Theorem Proving in Hierarchical Clausal Specifications *

J. Avenhaus
K. Madlener
Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
67653 Kaiserslautern

Abstract

In this paper we are interested in an algebraic specification language that (1) allows for sufficient expessiveness, (2) admits a well-defined semantics, and (3) allows for formal proofs. To that end we study clausal specifications over built-in algebras. To keep things simple, we consider built-in algebras only that are given as the initial model of a Horn clause specification. On top of this Horn clause specification new operators are (partially) defined by positive/negative conditional equations. In the first part of the paper we define three types of semantics for such a hierarchical specification: model-theoretic, operational, and rewrite-based semantics. We show that all these semantics coincide, provided some restrictions are met. We associate a distinguished algebra $\mathcal{A}_{spec}$ to a hierachical specification *spec*. This algebra is initial in the class of all models of *spec*. In the second part of the paper we study how to prove a theorem (a clause) valid in the distinguished algebra $\mathcal{A}_{spec}$. We first present an abstract framework for inductive theorem provers. Then we instantiate this framework for proving inductive validity. Finally we give some examples to show how concrete proofs are carried out.

*This report was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D4-Projekt)

1 Introduction

1.1 Motivation

For the production of software it is desirable to precisely describe the products created so far on each level of the development. This will allow one to communicate on the products, to perform some reasoning, and finally to prove some correctness properties. In this paper we are interested in describing the product on the level of the functional design.

For this purpose, we are interested in a specification language that
- allows for sufficient expressiveness,
- admits a well-defined semantics,
- allows for formal reasoning.

We are interested in executable algebraic specifications, so we require the axioms to be (implicitely universally quantified) equations. Then the semantics should be based on equational reasoning on the ground terms (i.e. variable-free terms). This will result in some kind of initial semantics. Normally a conditional equation is of the form $B \Longrightarrow l = r$ where B is a conjunction of equations. Furthermore, the specification defines the operators totally. In order to comply more realistically with the needs of a specification language, we allow for several extensions:

(1) We allow the axioms to be positive/negative conditional equations, i.e., in $B \Longrightarrow l = r$ we allow also negative equations of the form $u \neq v$. From the logical point of view this is a clause. So we speak of clausal specifications.

(2) One would like to have fixed built-in structures such as the integers, the rationals, or lists. To keep things simple, in this paper we consider built-in algebras only which are themselves given by algebraic specifications.

(3) The specification of interest may define new operators on top of the built-in algebra only partially.

We study how to assign natural semantics to such a specification. We will define denotational semantics (based on the notion of models), operational semantics (based on equational reasoning on ground terms) and rewrite semantics (based on conditional rewriting). Provided some restrictions are met, all these semantics coincide. This defines the notion of validity of a clause. We discuss a proof method to prove (or disprove) a clause to be valid.

Our definition of the semantics is so that it complies with the following demands:

(a) If a ground equation $s = t$ is valid then this should be supported by E. (This ensures initiality.)

(b) If a ground equation $s = t$ is valid then it should remain valid if the specification is extended by adding new axioms. We call this "monotonic extendability of specifications".

We go a little bit more into the details in order to get a feeling what the paper is about: A specification *spec* consists of a signature *sig*, a base algebra $\mathcal{A}$ and a set E of positive/negative equations. The algebra $\mathcal{A}$ is given as the initial algebra of a specification $spec_0 = (sig_0, E_0)$, where sig_0 is a subsignature of sig and $E_0 \subseteq E$ consists of positive conditional equations. The axioms in $E_1 = E - E_0$ partially define the new operators

We associate to *spec* the quotient algebra $\mathcal{A}_{spec} = Term(F)/=_E$ of the free term algebra $Term(F)$ according to a suitably defined E-equality $=_E$. Note also that we have to define carefully how to apply a positive/negative conditional equation, in particular how to evaluate the negative conditions. Since we do not restrict to positive conditional equations, an initial model of *spec* in the sence of first order logic may not exist. Nevertheless, $\mathcal{A}_{spec}$ is defined such that it contains (an isomorphic copy of) $\mathcal{A}$ and is initial in a natural class of models of *spec*. Hence we define "A clause is valid in *spec* iff it is valid in $\mathcal{A}_{spec}$".

We now demonstrate by some examples which problems arise with the specifications we consider and how these problems are solved in our approach.

Example 1.1 (Partially defined operators) *Let $\mathcal{A}$ be the initial model of E_0 over $F_0 = \{0, s, +\}$:*

$$\begin{array}{lrcl} E_0: & x+0 & = & x, \\ & x+s(y) & = & s(x+y). \end{array}$$

Then $0 + x = x$ is valid in $\mathcal{A}$. Now we partially define the operator $-$ by E_1:

$$\begin{array}{lrcl} E_1: & x-0 & = & x, \\ & s(x)-s(y) & = & x-y. \end{array}$$

Let $E = E_0 \cup E_1$. Note that $s(0) - 0$ is a defined term, *i.e., it is E-equal to a base term $t \in Term(F_0)$. But $0 - s(0)$ is not defined, so it is a* junk term. *We want $0 + x = x$ to hold in the distinguished model $\mathcal{A}_{spec}$ specified by E. For that we have to say over which terms a variable x in an equation may range. If we allow x to range over all ground terms then we may also substitute the junk term $t \equiv 0 - s(0)$ for x, but $0 + t =_E t$ does not hold. As a consequence, the principle of monotonic extendability for a specification does not hold. Hence, we allow x to range over the defined terms only, then $0 + x =_E x$ holds. One may prove that $(x + y) - y =_E x$ also holds under this interpretation. This is true despite the fact that $-$ is only partially defined by E_1. We may extend the definition of $-$ in different ways. E.g., adding to E the axiom $0 - s(y) = 0$ defines $-$ to be "minus on $\mathbb{N}$", and adding the axiom $0 - s(y) = s(y)$ defines $-$ to be $x - y = \mid x - y \mid$ on $\mathbb{N}$. In both extensions E to E' we have $(x + y) - y =_{E'} x$. So the principle of monotonic extendability of specifications is fulfilled.*

Example 1.2 (Another semantics for *spec***)** *The last example suggests to define "A clause is valid in spec iff it holds in all total models of spec", see [KM86]. We show that this definition is unappropriate in our context.*

Let $\mathcal{A}$ be given by $F_0 = \{0, s, nil, .\}$ and $E_0 = \emptyset$. Let $E = E_0 \cup E_1$, where E_1 defines the operators push, pop and top on lists:

$$
\begin{array}{lrcl}
E_1: & push(x,l) & = & x.l \\
 & pop(x.l) & = & l \\
 & top(x.l) & = & x
\end{array}
$$

So top and pop are defined on non-empty lists only. According to our semantics the clause $G \equiv pop(l) = nil \Longrightarrow push(top(l), pop(l)) = l$ is valid. This is true, because the assumption $pop(l) = nil$ is satisfied only if l has the form $l \equiv x.l'$. This is in accordance with the "natural intention" on the specification. But, G is not valid according to the definition mentioned above: G is not valid in any model of spec that satisfies $pop(nil) = nil$ and $x.l \neq nil$.

Example 1.3 (Negative conditions in axioms) *Let $\mathcal{A}$ be given by $F_0 = \{0, s, true, false\}$ and $E_0 = \emptyset$. We define the operators even und odd by E_1:*

$$
\begin{array}{lrclcrcl}
E_1: & & & & & even(0) & = & true \\
 & even(x) & = & true & \Longrightarrow & even(s(s(x))) & = & true \\
 & even(x) & \neq & true & \Longrightarrow & even(s(s(x))) & = & false \\
 & & & & & odd(s(0)) & = & true \\
 & & & & & odd(s(s(x)) & = & odd(x)
\end{array}
$$

We use this example to show that the conditions $even(x) \neq true$ and $not(even(x) = true)$ are basically different. The condition $even(t) \neq true$ holds iff $even(t)$ can be evaluated to a term $t_0 \in Term(F_0)$ and $t_0 \neq_{E_0} true$, so $even(t)$ has to be a defined term. (See the next example for the motivation for this evaluation of negative conditions.) On the other hand, $not(even(x) = true)$ holds iff $even(t)$ cannot be evaluated to true. As an example, $even(s^3(0)) =_{E_1} false$ does not hold in our formalism, since the term $even(s(0))$ is not defined. But $even(s^3(0)) = false$ holds if $even(x) \neq true$ is replaced by $not(even(x) = true)$. (Clearly, $not(even(x) = true)$ is a syntactic construct not allowed in our formalism, but in other formalisms it is allowed to express the oposite of $even(x) = true$.)
This example also shows that negative conditions naturally arise in specifications. One may prove that $G \ \equiv \Longrightarrow even(x) = true \vee odd(x) = true$ is valid. This holds, though even and odd are only partially defined.

Example 1.4 (Problems with negative conditions) *Let $\mathcal{A}$ be given by $F_0 = \{0, s\}$ und $E_0 = \emptyset$. We define operators f and g by E_1:*

$$
\begin{array}{lrclcrcl}
E_1: & & & & & f(s(x), y) & = & f(x, s(s(y))) \\
 & f(x,y) & = & f(y,x) & \Longrightarrow & g(x,y) & = & 0 \\
 & f(x,y) & \neq & f(y,x) & \Longrightarrow & g(x,y) & = & s(0)
\end{array}
$$

Here the question is how to evaluate the conditions of a conditional equation. By standard practice we evaluate the conditions before applying the conclusion. Clearly, for any $x \equiv s^i(0)$ we have $g(x,x) =_E 0$. But what about $g(0, s^2(0))$? Obviously, we have no support from E to prove $f(0, s^2(0)) = f(s^2(0), 0)$, so it is tempting to conclude $g(0, s^2(0)) =_E s(0)$. But that would contradict the principle of monotonic extendability of specifications: If we extend E to E' by adding $f(0, y) = 0$ then we have $f(x, y) =_{E'} 0$ for all $x \equiv s^i(0)$, $y \equiv s^j(0)$, and hence $g(0, s^2(0)) =_{E'} 0$. To resolve this problem, we define the condition $t_1 \neq t_2$ to be E-satisfied, if there are base terms $t_1', t_2' \in \mathcal{T}(F_0)$

such that $t_i =_E t'_i$ *for* $i = 1,2$ *and not* $t'_1 =_{E_0} t'_2$. *In the example above, neither* $g(0, s^2(0) =_E 0$ *nor* $g(0, s^2(0)) =_E s(0)$ *holds according to this definition. The term* $g(0, s^2(0))$ *is a junk term.*

From an operational point of view, there are two reasons for a function f to be partial. Either the computation of $f(t)$ stops without producing a base term as output, or the computation does not stop. The first case splits into two subcases: (1) One wants to extend the specification later to define the new operators for more inputs. This was discussed in Example 1.1. (2) One does not want to define each new operator totally. This is examplified in Example 1.2. Here it does not make sense to define $top(nil)$. Also, it does not make sense to define division by zero. We now give an example for non-terminating computations.

Example 1.5 (Nonterminating computations) *Let* $\mathcal{A}$ *be given by* $F_0 = \{0, s, +\}$ *and*

$$\begin{array}{llcl} E_0: & x + 0 & = & x \\ & x + s(y) & = & s(x + y) \end{array}$$

We define the operators search and div by E_1*:*

$$\begin{array}{llclcl} E_1: & x = v & \Longrightarrow & search(x, y, u, v) & = & u \\ & x \neq v & \Longrightarrow & search(x, y, u, v) & = & search(x, y, s(u), v + y) \\ & y \neq 0 & \Longrightarrow & div(x, y) & = & search(x, y, 0, 0) \end{array}$$

It is easy to see that the rewrite system R *associated to* $E = E_0 \cup E_1$ *is ground confluent but not terminating (see below). We have* $div(s^i(0), s^j(0)) \xrightarrow{*}_R s^k(0)$ *iff* $i = j \cdot k$, $j > 0$. *If* $j > 0$ *and* j *is not a divisor of* i *then the computation of* $div(s^i(0), s^j(0))$ *does not stop. In this case* $div(s^i(0), s^j(0))$ *is a junk term.*

Up to now we have discussed on semantic issues. We now shortly comment on how to prove a clause to be an inductive theorem of *spec*, i.e., to be valid in $\mathcal{A}_{spec}$. Here equational reasoning is not enough, one needs some kind of induction. We use Noetherian induction, i.e., induction based on a well-founded ordering $\succ_i$ on the ground terms.

We will first design an abstract prover based on abstract notions of syntactic units and semantic units. Here we will work out the fundamental concepts of an inductive prover, what has to be proved for a concrete inductive prover to be correct and refutationally complete. This is an abstraction of the method "proof by consistency" developed in [Bac88]. This abstract prover can not only be instantiated to prove inductive theorems but for other inductive proofs also, e.g., to prove "The specification *spec* is sufficiently complete" or "The rewrite system R is ground confluent".

Next we instantiate the abstract prover for proving inductive theorems. We will not design a deterministic proof procedure. Instead we will design an inference system which can be turned into a deterministic prover by fixing a heuristic for applying the inference rules. So we remain on a more conceptual level and do not go down to the technical problems of a concrete prover for inductive theorems. Our inference rules, however, are in principle powerful enough to simulate classical inductive theorem proving based on concrete recursion schemes [BM79, Wal94], as well as methods based on "proof by consistency" [Bac88] or "cover sets" [ZKK88] or "test sets" [BR93].

The paper is organized as follows: We define the syntax of hierarchical specifications in section 2 and the semantics in section 3. In section 4 we study hierarchical term rewriting systems and so define the rewrite semantics of a specification. We discuss principal issuses of an inductive theorem prover in section 5 and present an instantiation of the abstract prover for proving inductive theorems (i.e. theorems in $\mathcal{A}_{spec}$) in section 6. In section 7 we demonstrate by examples how the inference system presented in section 6 works on concrete examples. Finally, in section 8 we discuss how to instantiate our prover to simulate inductive theorem provers known in the literature.

1.2 Related Work

We first discuss semantic issues. To our knowledge there are almost no papers on abstract data types defined by clausal specifications. We mention the book of Padawitz [Pad92], but here the emphasis is more on logic programming than on functional programming. There are many papers on Horn clause specification (i.e., only positive conditional equations are allowed). If in addition to this restriction no partiality is allowed then the initial model of the specification is considered to define the semantics of *spec* [Wec92].

If positive/negative conditional equations are allowed, then there are two possibilities to evaluate these conditions for applying the equation: In [BG94] negation by failure is used to evaluate the conditions. The semantics are fixed by a reduction ordering $\succ$, according to which the conditions of an equation have to be smaller than the conclusion. The semantics of *spec* are then fixed by the perfect model of *spec* according to $\succ$. In this approach no partiality is considered and the principle of monotonic extendability of specification does not hold. In [AB94] this approach is combined with the concepts of built-in algebras and of partiality. Here new problems arise, e.g., how to integrate the built-in algebra into the rewrite mechanism and how to prove confluence and termination.

In this paper we follow the concept of [WG93, WG94a]. Here the negative conditions are evaluated constructively in the base algebra $\mathcal{A}$ as indicated in Example 1.4.

In this approach the principle of monotonic extendability holds. In [WG94b], for each sort s there are "base variables" ranging over base terms of sort s only, and "general variables" ranging over all terms (including the junk terms) of sort s. This gives rise to several different notions of inductive validity and so to several semantics for *spec*. This paper clarifies some problems resulting from the fact that in the literature different notions of inductive validity are used without commenting on the differences. Here we use the type-E semantics of [WG94b], but restrict to base variables only.

The classical way to model partiality is to consider partial algebras, see e.g. [BWP84]. Here the problem arises how to define the equality appropriately. We consider two approaches and examplify them using Example 1.4: strong and existential equality. One defines on variable-free terms [BWP84] $t_1 =_s t_2$ (strong equality) iff either t_1 and

t_2 are both undefined or they are both defined and E-equal. In this approach all junk terms are considered equal. In Example 1.4 this would result in $g(x, y) = 0$ to hold for all $x \equiv s^i(0)$, $y \equiv s^j(0)$, $i \neq j$. But adding the axiom $f(0, y) = y$ would result in $g(0, s(0)) = 0$ not to hold. This contradicts the "natural intention" of *spec*. One defines $t_1 =_e t_2$ (existential equality) iff both terms are defined and they are E-equal. In this approach any two junk terms are considered unequal. In Example 1.4 we have that $f(s(0), 0) = f(0, s(s(0)))$ is not valid, though this is an instance of an axiom. We have $g(x, y) = s(0)$ is valid for all $x \equiv s^i(0)$, $y \equiv s^j(0)$, but adding the axiom $f(0, y) = 0$ results in $g(x, y) = 0$ being valid for all these x and y. Again this contradicts the "natural intention" of *spec*. In [BWP84] the conditional equation $s = t \Longrightarrow l = r$ is defined to be valid iff $s =_s t$ implies $l =_e r$. But this is problematic also.

In [KM86] an equation $t_1 = t_2$ is defined to be valid in *spec* iff $t_1 = t_2$ holds in all total F_0-generated models of *spec*. We have already discussed on that in Example 1.2. Here we demonstrate some other consequences of this definition. First, the set of valid equations $t_1 = t_2$ (t_1, t_2 being variable free) ist not a complete theory: We may have that neither $t_1 = t_2$ nor $t_1 \neq t_2$ is valid. This is demonstrated with Example 1.2 using $t_1 \equiv top(nil)$ and $t_2 \equiv 0$. Second, $t_1 = t_2$ may be valid without E-support of this fact. For example [KM86], let $\mathcal{A}$ be given by $F_0 = \{true, false\}$ and $E_0 = \emptyset$ and let *not* be partially defined by $E_1 = \{not(true) = false\}$. Then $not(not(false)) = false$ is valid, since this equation holds in all total F_0-generated models $\mathcal{B}$ of *spec*: We either have $not(false) = true$ or $not(false) = false$ in $\mathcal{B}$, in both cases $not(not(false)) = false$ holds in $\mathcal{B}$. But there is no $E_0 \cup E_1$-support to prove $not(not(false)) = false$ to be valid.

Now we comment on inductive theorem proving. The classical proof procedure is the Boyer-Moore prover [BM79]. Sophisticated proof engineering is incorporated into this prover to make it powerful in practice. In principle it extracts an induction scheme and a termination ordering from the specification and performs induction according to this induction scheme and ordering. Only specifications are allowed which induce a terminating algorithm for computing $f(t_1, \ldots, t_n)$ for any input $t_1, \ldots, t_n$. See [Wal94] for extending this approach, in particular for computing induction schemes. For positive conditional specifications methods using "cover sets" [ZKK88] and "test sets" [BR93] are developed which, combined with a reduction ordering, allow for inductive proofs. In many cases the test or cover sets can be computed from the specification. No partiality is allowed here. For unconditional specifications the method "proof by consistency" is known [HH82], [JK89], [KM87], [Bac88], [Red90]. Again, no partiality is allowed, the rewrite system induced by the specification has to be terminating. For unconditional partial specifications we refer to [KM86]. We have already commented on that.

This paper gives a unifying presentation on the work carried out in the project "Deduction in equational theories" supported in the Special Research Project (SFB 314) funded by the German Research Foundation (DFG). Most of the results presented here have been published in technical papers earlier. We comment on that where appropriate. The researchers contributing to these results include K. Becker, B. Gramlich and C.-P. Wirth.

2 Hierarchical Specifications

We are interested in specifications with a fixed built-in algebra $\mathcal{A}$. As mentioned earlier, here we restrict to the case where $\mathcal{A}$ is the initial algebra of a positive conditional base specification. The reader may consult [AB94] for the case where an arbitrary algebra is built in.

2.1 Notations

In this section we review some main notations from equational logic. The reader may consult [DJ90] or [Ave95] for more details.

A *signature* $sig = (S, F, V, \alpha)$ consists of a set S of *sorts*, a set F of *function symbols* (or *operators*), a set V of variables, and a function $\alpha : F \to S^+$ which fixes the input and output sorts for each $f \in F$. We write $f : s_1, \ldots, s_n \to s$ instead of $\alpha(f) = s_1 \cdots s_n s$.

The *variable system* V for sig is a system $V = (V_s)_{s\in S}$ such that $V_s \bigcap V_{s'} = \emptyset$ for $s \neq s'$. By abuse of notation we also write $V = \bigcup_{s\in S} V_s$. We denote by $Term_s(F, V)$ the set of *terms* of sort s constructed from F and V. Then $Term(F, V) = \bigcup_{s\in S} Term_s(F, V)$. $Term(F) = Term(F, \emptyset)$ is the set of *ground terms* (variable-free terms). We assume $Term_s(F) = Term_s(F, \emptyset)$ to be non-empty for each $s \in S$. We write $sort(t) = s$ if $t \in Term_s(F, V)$. We denote by $O(t)$ the set of positions of t, by t/p the subterm of t at position $p \in O(t)$, and by $t[u]_p$ the term resulting from t by replacing the subterm t/p with term u. We use $\equiv$ to denote the syntactic identity on terms.

Terms are used to build more complex syntactic units. An *equality atom* over sig has the form $u = v$ with $u, v \in Term_s(F, V)$ for some $s \in S$. A *definiteness atom* (a *def-atom*, for short) has the form $def(t)$ with $t \in Term(F, V)$. Here def is a meta-symbol, later interpreted as "defined". An *atom* is an equality atom or a def-atom. A *clause* has the form $\Gamma \Longrightarrow \Delta$ where Γ and Δ are multisets of atoms. We call Γ the antecedens and Δ the succedens of the clause. We will write $\Gamma, u = v$ instead of $\Gamma \cup \{u = v\}$ and $\Longrightarrow \Delta$ instead of $\emptyset \Longrightarrow \Delta$.

A *positive/negative conditional equation* (a *conditional equation*, for short) has the form

$$\Gamma; \Delta \Longrightarrow u = v.$$

Its clausal form is $\Gamma \Longrightarrow u = v, \Delta$. So a conditional equation results from a clause $\Gamma \Longrightarrow u = v, \Delta$ by singling out an equality axiom from the succedens. We call the elements from Γ the positive conditions and the elements from Δ the negative conditions of $\Gamma; \Delta \Longrightarrow u = v$. We speak of a positive conditional equation if $\Delta = \emptyset$. We then write $\Gamma \Longrightarrow u = v$. If in addition $\Gamma = \emptyset$ then we write $\Longrightarrow u = v$. This is an *unconditional equation*.

A signature $sig_0 = (S_0, F_0, V_0, \alpha_0)$ is a *subsignature* of $sig = (S, F, V, \alpha)$ if (i) $S_0 \subseteq S$, (ii) $F_0 \subseteq F$, (iii) $V_{0,s} = V_s$ for $s \in S_0$ and (iv) α_0 is the restriction $\alpha\,|_{F_0}$ of α to F_0. We then call sig an *enrichment* of sig_0.

After fixing some syntactical notions we now turn to some semantical notions. Let $sig = (S, F, V, \alpha)$ be given. A *sig-algebra* is a pair $\mathcal{A} = ((A_s)_{s \in S}, (f^{\mathcal{A}})_{f \in F})$ such that (i) A_s is a non-empty set (the carrier set for sort s), for all $s \in S$ and (ii) $f^{\mathcal{A}}$ is a function $f^{\mathcal{A}} : A_{s_1} \times \cdots \times A_{s_n} \to A_s$ if $f : s_1, \ldots, s_n \to s$, for all $f \in F$. We write $|\mathcal{A}|_s = A_s$ and $|\mathcal{A}| = A = \bigcup_{s \in S} A_s$.

Now asume that $sig_0 = (S_0, F_0, V_0, \alpha_0)$ is a subsignature of sig and $\mathcal{A}_0$ is a sig_0-algebra. We say $\mathcal{A}_0$ is a *subalgebra* of $\mathcal{A}$ if (i) $|\mathcal{A}_0|_s \subseteq |\mathcal{A}|_s$ for all $s \in S_0$ and $f^{\mathcal{A}_0}$ is the restriction of $f^{\mathcal{A}}$ to $|\mathcal{A}_0|$ for all $f \in F_0$. If $\mathcal{A}_0$ is generated by $Term(F_0)$ then we call $\mathcal{A}_0$ the *base-reduct* of $\mathcal{A}$.

2.2 Syntax of specifications

In this section we describe our specification mechanism. We will use hierarchical specifications: The base specification will describe the built-in algebra $\mathcal{A}$ and the full specification will describe an extended algebra $\mathcal{A}' = \mathcal{A}_{spec}$. The new operators f in $\mathcal{A}_{spec}$ will only be defined partially, i.e., only for some inputs $a_1, \ldots, a_n \in |\mathcal{A}|$ there is a value $f^{\mathcal{A}'}(a_1, \ldots, a_n)$ in $\mathcal{A}$.

There are two ways to model this in an algebraic setting: (1) One can use the concepts of order sorted specifications and order sorted algebras. This is done in [AB94]. (2) One can use the concept of constraints to restrict the ranges of the variables to elements of $\mathcal{A}$. Here we follow the second approach (as in [WG94a]) since it needs a simplier syntactic overhead.

We start with the usual notion of a specification. A *specification spec* $= (sig, E)$ consists of a signature $sig = (S, F, V, \alpha)$ and a set of conditional equations E. If E contains positive conditional equations only, then we speak of a *positive conditional specification* or a Horn clause specification. (In general, we sometimes speak of a positive/negative conditional specification in order to emphasize that positive/negative conditional equations are allowed.) If $spec_0 = (sig_0, E_0)$ is a specification and $sig_0 = (S_0, F_0, V_0, \alpha_0)$ is a subsignature of sig, then $spec_0$ is called as *subspecification* of *spec*. Now we turn to hierarchic specifications *spec*.

Definition 2.1 *Let spec* $= (sig, E)$ *be a specification and* $spec_0 = (sig_0, E_0)$ *a subspecification. Let* $sig = (S, F, V, \alpha)$, $sig_0 = (S_0, F_0, V_0, \alpha_0)$, $S_1 = S - S_0$, $F_1 = F - F_0$ *and* $E_1 = E - E_0$. *Let* E_0 *consist of positive conditional equations only which do not contain a* def-atom. *Let for each conditional equation* $\Gamma; \Delta \Longrightarrow s = t$ *in* E_1 *the conclusion* $s = t$ *contain an operator* $f \in F_1$ *and let* Δ *contain no* def-atom. *Then spec is called a* hierarchical specification *over the* base specification $spec_0$.

A term $t \in Term(F_0, V_0)$ is called a *base term*. A syntactic object (atom, clause, ...) is a base object if it contains base terms only. It is called a ground object if it contains ground terms only.

Now we have to model the restriction that base variables may range over base terms only. This is done by the definition of a substitution.

Definition 2.2 *Let spec be a hierarchical specification over the base specification* $spec_0$. *Let* V *be a variable system for spec. A* substitution *is a mapping* $\sigma : V \rightarrow Term(F, V)$ *such that (i)* $dom(\sigma) = \{x \mid \sigma(x) \not\equiv x\}$ *is finite, (ii)* $sort(x) = sort(\sigma(x))$ *for all* $x \in V$ *and (iii)* $\sigma(x)$ *is a base term if* x *is a base variable (i.e.* $sort(x) \in S_0$*). We extend* σ *as usual to* $\sigma : Term(F, V) \rightarrow Term(F, V)$ *by* $\sigma(f(t_1, \ldots, t_n)) \equiv f(\sigma(t_1), \ldots, \sigma(t_n))$. *We call* σ *a* ground substitution *if* $\sigma(x)$ *is a ground term for all* $x \in dom(\sigma)$. *We also write* $\sigma = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ *if* $dom(\sigma) = \{x_1, \ldots, x_n\}$ *and* $t_i \equiv \sigma(x_i)$.

3 Semantics of hierarchical specifications

In this chapter we are going to define the denotational and operational semantics of a hierarchical specification $spec = (sig, E)$. The denotational semantics is given by defining the models of $spec$. The operational semantics is given by defining the E-equality on $Term(F)$. We will define a special model $\mathcal{A}_{spec}$ of $spec$ and show that, under reasonable conditions, it is initial in the class of all models of $spec$.

3.1 Denotational semantics

Let $spec = (sig, E)$ be a hierarchical specification over $spec_0 = (sig_0, E_0)$ as in Definition 2.1. In order to define the models of $spec$ we proceed in two steps. First we associate to $spec_0$ the initial model $\mathcal{A}$. Then a sig-algebra $\mathcal{B}$ is a model of $spec$ if its base-reduct is isomorphic to $\mathcal{A}$ and it satisfies all conditional equations in E.

Given the positive conditional specification $spec_0 = (sig_0, E_0)$ with $sig_0 = (S_0, F_0, V_0, \alpha_0)$, to define the initial algebra $\mathcal{A}$ of $spec_0$ we proceed in the classical way: We define congruence relations[1] $(\sim_i)_{i\in\mathbb{N}}$ on $Term(F_0)$ in the following way. We have $s \sim_0 t$ iff $s \equiv t$. Given $\sim_i$, then $\sim_{i+1}$ is the smallest equivalence relation such that $s \sim_{i+1} t$ if (1) $s \sim_i t$ or (2) there is a conditional equation $\Gamma \Longrightarrow l = r$, a substitution σ and a position $p \in O(s)$ such that $s/p \equiv \sigma(l)$, $t \equiv s[\sigma(r)]_p$ and $\sigma(u) \sim_i \sigma(v)$ for all $u = v$ in Γ. Then $=_{E_0} = \bigcup_{i\in\mathbb{N}} \sim_i$ is the $E_0 - equality$.

Definition 3.1 *Let $spec_0 = (sig_0, E_0)$ be a positive conditional specification. For $t \in Term(F_0)$ let $[t]$ denote the $=_{E_0}$-equivalence class of t. Then the* initial model *$\mathcal{A}$ of $spec_0$ is defined as follows*

$A_s = \{[t] \mid t \in Term_s(F_0)\}$
$f^{\mathcal{A}}([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$

It is well known (and easy to prove) that the functions $f^{\mathcal{A}}$ are well defined (i.e., $[t_i] = [s_i]$ implies $f^{\mathcal{A}}([t_1], \ldots, [t_n]) = f^{\mathcal{A}}([s_1], \ldots, [s_n])$) and that $\mathcal{A}$ is a model of $spec_0$ in the sense of first order logic. We associate to $spec_0$ its initial model $\mathcal{A}$.

Before defining the models of $spec$ we need to clarify the notion of "a sig-algebra $\mathcal{B}$ satisfies a conditional equation $\Gamma; \Delta \Longrightarrow l = r$".

Definition 3.2 *Let $spec = (sig, E)$ be a hierarchical specification over $spec_0 = (sig_0, E_0)$ as above. Let $\mathcal{B} = ((\mathcal{B}_s)_{s\in S}, (f^{\mathcal{B}})_{f\in F})$ be a sig-algebra and $\mathcal{B}^0$ the base-reduct of $\mathcal{B}$. Let $B = \mid\mathcal{B}\mid$ and $B^0 = \mid\mathcal{B}^0\mid$.*

a) *A function $\varphi : V \to B$ is an* evaluation function *if $\varphi(x) \in B_s$ if $sort(x) = s$ and $\varphi(x) \in B^0_s$ if $sort(x) \in S_0$. Then φ is extended to $\varphi : Term(F, V) \to B$ by $\varphi(f(t_1, \ldots, t_n)) = f^{\mathcal{B}}(\varphi(t_1), \ldots, \varphi(t_n))$.*

[1] A *congruence relation* on $Term(F)$ is an equivalence relation $\sim$ such that $s_i \sim t_i$ implies $f(s_1, \ldots, s_n) \sim f(t_1, \ldots, t_n)$ for all $f \in F$.

b) $(\mathcal{B}, \varphi)$ satisfies an equality axiom $u = v$ *iff* $\varphi(u) = \varphi(v)$ *in* $\mathcal{B}$ *and it* satisfies a def-atom $def(t)$ *iff* $\varphi(t) \in \mathcal{B}^0$. *Finally,* $(\mathcal{B}, \varphi)$ satisfies a clause $\Gamma \Longrightarrow \Delta$ *if it satisfies an atom in* Δ *whenever it satisfies all atoms in* Γ.

c) $\mathcal{B}$ satisfies *a clause* $\Gamma \Longrightarrow \Delta$ *(or* $\Gamma \Longrightarrow \Delta$ *is* valid *in* $\mathcal{B}$*) if, for each evaluation function* φ, $(\mathcal{B}, \varphi)$ *satisfies* $\Gamma \Longrightarrow \Delta$.

We write $\mathcal{B} \models \Gamma \Longrightarrow \Delta$ *if* $\mathcal{B}$ *satisfies* $\Gamma \Longrightarrow \Delta$. *We write* $\mathcal{B} \models A$ *(for an atom* A*) if* $\mathcal{B}$ *satisfies* $\Longrightarrow A$. *In this case we say that* $\mathcal{B}$ *satisfies* A.

With these definitions, $\mathcal{B}$ satisfies a conditional equation $\Gamma; \Delta \Longrightarrow s = t$ iff $\mathcal{B}$ satisfies the clause $\Gamma \Longrightarrow s = t, \Delta$. Now we can define the models of *spec*.

Definition 3.3 *Let* $spec = (sig, E)$ *be a hierarchical specification over* $spec_0 = (sig_0, E_0)$. *A sig-algebra* $\mathcal{B}$ *is a* model *of spec iff*
(1) *The base-reduct* $\mathcal{B}^0$ *of* $\mathcal{B}$ *is isomorphic to the initial algebra* $\mathcal{A}$ *of* $spec_0$.

(2) $\mathcal{B}$ *satisfies every conditional equation in* E.

Example 3.4 *We demonstrate these definitions using the specification from Example 1.1. We have* $sig_0 = (S_0, F_0, V_0, \alpha_0)$ *with* $S_0 = \{NAT\}$, $F_0 = \{0, s, +\}$ *and*

$$E_0\colon \quad x + 0 = x \qquad x + s(y) = s(x + y)$$

We also have $spec = (sig, E)$ *with* $sig = (S_0, F, V, \alpha)$, $F = F_0 \cup \{-\}$ *and* $E = E_0 \cup E_1$

$$E_1\colon \quad x - 0 = x \qquad s(x) - s(y) = x - y$$

Now, $\mathcal{A} = (A_{NAT}, \{0^{\mathcal{A}}, s^{\mathcal{A}}, +^{\mathcal{A}}\})$ *is given by*

$$\begin{array}{lcl} A_{NAT} & = & \{[s^i(0)] \mid i \in \mathbb{N}\} \\ 0^{\mathcal{A}} & = & [0] \qquad\qquad [s^i(0)] +^{\mathcal{A}} [s^j(0)] = [s^{i+j}(0)] \\ s^{\mathcal{A}}([s^i(0)]) & = & [s^{i+1}(0)] \end{array}$$

To define a model $\mathcal{B}$ *of spec we use the rewrite system*

$$\begin{array}{lllll} R\colon & x + 0 \to x & \qquad & x + s(y) & \to s(x + y) \\ & x - 0 \to x & & s(x) - s(y) & \to x - y \end{array}$$

It is confluent and terminating. So any $t \in Term(F)$ *has a unique* R*-normal form* $\overline{t}$. *We define* $\mathcal{B} = (B_{NAT}, \{0^{\mathcal{B}}, s^{\mathcal{B}}, +^{\mathcal{B}}, -^{\mathcal{B}}\})$ *by*

$$\begin{array}{lcl} B_{NAT} & = & \{\overline{t} \mid t \in Term(F)\} \\ 0^{\mathcal{B}} & = & 0 \qquad\qquad \overline{t_1} +^{\mathcal{B}} \overline{t_2} = \overline{t_1 + t_2} \\ s^{\mathcal{B}}(\overline{t}) & = & s(\overline{t}) \qquad\quad \overline{t_1} -^{\mathcal{B}} \overline{t_2} = \overline{t_1 - t_2} \end{array}$$

In order to prove that $\mathcal{B}$ *is indeed a model of spec one has to prove (i) that* $\mathcal{B}$ *satisfies each conditional equation in* E *and (ii) that* $\mathcal{A}$ *is isomorphic to the base-reduct* $\mathcal{B}^0$ *of* $\mathcal{B}$. *That is easily done. We sketch the proof of (ii): We have* $\mathcal{B}^0 = (\mathcal{B}^0_{NAT}, \{0^{\mathcal{B}^0}, s^{\mathcal{B}^0}, +^{\mathcal{B}^0}, -^{\mathcal{B}^0}\})$ *with*

$$\begin{array}{lcl} \mathcal{B}^0_{NAT} & = & \{s^i(0) \mid i \in \mathbb{N}\} \\ 0^{\mathcal{B}^0} & = & 0 \qquad\qquad s^i(0) +^{\mathcal{B}^0} s^j(0) = s^{i+j}(0) \\ s^{\mathcal{B}^0}(s^i(0)) & = & s^{i+1}(0) \end{array}$$

So $\psi : \mathcal{A} \to \mathcal{B}^0$, $\psi([s^i(0)]) = s^i(0)$ *is a* sig_0*-isomorphism.*

This example may demonstrate how we model partiality: We do not use partial algebras, both algebras $\mathcal{B}$ and $\mathcal{B}^0$ are total algebras (in the sense that all functions are totally defined.) But $-^{\mathcal{B}}$ on $\mathcal{B}$ induces a partial function $-^{\mathcal{B}^0}$ on $\mathcal{B}^0$: $s^i(0) -^{\mathcal{B}^0} s^j(0)$ is defined iff $s^i(0) -^{\mathcal{B}} s^j(0) \in \mathcal{B}^0_{NAT}$. This holds true iff $i \geq j$. If $s^i(0) -^{\mathcal{B}^0} s^j(0)$ is defined, then $s^i(0) -^{\mathcal{B}^0} s^j(0) = s^i(0) -^{\mathcal{B}} s^j(0)$).

Definition 3.5 *Let spec be a hierarchical specification over* sig_0*. Then* $Mod(spec)$ *is the class of all models of spec.*

Note that $Mod(spec)$ may be empty. This may happen because the equations in $E - E_0$ may produce "confusion" on $\mathcal{A}$. For example, if one adds the equation $0 - 0 = s(0)$ to E_1 in Example 3.4 then $0 = s(0)$ holds in any algebra $\mathcal{B}$ satisfying all conditional equations in E. So the base-reduct of $\mathcal{B}$ cannot be isomorphic to $\mathcal{A}$.

3.2 Operational semantics

We now define the E-equality $=_E$ for a hierarchical specification $spec = (sig, E)$. If E only contains positive conditional equations, then this can be done in the classical way (as in the definition of the initial model in section 3.1). The problem is how to evaluate the negative conditions for applying a positive/negative conditional equation. As mentioned earlier, we here choose to evaluate the negation constructively in the built-in algebra $\mathcal{A}$: A ground inequation $u \neq v$ is evaluated to true iff u and v evaluate to sig_0-ground terms u_0, v_0 such that $u_0 \neq v_0$ in $\mathcal{A}$. This approach is taken from [WG93], [WG94b]. For the approach *negation by failure* see [BG94] and [AB94].

Let $spec = (sig, E)$ be a hierarchical specification over $spec_0 = (sig_0, E_0)$ with $sig = (S, F, V, \alpha)$ and $sig_0 = (S_0, F_0, V_0, \alpha_0)$. In order to define the E-equality $=_E$ we first define an approximation $(\sim_i)_{i \in \mathbb{N}}$ on $Term(F)$ as we have done for positive conditional specifications in section 3.1. For that we need some additional notations.

Let $\sim$ be a congruence relation on $Term(F)$. We say
$\sim$ satisfies $u = v$ if $u \sim v$
$\sim$ satisfies $def(t)$ if $t \sim t_0$ for some $t_0 \in Term(F_0)$
$\sim$ satisfies $u \neq v$ if $u \sim u_0 \neq_{E_0} v_0 \sim v$ for some $u_0, v_0 \in Term(F_0)$
$\sim$ satisfies $\Gamma; \Delta$ if $\sim$ satisfies all $u = v$, $def(t)$ and $u' \neq v'$ such that $u = v$, $def(t) \in \Gamma$ and $u' = v' \in \Delta$.

Definition 3.6 *Let* $spec = (sig, E)$ *be a hierarchical specification over* $spec_0 = (sig_0, E_0)$*. Let* $(\sim_i)_{i \in \mathbb{N}}$ *be defined on* $Term(F)$ *by*

$\sim_0$	*is* $=_{E_0}$
$\sim_{i+1}$	*is the smallest congruence relation such that* $s \sim_{i+1} t$ *if (1)* $s \sim_i t$ *or (2) there is a conditional equation* $\Gamma; \Delta \Longrightarrow l = r$ *in* E*, a ground substitution* σ *and a position* $p \in O(s)$ *such that* $s/p \equiv \sigma(l)$*,* $t \equiv s[\sigma(r)]_p$ *and* $\sim_i$ *satisfies* $\sigma(\Gamma); \sigma(\Delta)$*.*

Then $=_E = \cup_{i\in\mathbb{N}} \sim_i$ *is the* E-equality defined by *spec.*

Remember that for any ground substitution σ we have $\sigma(x) \in Term(F_0)$ if $sort(x) \in S_0$ (by Definition 2.2). This realizes the restriction that base variables can be instantiated by base terms only. The next Lemma shows that the approximation $(\sim_i)_{i\in\mathbb{N}}$ of $=_E$ is monotonous. This is very similar to the case where only positive conditional equations are allowed in a specification.

Lemma 3.7
If $\sim_i$ *satisfies* $u = v$ *then* $\sim_{i+1}$ *satisfies* $u = v$.
If $\sim_i$ *satisfies* $def(t)$ *then* $\sim_{i+1}$ *satisfies* $def(t)$.
If $\sim_i$ *satisfies* $u \neq v$ *then* $\sim_{i+1}$ *satisfies* $u \neq v$.

Proof: This is easily proved by an induction on i. □

Now we are ready to associate a distinguished algebra $\mathcal{A}_{spec}$ to a hierarchical specification *spec*.

Definition 3.8 *Let* $spec = (sig, E)$ *be a hierarchical specification over* $spec_0 = (sig_0, E_0)$. *For* $t \in Term(F)$ *let* $[t]$ *denote the E-equivalence class of* t. *Then* $\mathcal{A}_{spec}$ *is defined by*

$$A_{spec,s} = \{[t] \mid t \in Term_s(F)\}$$
$$f^{\mathcal{A}_{spec}}([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$$

We call $\mathcal{A}_{spec}$ the *algebra specified* by *spec*.

In general, $\mathcal{A}_{spec}$ is not a model of *spec*. There are two reasons for that: (1) The base-reduct $\mathcal{A}^0_{spec}$ of $\mathcal{A}_{spec}$ may not be isomorphic to the initial algebra $\mathcal{A}$ of $spec_0$. (2) We evaluate negative conditions constructively, but that may not be reflected by the specification. We give some examples.

For (1): Consider $spec = (sig, E)$ with $S_0 = S = \{ANY\}$, $F_0 = \{a, b\}$, $F = \{a, b, c\}$, $E_0 = \emptyset$ and $E = \{\Longrightarrow c = a, \Longrightarrow c = b\}$. Then $a = b$ is valid in $\mathcal{A}_{spec}$, but not in $\mathcal{A}$. Hence, $\mathcal{A}_{spec} \notin Mod(spec)$.

For (2): Consider $spec = (sig, E)$ with $S_0 = S = \{ANY\}$, $F_0 = \{a\}$, $F = \{a, b, c\}$, $E_0 = \emptyset$ and $E_1 = \{a \neq b \Longrightarrow a = c\}$. Then $a = c$ or $a = b$ is valid in any model $\mathcal{B}$ of *spec* (since $\Longrightarrow a = b \vee a = c$ is the clausal form of $a \neq b \Longrightarrow a = c$). But neither $a = b$ nor $a = c$ is valid in $\mathcal{A}_{spec}$. So $\mathcal{A}_{spec} \notin Mod(spec)$.

We want to consider only those specifications *spec* such that $\mathcal{A}_{spec} \in Mod(spec)$. This is captured by the next definitions.

Definition 3.9 *Let* $spec = (sig, E)$ *be a hierarchical specification over* $spec_0 = (sig_0, E_0)$.
a) spec is a consistent extension *of* $spec_0$ *if for all base terms* $s, t \in Term(F_0)$ *we have* $s =_E t$ *iff* $s =_{E_0} t$.
b) A conditional equation $\Gamma; \Delta \Longrightarrow l = r$ *is* def-moderated *if* $def(u), def(v) \in \Gamma$ *for each equality atom* $u = v$ *in* Δ. *spec is* def-moderated *if each conditional equation in* E *is* def*-moderated.*

Note that in the preceeding examples the first specification *spec* is not a consistent extension of $spec_0$ and the second specification is not def-moderated. In the second example, if we replace E_1 by $E_1 = \{def(a), def(b), a \neq b \Longrightarrow a = c\}$, then spec is def-moderated, $\mathcal{A}_{spec}$ consists of three elements a, b and c, and we have $\mathcal{A}_{spec} \in Mod(spec)$. In the following we will consider only those hierarchical specifications that are consistent and def-moderated. Then $\mathcal{A}_{spec}$ will be a model of *spec*, as the next Theorem says.

Definition 3.10 *The hierarchical specification spec over* $spec_0$ *is* admissible *if (1) spec is a consistent extension of* $spec_0$ *and (2) spec is* def*-moderated.*

Theorem 3.11 *Let* $spec = (sig, E)$ *be an admissible hierarchical specification over* $spec_0 = (sig_0, E_0)$*. Then* $\mathcal{A}_{spec}$ *is a model of spec.*

Proof: a) We have $s =_{E_0} t$ iff $s =_E t$ for all $s, t \in Term(F_0)$, since *spec* is a consistent extension of *spec*. So $\mathcal{A}$ is the base-reduct of $\mathcal{A}_{spec}$.
b) We have to show that $\mathcal{A}_{spec}$ satisfies $\Gamma \Longrightarrow l = r, \Delta$ whenever $\Gamma; \Delta \Longrightarrow l = r$ is in E. So, for every evaluation function $\varphi : V \rightarrow \mathcal{A}_{spec}$ we have to show that $(\mathcal{A}_{spec}, \varphi)$ satisfies $\Gamma \Longrightarrow l = r, \Delta$. Let $[t]$ be the $=_E$-equivalence class of $t \in Term(F)$. For each $x \in V_s$ there is a $t_x \in Term(F)$ such that $\varphi(x) = [t_x]$. Then $\sigma(x) = t_x$ is a substitution (since $t_x \in Term(F_0)$ for $sort(x) \in S_0$). One easily proves $\varphi(t) = [\sigma(t)]$ for all $t \in Term(F)$.

We have to prove: If $(\mathcal{A}_{spec}, \varphi)$ satisfies all atoms in Γ but no atom in Δ, then $(\mathcal{A}_{spec}, \varphi)$ satisfies $l = r$. Since $\varphi(t) = [\sigma(t)]$ for all t this is equivalent to: If $\mathcal{A}_{spec}$ satisfies all atoms in $\sigma(\Gamma)$ and no atom in $\sigma(\Delta)$, then $\mathcal{A}_{spec}$ satisfies $\sigma(l) = \sigma(r)$. We have $def(u), def(v) \in \Gamma$ for all $u = v$ in Δ since *spec* is def-moderated. So we have to prove: If $=_E$ satisfies all atoms in $\sigma(\Gamma); \sigma(\Delta)$ then $=_E$ satisfies $\sigma(l) = \sigma(r)$. We have that $=_E$ satisfies a ground atom A iff $\sim_i$ satisfies A for some i. So, if $=_E$ satisfies $\sigma(\Gamma); \sigma(\Delta)$ then some $\sim_i$ satisfies $\sigma(\Gamma); \sigma(\Delta)$. But then $\sim_{i+1}$ satisfies $\sigma(l) = \sigma(r)$. Hence $=_E$ satisfies $\sigma(l) = \sigma(r)$. □

Corollary 3.12 *For any clause* G *we have:* $\mathcal{A}_{spec} \models G$ *iff* $\mathcal{A}_{spec} \models \tau(G)$ *for all ground substitutions* τ*.*

Proof: As in the last proof, any evaluation function $\varphi : V \rightarrow \mathcal{A}_{spec}$ defines a ground substitution τ such that $\varphi(t) = [\tau(t)]$ for all $t \in Term(F)$. Conversely, if τ is a ground substitution then it defines an evaluation function φ by $\varphi(x) = [\tau(x)]$. Now the claim directly follows from the definition of $\mathcal{A}_{spec} \models G$ (see Definition 3.2). □

Now we prove that the principle of monotonic extendability of specifications holds: Atoms valid in E remain valid if E is enlarged to E'.

Theorem 3.13 *Let* $spec = (sig, E)$ *and* $spec' = (sig, E')$ *be two admissible hierarchical specifications over* $spec_0 = (sig_0, E_0)$ *such that* $E \subseteq E'$*. If* s, t *are sig-terms and* $\mathcal{A}_{spec} \models s = t$*, then* $\mathcal{A}_{spec'} \models s = t$*. If* $\mathcal{A}_{spec} \models def(t)$*, then* $\mathcal{A}_{spec'} \models def(t)$*.*

Proof: Let $(\sim_i)_{i\in\mathbb{N}}$ and $(\sim'_i)_{i\in\mathbb{N}}$ be the approximations of $=_E$ and $=_{spec'}$, respectively. An induction on i proves: If A is a sig-atom and $\sim_i$ satisfies A then $\sim'_i$ satisfies A. So, for any ground substitution τ: If $\tau(s) =_E \tau(t)$ then $\tau(s) =_{E'} \tau(t)$. Now the claim follows from Corollary 3.12. $\square$

By Theorem 3.13 an atom remains valid when the specification $spec$ is extended to $spec'$. Clearly, a negated atom valid in $spec$ need not be valid in $spec'$. For example, if t is a ground term and the clause $def(t) \Longrightarrow$ is valid in $\mathcal{A}_{spec}$, then for no $t_0 \in Term(F_0)$ we have $t =_E t_0$. But $t =_{E'} t_0$ may hold for some $t_0 \in Term(F_0)$, i.e., the clause $def(t) \Longrightarrow$ may not be valid in $\mathcal{A}_{spec'}$. We next study which clauses remain valid in $\mathcal{A}_{spec'}$.

Theorem 3.14 *Let spec and spec$'$ be as in Theorem 3.13. Let $\Gamma \Longrightarrow \Delta$ be valid in $\mathcal{A}_{spec}$. If the clauses $\Longrightarrow def(t)$ for $t \in \{u, v \mid u = v \text{ in } \Gamma\} \cup \{u \mid def(u) \text{ in } \Gamma\}$ are valid in $\mathcal{A}_{spec}$, then $\Gamma \Longrightarrow \Delta$ is valid in $\mathcal{A}_{spec'}$, too.*

Proof: Let σ be any ground substitution. We have to prove that $\sigma(\Gamma) \Longrightarrow \sigma(\Delta)$ is valid in $\mathcal{A}_{spec'}$. By the assumptions and Theorem 3.13, if for some atom $A \in \Gamma$ $\sigma(A)$ is not valid in $\mathcal{A}_{spec}$ then it is not valid in $\mathcal{A}_{spec'}$. If for some $u = v \in \Delta$, $\sigma(u) = \sigma(v)$ is valid in $\mathcal{A}_{spec}$, then it is valid in $\mathcal{A}_{spec'}$. Hence, since $\sigma(\Gamma) \Longrightarrow \sigma(\Delta)$ is valid in $\mathcal{A}_{spec}$, it is valid in $\mathcal{A}_{spec'}$. $\square$

3.3 Initiality

In this section we show that $\mathcal{A}_{spec}$ is initial in the class $Mod(spec)$ of all models of $spec$. As a consequence, we have $t_1 = t_2$ is valid in $\mathcal{A}_{spec}$ iff it is valid in all term-generated models of $spec$. This justifies that we distinguish $\mathcal{A}_{spec}$ as the "algebra specified by $spec$".

We need some standard notations to work this out. Let $\mathcal{A} = ((A_s)_{s\in S}, (f^{\mathcal{A}})_{f\in F})$ and $\mathcal{B} = ((B)_{s\in S}, (f^{\mathcal{B}})_{f\in F})$ be two sig-algebras, where sig is $sig = (S, F, V, \alpha)$. A *sig-homomorphism* from $\mathcal{A}$ into $\mathcal{B}$ is a mapping $\psi : \mid\mathcal{A}\mid \to \mid\mathcal{B}\mid$ such that $\psi : A_s \to B_s$ and $\psi(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(\psi(a_1), \ldots, \psi(a_n))$ for all $a_i \in\mid\mathcal{A}\mid$ and $f \in F$.

Definition 3.15 *Let $\mathcal{A}$ be a sig-algebra and $\mathcal{K}$ a class of sig-algebras. $\mathcal{A}$ is called* initial *in $\mathcal{K}$ if (i) $\mathcal{A} \in \mathcal{K}$ and (ii) for each $\mathcal{B} \in \mathcal{K}$ there is exactly one sig-homomorphism $\psi : \mathcal{A} \to \mathcal{B}$.*

Now let $spec = (sig, E)$ be an admissible hierarchical specification over $spec_0 = (sig_0, E_0)$. It is well known from equational logic that the initial algebra $\mathcal{A}$ of $spec_0$ is indeed initial in the class of all models of $spec_0$. We prove that $\mathcal{A}_{spec}$ is initial in $Mod(spec)$.

Let $\mathcal{B}$ be a sig-algebra. We define $t^{\mathcal{B}}$ for $t \in Term(F)$ by

$$\begin{array}{lcll} t^{\mathcal{B}} & = & c^{\mathcal{B}} & \text{if} \quad t \equiv c,\ \alpha(c) = s \\ t^{\mathcal{B}} & = & f^{\mathcal{B}}(t_1^{\mathcal{B}}, \ldots, t_n^{\mathcal{B}}) & \text{if} \quad t \equiv f(t_1, \ldots, t_n) \end{array}$$

Lemma 3.16 *Let spec be an admissible hierarchical specification over $spec_0$. Let $\mathcal{B} \in Mod(spec)$. For all $u, v \in Term(F)$ we have $\mathcal{B} \models u = v$ if $u =_E v$.*

Proof: Let $\mathcal{B}^0$ be the base-reduct of $\mathcal{B}$ and let $(\sim_i)_{i \in \mathbb{N}}$ be the approximation of $=_E$. Induction on i gives for all $u, v \in Term(F)$:
If $\sim_i$ satisfies $u = v$ then $\mathcal{B} \models u = v$.
If $\sim_i$ satisfies $def(u)$ then $u^{\mathcal{B}} \in \mathcal{B}^0$.
If $\sim_i$ satisfies $u \neq v$ then $u^{\mathcal{B}}$, $v^{\mathcal{B}} \in \mathcal{B}^0$ and $u^{\mathcal{B}} \neq v^{\mathcal{B}}$.
The claim of the Lemma then follows immediately. □

Theorem 3.17 *Let $spec = (sig, E)$ be an admissible hierarchical specification over $spec_0 = (sig_0, E_0)$. Then $\mathcal{A}_{spec}$ is initial in $Mod(spec)$.*

Proof: $\mathcal{A}_{spec} \in Mod(spec)$ by Theorem 3.11. So we have to prove that for each $\mathcal{B} \in Mod(spec)$ there is exactly one sig-homomorphism $\psi : \mathcal{A}_{spec} \to \mathcal{B}$.
Define $\psi : \mathcal{A}_{spec} \to \mathcal{B}$ by

$$\psi([t]) = t^{\mathcal{B}}$$

Then ψ is well-defined, because $[t] = [t_1]$ implies $t =_E t_1$ and hence $t^{\mathcal{B}} = t_1^{\mathcal{B}}$ by Lemma 3.16. We prove that ψ is a sig-homomorphism: Clearly, $\psi([t]) \in B_s$ if $[t] \in A_s$. We have $\psi(f^{\mathcal{A}_{spec}}([t_1], \ldots, [t_n]) = \psi([f(t_1, \ldots, t_n)]) = f(t_1, \ldots, t_n)^{\mathcal{B}} = f^{\mathcal{B}}(t_1^{\mathcal{B}}, \ldots, t_n^{\mathcal{B}}) = f^{\mathcal{B}}(\psi([t_1]), \ldots, \psi([t_n]))$. Hence, ψ is a sig-homorphism. Assume $\psi' : \mathcal{A}_{spec} \to \mathcal{B}$ is any sig-homomorphism. One easily proves by induction on the term structure of t that $\psi([t]) = \psi'([t])$. So ψ is the only sig-homomorphism from $\mathcal{A}_{spec}$ into $\mathcal{B}$. □

Now we relate validity in $\mathcal{A}_{spec}$ to validity in all term-generated models of *spec*.

Definition 3.18 *Let $sig = (S, F, V, \alpha)$. A sig-algebra $\mathcal{B}$ is* term-generated *if for all $b \in \mathcal{B}$ there is a $t \in Term(F)$ such that $b = t^{\mathcal{B}}$.*

Theorem 3.19 *Let $spec = (sig, E)$ be an admissible hierarchical specification over $spec_0 = (sig_0, E_0)$. Let $u, v \in Term(F, V)$. Then $\mathcal{A}_{spec} \models u = v$ iff $\mathcal{B} \models u = v$ for all term-generated $\mathcal{B} \in Mod(spec)$.*

Proof: a) If $\mathcal{B} \models u = v$ for all term-generated $\mathcal{B} \in Mod(spec)$ then $\mathcal{A}_{spec} \models u = v$, since $\mathcal{A}_{spec}$ is a term-generated model of *spec*.
b) Assume $\mathcal{A}_{spec} \models u = v$ and $\mathcal{B} \in Mod(spec)$ is term-generated. Let $V' = Var(u = v) = \{x_1, \ldots, x_n\}$ be the set of variables in u and v. Let $\varphi : V' \to \mathcal{B}$ be an evaluation function. We have to show $(\mathcal{B}, \varphi)$ satisfies $u = v$, i.e., $\varphi(u) = \varphi(v)$ in $\mathcal{B}$. Let $[t]$ be the $=_E$-equivalence class of $t \in Term(F)$.

Since $\mathcal{B}$ is term-generated, there are $t_i \in Term(F)$ with $\varphi(x_i) = t_i^{\mathcal{B}}$. Let σ be the substitution with $\sigma(x_i) = t_i$. Since $\mathcal{A}_{spec}$ is initial in $Mod(spec)$, the function $\psi : \mathcal{A}_{spec} \to \mathcal{B}$, $\psi([t]) = t^{\mathcal{B}}$ is a sig-homomorphism. By induction on the term structure one shows $\varphi(t) = \psi([\sigma(t)])$ for all $t \in Term(F, V')$. Since $\mathcal{A}_{spec} \models u = v$ we have $[\sigma(u)] = [\sigma(v)]$. This gives $\varphi(u) = \psi([\sigma(u)]) = \psi([\sigma(v)]) = \varphi(v)$ in $\mathcal{B}$. $\square$

These results may justify why we call $\mathcal{A}_{spec}$ "the algebra specified by *spec*": We are interested in clauses that are valid in $\mathcal{A}_{spec}$.

Definition 3.20 *Let spec be an admissible hierarchic specification over* $spec_0$. *A clause* $\Gamma \Longrightarrow \Delta$ *is called an* inductive theorem *of spec if* $\mathcal{A}_{spec} \models \Gamma \Longrightarrow \Delta$. *Let* $\mathcal{I}Th(spec)$ *denote the set of inductive theorems of spec.*

4 Rewrite semantics

Given the results obtained so far, there are two problems left: (1) We are interested in executable specifications, so we want to effectively compute in $\mathcal{A}_{spec}$. (2) In order to apply the results of section 3.2 and 3.3, we have to prove that a given hierarchical specification *spec* is admissible. Since it is easy to check whether *spec* is *def-moderated*, the central problem is to prove that *spec* is a consistent extension of $spec_0$. In this chapter we are going to introduce hierarchical term rewrite systems. This will help us to solve these problems. Positive/negative conditional rewriting was introduced by Kaplan [Kap88].

Let $sig_0 = (S_0, F_0, V_0, \alpha_0)$ be a subsignature of $sig = (S, F, V, \alpha)$ as in section 2.1. A *positive/negative conditional rewrite rule* is an oriented positive/negative conditional equation, so it has the form $\Gamma; \Delta \Longrightarrow l \to r$ where Δ contains no *def-atom*. We require $\mathcal{V}ar(\Gamma) \cup \mathcal{V}ar(\Delta) \cup \mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$.

Definition 4.1 *A* positive/negative conditional rewrite system *R is the union of two rewrite systems R_0 and R_1.*

R_0 contains positive conditional rewrite rules $\Gamma \Longrightarrow l \to r$ over sig_0 only. No def-atoms *appear in R_0. R_1 contains positive/negative conditional rewrite rules $\Gamma; \Delta \Longrightarrow l \to r$ over sig such that l contains an operator $f \in F - F_0$ and Δ contains no* def-atom.

We are going to define the rewrite relations $\longrightarrow_{R_0}$ and $\longrightarrow_R$. This is similar to the definition of $=_{E_0}$ and $=_E$ in section 2.2. In particular, negative conditions $u = v$ in a rewrite rule are constructively evaluated over sig_0-term. Positive conditions are evaluated by joinability (see below).

We need some notations. Let $\longrightarrow$ be a binary relation on $Term(F)$. Then $\stackrel{+}{\longrightarrow}$ (and $\stackrel{*}{\longrightarrow}$ and $\stackrel{*}{\longleftrightarrow}$) is the transitive (transitive-reflexive and transitive-reflexive-symmetric, respectively) closure of $\longrightarrow$. Define the relations $\stackrel{\leq 1}{\longrightarrow}$, $\downarrow$ and $\downarrow_{\leq 1}$ by $\stackrel{\leq 1}{\longrightarrow} = \longrightarrow \cup \equiv$ and $\downarrow = \stackrel{*}{\longrightarrow} \circ \stackrel{*}{\longleftarrow}$ and $\downarrow_{\leq 1} = (\stackrel{\leq 1}{\longrightarrow} \circ \stackrel{*}{\longleftarrow}) \cap (\stackrel{*}{\longrightarrow} \circ \stackrel{\leq 1}{\longleftarrow})$. Terms s and t are called (strongly) *joinable* if $s \downarrow t$ (resp. $s \downarrow_{\leq 1} t$) holds. $\longrightarrow$ is called *confluent* if $\stackrel{*}{\longleftrightarrow} \subseteq \downarrow$ holds. $\longrightarrow$ is called *terminating* if there is no infinite sequence $(t_i)_{i \in \mathbb{N}}$ of terms such that $t_i \to t_{i+1}$ for all i.

We say

- $\longrightarrow$ satisfies $u = v$, if $u \downarrow v$ holds
- $\longrightarrow$ satisfies $def(t)$, if $t \stackrel{*}{\longrightarrow} t_0$ for some $t_0 \in Term(F_0)$
- $\longrightarrow$ satisfies $u \neq v$, if $u \stackrel{*}{\longrightarrow} u_0$, $v \stackrel{*}{\longrightarrow} v_0$ and not $u_0 \downarrow v_0$ for some $u_0, v_0 \in Term(F_0)$.
- $\longrightarrow$ satisfies $\Gamma; \Delta$, if $\longrightarrow$ satisfies all $u = v$, $def(t)$ and $u' \neq v'$ for all $u = v$, $def(t) \in \Gamma$ and $u = v \in \Delta$.

For denoting specifications we identify a conditional rewrite system with a set of conditional equations. This is done in the next definition. In the same spirit, $=_R$ is defined according to Definition 3.6.

Definition 4.2 *Let spec* $= (sig, R)$ *be a hierarchical specification over* $spec_0 = (sig_0, R_0)$.
a) We define an approximation $(\longrightarrow_i^0)_{i\in\mathbb{N}}$ *of* $\longrightarrow_{R_0}$ *on* $Term(F)$ *by:* $\longrightarrow_0^0 = \equiv$ *is the identity relation and*

$s \longrightarrow_{i+1}^0 t$ *if* $s \longrightarrow_i^0 t$ *or for some rule* $\Gamma \Longrightarrow l \to r$ *in* R_0, $p \in O(s)$ *and substitution* σ *we have* $s/p \equiv \sigma(l)$, $t \equiv s[\sigma(r)]_p$ *and* $\longrightarrow_i^0$ *satisfies* $\sigma(\Gamma)$.

Then $\longrightarrow_{R_0} = \bigcup_{i\in\mathbb{N}} \longrightarrow_i^0$.
b) We define an approximation $(\longrightarrow_i)_{i\in\mathbb{N}}$ *of* $\longrightarrow_R$ *on* $Term(F)$ *by* $\longrightarrow_0 = \longrightarrow_{R_0}$ *and*

$s \longrightarrow_{i+1} t$ *if* $s \longrightarrow_i t$ *or for some rule* $\Gamma;\Delta \Longrightarrow l \to r$ *in* R, $p \in O(s)$ *and substitution* σ *we have* $s/p \equiv \sigma(l)$, $t \equiv s[\sigma(r)]_p$ *and* $\longrightarrow_i$ *satisfies* $\sigma(\Gamma); \sigma(\Delta)$.

Then $\longrightarrow_R = \bigcup_{i\in\mathbb{N}} \longrightarrow_i$.

We say R is confluent (resp. terminating) if $\longrightarrow_R$ is. Note that $\longrightarrow_{R_0} \subseteq \longrightarrow_i$ for all i.

It is well known that $=_{R_1} = \stackrel{*}{\longleftrightarrow}_{R_1} = \downarrow_{R_1}$ for any positive conditional non-hierarchical rewrite system R_1 [Kap84]. This can be translated to our setting.

Theorem 4.3 *Let spec* $= (sig, R)$ *be a hierarchical specification over* $spec_0$. *If* R *is confluent then* $=_R = \stackrel{*}{\longleftrightarrow}_R = \downarrow_R$.

Proof: We only sketch the proof since it is tedious. Note that $=_{R_0} = \downarrow_{R_0}$ on $Term(F_0)$ since R is confluent and so is R_0. Let $(\sim_i)_{i\in\mathbb{N}}$ be the approximation of $=_R$. First one proves by induction on i that $\longrightarrow_i \subseteq \sim_i$. This gives $\stackrel{*}{\longleftrightarrow}_R \subseteq =_R$. Next, using the assumption that R is confluent, one proves by induction on i that $\sim_i \subseteq \downarrow_R$. This gives $=_R \subseteq \downarrow_R$. Since $\downarrow_R \subseteq \stackrel{*}{\longleftrightarrow}_R$, we have $=_R = \downarrow_R = \stackrel{*}{\longleftrightarrow}_R$. □

Now we use the general assumption on hierarchical rewrite systems that, for a rule $\Gamma;\Delta \Longrightarrow l \to r$ in $R_1 = R - R_0$, the term l contains an operator $f \in F - F_0$. So, no term $u \in Term(F_0)$ is reducible by a rule in R_1. Furthermore, if $u \stackrel{*}{\longrightarrow}_{R_0} u'$ then $u' \in Term(F_0)$ also. So $u \downarrow_R v$ implies $u \downarrow_{R_0} v$ for $u, v \in Term(F_0)$.

Theorem 4.4 *Let spec* $= (sig, R)$ *be a hierarchical specification over* $spec_0 = (sig_0, R_0)$. *If* R *is confluent then spec is a consistent extension of spec.*

Proof: We have to prove: For any $u, v \in Term(F_0)$ we have $u =_R v$ iff $u =_{R_0} v$. We have $u =_R v$ iff $u \downarrow_R v$ (by Theorem 4.3) iff $u \downarrow_{R_0} v$ (by the preceeding remark) if $u =_{R_0} v$ (by Theorem 4.3). □

As a consequence of this Theorem we get a sufficient condition that the results of section 3.2 and 3.3 are applicable.

Corollary 4.5 *Let spec* $= (sig, R)$ *be a hierarchical specification over spec* $= (sig_0, R_0)$. *Let* R *be confluent and* def-moderated. *Then spec is admissable.* □

Notice that these results require no termination assumption on R.

It is easy to check whether R is *def-moderated*. So it remains to develop conditions under which R is confluent. This is done by using the notion of critical pairs.

Definition 4.6

a) *Let R be a hierarchical rewrite system. Let $\Gamma_i; \Delta_i \Longrightarrow l_i \rightarrow r_i$, $i = 1, 2$ be (variable-disjoint) rules in Γ. Let $p \in O(l_1)$, $l_1/p \notin V$ and let $\sigma = mgu(l_1/p, l_2)$. Then $\sigma(\Gamma_1 \cup \Gamma_2); \sigma(\Delta_1 \cup \Delta_2) \Longrightarrow \sigma(r_1) = \sigma(l_1[r_2]_p)$ is a* critical pair *for R. Let $CP(R)$ be the set of all critical pairs.*

b) *A critical pair $\Gamma; \Delta \Longrightarrow u = v$ is (*strongly*)* joinable *if for any ground substitution σ such that $\longrightarrow_R$ satisfies $\sigma(\Gamma); \sigma(\Delta)$ we have $\sigma(u) \downarrow_R \sigma(v)$ (resp. $\sigma(u) \downarrow_{R,\leq 1} \sigma(v)$).*

We first study sufficient conditions for R to be confluent without the assumption that R is terminating. For this we use the fact that $\sigma(x) \in Term(F_0)$ for all $x \in V$, $sort(x) \in S_0$ and all substitutions σ. So $\sigma(x)$ is not reducible by any rule $\Gamma; \Delta \Longrightarrow l \rightarrow r$ in $R - R_0$. This gives rise to sophisticated methods to prove confluence of R (see [Wir95]). Here we just mention a very simple result concerning "free constructors".

Theorem 4.7 *Let $spec = (sig, R)$ be a hierarchical specification over $spec_0 = (sig_0, \emptyset)$ such that $S_0 = S$. If all critical pairs in $CP(R)$ are strongly joinable then R is confluent.*

Proof: We prove that $\longrightarrow_R$ is strongly confluent, i.e. ${}_R\longleftarrow \circ \longrightarrow_R \subseteq \downarrow_{R,\leq 1}$. Then $\longrightarrow_R$ is confluent by a result of Huet [Hue80].

We proceede as usual: Assume $t \rightarrow t_i$, $i = 1, 2$, using $\Gamma_i; \Delta_i \Longrightarrow l_i \rightarrow r_i$, $p_i \in O(t)$, σ_i. If t/p_1 and t/p_2 are disjoint subterms of t then clearly we have $t_1 \downarrow_{R,\leq 1} t_2$. So assume $p_2 = p_1 p$, i.e. t/p_2 is a subterm of t/p_1. If $p \in O(l_1)$, $l_1/p \notin V$, then this is covered by a critical pair and hence we have $t_1 \downarrow_{R,\leq 1} t_2$. Since $\sigma_2(l_2)$ cannot be a subterm of $\sigma_1(x)$, $x \in \mathcal{V}ar(l_1)$, there are no other possibilities than the two mentioned above. □

Now we assume that R is terminating. Again, we give only a simple result. It is based on the notion that R is decreasing: A partial ordering $\succ$ on $Term(F, V)$ is a *reduction ordering* if (i) it is well-founded (no infinite sequence $(t_i)_{i \in \mathbb{N}}$ exists with $t_i \succ t_{i+1}$), (ii) it is *stable* under substitutions (i.e. $t_1 \succ t_2$ implies $\sigma(t_1) \succ \sigma(t_2)$) and (iii) it is *monotonic* with respect to the term structure (i.e. $t_1 \succ t_2$ implies $t[t_1]_p \succ t[t_2]_p$). Let $\rhd$ be the subterm ordering. If $\succ$ is a sort preserving reduction ordering then $\succ_{st} = (\succ \ \cup \rhd)^+$ is well-founded and stable under substitutions.

Definition 4.8 *A hierarchical rewrite system R is* decreasing with respect to the recution ordering $\succ$ *if for each rule $\Gamma; \Delta \Longrightarrow l \rightarrow r$ we have*

$$\longrightarrow_R \subseteq \succ \text{ and}$$
$$\sigma(l) \succ_{st} \sigma(s) \text{ for all } s \in \{u, v \mid u = v \in \Gamma \cup \Delta\} \cup \{t \mid def(t) \in \Gamma\}$$
$$\text{and all substitutions } \sigma.$$

R is decreasing *if it is decreasing with respect to some reduction ordering.*

For example, if $\succ$ is a reduction ordering and

$$l \succ r \text{ and } l \succ_{st} s \text{ for all } s \in \{u, v \mid u = v \in \Gamma \cup \Delta\} \cup \{t \mid def(t) \in \Gamma\}$$

holds, then R is decreasing. If R is finite and decreasing then $\longrightarrow_R$ is effectively computable. We also have

Theorem 4.9 *Let R be a decreasing hierarchical rewrite system. If all critical pairs are joinable then R is confluent.*

Proof: The proof is similar to that for positive conditional rewrite systems. One proves by Noetherian induction on $\succ_{st}$ for all t: If $t_1 \; {}_R\!\overset{*}{\longleftarrow} \; t \; \overset{*}{\longrightarrow}_R t_2$ then $t_1 \downarrow_R t_2$. $\square$

We conclude this chapter by giving some examples. All the examples discussed in section 1.1, except Example 1.5 are easily proved to be confluent by using Theorem 4.9. We consider Example 1.3. There is only one critical pair

$$\{even(x) = true\}; \{even(x) = true\} \Longrightarrow true = false$$

(This is our formal way of writing $even(x) = true; even(x) \neq true \Longrightarrow true = false$.) For any substitution σ we have $\sigma(x) \equiv s^i(0)$. One easily proves by induction on i that $even(s^{2i}(0)) \longrightarrow_R true$ and $even(s^{2i+1}(0))$ is irreducible. So there is no σ such that $\longrightarrow_R$ satisfies $\sigma(\Gamma); \sigma(\Delta)$ with $\Gamma = \Delta = \{even(x) = true\}$. Hence all critical pairs are joinable. Since R is decreasing, R is confluent.

Let us go into some more detail here. For non-hierarchical decreasing rewrite systems the following is known [AL94]: To prove confluence of R, those critical pairs $\Gamma \Longrightarrow u = v$ need not be considered where Γ contains $s = t_1$ and $s = t_2$ and $t_1 \not\equiv t_2$ and $\sigma(t_1)$, $\sigma(t_2)$ are irreducible for all ground substitutions σ. (This holds, for example, it t_1 and t_2 are irreducible ground terms.) Clearly, this holds for hierarchical rewrite systems also. In [Wir95] it is proved that critical pairs $\Gamma; \Delta \Longrightarrow u = v$ often need not be considered if $\Gamma \cap \Delta \neq \emptyset$. E.g., the following system is confluent by Theorem 65 of [Wir95].

We now turn to Example 1.5. Let $F_0 = \{0, s\}$, $F = F_0 \cup \{+, search, div\}$, $R_0 = \emptyset$ and

$$
\begin{array}{lrl}
R: & & x + 0 \to x \\
 & & x + s(y) \to s(x + y) \\
y \neq 0 & \Longrightarrow & div(x, y) \to search(x, y, 0, 0\} \\
x \neq v & \Longrightarrow & search(x, y, u, v) \to search(x, y, s(u), v + y) \\
x = v & \Longrightarrow & search(x, y, u, v) \to u
\end{array}
$$

Here R is not terminating. One easily proves by induction on the term structure of t: It $t_1 \; {}_R\!\longleftarrow \; t \; \longrightarrow_R t_2$ then $t_1 \; \overset{\leq 1}{\longrightarrow}_R \; \circ \; {}_R\!\overset{\leq 1}{\longleftarrow} \; t_2$. (Notice that $\sigma(z) \equiv s^i(0)$ for any ground substitution σ and any $z \in dom(\sigma)$.) So $\longrightarrow_R$ is strongly confluent and hence confluent.

5 An abstract inductive prover

5.1 Motivation

In this chapter we present an abstract framework for a prover based on induction according to a Noetherian ordering (the induction ordering). The reason for this abstract setting is to single out the basic concepts of inductive provers and to look for conditions that guarantee correctness and refutational competeness of such a prover. The abstract prover is based on an abstract (unspecified) inference system. We formulate our conditions just mentioned by specifying properties of the inference system and by specifying minimal control on how to apply the inference rules.

There are several degrees of freedom to instantiate the abstract prover. First, it can be instantiated for different proof tasks. In this paper we will instantiate it to prove inductive theorems, i.e., to prove clauses valid in $\mathcal{A}_{spec}$ for a given specification *spec*. Other proof tasks may include to prove that a conditional rewrite system is ground confluent or that an operator $f \in F$ is totally defined by *spec*. Second, given a proof task, various well-founded orderings may be used as induction ordering. Third, there are various ways to design the inference rules and to use the induction hypothesis for the induction step. We will not comment on the third point in detail.

Our proof is an abstraction of Bachmair's method "proof by consistency" [Bac88], so it covers this method. Our prover can (in principle) also be instantiated to the cover-set method [ZKK88], the test-set method [BR93] and induction based on induction schemes [BM79], [Wal94]. But we do not discuss implementation and proof engineering aspects.

The approach presented here is similar to those in [Bec94] and [WB94]. There one can find more technical details and some refinements.

5.2 The abstract prover

We now present our abstract prover. We will try to give the intended intuition behind the abstract concepts by relating them to the concrete concepts for inductive theorem proving. We refer to that by the phrase "in our case".

We are working on *syntactic units* $G, H, \ldots$. In our case that are clauses of the form $\Phi \implies \Psi$. We denote by $\mathcal{G}$ and $\mathcal{H}$ sets of syntactic units. A syntactic unit G may describe (in general infinitely) many *semantic units* (G, τ). In our case, τ is any ground substitution. (G, τ) is called a $\mathcal{G}$*-instance* if $G \in \mathcal{G}$. Furthermore, we need a predicate P defined on the semantic units, the "property to be proved". In our case $P(G, \tau)$ holds iff $\tau(G)$ is valid in $\mathcal{A}_{spec}$. We write $P(G)$ if $P(G, \tau)$ holds for all τ, and we write $P(\mathcal{G})$ if $P(G)$ holds for all $G \in \mathcal{G}$.

For disproving $P(G)$ we need a failure predicate $Fail$. In our case, $Fail(G)$ holds if G is "obviously" not valid in $\mathcal{A}_{spec}$. We require that $Fail$ is *compatible* with P: If $Fail(G)$ holds then $P(G)$ does not hold. We write $Fail(\mathcal{G})$ if $Fail(G)$ for some $G \in \mathcal{G}$. So, if $Fail(\mathcal{G})$ holds then $P(\mathcal{G})$ does not hold.

Our inference system $\mathcal{I}$ operates on *states* of the form $(\mathcal{H};\mathcal{G})$. Here $\mathcal{G}$ is called the set of unprocessed goals and $\mathcal{H}$ is the set of processed goals.

We write

$$(\mathcal{H};\mathcal{G}) \vdash_{\mathcal{I}} (\mathcal{H}';\mathcal{G}')$$

if $(\mathcal{H}';\mathcal{G}')$ can be reached from $(\mathcal{H};\mathcal{G})$ by applying one inference step. To be more precise, we assume that $\mathcal{I}$ operates on state $(\mathcal{H};\mathcal{G})$ by processing a selected goal $G \in \mathcal{G}$. If $\mathcal{G}_0 = \mathcal{G} \cup \{G\}$ then $\mathcal{I}$ processes G by producing a set $\mathcal{G}'$ of new goals and shifting G into the set of processed goals. ($\mathcal{G}'$ may be empty.) We denote this by

$$(\mathcal{H};\mathcal{G}_0, G) \vdash_{\mathcal{I}} (\mathcal{H}, G;\mathcal{G}_0, \mathcal{G}')$$

We write $\vdash$ instead of $\vdash_{\mathcal{I}}$ if $\mathcal{I}$ is known from the context. An $\mathcal{I}$-derivation is a sequence $(\mathcal{H}_i;\mathcal{G}_i)_{i\geq 0}$ such that $(\mathcal{H}_i;\mathcal{G}_i) \vdash (\mathcal{H}_{i+1};\mathcal{G}_{i+1})$ for all i. We then write $(\mathcal{H}_0;\mathcal{G}_0) \overset{n}{\vdash} (\mathcal{H}_n, \mathcal{G}_n)$. An $\mathcal{I}$-derivation may be finite or infinite. $\overset{*}{\vdash}$ is the reflexive-transitive closure of $\vdash$.

We want $\mathcal{I}$ to have the following properties

(I) If $(\emptyset;\mathcal{G}) \overset{*}{\vdash} (\mathcal{H}';\emptyset)$ then $P(\mathcal{G})$ holds

(II) If $(\emptyset;\mathcal{G}) \overset{*}{\vdash} (\mathcal{H}';\mathcal{G}')$ and $Fail(\mathcal{G}')$ holds then $P(\mathcal{G})$ does not hold.

(III) If $P(\mathcal{G})$ does not hold then there is a state $(\mathcal{H}';\mathcal{G}')$ such that $(\emptyset;\mathcal{G}) \overset{*}{\vdash} (\mathcal{H}';\mathcal{G}')$ and $Fail(\mathcal{G}')$ holds.

Hence, by (I) one can prove $P(\mathcal{G})$ and by (II) one can disprove $P(\mathcal{G})$. We call $\mathcal{I}$ *inductively correct* if (I) holds and *refutationally correct* if (II) holds. We call $\mathcal{I}$ *refutationally complete* if (III) holds.

An *induction ordering* $\succeq_i$ is a quasi-ordering on the semantic units, the strict part $\succ_i$ of which is well-founded. Let $\succeq_i$ be a fixed induction ordering in the following. A *counter-example* is a semantic unit (G,τ) such that $P(G,\tau)$ does not hold. If $G \in \mathcal{G}$ then (G,τ) is a $\mathcal{G}$-counter-example. The following notion of an inductive state is crucial for our generic inductive prover.

Definition 5.1 *Let $(\mathcal{H};\mathcal{G})$ be a state.*
a) $P(\mathcal{H}_0)$ holds below (G,τ) *if $P(H,\pi)$ holds for all $\mathcal{H}_0$-instances (H,π) such that $(G,\tau) \succ_i (H,\pi)$.*
b) $(\mathcal{H};\mathcal{G})$ is inductive *if the following holds for each $\mathcal{G}$-instance (G,τ): If $P(\mathcal{H})$ holds below (G,τ) then $P(G,\tau)$ holds.*
c) The $\mathcal{G}$-instance (G,τ) is an inductive counter-example *for $(\mathcal{H};\mathcal{G})$ if $P(\mathcal{H} \cup \mathcal{G})$ holds below (G,τ) and $P(G,\tau)$ does not hold.*

The intention behind this definition is as follows. Since $\succeq_i$ is well-founded, any set of semantic units contains $\succeq_i$-minimal elements. The $\mathcal{G}$-instance (G,τ) is an inductive counter-example for $(\mathcal{H};\mathcal{G})$ iff it is $\succeq_i$-minimal in the set of all $\mathcal{H} \cup \mathcal{G}$-counter examples. The state $(\mathcal{H};\mathcal{G})$ is inductive iff there is no $\mathcal{G}$-counter example (G,τ) which is $\succeq_i$-minimal in the set of all $\mathcal{H} \cup \mathcal{G}$-counter examples. Obviously $P(\mathcal{G})$ holds iff $(\emptyset;\mathcal{G})$ is inductive. We want to prove that $(\emptyset, \mathcal{G})$ is inductive by computing a state transformation $(\emptyset; \mathcal{G}) = (\mathcal{H}_0; \mathcal{G}_0) \vdash (\mathcal{H}_1; \mathcal{G}_1) \vdash \cdots \vdash (\mathcal{H}_n; \mathcal{G}_n)$ until $\mathcal{G}_n = \emptyset$. Obviously $(\mathcal{H}_n; \emptyset)$ is inductive. So, if we assure that $(\mathcal{H}_i; \mathcal{G}_i)$ is inductive whenever $(\mathcal{H}_{i+1}; \mathcal{G}_{i+1})$ is inductive, then $(\emptyset; \mathcal{G})$ is inductive and $P(\mathcal{G})$ is proved. We make these considerations precise.

Definition 5.2 *$\mathcal{I}$ is* inductively sound *(with respect to P and $\succeq_i$) if $(\mathcal{H}; \mathcal{G})$ is inductive whenever $(\mathcal{H}; \mathcal{G}) \vdash (\mathcal{H}'; \mathcal{G}')$ and $(\mathcal{H}'; \mathcal{G}')$ is inductive.*

Theorem 5.3 *Let $\mathcal{I}$ be inductively sound and let $(\mathcal{H}; \mathcal{G}) \overset{n}{\vdash} (\mathcal{H}'; \emptyset)$. Then $(\mathcal{H}; \mathcal{G})$ is inductive. If $\mathcal{H} = \emptyset$ then $P(\mathcal{G})$ holds. Hence $\mathcal{I}$ is inductively correct.* □

We now come to property (II).

Definition 5.4 *$\mathcal{I}$ is* refutationally sound *(with respect to P) if $P(\mathcal{H}' \cup \mathcal{G}')$ holds whenever $(\mathcal{H}; \mathcal{G}) \vdash (\mathcal{H}'; \mathcal{G}')$ and $P(\mathcal{H} \cup \mathcal{G})$ holds.*

Intuitively speaking, $\mathcal{I}$ is refutationally sound if no counter-example is introduced by performing an inference step. The next theorem is easily proved by induction on n.

Theorem 5.5 *Let $\mathcal{I}$ be refutationally sound and $(\emptyset, \mathcal{G}) \overset{n}{\vdash} (\mathcal{H}'; \mathcal{G}')$. If $Fail(\mathcal{G}')$ holds then $P(\mathcal{G})$ does not hold.* □

It remains to give a sufficient condition for property (III) to hold. We need two conditions: (1) $\mathcal{I}$ decreases any inductive counter-example until is becomes "obvious". (2) $\mathcal{I}$ is powerful enough according to $Fail$: If $Fail(\mathcal{G})$ does not hold then an inference step is applicable to $(\mathcal{H}; \mathcal{G})$. We make this precise.

Definition 5.6 $\mathcal{I}$ decreases (strictly decreases) inductive counter-examples *(with respect to P and $\succeq_i$) if the following holds: If $(\mathcal{H}; \mathcal{G}) \vdash (\mathcal{H}'; \mathcal{G}')$ and (G, τ) is an inductive counter-example for $(\mathcal{H}; \mathcal{G})$, then either (G, τ) is also an inductive counter-example for $(\mathcal{H}'; \mathcal{G}')$ or there is an inductive $\mathcal{G}'$-counter-example (G', τ') for $(\mathcal{H}'; \mathcal{G}')$ such that $(G, \tau) \succeq_i (G', \tau')$ (resp. $(G, \tau) \succ_i (G', \tau')$).*

Corollary 5.7 *If $\mathcal{I}$ decreases inductive counter-examples then $\mathcal{I}$ is inductively sound.* □

Definition 5.8 *$\mathcal{I}$ is* Fail-complete *if for any state $(\mathcal{H}; \mathcal{G})$ such that $\mathcal{G} \neq \emptyset$ and $Fail(\mathcal{G})$ does not hold there is a state $(\mathcal{H}'; \mathcal{G}')$ such that $(\mathcal{H}; \mathcal{G}) \vdash (\mathcal{H}'; \mathcal{G}')$.*

We will prove that $\mathcal{I}$ is refutationally complete if it is $Fail$-complete and it strictly decreases inductive counter-examples. For that we consider fair $\mathcal{I}$-derivations.

Definition 5.9 *An $\mathcal{I}$-derivation $(\mathcal{H}_i; \mathcal{G}_i)_{i\in\mathbb{N}}$ is* fair *if (1) or (2) holds:*
(1) The derivation is finite and ends in $(\mathcal{H}_n; \mathcal{G}_n)$ such that $Fail(\mathcal{G}_n)$ or $\mathcal{G}_n = \emptyset$ holds.
(2) The derivation is infinite and the set of persistent goals $\bigcup_{i\geq 0} \bigcap_{j\geq i} \mathcal{G}_j$ is empty.

Obviously, if $\mathcal{I}$ is $Fail$-complete then any finite derivation can be extended to a fair derivation.

Theorem 5.10 *Assume that $\mathcal{I}$ strictly decreases inductive counter-examples. For any fair $\mathcal{I}$-derivation $(\emptyset; \mathcal{G}_0) \vdash (\mathcal{H}_1; \mathcal{G}_1) \vdash ...$ such that $P(\mathcal{G}_0)$ does not hold there is an $n \in \mathbb{N}$ such that $Fail(\mathcal{G}_n)$ holds.*

Proof: There is an inductive counter-example (G_0, τ_0) for $(\mathcal{H}_0; \mathcal{G}_0)$ since $P(\mathcal{G}_0)$ does not hold. Since $\mathcal{I}$ decreases inductive counter-examples, there is a sequence $(G_j; \tau_j)_{j\in\mathbb{N}}$ such that $(G_j, \tau_j) \succeq_i (G_{j+1}, \tau_{j+1})$ and (G_{j+1}, τ_{j+1}) is an inductive counter-example for $(\mathcal{H}_{j+1}; \mathcal{G}_{j+1})$. Since $\mathcal{I}$ strictly decreases counter-examples and the derivation is fair, there is a subsequence $(G_{j_k}; \tau_{j_k})_{k\in\mathbb{N}}$ such that $(G_{j_k}; \tau_{j_k}) \succ_i (G_{j_{k+1}}, \tau_{j_{k+1}})$. Since $\succ_i$ is well-founded, this subsequence is finite. Let (G_n, τ_n) bei $\succ_i$-minimal in the sequence of counter-examples. Assume $Fail(\mathcal{G}_i)$ holds for no $i \succeq n$. Then G_n has to be processed, since $\mathcal{I}$ is $Fail$-complete and the derivation is fair. Processing G_n produces a smaller counter-example than (G_n, τ_n). But this is impossible since (G_n, τ_n) is minimal. Hence the $\mathcal{I}$-derivation is finite. It ends in a state $(\mathcal{H}_n; \mathcal{G}_n)$ such that $Fail(\mathcal{G}_n)$ holds since it is fair. □

We now have by Theorem 5.10

Corollary 5.11 *An inference system that strictly decreases inductive counter-examples and is $Fail$-complete is refutationally complete.* □

We now assume that the inference steps of $\mathcal{I}$ can be described by inference rules $\mathcal{I}$. Then we are interested in conditions on these inference rules to guarantee properties (I) to (III) for $\mathcal{I}$.

As mentioned above, we assume that an inference rule I, when applied to a goal $G \in \mathcal{G}$ in state $(\mathcal{H}; \mathcal{G})$, produces a set $\mathcal{G}'$ of new goals and shifts G from the unprocessed goals to the set of processed goals. There may be several sets $\mathcal{G}'$ possible. We denote by $I(\mathcal{H}; \mathcal{G}, G)$ the system of these sets $\mathcal{G}'$. To ease notation we write $\mathcal{G}, G$ instead of $\mathcal{G} \cup \{G\}$ and $\mathcal{G}, \mathcal{G}'$ instead of $\mathcal{G} \cup \mathcal{G}'$. Using these notations we describe the inference steps based on I by

$$I\colon \frac{(\mathcal{H}; \mathcal{G}, G)}{(\mathcal{H}, G; \mathcal{G}, \mathcal{G}')}$$

So $(\mathcal{H}_1; \mathcal{G}_1) \vdash_{\mathcal{I}} (\mathcal{H}_2; \mathcal{G}_2)$ iff for some inference rule I we have $\mathcal{G}_1 = \mathcal{G} \cup \{G\}$, $\mathcal{G}' \in I(\mathcal{H}; \mathcal{G}, G)$, $\mathcal{H}_2 = \mathcal{H}_1 \cup \{G\}$ and $\mathcal{G}_1 = \mathcal{G} \cup \mathcal{G}'$.

We now come to the conditions on the inference rules I.

Definition 5.12 *a) I* decreases (strictly decreases) inductive counter-examples *if the following holds for every state $(\mathcal{H};\mathcal{G},G)$: If $\mathcal{G}' \in I(\mathcal{H};\mathcal{G},G)$ and (G,τ) is an inductive counter-example for $(\mathcal{H};G)$ then there is a $\mathcal{G}'$-counter-example (G',τ') such that $(G,\tau) \succeq_i (G',\tau')$ (resp. $(G,\tau) \succ_i (G',\tau')$).*
b) I is refutationally sound *if the following holds for every state $(\mathcal{H};\mathcal{G},G)$: If $\mathcal{G}' \in I(\mathcal{H};\mathcal{G},G)$ and $P(\mathcal{H},\mathcal{G},G)$ holds then $P(\mathcal{G}')$ holds.*

Lemma 5.13 *a) If each inference rule I in $\mathcal{I}$ is refutationally sound then $\mathcal{I}$ is refutationally sound.*
b) If each inference rule I in $\mathcal{I}$ decreases (strictly decreases) inductive counter examples then $\mathcal{I}$ decreases (strictly decreases) inductive counter-examples.

Proof: Assume $(\mathcal{H};\mathcal{G},G) \vdash (\mathcal{H},G;\mathcal{G},\mathcal{G}')$ because of $\mathcal{G}' \in I(\mathcal{H};\mathcal{G},G)$.
a) Assume that I is refutationally sound and $P(\mathcal{H},\mathcal{G},G)$ holds. We have to show that $P(\mathcal{H},G,\mathcal{G},\mathcal{G}')$ holds. By the assumption $P(\mathcal{H},G,\mathcal{G})$ holds. Since I is refutationally sound $P(\mathcal{G}')$ holds also. So $P(\mathcal{H},G,\mathcal{G},\mathcal{G}')$ holds.
b) Assume that I decreases inductive counter-examples and (G_0,τ_0) is an inductive counter-example for $(\mathcal{H};\mathcal{G},G)$. We have to show that there is an inductive counter-example (G_1,τ_1) for $(\mathcal{H},G;\mathcal{G},\mathcal{G}')$ such that $(G_0,\tau_0) \succeq_i (G_1,\tau_1)$.
b1) Assume $G_0 = G$. There is a $\mathcal{G}'$-counter example (G',τ') such that $(G_0,\tau_0) \succeq_i (G',\tau')$ since I decreases counter-examples. $P(\mathcal{H},\mathcal{G},G)$ holds below (G',τ') since (G_0,τ_0) is an inductive counter-example for $(\mathcal{H};\mathcal{G},G')$. If $P(\mathcal{G}')$ also holds below (G',τ') then (G',τ') is an inductive counter-example for $(\mathcal{H},G;\mathcal{G},\mathcal{G}')$. Otherwise, let (G_1,τ_1) be a $\succ_i$-minimal $\mathcal{G}'$-counter-example such that $(G',\tau') \succ_i (G_1,\tau_1)$. Then (G_1,τ_1) is an inductive counter-example for $(\mathcal{H},G;\mathcal{G},\mathcal{G}')$ with $(G_0,\tau_0) \succeq_i (G_1,\tau_1)$.
b2) Assume $G_0 \in \mathcal{G}$. As before $P(\mathcal{H},\mathcal{G},G)$ holds below (G_0,τ_0). If also $P(\mathcal{G}')$ holds below (G_0,τ_0) then (G_0,τ_0) is an inductive counter-example for $(\mathcal{H},G;\mathcal{G},\mathcal{G}')$. Otherwise, let (G_1,τ_1) be a minimal $\mathcal{G}'$-counter-example such that $(G_0,\tau_0) \succ_i (G_1,\tau_1)$. Then (G_1,τ_1) is an inductive counter-example for $(\mathcal{H},G;\mathcal{G},\mathcal{G}')$.
The case that $\mathcal{I}$ strictly decreases inductive counter-examples is similar. □

Now we restate our previous results in terms of inference rules instead in terms of inference systems. They will be applied later in this form.

Theorem 5.14 *a) Let all inference rules in $\mathcal{I}$ decrease inductive counter-examples. Then $\mathcal{I}$ is inductively correct: If $(\emptyset;\mathcal{G}) \overset{*}{\vdash} (\mathcal{H}';\emptyset)$ then $P(\mathcal{G})$ holds.*
b) Let all inference rules in $\mathcal{I}$ be refutationally sound. Then $\mathcal{I}$ is refutationally correct: If $(\emptyset;\mathcal{G}) \overset{*}{\vdash} (\mathcal{H}';\mathcal{G}')$ and $Fail(\mathcal{G}')$ holds then $P(\mathcal{G})$ does not hold.
c) Let all inference rules of $\mathcal{I}$ (i) strictly decrease inductive counter-examples and (ii) be refutationally sound. Let $\mathcal{I}$ be $Fail$-complete. Then $\mathcal{I}$ is refutationally complete.

Proof: a) This follows from Theorem 5.3, Corollary 5.7 and Lemma 5.13.
b) This follows from Theorem 5.5 und Lemma 5.13.
c) This follows from Corollary 5.11 and Lemma 5.13. □

Sometimes one has to verify that $\mathcal{G}' \in I(G) = I(\mathcal{H}; \mathcal{G}, G)$, i.e., that the inference step is $(\mathcal{H}; \mathcal{G}, G) \vdash (\mathcal{H}, G; \mathcal{G}, \mathcal{G}')$ is semantically valid. The verification then consists in proving that "$\overline{\mathcal{G}}$ holds in context $\mathcal{H}$". This is made precise by saying that state $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive. Then we have the "conditional inference rule": $\mathcal{G}' \in I(G)$ if $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive. The verification of $(\mathcal{H}; \overline{\mathcal{G}})$ being inductive can be done by recursively calling the inductive prover based on the inference system $\mathcal{I}$ and using Theorem 5.3. We will need this in section 6.2.

6 An inference system for inductive proofs

6.1 The general setting

In this chapter we instantiate the abstract prover described in chapter 5 for the following problem:

Input:	a)	$spec = (sig, R)$, an admissible hierarchical specification over $spec_0 = (sig_0, R_0)$
	b)	$G : \Phi \Longrightarrow \Psi$, a clause
Question:	Is	G valid in $\mathcal{A}_{spec}$?

We instantiate the abstract framework by saying what a syntactic and a semantic unit is, by defining the predicates P and $Fail$, by specifying the inference rules and by defining the induction ordering $\succeq_i$. In order to verify that the inference system $\mathcal{I}$ has properties (I), (II) and (III) respectively, we have to verify the corresponding assumptions, e.g. "P and $Fail$ are compatible", "$\mathcal{I}$ is inductively correct", "$\mathcal{I}$ is refutationally correct", and "$\mathcal{I}$ decreases counter-examples".

The inference system $\mathcal{I}$ will be described by only five inference rules. These inference rules are rather powerful. We will not specify a heuristic how to apply them. So we are not interested to propose a special prover for the problem stated above. Instead, we will prove that all crucial properties for $\mathcal{I}$ to be correct and refutationally complete are satisfied. So any concrete prover based on $\mathcal{I}$ is correct and refutationally complete.

Our inference system is based on results in [Bec94] and [WB94]. In [Bec94] one can find instantiations of the abstract prover for other proof tasks also, e.g. for proving that R is ground confluent. We mention that many different sets of inference rules are possible. Current work is directed to the design of such inference rules that can be directly implemented into a concrete prover. Note that there are no syntactic restrictions on R defining *spec*, except that *spec* is admissible. In particular, we do not assume that R contains left-linear rules only or that R follows some rules of constructor discipline. Simultaneous recursion in defining the operators $f \in F$ is allowed. R may be non-terminating. Furthermore, R may contain postive/negative conditional rules. So our setting is more general than that of any other inductive theorem prover we know of.

6.2 An instantiation of the abstract prover

We now fix the basic ingredients of the abstract prover. A *syntactic unit* $G^m = \langle G \mid m\rangle$ consists of a clause $G \equiv \Phi \Longrightarrow \Psi$ and a *measure term* $m = \beta(t_1, \ldots, t_n)$ with $\mathcal{V}ar(m) \subseteq \mathcal{V}ar(G)$. Here $\beta \notin F$ is a new (variadic) operator and the $t_i \in Term(F, V)$ are *sig*-terms. A *semantic unit* (G^m, τ) consists of a syntactic unit G^m and a ground substitution τ with $dom(\tau) = \mathcal{V}ar(G^m)$. The predicate P is defined by

$$P(G^m, \tau) \text{ iff } \mathcal{A}_{spec} \models \tau(G)$$

So $P(G^m, \tau)$ holds iff $\tau(G)$ is valid in $\mathcal{A}_{spec}$. We just say $\tau(G)$ is *valid* in this case. And $P(G^m)$ holds iff $G \in ITh(spec)$. We will define the failure predicate $Fail$ later.

Next we comment on the induction ordering $\succeq_i$ on the semantic units (G^m, τ). It is defined by a well-founded quasi-ordering $\succeq^i$ ordering on the measure ground terms

$$(G_1^{m_1}, \tau_1) \succeq_i (G_2^{m_2}, \tau_2) \text{ iff } \tau_1(m_1) \succeq^i \tau_2(m_2)$$

We will assume that $\succeq^i$ is *compatible with* R in the sence that $\tau_1(m) \succeq^i \tau_2(m)$ for all ground substitutions τ_1, τ_2 such that $\tau_1(x) =_R \tau_2(x)$ for all $x \in \mathcal{V}ar(m)$. So, if τ_1 and τ_2 are R-equal then $\tau_1(m)$ and $\tau_2(m)$ are of the same size. We will comment on this restriction later. Here we just mention that this restriction is not as severe as it may seem: In many applications the measure term $m \equiv \beta(t_1, \ldots, t_n)$ will contain base terms t_i only. Then, if $R_0 = \emptyset$ (i.e., the constructors are free) we have $\tau_1(x) \equiv \tau_2(x)$ whenever τ_1 and τ_2 are R-equal. In this case the restriction is empty.

We extend the ordering $\succeq^i$ to measure terms with variables by $m_1 \succeq^i m_2$ iff $\tau(m_1) \succeq^i \tau(m_2)$ for each ground substitution τ. Then $G_1^{m_1} \succeq_i G_2^{m_2}$ iff $m_1 \succeq^i m_2$.

There is some freedom for constructing those induction orderings $\succeq^i$. For example, let $\succeq_0$ be a well-founded quasi-ordering on $Term(F)$ such that $t_1 =_R t_2$ implies $t_1 \succeq_0 t_2$. Let $\gg_0$ be (strict part of the) the multiset ordering based on $\succeq_0$. Then we may define $\beta(t_1, \ldots, t_n) \succ^i \beta(t'_1, \ldots, t'_n)$ iff $\{t_1, \ldots, t_1\} \gg_0 \{t'_1, \ldots, t'_n\}$. Or, let $\succ_{0,lex}$ be the lexicographic ordering based on $\succ_0$. Then we may define $\beta(t_1, \ldots, t_n) \succ^i \beta(t'_1, \ldots, t'_n)$ iff $(t_1, \ldots, t_n) \succ_{0,lex} (t'_1, \ldots, t'_n)$. We do not go into details here how to choose $\succeq_0$. In the special case that the condition $\tau_1(x) =_R \tau_2(x)$ for all x (mentioned above) reduces to $\tau_1(x) \equiv \tau_2(x)$, one may use any reduction ordering for $\succeq_0$.

The inference system $\mathcal{I}$ has three parameters: $spec = (sig, R)$, the induction ordering $\succeq_i$ and a set $\mathcal{L}$ of lemmas. We assume $R \subseteq \mathcal{L} \subseteq ITh(spec)$. An inference rule I is specified by the set $I(\mathcal{H}; \mathcal{G}, G^m)$ of actions it can perform for processing G^m in state $(\mathcal{H}; \mathcal{G}, G^m)$. We define

$$(\mathcal{H}; \mathcal{G}, G^m) \vdash_I (\mathcal{H}, G^m; \mathcal{G}, \mathcal{G}') \quad \text{iff} \quad \mathcal{G}' \in I(\mathcal{H}; \mathcal{G}, G^m)$$

Then $\vdash_{\mathcal{I}}$ is the union of all $\vdash_I$ such that I is an inference rule of $\mathcal{I}$. As before, we write $\vdash$ instead of $\vdash_{\mathcal{I}}$. So, an inference step in state $(\mathcal{H}; \mathcal{G}_1)$ is possible iff $\mathcal{G}_1 = \mathcal{G}, G^m$ and $I(\mathcal{H}; \mathcal{G}, G^m) \neq \emptyset$ for some inference rule I.

Informally, the inference rule I_0 covers the trivial cases to process G^m. Rules I_1 and I_2 perform a case splitting on G, rule I_3 performs a simplification on G^m (using a clause $H^{m'} \in \mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$) and I_4 performs a subsumption of G^m (using a clause $H^{m'} \in \mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$). For rules I_3 and I_4 either "safe knowledge" $H^{m'} \in \mathcal{L}$ or "inductive knowledge" $H^{m'} \in \mathcal{H} \cup \mathcal{G}$ will be used. In the latter case $H^{m'}$ has to be smaller than G^m according to the induction ordering $\succeq_i$. This will be made precise below.

We first indicate how a proof (or disproof) of $\{G_1, \ldots, G_m\} \subseteq ITh(spec)$ is performed: We start in state $(\mathcal{H}_0; \mathcal{G}_0) = (\emptyset; \{G_1^m, \ldots, G^m\})$ with $m \equiv \beta(x_1, \ldots, x_h)$, $x_j \in \mathcal{V}ar(\mathcal{G}_0)$. If $(\mathcal{H}_0; \mathcal{G}_0) \overset{*}{\vdash} (\mathcal{H}'; \emptyset)$ holds then $\mathcal{G}_0 \subseteq ITh(spec)$ is proved. If $(\mathcal{H}_0, \mathcal{G}_0) \overset{*}{\vdash} (\mathcal{H}'; \mathcal{G}')$ and $Fail(\mathcal{G}')$ holds then $\mathcal{G}_0 \subseteq ITh(spec)$ is disproved.

We now define the inference rules I_0 to I_4. We demonstrate how they work using the current rewrite system

$$\begin{array}{llll} R^*: & \Longrightarrow & x - 0 \to x & (\rho_1) \\ & \Longrightarrow & s(x) - s(y) \to x - y & (\rho_2) \end{array}$$

Note that here the constructors are free, i.e. $R_0 = \emptyset$. We use the induction ordering $\succ_i$ which is defined by the multiset ordering based on the subterm ordering $\rhd$.

An atom is *trivial* if it is of the form $u = u$ or $def(t)$ with $t \in Term(F_0, V_0)$.

Rule I_0: Trivial actions
If $G \equiv \Phi \Longrightarrow A, \Psi$ and A is trivial then $\emptyset \in I_0(\mathcal{H}; \mathcal{G}, G^m)$. If $G \equiv \Phi,\ A \Longrightarrow \Psi$ and A is trivial then $\{G'^m\} \in I_0(\mathcal{H}; \mathcal{G}, G^m)$ with $G' \equiv \Phi \Longrightarrow \Psi$. If $G \equiv \Phi,\ A \Longrightarrow A, \Psi$ then $\emptyset \in I_0(\mathcal{H}; \mathcal{G}, G^m)$. If $G \equiv \Phi \Longrightarrow A, A, \Psi$ then $\{G'^m\} \in I_0(\mathcal{H}; \mathcal{G}, G^m)$ with $G' \equiv \Phi \Longrightarrow A, \Psi$. If $G \equiv \Phi, A, A \Longrightarrow \Psi$ then $\{G'^m\} \in I_0(\mathcal{H}; \mathcal{G}, G^m)$ with $G' \equiv \Phi, A \Longrightarrow \Psi$.

The rule I_0 is easy to understand: If $G \equiv \Phi \Longrightarrow \Psi$ and Ψ contains a trivial atom then the goal G^m is solved. If Φ contains a trivial atom A, then G can be simplified to G' by deleting A. If an atom A appears both, in the antecedens and in the succedens of G, then the goal G^m is solved. Note that G^m and G'^m have the same measure term m, so G^m is only weakly decreased to G'^m in the induction ordering $\succeq_i$. We write $I_0(G^m)$ instead of $I_0(\mathcal{H}; \mathcal{G}, G^m)$ since I_0 only depends on G^m, not on $\mathcal{H}$ and $\mathcal{G}$.

We now come to rule I_1. It is used to perform a case splitting of $G \equiv \Phi \Longrightarrow \Psi$ according to the succedens Ψ of G. This is done by computing a covering for Ψ as in the next definition.

Definition 6.1 *Let Ψ be a set of atoms. A covering $Cov(\Psi)$ for Ψ is a set of substitutions σ such that for each ground substitution τ with $dom(\tau) = \mathcal{V}ar(\Psi)$ there is a $\sigma \in Cov(\Psi)$ and a τ' such that $\tau(x) =_R \tau'(\sigma(x))$ for all $x \in \mathcal{V}ar(\Psi)$.*

Note that many coverings for Ψ may exist. For example, $Cov(\Psi) = \{\sigma_f \mid f \in F\}$ is a covering for any Ψ. Here $\sigma_f = \{x \leftarrow f(x_1, \ldots, x_n)\}$. It is in general a non-trivial task to compute a suitable covering, i.e. one that leads to a successful proof. We comment on that in section 7.1.

Rule I_1: Covering the succedens
Let $G^m = \langle \Phi \Longrightarrow \Psi \mid m \rangle$ and let $Cov(\Psi)$ be a covering for Ψ. Let $\mathcal{G}'$ consist of all $\langle \sigma(G) \mid \sigma(m) \rangle$ such that $\sigma \in Cov(\Psi)$. Then $\mathcal{G}' \in I_1(\mathcal{H}; \mathcal{G}, G^m)$.

Note that $I_1(\mathcal{H}; \mathcal{G}, G^m)$ is independent of $\mathcal{H}$ and $\mathcal{G}$, so we simply write $I_1(G^m)$. Note also that $I_1(G^m)$ consists of many $\mathcal{G}'$, each such $\mathcal{G}'$ is determined by the covering $Cov(\Psi)$.

We illustrate rule I_1 using the current rewrite system R^* and the goal $G^m = \langle \Longrightarrow x - x = 0 \mid \beta(x) \rangle$. Let $\Psi = \{x - x = 0\}$ and let $Cov(\Psi)$ consist of $\sigma_1 = \{x \leftarrow 0\}$ and $\sigma_2 = \{x \leftarrow s(x)\}$. Then $Cov(\Psi)$ is a covering for Ψ and $\mathcal{G}'$ consists of $\langle \Longrightarrow 0 - 0 = 0 \mid \beta(0) \rangle$ and $\langle \Longrightarrow s(x) - s(x) = 0 \mid \beta(s(x)) \rangle$.

We now come to rule I_2. It is used to perform a case splitting on an atom A in the antecedens of a clause $G \equiv \Phi,\ A \Longrightarrow \Psi$. The main idea is as follows. Call σ a *solution* of A iff $\mathcal{A}_{spec} \models \sigma(A)$. Clearly, in order to prove $\mathcal{A}_{spec} \models \Phi,\ A \Longrightarrow \Psi$ it is enough to prove $\mathcal{A}_{spec} \models \sigma(\Phi \Longrightarrow \Psi)$ for all solutions σ of A. Computing solutions of A can be done by narrowing techniques. This will be discussed in section 7.1.

Definition 6.2 *Let A be an atom. A weak covering $WCov(A)$ for A is a set of substitutions σ such that: For each ground substitution τ such that $dom(\tau) = \mathcal{V}ar(A)$ and τ is a solution of A there is a $\sigma \in WCov(A)$ and a τ' such that $\tau(x) =_R \tau'(\sigma(x))$ for all $x \in \mathcal{V}ar(A)$.*

We will discuss in section 7.1 how to compute a weak covering for A. For example, if R is confluent and terminating, let $WCov(A)$ consist of (1) all $\sigma = mgu(u,l)$ with $u \equiv A/p$ a non-variable subterm in A and l the left-hand side of a rule $\Gamma;\Delta \Longrightarrow l \rightarrow r$ in R and (2) $\mu = mgu(u,v)$ if A is $u = v$ and u, v are unifiable. Then $WCov(A)$ is a weak covering for A. We will use this fact below.

Rule I_2: Covering the antecedens
Let $G^m = \langle \Phi, A \Longrightarrow \Psi \mid m \rangle$ and let $WCov(A)$ be a weak covering for A. Let $\mathcal{G}'$ consist of all $\langle \sigma(G) \mid \sigma(m) \rangle$ such that $\sigma \in WCov(A)$. Then $\mathcal{G}' \in I_2(\mathcal{H};\mathcal{G},G^m)$.

Note that $I_2(\mathcal{H};\mathcal{G},G^m)$ is independent of $\mathcal{H}$ and $\mathcal{G}$, so we simply write $I_2(G^m)$. Note also that $I_2(G^m)$ may consist of many $\mathcal{G}'$, each $\mathcal{G}'$ is determined by the weak covering $WCov(A)$. Finally note that any atom A in the antecedens Φ of $G \equiv \Phi \Longrightarrow \Psi$ can be selected for the case splitting.

We illustrate rule I_2 using R^* and the goal $G^m = \langle x - y = 0 \Longrightarrow x = y \mid \beta(x,y) \rangle$. Note that R^* is confluent and terminating. We have $A \equiv x - y = 0$ and $WCov(A) = \{\sigma_1, \sigma_2\}$ is a weak covering for A, where $\sigma_1 = \{y \leftarrow 0\}$ and $\sigma_2 = \{x \leftarrow s(x), y \leftarrow s(y)\}$. So $\mathcal{G}'$ consists of $\langle x - 0 = 0 \Longrightarrow x = 0 \mid \beta(x,0) \rangle$ and $\langle s(x) - s(y) = 0 \Longrightarrow s(x) = s(y) \mid \beta(s(x),s(y)) \rangle$.

The rules I_1 and I_2 are used to perform a case splitting on the actual goal G^m and so to produce new goals. The rule I_0 is used to syntactically simplify or to delete G^m. The next two rules will be used to simplify or delete G^m using a lemma $H^{m'} \in \mathcal{L}$ or an induction hypothesis $H^{m'} \in \mathcal{H} \cup \mathcal{G}$. To make our inference rules more powerful we now extend the notion of a substitution. Recall that for a substitution σ we require that $\sigma(x) \in Term(F_0, V_0)$ if $sort(x) \in S_0$. (A base variable may only be replaced by a base term.) We relax this condition.

Definition 6.3 *A quasi-substitution is a mapping $\sigma : V \rightarrow Term(F,V)$ such that $sort(x) = sort(\sigma(x))$ for all $x \in V$. σ is extended to $\sigma : Term(F,V) \rightarrow Term(F,V)$ by $\sigma(f(t_1,\ldots,t_n)) \equiv f(\sigma(t_1),\ldots,\sigma(t_n))$.*

A quasi-substitution σ is *almost a substitution* if $\mathcal{A}_{spec} \models def(\sigma(x))$ for all $x \in dom(\sigma) \cap V_0$. We will allow σ to be a quasi-substitution in the following. But then we have to make sure that the goals $\Longrightarrow def(\sigma(x))$ are satisfied for all $x \in dom(\sigma) \cap V_0$ in the given context. If σ is almost a substitution then there is a substitution σ' such that $dom(\sigma) = dom(\sigma')$ and $\sigma(x) = \sigma'(x)$ is valid for all $x \in dom(\sigma)$. We require that $\sigma(m) \succeq^i \sigma'(m)$ for each measure term m. This is part of our general assumption that $\succeq^i$ is compatible with R. We comment on that in section 7.3.

If H is a clause and σ is a quasi-substitution then $Def(H,\sigma) = \{def(\sigma(x)) \mid x \in \mathcal{V}ar(H) \cap V_0 \ , \ \sigma(x) \notin Term(F_0,V_0)\}$.

Rule I_3: Clausal rewriting
Let $G^m \equiv \langle \Phi, A \Longrightarrow \Psi \mid m\rangle$ (resp. $G^m \equiv \langle \Phi \Longrightarrow A, \Psi \mid m\rangle$) be a goal. Let $H^{m'} \in \mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$ be a clause $H^{m'} \equiv \langle \Gamma \Longrightarrow u = v, \Delta \mid m'\rangle$. Let σ be a quasi-substitution such that $A/p \equiv \sigma(u)$. Let $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ if $H \in \mathcal{H} \cup \mathcal{G}$ and $\tau(G)$ is not valid. Let $\overline{\mathcal{G}}$ consist of all elements (1) $\langle \Phi, \sigma(B) \Longrightarrow \Psi \mid m\rangle$ with $B \in \Delta$ and (2) $\langle \Phi \Longrightarrow \sigma(B), \Psi \mid m\rangle$ with $B \in \Gamma$ and (3) $\langle \Phi \Longrightarrow B, \Psi \mid m\rangle$ with $B \in Def(H, \sigma)$. Let $(\mathcal{H}; \overline{\mathcal{G}})$ be inductive. Let $\mathcal{G}' = \{\langle \Phi, A[\sigma(v)]_p \Longrightarrow \Psi \mid m\rangle\}$ (resp. $\mathcal{G}' = \{\langle \Phi \Longrightarrow A[\sigma(v)]_p, \Psi \mid m\rangle\}$). Then $I_3(\mathcal{H}; \mathcal{G}, G^m)$ consists of all these $\mathcal{G}'$.

We comment on rule I_3. H transforms G into G' by replacing A with $A[\sigma(v)]_p$. We have to fulfill some conditions that this transformation is semantically allowed: (a) σ is almost a substitution. This is reflected by the goals (3) in $\overline{\mathcal{G}}$. (b) $\sigma(H)$ is applicable: This is reflected by the goals (1) and (2) in $\overline{\mathcal{G}}$. (c) $\sigma(H)$ can be used as an induction hypothesis. This reflected by the condition $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ for all ground substitutions τ such that $\tau(G)$ is not valid. This can be verified by showing $G^m \succ_i \sigma(H^{m'})$. The proof that $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive will be done by recursively calling the prover we are just describing. This is based on Theorem 5.3.

The condition $G^m \succ_i \sigma(H^{m'})$, i.e. $m \succ^i \sigma(m')$, is evaluated as follows. If σ is a substitution then $\sigma(m')$ is well defined. If σ is a quasi-substitution, then evaluating the goals (3) in $\overline{\mathcal{G}}$ may compute a substitution σ' such that $\Phi \Longrightarrow \sigma(x) = \sigma'(x), \Psi$ is valid in context $\mathcal{H}$. Then $m \succ^i \sigma'(m')$ has to be checked. In general, one has to perform an inductive proof for verifying $m \succ^i \sigma(m')$. This can be incorporated in the prover we are just describing (see [Wir96]).

Notice that $I_3(\mathcal{H}; \mathcal{G}, G^m)$ depends on $\mathcal{H}$, $\mathcal{G}$ and G^m. We will write $I_3(G^m)$ instead of $I_3(\mathcal{H}; \mathcal{G}, G^m)$ if $\mathcal{H}$, $\mathcal{G}$ is clear from the context. It contains several $\mathcal{G}'$ since different A/p with A in G can be selected and different $H \in \mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$ may be used.

We illustrate rule I_3 using the current rewrite system R^* and the goal $G \equiv s(x) - s(y) = 0 \Longrightarrow s(x) = s(y)$. Let H be the rule ρ_2 of $R^* \subseteq \mathcal{L}$, so $H \equiv \Longrightarrow s(x) - s(y) = x - y$. (We do not need the measure terms m in G^m and m' in $H^{m'}$ since $H \in \mathcal{L}$.) We choose $A \equiv s(x) - s(y) = 0$ and $\sigma = id$. This gives $\overline{\mathcal{G}} = \emptyset$ since σ is a substitution and Γ, Δ are empty. $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive. So $\mathcal{G}'$ consists of $\langle x - y = 0 \Longrightarrow s(x) = s(y) \mid m\rangle$ only.

Rule I_3 is used to rewrite a goal using a lemma or an induction hypothesis. It can also be used for contextual rewriting. We comment on that in section 7.2.

We now come to the last inference rule I_4. It performs a partial subsumption on a clause $G \equiv \Phi_0,\, \Phi_1 \Longrightarrow \Psi_0, \Psi_1$ using $H \equiv \Gamma_0,\, \Gamma_1 \Longrightarrow \Delta_0, \Delta_1$. If $\Gamma_0 \Longrightarrow \Delta_0$ subsumes $\Phi_0 \Longrightarrow \Psi_0$ and "the rest can be proved" then G can be deleted from the unprocessed goals without producing new goals.

Rule I_4: Subsumption
Let $G^m \equiv \langle \Phi_0, \Phi_1 \Longrightarrow \Psi_0, \Psi_1 \mid m\rangle$ be a goal and $H^{m'} = \langle \Gamma_0, \Gamma_1 \Longrightarrow \Delta_0, \Delta_1 \mid m'\rangle$ in $\mathcal{H} \cup \mathcal{G} \cup \mathcal{L}$. Let σ be a quasi-substitution such that $\sigma(\Gamma_0) \subseteq \Phi_0$ and $\sigma(\Delta_0) \subseteq \Psi_0$. Let $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ if $H \in \mathcal{H} \cup \mathcal{G}$ and $\tau(G)$ is not valid. Let $\overline{\mathcal{G}}$ consist of the elements (1) $\langle \Phi_1, \sigma(B) \Longrightarrow \Psi_1 \mid m\rangle$ with $B \in \Delta_1$ and (2) $\rangle \Phi_1 \Longrightarrow \sigma(B), \Psi_1 \mid m\rangle$ with $B \in \Gamma_1$ and (3) $\langle \Phi_1 \Longrightarrow B, \Psi_1 \mid m\rangle$ with $B \in Def(H, \sigma)$. Let $(\mathcal{H}; \overline{\mathcal{G}})$ be inductive. Then $I_4(\mathcal{H}; \mathcal{G}, G^m) = \{\emptyset\}$.

We comment on rule I_4. The goals (3) in $\overline{G}$ assure that σ is almost a substitution σ' as for rule I_3. The goals (1) and (2) in $\overline{G}$ make the term "the rest can be proved" precise. The condition $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ for all τ such that $\tau(G)$ is not valid assures that $\sigma(H^{m'})$ can be used as induction hypothesis.

Notice that $I_4(\mathcal{H}; \mathcal{G}, G^m)$ depends on $\mathcal{H}, \mathcal{G}$ and G^m; we will write $I_4(G^m)$ instead of $I_4(\mathcal{H}; \mathcal{G}, G^m)$ if $\mathcal{H}, \mathcal{G}$ is clear from the context.

We illustrate rule I_4 using R^* and the goal $G^m = \langle x - y = 0 \Longrightarrow s(x) = s(y) \mid \beta(s(x), s(y))\rangle$ and $H^{m'} = \langle x - y = 0 \Longrightarrow x = y \mid \beta(x, y)\rangle$ in $\mathcal{H}$. We choose $\Phi_0 = \Gamma_0 = \{x - y = 0\}$, $\Phi_1 = \Gamma_1 = \emptyset$, $\Psi_0 = \Delta_0 = \emptyset$, $\Psi_1 = \{s(x) = s(y)\}$ and $\Delta_1 = \{x = y\}$ and $\sigma = id$. We will use the lemma $L \equiv x = y \Longrightarrow s(x) = s(y) \in ITh(spec)$. We have $\overline{\mathcal{G}} = \{\langle x = y \Longrightarrow s(x) = s(y) \mid \beta(x, y)\rangle\}$ and have to prove that $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive. For that we start the prover and apply I_4 to $\langle \overline{G} \mid \overline{m}\rangle$, with $\overline{G} \equiv x = y \Longrightarrow s(x) = s(y)$, $\overline{m} \equiv \beta(x, y)$ and L. Obviously, $I_4(\overline{G}) = \{\emptyset\}$. So $(\mathcal{H}; \langle \overline{G} \mid \overline{m}\rangle) \vdash (\mathcal{H}, \langle \overline{G} \mid \overline{m}\rangle; \emptyset)$, and $(\mathcal{H}; \langle \overline{G} \mid \overline{m}\rangle)$ is inductive by Theorem 5.3. Since $\beta(s(x), s(y)) \succ_i \beta(x, y)$ this gives $I_4(G^m) = \{\emptyset\}$.

Definition 6.4 *The inference system $\mathcal{I}$ consists of the inference rules I_0 to I_4.*

We now give a proof for $x - y = 0 \Longrightarrow x = y \in ITh(spec)$ for $spec = (sig, R^*)$. For that we assume that $\mathcal{I}$ is inductively correct, which will be proved later. We need the following goals:

$$\begin{array}{lcl}
G^0 & = & \langle x - y = 0 \Longrightarrow x = y \mid \beta(x, y)\rangle \\
G^1 & = & \langle x - 0 = 0 \Longrightarrow x = 0 \mid \beta(x, 0)\rangle \\
G^2 & = & \langle s(x) - s(y) = 0 \Longrightarrow s(x) = s(y) \mid \beta(s(x), s(y))\rangle \\
G^3 & = & \langle x = 0 \Longrightarrow x = 0 \mid \beta(x, 0)\rangle \\
G^4 & = & \langle x - y = 0 \Longrightarrow s(x) = s(y) \mid \beta(s(x), s(y))\rangle
\end{array}$$

Based on our preceeding illustrations of the rules I_0 to I_4 we get

$$\begin{array}{ll}
(\emptyset; \{G^0\}) & \vdash_{I_2} \; (\{G^0\}; \{G^1, G^2\}) \vdash_{I_3} (\{G^0, G^1\}; \{G^2, G^3\}) \\
 & \vdash_{I_3} \; (\{G^0, G^1, G^2\}; \{G^3, G^4\} \vdash_{I_0} (\{G^0, G^1, G^2, G^3\}; \{G^4\}) \\
 & \vdash_{I_4} \; (\{G^0, \ldots, G^4\}; \emptyset)
\end{array}$$

From that we conclude that the clause $x - y = 0 \Longrightarrow x = y$ is in $ITh(spec)$.

6.3 Correctness of the inference system $\mathcal{I}$

In this section we will prove that the inference rules I_0 to I_4 decrease inductive counter-examples and that they are refutationally sound. We will also give a failure predicate $Fail$ that is compatible with P. This proves that $\mathcal{I}$ works correctly.

Let $spec = (sig, R)$ be an admissible hierarchical specification over $spec_0 = (sig_0, R_0)$. We say that an atom A or a clause G is *valid* if it is valid in $\mathcal{A}_{spec}$. So G is valid if $G \in ITh(spec)$.

Notice that the inference rules do not depend on the fact that R is a rewrite system, i.e., that the conditional equations are directed. So we may regard R to be a set of (undirected) conditional equations. Later in the paper we will restrict R to be a rewrite system in order to reduce the search space for carrying out inductive proofs.

We now come to the central lemmas of this section. The proofs are given in the Appendix.

Lemma 6.5 *The inference rules I_0 to I_4 decrease inductive counter-examples.*

Lemma 6.6 *The inference rules I_0 to I_4 are refutationally sound.*

We now study how to get refutational correctness. For that we need a failure predicate $Fail$. We first define a relatively weak failure predicate.

Definition 6.7 *Let $G \equiv \Phi \Longrightarrow \Psi$ be a clause. $Fail_0(G^m)$ holds iff for each atom $A \in G$ there is an atom A' such that $\Longrightarrow A'$ is in $\mathcal{L}$ if $A \in \Phi$ and $A' \Longrightarrow$ is in $\mathcal{L}$ if $A \in \Psi$ and (i) $\mathcal{V}ar(G) \cap \mathcal{V}ar(A') = \emptyset$ for all $A \in G$ and (ii) there is a substitution σ such that $\sigma(A) \equiv \sigma(A')$ for all $A \in G$.*

The condition (ii) states that all pairs (A, A') are simultaneously unifiable by some σ. For example, if $G \equiv \Longrightarrow def(g(0,x))$ and $def(g(y,h(z))) \Longrightarrow$ is in $\mathcal{L}$ (or in $ITh(spec)$ then $Fail_0(G^m)$ holds for any measure term m. One easily proves

Lemma 6.8 *$Fail_0$ is compatible with P.* □

Definition 6.9 *The inference system $\mathcal{I}_0$ is given by the inference rules I_0 to I_4, the predicate P and the failure predicate $Fail_0$.*

We now have the first main result of this section.

Theorem 6.10 *$\mathcal{I}_0$ is inductively and refutationally correct:*
a) If $(\emptyset; \mathcal{G}) \overset{*}{\vdash}_{\mathcal{I}_0} (\mathcal{H}'; \emptyset)$ then $P(\mathcal{G})$ holds.
b) If $(\emptyset; \mathcal{G}) \overset{*}{\vdash}_{\mathcal{I}_0} (\mathcal{H}'; \mathcal{G}')$ and $Fail_0(G^m)$ holds for some $G^m \in \mathcal{G}'$ then $P(\mathcal{G})$ does not hold.

Proof: This follows from Lemmas 6.5 to 6.8 and Theorem 5.14 □

We now come to a stronger failure predicate $Fail$. It can be applied only if R is a confluent and terminating rewrite system.

The basic idea is as follows: A clause $G \equiv \Longrightarrow \Psi$ is not an inductive theorem if there is a ground substitution τ such that $\longrightarrow_R$ satisfies no $\tau(A)$ with $A \in \Psi$. In [Bac88] R is restricted to be an unconditional rewrite system and Ψ is restricted to consist of an equality atom $u = v$ only. Then there is a τ such that $\longrightarrow_R$ does not satisfy $\tau(u = v)$ if $u = v$ is not ground reducible in R. We extend this approach to our setting.

Let γ and $\#$ be two new function symbols, let $G \equiv\Longrightarrow \Psi$ and $\Psi = \{u_1 = v_1, \ldots, u_n = v_n\} \cup \{def(t_1), \ldots, def(t_m)\}$. Then we denote by $\gamma(G)$ the term $\gamma(u_1\#v_1, \ldots, u_n\#v_n, t_1, \ldots, t_m)$. We denote by R_u and $R_u^\#$ the unconditional rewrite systems

$$\begin{array}{lcl} R_u & = & \{l \rightarrow r \mid \Gamma; \Delta \Longrightarrow l \rightarrow r \text{ in } R\}\ , \\ R_u^\# & = & R_u \cup \{x\#x \rightarrow x\}. \end{array}$$

We say that G is ground reducible in R_u if $\tau(\gamma(G))$ is reducible in $R_u^\#$ for each ground substitution τ. It is well known that ground reducibility is decidable for unconditional rewrite systems.

Definition 6.11 *a) $Fail(G^m)$ holds if either $Fail_0(G^m)$ holds or $G \equiv\Longrightarrow \Psi$, $I_0(G^m) = \emptyset$ and G is not ground reducible in R_u.*
b) The inference system $\mathcal{I}$ is given by the inference rules I_0 to I_4, the predicate P and the failure predicate $Fail$.

Lemma 6.12 *Let R be confluent and terminating. Then $Fail$ is compatible with P.*

Proof: Assume $Fail(G^m)$ holds, we have to show that $P(G^m)$ does not hold. If $Fail_0(G^m)$ holds then $P(G^m)$ does not hold by Lemma 6.8. So assume $G \equiv\Longrightarrow \Psi$, $I_0(G^m) = \emptyset$ and G is not ground reducible in R_u. We have to show that $\tau(G)$ is not valid for some ground substitution τ. Since G is not ground reducible in R_u, there is a τ such that $\tau(A)$ is irreducible in R_u (and hence in R) for all A in Ψ and $\tau(u) \not\equiv \tau(v)$ for all $u = v$ in Ψ. Hence $\longrightarrow_R$ satisfies no equality atom $\tau(A)$ with A in Ψ, since R is confluent. And $\longrightarrow_R$ satisfies no def-atom $\tau(A)$ with A in Ψ, since $I_0(G^m) = \emptyset$ and so Ψ contains no atom $def(t)$ with $t \in Term(F_0, V_0)$. So $\tau(G)$ is not valid. □

We are now ready to give the main result of this section. The next Theorem is much more powerful than Theorem 6.10. See the examples in section 7.

Theorem 6.13 *The inference system $\mathcal{I}$ is inductively correct. It is also refutationally correct for all R which are confluent and terminating.*

Proof: This follows from Lemmas 6.5, 6.6, 6.12 and Theorem 5.14. □

By Theorems 6.10 and 6.13 we can prove and disprove inductive theorems. We did not discuss refutational completeness of the inference system. For unconditional total specifications this is easy to reach, see [Bac88]. For conditional specifications this leads to undecidable problems. We will comment on that in section 7.2.

7 Applying the inference system $\mathcal{I}$

In this chapter we comment on how to apply the inference rules. We also give examples of inductive proofs based on $\mathcal{I}$.

7.1 On applying the rules I_1 and I_2

To apply rule I_1 on $G^m = \langle \Phi \Longrightarrow \Psi \mid m \rangle$, we have to find a covering $Cov(\Psi)$ for Ψ. To apply rule I_2 to $G^m = \langle \Phi, A \Longrightarrow \Psi \mid m \rangle$, we have to find a weak covering $WCov(A)$ for A. There are several approaches for that, they depend on assumptions on R, e.g., whether R is terminating or confluent.

Definition 7.1 *A* covering set of substitutions *is a set $CSubst$ of substitutions such that for each ground substitution τ there is a $\sigma \in CSubst$ and a τ' such that $\tau(x) \equiv \tau'(\sigma(x))$ for all $x \in dom(\tau)$.*

There are several ways to compute a covering set of substitutions. We discuss how to cover a fixed sort $s \in S$. Let $sort(x) = s$, $F_s = \{f \in F \mid f : s_1, \ldots, s_n \to s\}$ and $\sigma_f = \{x \leftarrow f(x_1, \ldots, x_n)\}$ for $f \in F_s$. Then $CSubst = \{\sigma_f \mid f \in F_s\}$ is a covering set of substitutions for any Ψ with $x \in Var(\Psi)$. It may be refined to get a finer case splitting. We demonstrate that with $sig = (S, F, V, \alpha)$, $S = \{NAT\}$ and $F = \{0, s\}$. Here $CSubst = \{\sigma_1, \sigma_2, \sigma_3\}$ is a covering set of substitutions for Ψ, where $\sigma_1 = \{x \leftarrow 0\}$, $\sigma_2 = \{x \leftarrow s(0)\}$ and $\sigma_3 = \{x \leftarrow s(s(x))\}$. One may also use several variables to cover sort $s \in S$. For example, $CSubst = \{\sigma_1, \sigma_2, \sigma_3\}$ is a covering set of substitutions for Ψ, where $\sigma_1 = \{x \leftarrow 0\}$, $\sigma_2 = \{x \leftarrow s(x), y \leftarrow 0\}$ and $\sigma_3 = \{x \leftarrow s(x), y \leftarrow s(y)\}$.

Note that any covering set of substitutions is a covering for Ψ and a weak covering for A, where Ψ and A are arbitrary. So, if a suitable set $CSubst$ is known then rules I_1 and I_2 can be applied. (Here "suitable" means that the corresponding case splitting leads to a successful proof.)

The notion of a covering set of substitutions refers to sig only, not to the defining equations in R. Now we assume that R is terminating. This allows one to compute smaller coverings for Ψ and weak coverings for A and so to compute smaller $\mathcal{G}' \in I_1(G^m)$ and $\mathcal{G}' \in I_2(G^m)$, respectively.

Definition 7.2 *Let R be terminating. An* R-covering set of substitutions *$CSubst$ is a set of substitutions such that for each R-irreducible ground substitution τ there is a $\sigma \in CSubst$ and a τ' such that $\tau(x) \equiv \tau'(\sigma(x))$ for all $x \in dom(\tau)$. Here τ is R-irreducible if $\tau(x)$ is R-irreducible for all $x \in dom(\tau)$.*

Note, if $CSubst$ is an R-covering set of substitutions then $CSubst$ is a covering for any Ψ and a weak covering for any A. So $CSubst$ may be used for applying rules I_1 and I_2.

To give an example, assume $spec_0 = (sig_0, R_0)$ with $sig_0 = (\{NAT\}, \{0, s, +\}, V, \alpha)$ and

R_0: $\Longrightarrow x + 0 \to x \qquad\qquad \Longrightarrow x + s(y) \to s(x + y)$

Then $CSubst$ consisting of $\{x \leftarrow 0\}$ and $\{x \leftarrow s(x)\}$ is an R-covering set of substitutions but not a covering set of substitutions.

We now discuss how to compute a weak covering for A. We may assume that A is not trivial, i.e., not of the form $u = u$ or $def(t)$ with $t \in Term(F_0, V_0)$.

Lemma 7.3 *Let R be confluent and terminating and A be a non-trivial atom. Let $WCov(A)$ consist of all $\sigma = mgu(A/p, l)$ where A/p is a non-variable subterm of A and $\Gamma; \Delta \Longrightarrow l \to r$ a rule in R. If A is $u = v$ and $\mu = mgu(u, v)$ exists, then let $WCov(A)$ contain μ also. Then $WCov(A)$ is a weak covering for A.*

Proof: Let τ be a ground substitution such that $\tau(A)$ is valid. We have to show that there is a $\sigma \in WCov(A)$ and a τ' such that $\tau(x) =_R \tau'(\sigma(x))$. Since R is terminating, there is an irreducible τ_0 such that $\tau(x) =_R \tau_0(x)$ for all $x \in dom(\tau) = dom(\tau_0)$. So it is enough to show that $\tau_0(x) \equiv \tau'(\sigma(x))$ for some $\sigma \in CSubst$ and τ' and for all $x \in dom(\tau_0)$. If A is $u = v$ and $\tau_0(u) \equiv \tau_0(v)$ then $\mu = mgu(u, v)$ exists and τ' exists such that $\tau_0(x) \equiv \tau'(\mu(x))$ for all $x \in dom(\tau_0)$. Otherwise, since R is confluent, $\tau_0(A)$ has to be reducible. So some subterm $\tau_0(A)/p$ has to be reducible by some rule $\Gamma; \Delta \Longrightarrow l \to r$ in R. We may assume that $\mathcal{V}ar(A) \cap \mathcal{V}ar(\Gamma; \Delta \Longrightarrow l \to r) = \emptyset$. Since τ_0 is irreducible, p is a position in A and A/p is not a variable. Hence, for some $\sigma \in WCov(A)$ and some τ' we have $\tau_0(x) \equiv \tau'(\sigma(x))$ for all $x \in dom(\tau_0)$. $\square$

We give two simple examples how to apply this Lemma.

Example 7.4
a) Assume $S_0 = S = \{NAT\}$, $F_0 = F = \{0, s\}$ and $R = \emptyset$. Consider $G \equiv x = 0$, $x = s(y) \Longrightarrow$. We have

$$\begin{array}{lcll} \langle G_0 \mid m_0 \rangle & = & \langle x = 0, x = s(y) \Longrightarrow \mid \beta(x, y) \rangle & \{G_1^{m_1}\} \in I_2(G_0^{m_0}) \\ \langle G_1 \mid m_1 \rangle & = & \langle 0 = 0, 0 = s(y) \Longrightarrow \mid \beta(0, y) \rangle & \emptyset \in I_2(G_1^{m_1}) \end{array}$$

This gives $(\emptyset; G_0^{m_0}) \vdash (G_0^{m_0}; G_1^{m_1}) \vdash (G_0^{m_0}, G_1^{m_1}; \emptyset)$, so $G \in ITh(spec)$.
b) Assume $S_0 = S = \{NAT\}$, $F_0 = \{0, s\}$, $F = \{0, s, -\}$, $R_0 = \emptyset$ and

R_1: $\Longrightarrow x - 0 \to x \qquad\qquad \Longrightarrow s(x) - s(y) \to x - y.$

So we are discussing the current example of section 6.2. Consider $G \equiv def(0 - s(y)) \Longrightarrow$. We have $\emptyset \in I_2(G^m)$, so $G \in ITh(spec)$ is immediately proved.

By Lemma 7.3 one can always compute a small set $WCov(A)$ from R for applying rule I_2. It is tempting to proceed in a similar way to compute a covering $Cov(\Psi)$ for applying rule I_1. Unfortunately, this does not work in general. This fact is reflected by the next Lemma, the proof of which is given in the Appendix.

Lemma 7.5 *Let R be a conditional rewrite system. Let Ψ be a set of non-trivial atoms and let $Cov(\Psi)$ consist of all $\sigma = mgu(A/p, l)$ where $A \in \Psi$, $A/p \notin V$ and*

$\Gamma; \Delta \Longrightarrow l \rightarrow r$ *in* R. *If* $u = v$ *in* Ψ *and* $\mu = mgu(u,v)$ *exists then let* $Cov(\Psi)$ *also contain* μ. *It is undecidable whether* $Cov(\Psi)$ *is a covering for* Ψ.

Despite the result of this Lemma, one may compute $Cov(\Psi)$ as indicated and try to prove that $Cov(\Psi)$ is a covering for Ψ. Then the resulting case splitting performed by I_1 is suitable in many cases.

One may compute a small covering for Ψ as follows: If $A \in \Psi$ and $A/p \equiv t \equiv f(t_1, \ldots, t_n)$ such that $\tau(t)$ is reducible for each ground substitution τ (e.g., f is totally defined) then $Cov(\Psi)$ consisting of $\sigma = mgu(t', l)$, where t' is a non-variable subterm of t and $\Gamma; \Delta \Longrightarrow l \rightarrow r$ in R, is a complete covering for Ψ.

Let R be confluent and terminating. Let $G \equiv\Longrightarrow \Psi$ and let $Cov(\Psi)$ be as in Lemma 7.5. If Cov is not a covering for Ψ then G is not ground reducible in $R_u = \{\Longrightarrow l \rightarrow r \mid \Gamma; \Delta \Longrightarrow l \rightarrow r \text{ in } R\}$, so $Fail(G^m)$ holds for any m. By Theorem 6.13 we can conclude that $G \notin ITh(spec)$. We formulate this as a Lemma.

Lemma 7.6 *Let* R *be confluent and terminating. Assume* $G \equiv\Longrightarrow \Psi$ *and* $Cov(\Psi)$ *is as in Lemma 7.5. If* $Cov(\Psi)$ *is not a covering for* Ψ *then* $G \notin ITh(spec)$. $\square$

See Example 7.7 below for an illustration of this result.

7.2 On applying the rules I_3 and I_4

Rule I_3 is used to perform equational reasoning on the actual goal G^m using $H^{m'} \in \mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$ and the quasi-substitution σ. Rule I_4 is used to subsume G^m using $H^{m'}$ and σ. There are two problems to be solved before carrying out the inference step: (1) One has to verify that the inference step is semantically allowed. This is reflected by proving that the state $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive (see the definition of rule I_3 and rule I_4). (2) If the clause $H^{m'}$ used is not safe (i.e. $H^{m'} \notin \mathcal{L}$) then one has to verify that $\sigma(H)$ can be used as induction hypothesis. This is reflected by proving $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ for all ground substitutions τ such that $\tau(G)$ is not valid.

Here we comment on the first problem and postpone the discussion on the second problem to section 7.3. We require to prove that $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive before carrying out the inference step to ensure refutational completeness. One could also think of carrying out the inference step directly and to add $\overline{\mathcal{G}}$ to the set of unprocessed goals. This would mean to verify the goals in $\overline{\mathcal{G}}$ lazily. This modification of the inference system may be a good choice if one is interested in successful proofs only. However, it destroys refutational correctness. We give an example for that. Assume $F_0 = \{a, b, c\}$ and $F = F_0 \cup \{f, g\}$ and R

$$
\begin{array}{lrcl}
R: & & \Longrightarrow & f(a) \rightarrow a \\
& f(x) = a & \Longrightarrow & g(x) \rightarrow a \\
& f(x) = b & \Longrightarrow & g(x) \rightarrow b
\end{array}
$$

Consider the goal $G :\Longrightarrow g(a) = a$. Using the modified inference rules I_3 and I_4, there is a successful derivation[2] $(\emptyset; G) \vdash (G; \Longrightarrow a = a, \Longrightarrow f(a) = a) \vdash (G, \Longrightarrow a = a; \Longrightarrow f(a) = a) \vdash (G, \Longrightarrow a = a, \Longrightarrow f(a) = a; \emptyset)$. So $G \in ITh(spec)$ holds. But there is also a refuting derivation

$(\emptyset; G) \quad \vdash \quad (G; \Longrightarrow b = a, \Longrightarrow f(a) = b) \quad \vdash \quad (G, \Longrightarrow f(a) = b; \Longrightarrow b = a, \; a = b)$
It is easy to see that $Fail(\Longrightarrow a = b)$ is not in $ITh(spec)$. (Notice that R is confluent and terminating.) If the modified inference system would be refutational correct then we could conclude $G \notin ITh(spec)$. But this is false.

We mention that in practice the proof for $(\mathcal{H}; \overline{\mathcal{G}})$ to be inductive is often trivial. This is the case, for example, if $\overline{\mathcal{G}} = \emptyset$ or each $G^m \in \overline{\mathcal{G}}$ is subsumed by a lemma $H^{m'} \in \mathcal{L}$ or by $H^{m'} \in \mathcal{H} \cup \mathcal{G}$ and σ, where σ is a substitution and $m \succ^i \sigma(m')$. We give some examples. If $G \equiv \Phi \Longrightarrow s = t, \Psi$ and $H \equiv \Longrightarrow u = v$ is an unconditional lemma (a rule in R, for example) and σ is a substitution with $\sigma(u) \equiv s/p$, then G' can be rewritten to $G' \equiv \Phi \Longrightarrow s[\sigma(v)]_p = t, \Psi$. (This is done several times in the preceeding examples illustrating the rules I_3 and I_4.) Also, if $G \equiv \Phi \Longrightarrow \Psi$ and $H \equiv \Gamma \Longrightarrow \Delta \in ITh(spec)$ totally subsumes G, i.e., $\sigma(\Gamma) \subseteq \Phi$ and $\sigma(\Delta) \subseteq \Phi$ for some substitution σ, then I_4 is applicable to G and so G can be processed. (This is also done in the preceeding examples.)

We now specialize rule I_3 for performing contextual rewriting: The goal $G^m \equiv \langle \Phi, u = v, A[u]_p \Longrightarrow \Psi \mid m \rangle$ may be rewritten to $G'^m \equiv \langle \Phi, u = v, A[v]_p \Longrightarrow \Psi \mid m \rangle$. This can be done by rule I_3 using the lemma $H \equiv u = v \Longrightarrow u = v$ and the substitution $\sigma = id$. Since contextual rewriting is a powerful inference rule in practice we state this explictly as rule I_{cr}. It may be added to the rules I_0 to I_4.

Rule I_{cr}: Contextual rewriting
Let $G^m \equiv \langle \Phi, u = v, A[u]_p \Longrightarrow \Psi \mid m \rangle$ and let $\mathcal{G}'$ consist of $G'^m \equiv \langle \Phi, u = v, A[v]_p \Longrightarrow \Psi \mid m \rangle$ only. Then $I_{cr}(\mathcal{H}; \mathcal{G}, G^m)$ consists of all these $\mathcal{G}'$. Let $G^m \equiv \langle \Phi, u = v \Longrightarrow A[u]_p, \Psi \mid m \rangle$ and let $\mathcal{G}'$ consist of $G'^m \equiv \langle \Phi, u = v \Longrightarrow A[v]_p, \Psi \mid m \rangle$ only. Then $I_{cr}(\mathcal{H}; \mathcal{G}, G^m)$ consists of all these $\mathcal{G}'$.

One may want to avoid the necessity to eagerly evaluate the condition that $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive in rules I_3 and I_4. This is possible by modifying these rules as follows:

Rule I_3': Clausal rewriting
Let $G^m \equiv \langle \Phi, A \Longrightarrow \Psi \mid m \rangle$ be a goal. Let $H^{m'} \in \mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$ be a clause $H^{m'} \equiv \langle \Gamma \Longrightarrow u = v, \Delta \mid m' \rangle$. Let σ be a quasi-substitution such that $A/p \equiv \sigma(u)$. Let $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ if $H \in \mathcal{H} \cup \mathcal{G}$ and $\tau(G)$ is not valid. Let $\mathcal{G}'$ consist of all elements (1) $\langle \Phi, A, \sigma(B) \Longrightarrow \Psi \mid m \rangle$ with $B \in \Delta$ and (2) $\langle \Phi, A \Longrightarrow \sigma(B), \Psi \mid m \rangle$ with $B \in \Gamma$ and (3) $\langle \Phi, A \Longrightarrow B, \Psi \mid m \rangle$ with $B \in Def(H, \sigma)$ and (4) $\langle \Phi, \sigma(\Gamma), A[\sigma(v)]_p, Def(H, \sigma) \Longrightarrow \Psi, \sigma(\Delta) \mid m \rangle$. Then $I_3'(\mathcal{H}; \mathcal{G}, G^m)$ consists of all these $\mathcal{G}'$. A similar rule holds for $G^m \equiv \langle \Phi \Longrightarrow A, \Psi \mid m \rangle$.

Rule I_4': Subsumption
Let $G^m \equiv \langle \Phi_0, \Phi_1 \Longrightarrow \Psi_0, \Psi_1 \mid m \rangle$ be a goal and $H^{m'} = \langle \Gamma_0, \Gamma_1 \Longrightarrow \Delta_0, \Delta_1 \mid m' \rangle$ in $\mathcal{H} \cup \mathcal{G} \cup \mathcal{L}$. Let σ be a quasi-substitution such that $\sigma(\Gamma_0) \subseteq \Phi_0$ and $\sigma(\Delta_0) \subseteq \Psi_0$.

[2] We do not need here measure terms m.

Let $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ if $H \in \mathcal{H} \cup \mathcal{G}$ and $\tau(G)$ is not valid. Let $\mathcal{G}'$ consist of the elements (1) $\langle \Phi_0, \Phi_1, \sigma(B) \Longrightarrow \Psi_0, \Psi_1 \mid m \rangle$ with $B \in \Delta_1$ and (2) $\langle \Phi_0, \Phi_1 \Longrightarrow \sigma(B), \Psi_0, \Psi_1 \mid m \rangle$ with $B \in \Gamma_1$ and (3) $\langle \Phi_0, \Phi_1 \Longrightarrow \Psi_0, \Psi_1, B \mid m \rangle$ with $B \in Def(H, \sigma)$. Then $I_4'(\mathcal{H}; \mathcal{G}, G^m)$ consists of all these $\mathcal{G}'$.

It can be shown that these inference rules are inductively and refutationally sound.

We now assume that R is confluent and terminating. So assume $\longrightarrow_R \subseteq \succ$ for some reduction ordering $\succ$. Then one may want to restrict the search space for finding a proof for $G \in ITh(spec)$ by applying rule I_3 in a simplifying manner only: One rewrites $G \equiv \Phi \Longrightarrow \Psi$ with $H \equiv \Gamma \Longrightarrow u = v, \Delta$ and σ only if the term $\sigma(u)$ in G is replaced by $\sigma(v)$ and $\sigma(u) \succ \sigma(v)$ holds. In this case one can refine the induction ordering $\succeq_i$ to $\succeq_i'$ such that I_3 is strictly decreasing. This is done as follows: The reduction ordering $\succ$ (with $\longrightarrow_R \subseteq \succ$) defines a well-founded ordering $\succ_c$ on clauses ($\succ_c$ is the two-fold multiset extension of $\succ$). Now $\succeq_i'$ is defined by

$$\langle G \mid m \rangle \succeq_i' \langle G' \mid m' \rangle \quad \text{iff} \quad \begin{array}{l} m \succ^i m' \text{ or} \\ m \succeq^i m',\, m' \succeq^i m,\, G \succ_c G' \end{array}$$

This allows one to define a weak notion of a fair derivation (see Definition 5.9) and to define a refutational complete prover. This is done according to the construction in [Bac88]. However, in the more general setting of conditional specifications we need an undecidable failure predicate $Fail^*$, see [BL90]. (The main problem is that ground reducibility of a term is not decidable.) So we do not go into details here. But we mention that, from a practical point of view, applying rule I_3 in a simplifying manner only is highly recommended.

7.3 On the induction ordering

Recall that an induction ordering $\succeq_i$ is a well-founded ordering on the semantic units (G^m, τ). In our instantiation of the abstract prover it is based on the measure terms m in G^m only and so is independent of the clause G. We have $(G^m, \tau) \succeq_i (H^{m'}, \tau')$ iff $\tau(m) \succeq^i \tau'(m')$. We require that $\succ_i$ is compatible with R: If $\tau(x) =_R \tau'(x)$ for all $x \in \mathcal{V}ar(m)$ then $(G^m, \tau) \succeq_i (G^m, \tau')$ for any two ground substitutions τ and τ'. (This is expressed by $\tau(m) \succeq^i \tau'(m)$.)

The compatibility of R and $\succeq_i$ expresses the fact that $\succeq_i$ is a semantic ordering, i.e., it can be interpreted as a well-founded ordering on $\mathcal{A}_{spec}$. This is a standard requirement in inductive theorem provers of the Boyer-Moore style (see [BM79] and [Wal94]). It is not a standard requirement in rewrite based inductive theorem provers ([Bac88], [BR93], [Gra90], [Red90], [ZKK88]. Here the induction ordering $\succeq_i$ is strictly coupled with the reduction ordering $\succ$ satisfying $\longrightarrow_R \subseteq \succ$, i.e., with the termination ordering for R. Note that both classes of provers require R to be terminating.

We allow R to be non-terminating and so allow the induction ordering to be completely independent of the termination of R. This is one reason why we need compatibility of R and $\succeq_i$. So the question arises how to find an appropriate induction ordering or to avoid compatibility with R.

We first comment on how to deal with the compatibility requirement. The main problems come with the application of rules I_3 and I_4. Here we have to verify that $(G^m, \tau) \succ_i (\sigma(H^{m'}), \tau)$ for all ground substitutions τ such that $\tau(G)$ is not valid. Here σ may be a quasi-substitution. In general this amounts to verify $m \succ^i \sigma(m')$, which may be hard to prove. One may proceed as follows: Define a relation $\succeq^i$ on $Term(F, V)$ that can provably be interpreted as a well-founded ordering on $\mathcal{A}_{spec}$ (or an isomorphic copy of $\mathcal{A}_{spec}$). Then we have besides the equality atoms $u = v$ and the def-atoms $def(t)$ also ordering atoms of the form $s > t$. Now, to prove $m \succ^i m'$ amounts to prove an inductive theorem $\Longrightarrow m \succ m'$. This proof task can be integrated in the general prover we are just describing. This will be worked out in [Wir96].

Now we discuss how to avoid the requirement that R and $\succeq_i$ are compatible. There are some easy cases: Many inductive theorem provers allow free constructors only. So we assume that $spec_0 = (sig_0, R_0)$ with R_0 being empty. Then $\tau(x) =_R \tau'(x)$ is equivalent to $\tau(x) \equiv \tau'(x)$ for all base variables x. So, if $m \equiv \beta(t_1, \ldots, t_n)$ consists of base terms t_i only and σ is a substitution (not a quasi-substitution), then one can base the definition of $\succeq^i$ on the measure terms (as given in section 6.2) on a reduction ordering on $Term(F_0, V_0)$. Here any RPO (recursive path ordering) or any LPO (lexicographic path ordering) [DJ90] can be used. We first note that compatibility of R and $\succeq_i$ is used to guarantee that rules I_1 and I_2 decrease inductive counter-examples. From section 7.1 we learn that compatibility of R and $\succeq_i$ is not needed if we use covering sets of substitutions for $Cov(\Psi)$ to apply rule I_1 and $WCov(A)$ to apply rule I_2. In this case any ground substitution τ is directly covered: There is a $\sigma \in Cov(\Psi)$ and a τ' such that $\tau(x) \equiv \tau'(\sigma(x))$ for all $x \in dom(\tau)$. (We need the compatibility of R and $\succeq_i$ if we only have $\tau(x) =_R \tau'(\sigma(x))$ for all $x \in dom(\tau)$, i.e., if τ is only R-covered.) By a similar argument we do not need compatibility of R and $\succeq_i$ in rule I_2, if (1) R is confluent and terminating and (2) $WCov(A)$ and $Cov(\Psi)$ are computed as in the Lemmas 7.3 and 7.5. We next note that compatibility of R and $\succeq_i$ is used in rules I_3 and I_4 to verify that $G^m \succ_i \sigma(H^{m'})$, where σ may be a quasi-substitution. So we have to verify $m \succ^i \sigma(m')$. This is no problem if σ is a substitution, since only base terms have to be compared. If σ is a quasi-substitution then also non-base terms need be compared. One can avoid this in many cases as follows: We asume that $\succeq^i$ is defined on measure terms $m \equiv \beta(t_1, \ldots, t_n)$ by comparing the arguments in a lexicographic order. Then the result of comparing two measure terms may be determined before two non-base arguments are compared.

7.4 Examples

We now perform some inductive proofs to demonstrate how the inference system $\mathcal{I}$ works. We start with the example used in section 6.2 to illustrate the inference rules.

Example 7.7 *Let $spec = (sig, R)$, $spec_0 = (sig_0, R_0)$ with $S_0 = S = \{NAT\}$, $F_0 = \{0, s\}$, $F = \{0, s, +, -\}$, $R_0 = \emptyset$ and*

$$
\begin{array}{llllllllll}
R: & \Longrightarrow x + 0 & \rightarrow & x & (\rho_1) & \quad & \Longrightarrow x - 0 & \rightarrow & x & (\rho_3) \\
 & \Longrightarrow x + s(y) & \rightarrow & s(x + y) & (\rho_2) & & \Longrightarrow s(x) \rightarrow s(y) & \rightarrow & x - y & (\rho_4)
\end{array}
$$

Note that R is confluent and terminating and that the constructors (i.e., the $f \in F_0$) are free. We consider $G \equiv\Longrightarrow (x-y)+y=x$. G is not ground reducible in $R=R_u$. For example, for $\tau=\{x \leftarrow 0, y \leftarrow s(0)\}$, $\tau(G) \equiv\Longrightarrow (0-s(0))+s(0)=0$ is not reducible. (Recall that $\sigma=\{x \leftarrow 0-s(0), y \leftarrow 0\}$ is not a substitution, so the second rule of R is not applicable to $\tau(G)$.) By Theorem 6.13, $Fail(G^m)$ holds for any m and so $G \notin ITh(spec)$.

For all examples in this section we have $R_0=\emptyset$ and all measure terms $\beta(t_1,\ldots,t_n)$ contain base terms t_i only. So $=_R$ is the identity on these base terms. Hence the ordering $\succeq^i$ on the measure terms is based on an (arbitrary) reduction ordering on $Term(F_0,V_0)$. Even more, R is confluent and terminating in all examples. So we compute coverings $Cov(\Psi)$ and weak coverings $WCov(A)$ by overlapping techniques as described in section 7.1. In a positive/negative-conditional specification we omit the def-atoms for simplicity reasons. So, when giving a specification we mean its def-moderated form (see Example 7.10).

From now on we use a simplified notation to describe $\mathcal{I}$-derivations. I_1 applied to G instantiates G to several $\sigma_i(G)$. If these $\sigma_i(G)$ can immediately be simplified to G_i using unconditional rules then we write $\{G_1,\ldots,G_n\} \in I_1'(G)$. The same holds for I_2 instead of I_1. We write $I_3(G,\Gamma)$, $\Gamma \subseteq \mathcal{H} \cup \mathcal{G}$ if I_3 can be applied to G using some $H \in \Gamma$. We write $I_3^*(\Gamma)$ to denote several simplifications of G using rule I_3.

Example 7.8 *Let spec be given as in Example 7.7. Now we consider $G \equiv def(x-y) \Longrightarrow (x-y)+y=x$. Now we get*

$$
\begin{array}{ll}
G_0 \equiv \langle def(x-y) \Longrightarrow (x-y)+y=x \mid \beta(x,y)\rangle & \{G_1,G_2\} \in I_2'(G_0)\\
G_1 \equiv \langle def(x) \Longrightarrow (x-0)+0=x \mid \beta(x,0)\rangle & \{G_3\} \in I_3^*(G_1,\emptyset)\\
G_2 \equiv \langle def(x-y) \Longrightarrow (s(x)-s(y))+s(y)=s(x) \mid \beta(s(x),s(y))\rangle & \{G_4\} \in I_3^*(G_2,\emptyset)\\
G_3 \equiv \langle def(x) \Longrightarrow x=x \mid \beta(x,0)\rangle & \emptyset \in I_0(G_3)\\
G_4 \equiv \langle def(x-y) \Longrightarrow s((x-y)+y)=s(x) \mid \beta(s(x),s(y))\rangle & \{G_5\} \in\\
 & \quad I_3(G_4,\{G_0\})\\
G_5 \equiv \langle def(x-y) \Longrightarrow s(x)=s(x) \mid \beta(s(x),s(y))\rangle & \emptyset \in I_0(G_5)
\end{array}
$$

We consider the simplification of G_2 in more detail. First G_2 is simplified to $G_2' \equiv def(x-y) \Longrightarrow (x-y)+s(y)=x$ using rule ρ_4 and substitution $\sigma=id$. Then G_2' is simplified to G_3 using rule ρ_2 and the quasi-substitution $\sigma=\{x \leftarrow x-y\}$. So we have to prove that $(\mathcal{H};\overline{\mathcal{G}})=(\mathcal{H};\{def(x-y) \Longrightarrow def(\sigma(x))\})$ is inductive. But that is trivial since $\sigma(x) \equiv x-y$. Now we have $(\emptyset;\{G\}) \overset{*}{\vdash} (\mathcal{H}';\emptyset)$. That proves $G \in ITh(spec)$.

Example 7.9 *This is Example 1.2 revisited. So we assume $spec=(sig,R)$, $spec_0=(sig_0,R_0)$ with $S_0=S=\{NAT,LIST\}$, $F_0=\{0,s,nil,.\}$, $F=F_0 \cup \{push,pop,top\}$, $R_0=\emptyset$ and*

$$
\begin{array}{llcll}
R: & \Longrightarrow push(x,l) & \rightarrow & x.l & (\rho_1)\\
 & \Longrightarrow pop(x.l) & \rightarrow & l & (\rho_2)\\
 & \Longrightarrow top(x.l) & \rightarrow & x & (\rho_3)
\end{array}
$$

We consider

$$G \;\equiv\; pop(l) = nil \Longrightarrow push(top(l), pop(l)) = l$$

This gives

$$\begin{array}{lll} G_0 & \equiv & \langle pop(l) = nil \Longrightarrow push(top(l), pop(l)) = l \mid \beta(l)\rangle \quad \{G_1\} \in I_2'(G_0) \\ G_1 & \equiv & \langle l = nil \Longrightarrow push(top(x.l), pop(x.l)) = x.l \mid \beta(x.l)\rangle \quad \{G_2\} \in I_3^*(G, \emptyset) \\ G_2 & \equiv & \langle l = nil \Longrightarrow x.l = x.l \mid \beta(x.l)\rangle \quad \emptyset \in I_0(G_1) \end{array}$$

This proves $G \in ITh(spec)$.

Example 7.10 *This is Example 1.3 slightly changed. So assume* $spec(sig, R)$, $spec_0 = (sig_0, R_0)$ *with* $S_0 = \{NAT\}$, $S = \{NAT, BOOL\}$, $F_0 = \{0, s\}$, $F = \{0, s, true, false, even, odd\}$, $R_0 = \emptyset$ *and*

$$\begin{array}{rcll} R: & \Longrightarrow & even(0) \to true & (\rho_1) \\ even(x) = true & \Longrightarrow & even(s(x)) \to false & (\rho_2) \\ even(x) \neq true & \Longrightarrow & even(s(x)) \to true & (\rho_3) \\ & \Longrightarrow & odd(s(0)) \to true & (\rho_4) \\ & \Longrightarrow & odd(s^2(x)) \to odd(x) & (\rho_5) \end{array}$$

We consider

$$G \;\equiv\; \Longrightarrow even(x) = true, odd(x) = true$$

That gives

$$\begin{array}{ll} G_0 \equiv \langle \Longrightarrow even(x) = true, odd(x) = true \mid \beta(x)\rangle & \{G_1 G_2\} \in I_1(G_0) \\ G_1 \equiv \langle \Longrightarrow even(0) = true, odd(0) = true \mid \beta(0)\rangle & \{G_3\} \in I_3(G_1, \emptyset) \\ G_2 \equiv \langle \Longrightarrow even(s(x)) = true, odd(s(x)) = true \mid \beta(s(x))\rangle & \{G_4, G_5\} \in I_1(G_2) \\ G_3 \equiv \langle \Longrightarrow true = true, odd(0) = true \mid \beta(0)\rangle & \emptyset \in I_0(G_3) \\ G_4 \equiv \langle \Longrightarrow even(s(0)) = true, odd(s(0)) = true \mid \beta(s(0))\rangle & \{G_6\} \in I_3(G_4, \emptyset) \\ G_5 \equiv \langle \Longrightarrow even(s^2(x)) = true, odd(s^2(x)) = true \mid \beta(s^2(x))\rangle & \{G_7\} \in I_3(G_5) \\ G_6 \equiv \langle \Longrightarrow even(s(0)) = true, true = true \mid \beta(s(0))\rangle & \emptyset \in I_0(G_6) \\ G_7 \equiv \langle \Longrightarrow even(s^2(x)) = true, odd(x) = true \mid \beta(s^2(x))\rangle & \{G_8\} \in I_3(G_7, \mathcal{H}) \end{array}$$

provided $(\mathcal{H}; \overline{\mathcal{G}})$ *is inductive, where* $\mathcal{H} = \{G_0, \ldots, G_7\}$, $\overline{\mathcal{G}} = \{\overline{G}\}$

$$\overline{G} \equiv \langle even(s(x)) = true \Longrightarrow odd(x) = true \mid \beta(s^2(x))\rangle \qquad \{\overline{G}_1\} \in I_3(\overline{G}, \mathcal{H})$$

provided $(\mathcal{H}; \mathcal{G}')$ *is inductive, where* $\mathcal{G}' = \{G'\}$

$$G' \equiv \langle \Longrightarrow even(x) = true, odd(x) = true \mid \beta(s^2(x))\rangle \qquad \emptyset \in I_4(G', \{G_0\})$$

$$\overline{G}_1' \equiv \langle false = true \Longrightarrow odd(x) = true \mid \beta(s^2(x))\rangle \qquad \emptyset \in I_2(\overline{G}_1)$$

$$G_8 \equiv \Longrightarrow true = true, odd(x) = true \mid \beta(s^2(x))\rangle \qquad \emptyset \in I_0(G_8)$$

Note how the prover is called recursively: To apply I_3 to G_7 needs to prove that $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive. For this proof I_3 is applied to $\overline{G}$, so $(\mathcal{H}; \mathcal{G}')$ has to be proved inductive. This succeeds, so $\overline{G}_1$ can be derived as the result of applying I_3 to $\overline{G}$. This in turn gives that $(\mathcal{H}; \overline{\mathcal{G}})$ is inductive, so G_8 can be derived. This immediately concludes the proof of $(\emptyset; \{G\}) \overset{*}{\vdash} (\mathcal{H}'; \emptyset)$ and so of $P(G)$.

8 Other instantiations of the abstract prover

We now relate our abstract prover to some inductive theorem provers known in the literature. Before doing that we mention a small variation of our abstract prover:

We decrease semantic units only weakly in performing an inference step: If $\mathcal{G}' \in I(G)$, then for each G-instance (G, τ) there is $\mathcal{G}'$-instance (G', τ') such that $(G, \tau) \succeq_i (G', \tau')$. As a consequence, we can apply in rules I_3 and I_4 and induction hypotheses $\sigma(H)$ on G only if $\sigma(H)$ is stictly smaller than G, i.e. $G \succ_i \sigma(H)$. Here one can switch: If one guarantees $(G, \tau) \succ_i (G', \tau')$ than one can allow $G \succeq \sigma(H)$. See [WB94].

This modification of the abstract prover is needed to comment on the inductive prover known in the literature. These provers do not introduce explicitly measure terms as we did. We found that the introduction of measure terms gives more flexibility in designing the inference rules so that they are useful in practice. For that we needed the unswitched version of the abstract prover.

8.1 Rewrite based inductive theorem provers

We first comment on the prover described in [Bac88]. Here only unconditional and flat specifications are allowed. So we have $spec = spec_0 = (sig, R)$ and R is an unconditional rewrite system. R is required to be confluent and terminating. Only unit clauses of the form $G \equiv \Longrightarrow u = v$ are considered when proving inductive theorems.

Assume $\succ$ is a reduction ordering and $\longrightarrow_R \subseteq \succ$. Then the induction ordering $\succ_i$ is given by

$$(\Longrightarrow u = v, \tau) \succ_i (\Longrightarrow s = t, \tau) \text{ iff } \{\tau(u), \tau(v)\} \gg \{\tau(s), \tau(t)\}.$$

Here $\gg$ is the multiset extension of $\succ$. The inference rules are similar to ours (with I_2 being superfluous). Rule I_1 is realized by *overlapping rules into a goal.* So $Cov(u = v)$ is computed as described in section 7.1 for R being confluent and terminating. The difference of our rule I_1 and case splitting in [Bac88] is that after the case splitting each new goal is immediately simplified (with our rule I_3). As a consequence, all inference rules are strictly decreasing and so refutational completeness can be reached along the lines discussed in sections 5.2 and 7.2. Note that the induction ordering $\succ_i$ is a syntactic ordering and not compatible with R. The reason why no semantic ordering is needed was given at the end of section 7.3.

We now comment on the provers described in [ZKK88] and [BR93]. Here hierarchic specifications are allowed, so partiality has to be considered. But no def-atoms are allowed in goals to be proved valid. No negative conditions are allowed in the conditional equations of the specification. R is assumed to be terminating and the induction ordering $\succ_i$ is tightly coupled with the termintion ordering for R. This is done as described above. The case splitting is done by using either covering sets of substitutions or R-covering sets of substitutions. So compatibility of R and $\succ_i$ is not required. Quasi-substitutions for rules I_3 and I_4 are not considered. In [ZKK88] this is reflected by the semantics of *spec*: Only algebras with no partial functions are considered as models of *spec* (see section 1.2). So any quasi-substitution automatically is basically a substitution.

8.2 Inductive provers of the Boyer-Moore style

It is not possible to directly describe the inductive theorem provers of the Boyer-Moore style by instantiating our abstract prover. This is true, because the specifications are given algorithmically and not algebraically (as we did) and because these provers are not described by inference rules on the level we did, but by macro inferences. These macro steps are of the form "compute an induction ordering and an appropriate case splitting" or "perform induction according to the precomputed induction scheme". Very sophisticated techniques are developed to find "appropriate" induction schemes, to perform the induction step using the induction hypotheses and to generate Lemmas that may help to find a proof.

Basically speaking, these provers consist of macros which are composed of the inference rules I_0 to I_4, or variants thereof. These macros are activated by proof heuristics. To make the prover powerful (and for theoretical reasons also) some restrictions are required to hold for the specification: The constructors have to be free. All defined operators have to be totally defined. The specification is so that the evaluation of all terms terminates. Syntactic restrictions on the specification guarantee that some problems do not appear (e.g. confluence, mutual recursion).

Semantic orderings are used, both for proving the specification terminating and for defining the induction ordering. This allows one to use very powerful orderings.

A Appendix A

We here give the proofs to lemmas 6.5, 6.6 and 7.5.

Lemma 6.5 Each of the inference rules I_0 to I_4 decreases inductive counter-examples.

Proof: Let $(\mathcal{H}; \mathcal{G}, G^m)$ be a state, (G^m, τ) be an inductive counter-example for $(\mathcal{H}; \mathcal{G}, G^m)$, $0 \leq j \leq 4$ and let $\mathcal{G}' \in I_j(G^m)$. We have to show that there is a $\mathcal{G}'$-counter-example $(G_1^{m_1}, \tau')$ such that $(G^m, \tau) \succeq_i (G_1^{m_1}, \tau_1)$. Recall that$(G^m, \tau)$ is a counter-example iff $\tau(G)$ is not valid (in $\mathcal{A}_{spec}$).

We consider I_0: We have $m \equiv m_1$ for all $G_1^{m_1} \in \mathcal{G}'$. This shows $(G^m, \tau) \succeq_i (G_1^{m_1}, \tau_1)$ for all $G_1^{m_1} \in \mathcal{G}'$. If G is of the form $\Phi \Longrightarrow A, \Psi$ with $A \equiv u = u$ or $A \equiv def(t)$ and $t \in Term(F_0, V_0)$, then $\tau(G)$ holds and so (G^m, τ) is not a counter-example. The same holds if $G \equiv \Phi, A \Longrightarrow A, \Psi$. So assume $G \equiv \Phi, A \Longrightarrow \Psi$ and A is either $u = u$ or $def(t)$ with $t \in Term(F_0, V_0)$. Then $\tau(A)$ is valid and $\tau(G)$ is valid iff $\tau(G_1)$ is valid where $G_1 \equiv \Phi \Longrightarrow \Psi$. Since$(G^m, \tau)$ is a counter-example, so is (G_1^m, τ).

We consider I_1: Assume $G^m = \langle \Phi \Longrightarrow \Psi \mid m \rangle$ and $\mathcal{G}'$ is based on $Cov(\Psi)$. Then any $G_1^{m_1} \in \mathcal{G}'$ has the form $G_1^{m_1} = \langle \sigma(G) \mid \sigma(m) \rangle$ with $\sigma \in Cov(\Psi)$. Since $Cov(\Psi)$ is a covering for Ψ, there is a τ' such that $\tau(x) =_R \tau'(\sigma(x))$ for all $x \in \mathcal{V}ar(m) \subseteq \mathcal{V}ar(G)$. This shows that $(G^m, \tau) \succeq_i (G_1^{m_1}, \tau_1)$ where $G_1 \equiv \sigma(G), m_1 \equiv \sigma(m)$ and $\tau_1 = \tau'$. Now $\tau(G)$ is valid iff $\tau_1(G_1)$ is valid. Since (G^m, τ) is a counter-example, so is $(G_1^{m_1}, \tau_1)$.

We consider I_2: Assume $G^m = \langle \Phi, A \Longrightarrow \Psi \mid m \rangle$ and $\mathcal{G}'$ is based on $WCov(A)$ and a τ' such that $\tau(x) =_R \tau'(\sigma(x))$ for all $x \in \mathcal{V}ar(m)$. Define $G_1 \equiv \sigma(\Phi \Longrightarrow \Psi)$, $m_1 \equiv \sigma(m)$ and $\tau_1 = \tau'$. Then $G_1^{m_1} \in \mathcal{G}'$ and $(G^m, \tau) \succeq_i (G_1^{m_1}, \tau_1)$. Since $\tau(A)$ is valid, so is $\tau_1(A)$. Since $\tau(G)$ is not valid, so is $\tau_1(G_1)$. Hence $(G_1^{m_1}, \tau_1)$ is a counter-example.

We consider I_3: Assume $G^m = \langle G \mid m \rangle$ and $G \equiv \Phi, A \Longrightarrow \Psi$. (The case $G \equiv \Phi \Longrightarrow A, \Psi$ is similar.) Let $H^{m_0} = \langle \Gamma \Longrightarrow u = v, \Delta \mid m_0 \rangle$ be in $\mathcal{L} \cup \mathcal{H} \cup \mathcal{G}$, let σ be a quasi-substitution such that $A/p \equiv \sigma(u)$ and let $m \succ^i \sigma(m_0)$. Finally, let $\overline{\mathcal{G}}$ be given as in Rule I_3 and let $(\mathcal{H}; \overline{\mathcal{G}})$ be sound. Then $\overline{\mathcal{G}}$ consists of $G_1^m = \langle \Phi, A[\sigma(v)]_p \Longrightarrow \Psi \mid m \rangle$ only. Then $(G^m, \tau) \succeq_i (G_1^m, \tau)$. We have to show that $\tau(G_1)$ is not valid. Since $P(\mathcal{H})$ holds below (G^m, τ) and $(\mathcal{H}; \overline{\mathcal{G}})$ is sound, $\tau(\overline{G})$ is valid for all $\langle \overline{G}, \overline{m} \rangle$ in $\overline{\mathcal{G}}$. To prove that $\tau(G_1)$ is not valid assume that each $\tau(B)$ with $B \in \Phi$ and no $\tau(B)$ with $B \in \Psi$ is valid. We have to prove that $\tau(A[\sigma(v)]_p)$ is valid. By the goals (3) in $\overline{\mathcal{G}}$ there is a substitution σ' such that $\tau(\sigma(x) = \sigma'(x))$ is valid for all $x \in \mathcal{V}ar(H)$. (This is true because $def(\tau(\sigma(x)))$ is valid.) Then by the goals (1) and (2) in $\overline{\mathcal{G}}$ all $\tau(\sigma(B))$ with $B \in \Gamma$ and no $\tau(\sigma(B))$ with $B \in \Delta$ is valid. Since $m \succ^i \sigma(m_0)$ we have $(G^m, \tau) \succ_i (\sigma(H^{m_0}), \tau)$ and $P(\mathcal{H} \cup \mathcal{G})$ is valid below (G^m, τ), $(\sigma(H), \tau)$ is valid if $H \in \mathcal{H} \cup \mathcal{G}$. Clearly, if $H \in \mathcal{L}$ then $(\sigma(H), \tau)$ is valid also. So $\tau(\sigma(u) = \sigma(v))$ is valid and hence $\tau(A)$ is valid iff $\tau(A[\sigma(v)]_p)$ is valid. But $\tau(A)$ is not valid since $\tau(G)$ is not valid and all $\tau(B)$ with $B \in \Phi$ but no $\tau(B)$ with $B \in \Psi$ is valid by our assumption. So $\tau(G')$ is not valid.

We consider I_4: Assume $G^m = \langle G \mid m \rangle$ with $G \equiv \Phi_0, \Phi_1 \Longrightarrow \Psi_0, \Psi_1$ and $H^{m_0} = \langle H \mid m_0 \rangle$ with $H \equiv \Gamma_0, \Gamma_1 \Longrightarrow \Delta_0, \Delta_1$ and τ is a quasi-substitution such that $\sigma(\Gamma_0) \subseteq \Phi_0$ and $\sigma(\Delta_0) \subseteq \Psi_0$. Let $m \succ^i \sigma(m_0)$, let $\overline{\mathcal{G}}$ be as in Rule I_4 and let $(\mathcal{H}; \overline{\mathcal{G}})$ be sound. We show that no inductive counter-example (G^m, τ) for $(\mathcal{H}; G^m)$ can exist. Then we have proved that I_4 decreases inductive counter-examples.

Assume at the contrary that (G^m, τ) is an inductive counter-example. Then $P(\mathcal{H})$ holds below (G^m, τ) and $(\mathcal{H}; \overline{\mathcal{G}})$ is sound. Hence $\tau(\overline{G})$ is valid for all $\langle \overline{G} \mid \overline{m} \rangle$ in $\overline{\mathcal{G}}$. Since $\tau(G)$ is not valid, each $\tau(B)$ with $B \in \Phi_0, \Phi_1$ and no $\tau(B)$ with $B \in \Psi_0, \Psi_1$ is valid. By goals (3) in $\overline{\mathcal{G}}$ there is a substitution τ' such that $\tau(\sigma(x) = \sigma'(x))$ is valid for all $x \in \mathcal{V}ar(H)$. Then by goals (1) and (2) in $\overline{\mathcal{G}}$ each $\tau(\sigma(B))$ with $B \in \Gamma_1$ and no $\tau(\sigma(B))$ with $B \in \Delta_1$ is valid. From $\sigma(\Gamma_0) \subseteq \Phi_0$ and $\sigma(\Delta_0) \subseteq \Psi_0$ we conclude that each $\tau(\sigma(B))$ with $\Delta \in \Gamma_0$ and no $\tau(\sigma(B))$ with $B \in \Delta_0$ is valid. So $\tau(\sigma(t))$ is not valid. But for $H \in \mathcal{H} \cup \mathcal{G}$ we have $(G^m, \tau) \succ_i (\sigma(H), \tau) = (H, \tau \circ \sigma)$ and $P(\mathcal{H} \cup \mathcal{G})$ is valid below (G^m, τ), so $\tau(\sigma(H))$ is valid. Clearly, $\tau(\sigma(H))$ is valid also if $H \in \mathcal{L}$. This is a contradiction, so (G^m, τ) cannot be an inductive counter-example for $(\mathcal{H}; G)$. $\Box$

Lemma 6.6 All inference rules I_0 to I_4 are refutationally sound.

Proof: We have to prove $(\mathcal{H}; \mathcal{G}, G^m) \vdash (\mathcal{H}, G^m; \mathcal{G}, \mathcal{G}')$ and $\mathcal{G}' \in I_j(G^m)$ and $\tau(H)$ holds for all $\langle H, m_0 \rangle \in \mathcal{H} \cup \{G^m\}$ and all ground substitutions τ then $\tau(G')$ holds for all τ and all $\langle G' \mid m' \rangle \in \mathcal{G}'$.

We consider I_0: If $\mathcal{G}' = \emptyset$ then nothing is to be proved. So assume $G \equiv \Phi, A \Longrightarrow \Psi$ and A is $u = u$ or $def(t)$ with $t \in Term(F_0, V_0)$. Then $\mathcal{G}'$ consists of $\langle G' \mid m \rangle = \langle \Phi \Longrightarrow \Psi \mid m \rangle$ only. Clearly, $\tau'(A)$ is valid for all τ'. Since $\tau(G)$ is valid for all τ, $\tau(G')$ is valid vor all τ.

We consider I_1: Let $G \equiv \Phi \Longrightarrow \Psi$, let $Cov(\Psi)$ be a covering for Ψ and let $\mathcal{G}'$ be based on $Cov(\Psi)$. Any $\langle G' \mid m' \rangle \in \mathcal{G}'$ has the form $\{\sigma(G) \mid \sigma(m)\rangle$ for some $\tau \in Cov(\Psi)$. So $\tau(G')$ is valid since $\tau(\sigma(G))$ is valid.

We consider I_2: Let $G \equiv \Phi, A \Longrightarrow \Psi$ (the case $G \equiv \Phi \Longrightarrow A, \Psi$ is similar), let $WCov(A)$ be a weak covering for A and let $\mathcal{G}'$ be based on $WCov(A)$. Then any $\langle G' \mid m' \rangle \in \mathcal{G}'$ has the form $\langle \sigma(G) \mid \sigma(m) \rangle$ for some $\sigma \in WCov(A)$. So $\tau(G')$ is valid, since $\tau(\sigma(G))$ is valid.

We consider I_3: Let $G \equiv \Phi, A \Longrightarrow \Psi$, $H \equiv \Gamma \Longrightarrow u = v, \Delta$ and $\sigma, \overline{\mathcal{G}}$ as in Rule I_3. Then $\mathcal{G}'$ consist of $\langle G' \mid m \rangle$ only where $G' \equiv \Phi, A \Longrightarrow \Psi$ and $A' \equiv A[\sigma(v)]_p$. Since $(\mathcal{H}; \overline{\mathcal{G}})$ is sound and $P(\mathcal{H})$ holds, $P(\overline{\mathcal{G}})$ holds also. Let τ be any ground substitution. To prove that $\tau(G')$ holds, we assume that $\tau(B)$ holds for each $B \in \Phi$ and for no $B \in \Psi$ and prove that $\tau(A')$ holds. By the clauses (I) in $\overline{\mathcal{G}}$ there is a substitution τ' such that $\tau(\sigma(x) = \sigma'(x))$ holds for all $x \in \mathcal{V}ar(H)$. Then by the clauses (1) and (2) in $\overline{\mathcal{G}}$ $\tau(\sigma'(H))$ is applicable. Hence $\tau(A)$ holds iff $\tau(A')$ holds. But $\tau(A)$ holds since $\tau(G)$ holds and$\tau(B)$ holds for each $B \in \Phi$ and for no $B \in \Psi$. Hence $\tau(G')$ holds.

We consider I_4: Here $\mathcal{G}' = \emptyset$, so nothing has to be proved. $\Box$

Lemma 7.5 Let R be an admissible hierarchic conditional rewrite system. Let $\Psi = \{A\}$ and let $Cov(\Psi)$ consist of all $\sigma = mgu(A/p, l)$, where $A/p \notin V$ and l is the left-hand side of a rule $\Gamma; \Delta \Longrightarrow l \to r$ in. If A is $u = v$ and $\mu = mgu(u, v)$ exists, let $Cov(\Psi)$ also contain Ψ. It is undecidable whether $Cov(\Psi)$ is a covering for Ψ.

Proof: We encode Post's Correspondence Problem PCP into this problem.

The PCP over alphabet $\Sigma = \{a_1, \ldots, a_r\}$ consists of two lists of words $A = (u_1, \ldots, u_n)$ and $B = (v_1, \ldots, v_n)$ over Σ. For $t \in \{1, \ldots, n\}^+$ we define $\alpha(t)$ and $\beta(t)$ by $\alpha(i) = u_i$, $\beta(i) = v_i$, $\alpha(it) = u_i\alpha(t)$ and $\beta(it) = v_i\beta(t)$. Then $PCP(A, B)$ iff $\alpha(t) = \beta(t)$ for some $t \in \{1, \ldots, n\}^+$. It is well known that PCP is undecidable.

Given lists A and B, we construct R and an equality atom A so that $Cov(\Psi)$ is a covering for Ψ iff $PCP(A, B)$ holds: We associate to each $a_i \in \Sigma$ the unary function symbol $\overline{a}_i \in F_0$. Then to any word $w = b_1 \ldots b_n$, $b_i \in \Sigma$ we associate the term $\overline{w}(x) \equiv \overline{b}_1(\ldots(\overline{b}_n(x))\ldots)$. Now let $S_0 = \{NAT, LIST, BOOL\}$, $F_0 = \{0, 1, \ldots, n\} \cup \{\overline{a}_1, \ldots, \overline{a}_r\} \cup \{nil, ., eq, f_A, f_B, f, true, false\}$. The arities of the operators are given by $i :\rightarrow NAT$ for $i \in \{0, 1, \ldots, n\}$, $\overline{a}_j : NAT \rightarrow NAT$, $nil :\rightarrow LIST$, $. : NAT, LIST \rightarrow LIST$, $eq : NAT, NAT \rightarrow BOOL$, $f_A, f_B : LIST \rightarrow NAT$, $f : LIST \rightarrow BOOL$ and $true, false :\rightarrow BOOL$. Let $S = S_0$, $F = F_0 \cup \{g\}$ and $R = R_0 \cup R_1$ where

$$\begin{array}{ll}
R_0: & \Longrightarrow f_A(nil) \rightarrow 0 \\
 & \Longrightarrow f_B(nil) \rightarrow 0 \\
 & \Longrightarrow f_A(i.l) \rightarrow \overline{u}_i(f_A(l)) \\
 & \Longrightarrow f_B(i.l) \rightarrow \overline{v}_i(f_B(l)) \\
 & \Longrightarrow eq(x, x) \rightarrow true \\
 & x \neq y \Longrightarrow eq(x, y) \rightarrow false \\
 & \Longrightarrow f(nil) \rightarrow true \\
 & eq(f_A(i.l), f_B(i.l)) = false \Longrightarrow f(i.l) \rightarrow true \\
R_1: & \Longrightarrow g(true) \rightarrow true \\
 & \Longrightarrow g(false) \rightarrow false \\
A: & g(z) = z
\end{array}$$

For $\Psi = \{A\}$ $Cov(\Psi)$ consists of $\sigma_1 = \{z \leftarrow true\}$ and $\sigma_2 = \{z \leftarrow false\}$. It is a covering for Ψ iff f is totally defined iff $PCP(A, B)$ does not hold. Since it is undecidable whether $PCP(A, B)$ holds, it is undecidable whether $Cov(\Psi)$ is a covering for Ψ.

Acknowledgement: We thank K. Becker, B. Gramlich, U. Kühler, and C.-P. Wirth for many helpful discussions on the topic of this report. The results presented here are based to a large extent on their work, as cited in the text.

References

[AB94] J. Avenhaus and K. Becker. Operational specifications with built-ins. In *Proc. 11^{th} Annual Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 263–274. Springer-Verlag, 1994.

[AL94] J. Avenhaus and C. Loría-Sáenz. On conditional rewrite systems with extra variables and deterministi c logic programs. In *Proc. Int. Conference on Logic Programming and Automated Reasoning*, volume 822 of *Lecture Notes in Computer Science*, pages 215–229. Springer-Verlag, 1994.

[Ave95] J. Avenhaus. *Reduktionssysteme (in German).* Springer-Verlag, 1995.

[Bac88] L. Bachmair. Proof by consistency in equational theories. In *Proc. 3^{rd} Annual IEEE Symposium on Logic in Computer Science*, pages 228–233, 1988.

[Bec94] K. Becker. *Rewrite operationalization of clausal specifications with predefined structures.* PhD thesis, Univ. Kaiserslautern, 1994.

[BG94] L. Bachmair and H. Ganzinger. Rewrite-based equational teorem proving with selection and simplification. *J. Logic and Computation*, 4(3):1–31, 1994.

[BL90] E. Bevers and J. Lewi. Proof by consistency in conditional equaltional theories. In *Proc. 2^{nd} International Workshop on Conditional and Typed Rewriting Systems*, volume 516 of *Lecture Notes in Computer Science*, pages 194–205. Springer-Verlag, 1990.

[BM79] R.R. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, 1979.

[BR93] A. Bouhoula and M. Rusinowitch. Automatic case analysis in proof by induction. In *Proc. of the 13^{th} IJCAI*, pages 88–94, 1993.

[BWP84] M. Broy, M. Wirsing, and C. Pair. A systematic study of models of abstract data types. *Theoretical Computer Science*, pages 139–174, 1984.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6, pages 243–320. Elsevier, 1990.

[Gra90] B. Gramlich. Completion based inductive theorem proving: A case study in verifying sorting algorithms. SEKI-Report SR-90-04, Fachbereich Informatik, Universität Kaiserslautern, 1990.

[HH82] G.P. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25:239–266, 1982.

[Hue80] G.P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, pages 797–821, 1980.

[JK89] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82:1–33, 1989.

[Kap84] S. Kaplan. Conditional rewrite rules. *Theoretical Computer Science*, 33:175–193, 1984.

[Kap88] S. Kaplan. Positive/negative conditional rewriting. In *Proc.* 1^{st} *International Workshop on Conditional Term Rewriting Systems*, volume 308 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, 1988.

[KM86] D. Kapur and D.R. Musser. Inductive reasoning with incomplete specifications. In *Proc.* 1^{st} *Annual IEEE Symposium on Logic in Computer Science*, pages 367–377. IEEE Computer Society Press, 1986.

[KM87] D. Kapur and D. Musser. Proof by consistency. *Artificial Intelligence*, 31:125–157, 1987.

[Pad92] P. Padawitz. *Deduction and declarative programming.* Cambridge University Press, 1992.

[Red90] U.S. Reddy. Term rewriting induction. In *Proc.* $1o^{th}$ *International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 162–177. Springer-Verlag, 1990.

[Wal94] Chr. Walther. Mathematical induction. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2. Oxford University Press, 1994.

[WB94] C.-P. Wirth and K. Becker. Abstract notions and inference systems for proofs by mathematical induction. In *Proc.* 4^{th} *International Workshop on Conditional and Typed Rewriting Systems*, volume 968 of *Lecture Notes in Computer Science*, pages 353–373. Springer-Verlag, 1994.

[Wec92] W. Wechler. *Universal algebra for computer scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1992.

[WG93] C.-P. Wirth and B. Gramlich. A constructor-based approach for positive/negative conditional equational specifications. In *Proc.* 3^{rd} *International Workshop on Conditional Term Rewriting Systems*, volume 656 of *Lecture Notes in Computer Science*, pages 198–212. Springer-Verlag, 1993.

[WG94a] C.-P. Wirth and B. Gramlich. A constructor-based approach to positive/negative-conditional equational specifications. *Journal of Symbolic Computation*, 17:51–90, 1994.

[WG94b] C.-P. Wirth and B. Gramlich. On notions of inductive validity for first-order equational clauses. In *Proc.* 12^{th} *International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 162–176. Springer-Verlag, June 1994.

[Wir95] C.-P. Wirth. Syntactic confluence criteria for positive/negative-conditional term rewriting systems. SEKI-Report SR-95-09, Fachbereich Informatik, Universität Kaiserslautern, 1995.

[Wir96] C.-P. Wirth. *Positive/negative-conditional equations: A constructor based framework for specification and inductive theorem proving*. PhD thesis, Univ. Kaiserslautern, 1996.

[ZKK88] H. Zhang, D. Kapur, and M.S. Krishnamoorthy. A mechanizable induction principle for equational specifications. In *Proc. 9^{th} International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 162–181. Springer-Verlag, 1988.