[W]$ (cf. II.6 )

**3.5 Corollary.** The set $\{\omega\tau \mid \tau \in cU_{\overline{\Sigma},E}(\Gamma), \omega \in \mu W_{\Sigma}(\tau)\}$ is a complete subset of $U_{\Sigma,E}(\Gamma) \cap SUB_{\Sigma}$ with respect to $\leq_{\Sigma,E}[W]$ . ■

In the case of elementary signatures, we have:

**3.6 Corollary.** For an elementary signature a minimal, complete subset of $U_{\Sigma,E}(\Gamma) \cap SUB_{\Sigma}$ with respect to $\leq_{\Sigma,E}[W]$ can be obtained as the union of minimal, complete subsets of the finite sets $\{\omega\tau \mid \omega \in \mu W_{\Sigma}(\tau)\}$ for $\tau \in cU_{\overline{\Sigma},E}(\Gamma)$ . ■

An interesting application of these theorems to matching problems in regular theories is the following: compute the matchers with an unsorted algorithm and delete the ill-sorted matchers:

**3.7 Corollary.** If $\mathcal{E}$ is regular, then a minimal and complete set of matchers can be obtained as: $\mu M_{\Sigma,E}(\Gamma) = SUB_{\Sigma} \cap \mu M_{\overline{\Sigma},E}(\Gamma)$.
That means that a regular theory $\mathcal{E}$ that has effectively finitary $\overline{\Sigma}$-matching problems, has also effectively finitary $\Sigma$-matching problems. ■

**3.8 Corollary.** Let $\Sigma$ be elementary, let $\mathcal{E}$ be regular and effectively finitary matching.

i) If $\mathcal{E}$ is of effective $\overline{\Sigma}$ -unification type 1 or $\omega$, then minimal set of $\Sigma$-unifiers are finite and effectively computable.

ii) If $\mathcal{E}$ is of $\overline{\Sigma}$-unification type $\infty$ and minimal set of $\overline{\Sigma}$-unifers are recursively enumerable , then minimal sets of $\Sigma$-unifers are recursively enumerable.

From Paragraph 2 it follows that Corollary 3.8 is applicable to all finite equational theories.

**3.9 Example.** The minimizing step for elementary signatures (Theorem 3.4) may be necessary to obtain a minimal set of unifiers:
Let $\Sigma = \{ A \sqsupset B$, $f: AxA \to A;\ AxB \to B;\ BxA \to B;\ BxB \to B,$
$g: B \to B;\ \ h: B \to B\}$
Let $E := \{ g(f(x_A, y_B)) = h(f(x_A, y_B)),\ f(x_A, y_A) = f(y_A, x_A) \}$.
This signature is elementary, furthermore the generated congruence is sort-preserving and congruence-closed due to Propositions II.6.11 and II.6.2.
Now consider the unification problem $\langle g(z_B) = h(z_B) \rangle$.
The unsorted solution is: $\{z_B \leftarrow f(x_1\ y_1)\ \}$, where $x_1$ , $y_1$ are of sort A.
We have $\mu W(z_B \leftarrow f(x_1\ y_1)) = \{(x_1 \leftarrow x_{1,A}, y_1 \leftarrow x_{1,B});(x_1 \leftarrow x'_{1,B}, y_1 \leftarrow x'_{1,A})\ \}$.
The combination, however, can be minimized, since f is commutative and the final set of unifiers is: $\{\ \{\ z_B \leftarrow f(x_{1,A}\ y_{1,B})\ \}\ \}$. ■

We give an example that unification may become undecidable, if sorts are introduced. A similar result that can be translated to ours can be found in [Bü86], where constants instead of sorts are added:

**3.10 Example.** Unification may become undecidable, if sorts are introduced even if the resulting congruence is sort-preserving and congruence-closed:
It is well-known that unification under distributivity and associativity is undecidable [Sz82]. The axioms are:

$x*(y+z) = x*y + x*z$ and $(x+y)*z = x*y + y*z$ (distributivity)

$x + (y + z) = (x + y) + z$ (associativity)

In order to construct an appropriate example, we add a third argument to + and * and write f for * and g for +. Furthermore we assume that we have a constant $a$ in the signature. Distributivity and associativity are translated into the following axioms:

$$f(x_1\ g(x_2\ x_3\ x_4)\ x_4) = g(f(x_1\ x_2\ x_4)\ f(x_1\ x_3\ x_4)\ x_4)$$
$$f(\ g(x_1\ x_2\ x_4)\ x_3\ x_4) = g(f(x_1\ x_3\ x_4)\ f(x_2\ x_3\ x_4)\ x_4)$$
$$g(g(x_1\ x_2\ x_4)\ x_3\ x_4) = g(x_1\ g(x_2\ x_3\ x_4)\ x_4)$$

Now assume the following axioms, and note that all problems are trivially unifiable in this theory:

$$f(x_1\ x_2\ a\ ) = a$$
$$f(x_1\ x_2\ f(x_3\ x_4\ x_5)\ ) = a$$
$$f(x_1\ x_2\ g(x_3\ x_4\ x_5)\ ) = a$$
$$g(x_1\ x_2\ a\ ) = a$$
$$g(x_1\ x_2\ f(x_3\ x_4\ x_5)\ ) = a$$
$$g(x_1\ x_2\ g(x_3\ x_4\ x_5)\ ) = a$$

We have that every equation system $\Gamma$ is trivially unifiable by instantiating the third argument of the toplevel function symbol, if necessary. Hence unification is decidable.

Now we add sorts to the signature and assume that
$\Sigma = \{A \sqsupset B, f{:}AxAxA \to A, g{:}AxAxA \to A, a{:}A\}$.
Furthermore we assume that all variables in the above axioms are of sort A. Obviously this theory is sort-preserving, since no term of sort B is equal to some term of sort A. In fact all terms of sort B are variables. Furthermore this theory is congruence-closed due to Proposition II.6.2.
We show that unification in this new theory with sorts is undecidable, since DA-unification problems are embeddable in this theory:
Consider the subset $T_B$ of terms that have as third argument of every subterm the same variable of sort B, say $z_B$. The mapping $\varphi$ defined by $f(s\ t\ z_B) \longmapsto s*t$ and $g(s\ t\ z_B) \longmapsto s+t$ , recursively applied, gives the translation of $T_B$-terms to DA-terms. Now unification of two terms s,t in $T_B$ is equivalent to DA-unifying the terms $\varphi s, \varphi t$, since instantiations into $z_B$ are irrelevant. ■

Next we consider an example for a sort-preserving and congruence-closed equational theory which is of type infinitary, but if we add sorts, the theory becomes nullary unifying.

**3.11 Example.** There exists a sort-preserving, congruence-closed equational theory $\mathcal{E}$ of type nullary such that the unsorted theory is of type infinitary:

Consider the following (unsorted) term rewriting system:

$f_1(g_1(x)) \to g_2(f_1(x), f_2(g_1(x)) \to g_2(f_2(x)$

$f_1(k(x)) \to f_2(k(x)$

$g_1(k(h(l(x)))) \to k(h(x))$

$g_2(f_2(k(h(l(x))))) \to f_2(k(h(x)))$

This term rewriting system is canonical and the generated theory is of unification type infinitary as proved in Appendix A.2.

Now we introduce sorts:

$$\begin{aligned} \text{Let } \Sigma = \{\ A \sqsupset B,\quad & f_1 : A \to A, B \to B, \\ & f_2 : A \to A, B \to B, \\ & g_2 : B \to B, \\ & g_1 : B \to B, \\ & k: A \to A,\ h: A \to A,\ l: A \to A, \\ & k(h(x_A)) : B\} \end{aligned}$$

The sorted term rewriting rules are as follows:

$f_1(g_1(x_B)) \to g_2(f_1(x_B), f_2(g_1(x_B)) \to g_2(f_2(x_B)$

$f_1(k(x_A)) \to f_2(k(x_A))$

$g_1(k(h(l(x_A)))) \to k(h(x_A))$

$g_2(f_2(k(h(l(x_A))))) \to f_2(k(h(x_A)))$.

We have to show that this term rewriting system is canonical, and that the equational theory is sort-preserving and congruence-closed.

1) $R_\Sigma$ is canonical and sort-decreasing:

Termination follows as in the unsorted case (cf. Appendix A.2).

The critical pairs are the same as in the unsorted case and are all confluent.

In order to use the criterion in Theorem II.3.5, we have to consider the critical sort relations. The only nontrivial critical sort-relation is $f_2(k(h(x_B)))$: B, which comes from unifying $f_1$:B $\to$ B with the rule $f_1(k(x_A)) \to f_2(k(x_A))$. This critical sort relation is satisfied.

2) $\mathcal{E}$ is sort-preserving:

In order to use the criterion of Corollary 3.7 we have to consider the critical sort-relation for the reversed rewrite rules. Again the only nontrivial critical sort-relation is related to rule 3, and is $f_1(k(h(x_B)))$: B, which is satisfied.

3) $\mathcal{E}$ is congruence-closed:

Use Criterion II.6.2 i). It is sufficient to consider the rewrite rules $f_1(g_1(x_B)) \to g_2(f_1(x_B)$, $f_2(g_1(x_B)) \to g_2(f_2(x_B)$. If a term of sort A is instantiated for $x_B$, then in every case both terms are ill-sorted.

4) $\mathcal{E}$ is of unification type 0:

Consider the unification problem $\langle f_1(x_A) = f_2(x_A) \rangle$.

A complete set is $\{\{x_A \leftarrow g_1{}^i(k(h(y_A)))\} \mid i = 0,1,2,\ldots\} \cup \{x_A \leftarrow k(y_A)\}$, which is easily proved by induction.

However, this set has no basis since $g_1{}^i(k(h(l(y_A))) =_E g_1{}^{i-1}(k(h(y_A))$. ■

The following table summarizes some results given in this paragraph. The column "unsorted" contains the assumptions and the column "elementary" and "not elementary" the conclusion.

| unsorted | elementary | not elementary | Reference |
|---|---|---|---|
| $\mathcal{E} \in \mathcal{M}_{1\omega,\mathrm{eff}}$ and $\lvert \mu U_{\bar{\Sigma},E}(\Gamma) \rvert < \infty$ | $\mu U_{\Sigma,E} \neq \emptyset$ decidable | $\mu U_{\Sigma,E} \neq \emptyset$ not decidable | 3.4, 3.6, III.3.5 III.6.1 |
| $\lvert \mu U_{\bar{\Sigma},E}(\Gamma) \rvert = \infty$, $\mu U_{\bar{\Sigma},E}(\Gamma) \neq \emptyset$ decidable | $\mu U_{\Sigma,E} \neq \emptyset$ not decidable | $\mu U_{\Sigma,E} \neq \emptyset$ not decidable | 3.10 |
| $\mu U_{\bar{\Sigma},E}(\Gamma)$ exists | $\mu U_{\Sigma,E}(\Gamma)$ exists | does not exist | 3.6, 3.11 |
| $\mu U_{\bar{\Sigma},E}(\Gamma)$ rec. enumerable and $\mathcal{E} \in \mathcal{M}_{1\omega,\mathrm{eff}}$ | $\mu U_{\Sigma,E}(\Gamma)$ rec. enumerable | ? | 3.4, 3.8 |

## 4. Example: Unification in Sets, Multisets, Semigroups and Groups.

The following example demonstrates how to compute complete and minimal sets of unifiers using Theorem 3.3. It shows also that the minimization procedure of Theorem 3.4 becomes incorrect if the signature is not elementary.

### 4.1 Example: Unification of Sets.

Unification of sets (where sets are not allowed as elements of sets) [LS76, Bü86b] can be modelled as unification of terms built from variables, constants and a binary ACI-function symbol f, i.e. f is associative, commutative and idempotent. For convenience we denote terms as sets of the occurring variables and constants. For example f(a x) is denoted as {a ,x}. Furthermore {a, a} = {a} . It follows from the axioms that it is allowed to remove duplicates.

It is well-known that ACI-unification is of type finitary and that ACI-matching is also finitary [LS76, Bü86b].

In unsorted ACI-unification problems it is not possible to make a difference between element variables and set-variables. However, there are applications, where sets have to be unified and some variables denote elements and not sets. Such a unification problem can easily be described using the following sort structure, modelling sets and singletons:

Let $\Sigma := \{\ \mathrm{SET} \sqsupseteq \mathrm{SING},\ f{:}\mathrm{SET}\mathrm{x}\mathrm{SET} \rightarrow \mathrm{SET},\ a_i{:}\mathrm{SING},\ i=1,\ldots,n\}$

$$
\begin{aligned}
\mathrm{ACI} := \{\quad & f(f(x_{SET}, y_{SET}), z_{SET}) = f(x_{SET}, f(y_{SET}, z_{SET})),\\
& f(x_{SET}, y_{SET}) = f(y_{SET}, x_{SET}),\\
& f(x_{SET}, x_{SET}) = x_{SET}\}
\end{aligned}
$$

In this elementary signature all terms are well-sorted and the equational theory is congruence-closed. However, the equational theory is not sort-preserving, since $f(x_{SING}, x_{SING}) =_{\Sigma,E} x_{SING}$ but $\mathbf{LS}_\Sigma(f(x_{SING}, x_{SING})) = \mathrm{SET}$ and $\mathbf{LS}_\Sigma(x_{SING}) = \mathrm{SING}$. We can use Theorem II.9.1 to obtain a sort-assignment SORT such that the equational theory is sort-preserving:

$$
\mathrm{SORT}(t) := \begin{cases} \mathrm{SING}, & \text{if t can be reduced to a variable or constant of sort SING.} \\ \mathbf{LS}_\Sigma(t), & \text{otherwise.} \end{cases}
$$

From now on we use SORT as sort-assignment, which is not elementary.
Theorem II.9.1 yields that the equational theory with respect to SORT is sort-preserving.
An interesting observation is that $\mu W_\Sigma(\{x \leftarrow t\})$ is always a singleton and that all terms in the

codomain of substitutions in $\mu W_\Sigma(\{x \leftarrow t\})$ are variables or constants:

We have the cases:

i) x is of sort SET, then $\mu W_\Sigma(\{x \leftarrow t\}) = \{Id\}$

ii) x is of sort SING, t contains more than one constant, then $\mu W_\Sigma(\{x \leftarrow t\}) = \emptyset$

iii) x is of sort SING, t contains exactly one constant a,

then $\mu W_\Sigma(\{x \leftarrow t\}) = \{\mu\}$, where $\mu$ maps all variables in V(t) to a.

iv) x is of sort SING, t contains only variables,

then $\mu W_\Sigma(\{x \leftarrow t\}) = \{\mu\}$, where $\mu$ maps all variables in t to a new variable of sort SING.

Theorem 3.3 now yields a finite set $cU_{\Sigma,ACI}(s,t)$ for arbitrary terms s,t.
Since ACI-matching is decidable, we can remove instances and obtain $\mu U_{\Sigma,ACI}(s,t)$ as a subset of $cU_{\Sigma,ACI}(s,t)$. Hence $\mu U_{\Sigma,ACI}(s,t)$ exists and is finite for all terms s and t.

The following example which is taken from [Bü86b] demonstrates how to compute the set $\mu U_{\Sigma,ACI}(s,t)$. As constants we use a,b,c,d.
Let $s = \{x_{SET}, y_{SING}, d\}$ and $t = \{a, b, w_{SET}\}$
The set of most general unifiers for the unsorted problem consists of 9 unifiers (where u and v are of sort SET):

$$
\begin{array}{lllll}
\sigma_1 & = & \{x \leftarrow u, & y \leftarrow \{v,a,b\}, & w \leftarrow \{u,v,d\}\ \} \\
\sigma_2 & = & \{x \leftarrow \{u,a\}, & y \leftarrow \{v,b\}, & w \leftarrow \{u,v,d\}\} \\
\sigma_3 & = & \{x \leftarrow \{u,a\}, & y \leftarrow \{v,a,b\}, & w \leftarrow \{u,v,d\}\} \\
\sigma_4 & = & \{x \leftarrow \{u,b\}, & y \leftarrow \{v,a\}, & w \leftarrow \{u,v,d\}\} \\
\sigma_5 & = & \{x \leftarrow \{u,b\}, & y \leftarrow \{v,a,b\}, & w \leftarrow \{u,v,d\}\} \\
\sigma_6 & = & \{x \leftarrow \{u,a,b\}, & y \leftarrow v & w \leftarrow \{u,v,d\}\} \\
\sigma_7 & = & \{x \leftarrow \{u,a,b\}, & y \leftarrow \{v,a\} & w \leftarrow \{u,v,d\}\} \\
\sigma_8 & = & \{x \leftarrow \{u,a,b\}, & y \leftarrow \{v,b\} & w \leftarrow \{u,v,d\}\} \\
\sigma_9 & = & \{x \leftarrow \{u,a,b\}, & y \leftarrow \{v,a,b\} & w \leftarrow \{u,v,d\}\}.
\end{array}
$$

The next computation step is to examine each of these unifiers and instantiate them into well-sorted ones. This is not possible for the unifiers $\sigma_1$, $\sigma_3$, $\sigma_5$ and $\sigma_9$, since $y_{SING}$ is of sort SING. We obtain a complete set $cU_{\Sigma,ACI}(s,t)$ consisting of 5 unifiers:

$$
\begin{array}{llllll}
\sigma_2' & = \{x \leftarrow \{u,a\}, & y \leftarrow b, & w \leftarrow \{u,b,d\}\} & \\
\sigma_4' & = \{x \leftarrow \{u,b\}, & y \leftarrow a, & w \leftarrow \{u,a,d\}\} & \\
\sigma_6' & = \{x \leftarrow \{u,a,b\}, & y \leftarrow v', & w \leftarrow \{u,v',d\}\} & \text{(v' new of sort SING)} \\
\sigma_7' & = \{x \leftarrow \{u,a,b\}, & y \leftarrow a, & w \leftarrow \{u,a,d\}\} & \\
\sigma_8' & = \{x \leftarrow \{u,a,b\}, & y \leftarrow b, & w \leftarrow \{u,b,d\}\} &
\end{array}
$$

Now we can remove $\sigma_7'$ and $\sigma_8'$, since they are instances of the unifier $\sigma_6'$. Note that this subsumption is only possible in the case of nonelementary signatures. The set $\mu U_{\Sigma,ACI}(s,t)$ consists of the three unifiers $\sigma_2'$, $\sigma_4'$, and $\sigma_6'$. It may be the case that a modification of the unification algorithm in [Bü86b] yields a more efficient unification algorithm for the theory ACI together with the sort structure described above.

If a sort structure is used, where SING has subsorts and these subsorts form a semi-lattice, then there are only minor changes to the above procedure. The set $\mu W_{\Sigma,ACI}(\{x \leftarrow t\})$ is a singleton in this case, too.

## 4.2. Example: Unification of Multisets.

The structure multisets [LS76, St81, Fa84, Fo85, HS87, Bü86a] is equivalent to the structure of AC-terms, generated by an associative and commutative function symbol, variables and constants.

The theory of AC-terms is regular and finite, furthermore it is well-known that most general unifier sets are finite [LS76]. Hence for every sort-structure that makes the congruence congruence-closed and sort-preserving, the set of most general unifiers $\mu U_{\Sigma,AC}(s,t)$ exists for every term pair s,t.

The sort-structure on multisets, however, is very often a regular, elementary signature.

In this case,we have:

i) $\mu W_{\Sigma}(\{x \leftarrow t\})$ is finite and consists of renamings.

ii) $\mu U_{\Sigma,AC}(s,t)$ exists and is finite

According to II.6.20, in a regular elementary signature the sorts form a commutative monoid that is compatible with the subsort-structure.

In order to give a more concrete example, we consider the addition (+) on integers (INT) considered as a associative and commutative function and let O and E be subsorts of the integers denoting odd and even integers.

That is $\Sigma = \{INT \sqsupset O, INT \sqsupset E, +: INT \times INT \to INT, O \times O \to E, O \times E \to O, E \times O \to O,$
$E \times E \to E\}$ and

$$E = \{ x_{INT} + (y_{INT} + z_{INT}) = (x_{INT} + y_{INT}) + z_{INT} , (x_{INT} + y_{INT}) = (y_{INT} + x_{INT}) \}$$

The theory is congruence-closed and sort-preserving:

Congruence-closedness follows from Proposition II.6.2.

In order to show sort-preservation, we use Corollary II.3.7. This is equivalent to checking equations like $O + (O + E) = (O + O) + E$ and $(O + E) = (E + O)$, which are all satisfied.

Consider the unification problem $< x_O + y_O = x'_E + y'_E >$. First we ignore the sorts and compute a minimal set of solutions:

1) $\{x_O \leftarrow z_1, y_O \leftarrow z_2, x'_E \leftarrow z_1, y'_E \leftarrow z_2\}$
2) $\{x_O \leftarrow z_1, y_O \leftarrow z_2, x'_E \leftarrow z_2, y'_E \leftarrow z_1\}$
3) $\{x_O \leftarrow z_2, y_O \leftarrow z_3 + z_4, x'_E \leftarrow z_3, y'_E \leftarrow z_2 + z_4\}$
4) $\{x_O \leftarrow z_1, y_O \leftarrow z_3 + z_4, x'_E \leftarrow z_1 + z_3, y'_E \leftarrow z_4\}$
5) $\{x_O \leftarrow z_1 + z_2, y_O \leftarrow z_4, x'_E \leftarrow z_1, y'_E \leftarrow z_2 + z_4\}$
6) $\{x_O \leftarrow z_1 + z_2, y_O \leftarrow z_3, x'_E \leftarrow z_1 + z_3, y'_E \leftarrow z_2\}$
7) $\{x_O \leftarrow z_1 + z_2, y_O \leftarrow z_3 + z_4, x'_E \leftarrow z_1 + z_3, y'_E \leftarrow z_2 + z_4\}$

Next we try to make the substitutions well-sorted and assume that all introduced variables $z_i$ are of sort INT.

1) and 2) have no well-sorted instance. The unifiers 3),4),5),6) have exactly one well-sorted instance and the unifer 7) has two well-sorted instances. In weakening the unifier 7), we have to solve problems of the form $x_O = z_1 + z_2$ which have two solutions, namely the sum of two even numbers is even and the sum of two odd numbers is even.

3') $\{x_O \leftarrow z_{2,O}, y_O \leftarrow z_{3,E} + z_{4,O}, x'_E \leftarrow z_{3,E}, y'_E \leftarrow z_{2,O} + z_{4,O}\}$
4') $\{x_O \leftarrow z_{1,O}, y_O \leftarrow z_{3,O} + z_{4,E}, x'_E \leftarrow z_{1,O} + z_{3,O}, y'_E \leftarrow z_{4,E}\}$
5') $\{x_O \leftarrow z_{1,E} + z_{2,O}, y_O \leftarrow z_{4,O}, x'_E \leftarrow z_{1,E}, y'_E \leftarrow z_{2,O} + z_{4,O}\}$
6') $\{x_O \leftarrow z_{1,O} + z_{2,E}, y_O \leftarrow z_{3,O}, x'_E \leftarrow z_{1,O} + z_{3,O}, y'_E \leftarrow z_{2,E}\}$
$7_1$) $\{x_O \leftarrow z_{1,O} + z_{2,E}, y_O \leftarrow z_{3,O} + z_{4,E}, x'_E \leftarrow z_{1,O} + z_{3,O}, y'_E \leftarrow z_{2,E} + z_{4,E}\}$
$7_2$) $\{x_O \leftarrow z_{1,E} + z_{2,O}, y_O \leftarrow z_{3,E} + z_{4,O}, x'_E \leftarrow z_{1,E} + z_{3,E}, y'_E \leftarrow z_{2,O} + z_{4,O}\}$

Due to Theorem 3.4 it is not possible that instances of different unsorted unifiers subsume each other. Furthermore $7_1$) and $7_2$) are independent. Hence the above set of unifiers is a minimal set of $\Sigma$-unifiers.

The same procedure applied for example to the unification problem $<x_O + y_O = x'_O + y'_E>$ yields no well-sorted solution.

**4.3 Example: Unification under Associativity.**

We give a similar example for associativity: Let + be the addition on integers (INT) considered as an associative function symbol and let O and E be subsorts of the integers denoting odd and even integers.

That is $\Sigma = \{\text{INT} \sqsupseteq \text{O}, \text{INT} \sqsupseteq \text{E}, +: \text{INT} \times \text{INT} \to \text{INT}, \text{O} \times \text{O} \to \text{E}, \text{O} \times \text{E} \to \text{O}, \text{E} \times \text{O} \to \text{O},$

$E \times E \rightarrow E\}$ and

$E = \{ x_{INT} + (y_{INT} + z_{INT}) = (x_{INT} + y_{INT}) + z_{INT}\}$

The theory E is congruence-closed and sort-preserving.
Consider the unification problem $\langle x_O + y_O = x'_E + y'_E\rangle$. First we ignore the sorts and compute a minimal set of solutions:

1) $\{x_O \leftarrow z_1 ;\ y_O \leftarrow z_2 ;\ x'_E \leftarrow z_1 ;\ y'_E \leftarrow z_2 \}$
2) $\{x_O \leftarrow z_3 + z_4 ;\ y_O \leftarrow z_5 ;\ x'_E \leftarrow z_3 ;\ y'_E \leftarrow z_4 + z_5 \}$
3) $\{x_O \leftarrow z_6 ;\ y_O \leftarrow z_7 + z_8 ;\ x'_E \leftarrow z_6 + z_7 ;\ y'_E \leftarrow z_8 \}$

Next we try to make the substitutions well-sorted resulting in:

2') $\{x_O \leftarrow z_{1,E} + z_{2,O} ;\ y_O \leftarrow z_{3,O} ;\ x'_E \leftarrow z_{1,E} ;\ y'_E \leftarrow z_{2,O} + z_{3,O} \}$
3') $\{x_O \leftarrow z_{4,O} ;\ y_O \leftarrow z_{5,O} + z_{6,E} ;\ x'_E \leftarrow z_{4,O} + z_{5,O} ;\ y'_E \leftarrow z_{6,E} \}$

The set consisting of the unifiers 2') and 3') is a complete and minimal set of unifiers for the problem $\langle x_O + y_O = x'_E + y'_E\rangle$.

The same procedure applied for example to the unification problem $\langle x_O + y_O = x'_O + y'_E\rangle$ yields no well-sorted solution.

### 4.4 Example. Unification in Groups.

In this example we show that a group defined by a sort-preserving congruence and by a regular, elementary and finite sort-structure can only have a many-sorted sort-structure. This shows also that for combining subsorts and equations in order to obtain a sort-preserving congruence, term declarations are indispensible.

We assume that the operations of the group are defined everywhere and that the signature is finite.
Due to II.6.20, the set of sorts forms a finite group with unit-sort E.

We show that E has no proper subsorts and supersorts:

Assume there is a sort A with $A \sqsubset E$ or $A \sqsupset E$. Since the group is finite, we have $A^n = E$ for some n. Furthermore by II.6.20, we have $A^n \sqsubset E$ or $A^n \sqsupset E$, respectively. This is impossible.

No subsort relation is possible:

Let $A \sqsubseteq B$. Then from $A^n = E$ we obtain $E = A*A^{n-1} \sqsubseteq B*A^{n-1}$. Hence $E = B*A^{n-1}$ and by cancellation we obtain $A = B$.

## 5. Narrowing.

Narrowing [Fa79, Hul80, SS81, Sz82] is a universal unification procedure that works for the class of equational theories that admit a canonical term rewriting system. The process of narrowing was extended to sorted signatures in [SNMG87], however, not for signatures with term declarations. As a further difference we will allow a more general kind of rewriting relation, namely weakly sort-decreasing instead of sort-decreasing relations.

In this paragraph we intend to give arguments that narrowing behaves as usual in the context of term declarations. We are not interested in the details of different narrowing techniques.

Assume given an equational theory $\mathcal{E}$ together with a canonical term rewriting system R and a unification problem $\langle s = t\rangle$. Narrowing performs unification by nondeterminstic successive steps, where one step searches for most general instances of s and t such that a rewrite rule becomes applicable. A typical narrowing step is performed as follows: Let $\pi$ be a nonvariable position in s and let $\sigma$ be a most general (Robinson-) unifier of $s\backslash\pi$ and l, where $l \to r$ is a rule in R. Then reursively try to unify the modified problem $\langle \sigma s[\pi \leftarrow \sigma r] = \sigma t\rangle$ keeping the substitution $\sigma$ in mind.
A key to narrowing is a lemma in [Hul80], we include a proof of a part of it in order to check whether it holds in a signature with term declarations.

**5.1 Lemma.** Let s be a term and let $\sigma$ be a normalized substitution.
Furthermore let $\sigma s$ be reduced to $t_1$ in a one-step reduction at position $\pi$ with the rule $l \to r$. Then narrowing of s at position $\pi$ with rule $l \to r$ produces a term $s_1$ that is more general than $t_1$.

**Proof.** Since $\sigma$ is normalized, $\pi$ is a nonvariable occurrence of s.
Let $t_1 = \sigma s[\pi \leftarrow \theta r]$, where $\theta l = \sigma s\backslash\pi$. Since we can assume that l and s are variable disjoint, we have that l and $s\backslash\pi$ are unifiable.
Let $\mu$ be a most general unifier of l and $s\backslash\pi$ such that $\mu \le \theta \cup \sigma\ [\mathbf{V}(l,s\backslash\pi)]$. Narrowing yields the term $s_1 := \mu s[\pi \leftarrow \mu r]$. There exists a substitution $\lambda$ such that $\lambda\mu = \theta \cup \sigma\ [\mathbf{V}(l,s]$ .
Now we have $\lambda s_1 = \lambda\mu s[\pi \leftarrow \lambda\mu r] = \sigma s[\pi \leftarrow \theta r] = t_1$. ■

This lemma allows to derive by induction on the length of a reduction that narrowing is a complete unification procedure for normalized unifiers. Now, since R is weakly sort-decreasing, there exists for every unifier also a normalized unifier, hence narrowing is complete for all substitutions. In contrast to usual narrowing sorted narrowing introduces a further nondeterminism. For a fixed position and rewrite rule, there may be an infinite number of possible narrowing steps, since the number of most general unifiers may be infinite.

There are several improvements of the first approach to narrowing, such as basic narrowing [Hul80, Re87, NRS87] and some other techniques. Presumably all these improvements are also applicable in the general case of sorted signatures considered in this thesis.

# Part V
# Sorted Resolution-Based Calculi

**Overview:** In this part we consider several resolution-based calculi with order-sorted signatures. We investigate resolution, paramodulation and factoring, G. Plotkin's resolution with built-in equational theories, J. Morris' E-resolution and M. Stickels theory resolution. We show that the completeness results that hold in the unsorted case or in the case of simple signatures [Wa83] hold also in the presence of term delarations. The results concerning the fact that the functionally reflexive axioms are not needed for clause sets with equations are in general not liftable, as shown in an example.

In this part we assume that there are no ill-sorted terms and literals. Furthermore, we sometimes omit the adjective 'well-sorted', but always mean that every thing is well-sorted, in particular substitutions are well-sorted.

## 1. Resolution, Paramodulation and Factoring.

**1.1 Definition.** [Ro65, RW69, CL73].

i) Resolution of two clauses is defined as:
Let $C_1 = \{P(s_1,\ldots,s_n)\} \cup C_{1,R}$ and $C_2 = \{-P(t_1,\ldots,t_n)\} \cup C_{2,R}$. We assume that $C_1$ and $C_2$ are variable-disjoint. Let $\sigma \in \mu U_\Sigma(s_1=t_1,\ldots,s_n=t_n)$. Then the clause $\sigma C_{1,R} \cup \sigma C_{2,R}$ is a **resolvent** of $C_1$ and $C_2$.

ii) Factoring is defined as follows:
Let $C = \{P(s_1,\ldots,s_n), P(t_1,\ldots,t_n)\} \cup C_R$ be a clause. Let $\sigma \in \mu U_\Sigma(s_1=t_1,\ldots,s_n=t_n)$. Then the clause $\{\sigma P(s_1,\ldots,s_n)\} \cup \sigma C_R$ is a **factor** of C.

iii) Paramodulation is defined as follows:
Let $C_1 = \{s = t\} \cup C_{1,R}$ and $C_2 = \{L\} \cup C_{2,R}$ be two variable-disjoint clauses. Let $\pi$ be an occurrence in L and let $\sigma \in \mu U_\Sigma(s, L/\pi)$. Then the clause $\{(\sigma L)[\pi \leftarrow \sigma t]\} \cup \sigma C_{1,R} \cup \sigma C_{2,R}$ is a **paramodulant.** ■

In general it may be the case that the clause obtained by paramodulation is not well-sorted. Ch. Walther [Wa83] gives examples and his calculus must be aware of this problem, but we make assumptions to avoid this:

**1.2 General Assumption.** If equations are in the clause set, we assume in the following that a greatest sort TOP is available, such that all literals and terms are well-sorted, and also that the unit equation $x_{TOP} = x_{TOP}$ is in every clause set. ■

In II.1 we have justified these assumptions and shown that the addition of a sort IT is a conservative transformation.

**1.3 Lemma.** Resolution, factoring and paramodulation are sound deduction rules. ■

Proposition I.13.7 implies that successively factoring a clause yields the same factors (up to renaming) as a generalized factoring step, where more than two literals are considered.

In order to prove the completeness of our sorted calculi, we use Herbrand's Theorem II.11.2 and the usual lifting arguments. These arguments work as follows: Given an unsatisfiable clause set, the Herbrand Theorem yields a finite unsatisfiable clause set consisting of ground instances of the original clauses. Then a ground refutation (a deduction of the empty clause) is lifted to a refutation in the original clause set.

Lifting a deduction step is defined as follows: Given n clauses $C_1,\ldots,C_n$, and n ground instances $C_{1,gr},\ldots,C_{n,gr}$, such that $\gamma_i C_i = C_{i,gr}$ and $\gamma_i$ is a 1-1 mapping on literals. If a clause $D_{gr}$ is deducable from $C_{1,gr},\ldots,C_{2,gr}$, then we say this deduction is **liftable**, iff we can deduce a clause D from $C_1,\ldots,C_n$, such that $D_{gr}$ is a ground instance of D and D and $D_{gr}$ have the same number of literals.

Factoring is needed to lift the merge operation, which is implictely done by set union: One literal on the ground level may be an instance of several literals on the general level. In order to obtain a 1-1 mapping between the general and the ground clauses, one may need some factoring steps on the general level.

For factoring we have: (cf. [WR73, CL73] )

**1.4 Lemma.** For every ground instance $C_{gr}$ of a clause C there exists a clause C' derivable by factoring, such that $C_{gr}$ is a ground instance of C' and both C' and $C_{gr}$ have the same number of literals.

**Proof.** Straightforward using Proposition I.13.7. ■

**1.5 Lemma.** [Ro65, WR73, CL73] A ground resolution step is liftable to a resolution step and factoring steps.

Suppose a sort TOP is in $\Sigma$, then we mean by the **functionally reflexive axioms** all axioms of the form $f(x_{1,TOP},\ldots,x_{n,TOP}) = f(x_{1,TOP},\ldots,x_{n,TOP})$.

**1.6 Lemma.** [RW69,WR73] If the Assumptions 1.2 hold, then a ground paramodulation step is liftable to one paramodulation step between clauses, some factoring steps and some paramodulation steps with the functionally reflexive axioms.

## 2. Deductions on Ground Clauses.

In this paragraph we consider deduction systems on ground clause sets and show how a refutation can be found. This is some preliminary work for the completeness results for sorted calculi. Furthermore we demonstrate how to use the completion procedure in II.3 to describe a decision procedure for sets of ground clauses.

The next proposition is well-known. Nevertheless, we will provide a (simple) proof for it.

**2.1 Proposition.** An unsatisfiable ground clause set CS without equations is refutable with resolution. Furthermore unsatisfiability is decidable.

**Proof.** We show this by induction on the k-parameter (or the excess literal number) [AB70]: $k(CS) := \Sigma\{(|C| - 1) \mid C \in CS\}$, where $|C|$ is the number of literals in the clause C.

Let CS be an unsatisfiable ground clause set.

If $k(CS) = 0$, then there are two complementary literals and hence a resolution on those literals yields the empty clause.

If $k(CS) > 0$, then there exists a non-unit clause C. We partition C into two parts $C_1$ and $C_2$ and obtain two unsatisfiable clause sets $CS_1$ and $CS_2$ by replacing C by $C_1$ or $C_2$ respectively. Since $k(CS_i) < k(CS)$, there are refutations of $CS_1$ and $CS_2$ by resolution. These two resolutions proofs can be combined to a resolution proof of the empty clause in CS, since all clauses are ground. ❑

Resolution provides a decision procedure, since only a finite number of clauses is derivable from CS. ■

In the rest of this paragraph we consider ground clause sets with equality and we will prove a similar result for them.

Note that we use the total ordering $<_s$ on ground terms given in II.3. We shall use the E-resolution technique [Mo69] in order to prove results on a calculus using paramodulation.

In the following we will assume that enough units $t = t$ are in ground clause sets rather than to use reflexivity implicitely. More precisely, we assume that every set of ground clauses $CS_{gr}$ that contains equality literals, also contains all the (finitely many) unit equations $\{t = t\}$, where $t \leq_s s$ for some term s in $CS_{gr}$.

This assumption avoids the need for two different ground E-resolution rules.

**2.2 Proposition.** An E-unsatisfiable set CS of ground unit clauses can be refuted by resolution and paramodulation. Furthermore it is decidable whether it is E-unsatisfiable.

**Proof.** Note that enough ground units of the form $t = t$ are available. The set of ground unit equations can be completed as in II.3.15. The resulting TRS is then used to normalize all other terms. Note that completion and normalization can be simulated by paramodulation steps. The clause set is E-unsatisfiable, iff there are two complementary literals. This includes the case of an inequation $s \neq s$, where s is in normalform, since then an equation $s = s$ is in the clause set. The complementary literals can be resolved and hence the empty clause can be obtained. ■

We give deduction rules for ground equations, which are essentially E-resolution rules.

**2.3 Definition.** A **ground E-resolution step** is defined as follows:

Take n clauses $C_1,\ldots,C_n$ from the set CS of ground clauses (some clauses $C_i$ may be identical) and from every clause a literal $L_i$, such that $L_1$ and $L_2$ have the same predicate, but a different sign and $L_3,\ldots,L_n$ are equations. If $L_1,\ldots,L_n$ are contradictory (with 2.2), then infer the clause $(C_1 - \{L_1\}) \cup \ldots \cup (C_n - \{L_n\})$.

A ground E-resolution step is **minimal**, if all equations $L_3,\ldots,L_n$ are necessary in order to make $L_1$ and $L_2$ complementary. ■

**2.4 Lemma.** A minimal ground E-resolution step can be simulated by paramodulation and resolution.

**Proof.** The procedure in 2.2 (completion and normalization) performed by paramodulation gives exactly the desired clause. ■

**2.5 Proposition.** An E-unsatisfiable ground clause set CS can be refuted by (minimal) E-resolution steps (and hence by resolution and paramodulation). Furthermore E-unsatisfiability of a set CS of ground clauses is decidable with this procedure.

**Proof.** We prove this by induction on the k-parameter

Base case. If $k(CS) = 0$, then CS consists of unit clauses and we can make a minimal E-resolution step due to Proposition 2.2.

Induction step. If $k(CS)\} > 0$, then there exists a clause C with $|C| > 1$. We partition C into $C_1 \cup C_2$ and consider two different clause sets $CS_1$ and $CS_2$, where in $CS_i$ the clause C is replaced by $C_i$. Obviously, if CS is unsatisfiable, then $CS_1$ and $CS_2$ are both unsatisfiable. We have $k(CS_i) < k(CS)$. By induction, the clause sets $CS_1$ and $CS_2$ both have a refutation using minimal ground E-resolution. Since C is ground, we can combine the two refutations and obtain a refutation by minimal E-resolution of CS. ❑

To recognize satisfiability, we perform ground E-resolution steps until no new clause can be derived. Since the number of derivable clauses is finite, this procedure terminates. ■

## 3. Completeness of Sorted Calculi Based on Resolution, Paramodulation and Factoring.

We give extensions of the theorems of [Ro65, RW69] to the sorted case. These results extend also the theorems in [Wa83].

**3.1 Theorem.** Every unsatisfiable clause set CS without equations can be refuted by resolution and factoring.

**Proof.** By the Herbrand theorem II.11.2 there exists a finite unsatisfiable clause set $CS_{gr}$ of ground instances of clauses from CS. This clause set can be refuted by resolution and factoring due to Proposition 2.2. Lemma 1.5 shows that this ground resolution is liftable to the general case. ■

**3.2 Theorem.** Let the assumptions in 1.2 be satisfied. Every unsatisfiable clause set CS with equations can be refuted by paramodulation, resolution and factoring, provided the functionally reflexive axioms are in CS.

**Proof.** By the Herbarand theorem II.11.2 there exists a finite unsatisfiable set $CS_{gr}$ of ground instances of clauses from CS. This clause set can be refuted by paramodulation and resolution due to Proposition 2.5. Lemma 1.5 and Lemma 1.6 show that these ground steps are liftable to resolution, paramodulation and factoring steps at the general level, provided the functionally reflexive axioms are in the clause set, hence there exists a deduction of the empty clause. ■

There are several proofs of the fact that for the unsorted case the functionally reflexive axioms are not needed in Theorem 3.2 (cf. [Bra75, Ri78, Pe83, HR86]). These proofs are rather complicated and either need involved arguments on semantic trees [Pe83] or they need arguments based on sequent calculus [Ri78]. Furthermore, in [Pe83] it is shown that paramodulation into variables is also not necessary in the unsorted case.

In the following examples we show that in the case of simple signatures paramodulation into variables is necessary and that for more general signatures, functionally reflexive axioms will be needed. We conjecture, that the functionally reflexive axioms are not needed for simple signatures.

**3.3 Example.**

i) If the signature is simple, then paramodulation into variables may be necessary:
Let $\Sigma := \{A, B \sqsubseteq TOP, a{:}A, b{:}B\}$ and let $CS := \{\{P(x_A)\}, \{\neg P(y_B)\}, \{a=b\}\}$.
This clause set is unsatisfiable, but can be refuted only by paramodulating into one of the variables $x_A$ or $y_B$.

ii) If the functionally reflexive axioms are not in the clause set and the signature is not simple, then resolution, paramodulation and factoring are not sufficient to deduce the empty clause for every unsatisfiable clause set:
Let $\Sigma := \{TOP \sqsupseteq A, B, C, D,\ a{:}A,\ b{:}B,\ c{:}C,\ d{:}D,\ g{:}TOP \to TOP,\ g{:}A \to C,\ g{:}B \to D\}$ and let $CS := \{\{z_{TOP} = z_{TOP}\}\{x_C \neq y_D\}, \{a=b\}\}$. This clause set is unsatisfiable, since $g(a) = g(b)$ holds in every model and $g(a)$ is of sort C and $g(b)$ is of sort D.
It is easy to see, that paramodulation and resolution are not sufficient to deduce the empty clause, since the symbol g cannot be introduced. ■

## 4. Resolution with Equational Theories.

If the clause set can be divided into two parts, a set of clauses CS without equations and a set of unit equations E, then we can use the method of [Plo72] to build them into an E-unification procedure (cf. part IV). The following theorem holds, which is due to G. Plotkin [Plo72] for unsorted equational theories:

**4.1 Theorem.** [Plo72] Let $\mathcal{E}$ be an equational theory and let CS be a clause set without equational literals. Suppose we have a complete E-unification procedure.
Then resolution and factoring with E-unification is a complete deduction system. ■

We can extend Proposition 2.2:

**4.2 Proposition.** If the theory $\mathcal{E}$ has a unification algorithm, i.e., for every set $\Gamma$ a finite complete and minimal set of unifiers is effectively computable, then for every set of ground clauses $CS_{gr}$ it is decidable , whether $CS_{gr}$ is contradictory under $\mathcal{E}$. ■

The extension of this calculus to clause sets that contain equations raises the problem that paramodulation has to take into account all potentially available subterms of some term, not only the syntactically given ones. In [Plo72] a modified unification procedure is proposed. In completing term rewriting systems modulo an equational theory this problem leads to the definition of a new condition, called coherence condition [JK84].

## 5. Morris` E-resolution.

E-resolution was first introduced by J. Morris in [Mo69]. R. Anderson [An70] defined it in terms of paramodulation and showed that E-resolution is a complete refutation procedure. The essential idea in the definition of E-resolution is to paramodulate only if complementary literals are generated such that a resolution step can be performed. For example, from the three clauses {P(a)}, {a=b,Q } and {-P(b)} an E-resolution step can deduce Q at once. The problem to find such literals that can be made complementary is very similar to unification with respect to an equational theory. However, in general the equations are conditional, hence unification should not only generate a set of unifiers, but also for every unifier $\sigma$ the instantiations which are used to prove that $\sigma$ is indeed a unifier. This would require that the unification procedure can copy equations. We do not consider this type of unification, but a more restricted one, which does not copy equations, and we assume that a higher level module provides the copies of clauses or equations.

Hence the unification problem is of the following type: Given a set of equations $E := \{l_i = r_i \mid i = 1,\dots,n\}$ and a set $\Gamma$ of unification problems $\langle s_i = t_i \mid i = 1,\dots,m\rangle$, find a substitution $\sigma$ such that the equations $\{\sigma l_i = \sigma r_i \mid i = 1,\dots,n\}$ imply the equations $\{\sigma s_i = \sigma t_i \mid i = 1,\dots,m\}$ where variables are considered as constants, i.e., by using the congruence closure method [NO80, Ga86,Koz76,Sh84] or completion and reduction as described in II.3. Note that $l_i,r_i,s_j,t_j$ may have variables in common. The problem is equivalent to the following: find a substitution $\sigma$, such that the universally quantified formula $\sigma(E \Rightarrow \Gamma)$ becomes a tautology. The same unification problem comes up in equational matings considered in [GRS87, And81], where this type of unification is called rigid E-unification.

The behaviour of equality in equational theories generated by ground equations was first considered by W. Ackermann [Ack54] who proved it to be a decidable problem. Recent investigations show that this problem has fast decision algorithms (even one of time-complexity O(m*log(m)) [NO79, NO80, DST80, Ga86,Sh84].

**5.1 Definition.** Let $E := \{l_i = r_i \mid i = 1,\dots,n\}$ be a set of equations and let $\Gamma := \{s_i = t_i \mid i = 1,\dots,m\}$ be a set of unification problems, then $\sigma$ is a **rigid E-unifier**, iff the implication $\sigma E \Rightarrow \sigma\Gamma$ holds in all interpretations. Equivalently $\sigma\Gamma$ is solved under the 'theory' $\sigma E$, where all variables in $\sigma E$ and $\sigma\Gamma$ are considered as constants. ■

A rigid E-unifier is an E-unifier, but the converse is not true (cf. [GRS87]). In [GRS87] there is also a discussion on the complexity and decidability of rigid E-unification, and it is shown, that rigid E-unification is NP-hard. Furthermore it is announced that rigid E-unification is decidable. An algorithm for rigid E-unification is also given in [GRS87], but there is no proof of completeness.

**5.2 Lemma.** The instance of a rigid E-unifier is also a rigid E-unifier:

**Proof.** If the formula $(\sigma E \Rightarrow \sigma\Gamma)$ holds, then it is true in all interpretations. Hence an instance $(\lambda\sigma E \Rightarrow \lambda\sigma\Gamma)$ is also true in all interpretations. This means $\lambda\sigma$ is also a rigid E-unifier of $\Gamma$. ■

We want to define complete sets of rigid E-unifiers and use them as substitutions in an E-resolution step. Unfortunately, this is troublesome as the following example shows: rigid-unify the terms x and y with respect to the equation $\{f(z\ a) = f(z\ b)\}$. Intuitively, the substitution $\sigma = \{x \leftarrow y\}$ should be the most general one. Now consider the substitution $\tau := \{x \leftarrow f(a\ b),\ y \leftarrow f(a\ a)\}$. Then $\tau$ is an E-instance of $\sigma$, if we use $\{f(z\ a) = f(z\ b)\}$ as an equational theory. However, the equation is used with the instantiation $\{z \leftarrow a\}$. This notion of instance would allow $\tau' := \{x \leftarrow f(f(a\ b)\ b),\ y \leftarrow f(\ f(a\ a)\ a)\}$ to be an instance of $\sigma$, which is not a rigid E-unifier, since then two instantiations of the equation $f(z\ a) = f(z\ b)$ are needed.

Thus we define the instance relation as follows (note that we define the subsumption different from [GRS87]) :

**5.3 Definition.** Let $E := \{l_i = r_i \mid i = 1,\ldots,n\}$ be a set of equations, let $\sigma$, $\tau$ be substitutions and let W be a set of variables.

Then we say $\tau$ is **rigid-equal** to $\sigma$ ( $\sigma =_{rig} \tau$ [E,W]) iff

i) $(\sigma E \Leftrightarrow \tau E)$ is valid.

ii) for all $x \in W$ : $(\tau E \Rightarrow \sigma x = \tau x)$ is valid.

Furthermore we say $\tau$ is a **rigid-instance** of $\sigma$ ($\sigma \leq_{rig} \tau$ [E,W]), iff there exists a substitution $\lambda$ such that $\lambda\sigma =_{rig} \tau$ [E,W].

We say two substitutions $\sigma$ and $\tau$ are **rigid-equivalent** ( $\tau \equiv_{rig} \sigma$ [E,W]) , iff $\tau \leq_{rig} \sigma$ [E,W] and $\sigma \leq_{rig} \tau$ [E, W] . ■

**5.4 Lemma.** Let $E := \{l_i = r_i \mid i = 1,\ldots,n\}$ be a set of equations and let $\Gamma := \{s_i = t_i \mid i = 1,\ldots,m\}$ be a set of unification problems. Let $W := \mathbf{V}(E,\Gamma)$

i) $\sigma =_{rig} \tau$ [E,W] implies $\lambda\sigma =_{rig} \lambda\tau$ [E,W] for substitutions $\lambda$.

ii) A rigid instance of a rigid E-unifier is a rigid E-unifier.

iii) $\leq_{rig}$ is a quasi-ordering.

iv) $\equiv_{rig}$ is an equivalence-relation.

**Proof.**

i) The statement $\sigma =_{rig} \tau$ [E,W] means that $(\sigma E \Leftrightarrow \tau E)$ is valid and that for all $x \in W$ $(\sigma E \Rightarrow \sigma x = \tau x)$ is valid. Both statements remain valid, if they are instantiated by $\lambda$.

ii) Let $\sigma$ be a rigid E-unifier of $\Gamma$ and let $\tau$ be a rigid-instance of $\sigma$ ($\sigma \leq_{rig} \tau$ [E,W]).
By definition we have that $(\lambda\sigma E \Leftrightarrow \tau E)$ is valid and that $(\tau E \Rightarrow \lambda\sigma x = \tau x)$ is valid for all

$x \in W$. Furthermore $\lambda\sigma$ is a rigid E-unifier of $\Gamma$ by Lemma 5.2. Together these facts imply that $\tau$ is a rigid E-unifier of $\Gamma$.

iii) Let $\sigma_1 \leq_{rig} \sigma_2 \leq_{rig} \sigma_3$ [E, W] for substitutions $\sigma_1,\sigma_2,\sigma_3$. Then there exist substitutions $\lambda_1$ and $\lambda_2$ such that $\lambda_1\sigma_1 =_{rig} \sigma_2$ [E, W] and $\lambda_2\sigma_2 =_{rig} \sigma_3$ [E,W]. By i) we obtain $\lambda_2\lambda_1\sigma_1 =_{rig} \lambda_2\sigma_2$ [E, W]. Furthermore, from the validity of $(\lambda_1\sigma_1 E \Leftrightarrow \sigma_2 E)$ and $(\lambda_2\sigma_2 E \Leftrightarrow \sigma_3 E)$ we obtain that $(\lambda_2\lambda_1\sigma_1 E \Leftrightarrow \lambda_2\sigma_2 E \Leftrightarrow \sigma_3 E)$ is valid. We conclude that $\lambda_2\lambda_1\sigma_1 =_{rig} \sigma_3$ [E, W] and hence $\sigma_1 \leq_{rig} \sigma_3$ [E, W].

iv) Follows from iii). ■

Similarily as for usual E-unification, we define complete and minimal sets of rigid E-unifiers, here with respect to the rigid-instance relation.

An interesting open problem is the existence of minimal sets of rigid E-unifiers. We conjecture, that a finite minimal complete and effectively computable set of rigid E-unifiers always exists.

As an example for rigid E-unification let $E := \{a = f(a)\}$ and $\Gamma := \{x = a\}$. The most general rigid E-unifier is $\sigma := \{x \leftarrow a\}$, since the theory is defined by ground equations. The set of all rigid E-unifiers is $\{ \{x \leftarrow f^n(a)\} \mid n \geq 0\}$, which is infinite.

Now we define E-resolution with respect to a procedure that computes a complete and perhaps minimal set of E-unifiers.

**5.5 Definition.** Let $A_{rid}$ be a procedure that computes complete sets of rigid E-unifiers. Then **E-resolution** is defined as follows:
Let $\{P(s_1,\ldots,s_m)\} \cup R_1$, $\{-P(t_1,\ldots,t_m)\} \cup R_2$, $\{l_3 = r_3\} \cup R_3 \ldots$, $\{l_n = r_n\} \cup R_n$ be n variable-disjoint clauses (where the clauses may be renamed copies), and let $\sigma$ be a unifier produced by $A_{rid}$ for $E = \{l_3 = r_3,\ldots,l_n = r_n\}$ and the problem $\Gamma := \langle s_1 = t_1,\ldots, s_m = t_m\rangle$. Then the E-resolvent is $\sigma(R_1 \cup \ldots \cup R_n)$. ■

**5.6 Lemma.** E-resolution is sound. ■

We propose to use E-resolution (together with a complete rigid E-unification algorithm) as a general inference rule together with factoring. We conjecture that this would provide a complete refutation procedure for arbitrary clause sets.

Let $A_{rid,pm}$ be the algorithm, that computes rigid-unifiers of $P(s_1,\ldots,s_n)$ and $\neg P(t_1,\ldots,t_n)$ and $l_1 = r_1,\ldots, l_m = r_m$ by paramodulating from $l_i = r_i$ into the two literals and that uses every equation at most once. The returned substitutions are the combined substitutions from the

paramodulation.
This algorithm is not a complete algorithm for rigid E-unification, however, it is sufficient for completeness of E-resolution:

**5.8 Theorem.** Let all functionally reflexive axioms be in the clause set.

Then E-resolution with the algorithm $A_{rid,pm}$ together with factoring is refutation-complete.

**Proof.** The theorem follows from Proposition 2.5, since in the presence of functionally reflexive axioms, all E-resolution steps are liftable due to Lemma 1.6. ■

An extension of the above algorithm $A_{rid,pm}$ is to make a paramodulation-like deduction and use equations more than once without copying them and without renaming them, but after a 'paramodulation' step, the corresponding substitution is applied to all involved clauses.

We conjecture, that in the unsorted case this extended algorithm is a complete rigid E-unification algorithm, even if paramodulation into variables is forbidden. For the case of simple signatures we conjecture, that full paramodulation provides a complete rigid E-unification algorithm. In polymorphic signatures Example 3.3 shows that functionally reflexive axioms are necessary.

The following example shows that the intuitive notion of most general E-resolvent has the same lifting problems as paramodulation:

**5.7 Example:**

Consider the three clauses $\{P(x, a), Q(x)\}$, $\{-P(y, b), R(y)\}$ $\{a=b\}$ and their respective ground instances $\{P(f(a), a), Q(f(a))\}$, $\{-P(f(b), b), R(f(b))\}$ $\{a=b\}$.

Then the most general E-resolvent should be $\{Q(x), R(x)\}$, whereas on the ground level, we obtain the E-resolvent $\{Q(f(a)), R(f(b))\}$. Obviously, this E-resolvent is not an instance of the general E-resolvent. ■

## 6. Theory Resolution.

In this paragraph we give an outline of how to extend the theory resolution (T-resolution) method of [St85] to order-sorted signatures. The idea of theory resolution is to exploit specialized algorithms for some theories such as taxonomic structures or partial orderings in the deductive machinery. For example in the theory ORD of partial orderings one can easily decide that $x < b \wedge b < c \wedge c < x$ is unsatisfiable without deducing further literals.
It has similarities with resolution using E-unification or with E-resolution, but there are differences as we will show in an example. Theory resolution with a taxonomic hierarchy as

theory is similar to but not the same as order-sorted deduction, since T-resolution may need more unifiers than order-sorted resolution (cf.[St85]).

For the convenience of the reader we repeat the definitions given in [St85]:
We assume that a signature $\Sigma$ is given.
Let T (the **theory**) be a satisfiable, finite set of first order axioms. Without loss of generality we can assume that T is a set of clauses. A set of ground literals C is **T-unsatisfiable**, iff $T \cup C$ is unsatisfiable, otherwise C is **T-satisfiable.**
A set I of ground literals is a **T-interpretation,** iff it is a model of T.
A set of ground literals LS is **minimal T-unsatisfiable**, iff LS is T-unsatisfiable and every proper subset LS' of LS is T-satisfiable.
A ground literal D is **valid** in a T-interpretation I, iff $D \in I$. A ground clause $C_{gr}$ is **valid** in I, iff $C_{gr} \cap I \neq \emptyset$. A clause C is **valid,** iff every ground instance of C is valid in I.
A T-interpretation I is a **T-model** of a clause set CS, iff every clause from CS is valid in I.
A clause set CS is **T-unsatisfiable,** iff it has no T-model.

**Ground T-Resolution**: Let $\{\{D_1\} \cup E_1,\ldots, \{D_k\} \cup E_k\}$ be a set of ground clauses ($D_i$ are literals). Let $\{D_1,\ldots,D_k\}$ be (minimally) T-unsatisfiable.
Then $\{E_1,\ldots,E_k\}$ is a **narrow total T-resolvent.**

M. Stickel [St85] defines different kinds of resolution, such as wide resolution instead of narrow resolution, or partial resolution instead of total resolution. We concentrate on narrow total T-resolution, since this is the straightforward extension of usual binary resolution.

**6.1 Lemma.** (Herbrand) A minimally T-unsatisfiable set LS of ground literals is finite.
**Proof.** Let LS be a T-unsatisfiable set of ground literals. Then $T \cup LS$ is unsatisfiable. By Herbrands Theorem (II.11.2) there exists a finite set of instances of $T \cup LS$ that is unsatisfiable. Hence there exists a finite subset LS' of LS such that $LS' \cup T$ is unsatisfiable. ■

**6.2 Definition.** A unification algorithm T-UNIFY for a theory T has as input a set of literals and generates a complete set of substitutions that make this set of literals contradictory. More precisely:
For every set of (variable-disjoint) literals $\{D_1,\ldots,D_n\}$, every ground substitution $\sigma_{gr}$ such that $\{\sigma_{gr}D_1,\ldots, \sigma_{gr}D_n\}$ is minimally T-contradictory and $\sigma_{gr}$: $\{D_1,\ldots,D_n\} \rightarrow \{\sigma_{gr}D_1,\ldots,\sigma_{gr}D_n\}$ is a bijection, there exists a substitution $\sigma \in$ T-UNIFY $(D_1,\ldots,D_n)$ such that $\sigma_{gr} \geq \sigma\ [V(D_1,\ldots,D_n)]$. Furthermore $\{\sigma D_1,\ldots,\sigma D_n\}$ should be T-unsatisfiable. In this case, we say T-UNIFY is a complete **T-unification algorithm** for the theory T.■

An algorithm will generate in general more substitutions, since the condition that a set of literals is minimally T-contradictory is hard to check.

**General (narrow total) T-Resolution**: Let $\{\{D_1\} \cup E_1,\ldots, \{D_k\} \cup E_k\}$ be a set of clauses (where clauses may be renamed copies) and let $\sigma$ be a substitution produced by a theory–unification algorithm.
Then $\{\sigma E_1,\ldots,\sigma E_k\}$ is a **narrow total T-resolvent** .

The T-calculus consists of T-resolution together with a T-unification algorithm and factoring.

**6.3 Lemma.** (Lifting )
Every ground T-resolvent is an instance of a clause deduced with T-resolution and factoring.

**Proof.** Let $\{D_i\} \cup E_i$, $i = 1,\ldots,n$ be the variable-disjoint general clauses and let $\sigma_{gr}$ be a ground substitution with $\sigma_{gr}(\{D_i\} \cup E_i) = \{D_{gr,i}\} \cup E_{gr,i}$ for $i = 1,\ldots,n$ . Furthermore let $\{D_{gr,1},\ldots,D_{gr,k}\}$ be minimally T-contradictory. The unification algorithm T-UNIFY yields a substitution $\sigma$ such that $\sigma_{gr} \leq \sigma\, [V(D_1,\ldots,D_n)]$. Let the corresponding resolvent be $\sigma E_1 \cup \ldots \cup \sigma E_k$. Obviously $\sigma_{gr} E_1 \cup \ldots \cup \sigma_{gr} E_k$ is a ground instance of the resolvent. ■

**6.4 Lemma.** For every T-unsatisfiable clause set CS, there exists a finite T-unsatisfiable set of ground instances.

**Proof.** Let CS $\cup$ T be unsatisfiable, Then by Herbrand´s Theorem II.11.2 there exists a finite set of ground instances of clauses from CS $\cup$ T. The finite set of ground instances of CS is T-unsatisfiable. ■

**6.5 Theorem.** [St85] Narrow T-resolution together with factoring is refutation-complete, i.e., every T-unsatisfiable set of clauses has a T-refutation of the empty clause.

**Proof.** Lemma 6.4 above on lifting shows that it suffices to consider a set CS of ground clauses. We can assume that CS is T-unsatisfiable. We make induction using the k-parameter.

Induction base. If $k(CS) = 0$, then CS consists of unit clauses. The clause set CS contains a minimally T-unsatisfiable clause set, hence in this case there exists a one-step refutation of CS.

Induction step. If $k(CS) > 0$, then CS contains a clause with more than one literal. We split C into two nonempty disjoint parts $C_1 \cup C_2$. $CS_i$ is the set CS where C is replaced by $C_i$. $CS_1$ and $CS_2$ are T-unsatisfiable, hence by induction on k(CS) there exists a refutation of $CS_1$ and $CS_2$. Since clauses are ground, the deductions are combinable as follows: If the deduction of the empty clause in $CS_1$ does not use the clause $C_1$, then we have already a refutation of CS. If a deduction uses $C_1$, then we can derive the clause $C_2$ and then we perform the deduction in $CS_2$. ■

The following example shows that M. Stickels argument that lifting is trivial and hence it is sufficient to consider only the ground case is not correct if applied to paramodulation or E-resolution. The hidden problem is that M. Stickel uses the usual (Robinson-) instance relation, and hence if T-resolution simulates E-resolution or paramodulation, far more T-resolvents are necessary.

E-resolution is not simulatable by T-resolution:

**6.6 Example.** Narrow T-resolvents may be not liftable, if the instance relation is not properly chosen:

This is an example, that E-resolution and narrow T-resolution with respect to the theory of equality are different notions.

We use the clause set in Example 5.7. and T should be the theory of equality.

Consider the three clauses {P(x , a), Q(x)}, {-P(y, b), R(y)} {a=b}.

For the ground instance {P(f(a), a), Q(f(a))}, {-P(f(b), b), R(f(b))} {a=b} we obtain the T-resolvent {Q(f(a)), R(f(b))} and for the ground instance {P(f(a), a), Q(f(a))}, {-P(f(a), b), R(f(a))} {a=b} we obtain the T-resolvent {Q(f(a)), R(f(a))}.

Thus a complete algorithm T-UNIFY has to generate not only the T-unifier $\{x \leftarrow y\}$, but also $\{x \leftarrow f(a), y \leftarrow f(b)\}$, and in fact it has to generate an infinite number of T-unifiers.

This is different from E-resolution, where only the unifier $\{x \leftarrow y\}$ is needed, if one adopts the completeness of the paramodulation based rigid E-unification algorithm. T-resolution in this case can be compared with E-resolution if the functionally reflexive axioms are present. ■

# VI A Sort Generating Algorithm

**Overview:** In this part we describe several transformations for sorted specifications. The main motivation for investigating these transformations is efficiency of the corresponding deduction system and the reduction of search spaces. The idea is to transform a complex and hard to prove specification into a simpler (easier to prove) one, where the number of clauses is reduced, but perhaps the sort structure is more complex. In order to gain efficiency these transformations should be fast and the result should be in some sense simpler.

In the first paragraph we describe several sensible transformation rules and call the set of rules also SOGEN. An evaluation of the rules of SOGEN-application is described. We prove that all these rules are correct and give some examples for the performance of SOGEN. In an extra paragraph the application of SOGEN to logic programs is described.

## 1. The Algorithm SOGEN.

The goal of this paragraph is to present the rule-based algorithm SOGEN. This algorithm takes as input a clause set and a signature nd searches for information in the clause set that can be encoded in terms of a sorted signature. In particular, it transforms unary predicates into appropriate sorts, adds new relations between sorts, and adds term declarations.

In this part, we always assume that a topsort TOP is present. Furthermore we assume that unary predicates have a unique maximal domain-sort. Both assumption are justified in § II.1.2 and II.7.8.

### 1.1 Preliminaries for SOGEN.

The algorithm SOGEN needs a memory to store already introduced relations on sorts and relationships between sorts and unary predicates. We use sort-predicate equivalences (SPE) and intersection constraints (ISC). The constraints and equivalences could be coded as special clauses, but we keep them separate from the clause set CS in consideration.

In the following we write P, if we mean a signed predicate. A unary predicate P for which a sort $S_P$ is generated, is called a **transformed** predicate. Note that if P is transformed, then -P may not be transformed into a sort.

Now we make precise what SPE and ISC means:

1) A pair $(P,S_P) \in SPE$, where $P \in \mathbf{P}$ and $S_P \in \mathbf{S}_\Sigma$ stands for "P is transformed into $S_P$". We denote this also by $P \leftrightarrow S_P$. Semantically, this means that P is valid exactly on elements of the sort $S_P$. In the clause set it can be simulated by the formulas $\forall x{:}S_P\ P(x)$ and $\forall x{:}S_{DP}\ P(x) \Rightarrow \exists y{:}S_P\ x = y$ , where $S_{DP}$ is the domain-sort of P.
In order to turn the axiom $\forall x{:}S_{DP}\ P(x) \Rightarrow \exists y{:}S_P\ x = y$ into a clause, we introduce the Skolem-function $g_P{:}S_{DP} \rightarrow S_P$ and then obtain the clause $\forall x{:}S_{DP}\ P(x) \Rightarrow x = g_P(x)$

2) A pair $(\{S_1,\ldots S_n\},T)$ with $S_i,T \in \mathbf{S}_\Sigma$ represents $S_1 \cap \ldots \cap S_n = T$. Semantically, this means that for every term t: if t has sorts $S_1,S_2,\ldots,S_n$, then t is also of sort T.
This information can be encoded by the relations $S_i \sqsupseteq T$ and an axiom like $\forall x_1{:}S_1, \ldots,x_n{:}S_n\ x_1 = x_2 \wedge \ldots \wedge x_1 = x_n \Rightarrow \exists z{:}T\ x_1 = z$. A short reflection shows that this can also be encoded as a clause with a function $h_{S1\rightarrow T}{:}S_1 \rightarrow T$ and the following clause: $\forall x_1{:}S_1, \ldots,x_n{:}S_n\ x_1 = x_2 \wedge \ldots \wedge x_1 = x_n \Rightarrow x_1 = h_{S1\rightarrow T}(x_1)$. ■

In general we use as signature $\Sigma$ only the part of the signature without symbols from ISC and SPE. If we use the whole signature (including the symbols $g_P$ and $h_{R\rightarrow T}$), we shall state it explicitely.

From the information in ISC, SPE and the signature we can immediately derive some new information such as sort-relations or equivalence of two sorts. For example from the information in ISC alone it may be possible to identify sorts. An algorithm for such a formal handling of sets is easy to construct using the idea of Venn-diagrams (cf. [Sh84] Example 3). In [Sh84] there are also unions and complements allowed. We use only intersection and the subset relation, and it turns out (see Lemma 3.4.3) that in this case there exists a more efficient algorithm than the Venn-diagram method.

In the following we describe the rules of SOGEN by an If-part that contains the conditions for firing and a then-part that contains the actions to be performed. Some rules that provide alternatives have in their then-part the alternatives in either, or,....

Not all rules are completely defined, as their implementation may rely on special heuristics or requires special algorithms. For example Rule BT1 has an undecidable precondition, but this condition describes best what is required. An implementation of the test for such a condition must only be correct, i.e. if the test says 'yes' , then the condition must be true, but the implementation may not be complete, i.e. it sometimes says 'no' or 'don't know' if the real answer is 'yes'.

## 1.2 Basic Transformation Rules.

**Rule BT1.** Introduction of sorts.

<u>If</u> CS contains a unary predicate P with domain-sort $S_{DP}$, and
$\exists x{:}S_{DP}\ P(x)$ is deducable from CS, and
P is not yet transformed into a sort,

<u>then</u> add a new sort symbol $S_P$,
add a new constant c of sort $S_P$,
add $S_P \sqsubseteq S_{DP}$, where $S_{DP}$ is the domain of P,
add $P \leftrightarrow S_P$ to SPE.

Note that the nonemptyness condition of BT1 is satisfied, if for example there is a unit clause P(t) or a clause $\{P(s_1),P(s_2),\ldots,P(s_n)\}$.

**Rule BT2.** Adding new term declarations to $\Sigma$.

<u>If</u> CS contains the clause $C = \{P(t)\}$, where t is not a variable, and
we have $P \leftrightarrow S_P$,

<u>then</u> add the term declaration $t{:}S_P$.

**Rule BT3.** Introduction of sort relations.

<u>If</u> CS contains the clause $\{P(x)\}$, where x has the sort $S_x$, and
we have $P \leftrightarrow S_P$

<u>then</u> add the relation $S_x \sqsubseteq S_P$.

**Rule BT4.** Changing the sort of variables in clauses.

<u>If</u> CS contains the clause $C = \{-P(x)\} \cup A$, where x has sort $S_x$, and
we have $P \leftrightarrow S_P$ and $S_P \cap S_x = T$,

<u>then</u> delete the literal -P(x) from C,
replace x by a new variable of sort T.

## 1.3 Reduction and Deletion Rules.

Besides the usual deduction rules like resolution, factoring or paramodulation and the usual reduction rules like subsumption and tautology reduction, we give a sligthly modified purity reduction rule and introduce some new deletion and reduction rules (which can be seen as subsumption or replacement resolution [Ro65,CL73], respectively)

**Rule DD1.** Purity deletion.

If CS contains a clause C such that C is a pure clause, i.e. $C = \{L\} \cup A$, the predicate P of L is not the equality predicate, neither P nor -P is transformed into a sort and there exists no complementary literal in any of the clauses of CS,

then delete C from CS.

**Rule DD2.** Special subsumption.

If CS contains a clause C of the form $C = \{P(t)\} \cup A$ and

$P \leftrightarrow S_P$ and

$S_P \in S_\Sigma(t)$,

then delete C from CS.

**Rule DD3.** Literal deletion (replacement resolution).

If CS contains $C = \{-P(t)\} \cup A$ and

$P \leftrightarrow S_P$ and

$S_P \in S_\Sigma(t)$,

then delete the literal -P(t) from C.

## 1.4 Manipulations Based on ISC.

We say ISC is **regular,** iff i) for all relations $S_1 \cap \ldots \cap S_n = T$, we have that T is the greatest common subsort of $S_1,\ldots,S_n$ and ii) all relations that follow only from ISC and $\Sigma$ are already in $\Sigma$.

**Rule ISC1.** Introduction of sort relations by intersection constraints.

If we can derive $S \sqsubseteq T$ from the relations in ISC,

then add $S \sqsubseteq T$ to $\Sigma$.

The following rule can be derived from the contraposition rule $(P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)$.

**Rule ISC2.** Application of contraposition.

If $S_1 \sqsubseteq S_Q$ and

$P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$ and $Q \leftrightarrow S_Q$ and $-Q \leftrightarrow S_{-Q}$ and

$S \cap S_P = S_1$ and $S \cap S_{-Q} = S_2$,

then add $S_2 \sqsubseteq S_{-P}$ to $\Sigma$.

**Rule ISC3**. Introducing a new sort as the intersection of sorts.

<u>If</u> $S_1,\dots,S_n$ have a common subsort,
and there is no sort S such that $S_1 \cap \dots \cap S_n = S$ is derivable from ISC,

<u>then</u> add a new sort $S_N$,
add the relation $R \sqsubseteq S_N$ for every sort R with $R \sqsubseteq S_i$ for all i,
add the relations $S_N \sqsubseteq S_i$ for all i=1,...,n
add the relation $S_1 \cap \dots \cap S_n = S_N$.

**Rule ISC4**. Introducing new intersection constraints.

<u>If</u> $S_1,\dots,S_n$ have a common subsort,
$S_1,\dots,S_n$ have $S_N$ as greatest common subsort,
the signature is regular,
there are no equations in CS,
ISC is regular,
SPE is empty,

<u>then</u> add the relation $S_1 \cap \dots \cap S_n = S_N$.

### 1.5 Equivalence of Sorts

**Rule ES1.** Deletion of cycles in $<S_\Sigma,\sqsubseteq>$.

<u>If</u> there exist sorts S,T such that $S \sqsubseteq T$ and $T \sqsubseteq S$,

<u>then</u> replace everywhere in the signature and CS (also in SPE and ISC) the two sorts S and T by one new symbol.

### 1.6 Manipulations within the Signature.

The rules MS2 and MS3 in this paragraph are meta-rules, i.e. they do not specify how they can be implemented. In general it is sufficient for rule MS2 to perform Σ-unification on the term declarations to find the interesting terms (see III.6.7).

As a standard rule we have that redundant term declarations have to be removed:

**Rule MS1.**

Remove redundant term declarations.

**Rule MS2.** Adding intersections of range-sorts.

If there is a term s with $S_\Sigma(s) = \{S_i \mid i = 1,\ldots,n\}$, and
$S_\Sigma(s)$ has no unique least sort,
there is no sort S with $S = \bigcap S_\Sigma(s)$,

then add a new sort $S_N$,
add a new constant c of sort $S_N$ and
add the relations $S_N \sqsubseteq S_i$ for all i and
add the relation $S_1 \cap \ldots \cap S_n = S_N$.

**Rule MS3.** Adding a term declaration for intersection information.

If there is a term s with $S_\Sigma(s) = \{S_i \mid i = 1,\ldots,n\}$, and
$S_\Sigma(s)$ has no unique least sort, and
there is a sort S such that $S = S_1 \cap \ldots \cap S_n$ is derivable from ISC,

then add the term declaration s:S to $\Sigma$.

## 1.7 Reducing ISC and SPE.

Here we give rules that provide a complete procedure to remove the extra clauses representing SPE and ISC.
The rule RSPE is actually subsumed by the next rule. Nevertheless, we state it explicitely because of its importance.

**Rule RSPE.** Non complementary predicates.

If neither P nor -P occurs in CS and
we have $P \leftrightarrow S_P$ but not $-P \leftrightarrow S_{-P}$

then remove P from the signature and remove the relation $P \leftrightarrow S_P$ from SPE.

**Rule R-SPE&ISC.** (complementary predicates, no equations).

If for all predicates P:
if P has domain sort $S_{DP}$ and
neither P nor -P occurs in CS and
for every ground term t with $S_{DP} \in S_\Sigma(t)$ we have either $S_P \in S_\Sigma(t)$ or $S_{-P} \in S_\Sigma(t)$, where $P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$ ,
the signature is ground-regular,
ISC is regular,

then for all transformed predicates P:

remove P from $\Sigma$ and remove the relations $P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$ from SPE,

remove all relations from ISC.

Note that in the next rule we use the E-semantical sort-assignment $S_{\Sigma,E}$ (cf. II. 9)

**Rule R-SPE&ISC-E** (complementary predicates with equations).

If CS can be separated into equality-free clauses $CS_{-E}$ and clauses for an equational theory E,

for all predicates P:

if P has domain sort $S_{DP}$ and

neither P nor -P occurs in CS and

for every ground term t with $S_{DP} \in S_{\Sigma,E}(t)$ we have either $S_P \in S_{\Sigma,E}(t)$ or $S_{-P} \in S_{\Sigma,E}(t)$, where $P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$ and

the sort-assignment $S_{\Sigma,E}$ is ground-regular,

ISC is regular with respect to $S_{\Sigma,E}$,

then for all transformed predicates P:

remove P from $\Sigma$ and remove the relations $P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$ from SPE.

remove all relations from ISC.

## 1.8 A Weakening Rule.

**Rule WT**

If CS contains no equations,

$\Sigma$ is regular,

ISC is regular,

For all transformed predicates P:

-P is not transformed,

there is no literal of the form P(s) in CS,

there is a transformed predicate $Q \leftrightarrow S_Q$,

there is a clause $C = \{-Q(t)\} \cup C_R\}$,

then replace C by the clauses $\sigma_1 C_R, \sigma_2 C_R, \ldots$ where $\{\sigma_1, \sigma_2, \ldots\} = \mu W_\Sigma(t \sqsubseteq S_Q)$.

## 1.9 Analysis by Cases.

The following rules work by analysis of cases. The different cases are given in the either and or part. For deduction systems this means that both cases have to be refuted separately.

**Rule AC1.** Either P is universally true or there exists y such that P(y) is false.

If P is a predicate with domain $S_{DP}$ and P is not yet transformed,

then either add the clause $\forall x{:}S_{DP}\ P(x)$ to the clause set CS.

or add a new constant c of sort $S_{DP}$ to the signature and the unit clause $-P(c)$ to the clause set CS.

**Rule AC2.** For a constant c either P(c) or -P(c) is valid.

If P is a predicate with domain $S_{DP}$ and c a constant of sort $S_c$ and
$S_c \sqsubseteq S_{DP}$, but $S_c \not\sqsubseteq S_P$ and $S_c \not\sqsubseteq S_{-P}$ and
$P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$

then either add the clause $\{P(c)\}$ to CS

or add the clause $\{-P(c)\}$ to CS.

**Rule AC3.** (For sets A,B we have either $B \subseteq A$ or $B \subseteq \bar{A}$ or $B \cap A \neq \emptyset$ and $B \cap \bar{A} \neq \emptyset$ ).

If P is a predicate with domain $S_{DP}$ and
S a sort with $S \sqsubseteq S_{DP}$, but $S \not\sqsubseteq S_P$ and $S \not\sqsubseteq S_{-P}$ and
$P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$

then either add the clause $\forall x{:}S\ P(x)$

or add the clause $\forall x{:}S\ -P(x)$

or add new constants $c_+$ and $c_-$ of sort S and add the clauses $P(c_+)$ and $P(c_-)$.

**Rule AC4.** Splitting a clause into two clauses.

If there is a predicate P with domain $S_{DP}$ and
there is a clause C with a variable x of sort $S_x \sqsubseteq S_{DP}$ and
$P \leftrightarrow S_P$, $-P \leftrightarrow S_{-P}$ and
$S_x \cap S_P = S_1$ and $S_x \cap S_{-P} = S_2$

then replace the clause C by the two clauses $C_1$ and $C_2$, where $C_i$ is obtained from C by replacing x in C by a new variable $x_i$ of sort $S_i$.

## 1.10 Termination Conditions.

**Rule TCO1**

If we have $P \leftrightarrow S_P$, $-P \leftrightarrow S_{-P}$, and
$S_P$ and $S_{-P}$ have a common subsort
then the specification is contradictory.

**Rule TCO2**

If the clause set CS is empty and ISC and SPE are empty
then the original clause set is satisfiable.

**Rule TCO3**

If the clause set CS is empty,
the signature is regular and elementary,
ISC is regular,
for every predicate P with domain $S_{DP}$ such that P and -P is transformed:
If for all $S \sqsubseteq S_{DP}$: either $S \sqsubseteq S_P$ or $S \sqsubseteq S_{-P}$
then the original clause set is satisfiable.

**Rule TCO4.**

If some clause is empty,
then a refutation has been found.

The algorithm SOGEN has succeeded, if at the end the rules RISC and RSPE are able to remove all relations from SPE and ISC. If some relations in SPE and ISC are retained, then there are the alternatives to either add the appropriate clauses (cf. 2.1) or else to delete these clauses and accept the incompleteness of the proof procedure.

## 1.11 Manipulations Caused by Equalities.

**Rule EQ1.** Existence of an intersection sort.

If CS contains a clause $\{s=t\}$, where $S \in S_\Sigma(s)$ and $T \in S_\Sigma(t)$ and
S and T have no common subsort
then add a new sort symbol $S_N$ with $S_N \sqsubset S$ and $S_N \sqsubset T$ and
add the relation $S \cap T = S_N$,
add a new constant $c_N$ of sort $S_N$.

**Rule EQ2.** New term declarations.

<u>If</u> CS contains a clause $\{s = t\}$, where $T \in S_\Sigma(s)$ and $S \in S_\Sigma(t)$ and we have the relation $S \cap T = R$

<u>then</u> add the term declarations s:R and t:R to $\Sigma$.

**Rule EQ3.** New sort relations.

<u>If</u> CS contains a clause $\{x = t\}$ where x is a variable of sort $S_x$ and $T \in S_\Sigma(t)$

<u>then</u> add the relation $S_x \sqsubseteq T$ to $\Sigma$.

## 1.12 How the Rules Work.

We describe, which rules are tightly connected and which combination of rules solve some subproblems, such as making the signature polymorphic. Furthermore we assign the blocks of rules their priorities for application.

These priorities of the rules is essential, since the set of rules without any priority may run in a loop. Furthermore we have implicitly as a metarule that a rule is only applicable if it actually changes anything.

1) The rules BTi and DDi have highest priority. They should be applied, whenever possible (but BT1 not if the predicate is inhibited for transformation). Every application of a rule BT2, BT3 or BT4 should be followed by the deletion of the corresponding literal.
2) The rules ISC1, ISC3, ISC4, ES1 form a block of rules, whose objective is to complete the sort structure, such that for all sorts $S_1,S_2$ either they have no common subsorts or they have a unique greatest common subsort, i.e. $\langle S_\Sigma, \sqsubseteq\rangle$ is a semi-lattice.
   However, the semilattice completion can be made as in II.7.12 or can be omitted.
3) The rule ISC2 permits to encode more information into the signature. It avoids the situation where the relations between the sorts $S_P$ depend on a sequence of rule applications. For example the clause $P \Rightarrow Q$ is equivalent to $-Q \Rightarrow -P$, but if the relation $S_P \sqsubseteq S_Q$ is generated, then the clause is deleted, but the relation $S_{-Q} \sqsubseteq S_{-P}$ may be missing. The rule ISC2 inserts this relation.
4) The rules MS1, MS2, MS3 together with the rules ISC1, ISC3, ISC4, ES1 can make the signature polymorphic. The priority of rules should be: MS2, MS3 , ISC1, ES1, ISC3, ISC4, MS1.
5) The rules RSPE, R-SPE&ISC or R-SPE&ISC-E should fire, if no other rules are applicable. If all relations SPE and ISC are removed, then the algorithm has succeeded.
6) The rules ACi need some control and heuristics, since it depends on global information or

knowledge, which of these applications may contribute to a proof. Note that every application of a rule ACi could be followed by a rule BTj.

7) The rules EQi are problematic, since in calculi that store the unifiers and use inheritance of unifers (like connection graph calculi ([Oh86, Ko75, KM84] or matrix calculi [And81, Bib81a, Bib81b]) all stored unifiers have to be recomputed after application of these rules.
8) If no equations are present, then the intersection constraints cannot be ignored. But at the start of SOGEN we can assume that the greatest common subsort of a set of sorts is also their intersection.
9) A control module for SOGEN may inhibit some unary predicates from transforming them into sorts. For example, the first run of SOGEN may have produced a failure and now SOGEN starts a transformation of a reduced set of predicates.

## 2 Sort Generation in Logic Programs.

There are several proposals to extend logic programming languages like PROLOG [CM81, Ko79a, Ko79b, Ll84] by sorts [ Bü85, GM85b, Sm86].

We give a subset of the rules of SOGEN as a transformation procedure for logic programs without equations.

One possibility is to transform the program together with the query. This approach has the disadvantage that for every new query the whole program and the new query have to be transformed again and that the result of such program transformations may depend on the queries.

We propose the following procedure: Transform the program without queries and keep the relations in SPE. Answering a query consists in transforming this query, making the deductions and transforming the answers back. An alternative is to give the new sort-information to the user and to answer queries without transforming answer back.

### 2.1 Example.

Let the clause set be : $P(a)$, $\{-P(x), P(f(x))\}$

The answer to a query of the form $?P(y)$ is an infinite set of substitutions, namely $y = a, y = f(a)$, $y = f^2(a), \ldots$

A sort generation yields an empty clause set and the signature:

$P \leftrightarrow S_P$, $a{:}S_P$ , $f{:}S_P \rightarrow S_P$.

Now the answer to a query $?P(y)$ is 'y has sort $S_P$'. An appropriate answer in terms of the original signature is to generate all ground terms of sort $S_P$, that is $\{a, f(a), f^2(a), \ldots\}$ ■

We give a description of a sort generating algorithm for logic programs, the exact formulation of the rules is similar to paragraph 1. We exhibit the complexity of every step and give hints to avoid exponentiality in an implementation.

**2.2 Definition.** The following procedure is used to transform a logic program and queries.

i) If ?P(x) succeeds as query,
   then introduce a new sort $S_P$ and the relation $S_P \leftrightarrow P$.

ii) If there is a fact P(t), where t is not a variable and we have $S_P \leftrightarrow P$,
   then add the term declaration $t{:}S_P$

iii) If we have the fact P(x), where the sort of x is $S_x$ and we have $S_P \leftrightarrow P$,
   then add the relation $S_x \sqsubseteq S_P$.

iv) If there is a clause C with a literal P(x) in its body, where the sort of x is $S_x$ and we have $P \leftrightarrow S_P$ and $S_{Px}$ is the intersection of $S_x$ and $S_P$,
   then delete P(x) from C, and
   replace x by a variable y of sort $S_{Px}$.

v) Make $\Sigma$ and ISC regular.

vi) Make simplifications like

   $P(t) \rightarrow$ TRUE, iff $S_P \in S_\Sigma(t)$.

   That means: delete literals in the body, if they satisfy this condition and if the literal is the head, then delete the whole clause.

vii) A query is transformed and answered as follows:

   If P(t) is a literal in the query with $P \leftrightarrow S_P$, then the answer-substitutions are exactly the most general weakenings for the problem $\langle t \sqsubseteq S_P \rangle$. The other literals are treated as usual. ❑

The procedure succeeds, if for every predicate P that is transformed (i.e. $P \leftrightarrow S_P$), there are no more literals starting with P or -P in the clause set. If the weakening rule is used also for literals in the clause, then the procedure succeeds for a larger class of logic programs. However, the complexity may be exponential in this case, since there may be an exponential number of clauses necessary even for polymorphic signatures (cf. III.4).

The procedure described above has the drawback that in the general case it may be possible that the weakening problems are undecidable (cf. III.6). This, however, is nothing really new since Horn clause deduction in itself is undecidable.

A remedy for this problem is to try to obtain a polymorphic signature as the result of this transformation. In order to achieve this the above rules have to be restricted such that only function declarations are introduced. In this case the weakening problem is NP-complete

(Proposition III.4.3).

A source of exponentiality is the completion to a semilattice, which should be made if it is not exponential and should be avoided otherwise.

**2.3 Theorem.** The transformation of a Horn clause program is possible in polynomial time, if the procedure satisfies the following preconditions:

i) It transforms the signature into an elementary one.

ii) The lattice-completion is not performed.

iii) It tests regularity instead of ground regularity.

iv) Intersection of sorts is only introduced if-needed, i.e., in rules BT4 and MS3.

v) The weakening rule is not used.

**Proof.** Lemma 3.4.5 shows that the tests corresponding to ISC can be performed in polynomial time. The intersection of a sort in BT4 is not critical, since a literal is deleted, hence rule BT4 can introduce at most as many sorts as literals are in ISC. In the case of elementary signatures it suffices for Rule MS3 to consider terms of the form $f(x_1,\ldots,x_n)$. The number of such terms is polynomial in the number of sorts and functions. Hence at most a polynomial number of new sorts has to be introduced. This implies that the whole procedure can be performed in polynomial time. ■

## 3 The Rules of SOGEN are Conservative.

This paragraph provides a theoretical foundation for the sort generating procedure described in §1. It can be seen as an extension of II.7.

We try to organize this chapter in the same way as chapter 1, such that the proofs and the examples for rules of paragraph 1.n are now in paragraph 3.n.

For the proofs of conservativeness we let $\mathcal{S}_1$ be the specification before the transformation and let $\mathcal{S}_2$ be the specification after the transformation. The aim of the proofs is then to show that $\mathcal{S}_1$ has a $\Sigma_1$-model iff $\mathcal{S}_2$ has a $\Sigma_2$-model.

### 3.1 ISC- and SPE-clauses.

We show that the clauses in 1.1 are appropriate for encoding equivalence of sorts and predicates and for encoding intersection information.

**3.1.1 Lemma.** Let P be a predicate with domain $S_{DP}$ and let $S_P \sqsubseteq S_{DP}$ be a sort.

Then the axioms $\forall x{:}S_P\ P(x)$ and $\forall x{:}S_{DP}\ (P(x) \Rightarrow \exists y{:}S_P\ x = y)$ imply that in every $\Sigma$-model $\mathcal{D}$ of every clause set CS we have $P_{\mathcal{D}} = S_{P,\mathcal{D}}$.

**Proof.** Obviously, the first axiom implies $S_{P,\mathcal{D}} \subseteq P_{\mathcal{D}}$. The second axiom implies the converse: Let $d \in P_{\mathcal{D}}$, then there exists a $\Sigma$-assignment $\varphi$ with $\varphi x = d$. Since $\mathcal{D}$ is a model, the first literal P(x) is true, hence the second assertion $(\exists y{:}S_P\ x = y)$ is true under $\varphi$. This means that d is in $S_{P,\mathcal{D}}$. ■

**3.1.2 Lemma.** Let $S_1, \ldots S_n, T$ be sorts with $S_i \sqsupseteq T$. Then the axiom $\forall x_1{:}S_1, \ldots, x_n{:}S_n\ x_1 = x_2 \wedge \ldots \wedge x_1 = x_n \Rightarrow \exists z{:}T\ x_1 = z$ implies that for every model $\mathcal{D}$ of this axiom we have $S_{1,\mathcal{D}} \cap \ldots \cap S_{n,\mathcal{D}} = T_{\mathcal{D}}$.

**Proof.** We show $S_{1,\mathcal{D}} \cap \ldots \cap S_{n,\mathcal{D}} \subseteq T_{\mathcal{D}}$:

Let $d \in S_{1,\mathcal{D}} \cap \ldots \cap S_{n,\mathcal{D}}$. Then there exists a $\Sigma$-assigment $\varphi$ such that $\varphi x_i = d$ for all i. Since $\mathcal{D}$ is a model, the first assertion of the axiom is true, hence also the second $\exists z{:}T\ x = z$. This means d is equal to an element of $T_{\mathcal{D}}$, hence $d \in T_{\mathcal{D}}$. ■

## 3.2 The Basic Transformation Steps are Conservative.

**3.2.1 Lemma.** Rule BT1 is conservative.

**Proof.**

i) Let $\mathcal{D}_1$ be a model of $\mathcal{S}_1$. Since $\exists x{:}S_{DP}\ P(x)$ is deducable, the set $P_{\mathcal{D}1}$ is not empty, hence we can construct a $\Sigma_2$-model by assigning the new constant c an element from $P_{\mathcal{D}1}$ and by defining $S_{P,\mathcal{D}2} := P_{\mathcal{D}1}$. The relation $P \leftrightarrow S_P$ is then satisfied. Furthermore all clauses remain valid.

ii) To show the converse, let $\mathcal{D}_2$ be a model of $\mathcal{S}_2$. Then obviously $\mathcal{D}_2$ is also a $\Sigma_1$-model of $\mathcal{S}_1$. ■

**3.2.2 Example.** The introduction of sorts is not conservative if $\exists x{:}S_{DP}\ P(x)$ is not deducable:

Let the clause set be {-P(x), Q}, {-P(x), -Q}

This clause set has a model, in which -P(x) is always valid.

The introduction of the sort $S_P$ transforms this clause set with the rules BT4 into Q,-Q, which is obviously unsatisfiable. ■

The next lemma is an important one, since it gives a direct correspondence between unit clauses P(t) and term declarations of the form $t{:}S_P$. In this lemma a lot of previous work culminates, such as work on algebras and on conservative transformations.

**3.2.3 Lemma.** The rule BT2 is conservative.

**Proof.**

i) Let $\mathcal{D}_1$ be a $\Sigma_1$-model of $\mathcal{S}_1$. The transformation H is well-sorted in the sense of II.7.1, since only a term-declarations is added to the signature. We use Theorem II.7.6 to prove that H is conservative. Therefore we have to show that $H(\mathcal{D}_1)$ is also a $\Sigma_2$-algebra. This follows trivially from Definition I.6.1ii).

ii) To show the converse, let Let $\mathcal{D}_2$ be a $\Sigma_2$-model of $\mathcal{S}_2$. We use Theorem II.7.6 ii) to prove that H is conservative. Therefore we have to show that $\mathcal{D}_1$ is a $\Sigma_1$-algebra, if $H(\mathcal{D}_1)$ is a $\Sigma_2$-algebra.

Due to Definition I.6.1ii) the only condition to check is that for the new term declaration $t{:}S_P$, and every $\Sigma_1$-assignment $\varphi$, we have $\varphi t \in S_{P,\mathcal{D}1}$. We have that $\varphi$ is also a $\Sigma_2$-assignment and that $S_{P,\mathcal{D}1} = S_{P,\mathcal{D}2}$, that $\mathcal{D}_2$ is a $\Sigma_2$-model of $\mathcal{S}_2$ and that $S_{P,\mathcal{D}1} = P_{\mathcal{D}1}$. Hence P(t) is true under $\varphi$, which implies $\varphi t \in P_{\mathcal{D}1} = S_{P,\mathcal{D}1}$. ■

**3.2.4 Lemma.** The rule BT3 is conservative.

**Proof.**

i) Let $\mathcal{D}_1$ be a $\Sigma_1$-model of $\mathcal{S}_1$. The transformation H is well-sorted in the sense of II.7.1, since only a sort-declaration is added to the signature. We use Theorem II.7.6 to prove that H is conservative. Therefore we have to show that $H(\mathcal{D}_1)$ is also a $\Sigma_2$-algebra. Due to Definition I.6.1i) we have to check that $S_{x,\mathcal{D}2} \subseteq S_{P,\mathcal{D}2}$. This is the case, since $\mathcal{D}_1$ is a model for P(x) and $S_{P,\mathcal{D}2} = P_{\mathcal{D}1}$.

ii) This direction is trivial. ■

**3.2.4 Lemma.** The rule BT4 is conservative.

**Proof.**

i) Let $\mathcal{D}_1$ be a $\Sigma_1$-model of $\mathcal{S}_1$. Then it is a $\Sigma_1$-model of the clause $C_1 = \{-P(x), A\}$. Consider the clause $C_2$ where the literal -P(x) is deleted and x is replaced by a variable y of sort T. Let $\varphi$ be a $\Sigma_2$-assignment. Then $\varphi$ is also a $\Sigma_1$-assignment, but $\varphi$ makes -P(x) false. This implies that $\varphi$ makes A valid, and hence also $C_2$.

ii) Let $\mathcal{D}_2$ be a $\Sigma_2$-model of $\mathcal{S}_2$. Then it is a $\Sigma_2$-model of the clause $C_2$ = A. Let $\varphi$ be a $\Sigma_1$-assignment. Then either $\varphi$ makes -P(x) true and hence the whole clause $C_1$ or $\varphi$ makes -P(x) false. In the second case $\varphi$ is also a $\Sigma_2$-assignment, hence $\varphi$ makes $C_2$ valid, which in turn implies that also $C_1$ is valid under $\varphi$. ■

## 3.3 The Deletion Rules are Conservative.

**3.3.1 Lemma.** Purity reduction is conservative.

**Proof.** Is the same as the usual proof, since neither $P \leftrightarrow S_P$ nor $-P \leftrightarrow S_{-P}$ holds. ■

**3.3.2 Lemma.** The rule DD2 is conservative.

**Proof.** The literal P(t) is always valid in models, since $t \in S_{P,\mathcal{D}1} = P_{\mathcal{D}1}$. ■

**3.3.3 Lemma.** The rule DD3 is conservative.

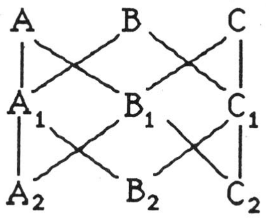
**Proof.** The literal -P(t) is always false in models, since $t \in S_{P,\mathcal{D}1} = P_{\mathcal{D}1}$, hence the literal can be deleted. ■

## 3.4 The ISC-Manipulations are Conservative.

First we give some examples, how sorts may be recognized as equivalent or how a subsort relation may be derived from information in ISC. We give also a short explanation of how the algorithm in [Sh84] works.

**3.4.1 Example.** Let $A,B,C,A_1,B_1,C_1,A_2,B_2,C_2 \in S_\Sigma$ and let $A \cap B = A_1$, $A \cap C = B_1$, $B \cap C = C_1$, $A_1 \cap B_1 = A_2$, $A_1 \cap C_1 = B_2$, $B_1 \cap C_1 = C_2$.

The following diagram shows the relationships:




A short reflection shows that $A_2,B_2$ and $C_2$ are equivalent, since they all represent $A \cap B \cap C$.

We demonstrate how the problem can be solved using Venn-diagrams :

There are 7 constituents of the diagram: $A = E_1 + E_2 + E_3 + E_4$, $B = E_2 + E_3 + E_5 + E_6$, $C = E_3 + E_4 + E_6 + E_7$. The above equations give the relations for $A_1$, $B_1$ and $C_1$: $A_1 = E_2 + E_3$, $B_1 = E_3 + E_4$, $C_1 = E_3 + E_6$. Finally, we get $A_2 = B_2 = C_2 = E_3$.

**3.4.2 Example.** Let A,B,C,D be sets, such that $A \cap B = C \cap D$ holds.

Then $B \cap C \supseteq A \cap B$, since $A \cap B = (A \cap B) \cap (C \cap D)$.

Again, we demonstrate the solution using Venn-diagrams:

The Venn-diagram contains 15 constituents, which we index by strings of the form

ABC instead of numbers: for example $E_{AB}$ means all $A \cap B \cap \bar{C} \cap \bar{D}$.
From $A \cap B = C \cap D$ we derive the equation
$E_{AB} + E_{ABC} + E_{ABD} + E_{ABCD} = E_{CD} + E_{ACD} + E_{BCD} + E_{ABCD}$.
This requires that all the consituents $E_{AB}$, $E_{ABC}$, $E_{ABD}$, $E_{CD}$, $E_{ACD}$, $E_{BCD}$ are empty.
Hence $A \cap B = A \cap B \cap C \cap D = E_{ABCD}$.
Now $B \cap C = E_{BC} + E_{ABCD}$ which is a superset of $A \cap B = E_{ABCD}$. ■

This method has as an advantage that it is rather intuitive. It demonstrates, that this theory is decidable and how to compute the relations. However, the number of constituents in a disjoint union is exponential in the number of different set symbols, hence its applicability is restricted. We show that our problem is not the full word problem in Boolean algebra and that there is a polynomial algorithm:

This translation of the ISC-deductions into propsitional Horn-clauses is more suitable for our purposes and better than the method of Venn-diagrams. Thus we used this approach for our implementation.

First we consider propositional Horn clauses. In [Bö85] it is mentionned on p. 384, that propositional Horn clauses can be decided in polynomial time.

**3.4.3 Lemma.** A set of propositional Horn clauses can be decided in at most quadratic time.
**Proof.** The decision algorithm is as follows:

1) If there are no unit clauses, then the clause set is satisfiable.
2) If an empty clause is obtained, then the clause set is unsatisfiable
3) If P is a (positive) unit, delete it from the body of all other clauses and if it is in the head of some clause, then delete this clause.

This algorithm is quadratic, since to find a unit is a linear task, and to delete a variable from the clause set is also linear, hence the whole procedure is at most quadratic. ■

Presumably the decision algorithm can be performed in quasilinear time by using a suitable datastructure that avoids the waste of time in searching for a unit and searching for a clause with an occurrence of some unit.

**3.4.4 Proposition.** A relation in ISC is decidable in polynomial time.
**Proof.** We have relations of the form $S_1 \cap \ldots \cap S_n = S$, which can be translated into the clause $S_1 \wedge \ldots \wedge S_n \Rightarrow S$ and n Horn-clauses $S \Rightarrow S_i$. These are 2*n literals. The subsort relation from $\Sigma$ can be translated into Horn-clauses of the form $R \Rightarrow S$ for $R \sqsubseteq S$. A relation to be decided is one of the following: i) $R \sqsubseteq S$? or ii) is $S_1 \cap \ldots \cap S_n = S$ ? or

iii) is $S_1 \cap \ldots \cap S_n$ equivalent to an existing sort? or iv) does a new relation $R \sqsubseteq S$ hold after changing ISC.

The time complexity is polynomial and we give an upper bound in terms of the number of literals $n_I$ of ISC , the number of literals $m_Q$ in the query and the number of sorts.

In the case i) $n_I^2$, In case ii): $(n_I+n_Q)^2$ in case iii) : $|S_\Sigma| * (n_I+n_Q)^2$, in case iv): $|S_\Sigma|^2 * n_I^2$. ■

**3.4.5 Lemma.** Rule ISC1 is conservative.

**Proof.**

i) If $\mathcal{S}_1$ has a $\Sigma_1$-model $\mathcal{D}_1$, then $\mathcal{D}_1$ is also a $\Sigma_2$-model of $\mathcal{S}_2$, since the newly derived relation holds in $\mathcal{D}_1$ .

ii) The converse is trivial. ■

**3.4.6 Lemma.** Rule ISC2 is conservative.

**Proof.**

i) This direction preserves models :

$S \cap S_P \subseteq S \cap S_Q$ implies $S \cap S_{-Q} \subseteq S \cap S_{-P}$ by taking the complements.

ii) The other direction is trivial. ■

**3.4.7 Lemma.** Rule ISC3 is conservative.

**Proof.**

i) If $S_1,\ldots,S_n$ have a common subsort, then in a model $\mathcal{D}_1$ the denotation of the newly introduced intersection-sort can be chosen as the intersection of the denotations of $S_1,\ldots,S_n$.

ii) trivial. ■

**3.4.8 Lemma.** Rule ISC4 is conservative.

**Proof.**

i) Let $\mathcal{S}_1$ have a $\Sigma_1$-model. Then CS has a $\Sigma_1$-model with $\mathbf{T}_{\Sigma,gr}$ as carrier (see I.8.7). (without regard to ISC). This model of ground terms satisfies all possible ISC-restriction of the kind $S_1 \cap \ldots \cap S_n = S_N$, where $S_N$ is the greatest common subsort of $S_1,\ldots,S_n$. Hence all such relations can be added.

ii) Trivial. ■

### 3.5 Using Equivalence of Sorts is Conservative.

**3.5.1 Lemma.** Rule ES1 is conservative.

**Proof.** Both directions are trivial, since equivalent sorts have the same denotation.■

### 3.6 Signature Manipulations are Conservative.

**3.6.1. Lemma.** The removal of redundant term declarations is conservative.

**Proof.** Trivial (cf. I.4) .

**3.6.2 Lemma.** Rule MS2 is conservative.

**Proof.** i) Let $\mathcal{D}_1$ be a $\Sigma_1$-model of $\mathcal{S}_1$. Then $S_{1,\mathcal{D}1} \cap \ldots \cap S_{1,\mathcal{D}1} \neq \emptyset$, since the sorts have s as common element. Hence we can add a new sort $S_N$ as subsort of $S_i$ and additionally as intersection $S_1 \cap \ldots \cap S_n$.

ii) trivial. ■

**3.6.3 Lemma.** Rule MS3 is conservative.

**Proof.** The proof is analogous to the proof of 3.2.3. ■

### 3.7 The Reduction Rules for ISC and SPE are Conservative.

**3.7.1 Lemma.** Rule RSPE is conservative.

**Proof.**

i) trivial

ii) Let $\mathcal{D}_2$ be a $\Sigma_2$-model of $\mathcal{S}_2$. Since P is completely removed, we can define the relation $P_{\mathcal{D}1}$ as $S_{P,\mathcal{D}1}$. Furthermore we define $d = g_P(d)$ for all $d \in S_{PD,\mathcal{D}1}$. Then the new structure is a model of $\mathcal{S}_1$. ■

**3.7.2 Lemma.** Rule R-SPE&ISC is conservative.

**Proof.** i) trivial,

ii) Let $\mathcal{D}_2$ be a $\Sigma_2$-model of $\mathcal{S}_2$. Then there exists a $\Sigma_2$-model of $CS_2$, and hence a $\Sigma_2$-model which has as carrier the set $T_{\Sigma,gr}$. Since the signature is ground-regular, all intersection constraints are satisfied, since ISC is regular. Furthermore the condition on the ground terms implies that $S_P \cup S_{-P} = S_{DP}$ and $S_P \cap S_{-P} = \emptyset$. Hence all relations in SPE can be satisfied. This means we have a $\Sigma_1$-model of $\mathcal{S}_1$. ■

**3.7.3 Lemma.** Rule R-SPE&ISC-E is conservative.

**Proof.** i) trivial.

ii) Let $\mathcal{D}_2$ be a $\Sigma_2$-model of $\mathcal{S}_2$. Then there exists a $\Sigma_2$-model of $CS_2$, and hence a $\Sigma_2$-model which has as carrier the set $\mathbf{T}_{\Sigma,gr} / =_{\Sigma,E}$ (see Corollary I.8.7). Since the signature is ground-regular with respect to E, all intersection constraints are satisfied, since ISC is regular with respect to $\mathbf{S}_{\Sigma,E}$. Furthermore the condition on the ground terms implies that $S_P \cup S_{-P} = S_{DP}$ and $S_P \cap S_{-P} = \emptyset$. Hence all relations in SPE can be satisfied.

This means we have a $\Sigma_1$-model of $\mathcal{S}_1$. ■

We give some examples which demonstrate that the removal of relations from ISC and SPE is not correct, if the preconditions of the rules are not satisfied.

**3.7.4 Example.** If the signature is not regular, then it is not correct to remove intersection information.

Let $\Sigma := \{A \sqsupset C, B \sqsupset C, c{:}A, c{:}B\}$ and let the clause set CS be $\{R(c); -R(x{:}C)\}$ and let ISC be $\{A \cap B = C\}$.

Then CS is unsatisfiable. However, if $A \cap B = C$ is removed, then CS becomes satisfiable.

**3.7.5 Example.** It is not sufficient to formulate the separation-condition in R-SPE&ISC and R-SPE&ISC-E for just one predicate:

Let $\Sigma := \{S \sqsupset S_Q, S \sqsupset S_{-Q}, S_Q \sqsupset S_P, S_Q \sqsupset S_{-P}, c{:}S, d_1{:}S_P, d_2{:}S_{-P}, d_3{:}S_{-Q}\}$ and let

SPE := $\{S_P \leftrightarrow P, S_{-P} \leftrightarrow -P, S_Q \leftrightarrow Q, S_{-Q} \leftrightarrow -Q\}$ and let the clause set be

CS := $\{\{R(c)\}, \{-R(x{:}S_P)\}, \{-R(x{:}S_{-P})\} \{-R(x{:}S_{-Q})\}\}$

This clause set is unsatisfiable, the signature is regular and satisfies the separation-condition for the predicate P. However, if we remove P, $S_P \leftrightarrow P$ and $S_{-P} \leftrightarrow -P$, then this clause set becomes satisfiable.

**3.7.6 Example.** It is not conservative to remove relations from ISC if SPE is not empty:

Let $\Sigma := \{S \sqsupset S_P, S \sqsupset S_Q, S \sqsupset S_{-P}, S \sqsupset S_{-Q}, S_P \sqsupset S_{PQ}, S_Q \sqsupset S_{PQ}, S_{-P} \sqsupset S_{-PQ}, S_Q \sqsupset S_{-PQ}, S_P \sqsupset S_{P-Q}, S_{-Q} \sqsupset S_{P-Q}, S_{-P} \sqsupset S_{-P-Q} S_{-Q} \sqsupset S_{-P-Q}, c{:}S\}$ and

ISC := $\{S_P \cap S_Q = S_{PQ}, S_P \cap S_{-Q} = S_{P-Q}, S_{-P} \cap S_Q = S_{-PQ}, S_{-P} \cap S_{-Q} = S_{-P-Q}\}$ and

SPE := $\{S_P \leftrightarrow P, S_Q \leftrightarrow Q, S_{-P} \leftrightarrow -P, S_{-Q} \leftrightarrow -Q\}$ and let the clause set be CS := $\{\{R(c)\}, \{-R(x{:}S_{PQ})\}, \{-R(x{:}S_{-PQ})\} \{-R(x{:}S_{P-Q})\} \{-R(x{:}S_{-P-Q})\}\}$

This specification is unsatisfiable, since the constant c is either in $S_{PQ}$, $S_{P-Q}$, $S_{-PQ}$ or in $S_{-P-Q}$, and hence R(c) contradicts one of the four other unit clauses.

If we remove the intersection information, this is no longer true. We have the four cases for a more exact sort information on c, but we can not conclude from $c{:}S_P$ and $c{:}S_Q$, that c is also in $S_{PQ}$. Hence the new specification is satisfiable. ■

**3.7.7 Example.** This example shows that even if the signature is regular, the clause set CS is empty, there is a transformed predicate P such that -P is also transformed and there is a ground term $t_{gr}$ of sort $S_{DP}$ that is not of sort $S_P$ or $S_{-P}$, then ISC and SPE may be contradictory.

This is also an example that the removal of a function symbol g from the signature is not conservative, even under the very restricted conditions, that the signature is regular before and after the removal, that the clause set does not contain the function symbol, and that all term declarations that contain g are function declarations.

i) Let $\Sigma := \{S_{DP} \sqsupseteq S_P, S_{DP} \sqsupseteq S_{-P}, S_{DP} \sqsupseteq S_c, g{:}S_P \to S_P, g{:}S_{-P} \to S_P, g{:}S_c \to S_{-P}, c{:}S_c, d{:}S_P, e{:}S_{-P}\}$ and let ISC be empty and let SPE := $\{S_P \leftrightarrow P, S_{-P} \leftrightarrow -P\}$
   This signature is regular and elementary, as can easily be verified.
   The specification is unsatisfiable, since c is either of sort $S_P$ or $S_{-P}$ and then g(c) is of sort $S_P$ and $S_{-P}$.
   If we remove the symbol g, then the signature remains regular, but the specification becomes satisfiable.

ii) This example shows that rule TCO3 is not conservative, if the signature is not elementary.
   Let $\Sigma := \{S_{DP} \sqsupseteq S_P, S_{DP} \sqsupseteq S_{-P}, g{:}S_P \to S_P, g{:}S_{-P} \to S_P, g(c){:}S_{-P}, c{:}S_{DP}, d{:}S_P, e{:}S_{-P}\}$ and let ISC be empty and let SPE := $\{S_P \leftrightarrow P, S_{-P} \leftrightarrow -P\}$
   This signature is regular, as can easily be verified.
   The specification is unsatisfiable, since c is either of sort $S_P$ or $S_{-P}$ and then g(c) is of sort $S_P$ and $S_{-P}$. ■

## 3.8 The Weakening Rule is Conservative.

**3.8.1 Lemma.** Rule WT is conservative.

**Proof.**

i) Let $\mathcal{S}_1$ be $\Sigma_1$-satisfiable. Then obviously $\mathcal{S}_2$ is also satisfiable, since the new clauses are instances of C.

ii) Let $\mathcal{S}_2$ be $\Sigma_2$-satisfiable. Then we can choose a $\Sigma_2$-model $\mathcal{D}_2$ of CS as follows:
   If we first ignore the conditions in ISC, then CS $\cup$ $\{P(x{:}S_P) \mid P$ is transformed$\}$ has a $\Sigma_2$-model. By Corollary I.8.7 we can choose the carrier of $\mathcal{D}_2$ as the set of ground terms

$T_{\Sigma 2,gr}$. Due to our assumption that in CS there are no positive occurrences of the predicate P, we can choose $P_{\mathcal{D}2} = S_{P,\mathcal{D}2}$ for all transformed predicates. In particular, SPE is then satisfied. Furthermore due to regularity of ISC and $\Sigma$, the relations in ISC are satisfied. Let $\sigma C$ be a ground instance of $C = \{-Q(t)\} \cup C_R\}$. If $-Q(\sigma t)$ is true, then $\sigma C$ is also true. In the other case $-Q(\sigma t)$ is false, that means $\sigma t$ is a ground term of sort $S_Q$. Hence $\sigma C$ is an instance of $\sigma_i C$ for some $\sigma_i \in \mu W_{\Sigma 2}(t \sqsubseteq S_Q)$, hence $\sigma C$ is valid.
Together we have proven that $\mathcal{D}_2$ is a $\Sigma_1$-model of $\mathcal{S}_1$. ■

### 3.9 Analysis by Cases is Conservative.

Since the rules AC1, AC2 and AC3 are equivalent to the addition of a tautology to the clause set, we have:

**3.9.1 Lemma.** The rules AC1, AC2 and AC3 are conservative. ■

**3.9.2 Lemma.** Rule AC4 is conservative.

**Proof.** The rule is conservative, since the union of the ground instances of the two new clauses is the same as the set of ground instances of the original one. ■

### 3.10 The Termination Conditions are Conservative.

It is obvious that the rules TCO1, TCO2 and TCO4 are conservative.

**3.10.1 Lemma.** Rule TCO3 is correct.

**Proof.** We have to construct a $\Sigma$-model for the clauses in SPE and ISC:

To this end we first construct a $\Sigma$-algebra as follows:

Let $S_{min,1}, \ldots, S_{min,m}$ be the minimal sorts of $\mathbf{S}_\Sigma$. Let $D := \{d_1, \ldots, d_m\}$ be the carrier of the algebra to be defined.

We define the denotation of $S_{min,i}$ as $\{d_i\}$ for every i. For the other sorts we define the denotation as the union of the denotations for the minimal sorts that it contains.

For a constant c we choose as $c_D$ an arbitrary element in D such that $c_D$ is in the denotation of sort $\mathbf{LS}_\Sigma(c)$. For a function f and elements $e_1, \ldots, e_n \in D$ we define $f(e_1, \ldots, e_n)$ as an element $e_{n+1}$ in D, such that $e_{n+1}$ is in the denotation of the sort of $f(x_1, \ldots, x_n)$, where the sort of $x_i$ is the minimal sort corresponding to $e_i$.

These definitions provide a $\Sigma$-algebra, since $\Sigma$ is regular andelementary. Obviously all intersection constraints in ISC are satisfied.

For a predicate P with $P \leftrightarrow S_P$ and $-P \leftrightarrow S_{-P}$ we have $S_{P,D} \cap S_{-P,D}$, since $S_P$ and $S_{-P}$ have no common subsort by assumption. Furthermore $S_{P,D} \cup S_{-P,D} = S_{DP,D}$, since $S_{DP,D}$ is the union of denotations of minimal subsorts of $S_{DP,D}$, and by assumption this is equivalent to the union of all minimal subsorts of $S_{P,D}$ and $S_{-P,D}$. ■

### 3.11 The Rules Dealing with Equations are Conservative.

**3.11.1** Lemma. Rule EQ1 is conservative.

**Proof.**

i) If we have a $\Sigma_1$-model $\mathcal{D}_1$ for $\mathcal{S}_1$, then the equation s = t enforces that $S_{\mathcal{D}1} \cap T_{\mathcal{D}1} \neq \emptyset$, hence we can construct a $\Sigma_2$-model $\mathcal{D}_2$ for $\mathcal{S}_2$.

ii) Trivial. ■

**3.11.2.** Lemma. Rule EQ2 is conservative.

**Proof.** Analogous to Lemma 3.2.3

**3.11.3.** Lemma. Rule EQ2 is conservative.

**Proof.** Analogous to Lemma 3.2.4.

### 3.12 Properties of SOGEN.

**3.12.1 Proposition.** The combination of the rules ISC1, ISC3 and ES1 provides a terminating algorithm that makes ISC regular. The resulting sort-structure is a semi-lattice.

**Proof.** Consider the elementary parts of the Venn-diagram. The number of this elementary parts is finite. Furthermore the three rules ISC1, ISC3 and ES1 do not increase the number of those basic parts. The number of possible relations is finite and every rule introduces new relations. We have assumed that new sorts are only introduced by ISC3, if after its addition it does not become equivalent to some other sort by ISC alone, hence the process terminates.

We show that ISC is regular and that the sort-structure is a semi-lattice:

Assume that no rule ISC1, ISC3 or ES1 is applicable.

Assume further that ISC is not regular. If there is a relation $R \cap S = T$, where T is not the greatest common subsort of R and S, we could apply rule ISC1. In the case where a derivable subsort relation does not hold, we again can apply ISC1. Hence ISC is regular.

Suppose now that the sort-structure is not a semi-lattice. This means there exist sorts R,S with a common subsort, but there is no greatest common subsort T (with $R \cap S = T$). In this case rule ISC3 is applicable.
The subsort-relation is a partial ordering, since otherwise ES1 is applicable. ■

**3.12.2 Proposition.** If we restrict term declarations to elementary ones, then the rules MS1, MS2, MS3 together with ISC1, ISC3 and ES1 provide a terminating algorithm to make the signature regular.

**Proof.** By rules ISC1, ISC3 and ES1 we can assume that the signature is a semi-lattice and that for all sorts R,S with a greatest common subsort T we have $R \cap S = T$.
In order to check that the signature is regular, it is sufficient to consider all constants and all terms $f(x_1,\ldots,x_n)$ and check that they have a least sort. Hence in rules MS2 and MS3 it is sufficient to consider only terms of this form. Using Venn-diagrams it is easy to see that rule MS2 can only introduce a finite number of new sorts. The number of new relations introduced by MS2 is hence also finite. ■

**3.12.3 Proposition.** If we restrict term declarations to elementary ones, then the rules BTi and DDi provide a terminating algorithm for sort-generation, if the rules MS1, MS2, MS3, ISC1, ISC3 and ES1 are used to manipulate the signature.

**Proof.** The rules BTi can be applied only a finite number of times, since rule BT1 decreases the number of not transformed predicates, and the rules BT2 , BT3 and BT4 (together with deletion rules) decrease the number of literals in CS. The other rules to make the signature regular also terminate, hence the application of rules terminates. ■

**3.12.4 Theorem.** Assume we restrict the signatures to elementary ones and that every rule terminates.
The rules BT1, BT2, BT3, BT4, DD1, DD2, DD3, MS1, MS2, MS3, ISC1, ISC3 and ES1 provide an algorithm for transforming a specification in a conservative way into another specification with a regular signature.

**Proof.** A combination of the lemmas and propositons above. ■

As summary of this chapter we have the theorem:

**3.12.5 Theorem.** All rules of SOGEN are conservative. ■

In the case of arbitrary signatures, the algorithm does not terminate in general. For example the procedure for making a signature regular may not terminate:

**3.12.6 Example.** The process of making an arbitrary signature regular may not terminate:

We modify the signature in III.5.1:

Let $\Sigma := \{A \sqsupset B \sqsupset D,\ A \sqsupset C \sqsupset D,\ b{:}B,\ f(b){:}B,\ f(f(x_B)){:}\ B,\ f(y_B){:}C\}$.

We assume that $B \cap C = D$.

Then the signature is not regular, since f(b) is of sort C and sort B.

Due to Theorem III.5.1, the set of most general instances of $f(y_B)$ that are of sort B is the infinite set $\{f^i(b) \mid i = 1,2,\ldots\}$

Rule MS3 then says: add f(b):C to $\Sigma$. This term declaration does not contribute to the sort of the other terms $f^i(b)$. It is easy to see, that infinitely many term declarations are to be added, namly $f^i(b)$:C for all i.

**3.12.7 Example.** If the requirement is given, that the intersection constraints should be completed during transformation, such that every sort S is the intersection of all of its supersorts, and we use only binary relations of the form $R \cap S = T$, then SOGEN may introduce an exponential number of sorts:

Consider the clause set CS consisting of the units $P_i(c)$ , $i = 1,\ldots,n$ for the same constant c. Then SOGEN introduces n sort $S_{pi}$ . The intersection $S_{p1} \cap \ldots \cap S_{pn}$ is nonempty, hence the rules of SOGEN would then generate the whole lattice of all possible intersections of sorts $S_{pi}$. That are $2^n$ -1 different sorts. ■

## 4. Examples

In this section some examples are given, which demonstrate the power of SOGEN:

### 4.1 Schubert's Steamroller [Wa85]

This example was presented in 1978 by Lenhart Schubert as a challenge to Automated Deduction Systems. There are many solutions to this problem by now (see [St86] for a comparison), the best solutions are obtained with an order-sorted formulation [Wa85, Co85].

The problem of Schubert reads as follows:
Wolves, foxes, birds, caterpillars, and snails are animals. Grains are plants. There exist wolves, foxes, birds, caterpillars, snails, and grains.
Every animal eats all plants or any smaller animal that eats some plants.
Birds are smaller than foxes which in turn are smaller than wolves. Wolves do not eat foxes or grains. Birds eat caterpillars, but no snails. Caterpillars and snails eat some plants.

The theorem to prove is:
There is a grain eating animal that is eaten by another animal.

Here is an axiomatization in first order predicate logic (without sorts):

| | |
|---|---|
| WOLF (x) | $\Rightarrow$ ANIMAL (x) ; |
| FOX (x) | $\Rightarrow$ ANIMAL (x) ; |
| BIRD (x) | $\Rightarrow$ ANIMAL (x) ; |
| CATERPILLAR (x) | $\Rightarrow$ ANIMAL (x) ; |
| SNAIL (x) | $\Rightarrow$ ANIMAL (x) ; |
| GRAIN (x) | $\Rightarrow$ PLANT (x) ; |

WOLF (LUPO) $\wedge$ FOX (FOXY) $\wedge$ BIRD (TWEEDY) $\wedge$ CATERPILLAR (MAGGIE)
$\wedge$ SNAIL (SLIMEY) $\wedge$ GRAIN (STALKY) ;

$\forall$w ANIMAL(w) $\Rightarrow$
((∀x PLANT (x) $\Rightarrow$ EATS (w x))
$\vee$ ((∀y ANIMAL(y) $\wedge$ SMALLER(y w) $\wedge$ (∃z: PLANT (z) $\wedge$ EATS(y z)))
$\Rightarrow$ EATS(w y))

| | | |
|---|---|---|
| CATERPILLAR (x) | $\wedge$ BIRD (y) | $\Rightarrow$ SMALLER (x y); |
| SNAIL (x) | $\wedge$ BIRD (y) | $\Rightarrow$ SMALLER (x y); |

| | | |
|---|---|---|
| BIRD (x) | $\wedge$ FOX (y) | $\Rightarrow$ SMALLER (x y); |
| FOX (x) | $\wedge$ WOLF (y) | $\Rightarrow$ SMALLER (x y) ; |
| WOLF (x) | $\wedge$ FOX (y) | $\Rightarrow$ $\neg$ EATS(x y); |
| WOLF (x) | $\wedge$ GRAIN (y) | $\Rightarrow$ $\neg$ EATS(x y); |
| BIRD (x) | $\wedge$ CATERPILLAR (y) | $\Rightarrow$ EATS(x y); |
| BIRD (x) | $\wedge$ SNAIL (y) | $\Rightarrow$ $\neg$ EATS(x y); |

CATERPILLAR (x) $\Rightarrow$ ($\exists$y: PLANT (y) $\wedge$ EATS (x y));
SNAIL (x) $\Rightarrow$ ($\exists$y: PLANT (y) $\wedge$ EATS (x y));

$\neg$ EATS (x x);
ANIMAL (x) $\Leftrightarrow$ $\neg$ PLANT (x) ;

Theorem:
$\exists$x,y: ANIMAL (x) $\wedge$ ANIMAL (y) $\wedge$ EATS (x y) $\wedge$ ($\forall$z GRAIN (z) $\Rightarrow$ EATS (y z))

Normalization and skolemization yields the clauses:

Ax1 -WOLF (x), ANIMAL (x) ;
Ax2 -FOX (x), ANIMAL (x) ;
Ax3 -BIRD (x), ANIMAL (x) ;
Ax4 -CATERPILLAR (x), ANIMAL (x) ;
Ax5 -SNAIL (x), ANIMAL (x) ;
Ax6 -GRAIN (x), PLANT (x) ;
Ax7 WOLF(LUPO) ;
Ax8 FOX(FOXY) ;
Ax9 BIRD (TWEEDY);
Ax10 CATERPILLAR (MAGGIE) ;
Ax11 SNAIL (SLIMEY) ;
Ax12 GRAIN (STALKY) ;
Ax13 -ANIMAL(w), -PLANT(x), EATS(w x), -ANIMAL(y), -SMALLER(y w),
-PLANT (z), -EATS(y z), EATS(w y) ;
Ax14 -CATERPILLAR (x), -BIRD (y), SMALLER(x y) ;
Ax15 -SNAIL(x), -BIRD(y), SMALLER (x y) ;
Ax16 -BIRD (x), -FOX(y), SMALLER(x y) ;
Ax17 -FOX(x), -WOLF(y), SMALLER(x y) ;
Ax18 -WOLF(x), -FOX(y), -EATS(x y) ;
Ax19 -WOLF(x), -GRAIN(y), -EATS(x y) ;

Ax20 -BIRD(x), -CATERPILLAR(y), EATS(x y) ;
Ax21 -BIRD(x), -SNAIL(y), -EATS(x y) ;
Ax22 -CATERPILLAR(x), PLANT($f_1$(x)) ;
Ax23 -CATERPILLAR(x), EATS(x $f_1$(x)) ;
Ax24 -SNAIL(x), PLANT($f_2$(x)) ;
Ax25 -SNAIL(x), EATS(x $f_2$(x)) ;
Ax26 ANIMAL(x), PLANT(x) ;
Ax27 -ANIMAL(x), -PLANT(x) ;
Ax28 -EATS (x x) ;

Th1 -ANIMAL(x), -ANIMAL(y), -EATS(x y), GRAIN($f_3$(y x)) ;
Th2 -ANIMAL(x), -ANIMAL(y), -EATS(x y), -EATS(y $f_3$(y x)) ;

The automated dedcution system MKRP [KM84] found a contradiction after 55 resolution steps. This proof uses only unit-resolution steps and was actually found by the Terminator-module [AO83].

This clause set was automatically transformed by SOGEN into its sorted version. The resulting signature and clauses are:

Sorts: TOP ⊒ S+ANIMAL, S+PLANT
S+ANIMAL ⊒ S+WOLF, S+FOX, S+BIRD, S+CATERPILLAR, S+SNAIL
S+PLANT ⊒ S+GRAIN

Constants: Lupo: S+WOLF; Foxy: S+FOX; Tweedy: S+BIRD;
Maggie: S+CATERPILLAR; Slimey: S+SNAIL; Stalky: S+GRAIN.

Functions:

| | | |
|---|---|---|
| $f_1$: | TOP | → TOP |
| | S+CATERPILLAR | → S+PLANT |
| $f_2$: | TOP | → TOP |
| | S+SNAIL | → S+PLANT |
| $f_3$: | TOP | → TOP |
| | S+ANIMAL x S+ANIMAL | → S+GRAIN. |

Clauses:

IC1 (Ax28) x:TOP -EATS(x x)
IC2 (Ax23) x:S+CATERPILLAR +EATS(x $f_1$(x))
IC3 (Ax25) x:S+SNAIL +EATS(x $f_2$(x))
IC4 (Ax14 x:S+CATERPILLAR, y:S+BIRD +SMALLER(x y)
IC5 (Ax15) x:S+SNAIL, y:S+BIRD +SMALLER(x y)
IC6 (Ax16) x:S+BIRD, y:S+FOX +SMALLER(x y)

IC7 (Ax17) x:S+FOX, S+WOLF +SMALLER(x y)
IC8 (Ax18) x:S+WOLF, x:S+FOX -EATS(x y)
IC9 (Ax19) x:S+WOLF, x:S+GRAIN -EATS(x y)
IC10 (Ax20) x:S+BIRD, x:S+CATERPILLAR +EATS(x y)
IC11 (Ax21) x:S+BIRD, x:S+SNAIL -EATS(x y)
IC12 (Ax13) x,y:S+ANIMAL z,u:S+PLANT , +EATS(x u), -SMALLER(y x),
-EATS(y u), +EATS(x y)
IC13 (Th2) x,y:S+ANIMAL -EATS(y x) , -EATS(x $f_3$(x y))

The MKRP Theorem Prover found a (unit-) refutation for this clause set after 11 steps (including 10 resolutions and one factorization). The used CPU-time for the transformation and the search for the proof in the sorted clause set was remarkably shorter than the search for the proof in the unsorted version.

We note some difficulties in getting this result with SOGEN.

1) The theorem clause Th1 was deleted by the literal reduction rule mentioned in [Sch85c]
3) The rule ISC2 is needed to identify the sorts S-PLANT ,S+ANIMAL and S-ANIMAL, S+PLANT.
4) The transformation is complete, since the preconditions of rule R-SPE&ISC are satisfied.

### 4.2 The Lion & Unicorn Examples

These examples are taken from "What is the Name of This Book" [SM78], which appears to be a goldmine for theorem proving examples. During a course on automated theorem proving in the semester ´85, the students had to translate these puzzles into first order predicate logic and to solve them with our theorem prover (Markgraf Karl Refutation Procedure) [KM84].

Two of these problems (Problem 47 + 48) read as follows:
"When Alice entered the forest of furgetfulness, she did not forget everything, only certain things. She often forgot her name, and the most likely to forget was the day of the week. Now, the lion and the unicorn were frequent visitors to this forest. These two are strange creatures. The lion lies on Mondays, Tuesdays and Wednesdays and tells the truth on the other days of the week. The unicorn, on the other hand lies on Thursdays, Fridays and Saturdays, but tells the truth on the other days of the week."

Problem 47: One day Alice met the lion and the unicorn resting under a tree. They made the following statements:

Lion: Yesterday was one of my lying days.

Unicorn: Yesterday was one of my lying days.

From these statements, Alice who was a bright girl, was able to deduce the day of the week. What was it?

Problem 48: On another occasion Alice met the Lion alone. He made the following two statements:

1) I lied yesterday
2) I will lie again tomorrow.

What day of the week was it?

We use the predicates MO(x), TU(x), ... , SO(x) for saying that x is a Monday, Tuesday etc. Furthermore we need the binary predicate MEMB, indicating set membership and a 3-ary predicate LA. LA(x y z) is true if x says at day y that he lies at day z; LDAYS(x) denotes the set of lying days of x. The remaining symbols are self explaining. One-character symbols like u,x,y,z are regarded as universally quantified variables.

Axiomatization of the days of the week:

$$MO(x) \Leftrightarrow \neg(TU(x) \vee WE(x) \vee TH(x) \vee FR(x) \vee SA(x) \vee SU(x))$$
$$TU(x) \Leftrightarrow \neg(WE(x) \vee TH(x) \vee FR(x) \vee SA(x) \vee SU(x) \vee MO(x))$$
$$WE(x) \Leftrightarrow \neg(TH(x) \vee FR(x) \vee SA(x) \vee SU(x) \vee MO(x) \vee TU(x))$$
$$TH(x) \Leftrightarrow \neg(FR(x) \vee SA(x) \vee SU(x) \vee MO(x) \vee TU(x) \vee WE(x))$$
$$FR(x) \Leftrightarrow \neg(SA(x) \vee SU(x) \vee MO(x) \vee TU(x) \vee WE(x) \vee TH(x))$$
$$SA(x) \Leftrightarrow \neg(SU(x) \vee MO(x) \vee TU(x) \vee WE(x) \vee TH(x) \vee FR(x))$$
$$SU(x) \Leftrightarrow \neg(MO(x) \vee TU(x) \vee WE(x) \vee TH(x) \vee FR(x) \vee SA(x))$$

Axiomatization of the function yesterday:

$$MO(yesterday(x)) \Leftrightarrow TU(x)$$
$$TU(yesterday(x)) \Leftrightarrow WE(x)$$
$$WE(yesterday(x)) \Leftrightarrow TH(x)$$
$$TH(yesterday(x)) \Leftrightarrow FR(x)$$
$$FR(yesterday(x)) \Leftrightarrow SA(x)$$
$$SA(yesterday(x)) \Leftrightarrow SU(x)$$
$$SU(yesterday(x)) \Leftrightarrow MO(x)$$

Axiomatization of the function two-after:

| | | |
|---|---|---|
| MO(two-after(x)) | $\Leftrightarrow$ | FR(x) |
| TU(two-after(x)) | $\Leftrightarrow$ | SA(x) |
| WE(two-after(x)) | $\Leftrightarrow$ | SU(x) |
| TH(two-after(x)) | $\Leftrightarrow$ | MO(x) |
| FR(two-after(x)) | $\Leftrightarrow$ | TU(x) |
| SA(two-after(x)) | $\Leftrightarrow$ | WE(x) |
| SU(two-after(x)) | $\Leftrightarrow$ | TH(x) |

Axiomatization of the function LDAYS:

| | | |
|---|---|---|
| MEMB(x LDAYS(lion)) | $\Leftrightarrow$ | MO(x) $\vee$ TU(x) $\vee$ WE(x) |
| MEMB(x LDAYS(unicorn)) | $\Leftrightarrow$ | TH(x) $\vee$ FR(x) $\vee$ SA(x) |

Axiomatization of the predicate LA:

| | | |
|---|---|---|
| $\neg$MEMB(x LDAYS(u)) $\wedge$ LA(u x y) | $\Rightarrow$ | MEMB(y LDAYS(u)) |
| $\neg$MEMB(x LDAYS(u)) $\wedge$ $\neg$LA(u x y) | $\Rightarrow$ | $\neg$MEMB(y LDAYS(u)) |
| MEMB(x LDAYS(u)) $\wedge$ LA(u x y) | $\Rightarrow$ | $\neg$MEMB(y LDAYS(u)) |
| MEMB(x LDAYS(u)) $\wedge$ $\neg$LA(u x y) | $\Rightarrow$ | MEMB(y LDAYS(u)) |

Theorem of Problem 47:

$\exists$x LA(lion x yesterday(x)) $\wedge$ LA(unicorn x yesterday(x))

Theorem of Problem 48:

$\exists$x LA(lion x yesterday(x)) $\wedge$ LA(lion x two-after(x))

The MKRP automated deduction system found a proof for the unsorted version of problem 47 after 183 resolution steps, among them 81 unnecessary steps, hence the final proof was 102 steps long. This proof contains plenty trivial steps corresponding to common sense reasoning (like: if today is Monday, it is not Tuesday etc.).

Later the sort structure and the signature of the problem 47 was generated automatically by SOGEN.

The sort structure and the signature contain all the relevant information about the relationship of unary predicates (like our days) and the domain-rangesort relation of functions. The sort structure of the subsorts of DAYS in our example is equivalent to the lattice of subsets of {Mo, Tu, We, Th, Fr, Sa, Su} without the empty set, ordered by the subset order. Hence there are 127 ($=2^7-1$) sorts. The functions "yesterday" and "two-after" are polymorphic functions with 127 domain-sort relations. For example: yesterday ({MO, WE}) = {SU, TU}.

The unification algorithm exploits this information and produces only well-sorted unifiers. For example the unifier of x:SO+TU and yesterday(y:MO+TU) is $\{x \leftarrow \text{yesterday}(y_1\text{: MO})\ ;\ y \leftarrow y_1\text{:MO}\}$.

The MKRP theorem-proving system [KM84] has proved the theorem of both problems in the sorted version immediately without any unnecessary steps. The length of the proof of problem 47 is 6, whereas the length of the proof of problem 48 is 4. As the protocol shows, the final substitution into the theorem clause (Problem 48) was $\{x \leftarrow y\text{:MO}\}$. Thus the ATP has found the answer, 'monday', in a very straightforward and humanlike way. A proof protocol for problem 47 can be found in [Sch85]. We give a proof protocol for Problem 48:

| | | |
|---|---|---|
| C1 | All x:MO | MEMB (x LDAYS(lion)) |
| C2 | All x:TU | MEMB (x LDAYS(lion)) |
| C3 | All x:WE | MEMB (x LDAYS(lion)) |

| | | |
|---|---|---|
| C4 | All x,y:DAYS z:Animal | MEMB(y LDAYS(z)) MEMB (x LDAYS(z)) -LA(z y x) |
| C5 | All x,y:DAYS z:Animal | MEMB(y LDAYS(z)) -MEMB(x LDAYS(z)) LA(z y x) |
| C6 | All x,y:Days z:Animal | -MEMB(y LDAYS(z)) MEMB(x LDAYS(z)) LA(z y x) |
| C7 | All x,y:Days z:Animal | -MEMB(y LDAYS(z)) -MEMB(x LDAYS(z)) -LA(z y x) |
| C8 | All x:TH+FR+SA+SU | -MEMB(x LDAYS(lion)) |
| Th1 | All x:Days | -LA(lion x yesterday(x)) -LA(lion x two-after(x)) |

<u>Proof:</u>

| | | |
|---|---|---|
| C1,1 & C6,1 | → R1: | All x:MO y:TH+FR+SA+SU MEMB(y LDAYS(lion)) LA(lion x y) |
| R1,2 & C8,1 | → R2: | All x:MO y:TH+FR+SA+SU LA(lion x y) |
| R2,1 & Th,2 | → R3: | All x:MO -LA(lion x yesterday(x)) |
| R3,1 & R2,1 | → R6: | ❑ |

### 5. Extension of SOGEN to Well-Formed Formulae.

In this paragraph some special rules for introducing sorts in wff's are given. The logical basis for this paragraph is paragraph II.12. The mixed application of sort-generation, simplification, normalization and skolemization has the advantage, that the generated clause set is simpler and that more unary predicates can be transformed into sorts. We introduce the rules in an informal way. We give no rules for simplification, normalization or skolemization. All proofs that these rules are sound and complete, are omitted, since they are either straightforward or similar to proofs in

**Remark.** We assume, that the wff W is the input to a theorem prover, which tests W for satisfiability or unsatisfiability. If $W = W_1 \wedge \ldots \wedge W_n$, and some $W_i$ is a clause, then the rules of SOGEN can be applied to $W_i$.

**5.1 Defintion.** Transformation rules for Wff´s:

We use the set ISC and SPE with the same meaning as in SOGEN.

i) If $P \leftrightarrow S_P$ and $S_P \cap S_x = S_0$
then replace $(\forall x{:}S_x \neg P(x) \vee A)$ by $(\forall x{:}S_0 \text{ FALSE} \vee A)$

ii) If $P \leftrightarrow S_P$ and $S_P \cap S_x = S_0$
then replace $(\exists x{:}S_x \ P(x) \wedge A)$ by $(\exists x{:}S_0 \text{ TRUE} \wedge A)$

iii) If $P \leftrightarrow S_P$ and $S_P \in S_\Sigma(t)$
then replace P(t) by TRUE.

iv) If $P \leftrightarrow S_P$ and $S_P \in S_\Sigma(t)$
then replace $\neg P(t)$ by FALSE

v) If $P \leftrightarrow S_P$ and $S_0 \sqsupseteq S_P$
then replace $(\forall x{:}S_0 \neg P(x) \wedge A)$ by FALSE.

vi) If $P \leftrightarrow S_P$ and $S_0 \sqsupseteq S_P$
then replace $(\exists x{:}S_0 \ P(x) \vee A)$ by TRUE.

vii) replace $(\forall x{:}S \ A \wedge B)$ by $(\forall x{:}S \ A) \wedge (\forall x{:}S \ B)$

viii) replace $(\exists x{:}S \ A \vee B)$ by $(\exists x{:}S \ A) \vee (\exists x{:}S \ B)$

**5.2 Example.** " Andrew´s Little Challenge" [EW83].

The formula W is :

$\{(\forall x_1 \ Q(x_1)) \Leftrightarrow (\exists x_2 \ Q(x_2))\} \Leftrightarrow \{\exists x_3 \ (\forall x_4 \ Q(x_3) \Leftrightarrow Q(x_4))\}$

1) We use Rule AC1 for Q, that means:

either $-Q \leftrightarrow S_{-Q}$ and $S_{-Q} = \text{TOP}$ or $Q \leftrightarrow S_Q$

Case 1. $-Q \leftrightarrow S_{-Q}$ and $S_{-Q} = \text{TOP}$

Then $W = \{\text{FALSE} \Leftrightarrow \text{FALSE}\} \Leftrightarrow \{\exists x_3 \ (\forall x_4 \ \text{FALSE} \Leftrightarrow \text{FALSE})\}$ by the rules of 5.1.

This formula evaluates to TRUE by simplification rules.

Case 2. $Q \leftrightarrow S_Q$:

Then $W = \{(\forall x_1 \ Q(x_1)) \Leftrightarrow \text{TRUE}\} \Leftrightarrow \{\exists x_3 \ (\forall x_4 \ Q(x_3) \Leftrightarrow Q(x_4))\}$ which simplifies to $(\forall x_1 \ Q(x_1)) \Leftrightarrow \{\exists x_3 \ (\forall x_4 \ Q(x_3) \Leftrightarrow Q(x_4))\}$

Case 2.1 $S_Q$ = TOP.

Then W = TRUE $\Leftrightarrow$ $\{\exists x_3\ (\forall x_4\ \text{TRUE} \Leftrightarrow \text{TRUE})\}$ , which evaluates to TRUE.

Case 2.2 $-Q \leftrightarrow S_{-Q}$

Then we can make the following transformations:

$$\text{FALSE} \Leftrightarrow \{\exists x_3\ (\forall x_4\ Q(x_3) \Leftrightarrow Q(x_4))\} \quad \rightarrow$$
$$\neg\{\exists x_3\ (\forall x_4\ Q(x_3) \Leftrightarrow Q(x_4))\} \quad \rightarrow$$
$$\neg\{\exists x_3\ (\forall x_4\ (\neg Q(x_3) \vee Q(x_4)) \wedge (Q(x_3) \vee \neg Q(x_4)))\} \quad \rightarrow$$
$$\neg\{\exists x_3\ (\forall x_4\ (\neg Q(x_3) \vee Q(x_4)) \wedge (\forall x_5\ Q(x_3) \vee \neg Q(x_5)))\} \quad \rightarrow$$
$$\neg\{\exists x_3\ (\forall x_4{:}S_{-Q}\ \neg Q(x_3)) \wedge (\forall x_5{:}S_Q\ Q(x_3))\} \quad \rightarrow$$
$$\neg\{\exists x_3\ \neg Q(x_3)) \wedge Q(x_3)\} \quad \rightarrow$$
$$\text{FALSE.}$$

**5.3 Example.** We demonstrate, how a formula that occurs in the first order formulation of "Schuberts Steamroller" [Wa85, St86] is normalized and skolemized using different methods: We have the three axioms $G(G_0)$; $A(A_0)$ and $\forall x,y\ \neg A(x) \vee \neg E(x,y) \vee (\exists z\ G(z) \wedge \neg E(y,z))$

i) Sort generation after normalization.

We obtain the following clauses after normalization:

$G(G_0)$;

$A(A_0)$;

$\forall x,y\ -A(x) \vee -E(x,y) \vee G(f(x,y))$;

$\forall x,y\ -A(x) \vee -E(x,y) \vee -E(y, f(x,y))$;

Sort generation yields:

$S_A \leftrightarrow A$, $A_0 : A$, $S_A \sqsubseteq \text{TOP}$, $S_G \sqsubseteq \text{TOP}$.

The clauses are:

$G(G_0)$;

$\forall x{:}S_A$ , y:TOP $-E(x,y) \vee G(f(x,y))$;

$\forall x{:}S_A$ , y:TOP $-E(x,y) \vee -E(y, f(x,y))$ ❑

ii) Sort generation during normalization. We get:

$S_G \leftrightarrow G$, $S_A \leftrightarrow A$, $A_0$: $S_A$ ; $G_0$: $S_G$ ; $S_A \sqsubseteq$ TOP; $S_G \sqsubseteq$ TOP; and the axiom

$\forall x{:}S_A$ , y:TOP $-E(x,y) \vee (\exists z{:}S_G\ -E(y, z))$

Skolemization then gives a function f: $S_A$ xTOP $\rightarrow S_G$ and the clause

$\forall x{:}S_A$ , y:TOP $-E(x,y) \vee -E(y, f(x, y))$ ❑

The difference between the two methods is that in i) the clause $\forall x{:}S_A$ ,y:TOP $-E(x,y) \vee G(f(x,y))$ contains the literal $-E(x,y)$, whereas in ii) this literal is avoided.

## 6. Conclusion of part VI

The main results of this part are:

i) An algorithm SOGEN is described, which transforms unsorted clause sets (respectively wffs) into a sorted version. Furthermore a proof is given, that this algorithm preserves (un)satisfiability.

ii) Conditions are given for the completeness of the transformation.

iii) A polynomial algorithm for sets of Horn-clauses is described.

It is not possible to give a sufficient and necessary condition for a clause set to be transformable into a sorted version. The reason is, that some deductions may be necessary for such a transformation.

The algorithm SOGEN was implemented at Kaiserslautern as a preprocessor for the MKRP Automated Theorem Prover [KM84]. It has shown remarkable improvements searching for a proof in several test runs.

Since this algorithm is in some sense deterministic (no search) the cpu-time consumed by SOGEN is negligible in most examples, but serious problems arise in cases, where the number of sorts is very large. The sort structure constructed in example 4.2 is isomorphic to the lattice of subsets of a set with 7 elements (i.e. 127 sorts) (see Example 3.12.7). I believe that a modified implementation of sorts (computing sorts and their relations if needed) could handle far bigger sort structures of this type.

In the case that SOGEN fails, the cpu-time consumed is not totally wasted, since the validity of the toplevel reductions (such as tautology deletion and replacement resolution) do not depend on the success of SOGEN.

## References.


Ack54 Ackermann, W., Solvable Cases of the Decision Problem, North-Holland, Amsterdam, (1954)

Ai86 Aït-Kaci, H., Nasr, R., Logic and Inheritance, Proc. 13th POPL, pp.219-228, (1985)

An70 Anderson, R., Completeness results for E-resolution. Proc. Spring Joint Conference, pp. 653-656, 1970

AB70 Anderson, R., Bledsoe, W.W., A linear format for resolution with merging and a new technique for establishing completeness, JACM 17, pp. 525-534, (1970)

And81 Andrews, P., Theorem Proving via General Matings, JACM 28, 2, pp. 193-214, (1981)

AO83 Antoniou, G., Ohlbach, H. J., Terminator, Proc. of the 8th IJCAI, Karlsruhe, pp. 916-919, (1983)

Ba76 Baxter, L.D., The Complexitiy of Unification, Ph. D. thesis, University of Waterloo, (1976)

Ba86 Baader, F., The Theory of Idempotent Semigroups is of Unification Type Zero., JAR 2, 3, pp. 283-286, (1986)

Bib81a Bibel, W., Automated Theorem Proving, Vieweg Verlag, Wiesbaden, (1981)

Bib81b Bibel, W., Matings in Matrices, GWAI 81, Informatik Fachberichte 47, Springer Verlag, pp 171-187, (1981)

BC83 M. Bidoit, J. Corbin: 'A Rehabilitation of Robinson´s Unification Algorithm', Inform. Processing 83, ed. R.E.A. Pavon, North Holland 1983, pp. 909-914

Bl83 Bläsius, K. H., Equality Reasoning in Clause Graphs. Proc. IJCAI-83, pp. 936-939, (1983)

Bl87 Bläsius, K. H., Equality Reasoning Based on Graphs. Tech Report SR-87-01, (1987)

Bra75 Brand, D., Proving theorems with the modification method, SIAM J. of Computing, 4, pp. 412-430, (1975)

BB86 Bernot, G., Bidoit, M., Choppy, C., Abstract Data Types with Exception Handling: An initial approach based on a distinction between exceptions and errors, Technical report N° 251, Universite de Paris-Sud, Centre d`Orsay, France

Bi35 Birkhoff, G., On the Structure of Abstract Algebras, Proceedings of the Cambridge Philosophical Society 31, pp. 433-454, (1935)

Bö85 Börger, E., 'Berechenbarkeit, Komplexität, Logik', Friedr. Vieweg & Sohn, (1985) BR87 Burmeister, P., Reichel, H., A Model Theoretic Approach to Partial Algebras, Mathematical research, Band 32, Akademie-Verlag, Berlin, (to appear)

BS81 Burris, S., Sankappanavar, H.P., 'A course in universal algebra', Springer-Verlag, (1981)

BS85 Book, R., Siekmann, J.H., On the Unification Hierarchy, Proc. of GWAI ´85, Springer-Verlag, pp. 111-117, (1985)

BS86 Bahlke, R., Snelting, G., The PSG System: From formal language definitions to interactive programming environment. ACM TOPLAS 8 (4), pp. 547-576, (1986)

Bu87 Buchberger, B., History and Basic Features of the Critical-Pair/Completion Procedure., JSC 3,1, pp. 3-38, (1987)

BHS87 Bürckert, H.-J., Herold A., Schmidt-Schauß, M., On Equational Theories, Unification and Decidability, Proc. RTA, LNCS 256, pp. 204-215 also (to appear in JSC, special issue on unification)

Bü84 Bürckert, H.-J., 'Unification Algorithms', PIPE-Working Paper, Universität Kaiserslautern, 1984

Bü85 Bürckert, H.-J., 'Extending the WARREN Abstract Machine to Many-Sorted Prolog', Tech. report SEKI-85-07, Universität Kaiserslautern, (1985)

Bü86 Bürckert, H.-J., Some relationships between Unification, Restricted Unification and Matching, in Proc. of $8^{th}$ CADE, Springer, LNCS 230, pp. 514-524, (1986) also: SEKI-Report, SR-86-05, Universität Kaiserslautern, 1986

Büt86a Büttner, W., Unification in the Datastructure Multiset, JAR, vol.2, pp. 75-88, (1986)

Büt86b Büttner, W., Unification in the Datastructure Set, Proc 8th CADE, Springer, LNCS 230, pp. 470-488

CD85 Cunningham, R.J., Dick, A.J.J, Rewrite Systems on a Lattice of Types. Acta Informatica 22, pp. 149-169, (1985)

CT82 Clark K. L., Tärnlund S.-A., Logic Programming, Academic Press, (1982)

CL73 Chang, C., Lee, R. C., Symbolic Logic and Mechanical Theorem Proving, Academic Press, (1973)

Co83a Cohn, A.G., Improving the Expressiveness of Many-Sorted Logic, AAAI-83, Washington, pp 84-87, (1983).

Co83b Cohn, A.,G., 'Mechanising a Particularly Expressive Many Sorted Logic', PhD Thesis, University of Essex, (1983)

Co85 Cohn, A.G., On the solution of Schubert`s steamroller in many sorted logic, Proc. of 9th IJCAI, Los Angeles, California, pp. 1169-1174, (1985)

Co86 Cohn, A.,G., Many-sorted Logic = Unsorted Logic + Control?, in M. Bramer (ed.) Expert System 86, Cambridge University Press, pp. 184-194, (1986) Co87 Cohn, A.G., A More Expressive Formulation of Many-Sorted Logic., JAR 3,2, pp. 113-200, (1987)

CM81 Clocksin, W.F., Mellish, C.S., Programming in Prolog, Springer Verlag, Berlin (1981)

De87 Dershowitz, N., Termination of Rewriting, JSC 3, 1, pp. 69-115

DM79 Dershowitz, N., Manna Z., Proving Termination with Multiset Orderings, CACM 22, pp. 465-476, (1979)
Also in Proc of ICALP 79, pp. 188-202, (1979)

Di79 Digricoli, V.J., Resolution by Unification and Equality. Proc. 4th CADE, Texas, pp. 43-52, (1979)

Di81 Digricoli, V.J. The Efficacy of RUE Resolution, Experimental Results and Heuristic Theory., Proc. IJCAI-81, Vancouver, pp. 539-547,(1981)

Di85 Digricoli, V.J., The Management of Heuristic Search in Boolean Experiments with RUE Resolution., Proc IJCAI-85, Los Angeles, pp. 1154-1161(1985)

DST80 Downey, P.J., Sethi, R., Tarjan, R.E., Variations on the common subexpression problem, JACM 27,4, pp. 758-771, (1980)

EF78 Ebbinghaus, H.-D., Flum, J., Thomas, W., Einführung in die mathematische Logik. Wissenschaftliche Buchgesellschaft. Darmstadt, (1978)

EM85 Ehrig, H., Mahr, B., Fundamentals of algebraic specification 1, EATCS 6, Springer-Verlag, (1985)

EW83 Eisinger N., Weigele M., A Technical Note on Splitting and Clausel Form Algorithms. Proc. GWAI-83, Springer Fachberichte, pp. 225-232, (1983)

Fa79 Fay, M., 'First Order Unification in an Equational Theory', Proc. 4th CADE, Texas, pp. 161-167, (1979)

Fa84 Fages, F., Associative-Commutative Unification, Proc 7th CADE, LNCS 170, pp. 194-208, (1984)

Fo85 Fortenbacher, A., An Algebraic Aproach to Unification under Associativity and Commutativity, Proc. RTA, LNCS 202, pp. 381-397, (1979)

FGJM85 Futatsugi, K., Goguen, J.A., Jouannaud, J.-P. and Meseguer, J., Principles of OBJ2, Proc. 1985 POPL , pp. 52-66, ACM, (1985)

FH83 Fages F., Huet G., Complete sets of unifiers and matchers in equational theories. Proc. CAAP-83, LNCS 159, (1983) .
Also in Theoretical Computer Science 43, pp. 189-200, (1986)

Ga86 Gallier, J. H., Logic for Computer Science, Harper & Row, (1986)

GS87 Gallier, J.H., Snyder, W., A general complete E-unification procedure, Proc. of RTA , LNCS 256, pp. 216-227, (1987)

GRS87 Gallier, J.H., Raatz, S., Snyder, W., Theorem Proving Using Rigid E-unification: Equational Matings, CREAS workshop, (1987), (to appear)

Gi58 Gilmore, P. C., An Addition to the Logic of Many-Sorted Theories, Compositio Math. 13, pp. 277-281, (1958)

Go83 Gogolla, M., Algebraic Specification with Partially Ordered Sorts and Declarations. Techn. Report, Institut für Informatik, Dortmund (1983)

Go86 Gogolla, M., Über partiell geordnete Sortenmengen und deren Anwendung zur Fehlerbehandlung in abstrakten Datentypen, Dissertation, Informatik, Universität Braunschweig, West Germany, (1986)

GDL84 Gogolla, M., Drosten K., Lipeck U., Ehrich H.-D., Algebraic and operational semantics of specifications allowing exceptions and errors., Theoretical Computer Science 34, pp. 289-313, (1984)

Gg78 Goguen, J.A., Order-sorted algebra, Technical report, UCLA computer science department, semantics and theorx of computation report № 14, (1978)

GJM85 Goguen, J.A. Jouannaud, J.-P., Meseguer, J. Operational Semantics of Order-sorted algebra, Proc. 12 ICALP, LNCS 194, pp. 221-231 Springer Verlag, (1985)

GM81 Goguen, J.A. Meseguer, J., Completeness of Many-sorted Equational Logic, Sigplan Notices 16,7, pp. 24-32 (1981)
latest version in Houston Journal of Mathematics 11,3 pp. 307-334 (1985).

GM84 Goguen, J.A. Meseguer, J.
Equalities, Types, Modules and Generics for Logic Programming, Journal of Logic Programming 1, pp. 179-210 (1984).

GM85a Goguen, J.A. Meseguer, J.
Order-Sorted Algebra I. Partial and Overloaded Operators, Error and Inheritance., SRI Report (1985).

GM85b Goguen, J.A. Meseguer, J. EQLOG: equality, types, and generic moduls for logic programming, to appear in Functional and Logic Programming, ed. DeGroot and Lindstrom, Prentice-Hall, (1985)

Gr79 Grätzer, G. Universal Algebra, Springer-Verlag, (1979)

Hay71 Hayes, P. A Logic of Actions.', Machine Intelligence 6, Metamathematics Unit, University of Edinburgh. , pp. 495-520, (1971)

Hai57 Hailperin, T., A theory of restricted quantification, JSL 22, pp. 19-35, (1957)

Her30 Herbrand, J., Recherches sur la théory de la démonstration, Travaux de la Soc. des Sciences et des Lettre de Varsovie, Nr. 33, 128, (1930)

Her71 Herbrand, J., Sur la Théorie de la Démonstration. In Logical Writings, W. Goldfarb ed., Cambridge, (1971)

He83 Herold, A., Some Basic Notions of First Order Unification Theory, Univ. Karlsruhe, Interner Report, (1983)

He86 Herold, A., 'Combination of Unification Algorithms', Proc. 8th CADE, ed. J. Siekmann , Springer-Verlag, LNCS 230, pp. 450-469, (1986) also: MEMO-SEKI 86-VIII-KL, Universität Kaiserslautern, 1985

HS85 Herold, A., Siekmann, J., Unification in Abelian Semigroups, Tech. report, Universität Kaiserslautern, Memo SEKI-85-III, (1985)

HS87 Herold, A., Siekmann, J., Unification in Abelian Semigroups, JAR 3 (3), pp. 247-283, (1987)

Hen72 Henschen, L.J.,'N-sorted Logic for Automated Theorem Proving in Higher-Order Logic.', Proc. ACM Conference, Boston (1972)

HLS72 Hindley, J.R., Lercher, B., Seldin, J.P., Introduction to Combinatory Logic, London Math. Soc. Lecture Note Series 7, Cambridge Univ. Press, (1972)

HO80 Huet, G.,Oppen, D.C.
Equations and Rewrite Rules, SRI Technical Report CSL-111, (1980)
also in :Formal Languages: Perspectives and open problems, R. Book.(ed), Academic Press, (1982)

HoUl79 Hopcraft J.E., Ullmann, J.D., Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, (1979)

HR86 Hsiang, J., Rusinowitch, M., A new method for establishing refutation completeness in Theorem Proving, Proc of the $8^{th}$ CADE, LNCS 230, pp. 141-152, (1986)

HoUl79 Hopcraft, J., Ullman, J., Introduction to Automata Theory, Languages and Computation, Addison -Wesley, (1979)

Ho76 Howie, J.M., An Introduction to Semigroup Theory, Academic Press, (1976)

Hu76 Huet, G. Résolution d`équations dans des langages d´ordre 1,2,...,ω,
Thèse d`État, Univ. de Paris, VII, (1976)

Hu80 Huet, G. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems, JACM 27, 4 , pp. 797-821, (1980)

Hul80 Hullot, J.-M., Canonical Forms and Unification, Proc. 5th CADE, LNCS 87, pp. 318-334, (1980)

IS85 Irani, K.B., Shin, D.G., A Many-Sorted Resolution Based on an Extension of a First-Order Language, Proc. 9th IJCAI, pp. 1175-1177, (1985)

JKK 83 Jouannaud, J.-P., Kirchner, C., Kirchner, H., 'Incremental Construction of Unification Algorithms in Equational Theories', Proc. of 10th ICALP ed J.Diaz, Springer-Verlag, LNCS 154, pp. 361-373 , (1983 )
also: Université de Nancy, Informatique, 82-R-047, 1982

JK 84 Jouannaud, J.-P. and Kirchner, H., 'Completion of a Set of Rules Modulo a Set of Equations', Proc. of 11th ACM Conference on Principles of Programming Languages, Salt Lake City, (1984)
also: Université de Nancy, Informatique, 84-R-046, (1984)

SIAM J. of Computing, Vol. 15,. 4, (1986)

HKi85 Kirchner, H., Preuves par complétion dans les variétés d`algèbres, Thèse d`Etat, University de Nancy I, (1985)

HKi87 Kirchner, H., Schematization of infinite sets of rewrite rules. Application to the divergence of completion processes., Proc. of RTA, LNCS 256, pp. 180-191, (1987)

CKi84 Kirchner, C., A New Equational Unification Method: A generalization of Martelli- Montanari´s Algorithm. 7th CADE, LNCS 170, pp. 224-247, (1984)

CKi85 Kirchner, C., Méthodes et outils de conception systématique d´algorithmes d´unification dans les théories équationelles, Thèse d´état de l´Univerité de Nancy I, (1985),

CKi 86 C.Kirchner: 'Computing Unification Algorithms', Conf. on Logic in Computer Science, pp. 206-216, (1986)

CKi87 Kirchner, C., Methods and Tools for Equational Unification, CNRS technical report Nr. 87-R-008, University of Nancy I, (1987)

KB70 Knuth,D.E., Bendix, P.B., 'Simple Word Problems in Universal Algebras', in: Computational Problems in Abstract Algebra, J.Leech ed. , Pergamon Press, Oxford, (1970)

KM84 Raph., Karl Mark G., The Markgraf Karl Refutation Procedure, SEKI-report MK-84-01, Universität Kaiserslautern (1984)

Ko75 Kowalsi, R., A proof procedure using connection graphs, JACM 22, 4, (1975)

Ko79a Kowalski, R., Logic for problem solving, North-Holland, (1979)

Ko79b Kowalski, R., Algorithm = Logic + Control. CACM 22, 7, pp. 424-436, (1979)

Koz76 Kozen, D., Complexity of Finitely Presented Algebras, Technical Report TR 76-294, Dept. of Comp. Science, Cornell University, Ithaca, NY, (1976)

Koz77 Kozen, D., Complexity of Finitely Presented Algebras, 9th STOC Symposium, Boulder Colorado, pp. 164-177, (1977)

LS76 Livesey, M., Siekmann, J., Unification of AC-terms (bags) and ACI-terms (sets), Tech. report 3-76, Universität Karlsruhe, (1976)

Lo78 Loveland, D., Automated Theorem Prroving: A Logical Basis, North-Holland, (1978)

Ll84 Lloyd, J.W., Foundations of Logic Programming, Springer-Verlag, (1984)

MG85a Meseguer, J., Goguen, J. A., Initiality, Induction and Computability. In M.Nivat and J. Reynolds (eds.), Algebraic Methods in Semantics, pp. 459-540, Cambridge University Press, (1985)

MG85a Meseguer, J., Goguen, J. A., Deduction with Many-Sorted Rewrite, Techn. report CSLI-85-42, (1985) MGS87 Meseguer, J., Goguen, J.A., Smolka, G., Order-sorted Unification, CREAS workshop, (1987) , to appear

Ml78 Milner, R., A Theory of Type Polymorphism in Programming, JCSS 17, pp. 348-375, (1978)

Ml84 Milner, R. A Proposal for Standard ML, 1984 ACM Symposium on LISP and Funcitonal Programming, Austin, Texas, pp.184-197, (1984)

MM82 Martelli, A., and Montanari, U., An Efficient Unification Algorithm, ACM Trans. Programming Languages and Systems 4, 2, pp. 258-282, (1982)

MMR 86 Martelli, A., Moiso, C. and Rossi, G.F., 'An Algorithm for Unification in Equational Theories', Proc. of Symp. on Logic Programming II, pp. 180-186, (1986)

Mc 76 McNulty, G., 'Undecidable Properties of Finite Sets of Equations', JSL 41, 3, pp. 589-604, (1976)

Mo69 Morris, J., Extension of resolution to include equality, Proc. first IJCAI-69, Washington D.C., pp. 287-294, (1969)

MO85 Mycroft, A., O`Keefe, R.A., A Polymorphic Type System for Prolog, Artificial Intelligence 23, 3, pp. 295-307, (1984)

NO79 Nelson, G., Oppen, D.C., Simplification by Cooperating Decision Procedures, ACM TOPLAS 1,2, pp. 245-257, (1979)

NO80 Nelson, G., Oppen, D.C., 'Fast Decision Procedures Based on Congruence Closure', JACM, 27, 2, pp. 356-364, (1980)

NR85 Nivat, M., Reynolds, J. C., Algebraic Methods in Semantics, Cambridge University Press, (1985)

NRS87 Nutt,W., Réty, P., Smolka, G., Basic narrowing revisited, Technical report SR-87-07, Universität Kaiserslautern, West Germany, (1987)

Ob62 Oberschelp A.,
Untersuchungen zur mehrsortigen Quantorenlogik, (in German), Mathematische Annalen 145, pp. 297-333, (1962)

Oh85 Ohlbach, H.J., 'Theory Unifcation in Abstract Clause Graphs', Technical report MEMO SEKI-85-I-KL, Universität Kaiserslautern, (1978)

Oh87 Ohlbach, H.J., Link Inheritance in Abstract Clause Graphs, JAR 3,1, pp. 1-34, (1987)

Pa86 Padawitz, P., Foundations of Specification & Programming with Horn Clauses, Fakultät für Mathematik und Informatik, Universität Passau, West Germany, draft, (1986)

Pe83 Peterson, G.E., A technique for establishing completeness results in theorem proving with equality, Siam J. Comput., 12,1, pp. 82-100, (1983)

PW78 Paterson, M.S., Wegman, M.N., Linear Unification, Journal of Computer and System Sciences 16, 158-167, (1978) Plo72 Plotkin, G., Building in equational theories, Machine Intelligence 7, pp. 73-90, (1972)

Re87 Réty,P., Improving basic narrowing techniques, Proc. 2nd RTA, Bordeaux France, LNCS 256, pp. 216-227, (1987)

Ri75 Richter, M.M., A Note on Paramodulation and the Functional Reflexive Axioms. Institut für angewandte Mathematik und Informatik, TH Aachen, (1975)

Ri78 Richter, M.M., Logikkalküle., Teubner Verlag, Stuttgart, (1978)

Ro65 Robinson. J.A. A machine-Oriented Logic Based on the Resolution Principle. JACM 12,1 pp. 23-41, (1965)

RW69 Robinson, G., Wos, L., Paramodulation and theorem proving in first order theories with equality, Machine Intelligence 4, pp. 135-150, (1969)

Sch38 Schmidt, A., Über deduktive Theorien mit mehreren Sorten von Grunddingen, Math. Annalen 115, pp. 485-506, (1938)

Sch51 Schmidt, A., Die Zulässigkeit der Behandlung mehrsortiger Theorien mittels der üblichen einsortigen Prädikatenlogik, Math Annalen 123, pp. 187-200, (1951)

Sch85a Schmidt-Schauss, M., A many-sorted calculus with polymorphic functions based on resolution and paramodulation. Proc. of the 9th IJCAI, Los Angeles, ed. A.Joshi, pp. 1162-1168, (1985)

Sch85b Schmidt-Schauss, M. A many-sorted calculus with polymorphic functions based on resolution and paramodulation. SEKI-report MK-85-2, university of Kaiserslautern (1985)

Sch85c Schmidt-Schauss, M., Mechanical generation of sorts in clause sets, Internal SEKI-report MK-85-6, university of Kaiserslautern (1985)

Sch85d Schmidt-Schauss M., Unification in a many-sorted calculus with declarations, Proc. of the 9th GWAI, Dassel/Solling, (ed. H. Stoyan), pp. 118-132, (1985)

Sch86a Schmidt-Schauss, M., Unification in Many-sorted equational theories, Proc of the 8th CADE, LNCS 230, pp. 538-552, (1986)

Sch86b Schmidt-Schauss, M., Unification under Associativity and Idempotence is of Type Nullary, JAR 2,3, pp. 277-281, (1986)

Sch86c Schmidt-Schauß, M.,'Unification Properties of Idempotent Semigroups', SEKI Report SR-86-07, Universität Kaiserslautern, 1986

Sh67 Shoenfield, J.R, 'Mathematical Logic', Addison-Wesley, (1967)

Sh84 Shostak, R.E., Deciding Combinations of Theories, JACM 31, 1, pp. 1-12, (1984)

SS81 Siekmann, J., Szabó, Unification and Regular ACFM Theories, Proc of IJCAI '81, Vancouver, pp. 532-538, (1981)

Si75 Siekmann, J., Stringunification, Essex university, Memo CSM-7, (1975) Si84 Siekmann, J.H., Universal Unification, Proc. $7^{th}$ CADE, Napa Valley, California, LNCS 170, pp. 1-42, (1984).

Si86 Siekmann, J.H., Unification Theory, Proc. of ECAI`86, Vol II, p. vi-xxxv, Brighton, (1986)

Si87 Siekmann, J.H., Unification Theory, to appear in Journal of Symbolic Computation, special issue on unification , (1987)

Smu78 Smullyan R., What is the Name of this Book? , Prentice Hall (1978)

Sm86 Smolka, G., Order-sorted Horn Logic, Semantic and Deduction, Seki-report SR-86-17, University of Kaiserslautern, (1986)

SNMG87 Smolka, G., Nutt, W., Meseguer, J., Goguen. J.A., Order-sorted Equational Computation., CREAS workshop, Austin, Texas, (1987) , to appear

SA87 Smolka, G., Aït-Kaci. H., Inheritance Hierarchies: Semantics and Unification, MCC Technical report AI-057-87, (1987)

Sn86 G.Snelting, 'Inkrementelle semantische Analyse in unvollständigen Programmfragmenten mit Kontextrelationen', Dissertation, Techn. Univ. Darmstadt, FB Informatik, 1986

SH85 G.Snelting, W.Henhapl: 'Unification in Many Sorted Algebras as a Device for Incremental Semantic Analysis', Internal Report PU2R2/85, Techn. Univ. Darmstadt, FB Informatik, 1985

St81 Stickel, M., A Unification Algorithm for Associative-Commutative Functions, JACM 28, 3 pp. 423-434, (1981)

St85 Stickel, M., Automated deduction by Theory Resolution, Journal of Automated Reasoning 1,4 , pp. 333-355, (1985)

St86 Stickel, M., Schubert's Steamroller Problem: Formulation and Solutions, JAR 2,1, pp. 89-101, (1986)

Sz82 Szabó, P., Theory of first order unification, (in German), Thesis, University of Karlsruhe, (1982)

Ta68 A.Tarski: 'Equational Logic and Equational Theories of Algebra', (Schmidt et al eds.) , Contributions to Mathematical Logic, North Holland, pp. 275-288, (1968)

Ta79 Taylor, W., 'Equational Logic', Houston J. of Math. , 5, (1979)

Ti86a Tidén, E., First-Order Unification in Combinations of Equational Theories, Thesis, Stockholm, (1986)

Ti86b E.Tidén: 'Unification in Combination of Collapse Free Theories with Disjoint Sets of Function Symbols', Proc. 8th CADE, LNCS 230, pp. 431-449, (1986)

Tu85 Turner, D.A., Miranda: A non-strict functional language with polymorphic types, in Functional and Programming Languages and Computer Architecture, LNCS 201, pp.1-16, (1985) Wad82 Wadge, W. W., 'Classified Algebras', Internal report N° 46, University of Warwick, England, (1982)

Wa83 Walther C., A many-sorted calculus based on resolution an paramodulation, Proc. of the 8th IJCAI, Karlsruhe, (1983)

Wa84 Walther C., Unification in many-sorted Theories, Proc. of the 6th ECAI, Pisa, ed. T. O´Shea, North-Holland, pp. 383-392 (1983)

Wa85 Walther C., A mechanical solution of Schubert´s Steamroller by many-sorted resolution. In Proc. 4th AAAI (Austin 1984), pp. 330-334, revised version in in Artificial Intelligence 26,2 (1985), pp. 217-224

Wa86 Walther C., A classification of many-sorted unification problems, Proc of the 8th CADE, LNCS 230, pp. 525-537, (1986)

Wa87 Walther, C., A many-sorted calculus based on resolution an paramodulation, Pitman & Kaufman publishers (to appear)

Wan52 Wang, H., Logic of Many-Sorted Theories, JSL 17, 2, pp. 105-116, (1952)

WR73 Wos, L., Robinson, G., Maximal Models and Refutation Completeness: Semidecision Procedures in Automated Theorem Proving. In "wordproblems" (W.W.Boone, F.B.Cannonito, R.C. Lyndon eds.) North-Holland, pp. 609-639, (1973)

WRC67 Wos L., Robinson G., Carson D., Shalla L., The Concept of Demodulation in Theorem Proving. JACM 14, pp. 698-709, (1967)

## Appendix

In the following we deal with monadic theories, i.e. regular equational theories in which every function symbol is unary. For convenience we omit brackets in terms and denote terms as strings. Sometimes we omit the variable, if it is clear from the context or does not matter.

**Theorem A.1:** There exists a theory E such that $\mu U_E(s = t)$ exists for all terms, but $\mu U_E(s_1 = t_1, s_2 = t_2)$ does not exist for some terms $s_i, t_i$, $i = 1,2$.

**Proof:** We construct a regular, simple, $\Omega$-free and monadic theory that has the property stated in the theorem.
Let E be the theory with the term rewriting system consisting of the following 11 rewrite rules:

$$
\begin{array}{lll}
R := \{ & f_i g_1 \to g_2 f_i\,,\ i = 1,2,3,4 & (R_1 - R_4) \\
& f_1 k_1 \to f_2\, k_1\,,\ f_3 k_2 \to f_4\, k_2 & (R_5, R_6) \\
& k_1 h \to k_2 h & (R_7) \\
& g_1 k_2 hl \to k_2 h\,, & (R_8) \\
& f_1 k_2 h \to f_2 k_2 h\,, & (R_9) \\
& g_2 f_2 k_2 hl \to f_2 k_2 h\,,\ g_2 f_4 k_2 hl \to f_4 k_2 h\ \} & (R_{10}, R_{11})
\end{array}
$$

This term rewriting system is canonical, the last three rules come from the completion of the first eight rules.
Note that the equational theory has some symmetries:
We can interchange i) $f_1$ and $f_2$, ii) $f_3$ and $f_4$, iii) $f_1, f_2, k_1$ and $f_3, f_4, k_2$ without changing the equational theory.
Furthermore the rewrite rules do never permute symbols.

i) E is simple:
Assume by contradiction that there are terms s,t such that $s =_E t$ and s is a proper subterm of t.Without loss of generality we can assume that s is in normalform. Since R is canonical, there exists a reduction from t to s. The rules 5,6,7,9 are not used during this reduction, since they increase the number of $f_2$, $f_4$ or $k_2$ and the other rules do not change the number of these symbols in terms. Hence $\#(F,s) = \#(F,t)$ for all function symbols $F \in \{f_1, f_2, f_3, f_4, k_1, k_2, h\}$. Thus t can be written as the string $t_0 s$, where $t_0$ contains only function symbols in $\{g_1, g_2, l\}$. The rules 8,10 and 11 are not used, since they delete l's from s as subterm of t, but there is no rule that adds the symbol l. The rules 1 - 4 alone are

not sufficient for such a reduction, since they increase the number of $g_2$´s. This is a contradiction.

ii) E is $\Omega$-free:

Assume by contradiction there is a function symbol F and terms s,t such that $F(s) =_E F(t)$ and $s \neq_E t$. We can assume that the pair s, t is a minimal such pair. Furthermore we can assume that s,t are in normalform. Since F(s) and F(t) are not literally equal at least one of them is reducible. Withour loss of generality we can assume that F(s) is reducible.

Obviously the symbol F is in $\{f_1,f_3,k_1,g_1,g_2\}$. We show that every possibility for F gives a contradiction:

1) $F \neq k_1$: If $F = k_1$, then s = hs' and hence t = ht'. The equation $k_2hs' =_E k_2ht'$ implies $s' =_E t'$ and hence $s =_E t$.

2) $F \neq g_1$: If $F = g_1$, then $s = k_2hls'$ and hence F(t) is reducible and $t = k_2hlt'$. The equation $k_2hs' =_E k_2ht'$ implies $s' =_E t'$ and hence $s =_E t$.

3) $F \neq g_2$ : If $F = g_2$, then either rule 10 or 11 is applicable to F(s). The term s is either of the form $f_2k_2hls'$ or $f_4k_2hls'$. Let us consider the first case $s = f_2k_2hls'$. Then F(t) is reducible and $t = f_2k_2hlt'$. We have $f_2k_2hs' =_E f_2k_2ht'$, hence $s' =_E t'$, since s and t are chosen as minimal. This implies the contradiction $s =_E t$.

4) $F \neq f_i$:

We have four subcases:

(a) $s = k_1s'$ and s' does nort start with h, $F = f_1$.

(b) $F = f_1,f_2,f_3,f_4$ , and $s = g_1{}^ns'$ with $n \geq 1$ and s' does not start with $g_1$.

(c) $s = k_2s'$, $F = f_3$.

(d) $s = k_2hs'$, $F = f_1$.

a) $s = k_1s'$ implies that $F(s) =_E f_2k_1s'$ and the symbols $f_2k_1$ can no longer be used by a rewite rule. Checking the rewrite rules it is easy to see that the only possibility for t is to be of the form $k_1t'$, hence $f_2k_1s' =_E f_2k_1t'$. Since s' and t' do not start with h (otherwise s is reducible), this implies $s' =_E t'$. and hence $s =_E t$.

b) $s = g_1{}^ns'$ implies $g_2{}^nf_is' =_E f_it$.

Assume the normalform of $f_is$ starts with $g_2$. Then t is of the form $g_1t'$, and we have $f_ig_1{}^{n-1}s' =_E f_it'$. Minimality of s,t yields $g_1{}^{n-1}s' =_E t'$, hence $s =_E t$.

Assume the normalform of $f_is$ starts with a symbol $f_j$. Then $f_j$ is $f_2$ or $f_4$. The term $g_2{}^nf_is'$ must be reducible to a term with topsymbol $f_j$. Considering the rules we see that $s' = k_2hl\ s''$, but then $s = g_1{}^ns'$ is reducible by rule 8, a contradiction.

c) $s = k_2s'$ implies that the normalform of $f_3k_2s'$ starts with $f_4k_2$ and this two top function symbols are no longer involved in a reduction. To reduce $f_3t$ to a term of this form there are the possibilites $t = k_2t'$ or $t = g_1t'$. The second case is not possible as proved in case b). Hence t is of the form $k_2t'$ and we have $f_4k_2s' = f_4k_2t'$. Since $f_4k_2$ is not reducible this implies $s' =_E t'$ and hence $s =_E t$.

d) $s = k_2hs'$ implies $F(s) =_E f_2k_2hs'$. Due to b) the term t has the form $k_2ht'$ and we have $f_2k_2hs' = f_2k_2ht'$. This implies $s' =_E t'$ and hence $s =_E t$.

iii) 1) $f_1s =_E f_2t \Rightarrow s =_E t$:

Assume by contradiction that the statement is false. Let $f_1s =_E f_2t$ with $s \neq_E t$ . We can assume that s and t are irreducible and minimal. Furthermore we can assume that $f_1s$ is reducible.

There are three cases $s = g_1{}^ns'$, $s = k_1s'$, $s = k_2hs'$. The second and third case are not possible since E is $\Omega$-free.

Hence $s = g_1{}^ns'$ where $n \geq 1$ and s' does not start with $g_1$. If the term $f_2t$ is reducible, then t is of the form $g_1t'$ and we reach a contradiction by minimality of s,t and $\Omega$-freeness.

Thus $g_2{}^nf_1s'$ must be reducible to $f_2t$, which is only possible if s' is of the form $k_2hls''$. This is a contradiciton to the irreducibility of s.

2) $f_3s =_E f_4t \Rightarrow s =_E t$: Symmetric to a)

3) $k_1s =_E k_2t \Rightarrow s =_E t$: Obvious, since only rule 7 is applicable.

iv) For all terms s, t there exists a minimal set of solutions for $\langle s =_E t \rangle$:

Assume by contradiction that there exist terms (in normalform) $s_0,t_0$ such that a minimal set of unifiers for $\langle s_0 = t_0 \rangle_E$ does not exist. Then there exists a unifier $\sigma \in U_E(s_0{=}t_0)$, such that there exists no minimal $\tau_m \in U_E(s_0{=}t_0)$ with $\sigma \geq_E \tau_m$ $[\mathbb{V}(s_0,t_0)]$. Hence there exists an infinite descending chain $\sigma_1 >_E \sigma_2 >_E \dots$ $[\mathbb{V}(s_0,t_0)]$ in $U_E(s_0{=}t_0)$ with $\sigma \geq_E \sigma_1$ $[\mathbb{V}(s_0,t_0)]$. We assume that $VCOD(\sigma_i) = \{z\}$. Let $\lambda_i = \{z \leftarrow r_i\}$ be the substitution with $\sigma_i =_E \lambda_i\, \sigma_{i+1}$ $[\mathbb{V}(s_0,t_0)]$. We have to consider the two cases $\mathbb{V}(s_0) = \mathbb{V}(t_0) = \{x\}$ and $\mathbb{V}(s_0) = \{x\}$ , $\mathbb{V}(t_0) = \{y\}$, where x and y are different variables.

Case $\mathbb{V}(s_0) = \mathbb{V}(t_0) = \{x\}$.

Let $\sigma_i = \{x \leftarrow t_i\}$, where $t_i$ is in normalform and $\mathbb{V}(t_i) = \{z\}$. Without loss of generality we can assume that the depth of $t_i$ is properly increasing and that the number of h´s, k´s (i.e., the sum of the occurrences of $k_1$ and $k_2$) and f´s (i.e., the sum of the occurrences of $f_1$, $f_2$, $f_3$, and $f_4$) is constant in $t_i$. Furthermore we can assume that $(\sigma_i)$ is a chain with a minimum number of these function symbols. Considering the reduction rules we see that $r_i$ must be of the form $l^m$ and that $t_i$ stops with $k_2h$. From the rewrite rules it follows that $t_i$stops either with $f_2k_2h$,$f_4k_2h$ or $g_1{}^mk_2h$. If $t_i$ stops with $f_2k_2h$ or $f_4k_2h$, then we can delete these symbols from $t_i$ (obtaining $t_i'$) and get a unifier of $s_0$ and $t_0$, since a reduction proof of $\sigma_is_0 =_E \sigma_it_0$

works also for $\sigma_i's_0 =_E \sigma_i't_0$ where $\sigma_i' = \{x \leftarrow t_i'\}$. Furthermore $\sigma_i >_E \sigma_i'$ $[\mathbb{V}(s_0,t_0)]$, since $\sigma_i'$ has a smaller number of h´s. If for some j there is no minimal unifier $\sigma'_{jm} \in U_E(s_0=t_0)$ with $\sigma_j' \geq_E \sigma'_{jm}$ $[\mathbb{V}(s_0,t_0)]$, then we could find an infinitely descending chain $(\mu_i)$ starting with $\sigma_j'$. However, this chain has a smaller number of h´s, k´s and f´s than $(\sigma_i)$, hence this is a contradiction.

We have shown that $t_i$ stops with $g_1{}^m k_2 h$.

A similar argument as for the deletion of $f_2k_2h$ or $f_4k_2h$ above shows that $t_i$ is already of the form $g_1{}^m k_2 h$. Hence we can assume that $t_i = g_1{}^i k_2 h$. Since $s_0 \neq_E t_0$ and $\{x \leftarrow t_i\}$ E-unifies them, the rewrite rules show that $s_0$ and $t_0$ are of the form $s_0 = s_0'f_s$ and $t_0 = t_0'f_t$, where $s_0' =_E t_0'$ and either $\{f_s,f_t\} = \{f_1,f_2\}$ or $\{f_s,f_t\} = \{f_3,f_4\}$. But then there are more general unifiers than $\sigma_i$ in $U_E(s_0=t_0)$: Either $\sigma_i'' := \{x \leftarrow g_1{}^i k_1\}$ or $\sigma_i'' = \{x \leftarrow g_1{}^i k_2\}$ are such unifiers. Since these unifiers have a smaller number of h´s than $\sigma_i$, we have reached a contradiction.

Case $\mathbb{V}(s_0) = \{x\}$ , $\mathbb{V}(t_0) = \{y\}$

We do not give the proof in full detail, since the technique is exactly the same as in case 1.

Let $\sigma_i = \{x \leftarrow s_i, y \leftarrow t_i\}$, where $s_i,t_i$ are in normalform and $\mathbb{V}(s_i,t_i) = \{z\}$. Without loss of generality we can assume that the depths of $s_i$ and $t_i$ are increasing, that the depth of one of them is properly increasing and that the number of h´s, k´s and f´s is constant in $s_i$ and in $t_i$. Furthermore we can assume that $\sigma_i$ is a chain with a minimum number of these functions symbols. Considering the reduction rules we see that $r_i$ must be of the form $l^m$ and that both $s_i$ and $t_i$ stop with $k_2h$. Furthermore $s_i$ and $t_i$ can only stop with $f_2k_2h$ , $f_4k_2h$ or $g_1k_2h$.

Since $\sigma_i$ is a unifier it is not possible that $s_i$ stops with $f_2k_2h$ and $t_i$ stops with $f_4k_2h$.

If both $s_i$ and $t_i$ stop with $f_2k_2h$ (or $f_4k_2h$), then the same argument as in the proof of the other case yields a contradiction.

If both $s_i$ and $t_i$ stop with $g_1k_2h$, then we first argue that $s_i$ and $t_i$ are of the form $g_1{}^n k_2 h$ and $g_1{}^m k_2 h$ and then that both $s_0$ and $t_0$ stop with an $f_j$. A similar argument as above shows that we can replace the right end of $s_i$ and $t_i$ by $g_1k_1$ or $g_1k_2$ and obtain a more general unifier.

The last case is that $s_i$ stops with $f_2k_2h$ and $t_i$ stops with $g_1k_2h$. We see that $t_i$ has the form $g_1{}^m k_2 h$ and then the structure of R shows that $t_0$ stops with some $f_j$. But then the same argument as above yields a more general substitution, replacing the right end of $s_i$ by $f_2k_1$ and the right end of $t_i$ by $g_1{}^m k_1$. This is a contradiction.

The case that $s_i$ stops with $f_4k_2h$ and $t_i$ stops with $g_1k_2h$ is analogous to the previous case.

v) There does not exist a minimal set of unifiers for some set of equations:

Consider the system $\langle f_1(x) = f_2(x), f_3(x) = f_4(x) \rangle_E$. A complete set of unifiers for $\langle f_1(x) = f_2(x) \rangle_E$ is $U_E := \{ \sigma_n \}$, where $\sigma_n := \{x \leftarrow g_1{}^n k_1 z \}$.

For completenes let $\theta$ be a normalized unifier of $f_1(x)$ and $f_2(x)$. Since $f_1\theta x$ is reducible, we have to check three cases:

$\theta x = g_1 s'$, $\theta x = k_1 s'$ and $\theta x = k_2 hs'$.

1) $\theta x = g_1 s'$. Then $g_2 f_1 s' =_E g_2 f_2 s'$, hence $\mu := \{x \leftarrow s'\}$ is already a solution of $\langle f_1 x = f_2 x \rangle_E$. Induction shows that $\mu$ is an instance of some $\sigma_n$. But then $\theta$ is an instance of $\sigma_{n+1}$.

2) $\theta x = k_1 s'$. Then $\theta$ is an instance of $\sigma_0$.

3) $\theta x = k_2 hs'$. Then $\theta$ is an instance of $\sigma_0$.

Common solutions of $\langle f_1(x) = f_2(x), f_3(x) = f_4(x) \rangle_E$ can be obtained from common solutions of $\langle x = g_1{}^n k_1 z , x = g_1{}^n k_2 z \rangle$. Hence $cU_E(f_1(x) = f_2(x), f_3(x) = f_4(x)) = \{ \{x \leftarrow g_1{}^n k_2 hz \} \mid n \geq 0\}$ is a complete set of E-unifiers that has no minimal, complete subset, since $g_1{}^n k_2 hlz = g_1{}^{n-1} k_2 hz$. ■

**Remark:** It is a pure technical task to construct theories $E_n$ from the above theory such that minimal sets of unifiers exist for all sets of n equations, but not for all sets of equations.

The next proposition shows that $E \in \mathcal{U}$ is not equivalent to the condition that every decreasing chain of substitutions in every $U_E(\Gamma)$ has a lower bound in $U_E(\Gamma)$:

**A.2 Proposition** There exists a theory $E \in \mathcal{U}$ such that there exist terms s,t and an infinite decreasing chain $\sigma_1 >_E \sigma_2 >_E \ldots$ of substitutions in $U_E(s,t)$ without a lower bound in $U_E(s,t)$.

**Proof:** We construct a regular, simple, $\Omega$-free and monadic equational theory E that has the properties stated in the proposition.

Let E be defined by the term rewriting system consisting of the following five rewrite rules:

$$R := \{ \quad f_1 g_1 \to g_2 f_1 \,,\ f_2 g_1 \to g_2 f_2 \,, \qquad (R_1, R_2)$$
$$f_1 k \to f_2 k \,, \qquad (R_3)$$
$$g_1 khl \to kh \,, \qquad (R_4)$$
$$g_2 f_2 khl \to f_2 kh \quad \} \qquad (R_5)$$

This term rewriting system is canonical, the last rule comes out of the completion of the first four rules. Note that interchanging $f_1$ and $f_2$ does not change the equational theory.

i) E is simple:

Assume by contradiction that there are terms s,t such that $s =_E t$ and s is a proper subterm of t. Without loss of generality we can assume that s is in normalform. Since R is canonical, there exists a reduction from t to s. Rule 3 is not used during this reduction, since it increases the number of $f_2$´s. Hence s and t have the same number of $f_1$´s, $f_2$´s, k´s and h´s. The rules 4 and 5 are not used, since these rules delete l´s from s as subterm of t, but there is no rule that adds the symbol l. The rules 1 and 2 alone are not sufficient for such a reduction, since they increase the number of $g_2$´s. This is a contradiction.

ii) E is $\Omega$-free:

Assume by contradiction there is a function symbol F and terms s,t such that $F(s) =_E F(t)$ and $s \neq_E t$. We can assume that the pair s, t is a minimal such pair. Furthermore we can assume that s,t are in normalform. Since F(s) and F(t) are not literally equal, at least one of them is reducible. We assume that F(s) is reducible. Obviously $F \in \{f_1,f_2,g_1,g_2\}$. That $F \in \{g_1,g_2\}$ can easily be excluded by considering the rules and by the minimality of s and t.

1) If $F = f_2$, then $s = g_1s'$. If F(t) is reducible, then $t = g_1t'$ and by minimality of s and t we obtain $f_2s' =_E f_2t'$, hence $s' =_E t'$, which is a contradiction. Hence F(t) is not reducible. The only possibility to reduce $f_2s$ to a term with topsymbol $f_2$ is that $s = g_1{}^nkhl^ns''$, but then s is reducible to khs'', a contradiction to the reducibility of F(s).

2) If $F = f_1$, then $s = ks'$ or $s = g_1s'$.

   a) If $s = ks'$, then $f_1t$ is reducible. From the minimality of s and t it follows that $t = kt'$ is not possible, hence $t = g_1{}^nt'$. The term $f_1g_1{}^nt'$ must be reducible to a term with topsymbol $f_2$. This is only possible if t is reducible, which is a contradiction.

   b) If $s = g_1s'$, then $f_1t$ is reducible, hence $t = g_1t'$ or $t = kt'$ . The case $t = g_1t'$ can be eliminated by induction. The other case is symmetric to a case in a).

iii) $E \in \mathcal{U}$:

Assume by contradiction that there exists a system $\Gamma$ and a nullary unifier $\sigma_0 \in U_E(\Gamma)$, i.e., there is no minimal unifier $\tau_m$ with $\sigma_0 \geq_E \tau_m[\mathbb{V}(\Gamma)]$ . Then there is an infinite descending chain $\sigma_0 >_E \sigma_1 >_E \sigma_2 >_E \dots [\mathbb{V}(\Gamma)]$ in $U_E(\Gamma)$. We can assume that all terms in $COD(\sigma_i)$ are in normalform, that $DOM(\sigma_i) = \mathbb{V}(\Gamma)$ and that $\sigma_0$ is minimal with respect to the number of h´s, k´s and f´s occurring in it, and hence the number of h´s, k´s and f´s is constant in $\sigma_ix$ for every $x \in \mathbb{V}(\Gamma)$. Furthermore we can assume that the term depth of every $\sigma_ix$ is non-decreasing. A further assumption is that there is at least one $x \in \mathbb{V}(\Gamma)$, such that the depth of $\sigma_ix$ is unbounded. The above assumptions and the rewrite rules show, that for every $x \in \mathbb{V}(\Gamma)$, such that $\sigma_i x$ is unbounded, the term $\sigma_i x$ stops with kh. Let Y := $\{x \in \mathbb{V}(\Gamma) \mid \sigma_i x$ is unbounded$\}$. Note that $Y \neq \emptyset$.

For every $s=t \in \Gamma$, either $\mathbb{V}(s,t) \subseteq Y$ or $\mathbb{V}(s,t) \cap Y = \emptyset$:

Assume by contradiction that $s=t \in \Gamma$ with $\mathbb{V}(s) = \{x_0\}$, $\mathbb{V}(t) = \{x_1\}$, $x_0 \in Y$ and $x_1 \notin Y$. Then there exsits a $n_0$ , such that depth($\sigma_j x_1$) is constant for all $j \geq n_0$. By choosing a subchain we can further assume that all $\sigma_j x_1$ are E-equal for $j \geq n_0$. Let $\lambda_i$ be a substitution with $\lambda_i \sigma_i =_E \sigma_{i-1}$ $[\mathbb{V}(\Gamma)]$. Then the rewrite rules show that only the symbol l can occur as a function symbol in the codomain of $\lambda_i$. Since the theory E is regular, we have $\mathbb{V}(\sigma_j x_1) = \mathbb{V}(\sigma_j x_0) = \{z_j\}$. The term $\lambda_j z_j$ is of the form $l^k$ for some $k > 0$, since the depth for $\sigma_i x_0$ is properly increasing. If we choose $r = \sigma_j x_1$ for $j > n_0$ then r has the property $rl^k =_E r$ with $k > 0$, which is impossible. ❑

Let $Z = \mathbb{V}(\sigma_0(Y))$ and let $\rho$ be a renaming of Z such that COD($\rho$) consists of new variables. We define $\tau$ as follows: $\tau x := \sigma_0 x$, for $x \in \mathbb{V}(\Gamma)\setminus Y$ , and $\tau x :=$ $t_x(\rho z_x)$, if $\sigma x = t_x h(z_x)$ and $x \in Y$.

The substitution $\tau$ is a unifier:

Let $s = t$ be any equation in $\Gamma$ and let $\sigma_0 s =_E \sigma_0 t$. The above consideration shows that either $\mathbb{V}(s,t) \cap Y = \emptyset$ or $\mathbb{V}(s,t) \subseteq Y$. In the first case we have obviously $\tau s =_E \tau t$ and in the second case every reduction proof of $\sigma_0 s =_E \sigma_0 t$ is a reduction proof for $\tau s =_E \tau t$ , since the rightmost h is not touched by a rewrite rule.

The substitution $\tau$ is more general than $\sigma_0$: Let $\lambda$ be defined as follows: $\lambda z' := h(z)$, if $z' = \rho z$ and $\lambda x = x$ otherwise. We have $\sigma_0 =_E \lambda\tau$ $[\mathbb{V}(\Gamma)]$:

If $x \notin Y$, then $\sigma_0 x =_E \lambda\tau x =_E \tau x$ .

If $x \in Y$, then $\lambda\tau x =_E \lambda(t_x(\rho z_x)) =_E t_x(\lambda\rho z_x) = t_x h(z_x) = \sigma_0 x$ . ❑

Minimality of $\sigma_0$ implies that $\tau$ is not nullary, hence $\sigma_0$ is not nullary, a contradiction to our assumption.

iv) There exists a set of equations $\Gamma$ and an infinitely descending chain of substitutions $\sigma_1 >_E \sigma_2 >_E \ldots$ $[\mathbb{V}(\Gamma)]$ in $U_E(\Gamma)$ without a lower bound in $U_E(\Gamma)$:

Let $\Gamma = \langle f_1 x = f_2 x\rangle$. The set $\{\{x \leftarrow g_1{}^n kz\} \mid n \geq 0\}$ is a complete subset of $U_E(\Gamma)$ as can easily be seen by induction using that E is $\Omega$-free. Let $\sigma_n$ be defined by $\sigma_n x :=$ $g_1{}^n khz$. We have $\sigma_1 >_E \sigma_2 >_E \ldots$ $[\mathbb{V}(\Gamma)]$. Assume there is a lower bound $\sigma \in U_E(\Gamma)$ with $\sigma_i \geq_E \sigma$ $[\mathbb{V}(\Gamma)]$ for all i. Since $\{\{x \leftarrow g_1{}^n kz\}\}$ is a complete subset of $U_E(\Gamma)$, there exists a number m such that $\{x \leftarrow g_1{}^m kz\}$ is more general than $\sigma$. Hence for all n: $g_1{}^n khz \geq_E g_1{}^m kz$. The rewrite rules imply that this is impossible. ■

# Index.

## Special Symbols.

| | | |
|---|---|---|
| $\mathcal{D}(f)$ | domain of the function f | I.1 |
| $\langle A,\leq\rangle$ | a set A ordered by $\leq$. | I.1 |
| $[a,\infty]$ | the set of elements greater than or equal to a | I.1 |
| $[-\infty,a]$ | the set of elements less than or equal to a | I.1 |
| $C(U)$ | set of complete subsets of U | I.1 |
| $cU$ | element of C(U) | I.1 |
| $M(U)$ | set of minimal, complete subsets of U | I.1 |
| $\mu U$ | element of U | I.1 |
| $\overline{\Sigma}$ | unsorted signature | I.2 |
| $\mathbf{F}_{\overline{\Sigma}}$ | set of function symbols, elements are denoted by f,g,h | I.2 |
| $\mathbf{V}_{\overline{\Sigma}}$ | set of variable symbols, elements are denoted by x,y,z | I.2 |
| $\mathbf{P}_{\overline{\Sigma}}$ | set of predicate symbols, elements are denoted by P,Q | I.2 |
| arity(f) | arity of the function symbols f | I.2 |
| $\mathbf{T}$ | set of terms terms are denoted as p,q,r,s,t,u,v,w | I.2 |
| $\mathbf{V}(t)$ | variables in term t. | I.2 |
| $\mathbf{A}$ | set of atoms | I.2 |
| $\mathbf{L}$ | set of literals | I.2 |
| $\mathbf{T}_{gr}$ | set of ground terms. | I.2 |
| $\Lambda$ | empty word | I.2 |
| D(t) | set of occurrences in t | I.2 |
| O(t) | set of nonvariable occurrences in t | I.2 |
| depth(t) | term depth of t | I.2 |
| $t\backslash\pi$ | subterm of t at occurrence $\pi$ | I.2 |
| $t[\pi \leftarrow s]$ | s replaces the subterm of t at occurrence $\pi$ | I.2 |
| subterms(t) | the set of subterms of t. | I.2 |
| $\varphi$, $\psi$ | endomorphisms, homomorphisms | I.2 |
| SUB | set of all substitutions, elements are denoted as $\sigma,\tau$ | I.2 |
| DOM($\sigma$) | $\{x \in \mathbf{V}(t) \mid \sigma x \neq x\}$ | I.2 |
| COD($\sigma$) | $\sigma$DOM($\sigma$) | I.2 |
| I($\sigma$) | V(COD($\sigma$)) | I.2 |
| $\sigma = \tau\ [W]$ | $\sigma x = \tau x$ for all $x \in W$ | I.2 |

| | | |
|---|---|---|
| Id | identical substitution | I.2 |
| $\sigma \circ \tau$ | composition of $\sigma$ and $\tau$ | I.2 |
| $\sigma_{\|W}$ | restriction of $\sigma$ to the set of variables W | I.2 |
| $\rho^-$ | converse of a renaming | I.2 |
| $\mathbf{S}_\Sigma$ | set of sort symbols | I.3 |
| $\Sigma$ | sorted signature | I.3 |
| t:S | term declaration | I.3 |
| $f: S_1 \times \ldots \times S_n \to S$ | function declaration | I.3 |
| $S \sqsubseteq T$ | subsort-declaration | I.3 |
| $\mathbf{TD}_\Sigma$ | set of term declarations | I.3 |
| $\mathbf{SD}_\Sigma$ | subsort-declarations | I.3 |
| $\sqsubseteq_\Sigma$ | subsort-ordering | I.3 |
| $\mathbf{T}_\Sigma$ | set of all well-sorted terms | I.4 |
| $\mathbf{T}_{\Sigma,S}$ | set of all well-sorted terms of sort S | I.4 |
| $\mathbf{S}_\Sigma(t)$ | $\{S \in \mathbf{S}_\Sigma \mid t \in \mathbf{T}_{\Sigma,S}\}$ | I.4 |
| $\mathbf{S}(x)$ | sort of a variable (also denoted as x:S) | I.4 |
| $\mathbf{LS}_\Sigma(t)$ | unique least sort in $\mathbf{S}_\Sigma(t)$ | I.4 |
| TOP | topsort | I.4 |
| $\mathrm{SUB}_\Sigma$ | well-sorted substitutions | I.4 |
| $s \geq_\Sigma t$ | s is a $\Sigma$-instance of t | I.5 |
| $s \equiv_\Sigma t$ | $s \geq_\Sigma t$ and $s \leq_\Sigma t$ | I.5 |
| $\sigma = \tau\ [W]$ | $\sigma x = \tau x$ for all $x \in W$ | I.5 |
| $\sigma \geq_\Sigma \tau\ [W]$ | $\exists \lambda \in \mathrm{SUB}_\Sigma\ \lambda\tau = \sigma\ [W]$ | I.5 |
| $\sigma \equiv_\Sigma \tau\ [W]$ | $\sigma \geq_\Sigma \tau\ [W]$ and $\sigma \leq_\Sigma \tau\ [W]$ | I.5 |
| $\mathcal{A}$ | $\Sigma$-quasi-algebra, $\Sigma$-algebra | I.6 |
| $S_A$ | denotation for S in the algebra A. | I.6 |
| $f_A$ | denotation for f in the algebra A | I.6 |
| $\varphi_h$ | homomorphic extension of $\varphi$ | I.6 |
| $A/\equiv$ | quotient algebra of A modulo $\equiv$ | I.7 |
| $\gamma$ | canonical mapping $\gamma: A \to A/\equiv$ | I.7 |
| $\mathcal{S}$ | specification ($\Sigma$, CS) | I.8 |
| I | interpretation $(\mathcal{M},\Phi)$ | I.8 |
| $\mathcal{M}$ | $\Sigma$-model | I.8 |
| $A \models s = t$ | s = t holds in algebra A, | I.9 |
| $\mathcal{T}(\mathcal{E})$ | the set of consequences of $\mathcal{E}$. | I.9 |
| $\vdash$ | deduction operator | I.9 |
| $=_{\Sigma,E}$ | equality relation induced by a theory $\mathcal{E} = (\Sigma,E)$ | I.9 |
| $\sigma =_{\Sigma,E} \tau\ [W]$ | $\sigma x =_{\Sigma,E} \tau x$ for all $x \in W$ | I.9 |

| | | |
|---|---|---|
| $\sigma \cup \tau$ | union of substitutions | I.10 |
| $\sigma^*$ | idempotent closure of substitution $\sigma$ | I.10 |
| $s \geq_{\Sigma,E} t$ | $\exists \lambda \in SUB_\Sigma\ \lambda t =_{\Sigma,E} s$ | I.10 |
| $\sigma \geq_{\Sigma,E} \tau\ [W]$ | $\exists \lambda \in SUB_\Sigma\ \lambda \tau =_{\Sigma,E} \sigma\ [W]$ | I.10 |
| $\sigma \equiv_{\Sigma,E} \tau\ [W]$ | $\sigma \geq_{\Sigma,E} \tau\ [W]$ and $\sigma \leq_{\Sigma,E} \tau\ [W]$ | I.10 |
| $\langle s_i = t_i \mid i = 1,\ldots,n \rangle_{\mathcal{E}}$ | E-unification problem, also denoted as $\Gamma$ | I.11 |
| $U_{\Sigma,E}(\Gamma)$ | set of $\mathcal{E}$-unifiers of $\Gamma$ | I.11 |
| $CU_{\Sigma,E}(\Gamma)$ | set of all complete set of $\mathcal{E}$-unifiers of $\Gamma$ elements are denoted as cU | I.11 |
| $MU_{\Sigma,E}(\Gamma)$ | set of all minimal, complete set of $\mathcal{E}$-unifiers of $\Gamma$ elements are denoted as $\mu$U | I.11 |
| $\mathcal{U}_1$ | class of unitary unifying theories | I.11 |
| $\mathcal{U}_\omega$ | class of finitary unifying theories | I.11 |
| $\mathcal{U}_\infty$ | class of infinitary unifying theories | I.11 |
| $\mathcal{U}_0$ | class of nullary unifying theories | I.11 |
| $\mathcal{U}$ | class of unification based theories, $\mathcal{U}_1 \cup \mathcal{U}_\omega \cup \mathcal{U}_\infty$ | I.11 |
| $\mathcal{U}_{1,\mathrm{eff}}$ | class of unitary unifiying theories with effectively computable set $\mu U(\Gamma)$ | I.11 |
| $\langle s_i = t_i \mid i = 1,\ldots,n \rangle_{\mathcal{E}}$ | E-unification problem, also denoted as $\Delta$ | I.11 |
| $M_{\Sigma,E}(\Gamma)$ | set of $\mathcal{E}$-matchers of $\Delta$ | I.11 |
| $\mathcal{M}_1$ | class of unitary matching theories | I.11 |
| $\mathcal{M}_\omega$ | class of finitary matching theories | I.11 |
| $\mathcal{M}_\infty$ | class of infinitary matching theories | I.11 |
| $\mathcal{M}_0$ | class of nullary matching theories | I.11 |
| $\mathcal{M}$ | class of matching based theories, $\mathcal{M}_1 \cup \mathcal{M}_\omega \cup \mathcal{M}_\infty$ | I.11 |
| $W_\Sigma(\tau)$ | set of weakening substitutions | I.11 |
| $W_\Sigma(t \sqsubseteq S)$ | set of weakening substitutions for t | I.11 |
| $s \longrightarrow_{\pi,e,\sigma} t$ | demodulation relation at occurrence $\pi$ in s, with equation e, and substitution $\sigma$ | I.12 |
| $s \xrightarrow{*} t$ | transitive closure of the demodulation relation | I.12 |
| $\longrightarrow_R$ | reduction relation induced by a term rewriting system | I.12 |
| $\xleftrightarrow{*}$ | symmetric closure of $\xrightarrow{*}$ | I.12 |
| $\Vert s \Vert_R$ | normalform of s with respect to R | I.12 |
| $UV(\Gamma)$ | set of used vauiables of $\Gamma$ during a transformation | I.13 |
| $\langle \sigma \rangle_{\mathcal{E}}$ | equation system induced by $\sigma$ | I.13 |
| $\Longrightarrow_{RS}$ | transfomation relation from the rule system RS | I.13 |
| $\xRightarrow{*}_{RS}$ | transitive, reflexive closure ofd the transformation-relation | I.13 |

| | | |
|---|---|---|
| $*$ | the unsolvable equation system | I.13 |
| $\Gamma_S$ | solved part of $\Gamma$ | I.13 |
| $\Gamma_U$ | unsolved part of $\Gamma$ | I.13 |
| $s =_{I,\Sigma,E} t$ | s and t are $\mathcal{E}$-equal modulo the initial algebra | I.14 |
| $\leq_{sI,\Sigma,E} [W]$ | strong initial subsumption | I.14 |
| $\leq_{wI,\Sigma,E} [W]$ | weak initial subsumption | I.14 |
| $\leq_{F,\Sigma,E} [W]$ | free subsumption $\leq_{\Sigma,E} [W]$ | I.14 |
| $QT(\mathcal{E})$ | quasi-terms with respect to $\mathcal{E}$. | I.14 |
| $\Longrightarrow_R$ | parallel reduction with respect to the TRS R | II.3 |
| $<_s$ | simplification ordering on ground terms | II.3 |
| $SUB_R$ | set of substitutions for a restricted R-system | II.8 |
| $\mathbf{T}_R$ | set of terms of an R-system | II.8 |
| $\leq_R$ | subsumption ordering with respect to $SUB_R$ | II.8 |
| $\equiv_R$ | equivalence relation corresponding to $\leq_R$ | II.8 |
| $[x]_{\equiv R}$ | equivalence class of x with respect to $\equiv_R$ | II.8 |
| $\mathcal{S}_{REL}$ | relativited specification | II.10 |
| $\mathcal{S}_{EQR}$ | equationally relativized specification | II.10 |
| $s — t$ | s is connected to t by an equation in $\Gamma$ | IV.1 |
| k(CS) | k.parameter, $k(CS) = \Sigma\{\lvert CS\rvert - 1 \mid C \in CS\}$ | V.2 |
| $\sigma =_{rig} \tau$ [E, W] | $\sigma$ is rigid-equal to $\tau$ modulo the theory E and the set of variables W | V.5 |
| $\sigma \leq_{rig} \tau$ [E, W] | $\tau$ is rigid-instance of $\sigma$ modulo the theory E and the set of variables W | V.5 |
| $A_{rid}$ | complete rigid E-unification procedure | V.5 |
| SPE | set of sort-predicate equivalences | VI.1 |
| $P \leftrightarrow S_P$ | $(P,S_P) \in$ SPE | VI.1 |
| ISC | set of intersection constraints | VI.1 |
| $S_1 \cap \ldots \cap S_n = T$ | $(\{S_1,\ldots,S_n\},T) \in$ ISC | VI.1 |

# Lebenslauf

| | |
|---|---|
| 9.8.1953 | geboren in Lorch/Rhg. |
| 1960-1964 | Besuch der Grundschule in Lorch/Rhg. |
| 1964-1972 | Besuch des Rheingau-Gymnasiums in Geisenheim<br>Abitur im Juni 1972. |
| 1972 | Studium der Mathematik mit Nebenfach Physik an der Johannes-Gutenberg Universität Mainz. |
| 1973-1974 | Wehrdienst: |
| 1974-1979 | Studium der Mathematik mit Nebenfach Physik an der Johannes-Gutenberg Universität Mainz.<br>Diplom im Juni 1979. |
| 1979-1983 | Organisator für Datenverarbeitung bei der Unternehmensberatung Heyde&Brandt in Bad Nauheim. |
| 1984-1987 | wissenschaftlicher Mitarbeiter am Fachbereich Informatik der Universität Kaiserslautern in der Arbeitsgruppe Siekmann im Bereich Künstliche Intelligenz und Automatische Deduktionssysteme. |