UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://jeuww.cs.uni-sb.de/pub/www/

SEKI Report

An Abstraction for Proof Planning: The $\mathcal{S}$-Abstraction

Serge Autexier

SEKI Report SR-97-05

An Abstraction for Proof Planning: The $\mathcal{S}$-Abstraction

Serge Autexier
Fachbereich 14, Informatik,
Postfach 15 11 50,
66041 Saarbrücken, Germany
E-mail: autexier@cs.uni-sb.de

Contents

Abstract

This paper presents a new kind of abstraction, which has been developed for the purpose of proof planning. The basic idea of this paper is to abstract a given theorem and to find an abstract proof of it. Once an abstract proof has been found, this proof has to be refined to a real proof of the original theorem. We present a goal oriented abstraction for the purpose of equality proof planning, which is parameterized by common parts of the left- and right-hand sides of the given equality. Therefore, this abstraction technique provides an abstract equality problem which is more adequate than those generated by the abstractions known so far. The presented abstraction also supports the heuristic search process based on the difference reduction paradigm. We give a formal definition of the abstract space including the objects and their manipulation. Furthermore, we prove some properties in order to allow an efficient implementation of the presented abstraction.

Chapter 1

Introduction

The background of the work presented in this paper is the domain of proof planning. The idea of proof planning got a strong enhancement by the introduction of explicit proof plans by A. Bundy ([Bundy, 1987]). The idea is to define tactics, which where firstly introduced in the LCF system ([Gordon et al., 1979]), as plan operators (also called methods) with pre- and postconditions to generate a part of a proof. These tactics allow to achieve certain goals (formulated by the postconditions) under certain conditions (preconditions). The search process for a proof plan is mainly the task to find a plan composed of these operators, allowing to prove a given theorem under certain conditions (axioms).

On the other hand syntactical abstractions of formulas have been used to guide the proof search. The underlying idea is to abstract a given theorem P to an abstract theorem Q, to prove Q and to refine the abstract solution into a proof of the original theorem P. E. g. Plaisted ([Plaisted, 1981]) used propositional abstractions of first order logic formulas to generate a propositional proof plan for a first order logic theorem. Abstractions used for the purpose of proof planning have to satisfy certain properties, and thus, among other things, Giunchiglia and Walsh presented a theory of abstraction where the required properties are formally defined.

Several abstractions like Plaisted's propositional abstraction have already been presented for the purpose of proof planning, but none of them had a real success. This is due to the fact, that the proposed abstractions are not goal oriented and subsequently for some problems the abstraction is too abstract where for some others it is not abstract enough. Therefore, we present a goal oriented abstraction for the purpose of equality proof planning, where the abstraction of an equality is parameterized by common parts of the left- and right-hand sides of a given equation.

The proposed abstraction for equations $s = t$ is based on common subterms of s and t and expresses some structural differences between s and t with respect to these common subterms. For example, $f(x, b)$ and $g(a, x)$ share x as a common subterm (a, b being constants, f and g unary function symbols). While in the first term x occurs in the first argument of f, it occurs in the second argument of g in the second term. Thus, a characteristic of the equation

$$\forall x \,.\, f(x, b) = g(a, x) \tag{1.1}$$

is that it behaves like "moving" x from the first argument of f into the second argument of g, or

vice versa. Ignoring any other subterms, this is expressed by the following notation

$$\langle f^1 \rangle_x = \langle g^2 \rangle_x \tag{1.2}$$

which is a concise representation of the structural behavior of $\forall x . f(x, b) = g(a, x)$. In order to find a proof of an equality problem $s = t$, an abstraction of it is build, which is an equational problem between the structures of s and t. Then the abstractions of the axioms, like (1.2), are used to equalize the structural abstractions of s and t. If such an abstract proof has been found, it is refined in order to obtain a proof of the actual equality problem $s = t$. Consider for example the equality problem $\forall y . f(y, c) = g(a, y)$ (c being a constant symbol like a and b): The equation is abstracted with respect to the common subterm y yielding the abstract equality problem

$$\langle f^1 \rangle_y = \langle g^2 \rangle_y \tag{1.3}$$

The arising task is then to move the first argument of f into the second argument of g, or vice versa. This is achieved by the application of (1.2), yielding a trivial equality problem. Thus, the plan for the proof of $\forall y . f(y, c) = g(a, y)$ is to apply an equation which behaves like specified by (1.2). In order to enable the application of $\forall x . f(x, b) = g(a, x)$ from left to right on $f(y, c)$, some work has to be done, which is to equalize $f(y, c) = \sigma(f(x, b))$, for some substitution σ. This is achieved by an equation

$$b = c \tag{1.4}$$

and then the application of $\forall x . f(x, b) = g(a, x)$ succeeds yielding a trivial equality problem. The relationship between the abstract proof and the concrete proof is viewed in figure 1.1.

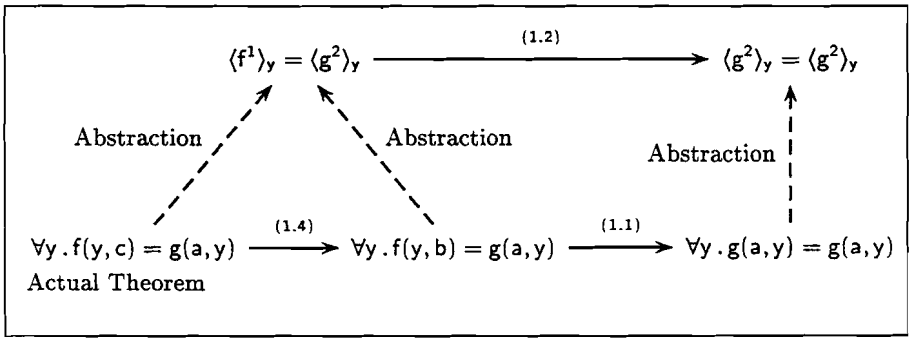


Figure 1.1: Proof and abstract proof

In the rest of this report we give a definition of the objects of the abstraction, namely the abstractions of terms and equations (Chapter 3) and try to provide a better understanding of the problems and advantages of this abstraction by some semantic considerations (Chapter 4). The remaining parts of the report are concerned with the definition of a calculus to specify the manipulation of the objects of the abstraction. Thus, a notion of substitution is introduced (Chapter 5) together with a matching algorithm for two objects of the abstraction (Chapter 6). Furthermore, a rewrite calculus is introduced which defines the manipulations of the objects (Chapter 7). Additionally, we establish some properties of the abstraction which enables an efficient implementation. Finally, we summarize the main results of this report and give an outline of some future works (Chapter 8).

Chapter 2

Preliminaries

In this chapter we introduce the notations used throughout this paper. These are the classical notations, as they are defined in [Huet and Oppen, 1980]. Then a new notion is introduced, the concept of *enriched occurences*, which is an extension of the classical notion of an occurence.

F_n	The set of function symbols of arity n
P_n	The set of predicate symbols of arity n
$\Sigma = ((F_i)_{i=0\ldots n}, (P_i)_{i=0\ldots m})$	A first order Signature
$\mathcal{V}$	A set of first order variables
$\mathcal{T}(\Sigma, \mathcal{V})$	The set of well formed first order terms over Σ and $\mathcal{V}$
$\mathsf{Eq}(\Sigma, \mathcal{V})$	The set of well formed first order equations over Σ and $\mathcal{V}$
$\mathcal{V}(t)$	The variables occurring in t.
occ(t)	The set of all *valid* occurrences of t
depth(t)	The depth of the term t, where depth(t) = 0 if $t \in \mathcal{V} \cup F_0$. I. e. the depth() of a term is the length of the longest valid occurrence of this term.
size(t)	The size of the term t, where size(t) = 1 if $t \in \mathcal{V} \cup F_0$. I. e. the size() of a term is number of symbols occurring in this term.
$t_{/\pi \leftarrow v}$	The replacement of the subterm $t_{/\pi}$ of t by v, if $\pi \in$ occ(t).
Id	the identity substitution; for any substitution σ it holds: $\sigma \cdot \mathsf{Id} = \sigma$ and $\mathsf{Id} \cdot \sigma = \sigma$.
$\sigma \leq \tau$	The substitution σ is more general than τ, i.e. there exists a substitution ρ, such that $\rho \cdot \sigma = \tau$.
2^M	the power set of the set M

As mentioned in the introduction, we are interested in an extension of *occurrences*. The idea of the extension is to add function symbols to the argument positions of an occurrence. Consider the

term $f(g(x),a)$: $\langle 1,1\rangle$ is a valid occurrence of this term denoting the subterm x. This occurrence specifies a path from the top level of $f(g(x),a)$ down to x. The function symbols occurring in $f(g(x),a)$ along this path are f and g. Thus, adding these function symbols to the occurrence $\langle 1,1\rangle$ yields $\langle f^1, g^1\rangle$. This is called an *enriched occurrence* and contains both the information about the path down to the occurrence of x and the function symbols along this path. Note that a similar extension is used for term-indexing techniques.

Definition 1 [Enriched Occurrences]
Let t *be a term and let* $\langle i_1,\ldots,i_n\rangle \in \mathsf{occ}(t)$ *be a valid occurrence of* t. *Then,* $\langle f^{i_1}_1,\ldots,f^{i_n}_n\rangle$ *is called an* enriched occurrence *of* $\langle i_1,\ldots,i_n\rangle$ *iff* $\forall 1 \leq k \leq n$ *it holds that* $t_{|\langle i_1,\ldots,i_{k-1}\rangle}$ *is of the form* $f_k(\ldots)$.
Notations: *An* enriched occurrence *corresponding to an occurrence* π *and a term* t *is given by* $\mathsf{enrich}(t,\pi)$. *The inverse function computing the corresponding occurrence to an* enriched occurrence $\tilde{\pi}$ *is given by* $|\tilde{\pi}|$.

Enriched occurrences play a major role throughout this paper. That for, some notational conventions are introduced providing a better readability. Firstly, given two enriched occurrences $\tilde{\pi} = \langle f^1, g^2\rangle$ and $\tilde{\pi}' = \langle h^1, g^1, f^2\rangle$, we agree that $\langle \tilde{\pi}, \tilde{\pi}'\rangle$ denotes the enriched occurrence

$$\langle \overbrace{f^1, g^2}^{\tilde{\pi}}, \overbrace{h^1, g^1, f^2}^{\tilde{\pi}'}\rangle$$

Furthermore, we agree that $\langle h^1, \tilde{\pi}\rangle$ denotes the enriched occurrence

$$\langle h^1, \overbrace{f^1, g^2}^{\tilde{\pi}}\rangle$$

and that $\langle \tilde{\pi}, h^1\rangle$ denotes

$$\langle \overbrace{f^1, g^2}^{\tilde{\pi}}, h^1\rangle.$$

Enriched occurrences have a strong analogy to lists and thus the notions of prefix and suffix of an enriched occurrence can be defined. For example $\langle f^1, g^2\rangle$ is a prefix of $\langle f^1, g^2, f^2\rangle$ and $\langle g^2, f^2\rangle$ is a suffix of it.

Definition 2 [Prefix, Suffix of an enriched occurrence]
Given two enriched occurrences $\tilde{\pi}$, $\tilde{\pi}'$, *then* $\tilde{\pi}'$ *is a* prefix *of* $\tilde{\pi}$, *iff there exists an enriched occurrence* $\tilde{\pi}''$, *such that* $\langle \tilde{\pi}', \tilde{\pi}''\rangle = \tilde{\pi}$. *Similarly,* $\tilde{\pi}'$ *is a* suffix *of* $\tilde{\pi}$, *iff there exists an enriched occurrence* $\tilde{\pi}''$, *such that* $\langle \tilde{\pi}'', \tilde{\pi}'\rangle = \tilde{\pi}$.
Furthermore, let $\mathsf{Prefixes}(\tilde{\pi})$ *denote the set of all prefixes of* $\tilde{\pi}$ *and* $\mathsf{Suffixes}(\tilde{\pi})$ *the set of all suffixes. Note that for some enriched occurrence* $\tilde{\pi}$, *the only enriched occurrence, being both a prefix and a suffix of* $\tilde{\pi}$ *is* $\tilde{\pi}$ *itself.*

With this definition, the $\mathcal{S}$-term construction can be defined as follows:

Definition 6.
Given a n*-ary function symbol* f *and $\mathcal{S}$-terms $\mathcal{T}_1, \ldots, \mathcal{T}_n$, we define:*

$$f(\mathcal{T}_1, \ldots, \mathcal{T}_n) := \bigcup_{i=1}^{n} f^i \cdot \mathcal{T}_i$$

For example the $\mathcal{S}$-term $\mathcal{T}_1 := \{\langle g^1 \rangle_x\}$ is a structural representation of $g(x)$ with respect to x and the $\mathcal{S}$-term $\mathcal{T}_2 := \{\langle f^1 \rangle_a\}$ is a structural representation of $f(a, b)$ with respect to a. Then the $\mathcal{S}$-term

$$f(\mathcal{T}_1, \mathcal{T}_2) = f^1 \cdot \mathcal{T}_1 \bigcup f^2 \cdot \mathcal{T}_2 = \{\langle f^1, g^1 \rangle_x, \langle f^2, f^1 \rangle_a\}$$

is a structural representation of $f(g(x), f(a, b))$.

In the previous paragraphs we often mentioned, that a $\mathcal{S}$-term represents the structure of a term. We now want to formulate this notion formally.

Definition 7 [Abstractions of a Term]
A $\mathcal{S}$-term $\mathcal{T}$ is an abstraction *of a term* t*, iff for all $\tilde{\pi}_s \in \mathcal{T}$ both $|\tilde{\pi}| \in \mathrm{occ}(t)$ and $t_{/\tilde{\pi}} = s$ hold. $\mathcal{ST}(t)$ denotes the set of all* abstractions *of* t.

Thus, in the example above the $\mathcal{S}$-term $\{\langle f^1, g^1 \rangle_x, \langle f^2, f^1 \rangle_a\}$ is an *abstraction* of the term $f(g(x), f(a, b))$. We now want to explore the relation between the subset relationship of two $\mathcal{S}$-terms and the abstractions $\mathcal{ST}(t)$ of a given term t. It turns out to be a rather simple relation: If a $\mathcal{S}$-term $\mathcal{T}'$ is a subset of a $\mathcal{S}$-term $\mathcal{T}$, then for each first order term t with structural representation $\mathcal{T}$ (i.e. an abstraction), also $\mathcal{T}'$ is a structural representation of t.

Lemma 8.
Given two $\mathcal{S}$-terms $\mathcal{T}, \mathcal{T}'$, such that $\mathcal{T}' \subseteq \mathcal{T}$ and a term t*, then $\mathcal{T} \in \mathcal{ST}(t)$ implies $\mathcal{T}' \in \mathcal{ST}(t)$.*

Proof. For all $\tilde{\pi}_s \in \mathcal{T}'$ it holds, that $\tilde{\pi}_s \in \mathcal{T}$, and therefore $|\tilde{\pi}| \in \mathrm{occ}(t)$ and $t_{/\tilde{\pi}} = s$. Thus, $\mathcal{T}' \in \mathcal{ST}(t)$ holds by definition. □

Analogously to the notion of a subterm of a first order term, the notion of a $\mathcal{S}$-subterm is introduced. For example is the $\mathcal{S}$-term $\{\langle g^1 \rangle_x\}$ a $\mathcal{S}$-subterm of

$$\{\{\langle f^1, g^1 \rangle_x, \langle f^2, f^1 \rangle_a\}$$

at the enriched occurrence $\langle f^1 \rangle$. Formally, a $\mathcal{S}$-subterm is defined by:

Definition 9 [$\mathcal{S}$-subterm]
*Given $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$ and $\tilde{\pi} \in \mathcal{OCC}(\mathcal{T})$. Then the $\mathcal{S}$-*subterm *of $\mathcal{T}$ denoted by $\tilde{\pi}$ is defined by:*

$$\mathcal{T}_{/\tilde{\pi}} := \{\tilde{\pi}'_t \mid \langle \tilde{\pi}, \tilde{\pi}' \rangle_t \in \mathcal{T}\}.$$

By abuse of notation, if π is an occurrence and there is an $\tilde{\pi} \in \mathcal{OCC}(\mathcal{T})$, such that $|\tilde{\pi}| = \pi$, then we define $\mathcal{T}_{/\pi} := \mathcal{T}_{/\tilde{\pi}}$.

In order to enable induction proofs, the notions of *depth* and *size* of a $\mathcal{S}$-term are introduced. These are defined similarly to their corresponding notions over first order terms.

Definition 10 [Depth]
The depth *of a $\mathcal{S}$-term is the maximum depth of its elements, where the depth of an element is the sum of the length of the enriched occurrence and the depth of the first order term. Thus:*

$$\mathsf{depth}(\mathcal{T}) := \max_{\tilde{\pi}_s \in \mathcal{T}} (\mathsf{length}(\tilde{\pi}) + \mathsf{depth}(s)).$$

Definition 11 [Size]
The size *of a $\mathcal{S}$-term is the sum of the sizes of each element in this $\mathcal{S}$-term. The size of a element is defined by the sum of the length of the enriched occurrence and the size of the first order term. Thus:*

$$\mathsf{size}(\mathcal{T}) := \sum_{\tilde{\pi}_s \in \mathcal{T}} (\mathsf{length}(\tilde{\pi}) + \mathsf{size}(s)).$$

To define $\mathcal{S}$-equations (cf. section 3.2) we need to know the set of terms with respect to which a given $\mathcal{S}$-term $\mathcal{T}$ has been constructed. For example has the $\mathcal{S}$-term

$$\{\langle f^1, g^1\rangle_a, \langle f^2\rangle_b\}$$

been constructed with respect to occurrences of a and b. Another example is the $\mathcal{S}$-term $\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_a\}$, which has been constructed with respect to g(x) and a. Thus, we need a function Terms computing the set of these terms:

Definition 12.
The terms *occurring in $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$ are denoted by the function* $\mathsf{Terms} : \mathcal{ST}(\Sigma, \mathcal{V}) \to 2^{\mathcal{T}(\Sigma,\mathcal{V})}$, *which is defined by:* $\mathsf{Terms}(\mathcal{T}) := \{t \mid \tilde{\pi}_t \in \mathcal{T}\}$.

In the rest of this report we will also call the elements of $\mathsf{Terms}(\mathcal{T})$ the *subterms of* $\mathcal{T}$.

3.2 $\mathcal{S}$-equations

Basically, the structural representation of a first order equation is an equation between two $\mathcal{S}$-terms. Since the abstractions of equations are especially used to provide a manipulation of the enriched occurrences to the selected subterms, we only consider those abstract equations, which left- and right-hand side $\mathcal{S}$-terms share the same subterms. This restriction is taken into account during the definition of the so-called $\mathcal{S}$-equations. Take $f(g(x), f(a, b)) = g(f(x, a))$ as an example: Then the equation

$$\{\langle f^1, g^1\rangle_x\} = \{\langle g^1, f^1\rangle_x\}$$

is a valid $\mathcal{S}$-equation, whereas

$$\{\langle f^1\rangle_{g(x)}\} = \{\langle g^1, f^1\rangle_x\} \text{ and } \{\langle f^1, g^1\rangle_x\} = \{\langle g^1, f^2\rangle_a\}$$

are not valid $\mathcal{S}$-equations, since they do not share the same subterms. Formally, the set $\mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$ of valid $\mathcal{S}$-equations is defined by:

Definition 13 [$\mathcal{S}$-equations]
The set of $\mathcal{S}$-equations over a given signature Σ and a set of variables $\mathcal{V}$ is defined by:

$$\mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V}) := \{\mathcal{Q} = \mathcal{R} \mid \mathcal{Q}, \mathcal{R} \in \mathcal{ST}(\Sigma, \mathcal{V}) \text{ and } \mathsf{Terms}(\mathcal{Q}) = \mathsf{Terms}(\mathcal{R})\}$$

Analogously to the definition of the abstractions of a first order term, the abstractions of a first order equation is defined.

Definition 14 [Abstractions of an Equation]
Given $\mathsf{q} = \mathsf{r} \in \mathsf{Eq}(\Sigma, \mathcal{V})$ *and* $\mathcal{Q} = \mathcal{R} \in \mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$, *then* $\mathcal{Q} = \mathcal{R}$ *is an* abstraction *of* $\mathsf{q} = \mathsf{r}$, *iff* $\mathcal{Q} \in \mathcal{ST}(\mathsf{q})$ *and* $\mathcal{R} \in \mathcal{ST}(\mathsf{r})$. *The set of all* abstractions *of an equation* $\mathsf{q} = \mathsf{r}$ *is denoted by* $\mathcal{S}\mathsf{Eq}(\mathsf{q} = \mathsf{r})$.

Chapter 4

Semantic Considerations

The aim of this chapter is to provide a better understanding of the $\mathcal{S}$-abstraction and the problems occurring in using this abstraction. That for possible semantics for $\mathcal{S}$-terms and $\mathcal{S}$-equations are defined according the intuition on $\mathcal{S}$-terms and the encountered problems are analyzed.

The intuition underlying the definition of the $\mathcal{S}$-abstraction is to represent sets of first order terms by $\mathcal{S}$-terms. The idea is that all first order terms sharing a specific structure can be denoted by a single $\mathcal{S}$-term. Thus, a first attempt towards a semantics of $\mathcal{S}$-terms is to define which set of terms is denoted by a given $\mathcal{S}$-term. Consider a $\mathcal{S}$-term $\{\langle \mathsf{f}^2, \mathsf{g}^1\rangle_{\mathsf{x}}\}$, where f is a binary and g is a unary function symbol. This $\mathcal{S}$-term represents any term t in which x occurs in the first argument of g and g itself occurs in the second argument of f. Therefore, $\mathsf{f}(\mathsf{a}, \mathsf{g}(\mathsf{x}))$ and $\mathsf{f}(\mathsf{f}(\mathsf{x}, \mathsf{a}), \mathsf{g}(\mathsf{x}))$ can both be represented by $\{\langle \mathsf{f}^2, \mathsf{g}^1\rangle_{\mathsf{x}}\}$ and thus both are elements of the so-called *representation set of the $\mathcal{S}$-term* $\{\langle \mathsf{f}^2, \mathsf{g}^1\rangle_{\mathsf{x}}\}$. The function assigning the corresponding *representation set* to a given $\mathcal{S}$-term is

$$\mathcal{I} : \mathcal{ST}(\Sigma, \mathcal{V}) \rightarrow 2^{\mathcal{T}(\Sigma, \mathcal{V})}$$

Formally, the *representation set* is defined by:

Definition 15 [Representation Set]
Given a $\mathcal{S}$-term $\mathcal{T}$, then the representation set of $\mathcal{T}$ *is given by the function $\mathcal{I} : \mathcal{ST}(\Sigma, \mathcal{V}) \rightarrow 2^{\mathcal{T}(\Sigma, \mathcal{V})}$ and is defined by: $\mathcal{I}(\mathcal{T}) := \{\mathsf{t} \mid \mathcal{T} \in \mathcal{ST}(\mathsf{t})\}$.*

The interpretation of a $\mathcal{S}$-term as a representation of a set of first order terms already allows one to establish some properties about $\mathcal{S}$-terms. For example, a lemma can be stated, establishing a relationship between the subset relation of two $\mathcal{S}$-terms and their respective representation sets. Consider for instance the $\mathcal{S}$-terms

$$\{\langle \mathsf{f}^2, \mathsf{g}^1\rangle_{\mathsf{x}}\} \text{ and } \{\langle \mathsf{f}^2, \mathsf{g}^1\rangle_{\mathsf{x}}, \langle \mathsf{f}^1, \mathsf{f}^2\rangle_{\mathsf{a}}\}.$$

The first $\mathcal{S}$-term represents all terms, in which x occurs in the first argument of g and g itself occurs in the second argument of f. The second $\mathcal{S}$-term represents a subset of these terms with the additional constraint, that a occurs in the second argument of f and f itself occurs in the first argument of another f. Thus, the first $\mathcal{S}$-term represents more terms than the second, and furthermore the terms represented by the second $\mathcal{S}$-term are a subset of the terms represented by the first. Thus, the statement is, that if $\mathcal{T}'$ is a subset of $\mathcal{T}$, then the *representation set* of $\mathcal{T}$ is a subset of $\mathcal{T}'$.

Lemma 16.
Given $\mathcal{T}$,$\mathcal{T}' \in \mathcal{ST}(\Sigma, \mathcal{V})$, then it holds:

$$\mathcal{T}' \subseteq \mathcal{T} \Rightarrow \mathcal{I}(\mathcal{T}) \subseteq \mathcal{I}(\mathcal{T}')$$

Proof. *Let* $t \in \mathcal{T}(\Sigma, \mathcal{V})$ *be a term. If* $\mathcal{T} \in \mathcal{ST}(t)$, *then by lemma 8* $\mathcal{T}' \in \mathcal{ST}(t)$. *Therefore,* $\{t \mid \mathcal{T} \in \mathcal{ST}(t)\} \subseteq \{t \mid \mathcal{T}' \in \mathcal{ST}(t)\}$. □

This relationship already yields a notion of *generality of a $\mathcal{S}$-term.* Indeed, does $\mathcal{T} \subseteq \mathcal{T}'$ express, that $\mathcal{T}$ is *more general* than $\mathcal{T}'$, since $\mathcal{T}$ represents more terms than $\mathcal{T}'$. Unfortunately, this notion of generality is not complete enough. For example, the $\mathcal{S}$-term $\{\langle f^1, f^1\rangle_a\}$ is more general than $\{\langle f^1\rangle_{f(a,b)}, \langle f^2\rangle_b\}$, a and b being constants and f a binary function symbol, although the first term is not a subset of the second. Hence, in order to obtain a complete definition of generality, we define the set of subsuming $\mathcal{S}$-terms of a given $\mathcal{S}$-term:

$$\mathsf{Subsuming}(\mathcal{T}) := \{\mathcal{T}' \in \mathcal{ST}(\Sigma, \mathcal{V}) \mid \mathcal{I}(\mathcal{T}) \subseteq \mathcal{I}(\mathcal{T}')\}$$

for all $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$. This set is composed of all $\mathcal{S}$-terms representing a part of the structure information represented by $\mathcal{T}$. Consider again the example above: It holds, that $\{\langle f^1, f^1\rangle_a\} \in \mathsf{Subsuming}(\{\langle f^1\rangle_{f(a,b)}, \langle f^2\rangle_b\})$. Thus, given a $\mathcal{S}$-term $\mathcal{T}$, each $\mathcal{S}$-term $\mathcal{T}' \in \mathsf{Subsuming}(\mathcal{T})$ is *more general than* $\mathcal{T}$. Note that the relation "*is more general*" is reflexive by definition, i. e. $\mathcal{T} \in \mathsf{Subsuming}(\mathcal{T})$ for any $\mathcal{S}$-term $\mathcal{T}$.

The introduction of $\mathcal{I}$ already allowed one to state some properties about $\mathcal{S}$-terms and especially allowed the definition of an $\mathcal{S}$-term to be *more general* than another. But, this is still not a semantics in a classical understanding. A classical semantics for a set of first order axioms of a given signature $\Sigma = ((F_i)_{i=0\ldots n}, (P_j)_{j=0\ldots m})$ (see [Loeckx et al., 1996]) is an algebra A, which is composed of a set A of individuals and a mapping

$$A : F_0 \to A$$

from constant symbols of the signature into the set of individuals. Additionnally, the non-constant function symbols are mapped onto total functions over this set of individuals. By abuse of notation, this is also accomplished by the function A; i.e. for a given n-ary function symbol f, A(f) is a function of $A^n \to A$. Thus, it is possible to assign to each ground term a single individual, which is the *semantics* of this term. The semantics of a ground term t is denoted by A(t). In the case of non-ground terms, additionally an assignment $b : \mathcal{V} \to A$ of the variable symbols onto the set of individuals is provided, and a term is evaluated with respect to a given variable binding. The semantics of a non-ground term t is then denoted by A[b](t). Furthermore, n-ary predicate symbols are mapped onto n-ary relations over the set of individuals and one defines, that an n-ary predicate holds for n terms, if and only if the semantics of these terms are in the relation assigned to this predicate. Thus, again by abuse of notation, for a given n-ary predicate symbol P, A(P) denotes a subset of A^n.

Then the semantics of $\mathcal{S}$-terms is defined as follows:

Definition 17 [Semantics of $\mathcal{S}$-terms]
Given a first order signature $\Sigma = ((F_i)_{i=0\ldots n}, (P_j)_{j=0\ldots m})$, A *an algebra to* Σ *and* b *an assignment, then the semantics of the* $\mathcal{S}$*-terms of* $\mathcal{ST}(\Sigma, \mathcal{V})$ *with respect to* A *and* b *is defined by:*

- $A[b](\emptyset) := A$.
- $A[b](\{\langle\rangle_t\}) := \{A[b](t)\}$, *for each term* t.
- $A[b](f(\mathcal{T}_1, \ldots, \mathcal{T}_n)) := \{A(f)(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in A[b](\mathcal{T}_1) \times \ldots \times A[b](\mathcal{T}_n)\}$, $\mathcal{T}_1, \ldots, \mathcal{T}_n$ *being* $\mathcal{S}$*-terms and* $f \in F_n$.

Note that the semantics of a $\mathcal{S}$*-term is the set of the semantics of the terms which it represents.*

This definition is conforming the intuitional semantics underlying the development of the abstraction. But a problem arises for the definition of the semantics of a $\mathcal{S}$-equation. At first, one would define the semantics of $=$ by the equality relation over the power-set of A. Then consider the following equation

$$1 + 0 = 1 \tag{4.1}$$

and the natural numbers $\mathbb{N}$ as a semantics of the signature $(\{0, 1, +\}, \{=\})$. An abstraction of this equation is

$$\{\langle +^1\rangle_1\} = \{\langle\rangle_1\} \tag{4.2}$$

The semantics of the left $\mathcal{S}$-term of (4.2) is the set $\mathbb{N} \setminus \{0\}$, and not only $\{1\}$ as is the semantics of $1+0$. On the other hand is the semantics of the right $\mathcal{S}$-term the singleton $\{1\}$, like the original term 1. The problem now is, that while $\mathbb{N}(1 + 0 = 1)$ holds, since $\mathbb{N}(1 + 0) \equiv \mathbb{N}(1) \equiv \{1\}$, the semantics of $\mathbb{N}(\{\langle +^1\rangle_1\}) \equiv \mathbb{N} \setminus \{0\}$ and $\mathbb{N}(\{\langle\rangle_1\}) \equiv \{1\}$ are different. Thus, the semantics of $=$ can not be defined as the equality relation over $2^{\mathbb{N}}$. This is not a surprising fact, since we get a degree of freedom for the semantics of the left $\mathcal{S}$-term with respect to the original term, which we did not got for the second $\mathcal{S}$-term. To illustrate this, we compare the computing of the semantics of the right $\mathcal{S}$-term with respect to the computing of the semantics of its original term:

$$\begin{aligned}
\mathbb{N}[b](\{\langle +^1\rangle_1\}) &:= \{\mathbb{N}(+)(a_1, a_n) \mid (a_1, a_2) \in \mathbb{N}[b](\langle\rangle_1) \times \mathbb{N}[b](\emptyset)\} \\
&:= \{\mathbb{N}(+)(a_1, a_n) \mid (a_1, a_2) \in \{1\} \times \mathbb{N}\} \\
&:= \{\mathbb{N}(+)(1, x) \mid x \in \mathbb{N}\} \\
&:= \mathbb{N} \setminus \{0\}
\end{aligned}$$

$$\begin{aligned}
\mathbb{N}[b](1 + 0) &:= \mathbb{N}(+)(\mathbb{N}(1), \mathbb{N}(0)) \\
&:= \mathbb{N}(+)(1, 0) \\
&:= 1
\end{aligned}$$

Note that in both cases, the semantics of the first order term is an element of the semantics of its abstraction, i. e. $\mathbb{N}(1 + 0) \in \mathbb{N}(\{\langle +^1\rangle_1\})$ and $\mathbb{N}(1) \in \mathbb{N}(\{\langle\rangle_1\})$. Based on this observation, we could define the semantics $A(=)$ as follows: Two $\mathcal{S}$-terms $\mathcal{S}$ and $\mathcal{T}$ are equal with respect to an algebra A and an assignment b, iff the intersection of the semantics of $\mathcal{S}$ and $\mathcal{T}$ is not empty, i. e. if $A[b](\mathcal{S}) \cap A[b](\mathcal{T}) \neq \emptyset$. This definition would express, that an abstract equality $\mathcal{S} = \mathcal{T}$ holds with respect to A and b, whenever there is at least one valid equation $s = t$ with respect to A and b where $s \in \mathcal{I}(\mathcal{S})$ and $t \in \mathcal{I}(\mathcal{T})$. The advantage of this semantics of $=$ is that it is strongly conforming the intuition underlying the definition of $\mathcal{S}$-terms and especially the one of $\mathcal{S}$-equations. Formally this semantics of $=$ can be defined as follows:

For all $\mathcal{S}$-terms $\mathcal{S}$ and $\mathcal{T}$ $(\mathsf{A}[\mathsf{b}](\mathcal{S}), \mathsf{A}[\mathsf{b}](\mathcal{T})) \in \mathsf{A}(=)$ iff $\mathsf{A}[\mathsf{b}](\mathcal{S}) \cap \mathsf{A}[\mathsf{b}](\mathcal{T}) \neq \emptyset$. We denote this by $\mathsf{A}, \mathsf{b} \models \mathcal{S} = \mathcal{T}$.

Unfortunately, the defined semantics of $=$ is not an equivalence relation, since it does not satisfy the transitivity property. Take for example the two $\mathcal{S}$-equations $\mathcal{S} = \mathcal{T}$ and $\mathcal{T} = \mathcal{U}$: If for a given algebra A and a given assignment b both

$$\mathsf{A}, \mathsf{b} \models \mathcal{S} = \mathcal{T} \text{ and } \mathsf{A}, \mathsf{b} \models \mathcal{T} = \mathcal{U}$$

hold, then we only know, that there exists $\mathsf{s} \in \mathcal{I}(\mathcal{S})$, $\mathsf{t}, \mathsf{t}' \in \mathcal{I}(\mathcal{T})$ and $\mathsf{u} \in \mathcal{I}(\mathcal{U})$, for which it holds

$$\mathsf{A}, \mathsf{b} \models \mathsf{s} = \mathsf{t} \text{ and } \mathsf{A}, \mathsf{b} \models \mathsf{t}' = \mathsf{u}.$$

But this does not imply, that $\mathsf{A}, \mathsf{b} \models \mathsf{s} = \mathsf{u}$, since we do not know anything about the validity of $\mathsf{t} = \mathsf{t}'$. This is the reason why using this abstraction leads to false proof plans. Indeed, if we apply $\mathcal{S}$-equations as if the semantics of $=$ would be a congruence relation, we implicitly assume that we always can prove the equality of two terms s and t both belonging to a same $\mathcal{S}$-term $\mathcal{T}$, like the equality of t and t' in the example above. Of course this assumption is wrong and thus we may obtain false proof plans when using this abstraction. The interesting point here is, that this wrong assumption is the only reason why we may get false proof plans.

We have shown, that it is not reasonable to provide a formal semantics for $\mathcal{S}$-terms in the classical meaning. But we think, that both the attempt of a formal definition of a semantics and the analysis of the occurring problems provided a better understanding of the given abstraction to the reader.

Chapter 5

$\mathcal{S}$-substitutions

In this section the notion of substitutions on $\mathcal{S}$-terms and $\mathcal{S}$-equations is defined. These so called $\mathcal{S}$-substitutions have to address the facilities of classical first order logic substitutions, in effect the substitution of variables by first order terms, and further the instantiation of variables by $\mathcal{S}$-terms. Consider the $\mathcal{S}$-term $\{\langle \mathsf{f}^1, \mathsf{g}^1 \rangle_{\mathsf{x}}\}$, where x is a variable, f a binary and g a unary function symbol. This $\mathcal{S}$-term is an abstraction of $\mathsf{f}(\mathsf{g}(\mathsf{x}), \mathsf{b})$, b being a constant function symbol. We want to have the possibility to instantiate x by a term, say $\mathsf{g}(\mathsf{a})$, in order to get the $\mathcal{S}$-term

$$\{\langle \mathsf{f}^1, \mathsf{g}^1 \rangle_{\mathsf{g}(\mathsf{a})}\}.$$

This $\mathcal{S}$-term is an abstraction of $\mathsf{f}(\mathsf{g}(\mathsf{g}(\mathsf{a})), \mathsf{b})$. That for, we need a classical first order substitution part in an $\mathcal{S}$-substitution. On the other hand we want the possibility of instantiating x by another $\mathcal{S}$-term, say $\{\langle \mathsf{f}^2 \rangle_{\mathsf{a}}\}$ which would yield

$$\{\langle \mathsf{f}^1, \mathsf{g}^1, \mathsf{f}^2 \rangle_{\mathsf{a}}\},$$

which is an abstraction of $\mathsf{f}(\mathsf{g}(\mathsf{f}(\mathsf{b}, \mathsf{a})), \mathsf{b})$. That for, in a $\mathcal{S}$-substitution we need a part in order to instantiate variables by $\mathcal{S}$-terms. Thus, a $\mathcal{S}$-substitution is composed by a classical first order logic substitution $\sigma^{\mathcal{V}}$ and a so called *pure $\mathcal{S}$-substitution* $\sigma^{\mathcal{V}}_{\mathcal{S}}$, which is defined in the following section. Furthermore we have to ensure, that classical substitution and the pure $\mathcal{S}$-substitution are compatible in that there is no x in the domain of the classical substitution, which is in the domain of the pure $\mathcal{S}$-substitution too.

5.1 Definition and Properties

Pure $\mathcal{S}$-substitutions have to address the instantiation of variables by $\mathcal{S}$-terms. Therefore, a *pure $\mathcal{S}$-substitution* is defined as a partial function $\sigma^{\mathcal{V}}_{\mathcal{S}} : \mathcal{V} \hookrightarrow \mathcal{ST}(\Sigma, \mathcal{V})$ mapping variables onto $\mathcal{S}$-terms. This function has to be a partial function, since the variables, which remain unchanged by the pure $\mathcal{S}$-substitution, can not be mapped into $\mathcal{ST}(\Sigma, \mathcal{V})$.

Definition 18 [Pure $\mathcal{S}$-substitution]
Given a partial function $\sigma^{\mathcal{V}}_{\mathcal{S}} : \mathcal{V} \hookrightarrow \mathcal{ST}(\Sigma, \mathcal{V})$. The domain $\mathsf{dom}(\sigma^{\mathcal{V}}_{\mathcal{S}})$ of this partial function is the subset of $\mathcal{V}$, for which $\sigma^{\mathcal{V}}_{\mathcal{S}}$ is defined. If $\mathsf{dom}(\sigma^{\mathcal{V}}_{\mathcal{S}})$ is finite, then $\sigma^{\mathcal{V}}_{\mathcal{S}}$ is called a pure $\mathcal{S}$-substitution.

A $\mathcal{S}$-substitution is composed of a first order logic substitution $\sigma^{\mathcal{V}}$ and a pure $\mathcal{S}$-substitution $\sigma^{\mathcal{V}}_{\mathcal{S}}$. The domains of both substitutions are required not to share same variables, since a $\mathcal{S}$-substitution has to be a function.

Definition 19 [$\mathcal{S}$-substitution]
A $\mathcal{S}$-substitution is defined as a pair of functions $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$, where $\sigma^{\mathcal{V}} : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ is a first order logic substitution and $\sigma_{\mathcal{S}}^{\mathcal{V}}$ is a pure $\mathcal{S}$-substitution, such that $\mathsf{dom}(\sigma^{\mathcal{V}}) \cap \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) = \emptyset$. The domain of the $\mathcal{S}$-substitution $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$ is defined by $\mathsf{dom}((\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})) := \mathsf{dom}(\sigma^{\mathcal{V}}) \cup \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}})$. For $\mathcal{S}$-substitutions a notation analogous to the one of first order logic substitution is introduced: A $\mathcal{S}$-substitution $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$, where $\mathsf{dom}(\sigma^{\mathcal{V}}) = \{x_1, \ldots, x_n\}$ and $\mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) = \{y_1, \ldots, y_m\}$, is denoted by

$$([x_1 \leftarrow \sigma^{\mathcal{V}}(x_1), \ldots, x_n \leftarrow \sigma^{\mathcal{V}}(x_n)], [y_1 \leftarrow \sigma_{\mathcal{S}}^{\mathcal{V}}(y_1), \ldots, y_m \leftarrow \sigma_{\mathcal{S}}^{\mathcal{V}}(y_m)])$$

The notion of a $\mathcal{S}$-substitution $\sigma_{\mathcal{V}}$ can be homomorphically extended to a function $\sigma : \mathcal{ST}(\Sigma, \mathcal{V}) \to \mathcal{ST}(\Sigma, \mathcal{V})$. The application of the homomorphic extension $\sigma = (\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$ on a $\mathcal{S}$-term $\mathcal{T}$ is divided into five cases. This complexity in the definition results from the problem of mixing the notions of first order logic substitutions and pure $\mathcal{S}$-substitutions. The first case deals with $\mathcal{S}$-terms which are either empty or are composed of a single $\langle\rangle_t$, where there are no variables in t belonging to the domain of $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$. In this case the $\mathcal{S}$-term remains unchanged.

The second case deals with $\mathcal{S}$-terms, which are composed of a single $\langle\rangle_t$, where in t occur only variables of the domain of the first order substitution $\sigma^{\mathcal{V}}$. In this case the application of σ applies $\sigma^{\mathcal{V}}$ on t and yields $\{\langle\rangle_{\sigma^{\mathcal{V}}(t)}\}$.

The third case deals with $\mathcal{S}$-terms composed of a single $\langle\rangle_x$, where x is a variable symbol occurring in the domain of the pure substitution part $\sigma_{\mathcal{S}}^{\mathcal{V}}$ of σ. In this case the substituted $\mathcal{S}$-term of x by $\sigma_{\mathcal{S}}^{\mathcal{V}}$ is the result of the application of σ.

The fourth case deals with $\mathcal{S}$-terms of the form $\{\langle\rangle_{f(t_1,\ldots,t_n)}\}$, where variables of the domain of the pure substitution $\sigma_{\mathcal{S}}^{\mathcal{V}}$ of σ occur in $f(t_1, \ldots, t_n)$. In this case, $f(t_1, \ldots, t_n)$ is decomposed into $\mathcal{S}$-terms $\{\langle\rangle_{t_i}\}$ and the substitution is recursively applied on them. Thus, the result of the application of σ is $f(\sigma(\{\langle\rangle_{t_1}\}), \ldots, \sigma(\{\langle\rangle_{t_n}\}))$.

The fifth and last case deals with $\mathcal{S}$-terms which are composed of more than one pair. This implies, that the $\mathcal{S}$-term must be of the form $f(\mathcal{T}_1, \ldots, \mathcal{T}_n)$. In this case σ is recursively applied on the $\mathcal{T}_i$ and yields $f(\sigma(\mathcal{T}_1), \ldots, \sigma(\mathcal{T}_n))$.

Definition 20 [Homomorphic Extension of $\mathcal{S}$-substitutions to $\mathcal{S}$-terms]
The homomorphic extension *of a $\mathcal{S}$-substitution $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$ is a function*

$$\sigma : \mathcal{ST}(\Sigma, \mathcal{V}) \to \mathcal{ST}(\Sigma, \mathcal{V}),$$

which application on a $\mathcal{S}$-term $\mathcal{T}$ is defined by

$$\begin{array}{ll}
\mathcal{T}, & \text{if } \mathcal{T} = \emptyset \text{ or } \mathcal{T} = \{\langle\rangle_t\} \text{ and } \mathcal{V}(t) \cap \mathsf{dom}(\sigma) = \emptyset \\
\{\langle\rangle_{\sigma^{\mathcal{V}}(t)}\} & \text{if } \mathcal{T} = \{\langle\rangle_t\} \text{ and } \mathcal{V}(t) \cap \mathsf{dom}(\sigma^{\mathcal{V}}) \neq \emptyset \text{ and } \mathcal{V}(t) \cap \\
 & \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) = \emptyset \\
\sigma_{\mathcal{S}}^{\mathcal{V}}(x), & \text{if } \mathcal{T} = \{\langle\rangle_x\},\ x \in \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) \\
f(\sigma(\{\langle\rangle_{t_1}\}), \ldots, \sigma(\{\langle\rangle_{t_n}\})), & \text{if } \mathcal{T} = \{\langle\rangle_{f(t_1,\ldots,t_n)}\},\ \mathcal{V}(t) \cap \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) \neq \emptyset \\
f(\sigma(\mathcal{T}_1), \ldots, \sigma(\mathcal{T}_n)), & \text{if } \mathcal{T} = f(\mathcal{T}_1, \ldots, \mathcal{T}_n).
\end{array}$$

Similarly, the application *of σ on a $\mathcal{S}$-equation $\mathcal{U} = \mathcal{R}$ yields the $\mathcal{S}$-equation $\sigma(\mathcal{U}) = \sigma(\mathcal{R})$.*

The following lemma ensures that the definition of the homomorphic extension of a $\mathcal{S}$-substitution is complete, i.e., that the homomorphic extension is defined for every $\mathcal{S}$-term $\mathcal{T}$.

Lemma 21.
The definition of the homorphic extension σ of $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$ is complete.

Proof. *The proof is done by induction over the size of a $\mathcal{S}$-term $\mathcal{T}$.*

Base case 1 – $\mathsf{size}(\mathcal{T}) = 0$: *This means, that $\mathcal{T} = \emptyset$, and σ is defined for empty $\mathcal{S}$-terms in its first case.*

Base case 2 – $\mathsf{size}(\mathcal{T}) = 1$: *Thus, $\mathcal{T} = \{\langle\rangle_a\}$, where a is either a variable or a constant. If a is a constant or a variable not occurring in the domain of σ then σ is defined on it in its first case. If it is a variable occurring in the domain of σ, then it occurs either in the domain of $\sigma^{\mathcal{V}}$ or in the domain of $\sigma_{\mathcal{S}}^{\mathcal{V}}$. In both cases the application of σ is defined.*

Induction step – $\mathsf{size}(\mathcal{T}) > 1$: *In this case $\mathcal{T}$ is either composed of more than one pair and σ is defined on it in its fifth case, or it is composed of a single pair $(f(t_1, \ldots, t_n), \langle\rangle)$. Then, either there is not any variable in $f(t_1, \ldots, t_n)$ occurring in the domain of σ, or there are variables occurring either in the domain of $\sigma^{\mathcal{V}}$ or $\sigma_{\mathcal{S}}^{\mathcal{V}}$. In all three cases the application of σ is defined, respectively the first, the second and the fourth case.* □

For sake of simplicity the homomorphic extension of $\mathcal{S}$-substitution $(\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$ will be denoted by $(\sigma, \sigma_{\mathcal{S}})$. Furthermore, a $\mathcal{S}$-substitution and its homomorphic extension are no longer differentiated.

The following lemma ensures the soundness of definition 20, in that it states that the result of the application of a $\mathcal{S}$-substitution on a $\mathcal{S}$-term also is a $\mathcal{S}$-term.

Lemma 22 [Soundness of Definition 20]
For every $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$, every $\mathcal{U} = \mathcal{R} \in \mathcal{SEq}(\Sigma, \mathcal{V})$ and every $\mathcal{S}$-substitution σ it holds:

(a) $\sigma(\mathcal{T}) \in \mathcal{ST}(\Sigma, \mathcal{V})$ *and*

(b) $\sigma(\mathcal{U} = \mathcal{R}) \in \mathcal{SEq}(\Sigma, \mathcal{V})$.

Proof of 22.(a). *In each non-recursive case of the application of a $\mathcal{S}$-substitution a valid $\mathcal{S}$-term is yielded. In each recursive case, assuming the recursive cases yield $\mathcal{S}$-terms, a valid $\mathcal{S}$-term is yielded. Thus, the definition is sound.* □

Proof of 22.(b). *From 22.(a) it is known, that $\sigma(\mathcal{U}), \sigma(\mathcal{R}) \in \mathcal{ST}(\Sigma, \mathcal{V})$. It remains to prove, that $\mathsf{Terms}(\sigma(\mathcal{U})) = \mathsf{Terms}(\sigma(\mathcal{R}))$. That for, we will first prove that*

$$\begin{aligned}\mathsf{Terms}(\sigma(\mathcal{T})) = \; & \{t \in \mathsf{Terms}(\mathcal{T}) \mid \mathcal{V}(t) \cap \mathsf{dom}(\sigma) = \emptyset\} \cup \\ & \{\sigma^{\mathcal{V}}(t) \mid t \in \mathsf{Terms}(\mathcal{T}), \mathcal{V}(t) \cap \mathsf{dom}(\sigma) \neq \emptyset, \mathcal{V}(t) \cap \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) = \emptyset\} \cup \\ & \{s \in \mathsf{Terms}(\sigma_{\mathcal{S}}^{\mathcal{V}}(\{\langle\rangle_t\})) \mid t \in \mathsf{Terms}(\mathcal{T}), \mathcal{V}(t) \cap \mathsf{dom}(\sigma) \neq \emptyset, \\ & \qquad \mathcal{V}(t) \cap \mathsf{dom}(\sigma_{\mathcal{S}}^{\mathcal{V}}) \neq \emptyset\}\end{aligned}$$

from which follows the attended statement directly. That for let $\sigma = (\sigma^{\mathcal{V}}, \sigma_{\mathcal{S}}^{\mathcal{V}})$ be a $\mathcal{S}$-substitution. The proof will be done by induction over the size of $\mathcal{T}$.

Base case 1 – $\mathsf{size}(\mathcal{T}) = 0$: *In this case $\mathcal{T} = \emptyset$ and thus $\sigma(\mathcal{T}) = \emptyset$ and the statement holds trivially.*

Base case 2 – $\mathsf{size}(\mathcal{T}) = 1$: *In this case $\mathcal{T}$ is of the form $\{\langle\rangle_a\}$, where a is either a variable or a constant and $\mathsf{Terms}(\mathcal{T}) = \{a\}$. If a is a constant or a variable not occurring in the domain of σ, then the right hand side reduces to $\{a\}$. On the other hand $\sigma(\mathcal{T}) = \mathcal{T}$ and thus*

$\mathsf{Terms}(\sigma(\mathcal{T})) = \{a\}$, *which proves the statement. If* a *is a variable occurring in the domain of* $\sigma^{\mathcal{V}}$, *then the right hand side reduces to* $\{\sigma^{\mathcal{V}}(a)\}$. *On the other hand is* $\sigma(\mathcal{T}) = \{\langle\rangle_{\sigma^{\mathcal{V}}(a)}\}$, *and thus* $\mathsf{Terms}(\sigma(\mathcal{T})) = \{\sigma^{\mathcal{V}}(a)\}$, *which proves the statement. If* a *occurs in the domain of* $\sigma^{\mathcal{V}}_{\mathcal{S}}$, *then the right hand side reduces to* $\mathsf{Terms}(\sigma^{\mathcal{V}}_{\mathcal{S}}(\{\langle\rangle_a\})$. *On the other hand* $\mathsf{Terms}(\sigma(\mathcal{T})) = \mathsf{Terms}(\sigma(\{\langle\rangle_a\}))$, *which proves the statement.*

Induction step – $\mathsf{size}(\mathcal{T}) > 1$: *In this case* $\mathcal{T}$ *is either of the form* $f(\mathcal{T}_1, \ldots, \mathcal{T}_n)$ *or of the form* $\{\langle\rangle_{f(t_1,\ldots,t_n)}\}$. *In the first case* $\sigma(\mathcal{T}) = f(\sigma(\mathcal{T}_1), \ldots, \sigma(\mathcal{T}_n))$, *and thus*

$$\mathsf{Terms}(\sigma(\mathcal{T})) = \mathsf{Terms}(\sigma(\mathcal{T}_1)) \cup \ldots \cup \mathsf{Terms}(\sigma(\mathcal{T}_n)).$$

Applying the induction hypothesis on the $\mathcal{T}_i$ *and simplifying proves the statement. In the second case if* $\mathcal{V}(f(t_1, \ldots, t_n)) \cap \mathsf{dom}(\sigma) = \emptyset$, *the statement holds trivially. If* $\mathcal{V}(f(t_1, \ldots, t_n)) \cap \mathsf{dom}(\sigma) \neq \emptyset$ *and* $\mathcal{V}(f(t_1, \ldots, t_n)) \cap \mathsf{dom}(\sigma^{\mathcal{V}}_{\mathcal{S}}) = \emptyset$, *then* $\sigma(\mathcal{T}) = \{\langle\rangle_{\sigma^{\mathcal{V}}(f(t_1,\ldots,t_n))}\}$ *and thus* $\mathsf{Terms}(\sigma(\mathcal{T})) = \{\sigma^{\mathcal{V}}(f(t_1, \ldots, t_n))\}$. *Simplifying the right hand side of the statement also yields* $\{\sigma^{\mathcal{V}}(f(t_1, \ldots, t_n))\}$, *which proves the statement.*
In the last case $\mathcal{V}(f(t_1, \ldots, t_n)) \cap \mathsf{dom}(\sigma) \neq \emptyset$ *and* $\mathcal{V}(f(t_1, \ldots, t_n)) \cap \mathsf{dom}(\sigma^{\mathcal{V}}_{\mathcal{S}}) \neq \emptyset$, $\sigma(\mathcal{T}) = f(\sigma(\mathcal{T}_1), \ldots, \sigma(\mathcal{T}_n))$ *and thus* $\mathsf{Terms}(\sigma(\mathcal{T})) = \mathsf{Terms}(\sigma(\mathcal{T}_1)) \cup \ldots \cup \mathsf{Terms}(\sigma(\mathcal{T}_n))$. *Applying the induction hypothesis on the* $\mathcal{T}_i$ *and simplifying proves the statement.* □

After the definition of a $\mathcal{S}$-substitution, we have to prove its soundness on $\mathcal{S}$-terms and $\mathcal{S}$-equations. The soundness of the application of a $\mathcal{S}$-substitution states that whenever a $\mathcal{S}$-substitution σ is applied on a $\mathcal{S}$-term $\mathcal{T}$, then it holds, that for every first order logic term t of which $\mathcal{T}$ is an abstraction, there is a first order substitution σ', such that $\sigma(\mathcal{T})$ is an abstraction of $\sigma(t)$. Consider the $\mathcal{S}$-term

$$\{\langle f^1\rangle_{g(x)}\}$$

as an example, where x is a variable, g a unary and f a binary function symbol. This $\mathcal{S}$-term is an abstraction of $f(g(x), y)$, y being another variable. The application of the $\mathcal{S}$-substitution

$$(\mathsf{Id}, [x \leftarrow \{\langle f^2\rangle_a\}])$$

yields $\{\langle f^1, g^1, f^2\rangle_a\}$. Take further the substitution $[x \leftarrow f(b, a), y \leftarrow b]$, b being a constant symbol. The application of this substitution onto $f(g(x), y)$ yields $f(g(f(b, a)), b)$. The $\mathcal{S}$-term resulting from the application of the $\mathcal{S}$-substitution onto the given $\mathcal{S}$-term is an abstraction of this term. This is due to the fact, that the $\mathcal{S}$-term $\{\langle f^2\rangle_a\}$ substituted for x by the $\mathcal{S}$-substitution is an abstraction of the term $f(b, a)$ substituted to x by the substitution. Thus, we need a relationship between a $\mathcal{S}$-substitution and a substitution, expressing the fact, that the $\mathcal{S}$-substitution is an *abstraction* of the substitution. That for the notion of a *representation set* $\mathcal{I}(\sigma)$ of a $\mathcal{S}$-substitution σ is introduced, which defines $\mathcal{I}(\sigma)$ as a set of first order substitutions σ'. The relationship between σ and one of these σ' is that for every $t \in \mathcal{I}(\mathcal{T})$ it holds: $\sigma'(t) \in \mathcal{I}(\sigma(\mathcal{T}))$. Furthermore, in order to prove the soundness of the $\mathcal{S}$-substitution application, we need a lemma establishing a relationship between a $\mathcal{S}$-term $\mathcal{T}$ and one of its instantiations $\sigma(\mathcal{T})$.

Definition 23 [Representation set of a $\mathcal{S}$-substitution]
Given a $\mathcal{S}$*-substitution* $\sigma = (\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})$. *The* representation set *of* σ *is the set* $\mathcal{I}(\sigma)$ *of first order substitutions* σ', *for which it holds:*

$$\sigma^{\mathcal{V}} \leq \sigma'_{|\mathsf{dom}(\sigma^{\mathcal{V}})} \text{ and } \sigma'(x) \in \mathcal{I}(\sigma^{\mathcal{V}}_{\mathcal{S}}(x)), \forall x \in \mathsf{dom}(\sigma^{\mathcal{V}}_{\mathcal{S}})$$

Using this notion of a representation set of a $\mathcal{S}$-substitution, the needed lemma for the proof of the soundness of $\mathcal{S}$-substitutions can be stated.

Lemma 24.
Given a $\mathcal{S}$-term $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$, $\mathcal{U} = \mathcal{R} \in \mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$ and a $\mathcal{S}$-substitution $\sigma = (\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})$, it holds:

(a) $\mathcal{I}(\sigma(\mathcal{T})) = \{\sigma'(t) \mid t \in \mathcal{I}(\mathcal{T}) \text{ and } \sigma' \in \mathcal{I}(\sigma)\}$

(b) $\mathcal{I}(\sigma(\mathcal{U} = \mathcal{R})) = \{\sigma'(u = r) \mid u = r \in \mathcal{I}(\mathcal{U} = \mathcal{R}) \text{ and } \sigma' \in \mathcal{I}(\sigma)\}$

Proof of 24.(a). The proof will be done by induction over the size of a $\mathcal{S}$-term $\mathcal{T}$:

Base case 1 – $\mathsf{size}(\mathcal{T}) = 0$: *In this case $\mathcal{T} = \emptyset$ and thus $\sigma(\mathcal{T}) = \emptyset$ holds for every $\mathcal{S}$-substitution σ. By definition, $\mathcal{I}(\emptyset) = \mathcal{T}(\Sigma, \mathcal{V})$, which proves the statement.*

Base case 2 – $\mathsf{size}(\mathcal{T}) = 1$: *In this case $\mathcal{T} = \{\langle\rangle_a\}$, where a is either a constant or a variable. If a is a constant or a variable not occurring in the domain of σ, then $\sigma(\mathcal{T}) = \mathcal{T}$ and for each $\sigma' \in \mathcal{I}(\sigma)$ it holds: $\sigma'(a) = a$. Thus,*

$$\begin{aligned}\{\sigma'(t) \mid t \in \mathcal{I}(\mathcal{T}) \text{ and } \sigma' \in \mathcal{I}(\sigma)\} &= \{\sigma'(t) \mid t \in \{a\} \text{ and } \sigma' \in \mathcal{I}(\sigma)\} \\ &= \{\sigma'(a) \mid \sigma' \in \mathcal{I}(\sigma)\} \\ &= \{a\} \\ &= \mathcal{I}(\mathcal{T})\end{aligned}$$

On the other hand, if a is a variable occurring in the domain of σ but not in the domain of $\sigma^{\mathcal{V}}_{\mathcal{S}}$ then $\sigma(\mathcal{T}) = \{\langle\rangle_{\sigma^{\mathcal{V}}(a)}\}$ and for each substitution $\sigma' \in \mathcal{I}(\sigma)$ it holds: $\sigma'(a) = \sigma^{\mathcal{V}}(a)$. Thus:

$$\begin{aligned}\{\sigma'(t) \mid t \in \mathcal{I}(\mathcal{T}) \text{ and } \sigma' \in \mathcal{I}(\sigma)\} &= \{\sigma'(t) \mid t \in \{a\} \text{ and } \sigma' \in \mathcal{I}(\sigma)\} \\ &= \{\sigma'(a) \mid \sigma' \in \mathcal{I}(\sigma)\} \\ &= \{\sigma^{\mathcal{V}}(a)\} \\ &= \mathcal{I}(\sigma(\mathcal{T}))\end{aligned}$$

Induction Step: $\mathsf{size}(\mathcal{T}) > 1$: *In this case $\mathcal{T}$ is either of the form $f(\mathcal{T}_1, \ldots, \mathcal{T}_n)$ or of the form $\{\langle\rangle_{f(t_1,\ldots,t_n)}\}$. In the first case $\sigma(\mathcal{T}) = f(\sigma(\mathcal{T}_1), \ldots, \sigma(\mathcal{T}_n))$ and it holds:*

$$\begin{aligned}&\{\sigma'(f(t_1, \ldots, t_n)) \mid f(t_1, \ldots, t_n) \in \mathcal{I}(\mathcal{T}) \text{ and } \sigma' \in \mathcal{I}(\sigma)\} \\ = \;&\{f(\sigma'(t_1), \ldots, \sigma'(t_n)) \mid t_i \in \mathcal{I}(\mathcal{T}_i), \forall 1 \leq i \leq n, \text{ and } \sigma' \in \mathcal{I}(\sigma)\} \\ = \;&\{f(t'_1, \ldots, t'_n) \mid t'_i \in \mathcal{I}(\sigma(\mathcal{T}_i)), \forall 1 \leq i \leq n\} \\ = \;&\{f(t'_1, \ldots, t'_n) \in \mathcal{I}(\sigma(\mathcal{T}))\} \\ = \;&\mathcal{I}(\sigma(\mathcal{T}))\end{aligned}$$

In the second case, if $\mathcal{V}(f(t_1, \ldots, t_n)) \cap \mathsf{dom}(\sigma^{\mathcal{V}}_{\mathcal{S}}) = \emptyset$, then

$$\sigma(\mathcal{T}) = \{\langle\rangle_{\sigma^{\mathcal{V}}(f(t_1,\ldots,t_n))}\}$$

and the proof is analogous to the last case of the second base case. Otherwise, it follows from definition that

$$\sigma(\mathcal{T}) = f(\sigma(\{\langle\rangle_{t_1}\}), \ldots, \sigma(\{\langle\rangle_{t_n}\}))$$

and the proof is similar to the proof in the first case of the induction step. □

Proof of 24.(b). Follows directly from part (a) of the lemma. □

5.2 Soundness of the $\mathcal{S}$-substitution

The intended meaning of the soundness of $\mathcal{S}$-substitutions is, that whenever a $\mathcal{S}$-substitution σ is applied on a $\mathcal{S}$-term $\mathcal{T}$, which is an abstraction of a first order term t, then there exists a substitution σ', such that $\sigma(\mathcal{T})$ is an abstraction of $\sigma'(t)$. This property is important for the soundness of the calculus which is defined in section 7 and formulated by the following theorem:

Theorem 25 [Soundness on $\mathcal{S}$-terms]
Given $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$, $t \in \mathcal{T}(\Sigma, \mathcal{V})$, it holds:

$$t \in \mathcal{I}(\mathcal{T}) \Rightarrow \forall\sigma\exists\sigma' \,.\, \sigma'(t) \in \mathcal{I}(\sigma(\mathcal{T}))$$

Proof. Follows directly from 24.(a). □

Obviously, the soundness of the $\mathcal{S}$-substitution also holds for $\mathcal{S}$-equations, which is the object of the following corollary.

Corollary 26 [Soundness on $\mathcal{S}$-equations]
Given $\mathcal{U} = \mathcal{R} \in \mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$ and $u = v \in \mathsf{Eq}(\Sigma, \mathcal{V})$, it holds:

$$u = v \in \mathcal{I}(\mathcal{U} = \mathcal{R}) \rightarrow \forall\sigma\exists\sigma' \,.\, \sigma'(u = v) \in \mathcal{I}(\sigma(\mathcal{U} = \mathcal{R}))$$

Proof. Follows directly from 24.(b). □

Chapter 6

$\mathcal{S}$-matching

Like in first order logic, a notion expressing the matching of a $\mathcal{S}$-term onto another is introduced. Analogeously to the first order notion of matching, the *$\mathcal{S}$-matching* of a $\mathcal{S}$-term $\mathcal{T}$ onto a $\mathcal{S}$-term $\mathcal{S}$ holds, if there exists a $\mathcal{S}$-substitution σ, such that the application of σ on $\mathcal{T}$ yields exactly $\mathcal{S}$.

Definition 27 [**$\mathcal{S}$-matching**]
*Let be $\mathcal{T}, \mathcal{S} \in \mathcal{ST}(\Sigma, \mathcal{V})$. Then $\mathcal{T}$ $\mathcal{S}$-*matches *$\mathcal{S}$, iff there is a $\mathcal{S}$-substitution σ, such that $\sigma(\mathcal{T}) = \mathcal{S}$. Such a $\mathcal{S}$-substitution σ is called an* $\mathcal{S}$-matcher *of $\mathcal{T}$ onto $\mathcal{S}$.*

Now, again like in first order logic, a procedure is required, which computes whether a $\mathcal{S}$-term $\mathcal{S}$-matches another and if so, what the required $\mathcal{S}$-substitution is. This procedure is called the *$\mathcal{S}$-matching algorithm* and is defined in the following section.

6.1 $\mathcal{S}$-matching Algorithm

The $\mathcal{S}$-matching algorithm is given by a set of transformation rules, each transforming a pair of sets $\langle \mathsf{E}, \mathsf{M} \rangle$ into a new pair $\langle \mathsf{E}', \mathsf{M}' \rangle$. The set E is a set of first order equations and the set M is a set of $\mathcal{S}$-equations. This is due to the two parts of a $\mathcal{S}$-substitution: The set E will contain the classical first order substitution and M the pure $\mathcal{S}$-substitution. A pair $\langle \mathsf{E}, \mathsf{M} \rangle$ is in a solved form, if all equations in E are of the form $\mathsf{x} = \mathsf{t}$, $\mathsf{x} \in \mathcal{V}$ and all $\mathcal{S}$-equations in E are of the form $\{\langle\rangle_{\mathsf{x}}\} = \mathcal{U}$, $\mathsf{x} \in \mathcal{V}$. Furthermore, these variables x have to occur only once in E and M. The problem whether a $\mathcal{S}$-term $\mathcal{T}$ $\mathcal{S}$-matches a $\mathcal{S}$-term $\mathcal{S}$ is encoded by $\langle \emptyset, \{\mathcal{T} = \mathcal{S}\} \rangle$. The rules defining the $\mathcal{S}$-matching algorithm are:

$\mathcal{S}$-Term-Decomposition:

$$\frac{\langle \mathsf{E}, \{\mathsf{f}(\mathcal{T}_1, \ldots, \mathcal{T}_n) = \mathsf{f}(\mathcal{S}_1, \ldots, \mathcal{S}_n)\} \cup \mathsf{M} \rangle}{\langle \mathsf{E}, \{\mathcal{T}_1 = \mathcal{S}_1, \ldots \mathcal{T}_n = \mathcal{S}_n\} \cup \mathsf{M} \rangle}$$

$\mathcal{S}$-Variable-Elimination:

$$\frac{\langle \mathsf{E}, \{\{\langle\rangle_{\mathsf{x}}\} = \mathcal{T}\} \cup \mathsf{M} \rangle}{\langle \mathsf{E}, \{\{\langle\rangle_{\mathsf{x}}\} = \mathcal{T}\} \cup \mathsf{M}[\mathsf{x} \leftarrow \mathcal{T}] \rangle}$$

if $\mathsf{x} \in \mathcal{V}$, $\mathsf{x} \notin \mathcal{V}(\mathcal{T})$, $\mathcal{T} \neq \emptyset$, $\mathsf{x} \notin \mathcal{V}(\mathsf{E})$ and x occurs[1] in M

[1] The requirement that x must occur in M is only needed to prove the termination of the $\mathcal{S}$-matching algorithm.

$\mathcal{S}$-Tautology-Elimination:

$$\frac{\langle E, \{\mathcal{T} = \mathcal{T}\} \cup M\rangle}{\langle E, M\rangle}$$

"Bridge":

$$\frac{\langle E, \{\{\langle\rangle_t\} = \{\langle\rangle_s\}\} \cup M\rangle}{\langle \{t = s\} \cup E, M\rangle}$$

where $t \notin \mathcal{V}$.

Term-Decomposition:

$$\frac{\langle \{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \cup E, M\rangle}{\langle \{t_1 = s_1, \ldots, t_n = s_n\} \cup E, M\rangle}$$

Variable-Elimination:

$$\frac{\langle \{x = t\} \cup E, M\rangle}{\langle \{x = t\} \cup E[x \leftarrow t], M[x \leftarrow t]\rangle}$$

if $x \in \mathcal{V}$, $x \notin \mathcal{V}(t)$ and $x \in E \cup M$.

Tautology-Elimination:

$$\frac{\langle \{t = t\} \cup E, M\rangle}{\langle E, M\rangle}$$

Consider the $\mathcal{S}$-terms $\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\}$ and $\{\langle f^1\rangle_{g(a)}, \langle f^2, f^1, f^2\rangle_b\}$, where x and y are variables, a and b constants, g a unary and f a duary function symbol. These two $\mathcal{S}$-terms are respectively abstractions of $f(g(x), f(y, a))$ and $f(g(a), f(f(a, b), a))$. The $\mathcal{S}$-matching algorithm is given the following pair:

$$\langle \emptyset, \{\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\} = \{\langle f^1\rangle_{g(a)}, \langle f^2, f^1, f^2\rangle_b\}\}\rangle$$

The $\mathcal{S}$-matching algorithm transforms this pair as follows:

$$\langle \emptyset, \{\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\} = \{\langle f^1\rangle_{g(a)}, \langle f^2, f^1, f^2\rangle_b\}\}\rangle$$

——— $\mathcal{S}$-term-Decomposition ———

$$\langle \emptyset, \{\{\langle\rangle_{g(x)}\} = \{\langle\rangle_{g(a)}\}, \{\langle f^1\rangle_y\} = \{\langle f^1, f^2\rangle_b\}\}\rangle$$

——— $\mathcal{S}$-term-Decomposition ———

$$\langle \emptyset, \{\{\langle\rangle_{g(x)}\} = \{\langle\rangle_{g(a)}\}, \{\langle\rangle_y\} = \{\langle f^2\rangle_b\}\}\rangle$$

——— "Bridge" ———

$$\langle \{g(x) = g(a)\}, \{\{\langle\rangle_y\} = \{\langle f^2\rangle_b\}\}\rangle$$

——— Term-Decomposition ———

$$\langle \{x = a\}, \{\{\langle\rangle_y\} = \{\langle f^2\rangle_b\}\}\rangle$$

Thus, the $\mathcal{S}$-matcher of $\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\}$ on $\{\langle f^1\rangle_{g(a)}, \langle f^2, f^1, f^2\rangle_b\}$ is

$$([x \leftarrow a], [y \leftarrow \{\langle f^2\rangle_b\}])$$

Since the $\mathcal{S}$-matching algorithm is intended to be a decision procedure about whether a $\mathcal{S}$-term $\mathcal{S}$-matches another, a few facts need to be established. The first fact is the termination of the algorithm, since otherwise it would not be a decision procedure. Furthermore, in order to be sure of always getting a correct answer, its soundness and completeness have to be established. aa

Theorem 28 [Termination of the $\mathcal{S}$-matching-Algorithm]
The $\mathcal{S}$-matching algorithm terminates for every pair of finite sets $\langle E, M\rangle$.

Proof. The statement is proved by mapping each pair $\langle E, M\rangle$ into a triple (k, l, m) of natural numbers. Then it will be shown that each rule reduces the adjoined triple with respect to the lexicographic ordering over tupels.
The components of the triple adjoined to a pair $\langle E, M\rangle$ are defined as follows:

k: the size of the set

$$\mathcal{V}(\langle E, M\rangle) \setminus \{\, x \mid \begin{array}{l} x \text{ occurs only once in } E \text{ and } M, \text{ either in an equation} \\ \text{of the kind } x = t \text{ in } E \text{ or in a } \mathcal{S}\text{-equation of the kind} \\ \{\langle\rangle_x\} = \mathcal{T}\} \text{ in } M. \end{array} \}$$

where $\mathcal{V}(\langle E, M\rangle)$ is the set of variables occurring in E in M.

l: The sum of the sizes of the left- and right-hand sides of the $\mathcal{S}$-equations in M.

m: The sum of the sizes of the left- and right-hand sides of the equations in E.

Then it is easy to see, that the $\mathcal{S}$-Term-Decomposition and the $\mathcal{S}$-Tautology-Elimination reduce the second component, without changing the first component. The Term-Decomposition and the Tautology-Elimination reduce the third component, without affecting the others. The $\mathcal{S}$-Variable-Elimination and the Variable-Elimination reduce the first component and the "Bridge" rule reduces the second component, without affecting the first, which completes the proof of the termination of the $\mathcal{S}$-matching algorithm. □

In order to make the statements about the soundness and completeness of the $\mathcal{S}$-matching algorithm, the notion of a $\mathcal{S}$-matcher of a pair $\langle E, M\rangle$ must be defined. This notion splits, like the notion of a $\mathcal{S}$-substitution, into two parts: The first part is a classical first order matcher of a set of first order equations and belongs to the set E.

Definition 29 [Matcher of a set of equations]
A substitution $\sigma^{\mathcal{V}}$ is a matcher of a set E of equations, iff for each $t = s \in E$ it holds, that $\sigma^{\mathcal{V}}(t) = s$.

The second part is a pure $\mathcal{S}$-substitution $\sigma^{\mathcal{V}}_{\mathcal{S}}$ belonging to the set M. Unfortunately, it can not be stated that for each $\mathcal{S}$-equation $\mathcal{U} = \mathcal{V}$, $\sigma^{\mathcal{V}}_{\mathcal{S}}(\mathcal{U}) = \mathcal{V}$ holds, since it does not take into account the required first order substitutions. Therefore, the notion of a *pure $\mathcal{S}$-matcher* of a set of $\mathcal{S}$-equations has to be defined relatively to a first order substitution.

Definition 30 [Pure $\mathcal{S}$-matcher of a set of $\mathcal{S}$-equations]
Given a first order substitution $\sigma^{\mathcal{V}}$, a pure $\mathcal{S}$-substitution $\sigma^{\mathcal{V}}_{\mathcal{S}}$ is a pure $\mathcal{S}$-matcher *with respect to $\sigma^{\mathcal{V}}$ of a set M of $\mathcal{S}$-equations, iff for each $\mathcal{T} = \mathcal{R} \in M$ it holds, that $(\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})(\mathcal{T}) = \mathcal{S}$.*

Now the notion of a $\mathcal{S}$-matcher of a pair $\langle E, M\rangle$ can be defined as follows:

Definition 31 [**$\mathcal{S}$-matcher of a pair $\langle E, M\rangle$**]
A $\mathcal{S}$-substitution $\sigma = (\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})$ is a $\mathcal{S}$-matcher of $\langle E, M\rangle$, iff $\sigma^{\mathcal{V}}$ is a matcher of E and $\sigma^{\mathcal{V}}_{\mathcal{S}}$ is a pure $\mathcal{S}$-matcher of M with respect to $\sigma^{\mathcal{V}}$.

The $\mathcal{S}$-matcher of a pair $\langle E, M\rangle$ which is in a solved form can be extracted from E and M in a canonical way: Each equation in E is of the form $x = t$, where x is a variable not occurring in t and any other equation of E or any $\mathcal{S}$-equation of M. Similarly, every $\mathcal{S}$-equation in M is of the form $\{\langle\rangle_x\} = \mathcal{T}$, where x is a variable not occurring in $\mathcal{T}$ or any other $\mathcal{S}$-equation in M or any equation in E. Thus the $\mathcal{S}$-matcher for $\langle E, M\rangle$ is a $\mathcal{S}$-substitution $\sigma = (\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})$ defined by:

$$\sigma^{\mathcal{V}} = \{x \leftarrow t \mid x = t \in E\}$$

$$\sigma^{\mathcal{V}}_{\mathcal{S}} = \{x \leftarrow \mathcal{T} \mid \{\langle\rangle_x\} = \mathcal{T} \in M\}$$

Obviously, σ is a $\mathcal{S}$-matcher of $\langle E, M\rangle$ and thus if the $\mathcal{S}$-matching algorithm terminates in a solved form, an $\mathcal{S}$-matcher can be extracted from it. By the soundness and completeness of the $\mathcal{S}$-matching algorithm, which will be established in the next paragraph, this $\mathcal{S}$-matcher also is a $\mathcal{S}$-matcher of the original matching problem.

In order to proof the soundness and completeness of the $\mathcal{S}$-matching algorithm, a lemma is required, which states, that if a pair $\langle E, M\rangle$ is not in a solved form, but no rule of the $\mathcal{S}$-matching algorithm is further applicable, then there does not exist a $\mathcal{S}$-matcher for $\langle E, M\rangle$.

Lemma 32.
Given a pair $\langle E, M\rangle$, then if no rule of the $\mathcal{S}$-matching algorithm is applicable on $\langle E, M\rangle$ and $\langle E, M\rangle$ is not in a solved form, then there does not exist a $\mathcal{S}$-matcher for $\langle E, M\rangle$.

Proof. If no rule is applicable on $\langle E, M\rangle$ and $\langle E, M\rangle$ is not in a solved form, then $\langle E, M\rangle$ must be of one of the following forms:

1. *$\langle E, M \cup \{f(\mathcal{T}_1, \ldots, \mathcal{T}_n) = g(\mathcal{S}_1, \ldots, \mathcal{S}_m)\}\rangle$: Obviously, there can not exist any $\mathcal{S}$-matcher in this case, since if so, $\sigma(f(\mathcal{T}_1, \ldots, \mathcal{T}_n)) = g(\mathcal{S}_1, \ldots, \mathcal{S}_m)$ should hold, which is impossible.*
2. *$\langle E, M \cup \{\{\langle\rangle_s\} = f(\mathcal{T}_1, \ldots, \mathcal{T}_n)\}\rangle$, $s \notin \mathcal{V}$: Again, there can not exist any $\mathcal{S}$-matcher in this case.*
3. *$\langle E \cup \{f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m)\}, M\rangle$: The statement holds trivially.*
4. *$\langle E \cup \{a = t\}, M\rangle$, where $t \neq a$: Again, the statement holds trivially, which completes the proof of the lemma.* □

After these preliminaries, the soundness and the completeness of the $\mathcal{S}$-matching algorithm can be stated. The completeness of the $\mathcal{S}$-matching algorithm is proved by case analysis over the rules defining the algorithm. The following theorem states for each rule transforming $\langle E, M\rangle$ into $\langle E', M'\rangle$, that whenever a $\mathcal{S}$-substitution σ is a $\mathcal{S}$-matcher of $\langle E, M\rangle$ then it is also a $\mathcal{S}$-matcher of $\langle E', M'\rangle$. Together with the lemma above it follows that if the $\mathcal{S}$-matching algorithm terminates with $\langle E', M'\rangle$ not in a solved form, then the original $\mathcal{S}$-matching problem does not have a solution.

Theorem 33 [**Completeness of the $\mathcal{S}$-matching-Algorithm**]
Given $\langle E, M\rangle$, $\langle E', M'\rangle$, such that $\langle E, M\rangle$ results from $\langle E', M'\rangle$ by the application of one of the $\mathcal{S}$-matching rules above. Then every $\mathcal{S}$-matcher $\sigma = (\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})$ of $\langle E, M\rangle$ is a $\mathcal{S}$-matcher of $\langle E', M'\rangle$.

Proof. *The proof is done by case analysis over the rules:*

$\mathcal{S}$-Term-Decomposition: *I.e.* $\mathsf{E} = \mathsf{E}'$ *and* $\mathsf{M}' = \mathsf{M} \cup \{\mathcal{T}_1 = \mathcal{S}_1 \ldots \mathcal{T}_n = \mathcal{S}_n\}$. *If* σ *is a* $\mathcal{S}$*-matcher of* $\langle \mathsf{E}, \{\mathsf{f}(\mathcal{T}_1, \ldots, \mathcal{T}_n) = \mathsf{f}(\mathcal{S}_1, \ldots, \mathcal{S}_n)\} \cup \mathsf{M} \rangle$, *then* $\sigma^{\mathcal{V}}$ *is a matcher of* E *and* $\sigma^{\mathcal{V}}_{\mathcal{S}}$ *is a pure* $\mathcal{S}$*-matcher of* M *with respect to* $\sigma^{\mathcal{V}}$. *Furthermore, it holds*

$$\sigma(\mathsf{f}(\mathcal{T}_1, \ldots, \mathcal{T}_n)) = \mathsf{f}(\mathcal{S}_1, \ldots, \mathcal{S}_n)$$

and thus it holds $\sigma(\mathcal{T}_i) = \mathcal{S}_i$, *for all* $1 \leq i \leq n$. *Thus,* σ *is a* $\mathcal{S}$*-matcher of* $\langle \mathsf{E}, \mathsf{M} \cup \{\mathcal{T}_1 = \mathcal{S}_1 \ldots \mathcal{T}_n = \mathcal{S}_n\} \rangle$.

$\mathcal{S}$-Variable-Elimination: *If* σ *is a* $\mathcal{S}$*-matcher of* $\langle \mathsf{E}, \{\{\langle\rangle_x\} = \mathcal{T}\} \cup \mathsf{M} \rangle$, *then it holds, that* $\sigma^{\mathcal{V}}_{\mathcal{S}}(\{\langle\rangle_x\}) = \mathcal{T}$, *and thus* $\sigma \circ (\emptyset, [x \leftarrow \mathcal{T}]) = \sigma$, *which proves the statement.*

$\mathcal{S}$-Tautology-Elimination: *The statement holds trivially in this case.*

"Bridge": *In this case it must hold, that* $\sigma(\{\langle\rangle_t\}) = \{\langle\rangle_s\}$. *Since* $t \notin \mathcal{V}$, *it must hold that* $\sigma^{\mathcal{V}}(t) = s$, *and thus* σ *is a* $\mathcal{S}$*-matcher of* $\langle \mathsf{E} \cup \{t = s\}, \mathsf{M} \rangle$.

Term-Decomposition: *Since* σ *is a* $\mathcal{S}$*-matcher of*

$$\langle \mathsf{E}, \{\mathsf{f}(t_1, \ldots, t_n) = \mathsf{f}(s_1, \ldots, s_n)\} \cup \mathsf{M} \rangle,$$

it holds that $\sigma^{\mathcal{V}}(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}(s_1, \ldots, s_n)$, *and thus* $\sigma^{\mathcal{V}}(t_i) = s_i$ *for each* $1 \leq i \leq n$ *and* σ *is a* $\mathcal{S}$*-matcher of* $\langle \mathsf{E} \cup \{t_1 = s_1, \ldots, t_n = s_n\} \rangle$.

Variable-Elimination: $\sigma^{\mathcal{V}}(x) = t$, *and thus* $\sigma \circ ([x \leftarrow t], \emptyset) = \sigma$ *and thus* σ *is an* $\mathcal{S}$*-matcher of* $\langle \{x = t\} \cup \mathsf{E}[x \leftarrow t], \mathsf{M}[x \leftarrow t] \rangle$.

Tautology-Elimination: *The statement holds trivially in this case.* □

The above theorem implies the completeness of the $\mathcal{S}$-matching algorithm; indeed, we already showed the termination of the $\mathcal{S}$-matching algorithm. Thus, a $\mathcal{S}$-matching problem $\langle \mathsf{E}, \mathsf{M} \rangle$ is transformed in finitely many steps into a $\mathcal{S}$-matching problem $\langle \mathsf{E}', \mathsf{M}' \rangle$. If $\langle \mathsf{E}', \mathsf{M}' \rangle$ is in a solved form, then $\langle \mathsf{E}', \mathsf{M}' \rangle$ has a unique $\mathcal{S}$-matcher; otherwise there does not exist any $\mathcal{S}$-matcher for $\langle \mathsf{E}', \mathsf{M}' \rangle$. Thus, if the original problem $\langle \mathsf{E}, \mathsf{M} \rangle$ has a $\mathcal{S}$-matcher it follows by induction over the transformation steps and from the theorem above, that $\langle \mathsf{E}', \mathsf{M}' \rangle$ has a $\mathcal{S}$-matcher and thus must be in a solved form. Therefore, the $\mathcal{S}$-matching algorithm is complete.

The soundness of the $\mathcal{S}$-matching algorithm is also proved by case analysis over the rules defining the $\mathcal{S}$-matching algorithm. On contrary to the completeness, it is proved that given $\langle \mathsf{E}, \mathsf{M} \rangle$ and $\langle \mathsf{E}', \mathsf{M}' \rangle$ if $\langle \mathsf{E}, \mathsf{M} \rangle$ does not have a $\mathcal{S}$-matcher and $\langle \mathsf{E}', \mathsf{M}' \rangle$ results by the application of a transformation rule to $\langle \mathsf{E}, \mathsf{M} \rangle$, then $\langle \mathsf{E}', \mathsf{M}' \rangle$ does not have a $\mathcal{S}$-matcher too.

Theorem 34 [Soundness of the $\mathcal{S}$-matching-Algorithm]
Given $\langle \mathsf{E}, \mathsf{M} \rangle$, $\langle \mathsf{E}', \mathsf{M}' \rangle$, *such that* $\langle \mathsf{E}, \mathsf{M} \rangle$ *results from* $\langle \mathsf{E}', \mathsf{M}' \rangle$ *by the application of one of the* $\mathcal{S}$*-matching rules above. Then every* $\mathcal{S}$*-matcher* $\sigma = (\sigma^{\mathcal{V}}, \sigma^{\mathcal{V}}_{\mathcal{S}})$ *of* $\langle \mathsf{E}', \mathsf{M}' \rangle$ *is a* $\mathcal{S}$*-matcher of* $\langle \mathsf{E}, \mathsf{M} \rangle$.

Proof. *The proof is done by case analysis over the rules:*

$\mathcal{S}$-**Term-Decomposition:** *If* σ *is a* $\mathcal{S}$*-matcher of* $\langle E, \{\mathcal{T}_1 = \mathcal{S}_1, \ldots, \mathcal{T}_n = \mathcal{S}_n\} \cup M\rangle$*, then* $\sigma^{\mathcal{V}}_{\mathcal{S}}(\mathcal{T}_i) = \mathcal{S}_i$ *for each* $1 \leq i \leq n$*. Thus* $\sigma^{\mathcal{V}}_{\mathcal{S}}(f(\mathcal{T}_1, \ldots, \mathcal{T}_n)) = f(\mathcal{S}_1, \ldots, \mathcal{S}_n)$ *and* σ *is a* $\mathcal{S}$*-matcher of* $\langle E, \{f(\mathcal{T}_1, \ldots, \mathcal{T}_n) = f(\mathcal{S}_1, \ldots, \mathcal{S}_n)\} \cup M\rangle$.

$\mathcal{S}$-**Variable-Elimination:** *Since* $\sigma(\{\langle\rangle_x\}) = \mathcal{T}$, $\sigma \circ (\emptyset, [x \leftarrow \mathcal{T}]) = \sigma$*. Thus,* σ *is a* $\mathcal{S}$*-matcher of* $\langle E, \{\{\langle\rangle_x\} = \mathcal{T}\} \cup M\rangle$ *too.*

$\mathcal{S}$-**Tautology-Elimination:** *The statement holds trivially.*

"Bridge": *It holds that* $\sigma^{\mathcal{V}}(t) = s$*. Therefore,* $\sigma(\{\langle\rangle_t\}) = \{\langle\rangle_s\}$ *and thus* σ *is a* $\mathcal{S}$*-matcher of* $\langle E, \{\{\langle\rangle_t\} = \{\langle\rangle_s\}\} \cup M\rangle$ *too.*

Term-Decomposition: *Since* $\sigma^{\mathcal{V}}(t_i) = s_i$ *for each* $1 \leq i \leq n$*, it holds that* $\sigma^{\mathcal{V}}(f(t_1, \ldots, t_n)) = f(s_1, \ldots, s_n)$*. Thus,* σ *is a* $\mathcal{S}$*-matcher of*

$$\langle E \cup \{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\}, M\rangle.$$

Variable-Elimination: *Since* $\sigma^{\mathcal{V}}(x) = t$*, it holds that* $\sigma = \sigma \circ ([x \leftarrow t], \emptyset)$*. Thus,* σ *is a* $\mathcal{S}$*-matcher of* $\langle E \cup \{x = t\}, M\rangle$.

Tautology-Elimination: *The statement holds trivially.* □

We now show that this theorem implies the soundness of the $\mathcal{S}$-matching algorithm. Assume again that the original $\mathcal{S}$-matching problem $\langle E, M\rangle$ is transformed by finitely many steps into a $\langle E', M'\rangle$ on which no further transformation rules are applicable. Thus, if $\langle E, M\rangle$ does not have a $\mathcal{S}$-matcher, then $\langle E', M'\rangle$ does not have a $\mathcal{S}$-matcher and thus must be in an unsolved form. Thus, the $\mathcal{S}$-matching algorithm is sound.

Chapter 7

$\mathcal{S}$-Calculus

In this section a calculus for the $\mathcal{S}$-abstraction is introduced, the so-called $\mathcal{S}$-Calculus. This calculus is a rewrite calculus and has only a single rule. This rewrite rule describes how an $\mathcal{S}$-equation can be applied on a $\mathcal{S}$-term. In order to define this rule, the notion of replacement of a subterm of an $\mathcal{S}$-term by another is introduced.

Definition 35 [**$\mathcal{S}$-subterm-Replacement**]
Given $\mathcal{T}, \mathcal{R} \in \mathcal{ST}(\Sigma, \mathcal{V})$ and $\tilde{\pi} \in \mathcal{OCC}(\mathcal{T})$, the replacement *of $\mathcal{T}_{|\tilde{\pi}}$ by $\mathcal{R}$ is defined by*

- *if $\tilde{\pi} = \langle\rangle$, then $\mathcal{T}_{|\langle\rangle \leftarrow \mathcal{R}} = \mathcal{R}$,*
- *if $\tilde{\pi} = \langle \mathsf{f}^i, \tilde{\pi}' \rangle$ and $\mathcal{T} = \mathsf{f}(\mathcal{T}_1, \ldots, \mathcal{T}_n)$, then*

$$\mathcal{T}_{|\langle \mathsf{f}^i, \tilde{\pi}' \rangle \leftarrow \mathcal{R}} = \mathsf{f}(\mathcal{T}_1, \ldots, \mathcal{T}_{i|\tilde{\pi}' \leftarrow \mathcal{R}}, \ldots, \mathcal{T}_n).$$

Later on the soundness of the abstract rewrite rule is stated and for its proof the soundness of the $\mathcal{S}$-subterm-replacement is required. The soundness property of the $\mathcal{S}$-subterm-replacement states, that if two terms t and r are respectively in $\mathcal{I}(\mathcal{T})$ and $\mathcal{I}(\mathcal{R})$, then the term $\mathsf{t}_{|\tilde{\pi} \leftarrow \mathsf{r}}$ is in the $\mathcal{S}$-term resulting from the replacement of $\mathcal{T}_{|\tilde{\pi}}$ by $\mathcal{R}$.

Lemma 36 [**Soundness of the $\mathcal{S}$-subterm-Replacement**]
Given $\mathcal{T}, \mathcal{R} \in \mathcal{ST}(\Sigma, \mathcal{V})$, $\mathsf{t}, \mathsf{r} \in \mathcal{T}(\Sigma, \mathcal{V})$ and $\tilde{\pi} \in \mathcal{OCC}(\mathcal{T})$, it holds:

$$\mathcal{I}(\mathcal{T}_{|\tilde{\pi} \leftarrow \mathcal{R}}) = \{\mathsf{t}_{|\tilde{\pi} \leftarrow \mathsf{r}} \mid \mathsf{t} \in \mathcal{I}(\mathcal{T}) \text{ and } \mathsf{r} \in \mathcal{I}(\mathcal{R})\}$$

Proof. The proof is done by induction over the length of $\tilde{\pi}$.

Base Case $\mathsf{length}(\tilde{\pi}) = 0$: *i.e. $\tilde{\pi} = \langle\rangle$ and $\mathcal{T}_{|\tilde{\pi}} = \mathcal{T}$. Then $\mathcal{T} = \mathcal{R}$ and $\mathcal{T}_{|\tilde{\pi} \leftarrow \mathcal{R}} = \mathcal{R}$, and thus the statement holds trivially.*

Induction Step $\mathsf{length}(\tilde{\pi}) > 0$: *i.e. $\tilde{\pi} = \langle \mathsf{f}^i, \tilde{\pi}' \rangle$. Then let be $\mathcal{T}' := \mathcal{T}_{|\langle \mathsf{f}^i \rangle}$. Then the induction hypotheses applies on $\mathcal{T}'$ with $\tilde{\pi}'$ and it holds:*

$$\mathsf{t}' \in \mathcal{I}(\mathcal{T}') \wedge \mathsf{r} \in \mathcal{I}(\mathcal{R}) \Rightarrow \mathsf{t}'_{|\tilde{\pi}' \leftarrow \mathsf{r}} \in \mathcal{I}(\mathcal{T}'_{|\tilde{\pi} \leftarrow \mathcal{R}})$$

Let be $\mathcal{T}'' := \mathcal{T}'_{|\tilde{\pi}\leftarrow\mathcal{R}}$. *Then the induction hypotheses applies on* $\mathcal{T}$ *with* $\langle \mathsf{f}^i \rangle$ *and it holds furthermore:*

$$\mathsf{t} \in \mathcal{I}(\mathcal{T}) \wedge \mathsf{t}'' \in \mathcal{I}(\mathcal{T}'') \Rightarrow \mathsf{t}_{|\langle i\rangle\leftarrow \mathsf{t}''} \in \mathcal{I}(\mathcal{T}_{|\langle \mathsf{f}^i\rangle\leftarrow\mathcal{T}''})$$

Given $\mathsf{t} \in \mathcal{I}(\mathcal{T})$ *and* $\mathsf{r} \in \mathcal{R}$. *Then* t *is of the form* $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_n)$ *and* $\mathsf{t}_i \in \mathcal{I}(\mathcal{T}')$. *Thus, by the first implication,* $\mathsf{t}_{i|\tilde{\pi}\uparrow\leftarrow \mathsf{r}} \in \mathcal{I}(\mathcal{T}'_{\tilde{\pi}'\leftarrow\mathcal{R}})$, *and by the second implication, it holds:*

$$\mathsf{t}_{|\langle i\rangle\leftarrow \mathsf{t}_{i|\tilde{\pi}\uparrow\leftarrow \mathsf{r}}} \in \mathcal{I}(\mathcal{T}_{|\langle \mathsf{f}^i\rangle\leftarrow\mathcal{T}'_{\tilde{\pi}'\leftarrow\mathcal{R}}}),$$

which completes the proof.

Based on the notion of $\mathcal{S}$-subterm-replacement, the $\mathcal{S}$-Rewriting can be defined. Analogously to the first order rewriting, the $\mathcal{S}$-Rewriting is a combination of $\mathcal{S}$-matching and $\mathcal{S}$-subterm-replacement.

Definition 37 [$\mathcal{S}$-Rewriting]
Given $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$, $\mathcal{U} = \mathcal{R} \in \mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$ *and* $\tilde{\pi} \in \mathcal{OCC}(\mathcal{T})$, *then* $\mathcal{U} = \mathcal{R}$ *is* applicable *on* $\mathcal{T}$ *at the enriched occurence* $\tilde{\pi}$, *iff there is a* $\mathcal{S}$*-substitution* σ, *such that* $\mathcal{T}_{|\tilde{\pi}} = \sigma(\mathcal{U})$. *The result of the application is* $\mathcal{T}_{|\tilde{\pi}\leftarrow\sigma(\mathcal{V})}$.

Theorem 38 [Soundness of $\mathcal{S}$-Rewriting]
Given $\mathcal{T} \in \mathcal{ST}(\Sigma, \mathcal{V})$, $\mathcal{U} = \mathcal{R} \in \mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$, $\mathsf{t} \in \mathcal{T}(\Sigma, \mathcal{V})$, $\mathsf{u} = \mathsf{r} \in \mathsf{Eq}(\Sigma, \mathcal{V})$, $\tilde{\pi} \in \mathcal{OCC}(\mathcal{T})$ *and a* $\mathcal{S}$*-substitution* σ, *such that* $\sigma(\mathcal{U}) = \mathcal{T}_{|\tilde{\pi}}$, *then it holds:*

$$\mathsf{t} \in \mathcal{I}(\mathcal{T}) \wedge \mathsf{u} = \mathsf{r} \in \mathcal{I}(\mathcal{U} = \mathcal{R}) \Rightarrow \mathsf{t}_{|\tilde{\pi}\leftarrow\sigma'(\mathsf{r})} \in \mathcal{I}(\mathcal{T}_{|\tilde{\pi}\leftarrow\sigma(\mathcal{R})}),$$

where $\mathsf{r} \in \mathcal{I}(\mathcal{R})$ *and* $\sigma' \in \mathcal{I}(\sigma)$.

Proof. The statement follows from corollary 26 and lemma 36.

Note that in the theorem above it does not hold in general, that $\sigma'(\mathsf{u}) = \mathsf{t}_{|\tilde{\pi}}$, since only the abstractions of $\mathsf{t}_{|\tilde{\pi}}$ and $\sigma'(\mathsf{u})$ are equal.

The problem arising when trying to apply an $\mathcal{S}$-equation $\mathcal{U} = \mathcal{R}$ on an $\mathcal{S}$-term $\mathcal{T}$ is that $\mathcal{U}$ may not be abstract enough to be applied on $\mathcal{T}$. Indeed, it could be possible that there exists an $\mathcal{S}$-substitution σ, such that $\mathcal{T} \subseteq \sigma(\mathcal{U})$. But since the $\mathcal{S}$-matching has been defined as the equality of $\sigma(\mathcal{U})$ and $\mathcal{T}$, we need a technique which allows to compute an abstraction $\mathcal{U}'$ of $\mathcal{U}$, such that if $\mathcal{T} \subseteq \sigma(\mathcal{U})$ holds, then $\sigma(\mathcal{U}') = \mathcal{T}$ holds. It will be showed that $\mathcal{U}'$ is unique and may be computed before the $\mathcal{S}$-matching process. The computation of $\mathcal{U}'$ up from $\mathcal{U}$ is called *preprocessing.*

7.1 Preprocessing

The preprocessing algorithm is given by a set of transformation rules. The algorithm takes two $\mathcal{S}$-terms $\mathcal{T}$ and $\mathcal{S}$ as arguments and yields in case of success an abstraction of $\mathcal{T}$ which is the only abstraction of $\mathcal{T}$ possibly $\mathcal{S}$-matching $\mathcal{S}$. The transformation rules are operating on sets of triples $\langle \mathcal{T}', \mathcal{S}', \tilde{\pi}\rangle$, where $\tilde{\pi}$ is the enriched occurence of $\mathcal{T}'$ in the original $\mathcal{S}$-term $\mathcal{T}$. Thus, the preprocessing algorithm starts with $\{\langle \mathcal{T}, \mathcal{S}, \langle\rangle\rangle\}$ and transforms this set until no further rule application is possible. We say that a set M of such triples is in a *solved form*, if for each $\langle \mathcal{T}', \mathcal{S}', \tilde{\pi}\rangle$ of M it holds:

- $\mathcal{S} \neq \langle\rangle$ and
- either $\mathcal{T}'$ is of the form $\{\langle\rangle_{f(t_1,\ldots,t_n)}\}$ and $\mathcal{S}$ is not of the form $f(\mathcal{S}_1,\ldots,\mathcal{S}_n)$,
- or $\mathcal{T}'$ is of the form $\{\langle\rangle_a\}$ and a is either a constant or a variable.

Otherwise, if no further rule application on M is possible and M is not in a solved form, then it is in an *unsolved form*. Three properties of the algorithm will be proved: First, it is shown that the algorithm terminates. Then, the soundness of the algorithm is proved, which garantees, that whenever for a given input $\{\langle\mathcal{T},\mathcal{S},\langle\rangle\rangle\}$ the algorithm terminates in a solved form M, then M is an abstraction of $\mathcal{T}$, i.e. $M \in \mathcal{ST}(\mathcal{T})$. Third, the completeness is proved, which ensures that whenever for the given input an unsolved form M is reached, then there is not any abstraction of $\mathcal{T}$ ($\mathcal{T}$ included) which $\mathcal{S}$-matches $\mathcal{S}$.

The transformation rules of the preprocessing algorithm are as follows:

$\mathcal{S}$-term-decomposition:

$$\frac{\{\langle f(\mathcal{T}_1,\ldots,\mathcal{T}_n), f(\mathcal{S}_1,\ldots,\mathcal{S}_n), \tilde{\pi}\rangle\} \cup M}{\{\langle \mathcal{T}_1, \mathcal{S}_1, \langle\tilde{\pi}, f^1\rangle\rangle, \ldots, \langle \mathcal{T}_n, \mathcal{S}_n, \langle\tilde{\pi}, f^n\rangle\rangle\} \cup M}$$

Term-decomposition:

$$\frac{\{\langle\{\langle\rangle_{f(t_1\ldots,t_n)}\}, f(\mathcal{S}_1,\ldots,\mathcal{S}_n), \tilde{\pi}\rangle\} \cup M}{\{\langle\{\langle\rangle_{t_1}\}, \mathcal{S}_1, \langle\tilde{\pi}, f^1\rangle\rangle, \langle\{\langle\rangle_{t_n}\}, \mathcal{S}_n, \langle\tilde{\pi}, f^n\rangle\rangle\} \cup M}$$

Abstraction:

$$\frac{\{\langle\mathcal{T}, \emptyset, \tilde{\pi}\rangle\} \cup M}{M}$$

Consider the problem of $\mathcal{S}$-matching $\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\}$ onto $\{\langle f^1, g^1\rangle_a, \langle f^2, f^1, f^2\rangle_b\}$, where x and y are variables, a and b constants, g a unary and f a duary function symbol. The preprocessing algorithm is given the following set

$$\{\langle\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\}, \{\langle f^1, g^1\rangle_a, \langle f^2, f^1, f^2\rangle_b\}, \langle\rangle\rangle\},$$

which is transformed as follows:

$$\{\langle\{\langle f^1\rangle_{g(x)}, \langle f^2, f^1\rangle_y\}, \{\langle f^1, g^1\rangle_a, \langle f^2, f^1, f^2\rangle_b\}, \langle\rangle\rangle\}$$

——— $\mathcal{S}$-term-Decomposition ———

$$\{\langle\{\langle\rangle_{g(x)}\}, \{\langle g^1\rangle_a\}, \langle f^1\rangle\rangle, \langle\{\langle f^1\rangle_y\}, \{\langle f^1, f^2\rangle_b\}\}, \langle f^2\rangle\rangle\}$$

——— $\mathcal{S}$-term-Decomposition ———

$$\{\langle\{\langle\rangle_{g(x)}\}, \{\langle g^1\rangle_a\}, \langle f^1\rangle\rangle, \langle\{\langle\rangle_y\}, \{\langle f^2\rangle_b\}\}, \langle f^2, f^1\rangle\rangle\}$$

——— Term-Decomposition ———

$$\{\langle\{\langle\rangle_x\}, \{\langle\rangle_a\}, \langle f^1, g^1\rangle\rangle, \langle\{\langle\rangle_y\}, \{\langle f^2\rangle_b\}\}, \langle f^2, f^1\rangle\rangle\}$$

——— Rebuild* ———

$$\{\langle f^1, g^1\rangle_x, \langle f^2, f^1\rangle_y\}$$

The $\mathcal{S}$-matching algorithm of the resulting $\mathcal{S}$-term, which is an abstraction of

$$\{\langle \mathsf{f}^1\rangle_{\mathsf{g}(\mathsf{x})}, \langle \mathsf{f}^2, \mathsf{f}^1\rangle_{\mathsf{y}}\},$$

yields the $\mathcal{S}$-matcher

$$([\mathsf{x} \leftarrow \mathsf{a}], [\mathsf{y} \leftarrow \{\langle \mathsf{f}^2\rangle_{\mathsf{b}}\}])$$

Now, as discussed above, we first prove the termination of the preprocessing algorithm.

Theorem 39 [Termination of the Preprocessing Algorithm]
Given a set M *of triples as defined above, then the preprocessing algorithm terminates either in a solved or in an unsolved form.*

Proof. The preprocessing algorithm may be viewed as being splitted into two parts: In the first part only the rules "$\mathcal{S}$-term-decomposition" and "term-decomposition" are applied, until none of these can be further applied. In the second part only the "Abstraction"-rule is applied. Then let size(M) *be the sum of the sizes of all left- and right-hand sides of the $\mathcal{S}$-equations in* M. *Obviously, the "$\mathcal{S}$-term-decomposition" and "term-decomposition" reduces this size and therefore the first part terminates. On the other hand let $\mathcal{S}$-*Eq(M) *be the amount of $\mathcal{S}$-equations in* M. *Again it is obvious to see, that the "Abstraction"-rule reduces the amount of $\mathcal{S}$-equations and therefore the second part of the preprocessing algorithm also terminates. Since both parts terminate, the entire preprocessing algorithm terminates.* □

The next property to be proven is the soundness of the preprocessing algorithm. It states that whenever for a given input $\{\langle \mathcal{T}, \mathcal{S}, \langle\rangle\rangle\}$ the preprocessing algorithm terminates in a solved form M, then $\mathsf{M} \in \mathcal{ST}(\mathcal{T})$.

Theorem 40 [Soundness of the Preprocessing Algorithm]
Given two $\mathcal{S}$-terms $\mathcal{T}$ and $\mathcal{S}$, if the preprocessing algorithm terminates on input $\{\langle \mathcal{T}, \mathcal{S}, \langle\rangle\rangle\}$ in a solved form M, *then $\mathsf{M} \in \mathcal{ST}(\mathcal{T})$ holds.*

Proof. We define a set M *of triples of being "adequate" with respect to a given $\mathcal{S}$-term $\mathcal{T}$, if and only if for each element of* M *hold:*

- *if the triple $\{\langle \mathcal{U}, \mathcal{R}, \tilde{\pi}\rangle\}$ does not fulfill the solved form properties, then $\tilde{\pi} \cdot \mathcal{U}$ must be an element of $\mathcal{ST}(\mathcal{T})$ or*
- *if the element is a triple $\langle\{\mathsf{t}_{\langle\rangle}, \mathcal{R}, \tilde{\pi}\rangle$ fulfilling the solved form properties, then $\{\tilde{\pi}_{\mathsf{t}}\} \in \mathcal{ST}(\mathcal{T})$ must hold.*

and the enriched occurences of all triples in M *must be compatible, in that none is a prefix of any other.*

For a given $\mathcal{S}$-matching problem of $\mathcal{T}$ onto $\mathcal{S}$ the input $\{\langle \mathcal{T}, \mathcal{S}, \langle\rangle\rangle\}$ is adequate with respect to $\mathcal{T}$ since $\mathcal{T} \in \mathcal{ST}(\mathcal{T})$. Now it is easy to see, that each transformation rule transforms a set M, *which is adequate with respect to $\mathcal{T}$ into a set* M′, *which also is adequate with respect to $\mathcal{T}$. Thus, if the preprocessing terminates in a solved form* M, *each element of* M *is an abstraction of $\mathcal{T}$. Furthermore, since* M *is adequate, the enriched occurences of the pairs in* M *are compatible and* M *is a $\mathcal{S}$-term and an abstraction of $\mathcal{T}$.* □

Finally, the completeness of the preprocessing algorithm is proved. The completeness property ensures that whenever the preprocessing algorithm terminates in an unsolved form, then there does not exist any abstraction $\mathcal{T}'$ of $\mathcal{T}$, such that $\mathcal{T}'$ $\mathcal{S}$-matches $\mathcal{S}$.

Theorem 41 [Completeness of the Preprocessing Algorithm]
Given two $\mathcal{S}$-terms $\mathcal{T}$ and $\mathcal{S}$, if the preprocessing algorithm terminates on input $\{\langle\mathcal{T},\mathcal{S},\langle\rangle\rangle\}$ in an unsolved form then it holds, that every abstraction $\mathcal{T}' \in \mathcal{ST}(\mathcal{T})$, $\mathcal{T}'$ does not $\mathcal{S}$-match $\mathcal{S}$.

Proof. If the preprocessing algorithm terminates in an unsolved form M *then there must be at least one triple in* M *being either of the form*

$$\langle \mathsf{f}(\mathcal{T}_1,\ldots,\mathcal{T}_n), \mathsf{g}(\mathcal{S}_1,\ldots,\mathcal{S}_n), \tilde{\pi}\rangle$$

or of the form

$$\langle \{\langle\rangle_{\mathsf{f}(t_1,\ldots,t_n)}\}, \mathsf{g}(\mathcal{S}_1,\ldots,\mathcal{S}_n), \tilde{\pi}\rangle,$$

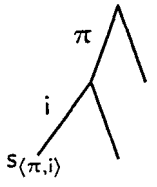
where $\mathsf{f} \neq \mathsf{g}$. *Thus, there can not be any abstraction $\mathcal{T}'$ of $\mathcal{T}$, such that $\mathcal{T}'_{|\tilde{\pi}}$ $\mathcal{S}$-matches* $\mathsf{g}(\mathcal{S}_1,\ldots,\mathcal{S}_n)$. □

7.2 Consequences

The preprocessing algorithm and its properties have nice consequences for the implementation of the $\mathcal{S}$-abstraction. Indeed, for a given equation $\mathsf{s} = \mathsf{t}$ not all abstractions have to be computed, but only the "least abstract" one. It holds, that any abstraction of some $\mathsf{s} = \mathsf{t}$ is more abstract than the "least abstract" abstraction of $\mathsf{s} = \mathsf{t}$. For a formal definition of this notion, a concept of *maximal common subterms* of two terms is required.

Definition 42 [Maximal Common Subterms]
Given two term s *and* t, *then their* maximal common subterms *are two sets of occurences* $\mathsf{Max}_\mathsf{s} \subseteq \mathsf{occ}(\mathsf{s})$ *and* $\mathsf{Max}_\mathsf{t} \subseteq \mathsf{occ}(\mathsf{t})$, *for which it holds:*

- *For each* $\langle\pi,\mathsf{i}\rangle \in \mathsf{Max}_\mathsf{s}$ s_π *does not occur in* t. *To illustrate this consider the following partial tree representation of* s*:*



  Thus, $\mathsf{s}_{\langle\pi,\mathsf{i}\rangle}$ *is a maximal common subterm, iff the term "preceding"* $\mathsf{s}_{\langle\pi,\mathsf{i}\rangle}$ *in the tree does not occur in* t, *but* $\mathsf{s}_{\langle\pi,\mathsf{i}\rangle}$ *does.*

- *For each* $\langle\pi,\mathsf{i}\rangle \in \mathsf{Max}_\mathsf{t}$ t_π *does not occur in* s *and*

- *For each subterm occurring in both* s *and* t *at the respective occurences* π_s *and* π_t *it holds, that either* π_s *or a prefix of it has to be in* Max_s *and that either* π_t *or a prefix of it has to be in* Max_t.

Now the *least abstract* $\mathcal{S}$-equation can be formally defined as follows:

Definition 43 [Least Abstract $\mathcal{S}$-equation]
Given an equation $s = t$, *then the* least abstract $\mathcal{S}$*-equation of this equation is its abstraction with respect to all maximal common subterms of* s *and* t. *We denote this* $\mathcal{S}$*-equation by* $\mathsf{least\text{-}abs}(s = t)$.

Furthermore, a binary relation "is less abstract as" can be defined over $\mathcal{S}$-equations. It will be shown, that this relation is a noetherian partial order over the $\mathcal{S}$-equations.

Definition 44 [Partial Order on $\mathcal{S}$-equations]
Given two $\mathcal{S}$*-equations* $\mathcal{S} = \mathcal{T}$ *and* $\mathcal{U} = \mathcal{V}$, *we define the partial order* $\prec$ *over* $\mathcal{S}$*-equations by*

$$\mathcal{S} = \mathcal{T} \prec \mathcal{U} = \mathcal{V} :\Leftrightarrow \mathcal{I}(\mathcal{S} = \mathcal{T}) \subset \mathcal{I}(\mathcal{U} = \mathcal{V})$$

Thus, $\mathcal{S} = \mathcal{T}$ *is less abstract than* $\mathcal{U} = \mathcal{V}$, *iff it represents less terms than* $\mathcal{U} = \mathcal{V}$. *The reflexive pendant* $\preceq$ *of* $\prec$ *is defined by*

$$\mathcal{S} = \mathcal{T} \prec \mathcal{U} = \mathcal{V} :\Leftrightarrow \mathcal{I}(\mathcal{S} = \mathcal{T}) \subseteq \mathcal{I}(\mathcal{U} = \mathcal{V})$$

Theorem 45 [$\prec$ is a noetherian partial order]
The binary relation $\prec$ *is a partial order over* $\mathcal{S}\mathsf{Eq}(\Sigma, \mathcal{V})$.

Proof. *The statement follows from the definition and the fact, that* $\subset$ *is a noetherian partial order on sets.* □

Now it will be shown, that for a given first order equation $s = t$, $\mathsf{least\text{-}abs}(s = t)$ is minimal in $\mathcal{S}\mathsf{Eq}(s = t)$ with respect to $\prec$.

Lemma 46.
Given a first order equation $s = t$, $\mathsf{least\text{-}abs}(s = t) \preceq \mathcal{S} = \mathcal{T}$ *holds for each* $\mathcal{S} = \mathcal{T} \in \mathcal{S}\mathsf{Eq}(s = t)$.

Proof. *Let* Π_s *and* Π_t *be the sets of maximal common subterms of* s *and* t *as defined in definition 42. Furthermore, let* $\Pi'_s := \{|\tilde{\pi}|| \ \tilde{\pi} \in \mathcal{OCC}(\mathcal{S})\}$ *and* $\Pi'_t := \{|\tilde{\pi}|| \ \tilde{\pi} \in \mathcal{OCC}(\mathcal{T})\}$[1]. *Since* Π_s *and* Π_t *are the sets of maximal common subterms, it holds that*

- *for all* $\pi' \in \Pi'_s$ *there exists a* $\pi \in \Pi_s$ *such that* π *is a prefix of* π' *and*
- *for all* $\pi' \in \Pi'_t$ *there exists a* $\pi \in \Pi_t$ *such that* π *is a prefix of* π'.

Thus, $\mathcal{S} = \mathcal{T} \in \mathcal{S}\mathsf{Eq}(\mathsf{least\text{-}abs}(s = t))$ *holds. This implies that* $\mathcal{S}\mathsf{Eq}(\mathcal{S} = \mathcal{T}) \subseteq \mathcal{S}\mathsf{Eq}(\mathsf{least\text{-}abs}(s = t))$. *By the equivalence relation between* $\mathcal{S}\mathsf{Eq}$ *and* $\mathcal{I}$ *it follows that* $\mathcal{I}(\mathsf{least\text{-}abs}(s = t)) \subseteq \mathcal{I}(\mathcal{S} = \mathcal{T})$ *and thus* $\mathsf{least\text{-}abs}(s = t) \preceq \mathcal{S} = \mathcal{T}$ *holds.* □

Since the preprocessing allows to reach any abstraction of a given $\mathcal{S}$-equation, it follows from this lemma that for a given set Φ of first order equations only the least abstractions need to be computed. Therefore, the cardinality of the set of needed abstractions of Φ is at most the cardinality of Φ. It is now shown that in general the cardinality is less; indeed it is shown that there exists for every set Φ a subset $\mathsf{Kernel}(\Phi)$, which contains the minimal set of equations for which the least abstractions have to be computed. Then it is shown, that from it any other $\mathcal{S}$-equation may be reached by the preprocessing algorithm. We first give the definition of the set $\mathsf{Kernel}(\Phi)$ and then show that this set fulfills the required properties.

[1] Note, that Π'_s and Π'_t only denote common subterms of s and t by definition of $\mathcal{S}$-equations.

Definition 47 [Kernel of a Set of Equations]
Given a set of first order equations Φ, then the kernel $\mathsf{Kernel}(\Phi)$ of Φ *is the smallest subset of* Φ *fulfilling the following properties:*

- *For all* $\mathsf{e}, \mathsf{e}' \in \mathsf{Kernel}(\Phi)$ *where* $\mathsf{e} \neq \mathsf{e}'$ *it holds*

$$\mathsf{least\text{-}abs}(\mathsf{e}) \not\prec \mathsf{least\text{-}abs}(\mathsf{e}') \text{ and}$$

- *For all* $\mathsf{e} \in \Phi$ *there exists an* $\mathsf{e}' \in \mathsf{Kernel}(\Phi)$, *such that*

$$\mathsf{least\text{-}abs}(\mathsf{e}') \preceq \mathsf{least\text{-}abs}(\mathsf{e}).$$

Note that if Φ is a finite set, then its kernel may be computed statically, since only the least abstractions of the equations of Φ have to be compared. This is especially an important property to provide an efficient implementation of the $\mathcal{S}$-abstraction.

We now prove, that the definition of the kernel of Φ is sound, i.e. that whenever Φ is a non empty set of first order equations then $\mathsf{Kernel}(\Phi)$ is not empty as well.

Theorem 48 [Soundness of Definition 47]
Given a non empty set Φ *of first order equations, then* $\mathsf{Kernel}(\Phi) \neq \emptyset$ *holds.*

Proof. The proof is done by induction over the cardinality n *of* Φ.

Base Case $-\ \mathsf{n} = 1$: *Thus* $\mathsf{Kernel}(\Phi) = \Phi$ *and* $\mathsf{Kernel}(\Phi)$ *fulfills both required properties of a kernel.*

Induction Step $-\ \mathsf{n} \rightarrow \mathsf{n}+1$: *Thus* $\Phi = \Phi' \cup \{\mathsf{e}\}$, $\mathsf{e} \notin \Phi'$ *and by induction hypothesis it follows that* $\mathsf{Kernel}(\Phi')$ *is defined and not empty. If for all* $\mathsf{e}' \in \mathsf{Kernel}(\Phi')$ $\mathsf{least\text{-}abs}(\mathsf{e}') \prec \mathsf{least\text{-}abs}(\mathsf{e})$, *then define* $\mathsf{Kernel}(\Phi) := \mathsf{Kernel}(\Phi')$ *and the statement holds trivially. Otherwise define*

$$\begin{aligned}\mathsf{Kernel}(\Phi) := \ & (\mathsf{Kernel}(\Phi') \cup \{\mathsf{e}\}) \\ & \setminus \{\mathsf{e}' \in \mathsf{Kernel}(\Phi') \mid \mathsf{least\text{-}abs}(\mathsf{e}) \prec \mathsf{least\text{-}abs}(\mathsf{e}')\}\end{aligned}$$

and the statement holds as well. □

The final theorem expresses that for each $\mathcal{S}$-equation belonging to an equation of Φ there is a least abstraction $\mathsf{least\text{-}abs}(\mathsf{e})$, $\mathsf{e} \in \mathsf{Kernel}(\Phi)$, such that $\mathsf{least\text{-}abs}(\mathsf{e})$ is less than the $\mathcal{S}$-equation with respect to $\prec$. This implies that the $\mathcal{S}$-equation may be reached from $\mathsf{least\text{-}abs}(\mathsf{e})$ by preprocessing, if required.

Theorem 49 [Minimality of the Least Abstractions of $\mathsf{Kernel}(\Phi)$**]**
Given a set Φ *of first order equations, it holds:*

$$\forall \mathsf{e} \in \Phi\ \forall \mathsf{E} \in \mathcal{S}\mathsf{Eq}(\mathsf{e})\ \exists \mathsf{e}' \in \mathsf{Kernel}(\Phi)\ .\ \mathsf{least\text{-}abs}(\mathsf{e}') \preceq \mathsf{E}$$

Proof. Given $\mathsf{e} \in \Phi$ *and* $\mathsf{E} \in \mathcal{S}\mathsf{Eq}(\mathsf{e})$, *it follows from lemma 46, that* $\mathsf{least\text{-}abs}(\mathsf{e}) \preceq \mathsf{E}$. *If* $\mathsf{e} \in \mathsf{Kernel}(\Phi)$, *then the statement of the theorem holds trivially. Otherwise, it follows from the definition of* $\mathsf{Kernel}(\Phi)$ *that there is an* $\mathsf{e}' \in \mathsf{Kernel}(\Phi)$, *such that* $\mathsf{least\text{-}abs}(\mathsf{e}') \preceq \mathsf{least\text{-}abs}(\mathsf{e})$. *By the transitivity of* $\prec$ *it follows that* $\mathsf{least\text{-}abs}(\mathsf{e}') \preceq \mathsf{E}$, *which proves the statement.* □

The significance of the existence of a kernel for each Φ is that the equations of the kernel share the structure information of all equations of Φ. For example does the equation $f(x,y) = g(x,y)$ somewhat "subsume" the structure information of $f(x,a) = g(x,b)$. Indeed, all structure manipulations encoded into the equations of Φ may already be performed if only the equations of $\mathsf{Kernel}(\Phi)$ would be under consideration. Thus, the notion of a kernel of a set Φ provides, from a structural point of view, a redundancy criteria for the equations of Φ.

7.3 Properties of the $\mathcal{S}$-Calculus

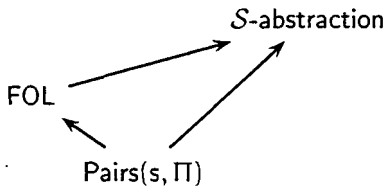
The $\mathcal{S}$-Calculus is not sound in the understanding that each proof performed by the $\mathcal{S}$-Calculus may be refined to a first order logic proof. But it is complete in the understanding that whenever there exists a rewrite proof for a first order logic equality problem, then there exists an abstract rewrite proof. This implies that whenever there exists a first order logic rewrite proof, if all abstract proofs for the abstract equality problem could be listed, then at least one of these can be refined to a first order logic rewrite proof. Obviously, this is beyond practicability. That for heuristics are used to guide the proof search and the quality measure of these heuristics is the percentage of refinable abstract proofs. However, the used heuristics for the proof search guidance are not complete. These results are already known from [Autexier, 1996] and thus their proofs are omitted in this report.

Chapter 8

Conclusion and Future Works

We presented the formal definition of an abstraction of first order terms, the so-called $\mathcal{S}$-abstraction. The idea of the abstraction is to focusize on the paths to some selected subterms of a given term. Furthermore, we proved several properties of this abstraction, some of which are very helpful for a space efficient implementation of the abstraction. Additionally, some obscure points, like the matching of two $\mathcal{S}$-terms, have been cleared out by a formal analysis of some properties. Therefore, the formal definition and analysis of the $\mathcal{S}$-abstraction provided a more efficient implementation of the abstraction. Especially, the properties of the preprocessing algorithm lead to a space efficient representation of the abstractions of a given equation database. This feature has been used for the implementation of the $\mathcal{S}$-abstraction into the INKA-system ([Biundo et al., 1986, Hutter and Sengler, 1996]).

The presented abstraction does not fit into the class of abstractions defined by Giunchiglia and Walsh [Giunchiglia and Walsh, 1992] since it is a *parameterized* abstraction and merely describes by itself an entire class of abstractions in the understanding of Giunchiglia and Walsh. In order to analyze the properties of the abstraction, a theoretical trick has to be used, such that the abstraction fits into the class of abstractions defined by Giunchiglia and Walsh. The idea is to define for a given first order signature a formal system for pairs (s, Π), where s is a term of $\mathcal{T}(\Sigma, \mathcal{V})$ and Π a set of occurences belonging to s. The intention of Π is to denote the parts of s which are preserved during the abstraction process. Then we could define an abstraction which maps pairs (s, Π) into the corresponding $\mathcal{S}$-term. This abstraction is an abstraction in the understanding of Giunchiglia and Walsh and we can analyze its properties. Observe that the first order logic is an abstraction of the formal system for pairs (s, Π). Thus, we get the following diagramm:



Thus, a future work is to establish the properties defined in [Giunchiglia and Walsh, 1992] of the $\mathcal{S}$-abstraction. Another future work on this abstraction is the integration of techniques for a better support of the heuristics based on the difference reduction paradigm. Since difference reduction techniques try to preserve common parts while trying to adapt non common parts, the

introduction of an annotation like colors in the $\mathcal{C}$-Logic of Hutter ([Hutter, 1991, Hutter, 1994]) or rippling ([Bundy et al., 1990]) would provide a strong support of the heuristics. Therefore, the definition of a *colored $\mathcal{S}$-abstraction* is one future work. Another interesting future work is the incorporation of structural abstractions of general formulae into the $\mathcal{S}$-abstraction. Obviously, this seems to be a rather simple task, although it has to be cleared out, whether formulas may be abstracted with respect to subterms, subformulas or both. The heuristics have also to be adapted to be able to deal with general abstract formulae. One possibility would be to use structural abstractions of implications together with heuristics similar to those used for $\mathcal{S}$-equations in [Autexier, 1996].

Bibliography

Autexier, S. (1996). Heuristiken zum Beweisen von Gleichungen. Diplomarbeit der Universität des Saarlandes FB14 (Siekmann), Universität des Saarlandes, Saarbrücken.

Biundo, S., Hummel, B., Hutter, D., and Walther, C. (1986). The Karlsruhe Induction Theorem Proving System. In Siekmann, J., editor, *Proceedings of the 8th International Conference on Automated Deduction (CADE)*, LNCS , pages 672–674. Springer.

Bundy, A. (1987). The use of explicit plans to guide inductive proofs. DAI Research Report 349, Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland.

Bundy, A., van Harmelen, F., Smaill, A., and Ireland, A. (1990). Extension to the rippling-out tactic for guiding inductive proofs. In Stickel, M. E., editor, *Proceedings 10th International Conference on Automated Deduction (CADE)*, volume 449 of *LNAI* , pages 132–146, Kaiserslautern, Germany. Springer.

Giunchiglia, F. and Walsh, T. (1992). A Theory of Abstraction. *Journal of Artificial Intelligence*, 56(2-3):323–390. Also as technical report IRST-Technical Report 9001-14.

Gordon, M. J., Milner, A. J., and Wadsworth, C. P. (1979). *Edinburgh LCF - A mechanised logic of computation.* Springer Verlag. LNCS 78.

Huet, G. and Oppen, D. C. (1980). Equations and Rewrite Rules: A Survey. *Formal Language Theory: Perspectives and Open Problems.*

Hutter, D. (1991). *Mustergesteuerte Strategien zum Beweisen von Gleichheiten.* PhD thesis, Universität Karlsruhe, Karlsruhe.

Hutter, D. (1994). Colouring terms to control equational reasoning. *Journal of Automated Reasoning.* to appear.

Hutter, D. and Sengler, C. (1996). INKA - The Next Generation. In McRobbie, M. A. and Slaney, J. K., editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE)*, volume 1104 of *LNCS* , New Brunswick, N. Y. Springer.

Loeckx, J., Ehrig, H.-D., and Wolf, M. (1996). *Specification of Abstract Data Types.* Teubner, Chichester;New York;Brisbane. ISBN 3-519-02115-3.

Plaisted, D. (1981). Theorem Proving with Abstractions. *Journal of Artificial Intelligence*, 16:47–108.

Index

Symbols

Terminology