Inductive Theorem Proving Using Refined Unfailing Completion Techniques

Bernhard Gramlich
SEKI Report SR-89-14

Inductive Theorem Proving Using Refined Unfailing Completion Techniques

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

This work was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D2).

Inductive Theorem Proving Using Refined Unfailing Completion Techniques

Bernhard Gramlich

Universität Kaiserlautern
Fachbereich Informatik
Postfach 3049
D-6750 Kaiserslautern
E-mail: gramlich@uklirb.uucp

Abstract

We present a brief overview on completion based inductive theorem proving techniques, point out the key concepts for the underlying "proof by consistency" - paradigm and isolate an abstract description of what is necessary for an algorithmic realization of such methods.

In particular, we give several versions of proof orderings, which - under certain conditions - are well-suited for that purpose. Together with corresponding notions of (positive and negative) covering sets we get abstract "positive" and "negative" characterizations of inductive validity. As a consequence we can generalize known criteria for inductive validity, even for the cases where some of the conjectures may not be orientable or where the base system is terminating but not necessarily ground confluent.

Furthermore we consider several refinements and optimizations of completion based inductive theorem proving techniques. In particular, sufficient criteria for being a covering set including restrictions of critical pairs (and the usage of non-equational inductive knowledge) are discussed.

Moreover a couple of lemma generation methods are briefly summarized and classified. A new techniques of save generalization is particularly interesting, since it provides means for syntactic generalizations, i.e. simplifications, of conjectures without loosing semantic equivalence.

Finally we present the main features and characteristics of UNICOM, an inductive theorem prover with refined unfailing completion techniques and built on top of TRSPEC, a term rewriting based system for investigating algebraic specifications.

1 Introduction

Equational reasoning is a fundamental basis for many fields of computer science like functional programming, abstract data type specifications, program synthesis and verification. It is well-known that the validity of an equation in all models of a set of equations can easily be characterized syntactically by the inference rules of instantiation and replacement of equals by equals.The corresponding characterization of inductive validity, i.e. validity in the standard initial model, involves inference rules with infinitely many premises. Thus inductive validity is much harder to obtain than general validity. A common approach for proving such inductive theorems

consists in using explicitely inductive proof techniques based on any well-founded ordering on (ground) terms (e.g. [Bu69], [BoMo79]). Starting with the work of Musser ([Mu80]), Goguen ([Go80]) and Huet/Hullot ([HuHo82]) an alternate approach of "proof by consistency" using completion techniques for term rewriting has been developed and further refined by many people. We provide a brief overview of these approaches and point out the key concepts using the theoretical framework of proof orderings and proof simplification (cf. [Ba87],[Ba88]). In particular, we show that inductive theorem proving may be considered from an abstract "positive" or "negative" point of view. This leads to a better understanding and generalizations of various sufficient operational criteria for inductive validity (cf. [JoKo86], [Kü87], [HoKu88]).

Furthermore we consider and discuss several refinements and optimizations of completion based inductive theorem proving techniques which are very important from a practical point of view. The crucial aspects and problems are illustrated by instructive examples.

Moreover various basic techniques for lemma generation by means of generalization including a classification scheme are presented and discussed. A new techniques of save generalization is particularly interesting, since it provides means for syntactic generalizations, i.e. simplifications, of conjectures without loosing semantic equivalence.

Finally we present the main features and characteristics of UNICOM, an inductive theorem prover with refined unfailing completion techniques and built on top of TRSPEC, a term rewriting based system for investigating algebraic specifications (cf. [AvGrGöMa87], [Sc88]).

2 Definitions and Notations

We assume familiarity with the basic notions of term algebras, equational logic and rewriting systems (cf. [HuOp80]). For the sake of readability we restrict ourselves to the one-sorted case. The results easily carry over to the many-sorted case. In the following we recall the essential terminology used subsequently. We are dealing with first order terms over some set of operator symbols F and some set of variables V. We assume that F contains at least one constant. Thus the set of ground terms T is non-empty. By t/p we denote the subterm of t at position p and by σt the result of applying a substitution σ to t. We write u[s] to indicate that the term u contains s as a subterm and (ambiguously) denote by u[t] the result of replacing a particular occurrence of s in u by t. An equation is a pair of terms, written $s = t$. A rewrite rule is a directed equation, written $s \to t$. A term rewriting system (TRS) R is a set of rewrite rules. For a set E of equations we denote by $E^{\leftrightarrow}$ the symmetric closure of E. For a binary relation $\to$ on terms the symbols $\overset{0}{\to}$, , $\overset{+}{\to}$, $\overset{*}{\to}$, $\leftrightarrow$ denote the reflexive, transitive, reflexive-transitive and symmetric closure of $\to$, respectively. The relation $\leftarrow$ is the inverse of $\to$. By $\to_R$ we denote the reduction relation generated by R and by $\leftrightarrow_E$ the (one-step) equality relation induced by E. The equational theory of E is defined to be $Th(E) := \{s = t \mid s \overset{*}{\leftrightarrow}_E t\}$ and the inductive theory is given by $ITh(E) := \{s = t \mid \sigma s \overset{*}{\leftrightarrow}_E \sigma t$ for all ground substitutions $\sigma\}$. Equations from Th(E) and ITh(E) are called equational and inductive theorems of E, respectively.

By CP(R,R') we denote the set of critical pairs obtained from overlapping the rules of R into those of R'. If p is a non-variable position of a term l then $CP(R,p,l \to r)$ denotes the set of critical

pairs obtained by overlapping R into $l \to r$ at position p. Accordingly $CP(R,P,l \to r)$ denotes the set of critical pairs obtained by overlapping R into $l \to r$ at non-variable positions from P.

A TRS R is terminating if $\xrightarrow{+}_R$ is a well-founded (strict partial) ordering. R is confluent if ${}_R\xleftarrow{*} \circ \xrightarrow{*}_R \subseteq \xrightarrow{*}_R \circ {}_R\xleftarrow{*}$ and Church-Rosser if $\xleftrightarrow{*}_R \subseteq \xrightarrow{*}_R \circ {}_R\xleftarrow{*}$. It is ground confluent (ground Church-Rosser) if it is confluent (Church-Rosser) on ground terms. Note that the (ground) Church-Rosser property is equivalent to (ground) confluence. R is (ground) convergent if it is terminating and (ground) confluent. A reduction ordering > is a well-founded ordering on terms which is monotonic w.r.t. to replacement ($s > t \Rightarrow u[s] > u[t]$) and substitution ($s > t \Rightarrow \sigma s > \sigma t$). A simplification ordering is a reduction ordering that satisfies the subterm property ($s[t] \geq t$).

By a proof in E we mean a sequence of equational replacements $t_0 \longleftrightarrow_E t_1 \longleftrightarrow_E \ldots \longleftrightarrow_E t_n$. A proof in $E \cup R$ consists of proof steps of the form $t_i \longleftrightarrow_E t_{i+1}$, $t \longrightarrow_R t_{i+1}$ or $t\ {}_R\longleftarrow t_{i+1}$. If E and R are unimportant or clear from the context we also denote a proof of the form $t_0 \xleftrightarrow{*}_{E\cup R} t_n$ by the sequence $(t_0,\ldots,t_n)$ of its intermediate results. Two proofs of the form $s \xleftrightarrow{*}_{E\cup R} t$ and $u \xleftrightarrow{*}_{E'\cup R'} v$ are said to be equivalent if $s \equiv u$ and $t \equiv v$ ($\equiv$ means syntactical equality). Proofs of the form $\circ \xrightarrow{*}_R \circ {}_R\xleftarrow{*} \circ$ are called rewrite proofs. For a proof P we denote by σP the proof obtained from P by instantiating all intermediate results with σ. The notion P[P'] ambiguously indicates that P contains P' as a subproof. For a proof $P = (t_0,\ldots,t_n)$ and a term c we denote by c[P] the proof $(c[t_0],\ldots,c[t_n])$. A proof ordering is a (strict partial) ordering >> on proofs. It is said to be a proof reduction ordering if it is well-founded and monotonic w.r.t. replacement ($P >> P' \Rightarrow c[P] >> c[P']$), substitution ($P >> P' \Rightarrow \sigma P >> \sigma P'$) and embedding ($P >> P' \Rightarrow Q[P] >> Q[P']$). A proof P (in $E \cup R$) of the form $s \xleftrightarrow{*}_{E\cup R} t$ is said to be (equivalently) simplifiable into a proof P' (in $E' \cup R'$) if P' is of the form $s \xleftrightarrow{*}_{E'\cup R'} t$ (see [Ba87] for a more detailed introduction of equational proofs).

3 Foundations of Completion Based Inductive Theorem Proving

Assume that we are given a set E of equations represented by a ground equivalent term TRS R, i.e. $\xleftrightarrow{*}_E = \xleftrightarrow{*}_R$ on T. The inductive validity of an equational conjecture $s = t$ may then be equivalently expressed as follows:

$s = t \in ITh(E)$ (1)

iff $\sigma s \xleftrightarrow{*}_E \sigma t$ for all ground substitutions σ

iff $\sigma s \xleftrightarrow{*}_R \sigma t$ for all ground substitutions σ

iff $\xleftrightarrow{*}_R = \xleftrightarrow{*}_{R\cup\{s\to t\}}$ on T

iff $T/\xleftrightarrow{*}_R \cong T/\xleftrightarrow{*}_{R\cup\{s\to t\}}$ (2)

This characterization reveals that proving inductive validity of $s = t$ (1) amounts to showing consistency (2), i.e. the initial algebra defined by E is not destroyed by adding $s = t$.

Now, if we do not assume any special properties of R the only possibility of establishing (1) seems to consist in trying an explicit inductive proof using some correct (and appropriate) induction scheme for the variables occurring in $s = t$. Of course, from the viewpoint of automated reasoning

this is not very satisfactory since all the problems concerning automated equational reasoning are still present. If however we know that R satisfies certain properties, we can use the proof by consistency approach in a more or less automated way. Perhaps the most interesting and often satisfied case for that purpose is given, when E can be represented by a ground convergent, rewrite system R. In this case the search for an inconsistency can be automated as follows. A key observation is that for ground convergent R the algebra of unique ground normal forms of R is (isomorphic to) the initial algebra $T/\overset{*}{\leftrightarrow}_R$. This leads to the following characterization for being an inductive theorem.

Lemma 3.1 ([De82])
Let R be a ground convergent rewrite system and C be a set of equational conjectures. Then all equations of C are inductive consequences of R iff for every ground instance $\sigma s = \sigma t$ of an equation $s = t \in C$, σs and σt have identical normal forms w.r.t. R. ■

This result implies a necessary criterion for being inductively valid.

Lemma 3.2
Let $>$ be a reduction ordering, R a $>$-ordered ground convergent rewrite system and $s = t$ be an inductive consequence of R . Then the following properties hold:

(a) $s > t \Rightarrow$ every ground instance σs of s is R-reducible, and

(b) $s <> t \Rightarrow$ for every ground instance $\sigma s = \sigma t$ with $\sigma s \not\equiv \sigma t$ we have
$\sigma s > \sigma t \Rightarrow \sigma s$ is R-reducible
$\sigma s <> \sigma t \Rightarrow \sigma s$ or σt is R-reducible.

Proof:

(a) Assume some ground instance σs of s is R-irreducible and let u be the R-normal form of σt. From $s > t$ and $\sigma t \overset{*}{\rightarrow}_R u$,we get $\sigma s > \sigma t \geq u$ and thus $\sigma s \not\equiv u$ contradicting lemma 3.1.

(b) Straightforward using the same argument as in (a) for the case $\sigma s > \sigma t$, and using lemma 3.1 else. ■

This motivates the notion of inductive reducibility (also called quasi- or ground reducibility) which was first introduced in [JoKo86] for terms and generalized to equations, i.e. pairs of terms, in [Ba88].

Definition 3.1
A term s is **inductively (R-) reducible** iff all its ground instances are (R-) reducible. An equation $s = t$ is **inductively (R-) reducible** iff σs or σt is (R-) reducible for every ground instance $\sigma s = \sigma t$ of $s = t$ with $\sigma s \not\equiv \sigma t$.

With this notion lemma 3.2 may be reformulated in a slightly weaker version as

Corollary 3.3

Let $>$ be a reduction ordering, R a $>$-ordered ground convergent rewrite system and $s = t$ be an inductive consequence of R. Then we have

(a) $s > t \Rightarrow$ s is inductively reducible, and

(b) $s <> t \Rightarrow s = t$ is inductively reducible. ■

The necessary conditions (a) and (b) for being inductively valid can effectively be tested due to decidability of inductive reducibility (assuming decidability of $>$). This was first shown in [Pl85]. Various methods and algorithms for solving the problem of inductive reducibility (which has an exponential complexity even for left-linear R ([KaNaRoZh87])) more efficiently for practical cases, have been proposed (cf. [JoKo86], [KaNaZh86], [Ku88], [BüKü89]). For theories with constructors the set of ground substitutions to be tested is reduced and in the important subcase of free constructors the problem becomes trivial (cf. [HuHu80], [JoKo86]). The results on decidability, complexity and algorithmic solutions for inductive reducibility are easily generalized from terms to equations, i.e. pairs of terms. This may be done using the well-known technique of encoding equations $s = t$ into terms eq(s,t) where eq is a new binary function symbol, and adding to R a rule with left-hand side eq(x,x).

The following result now uses the notion of inductive reducibility and provides a characterization of inductive validity of equations that can be oriented w.r.t. the underlying reduction ordering for R. It is a generalized version of theorem 1 in [JoKo86].

Theorem 3.4

Let $>$ be a reduction ordering, R a $>$-ordered ground convergent rewrite system and $l = r$ an equation with $l > r$. Then $l = r$ is an inductive theorem of R iff l is inductively R-reducible and $R \cup \{l \to r\}$ is ground convergent.

Proof: If $l = r \in$ ITH(R) then Corollary 3.3 implies that l is inductively reducible. Since in an arbitrary ground proof every $l \to r$ - step can be replaced by a sequence of R-steps we can use the assumption that R is ground convergent and conclude that $R \cup \{l \to r\}$ also must be ground convergent. Conversely, if $\sigma l = \sigma r$ is any ground instance of $l = r$, then the normal forms of σl and σr w.r.t. R are also normal forms w.r.t. $R \cup \{l \to r\}$ due to inductive reducibility of l. Moreover these normal forms must coincide because $R \cup \{l \to r\}$ is ground convergent. ■

In fact, theorem 3.4 can be used now to design a completion based inductive proof technique, which roughly spoken, proceeds as follows: The inductive conjecture $s = t$ is added as a rewrite rule $s \to t$ with $s > t$ to the ground convergent $>$-ordered base system R. If ground completion, i.e. completion for verifying ground confluence of $R \cup \{s \to t\}$ yields a ground convergent rewrite system R' such that all left-hand sides of R' are inductively R-reducible, then $s = t$ as well as all other generated rules (equations) are inductive consequences of R. Note that for terminating rewrite systems there exists no finite test for ground confluence - in contrast to the critical pair test for general confluence. Convergence of critical pairs is only a sufficient criterion for ground

confluence, but not a necessary one in general.

The main drawback of the method up to now is its inability of handling non-orientable equations. But using the concepts of proof orderings and unfailing completion (cf. [HsRu86], [BaDeHs86], [Ba87], [BaDePl87]) this problem can also be solved - at least partially.

Formally this approach has been developped in [Ba88] resulting in a refutationally complete method for (equational) proofs by consistency. In [Sc88] the idea of unfailing ground completion has also been used in combination with ground confluence criteria and various other improvements (cf. [Kü87]) to provide a refined completion based proof technique. Later we will describe the resulting system UNICOM (cf. [Sc88]) which has been implemented on top of TRSPEC (cf. [AvGöGrMaSt87]), a rewriting based system for the specification, verification and rapid prototyping of algebraic specifications.

Before going into details let us roughly explain why the framework of (equational) proof transformation, simplification and proof orderings is much better suited to understand and to solve many problems related to completion based reasoning than any specific completion algorithm. First of all, using the notion of equational proofs, we can clearly state the problem in mind and identify problematic situations. For example, if we want to construct a convergent rewrite system R equivalent to a given set $E_o \cup R_o$ of equations and rewrite rules, the problem consists in finding for all equational proofs of the form $s \overset{*}{\longleftrightarrow}_{Eo \cup Ro} t$ equivalent rewrite proofs, i.e. proofs of the form $s \overset{*}{\longrightarrow}_R \cdot {}_R\overset{*}{\longleftarrow} t$. Here, problematic situations are given by proofs containing subproofs of the form $\circ \leftrightarrow_{Eo} \circ$ (equality pattern) or $\circ {}_{Ro}\!\leftarrow \circ \rightarrow_{Ro} \circ$ (peak). Such problematic situations may be eliminated by transforming (E_o,R_o) into an equivalent pair (E_1,R_1) using an appropriate equivalence preserving inference rule for pairs (E_i,R_i) (here: orienting an equation and adding an equational consequence, respectively). If one can additionally specify a well-founded proof ordering such that eliminating problematic proof patterns as above results in proof simplification w.r.t. this ordering the original problem has been transformed into a proof simplification problem. Thus, appropriate equivalence preserving systems of inference rules provide the logical basis for completion based reasoning, whereas abstract conditions concerning the control of inference rule application (fairness conditions) may be designed in order to guarantee terminating proof simplication of any problematic proof pattern. Within such a general framework specific correct completion procedures are easily derived by fixing a specific fair control structure. The clear distinction between logic and control for solving problems related to completion techniques greatly simplifies correctness proofs and leads to a better understanding of what is going on. This situation is similar to approaches for unification and disunification (or more general: equational) problems (cf. [Co88]) where the separation of solution techniques into a logical and a control component has turned out to be very successful.

Let us now come back to inductive proofs. We assume in the following that $>$ is a reduction ordering and R a $>$-ordered ground convergent rewrite system.

An important abstract observation is the following: The problem of proving the inductive validity or invalidity of a set of equational conjectures C may be considered either from a positive or from a negative point of view. In the negative case one tries to find an inconsistency whereas in the

positive case the aim consists in showing that every ground proof using $R \cup C$ may be replaced by an equivalent ground proof using only R. We shall present both the negative and the positive approach within a unified framework. Whereas the negative approach essentially is a reformulation of the main ideas of [Ba88] the positive approach is new and will be useful for generalizing known results.

The negative approach

As already mentioned, the notion of inductive reducibility may be used to detect direct or provable inconsistency of an equational conjecture $s = t$. This leads to

Definition 3.2 (cf. [Ba88])
Let C be a set of equational conjectures. An **inconsistency witness** for C is any ground proof of the form $s' \,{}_R\!\xleftarrow{*} \sigma s \leftrightarrow_C \sigma t \xrightarrow{*}_R t'$, such that s' and t' are distinct (R-) irreducible ground terms and the C-step is an application of $s = t \in C$ at top position.An inconsistency witness for C is said to be **indirect** if it is of the above form and additionally $s' \,{}_R\!\xleftarrow{+} \sigma s$ and $\neg(t > s)$.It is **direct**, if it is not indirect, i.e. if it is of the above form with $s' \equiv \sigma s$ and additionally $s > t$ or else $s \mathrel{<>} t$ and $t' \equiv \sigma t$. C is said to be **inconsistent** if there is an inconsistency witness for C. C is said to be **provably inconsistent** if there is a direct inconsistency witness, i.e. if it contains an equation that is not inductively reducible or an equation $s = t$ such that $s > t$ and s is not inductively reducible. C is **consistent** if it is not inconsistent.

Obviously, consistency of C is equivalent to inductive validity of C and provable inconsistency implies inconsistency. It is also clear that provable inconsistency is decidable because inductive reducibility is decidable.

Now, if C is inconsistent but not provably inconsistent, then there is an indirect inconsistency witness. By deducing appropriate equations from R and C any such proof may be simplified w.r.t. an appropiate proof ordering until eventually a smaller direct inconsistency witness is obtained. For that purpose the critical pair mechanism is crucial. This is reflected in Bachmair's inference system $\mathcal{P}$ for proofs by consistency by the deduction rule. Here, C is any set of equational conjectures and L any set of lemmas, i.e. inductive consequences of R.

Deduction $\dfrac{L,\ C}{L,\ C \cup \{s = t\}}$, if $s \leftrightarrow_{CP(R,C^{\leftrightarrow})} t$

Induction $\dfrac{L,\ C}{L \cup \{s = t\},\ C}$, if $s = t \in \mathrm{ITh}(R)$

Deletion $\dfrac{L,\ C \cup \{s = t\}}{L,C}$, if $s = t \in \mathrm{ITh}(R)$

Simplification $\dfrac{L, C \cup \{s = t\}}{L, C \cup \{u = t\}}$, if $s > u$ and either $s \overset{+}{\longleftrightarrow}_{R \cup L} u$ or $s \leftrightarrow_C u$ by an equation $v = w$ such that $s \triangleright v$ and $v > w$.

Here $\triangleright$ denotes the proper specialization ordering: $s \triangleright v$ iff some subterm of s is an instance of v but not vice versa. The application of one of the above inference rules to L,C yielding L',C' is denoted by $L,C \vdash_{\mathcal{P}} L',C'$. A (possibly infinite) sequence of such steps $L_o,C_o \vdash_{\mathcal{P}} L_1,C_1 \vdash_{\mathcal{P}} \ldots$ is **a derivation** from L_o and C_o.

The proof ordering mentioned above is tailored to inconsistency witnesses and designed such that applying the simplification rule does not increase the complexity of proofs. Assume that $>'$ is any simplification ordering containing R and the given reduction ordering $>$. One may take for instance for $>'$ the transitive closure of the union of $>$ and the proper subterm ordering. Then the complexity cneg(σs,σt) of a single proof step $\sigma s \leftrightarrow_C \sigma t$ (using $s = t \in C$ at top position) is given by

$$\text{cneg}(\sigma s,\sigma t) := \begin{cases} (\{\sigma s\}, s, \sigma t), & \text{if } s > t \\ (\{\sigma t\}, t, \sigma s), & \text{if } t > s \\ (\{\sigma s,\sigma t\}, -, -), & \text{if } s \not> t \text{ and } t \not> s \end{cases}$$

If P is an inconsistency witness for C of the form $s' \ {}_R\!\overset{*}{\longleftarrow} \sigma s \leftrightarrow_C \sigma t \overset{*}{\longrightarrow}_R t'$, then cneg(P) := cneg($\sigma$s,$\sigma$t). For all other proofs P let cneg(P) := max. The symbol max is taken to be maximal in the ordering $>^{cneg}$, which is defined to be the lexicographic combination of the multiset ordering $>>'$ induced by $>'$, the proper specialization ordering $\triangleright$ and the simplification ordering $>'$. The ordering $>^{cneg}$ is well-founded because the component orderings are well-founded. Clearly $>^{neg}$ defined by $P >^{neg} P'$ iff $\text{cneg}(P) >^{cneg} \text{cneg}(P')$ is also well-founded. With these definitions it can be shown by case analysis that every $\vdash_{\mathcal{P}}$-step does not increase the complexity of proofs.

Lemma 3.5 ([Ba88])
Whenever $L,C \vdash_{\mathcal{P}} L',C'$ and P is a proof in $R \cup C$ then there is a proof P' in $R \cup C'$ with $P \geq^{neg} P'$.

■

To assure simplification of inconsistency witnesses of a set C which is inconsistent, but not provably inconsistent, the notion of a covering set is introduced.

Definition 3.3 (cf. [Ba88])
C' is said to be a **(negative) covering set** for C (w.r.t. R and $>^{neg}$) iff we have $C \subseteq \text{ITh}(R) \Leftrightarrow C \cup C' \subseteq \text{ITh}(R)$ and for every indirect inconsistency witness of C there is another inconsistency witness of $C \cup C'$ which is smaller (w.r.t $>^{neg}$).

This definition of a covering set slighty differs from the one give in [Ba88] in that we require C' to satisfy additionally $C \subseteq \text{ITh}(R) \Leftrightarrow C \cup C' \subseteq \text{ITh}(R)$. This restriction is motivated by the fact that

we don't want to introduce arbitrary new conjectures C' that have nothing to do with the original C.

As it can be easily verified the set $CP(R,C^{\leftrightarrow})$ of critical pairs obtained by overlapping R into $C^{\leftrightarrow}$ is a (negative) covering set for C.

To state Bachmair's main result the following definitions of [Ba88] are needed. A **proof by consistency procedure** is any program accepting as input a reduction ordering $>$, a ground convergent $>$-ordered rewrite system R, a set L of inductive theorems and a set C of conjectures, and generates a derivation from L and C. A derivation $L_o,C_o \vdash_{\mathcal{P}} L_1,C_1 \vdash_{\mathcal{P}} \ldots$ is **fair** iff the set $\bigcup_i C_i$ of all deduced equations is a (negative) covering set for the set $\bigcup_i \bigcap_{j\geq i} C_j$ of all persisting equations. A proof by consistency procedure is **fair** iff it produces only fair derivations.

Theorem 3.6 ([Ba88])
Every fair proof by consistency procedure is refutationally complete, i.e. from every inconsistent set of conjectures C_o it generates a derivation such that some set C_i is provably inconsistent.

Proof idea: Using the fairness assumption and the definition of a covering set one can show: Starting from an inconsistent but not provably inconsistent set C_o and any indirect inconsistency witness $P_{io} := P_o$ for $C_{io} := C_o$ it is possible to construct a sequence of inconsistency witnesses P_{ij} for $R \cup C_{ij}$ such that $P_{io} >^{neg} P_{i1} >^{neg} P_{i2} >^{neg} \ldots$. Since $>^{neg}$ is well-founded this sequence must terminate so that for some $j > 0$, P_{ij} is a direct inconsistency witness of C_{ij}, i.e. C_{ij} is provably inconsistent. ■

Now, in practice fairness may be guaranteed by some marking scheme for the computation of covering sets analogous to standard completion. Inductive validity is characterized by

Lemma 3.7
A set C of equations is inductively valid (w.r.t. R) iff C is a (negative) covering set for itself and is not provably inconsistent.

Proof: The only-if-direction is trivial using the definition of (negative) covering set and provable inconsistency. Conversely assume that C is a covering set for itself and is not provably inconsistent and that there exists an equation $s = t \in C$ with $s = t \notin ITh(R)$. Then one can construct an infinite sequence of indirect inconsistency witnesses which is strictly decreasing w.r.t. $>^{neg}$. This yields a contradiction to the well-foundedness of $>^{neg}$. ■

Hence, for successful inductive proofs we have to compute (and simplify) covering sets using the inference system $\mathcal{P}$ until eventually a verifiably self-covering set of conjectures is obtained such that no provable inconsistency has been encountered.

The positive approach

As already mentioned proving inductive validity of a set of equational conjectures C from a positive point of view can be considered as the following problem: Show that any ground proof using $R \cup C$ can be transformed into an equivalent ground proof using only R, i.e. any C-step can be eliminated. In order to turn this task into a proof simplification problem, too, we use Bachmair's inference system $\mathcal{P}$ and define an appropriate ordering on proofs (similar to [BaDePl87]) as follows:

If (u,v) is a proof step of the form $u \leftrightarrow_C v$ applying an equation $s = t \in C$ at position p then its complexity is defined by

$$cpos(u,v) := \begin{cases} (\{u\}, u/p, s, v) , & \text{if } s > t \\ (\{v\}, v/p, t, u) , & \text{if } t > s \\ (\{u,v\}, -, -, -) , & \text{if } s <> t \end{cases}$$

The complexity of a proof P in $R \cup C$ is defined to be the multiset of the complexities of all its C-steps. Complexities of proofs are ordered by the multiset extension $>>^{cpos}$ of $>^{cpos}$. The ordering $>^{cpos}$ is defined to be the lexicographic combination of the multiset extension $>>$ of the reduction ordering $>$, the proper subterm ordering, the proper subsumption ordering and the reduction ordering $>$. Clearly $>^{cpos}$ as well as $>>^{cpos}$ is well-founded. Thus the induced ordering $>^{pos}$ on proofs is also well-founded. Moreover, it is easily verified analogous to [BaDePl87] that $>^{pos}$ is monotonic w.r.t. embedding, replacement and substitution. Thus it is a proof reduction ordering. Note that there is a unique minimal complexity among proofs, namely the empty multiset (of quadruplets) which corresponds to proofs using only R-steps.

Analogous to lemma 3.5 one can prove now by case analysis, that the application of an inference rule of $\mathcal{P}$ does not increase the complexity of ground proofs.

Lemma 3.8
Whenever $L,C \vdash_{\mathcal{P}} L',C'$ and P is a ground proof in $R \cup C$ then there is an equivalent ground proof P' in $R \cup C'$ with $P \geq^{pos} P'$. ■

Now, in order to enable proof simplification we may again define and use a corresponding notion of a (positive) covering set as follows:

Definition 3.4 (cf. definition 3.3)
C' is said to be a **(positive) covering set** for C (w.r.t. R and $>^{pos}$) iff we have $C \subseteq ITh(R) \Leftrightarrow C \cup C' \subseteq ITh(R)$ and for every ground proof P in $R \cup C$ with at least one C-step there is an equivalent ground proof P' with $P >^{pos} P'$.

Again we obtain a characterization for inductive validity as follows:

Lemma 3.9
A set C of equations is inductively valid (w.r.t. R) iff C is a (positive) covering set for itself.

Proof: The only-if-direction is trivial using the definition of a (positive) covering set. Conversely, assume that C is a (positive) covering set for itself and that there is an equation s = t in C which is not inductively valid. Then there exists an inconsistency witness $s' \ {}_R\!\xleftarrow{*} \sigma s \leftrightarrow_C \sigma t \xrightarrow{*}_R t'$. Since in any ground proof of the form $\sigma s \xleftrightarrow{*}_{R \cup C} \sigma t$ there must be a C-step which is not valid (w.r.t. R) we can construct an infinite sequence of equivalent ground proofs starting from $\sigma s \leftrightarrow_C \sigma t$ with strictly decreasing complexity (note that the monotonicity properties of $>^{pos}$ w.r.t. embedding and replacement are needed here !). Hence we have a contradiction to $>^{pos}$ being well-founded. ■

Now, if C is not provably inconsistent, but we don't know already whether C is inductively valid, i.e. a covering set for itself, then we have to compute a covering set for it. Again this is done by constructing critical pairs between R and $C^{\leftrightarrow}$ as we will see below. But if we simply add such a critical pair to C, and try to simplify it later according to the inference system $\mathcal{P}$, an important information concerning simplifiability has been lost. Namely, the superposition term belonging to the corresponding critical overlap has been ignored. By taking into account this additional information we get the following abstract characterization of inductive validity be means of simplifiability of ground instances of critical pair overlaps.

Lemma 3.10
A set C of equational conjectures is inductively valid (w.r.t. R) iff
(1) C is not provably inconsistent, and
(2) all ground instances of critical pair overlaps corresponding to $CP(R,C^{\leftrightarrow})$ are equivalently simplifiable with R union C (w.r.t. $>^{pos}$).

Proof: The only-if-direction is obvious. For the converse let $u \leftrightarrow_C v$ be any ground proof using one C-step, say s = t at position p, i.e. $u' := u/p = \sigma s$, $v' := v/p = \sigma t$ for some u', v', σ such that $\neg(t > s)$ w.l.o.g. Since C is not provably inconsistent, we may assume that σs is R-reducible say to s' with one application of a rule $l \to r$ in R. Then the usual case analysis for rewriting ambiguities reveals that we either have a variable or a critical overlap. In the first case the ground proof $\sigma s \leftrightarrow_C \sigma t$ may be equivalently replaced by a smaller proof of the form $\sigma s \xrightarrow{+}_{l \to r} \circ \leftrightarrow_C \circ \ {}_{l \to r}\!\xleftarrow{*} \sigma t$. In the latter case the critical overlap is a ground instance of a critical pair overlap which by assumption is simplifiable. Using the monotonicity property of $>^{pos}$ w.r.t. replacement we may infer that the original ground proof $u \leftrightarrow_C v$ can be replaced by an equivalent but simpler ground proof using R union C. Well-foundedness of $>^{pos}$ implies that by recursively constructing such simpler ground proofs we eventually obtain a proof of the form $u \xleftrightarrow{*}_R v$. ■

Note that this result essentially is a generalized version of theorem 14 in [Kü87], in the sense that it does not require all conjectures to be >-orientable. Indeed, for the above characterization of

inductive validity, a slightly weaker form of the ordering $>^{pos}$ suffices, too. Namely, we only have to take into account the first component of the complexity of C-steps. This yields a "standard" ordering $>^{pos1}$ where the corresponding ordering $>^{cpos1}$ for proof complexities is defined by double multiset extension of the underlying reduction ordering $>$, i.e. $>^{cpos1} := >> >>$. Thus we get a straightforward sufficient operational criterion for inductive validity as follows:

Corollary 3.11

Let $>$ be a reduction ordering, R a $>$-ordered ground convergent rewrite system, C a set of conjectures which is not provably inconsistent, and L a set of inductive lemmas, i.e. $L \subseteq ITh(R)$. Assume further that

(*) for all critical pairs $(s,t) \in CP(R,C^{\leftrightarrow})$ with corresponding superposition term o we can find a proof for $s = t$ of the form $s \xleftrightarrow{*}_{R \cup C \cup L} t$ such that all intermediate results are either smaller than o or smaller than t (w.r.t. $>$).

Then we can conclude: $C \subseteq ITh(R)$.

Proof: Straightforward using the ordering $>^{cpos1}$ and the fact that in every ground proof each L-step may be equivalently replaced by R-steps which do not contribute to ground proof complexities. ■

Note that condition (*) above is not yet effectively testable in general. But it may be replaced for instance by the following stronger condition:

(**) for all critical pairs $(s,t) \in CP(R,CT) \cup CP(R,CU^{\leftrightarrow})$ with corresponding superposition term o there exists a proof for $s = t$ of the form $s \xrightarrow{*}_{R \cup CT \cup LT} \circ \xleftrightarrow{}_{LU \cup CU} \circ \xleftarrow{*}_{R \cup CT \cup LT} t$, where $C = CT \cup CU$ and $L = LT \cup LU$ with CT,LT $>$-ordered and CU,LU $>$-uncomparable.

Here, all intermediate results are automatically either smaller than o or smaller than t. And moreover the existence of such a proof is clearly decidable.

Note that taking $CP(R,CT) \cup CP(R,CU^{\leftrightarrow})$ as a covering set instead of $CP(R,CU^{\leftrightarrow})$ is possible in the previous results, too, whenever we have a partition of C into $>$-ordered conjectures CT and $>$-uncomparable conjectures $CU^{\leftrightarrow}$. This is easily verified and conforms to the usual practice in completion procedures.

Using the proof ordering $>^{pos}$, the notion of a (positive) covering set and corresponding definitions of a fair derivation and a fair proof by consistency procedure one may easily state and prove a refutational completeness result analogous to theorem 3.6 within this "positive" framework.

Summarizing one can say that the presented negative and positive approaches to proofs by consistency essentially yield the same results for ground convergent base systems. The technical differences concerning the underlying notions of orderings on proofs and covering sets reflect the duality of the corresponding viewpoint of the problem.

But within our positive approach we may even relaxe the preconditions concerning the base system R and still provide a sufficient condition for inductive validity as follows.

Assume that R is terminating with a reduction ordering $>$ but not necessarily ground confluent. Then the notion of (provable) inconsistency is not adequate any more. Indeed, a ground proof of the form $s' \,{}_R{\leftarrow^*}\, \sigma s \leftrightarrow_C \sigma t \,{\rightarrow^*_R}\, t'$ with $s = t \in C$ and s',t' R-irreducible and distinct does not necessarily indicate inductive invalidity of the conjecture $s = t$, because $s' \,{\leftrightarrow^*_R}\, t'$ may still be possible. But the following modified version of lemma 3.10 still holds using the same proof simplification argument as before.

Lemma 3.12

Let $>$ be a reduction ordering and R a $>$-ordered rewrite system. Assume further that a set $C = CT \cup CU$ of equational conjectures with CT $>$-ordered and CU $>$-uncomparable is given such that all left hand sides of CT and all equations of CU are inductively reducible.

Then $C \subseteq \mathrm{ITh}(R)$ if all ground instances of critical pair overlaps corresponding to $\mathrm{CP}(R,CT) \cup \mathrm{CP}(R,CU^{\leftrightarrow})$ are equivalently simplifiable with R union C (w.r.t. $>^{pos}$).

■

A sufficient operational criterion for this context is again provided by corollary 3.11which also holds if R is not necessarily ground convergent. Of course one has to replace the condition, that C is not provably inconsistent, by a corresponding inductive reducibility condition.

The main result of [HoKu88] is still another sufficient operational condition for inductive validity w.r.t. a terminating, but not necessarily ground convergent system R.

Lemma 3.13 ([HoKu88])

Let R, $C = CT \cup CU$, $L = LT \cup LU$ be rewrite systems such that

(1) $R \cup CT \cup LT$ is terminating,

(2) $L \subseteq \mathrm{ITh}(R)$,

(3) the left hand sides of C are inductively reducible,

(4) for every critical pair $(s,t) \in \mathrm{CP}(R,CT)$:

(*) $\quad s \,{\rightarrow^*_{R\cup CT\cup LT}}\circ \,{\rightarrow\!\!\!=}_{CU} \circ \,{\leftarrow^*_L}\circ \,{}_{R\cup CT\cup LT}{\leftarrow^*}\, t$

(5) for every critical pair $(s,t) \in \mathrm{CP}(R,CU)$:

(**) $\quad s \,{\rightarrow^*_{R\cup CT\cup LT}}\circ \,{\rightarrow\!\!\!=}_{CU} \circ \,{\leftarrow^*_L}\, t$

Then $C = CT \cup CU \subseteq \mathrm{ITh}(R)$. ■

In [HoKu88] this is proved by noetherian induction according to the well-founded ordering $\rightarrow^+_{R\cup CT\cup LT}$. Using our positive approach for inductive proofs the above result can be proved more elegantly even in a generalized version. For that purpose we define another "positive" complexity cpos2 of proofs using $R \cup C$ as follows.

The complexity of a single proof step $s \rightarrow_C t$ or $t \,{}_C{\leftarrow}\, s$ is the multiset $\{s\}$. The complexity cpos2(P) of a proof P of the form $s_0 \,{\leftarrow^*_{R\cup C}}\, s_n$ is defined to be the multiset union of the complexities of all C-steps in C. Complexities of proofs are compared by $>^{cpos2}$, which is taken to be the multiset extension of $\rightarrow_{R\cup LT\cup CT}$. Then we define $P >^{pos2} P'$ iff $\mathrm{cpos2}(P) >^{cpos2} (P')$. Again, $>^{pos2}$ is a proof reduction ordering. With these definitions it is easily verified that for every ground

instance of a critical overlap of

(4) $s\ {}_R\!\leftarrow \circ \rightarrow_{CT} t$ and (5) $s\ {}_R\!\leftarrow \circ \rightarrow_{CU} t$ the corresponding ground instance of (*) and (**), respectively, is equivalent but smaller w.r.t. $>^{pos2}$ (after replacing the L-steps by R-steps). Moreover, this simplifiability argument still applies if instead of (*) we use

$$(*')\ s \xrightarrow{*}_{R\cup CT\cup LT} \circ\ -\!=\!\rightarrow_{CU} \circ \xleftrightarrow{*}_{L} \circ\ {}_{CU}\!\leftarrow\!=\!- \circ\ {}_{R\cup CT\cup LT}\!\xleftarrow{*} t.$$

Even more generally we may allow in (*) and (**) any mixture of R,L and C-steps provided that certain conditions for C-steps are satisfied, i.e. we can replace (4) and (5) by

(6) For every critical pair overlap $s\ {}_R\!\leftarrow o \rightarrow_{CT\cup CU} t$ there exists a proof P for $s = t$ of the form $s \xleftrightarrow{*}_{R\cup L\cup C} t$ such that for every proof step in P of the form $u \rightarrow_{CT\cup CU} v$ or $v\ {}_{CT\cup CU}\!\leftarrow u$ we have $o > u$, where $>$ is any reduction ordering containing $\rightarrow_{R\cup CT\cup LT}$.

We have seen that the framework of proof orderings together with the concepts of inductive reducibility and (positive or negative) covering sets provide the basis for completion based inductive theorem proving. Algorithmic realizations of the method now consist in testing conjectures on inductive reducibility and computing (and simplifying) covering sets until a provably inconsistent conjecture is detected or a self-covering set is obtained. For ground convergent base systems R the method is unfailing in the sense that no failure case is possible due to situations where one does not know how to continue, and refutationally complete under some reasonable fairness assumption.

If the requirement that R must be ground confluent is omitted, our positive approach still provides sufficient (abstract and operational) criteria for inductive validity but in this case failure situations are possible.

Under certain conditions the method may also be generalized to non-terminating base systems R. This is achieved by using equational rewriting and completion techniques (cf. [JoKi86]) and testing for inductive reducibility w.r.t. the corresponding equational rewriting relation as it has been sketched in [JoKo86]. Whereas inductive reducibility remains decidable for the AC-case (cf. [JoKo86]), i.e. the equational part of the base system consists of the associativity and commutativity axioms for some function symbols, this problem becomes undecidable in general as shown in [KaNaRoZh87]. The details of such an extension of completion based inductive proof techniques to equational rewriting systems (as well as to conditional systems) still have to be worked out and constitute an interesting field of current and future research.

4. Refinements and Optimizations

Let us assume in the following that again $>$ is a reduction ordering, and R a $>$-ordered ground convergent rewrite system.

Minimization of Covering Sets

As we have seen the computation of covering sets is fundamental for completion based inductive

proof techniques, both in the negative and positive approach. In order to obtain terminating inductive proofs for an initial starting set C of not provably inconsistent conjectures it is necessary to compute covering sets until eventually a self-covering set C' is reached. Hence the minimization of covering sets is crucial in order to avoid infinite computations. As mentioned before a first (standard) optimization consists in computing only $CP(R,CT) \cup CP(R,CU^{\leftrightarrow})$ instead of $CP(R,C^{\leftrightarrow})$ where $C = CT \cup CU$ and CT,CU denote the sets of $>$-orientable and $>$-uncomparable conjectures. Stronger restrictions are possible if R-reducibility of all ground instances of s (for $s = t \in C, \neg(t > s)$) is given at certain positions in s. This observation leads to

Definition 4.1 ([Fr86])
A position p in a term s is said to be (inductively) **complete** (w.r.t. R) if for every ground instance σs with σx (R-) irreducible for every $x \in V(s)$, σs is (R-) reducible at position p.

If p is a complete position in s for $s = t \in C, \neg(t > s)$, then it suffices to take $CP(R,p,\{s \rightarrow t\})$ as a covering set for s = t. It is straightforward to generalize this idea to a complete set P of positions for s yielding $CP(R, P, \{s \rightarrow t\})$ as a covering set for $\{s = t\}, \neg(t > s)$ (cf. [Kü87]). Moreover, one may get an even smaller covering set $CP(R', P, \{s \rightarrow t\})$ with $R' \subseteq R$, if for establishing the completeness property of P the subset R' of R suffices (cf. [Kü87]). Thus, if we think of R as a base set of function definitions, we may use alternate equivalent definitions for a completion based inductive proof. Such a usage of alternate function definitions has already been described in [Gö85] but without restricting the computation of critical pairs to complete (sets of) positions. Compared to classical inductive theorem proving choosing a complete position corresponds to fixing an induction term and the induction variable(s), whereas choosing a specific function definition corresponds to fixing the induction scheme. Due to this analogy techniques from classical inductive theorem proving for finding "good" induction terms and induction schemes (cf. [BoMo79]) now can also be used for completion based inductive proofs. Note that different choices of a complete (set of) positions together with a corresponding set of definition rules allow to perform independent inductive proofs in parallel.

Elimination of subsumed conjectures

Another observation may also be very useful in practice to minimize covering sets. Assume that the set of conjectures C contains two equations s = t and $u[\sigma s] = u[\sigma t]$. Then the latter conjecture is subsumed by s = t and can be safely eliminated because any covering set for s = t also is a covering set for $u[\sigma s] = u[\sigma t]$. Note that this subsumption principle is inherent in the refined version of corollary 3.11 (using condition (**)).

Using non-equational inductive knowledge

We have seen that equational lemmas may be used for simplifying conjectures and covering sets. But it is also possible to profit of non-equational inductive knowledge, in particular implications

and equivalences (cf. [Pa84]). This may considerably speed up the proof or disproof of conjectures. For instance, if we take the usual specification for natural numbers, we may provide the following non-equational inductive knowledge which is either positive, e.g.

(1) $x+y = x+z \Rightarrow y = z$
(2) $s(x)*y = s(x)*z \Rightarrow y = z$
or negative, e.g.
(3) $x+s(y) = 0 \Rightarrow$ true = false
(4) $s(x)*s(y) = 0 \Rightarrow$ true = false,

where true = false indicates inconsistency. Thus, if a conjecture s = t entails a covering set containing an equation e.g. of the form u+v = u+w, we may deduce v = w and hence delete u+v = u+w due to the subsumption principle. In the case that we have a conjecture of the form u+s(v) = 0, we can immediately infer disproof from (3), whereas without (3) the inconsistency might be detected much later. The form of the inference rules (1) - (4) may be generalized to Horn clauses where equality is the only predicate. Applying such a Horn clause for inferring a new equation from a set of conjectures then amounts to what is called hyperresolution in resolution theorem proving (cf. [Pa84]).

The special case of free constructors

If R is a ground convergent base system over a signature $F = C \cup D$ such that the set T(C) of ground terms built over C exactly consists of all ground normal forms of R, then the operators from C and D are called (free) constructors and defined operators, respectively. In this practically important case which has been studied in detail in [HuHu82]), the problem of inductive reducibility which is very hard in general becomes trivial, because any term s is inductively reducible iff it contains a defined function symbol. Moreover the following non-equational inductive knowledge is provided gratuitously:

For every constructor $c \in C$, say of arity n, we have

$$c(s_1,...,s_n) = c(t_1,...,t_n) \in \mathrm{ITh}(R) \Leftrightarrow s_1 = t_1,...,s_n = t_n \in \mathrm{ITh}(R).$$

Hence $\{s_1 = t_1,..., s_n = t_n\}$ is a covering set for $\{c(s_1,...,s_n) = c(t_1,...,t_n)\}$. Moreover every conjecture of the form $c_1(...) = c_2(...)$, where c_1, c_2 are distinct constructors, cannot be an inductive theorem of R. This negative knowledge may even be generalized to conjectures of the form c(...) = x or x = y, provided that at least two different constructors (for the corresponding sort) exist.

Refined Interpretation of Inductive Completion Results

If inductive completion starts with an initial set C of conjectures and eventually encounters an inconsistency, we know that there is an equation in C that is not inductively valid. But there may be others which have indeed been proved. In order to improve such an incomplete knowledge one

may incorporate some kind of memory mechanism into the inductive completion procedure. For that purpose it is necessary to keep for every conjecture the information where it stems from, i.e. whether it was produced as an element of a covering set of a former conjecture or obtained by simplification of another conjecture. In the latter case the conjectures used for simplification have to be stored, too. From the resulting conjecture dependency graph valid partial proofs can then be extracted (cf. [Kü87]). But in general such an approach may be very complicated and expensive.

A reasonable compromise in practice consists in proceeding incrementally. That means, try first to prove simple, basic and auxiliary conjectures, before attacking the main conjecture(s). Of course, the question which auxiliary lemmas might be useful to succeed in a proof for a main conjecture is very difficult in general. Inductive completion may be seen as a straightforward way of producing auxiliary conjectures. Namely, if an initial conjecture s = t entails a covering set C then any equation from C that cannot be simplified to a trivial equation may be considered to be a non-trivial auxiliary conjecture for proving s=t. But, as it is well-known from practical experience, this mechanism often is not sufficient for successful inductive proofs. Technically such a situation is reflected by divergence of inductive completion methods, i.e. more and more auxiliary conjectures are generated. This phenomen closely corresponds in the classical approach to the situation, that for succeeding in the induction step, a new lemma is needed the proof of which again leads to a "gap" in the corresponding induction step, and so on.

Using inessential critical pairs as auxiliary conjectures

As mentioned above it is possible to restrict the computation of critical pairs of $R \cup C$ to overlapping R into C at complete positions. Doing so, divergence of general completion may often be avoided. As an example (cf. [Fr86]) let R be the definition of natural number addition given by $0+x \rightarrow x$, $s(x)+y \rightarrow s(x+y)$ and C consist of the conjecture A: $x+(y+z) = (x+y)+z$. Here, general completion provided with an appropriate reduction ordering that directs A from left to right diverges, i.e. does not terminate. But identifying the top position ε in the left hand side of A as a complete position and restricting the covering set to be computed to $CP(R,\varepsilon,C)$ leads to a successful proof. But in other cases such a restriction leads to divergence whereas general completion successfully terminates. A famous example (cf. [HuHu82]) for this phenomenon is the proof of the involution property C: rev(rev(x)) = x for reversing lists, where R consists of the usual recursive definitions for app(end) and rev(erse) using constructors nil and cons. Restricting critical pairs to CP(R,C) results in divergence, whereas computing all critical pairs lead to a successful proof producing the auxiliary lemma rev(app(x,cons(n,nil))) = cons(n,rev(x)). These observations may be exploited as follows (cf. [Kü87]). One may compute all critical pairs of $R \cup C$, in particular also CP(C,C) by overlapping conjectures, but distinguish between those that contribute to covering sets ("essential" critical pairs) and others that are not really necessary ("inessential" ones). Inessential critical pairs may then be useful to simplify essential ones. If such a simplification step is performed during completion then the correctness of this step of course depends on the inductive validity of the inessential equation used. Thus it has to be turned into an essential one which also has to be proved. Note that such a refined version of inductive completion also requires an

appropriate mechanism for keeping track of dependencies between conjectures. Although such a refinement is helpful in some cases for producing important auxiliary conjectures, it is in general very expensive. This is due to the amount of bookkeeping work that has to be performed and to the potentially huge number of inessential critical pairs. Moreover, it is in a sense a purely syntactic and relatively blind method for deducing new conjectures in that it is not goal-directed. That means when computing an inessential critical pair we do not know in advance whether it will ever be useful for the original proof task.

Refining (ground) subconnectedness proofs

Assume again that $>$ is a reduction ordering, R a $>$-ordered ground convergent base system, $C = CT \cup CU$ a set of not provably inconsistent conjectures and $L = LT \cup LU$ a set of inductive lemmas, such that CT, LT are $>$-ordered and CU, LU $>$-uncomparable. Now, it may be the case that for some critical pair $(s,t) \in CP(R,CT) \cup CP(R,CU^{\leftrightarrow})$ with superposition term o it is impossible to find a proof P for $s = t$ of the form

(**) $s \xrightarrow{*}_{R \cup CT \cup LT} \circ \leftarrow\!\!=\!\!\rightarrow_{LU \cup CU} \circ {}_{R \cup CT \cup LT}\!\xleftarrow{*} t$ (see Corollary 3.11).

Nevertheless it may be possible that an equivalent proof P' of the form

(*) $s \xleftrightarrow{*}_{R \cup C \cup L} t$

exists, such that the complexity (cpos1) of each C-step in P' is smaller than cpos1(o,t). But whereas (**) may be effectively tested, this is not possible in general for (*). What can instead be done is to use some heuristics for finding a proof of the form (*). Closely related to this problem is the choice of the underlying reduction ordering $>$ as well as the choice of some appropriate subset of already available lemmas. Let us illustrate these aspects a little bit more detailed by means of an example borrowed from [HoKu88].

Example 4.1 (binomial coefficients)

Assume that we are given the following specification $R_0 \cup R_1$ for natural numbers with addition and binomial coefficients.

R_0:	(a1)	0+y	$\Rightarrow$	y
	(a2)	s(x)+y	$\Rightarrow$	s(x+y)
R_1:	(b1)	b(0,s(k))	$\Rightarrow$	0
	(b2)	b(n,0)	$\Rightarrow$	s(0)
	(b3)	b(s(n),s(k))	$\Rightarrow$	b(n,s(k)) + b(n,k)

Taking the recursive path ordering induced by the precedence $>_F$ with $b >_F + > s$ as reduction $>$ ordering, $R := R_0 \cup R_1$ is $>$-ordered and ground convergent (it is even convergent). Assume further that the following set L of ($>$-ordered) inductive lemmas of R_0 (as well as of R) is available:

L:	(a3)	x+0	$\Rightarrow$	x
	(a4)	x+s(y)	$\Rightarrow$	s(x+y)

Let us now try to prove that

(b4) b(n,n+s(m)) = 0

is an inductive consequence of R. Computation of a covering set for (b4) requires to consider the

critical overlaps corresponding to CP({(a1),(a2)},2,(b4):

(i) $b(0,s(m)) \;{}_{(a1)}\!\leftarrow\; b(0,0+s(m)) \rightarrow_{(b4)} 0$ which is simplified to
$b(0,s(m)) \rightarrow_{(b1)} 0$, and

(ii) $b(s(n),s(n+s(m)) \;{}_{(a2)}\!\leftarrow\; b(s(n),s(n)+s(m)) \rightarrow_{(b4)} 0$. Here simplification yields
$b(s(n),s(n+s(m))) \rightarrow_{(b3)} b(n,s(n+s(m))) + b(n,n+s(m))$
$\rightarrow_{(b4)} b(n,s(n+s(m))) + 0 \rightarrow_{(a3)} b(n,s(n+s(m)))$, but
$b(s(n),s(n+s(m)) \xrightarrow{*}_{R\cup L\cup\{b4\}} \circ \;{}_{R\cup L\cup\{b4\}}\!\xleftarrow{*} 0$ is impossible.

Thus the simple version of inductive completion requiring rewrite proofs for critical pairs diverges producing the infinite sequence of rules

(b4k) $b(n,s^k(n+s(m)) \rightarrow 0$,

if only (a3) is used as a lemma, or

(b4k') $b(n,s^{k+1}(n+m)) \rightarrow 0$,

if (a3) and (a4) are available as lemmas. In order to close the "gap" above between b(n,s(n+s(m))) and 0 we use an L-step in reverse direction as follows:

$b(n,s(n+s(m)) \;{}_{(a4)}\!\leftarrow\; b(n,n+s(s(m))) \rightarrow_{(b4)} 0.$

But we have to verify that b(n,n+s(s(m))) is smaller than the corresponding superposition term b(s(n),s(n)+s(m)). This is not the case for > as above. But when we modify > into a recursive path ordering >' with status such that status(b) = left-to-right this condition is satisfied and all equations of $R \cup L \cup \{b4\}$ are oriented as before. Using still another reduction ordering based on monotonic interpretations (cf. [De87] one may even get a rewrite proof equivalent to (ii) (cf. [HoKu88]).

Indeed, it turns out that in order to succeed in the above example some non-trivial intelligent decisions are necessary. Useful heuristics for the question which equation (from $R \cup L \cup C$) should be used at which step in a proof simplification process as above might be developed from a more goal-directed approach. In the example the goal to prove b(n,s(n)+s(m)) = 0 can be reduced to the subgoal of how to transform b(n,s(n+s(m))) into another term such that the "induction hypothesis" (b4) $b(n,n+s(m)) \Rightarrow 0$ becomes applicable (again). For that purpose one may try to eliminate the "bad" parts of b(n,s(n+s(m))). Here, instead of the subterm s(n+s(m)) we need a subterm of the form n+s(m') to make (b4) applicable. Indeed, the problematic "context s" at top position can be moved "downward" using (a4) in reverse direction yielding n+s(s(m)) which is of the desired form. An attempt to formalize such ideas of goal-directed (equational) reasoning and an extended example is given in [Hu89] (for classical inductive theorem proving).

Refinement by Case Analysis

A further technique for obtaining simpler proofs for ground instances of critical overlaps consists in splitting the problem into (hopefully simpler) subproblems. Let us demonstrate this technique again via an example (from [Gö88]).

Example 4.2

Let > be the rpos induced by an empty precedence on {e,f} and with status (f) = left-to-right, and

R, C be given as follows in >-ordered form:

R: (1) $f(e,x) \to x$

(2) $f(f(x,y),z) \to f(x,f(y,z))$

C: (3) $f(x,x) \to x$

In order to show that (3) is an inductive consequence of R = {(1),(2)} we first verify that the left hand side f(x,x) of (3) obviously is inductively reducible. The first critical overlap corresponding to CP((1),(3)) to be considered is trivial, but for the second one we have to simplify all ground instances of the proof

P: $f(x,f(y,f(x,y))) \;{}_{(2)}\!\leftarrow\; f(f(x,y),f(x,y)) \;\rightarrow_{(3)}\; f(x,y)$

Now, it is impossible to simplify P itself but it would also suffice to simplify $P_1 := \sigma_1 P$ and $P_2 := \sigma_2 P$ with $\sigma_1 := \{x \leftarrow e\}$, $\sigma_2 := \{x \leftarrow f(u,v)\}$, because σ_1 and σ_2 consitute a complete case distinction for the ground case. Hence we get

P_1: $f(e,f(y,f(e,y))) \;{}_{(2)}\!\leftarrow\; f(f(e,y),f(e,y)) \;\rightarrow_{(3)}\; f(e,y)$, which can be simplified to

P_1': $f(e,f(y,f(e,y))) \;\rightarrow_{(1)}\; f(e,f(y,y)) \;\rightarrow_{(3)}\; f(e,y)$ and

P_2: $f(f(u,v),f(y,f(f(u,v),y))) \;{}_{(1)}\!\leftarrow\; f(f(f(u,v),y),f(f(u,v),y)) \;\rightarrow_{(3)}\; f(f(u,v),y)$

which is simplifiable to

P_2': $f(f(u,v),f(y,f(f(u,v),y))) \;\rightarrow_{(1)}\; f(f(u,v),f(y,f(u,f(v,y))))$

$\;{}_{(1)}\!\leftarrow\; f(f(f(u,v),y),f(u,f(v,y))) \;\rightarrow_{(1)}\; f(f(u,f(v,y)),f(u,f(v,y)))$

$\;\rightarrow_{(3)}\; f(u,f(v,y)) \;{}_{(1)}\!\leftarrow\; f(f(u,v),y).$

Note in particular that the intermediate "peak" result f(f(f(u,v),y),f(u,f(v,y))) in P_2' is smaller than the superposition term f(f(f(u,v),y),f(f(u,v),y)) (w.r.t. >). Hence (3) is indeed an inductive consequence of R. Using inductive completion without such an elaborate case analysis again results in divergence. The question of how to perform a case analysis as above is treated in detail in [Gö88] where a general ground confluence criterion is developed. In fact the different cases are obtained there by overlapping again into the original superposition term. Of course one has to verify that the resulting case distinction is complete, and if not, to consider the remaining cases, too.

5. Generalization Techniques

All the refinements and optimizations presented in the previous chapter are not sufficient in many cases for successful inductive proofs. This may be due to the lack of some important auxiliary lemma. But it may also be the case that the induction hypothesis provided by the conjecture is too weak to be applicable within the induction step. Or in other words, it may be easier to prove a conjecture which is more general than the original one, because the corresponding induction hypothesis is also stronger and thus may become applicable now. Within the field of classical inductive theorem proving generalization techniques and heuristics have been extensively studied (cf. [Au79],[BoMo79],[Ca85],[VyAb85],[Hu87] among others). All these techniques can also be used (in more or less modified form) for completion based inductive theorem proving. In the following we will briefly summarize some of these techniques including a classification scheme. Moreover a new technique for save generalization is presented. Let us start with

Definition 5.1
Let E, C, C' be sets of equations over a fixed signature. We say that
C' is an **(equational) generalization** of C (w.r.t. E) if $C \subseteq Th(E \cup C')$
C' is a **syntactical generalization** of C if $C \subseteq Th(C')$
C' is an **inductive generalization** of C (w.r.t. E) if $C \subseteq ITh(E \cup C')$.

Clearly every syntactical generalization C' of C also is an equational generalization (w.r.t. an arbitrary E) and every equational generalization C' of C (w.r.t. some E) also is an inductive generalization of C (w.r.t. the same E). Now, most generalization techniques are based on equational or even syntactical generalization since appropriate and non-trivial inductive generalizations are hard to obtain. Syntactical generalizations are constructed according to the replacement and substitution axioms of equality. This yields the following generalization rules:

Generalization w.r.t. the Subsumption Quasi-ordering

If for all $u = v \in C$ there exists $s = t \in C'$ and σ with $\sigma s = u$ and $\sigma t = v$ then C' is a syntactical generalization of C.

If we consider equations as ordered pairs of terms and require the generalization to be a single equation, then there exists a minimal generalization which is unique up to variable renaming and may be effectively computed (cf. [Hu80]).

In practice one often considers a single equation s = t as a candidate for generalization. In order not to generalize too strong, one may then search for minimal non-variable common subterms in s and t and replace (some of) them by a new variable. This can be restricted to independent common subterms of s = t, i.e. the variables in the subterms of s = t to be replaced do not occur anywhere else in s = t. Thus variable bindings between the common subterms to be replaced and the context in s = t won't get lost.

Example: $(x+y)+0 = x+y \rightarrow z+0 = z$.

Another elementary generalization step consists in splitting variables, i.e. replace some occurrence(s) of a variable in s = t by a new variable.

Example: $x+(x+x) = (x+x)+x \rightarrow y+(x+x) = (x+x)+y$.

By combining both kinds of generalization steps we obtain for instance

$$x+(x+x) = (x+x)+x \rightarrow y+z = z+y.$$

Generalization by Extraction

According to the replacement axiom for equality one may establish $f(s_1,...,s_n) = f(t_1,...,t_n)$ by extracting and equating the corresponding argument pairs yielding $s_1 = t_1,...,s_n = t_n$. Of course, we may ignore any resulting trivial equation.

Example: $x+((y+0)+z) = x+(y+z)$
$\rightarrow$ $(y+0)+z = y+z$
$\rightarrow$ $y+0 = y$

Equational Generalization

Equational generalizations (w.r.t. E) are obtained by combining syntactical generalization steps with E-steps.

Example: $(x+y)+0 = y+x \rightarrow z+0 = z$, with $E = \{x+y = y+x\}$.

The technique of accumulator introduction also falls into this category (cf. [Au79]).

Example 5.1
Assume for illustration that app(end) and rev(erse) for lists over some element type are defined by
(1) app(nil,y) → y
(2) app(cons(n,x),y) → cons(n,app(x,y))
(3) rev(nil) → nil
(4) rev(cons(n,x)) → app(rev(x),cons(n,nil))
and
(L1) app(x,nil) = x
(L2) app(app(x,y),z) = app(x,app(y,z))
are available as inductive lemmas. Assume further that another function revit on lists is defined as follows:
(5) revit(nil,y) → y
(6) revit(cons(n,x),y) → revit(x,cons(n,y))
Then the conjecture (7) revit(x,nil) = rev(x) saying that revit provides an alternative (iterative) version for reversing lists may be equationally generalized to

(8) revit(x,y) = app(rev(x),y)

using E = {(L1)}. A completion based inductive proof for (8) is easily performed (with an appropriate reduction ordering and making use of (L2)) whereas a proof of (7) fails due to divergence.

Another subcase of equational generalization which is very important in practice, will also be demonstrated via the following simple but extended

Example 5.2

Assume that the addition on natural numbers is defined by

(1) $0+x = x$

(2) $s(x)+y = x+s(y)$

and that we want to prove the inductive conjecture

(L_0) $x+0 = x$.

Now, inductive completion starting with (1) - (3) ordered from left to right (by an appropriate rpos) diverges producing

(L_1) $x+s(0) = s(x)$

(L_2) $x+s(s(0)) = s(s(x))$

. . . .

(L_n) $x+s^n(0) = s^n(x)$

. . . .

The new equations (rules) (L_n) cannot be generalized syntactically to a common generalization that will still be inductively valid. But combining (L_n) with (L_{n+1}) as follows leads to (infinitely many) new equations (L_n') that can be generalized syntactically into a non-trivial new conjecture (cf. [Gr85]). Identifying the subterm of the right hand side of (L_{n+1}) at position 1 as the right hand side of (L_n) and replacing this subterm by the corresponding left hand side of (L_n) we obtain:

(L_0),(L_1): $x+s(0) = s(x+0)$ (L_0')

(L_1),(L_2): $x+s(s(0)) = s(x+s(0))$ (L_1')

. . . .

(L_n),(L_{n+1}): $x+s(s^n(0)) = s(x+s^n(0))$ (L_n')

. . . .

Obviously, all these equations (L_n') can now be generalized syntactically into (L) $x+s(y) = s(x+y)$ by replacing the (independent) common subterm $s^n(0)$ by the new variable y. Inductive completion with conjecture set $C = \{(L_0),(L)\}$ now immediately succeeds. This technique of combination of equations plus subsequent generalization of non-trivial common subterms corresponds to what is called cross-fertilization in classical inductive theorem proving (see e.g. [BoMo79]).

Now, all presented generalization techniques as well as many (more or less sophisticated) refinement entail the danger that the property of being inductively valid does not remain invariant. Hence, if a generalization turns out not to be inductively valid, we can in general conclude nothing about the inductive validity of the original conjecture. But in some cases we can get rid of this uncertainty as follows:

Safe Inductive Generalization

Definition 5.2

Let E, C, C' be sets of equations over a fixed signature. We say that C' is a **safe** (inductive) generalization of C (w.r.t. E) iff $C \subseteq ITh(E \cup C')$ and additionally $C \subseteq ITh(E) \Leftrightarrow C' \subseteq ITh(E)$.

One important application of safe generalization is provided in [Gr85] for the case of specifications with constructors. Assume that E_0 is a base specification over F_0 and $E = E_0 \cup E_1$ is a complete and consistent enrichment of E_0 over $F = F_0 \cup F_1$. In this case F_0 may be considered to be a set of constructors (which are free for $E_0 = \emptyset$) and F_1 the set of defined operators.

Definition 5.3
Let E_0, F_0, E_1, F_1 be given as above. Then we say that a term $t \in T(F,V)$ is **complete** for $T(F_0)$ (w.r.t. E) iff for every $t' \in T(F_0)$ there exists a ground instance $\sigma t \in T(F)$ such that $\sigma t =_E t'$.

Examples for complete terms are x+y in example 5.2 or rev(x) in example 5.1 whereas x+x and app(x,x) are not complete. We can now safely generalize an inductive conjecture by replacing independent complete subterms of a conjecture by new variables.

Lemma 5.1
Let E_0, F_0, E_1, F_1 be given as above and let $s = t$ be an inductive conjecture over $F = F_0 \cup F_1$. Assume further that $r \in T(F,V)$ occurs as subterm of s at positions $u_1,...,u_m$ and as subterm of t at positions $v_1,...,v_n$ such that the variables of r do not occur in s and t outside of the subterms s/u_i and t/v_j, respectively. Moreover, let $s' = t'$ be the syntactical generalization obtained from $s = t$ by replacing all the s/u_i, t/v_j by the (same) new variable z. Then $s' = t'$ is a safe generalization of $s = t$ if r is complete for $T(F_0)$.

Proof: see [Gr85]. ■

For instance, in example 5.1we may safely generalize the conjecture app(rev(x),nil) = rev(x) into the syntactically simpler form app(z,nil) = z.

In practice this technique of save generalization may be supported by collecting the information that certain terms possess the above completeness property in a kind of inductive knowledge base.

6. UNICOM: A Refined Completion Based Inductive Theorem Prover

Based on the presented theoretical framework we have implemented UNICOM, a system for refined **un**failing **i**nductive **com**pletion which is described in detail in [Sc88]. UNICOM is a central tool of TRSPEC, a **t**erm **r**ewriting based system for investigating algebraic specifications (cf. [AvGöGrMaSt87]. TRSPEC is able to treat hierarchically structured many-sorted specifications of functions (rewrite programs) and inductive conjectures (properties to be proved). Input specifications have to satisfy the following conditions. Constructors have to be declared and are required to be free. The left hand sides of the definition rules for a (n-ary) non-constructor symbol f have to be of the form $f(t_1,...,t_n)$, where all t_i are constructor terms. The system

comprises the following tools:

The **parser** checks the syntax of input specifications, imports used subspecifications and produces an internal representation.

The **checker** tests the function definitions for completeness and consistency using the syntactical, restrictions mentioned above. In particular, termination of the definition rules is established by automatically generating a suitable recursive path ordering with status which is also used for subsequent inductive proofs. Ground convergence is established by investigating critical pairs. A special feature of the implemented completeness test allows to identify minimal complete sets of defining rules for the same function symbol, i.e. alternate but equivalent function definitions.

The **compiler** provides a means for rapid prototyping by translating correct specifications (only the definition part) into executable LISP code.

The central part of the system is the **prover** which tries to prove or disprove the inductive conjectures of the actual specification. The original completion based prover of TRSPEC was already able to handle certain non-orientable conjectures leading to globally finite instead of terminating rewriting systems (cf. [Gö85]). UNICOM now admits arbitrary possibly non-orientable conjectures.

The following features are characteristic for UNICOM and partly also for the old prover:

- parallel independent inductive proofs are possible according to the different possibilities of choosing complete positions in conjectures together with corresponding minimal complete function definitions

- inessential critical pairs may be computed as potentially useful auxiliary conjectures

- a simple generalization technique (looking for minimal non-variable common subterms) is available

- elimination of subsumed non-orientable conjectures is integrated

- non-equational inductive knowledge about free constructors is used for speeding up the proof or disproof of conjectures

- various user interface parameters allow for switching on/off optional features (e.g. generalization, computation of inessential critical pairs) and enable a fully automatic or more or less strongly user-controlled running mode.

Numerous experiments with UNICOM have shown that many inductive properties of common abstract specifications for booleans, natural numbers, lists, stacks etc. can easily be proved using such a flexible completion based inductive proof technique (see the appendix for an example session). But there are also interesting examples where the method fails, i.e. diverges generating more and more equations. In such cases a more careful analysis and good heuristics for applying the refinements, optimizations and generalization techniques discussed above are necessary.

7. Conclusion

We have presented an overview of completion based inductive theorem proving techniques. The key concepts of the underlying theoretical framework have been pointed out. In particular we have shown how to get "positive" and "negative" abstract characterizations of inductive validity as well as easily understable and generalized operational criteria. Furthermore we have presented and discussed various practically important refinements, optimizations and generalization techniques. The theoretical investigations as well as practical experiments with our implementation of UNICOM indicate that challenging problems remain to be solved, e.g.:

(1) How can we design good strategies and heuristics for applying and combining the refinements, optimizations and generalization techniques presented ?

(2) What should a powerful, comprehensible and flexible completion based inductive theorem prover look like, that is able to combine automatic tools as well as human intuition and intelligence ?

References

[Au79] Aubin, R.: Mechanizing Structural Induction, Part I: Formal System, Part II: Strategies, TCS, Vol. 9, 1979

[AvGöGrMaSt87] Avenhaus, J., Göbel, R., Gramlich, B., Madlener, K, Steinbach, J.: TRSPEC: A Term Rewriting Based System for Algebraic Specifications, Proc. of the 1st International Workshop on Conditional Term Rewriting Systems, Orsay, France, 1987, LNCS 308, eds. S. Kaplan, J.-P. Jouannaud, 1988

[Ba87] Bachmair, L: Proof Methods for Equational Theories, PhD Thesis, Univ. of Illinois, Urbana Champaign, 1987

[Ba88] Bachmair, L.: Proof by Consistency in Equational Theories, Proc. of LICS, pp. 228-233, 1988

[BaDeHs86] Bachmair, L., Dershowitz, N., Hsiang,J.: Orderings for equational proofs, Proc. of Symb. Logic in Computer Science, Bostin, Massachusetts, pp. 346-357, 1986

[BaDePl87] Bachmair, L.; Dershowitz, N., Plaisted, D.A.: Completion without failure, CREAs Proc., Lakeway, Texas, 1987

[BoMo79] Boyer, R., Moore, J.: A Computational Logic, Academic Press, 1979

[BüKü89] Bündgen, R., Küchlin, W.: Computing Ground Reducibility and Inductively

Complete Positions, Proc. 3rd RTA, Chapel Hill, North Carolina, USA, LNCS 355, pp. 59-75, 1989

[Bu69] Burstall, R.: Proving properties of programs by structural induction, Computer Journal 12/1, pp. 41-48, 1969

[Ca85] Castaing, J.: How to facilitate the proof of theorems by using induction-matching and by generalization, Proc. 9th IJCAI, Los Angeles, pp. 1208-1213, 1985

[De82] Dershowitz, N.: Applications of the Knuth-Bendix completion procedure, in Proc. of the Seminaire d'Informatique Theorique, pp. 95-111, Paris, France, 1982

[Fr86] Fribourg, L.: A strong restriction of the inductive completion procedure, Proc. 13th ICALP, Rennes, France, LNCS 226, pp. 105-116, 1986

[Go80] Goguen, J.A.: How to prove algebraic inductive hypotheses without induction, Proc. of 5th CADE, ed. W. Bibel and R. Kowalski, LNCS 87, pp. 356-373, 1980

[Gö85] Göbel, R.: Completion of Globally Finite Term Rewriting Systems for Inductive Proofs, Proc. of GWAI 85, Springer Verlag, 1985

[Gö87] Göbel, R.: Ground Confluence, Proc. 2nd RTA, Bordeaux, France, LNCS 256, 1987

[Gö88] Göbel, R.: A Completion Procedure for Generating Ground Confluent Term Rewriting Systems, Dissertation, FB Informatik, Universität Kaiserslautern, Feb. 1988

[Gr85] Gramlich, B.: Zum Beweisen durch Reduktion und Induktion, Diplomarbeit, Fakultät für Informatik I, Universität Karlsruhe, 1985

[HoKu88] Hofbauer, D., Kutsche, R.-D.: Proving inductive theorems based on term rewriting systems, Techn. Report 88-12, TU Berlin, 1988, see also Proc. of an International Workshop on Algebraic and Logic Programming, Gaussig, GDR, Nov. 1988

[HsRu86] Hsiang, J., Rusinowitch, M.: On word problems in equational theories, Techn. Rep. 86/29, SUNY at Stony Brook, USA, 1986

[Hu80] Huet, G.: Confluent reductions: abstract properties and applications to term rewriting systems, JCSS 25, pp. 239-266, 1982

[Hu87] Hummel, B.: An investigation of formula generalization heuristics for induction proofs, Int. Bericht 6/87, Institut f. Logik, Komplexität und Deduktionssysteme, Univ. Karlsruhe, 1987

[Hu89] Hutter, D.: Verwendung von Induktionshypothesen in Induktionsbeweisen, Workshop Verifikation, Konstruktion und Synthese von Programmen, Karlsruhe, April 1989

[HuHu82] Huet, G., Hullot, J.: Proofs by Induction in Equational Theories with Constructors, JACM 25(2), 1982

[HuOp80] Huet, G., Oppen, D.C.: Equations and rewrite rules: A survey, in Formal Language Theory: Perspectives and Open Problems, pp. 349-405, ed. R. Book, New York, Academic Press, 1980

[JoKi86] Jouannaud, J.-P., Kirchner, H.: Completion of a set of rules modulo a set of equations, SIAM J. Comp., 15/4, pp. 1155-1194, 1986

[JoKo86] Jouannaud, J.-P., Kounalis, E.: Automatic proofs by induction in equational theories without constructors, Proc. Symb. Logic in Computer Science, pp. 358-366, Boston, Massachusetts, 1986

[KaNaRoZh87] Kapur, D.; Narendran, P., Rosenkrantz, D.J., Zhang, H.: Sufficient Completeness, Quasi-Reducibility and Their Complexity, Techn. Rep. 87-26, Dept. of Comp. Sci., State Univ. of New York at Albany

[KaNaZh86] Kapur, D., Narendran, P., Zhang, H.: Proof by induction using test sets, Proc. of 8th CADE, pp. 99-117, ed. J. Siekmann, LNCS 230, 1986

[Ki84] Kirchner, H.: A general inductive completion algorithm and application to abstract data types, Proc. of 7th CADE, LNCS 170, pp. 282-302, 1984

[Kü87] Küchlin, W.: Inductive Completion by Ground Proof Transformation, Proc. CREAS, Lakeway, Texas, 1987

[Ku88] Kucherov, G.: A new quasi-reducibility testing algorithm and its applications to proof by induction, Proc. of an Int. Workshop on Algebraic and Logic Programming, Gaussig, GDR, 1988

[La80] Lankford, D.: Some remarks on inductionless induction, Techn. Report MTP-11, Mathematics Department, Louisiana Tech. Univ., Ruston, USA, 1980

[La81] Lankford, D.: A simple explanation of inductionless induction, Techn. Report MTP-14, Mathematics Department, Louisiana Tech. Univ., Ruston, USA, 1981

[Mu80] Musser, D.: On proving inductive properties of abstract data types, Proc. 7th ACM Symp. on Principles of Programming Languages, pp. 154-162, Las Vegas, Nevada, USA, 1980

[Pa84] Paul, E.: Proof by induction in equational theories with relations between constructors, Proc. of 9th CAAP, ed. B. Courcelle, Cambridge Univ. Press, 1984

[Pl85] Plaisted, D.A.: Semantic confluence tests and completion methods, Information and Control 65, pp. 182-2151985

[Sc88] Scherer, R.: UNICOM: Ein verfeinerter Rewrite-basierter Beweiser für induktive Theoreme, Diplomarbeit, FB Informatik, Universität Kaiserslautern, 1988

[VyAb85] Vytopil, J., Abdali, S.K.: Generalization Heuristics for Theorems Related to Recursively Defined Functions, Proc. of the National Conference on Artificial Intelligence, Univ. of Texas, at Austin, pp. 1-5, 1984

Appendix: An example session with UNICOM

The following is an example session with UNICOM which should provide a rough idea of how to work with the system. For the sake of readability user inputs are preceded by a "#" and meta comments explaining what is going on start with a ";".

```
; Starting UNICOM produces the following top-level menue

**********************************
*   Program parameter UNICOM :   *
**********************************

single-step : YES
generalize  : NO
protocol    : NO

**********************************
*       Program menue UNICOM     *
*                                *
*                                *
*   1 PARSE-SPECIFICATION        *
*   2 CHECK-SPECIFICATION        *
*   3 PROVE-SPECIFICATION        *
*   4 PARSE-CHECK-AND-PROVE-SPEC *
*   5 COMPILE-SPECIFICATION      *
*   6 RUN-SPECIFICATION          *
*   7 EDIT-SPECIFICATION         *
*   8 CHANGE-STATUS-SINGLE-STEP  *
*   9 CHANGE-STATUS-GENERALIZE   *
*  10 CHANGE-STATUS-PROTOCOL     *
*  11 EXIT                       *
*                                *
**********************************

Type any symbol --->  #4

You choose PARSE-CHECK-AND-PROVE-SPEC

Type the filename ---> #boolean

; We first want to investigate the following specification
; of booleans, in particular we would like to show that
; the given equations are inductive theorems of the set of
; definition rules for "or" and "not" (interpreted as equations).
; First the parser is invoked producing the following output.

SPEC boolean

SORTS bool

OPS t, f :           --> bool
    not  :      bool --> bool
    or   : bool bool --> bool

CONSTRUCTORS t, f

EQUATIONS or(x,y) = or(y,x)
          or(or(x,y),z) = or(x,or(y,z))

RULES not(f) --> t
      not(t) --> f

      or(f,x) --> x
```

```
      or(t,x) --> t

ENDSPEC

; Next the checker tests the specification for various
; properties, which assure in particular that all
; functions which are non-constructors (here: "not" and "or")
; are completely and consistently defined. Note that
; termination is proved by automatically generating an
; appropiate recursive path ordering with status (rpos).

Checking left hand sides of rules

All left hand sides of rules are correct !
==========================================

Checking termination of rules

System is terminating !
=======================

Checking confluence

System is confluent !
=====================

Checking totality of definitions

Checking totality of NOT

1 definition for the function symbol NOT generated !

Checking totality of OR

1 definition for the function symbol OR generated !

All functions are totally defined !
===================================

; If the "RULES"-part of the specification contains
; several alternate but equivalent definitions for
; a non-constructor function this is recognized and
; used later on for inductive proofs. Here there is
; only one definition for "not" and for "or".
;
; Now the prover is invoked trying to prove inductive
; validity of the two equations. This is done in
; parallel and independently for both conjectures
; (hypotheses). Internally the smallest equation is
; processed first (here: commutativity of "or").

          ************************
          *                      *
          *                      *
          *______________________*
          *                      *
          *  1 STOP              *
          *  2 NEXT              *
          *  3 CONTINUE          *
          *  4 HYPS              *
          *  5 ACCEPTED-HYPS     *
          *  6 REJECTED-HYPS     *
          *  7 RULES             *
          *  8 UNCOMP-RULES      *
          *  9 CHANGE-PARAMETER  *
          *                      *
          ************************
```

```
          Type any symbol --->  #3

          You have chosen CONTINUE

; In order to be as flexible as possible the user
; is asked what to do when equations are processed
; that cannot be directed with the current rpos
; (or a possible extension). A save automatic
; variant would be to consider every non-orientable
; equation as oriented "twoway".

Orient uncomparable equation  OR(Y,X) == OR(X,Y)

Straight (1), Reverse (2) or Twoway (3)  --> #3

; Of course, this is the only sound possibiliy here!

New uncomparable equation generated :
(BOOLEAN 5): OR(Y,X) == OR(X,Y)

New rule generated :
LHS BOOLEAN 5, BOOLEAN 3 ==> (BOOLEAN 6): OR(X,F) --> X

New rule generated :
LHS BOOLEAN 5, BOOLEAN 4 ==> (BOOLEAN 7): OR(X,T) --> T

Hypothesis accepted : OR(Y,X) == OR(X,Y)

; Now, commutativity of "or" has been proved
; generating the additional inductive
; theorems "(BOOLEAN 6)" and "(BOOLEAN 7)".
;
; Next the other conjecture is processed

Orient uncomparable equation  OR(OR(X,Y),Z) == OR(X,OR(Y,Z))

Straight (1), Reverse (2) or Twoway (3)  --> #1

; Note that this orientation is sound considering
; an rpos with status(or) = left-to-right. But the
; proof would also succeed choosing the save "twoway"-orientation.

New rule generated : (BOOLEAN 8): OR(OR(X,Y),Z) --> OR(X,OR(Y,Z))

Hypothesis accepted : OR(OR(X,Y),Z) == OR(X,OR(Y,Z))

          **********************
          *                    *
          *                    *
          *--------------------*
          *                    *
          *  1 STOP            *
          *  2 HYPS            *
          *  3 ACCEPTED-HYPS   *
          *  4 REJECTED-HYPS   *
          *  5 RULES           *
          *  6 UNCOMP-RULES    *
          *                    *
          **********************

          Type any symbol --->  #1

          You have chosen STOP

; All proved conjectures are stored and the system
; returns to top-level.
```

```
**********************************
*   Program parameter UNICOM :   *
**********************************

single-step : YES
generalize  : NO
protocol    : NO

**********************************
*       Program menue UNICOM     *
*                                *
*--------------------------------*
*                                *
*    1 PARSE-SPECIFICATION       *
*    2 CHECK-SPECIFICATION       *
*    3 PROVE-SPECIFICATION       *
*    4 PARSE-CHECK-AND-PROVE-SPEC*
*    5 COMPILE-SPECIFICATION     *
*    6 RUN-SPECIFICATION         *
*    7 EDIT-SPECIFICATION        *
*    8 CHANGE-STATUS-SINGLE-STEP *
*    9 CHANGE-STATUS-GENERALIZE  *
*   10 CHANGE-STATUS-PROTOCOL    *
*   11 EXIT                      *
*                                *
**********************************

Type any symbol --->  #4

You choose PARSE-CHECK-AND-PROVE-SPEC

Type the filename ---> #natural

; Now we shall prove the transitivity property of
; the less-or-equal (le) predicate on natural
; numbers.

SPEC natural

USE boolean

SORTS  nat

OPS o  :          --> nat
    s  : nat      --> nat
    le : nat nat --> bool

CONSTRUCTORS o, s

EQUATIONS or(not(le(x,y)),or(not(le(y,z)),le(x,z))) = t

RULES le(o,x) --> t
      le(s(x),o) --> f
      le(s(x),s(y)) --> le(x,y)

ENDSPEC

Checking left hand sides of rules

All left hand sides of rules are correct !
==========================================

Checking termination of rules

System is terminating !
=======================
```

```
Checking confluence

System is confluent !
=====================

Checking totality of definitions

Checking totality of LE

1 definition for the function symbol LE generated !

All functions are totally defined !
==================================

          ************************
          *                      *
          *______________________*
          *                      *
          *  1 STOP              *
          *  2 NEXT              *
          *  3 CONTINUE          *
          *  4 HYPS              *
          *  5 ACCEPTED-HYPS     *
          *  6 REJECTED-HYPS     *
          *  7 RULES             *
          *  8 UNCOMP-RULES      *
          *  9 CHANGE-PARAMETER  *
          *                      *
          ************************

          Type any symbol --->  #3

          You have chosen CONTINUE

     RULE :
     ======
     (NATURAL 4): OR(NOT(LE(Y,X)),OR(NOT(LE(X,Z)),LE(Y,Z))) --> T

****************************************
*  inductively complete positions are  *
*______________________________________*
*                                      *
*  1 ((1 1))                           *
*  2 ((2 2))                           *
*  3 ((2 1 1))                         *
*                                      *
****************************************

Type any symbol --->  #1

; The current default concerning inductively complete positions is
; to choose only one. If there are more than one such positions the
; decision which to take is left to the user.

You have chosen ((1 1))

New rule generated :
(NATURAL 4): OR(NOT(LE(Y,X)),OR(NOT(LE(X,Z)),LE(Y,Z))) --> T

     RULE :
     ======
     NATURAL 4, NATURAL 3 ==> (NATURAL 5):
     OR(NOT(LE(Y,X)),OR(NOT(LE(S(X),Z)),LE(S(Y),Z))) --> T

************************************
```

```
*  inductively complete positions are  *
*                                      *
*--------------------------------------*
*                                      *
*  1 ((1 1))                           *
*  2 ((2 2))                           *
*  3 ((2 1 1))                         *
*                                      *
****************************************

Type any symbol --->  #2

You have chosen ((2 2))

New rule generated : NATURAL 4, NATURAL 3 ==> (NATURAL 5):
OR(NOT(LE(Y,X)),OR(NOT(LE(S(X),Z)),LE(S(Y),Z))) --> T

Hypothesis accepted : OR(NOT(LE(Y,X)),OR(NOT(LE(X,Z)),LE(Y,Z))) == T

; Again the proof was successful producing an additional lemma,
; namely (NATURAL 5).

          **********************
          *                    *
          *--------------------*
          *                    *
          *  1 STOP            *
          *  2 HYPS            *
          *  3 ACCEPTED-HYPS   *
          *  4 REJECTED-HYPS   *
          *  5 RULES           *
          *  6 UNCOMP-RULES    *
          *                    *
          **********************

          Type any symbol --->  #1

          You have chosen STOP

**********************************
*   Program parameter UNICOM :   *
**********************************

single-step : YES
generalize  : NO
protocol    : NO

**********************************
*       Program menue UNICOM     *
*--------------------------------*
*                                *
*   1 PARSE-SPECIFICATION        *
*   2 CHECK-SPECIFICATION        *
*   3 PROVE-SPECIFICATION        *
*   4 PARSE-CHECK-AND-PROVE-SPEC *
*   5 COMPILE-SPECIFICATION      *
*   6 RUN-SPECIFICATION          *
*   7 EDIT-SPECIFICATION         *
*   8 CHANGE-STATUS-SINGLE-STEP  *
*   9 CHANGE-STATUS-GENERALIZE   *
*  10 CHANGE-STATUS-PROTOCOL     *
*  11 EXIT                       *
*                                *
**********************************

; Let us now prove transitivity of "le" again, but by performing
; automatically inductive proofs in parallel according to all
```

```
; inductively complete positions in the left hand side of the
; conjecture.

Type any symbol --->  #3

You choose PROVE-SPECIFICATION

Type the filename ---> #natural

          ************************
          *                      *
          *                      *
          *______________________*
          *                      *
          *  1 STOP              *
          *  2 NEXT              *
          *  3 CONTINUE          *
          *  4 HYPS              *
          *  5 ACCEPTED-HYPS     *
          *  6 REJECTED-HYPS     *
          *  7 RULES             *
          *  8 UNCOMP-RULES      *
          *  9 CHANGE-PARAMETER  *
          *                      *
          ************************

          Type any symbol --->  #9

          You have chosen CHANGE-PARAMETER

**********************************************************************
*                Program parameter UNICOM are :                      *
**********************************************************************

compute inesscritical-pairs                    : NO
change inesscritical-pairs to esscritical-pairs : NO
choose only one inductively-complete-position   : YES
generalize                                      : NO
protocol                                        : NO

**********************************************************************
*                 change program parameter unicom :                  *
*                                                                    *
*____________________________________________________________________*
*                                                                    *
*  1 CHANGE-STATUS-COMPUTE-INESSCRITICAL-PAIRS                       *
*  2 CHANGE-STATUS-CHANGE-INESSCRITICAL-PAIRS-TO-ESSCRITICAL-PAIRS   *
*  3 CHANGE-STATUS-CHOOSE-ONLY-ONE-INDUCTIVELY-COMPLETE-POSITION     *
*  4 CHANGE-STATUS-GENERALIZE                                        *
*  5 EXIT                                                            *
*                                                                    *
**********************************************************************

Type any symbol --->  #3

You have chosen CHANGE-STATUS-CHOOSE-ONLY-ONE-INDUCTIVELY-COMPLETE-POSITIOI

**********************************************************************
*                Program parameter UNICOM are :                      *
**********************************************************************

compute inesscritical-pairs                    : NO
change inesscritical-pairs to esscritical-pairs : NO
choose only one inductively-complete-position   : NO
generalize                                      : NO
protocol                                        : NO

**********************************************************************
```

```
*                change program parameter unicom :                *
*                                                                 *
*-----------------------------------------------------------------*
*                                                                 *
*  1 CHANGE-STATUS-COMPUTE-INESSCRITICAL-PAIRS                    *
*  2 CHANGE-STATUS-CHANGE-INESSCRITICAL-PAIRS-TO-ESSCRITICAL-PAIRS *
*  3 CHANGE-STATUS-CHOOSE-ONLY-ONE-INDUCTIVELY-COMPLETE-POSITION  *
*  4 CHANGE-STATUS-GENERALIZE                                     *
*  5 EXIT                                                         *
*                                                                 *
*******************************************************************

Type any symbol --->  #5

You have chosen EXIT

          ************************
          *                      *
          *----------------------*
          *                      *
          *  1 STOP              *
          *  2 NEXT              *
          *  3 CONTINUE          *
          *  4 HYPS              *
          *  5 ACCEPTED-HYPS     *
          *  6 REJECTED-HYPS     *
          *  7 RULES             *
          *  8 UNCOMP-RULES      *
          *  9 CHANGE-PARAMETER  *
          *                      *
          ************************

          Type any symbol --->  #3

          You have chosen CONTINUE

New rule generated : NATURAL 4, NATURAL 1 ==>
(NATURAL 5): OR(NOT(LE(X,O)),LE(X,Y)) --> T

New rule generated : NATURAL 4, NATURAL 2 ==>
(NATURAL 6): OR(NOT(LE(S(X),Y)),NOT(LE(Y,O))) --> T

New rule generated : NATURAL 4, NATURAL 3 ==>
(NATURAL 7): OR(NOT(LE(Y,S(X))),OR(NOT(LE(X,Z)),LE(Y,S(Z)))) --> T

Hypothesis accepted : OR(NOT(LE(Y,X)),OR(NOT(LE(X,Z)),LE(Y,Z))) == T

; This time all three inductive proof attempts are run in parallel.
; As soon as the first of them has succeeded the whole process stops
; successfully. Equations (rules) deduced so far within the other proof
; environments are all inductively valid then and are collected as
; additional lemmas. Here the proof attempt corresponding to the
; inductively complete position (2 1 1) was the first to succeed.
;
; If the user wants to get more detailed information and better control
; over the proof process he (or she) can switch on the single step mode
; and e.g. inspect the status of every proof environment.

          ********************
          *                    *
          *--------------------*
          *                    *
          *  1 STOP            *
          *  2 HYPS            *
          *  3 ACCEPTED-HYPS   *
          *  4 REJECTED-HYPS   *
          *  5 RULES           *
```

```
          *  6 UNCOMP-RULES   *
          *                   *
          *********************

          Type any symbol --->  #1

          You have chosen STOP

**********************************
*   Program parameter UNICOM :   *
**********************************

single-step : YES
generalize  : NO
protocol    : NO

**********************************
*       Program menue UNICOM     *
*                                *
*                                *
*   1 PARSE-SPECIFICATION        *
*   2 CHECK-SPECIFICATION        *
*   3 PROVE-SPECIFICATION        *
*   4 PARSE-CHECK-AND-PROVE-SPEC *
*   5 COMPILE-SPECIFICATION      *
*   6 RUN-SPECIFICATION          *
*   7 EDIT-SPECIFICATION         *
*   8 CHANGE-STATUS-SINGLE-STEP  *
*   9 CHANGE-STATUS-GENERALIZE   *
*  10 CHANGE-STATUS-PROTOCOL     *
*  11 EXIT                       *
*                                *
**********************************

Type any symbol --->  #11

You choose EXIT

; Now the system returns to (COMMON) LISP top-level.
```