Induktives Theorembeweisen für Hornklauseln

Diplomarbeit
im Fach Informatik
an der
Universität Dortmund

vorgelegt von
Jürgen Stuber

Dortmund, 4. April 1991

Inductive Theorem Proving for Horn Clauses

Jürgen Stuber

Universität Dortmund
Lehrstuhl Informatik V
Postfach 500500
W-4600 Dortmund 50
Germany
juergen@ls5.informatik.uni-dortmund.de

Master's thesis

April 4, 1991

Printed September 18, 1992

Contents

1 Introduction

Horn clauses provide a convenient language for specifications of abstract data types. Most often, the intended semantics is the initial algebra associated with the specification. It is then desirable to prove properties of the initial algebra; for instance, before it can be used as an actual parameter of some parameterized data type, the clauses in the formal parameter must be proved. In order to do this, induction is required, since most interesting properties of the initial algebra stem from the facts that (a) it is generated by ground terms, and that (b) only provable ground equations are valid in the initial algebra. (a) has traditionally been the basis of inductive proofs. (b) becomes relevant in the case of horn clauses, since the validity of a horn clause may depend on the fact that the conditions do not become true too often. In this work we study both properties separately, since (a) is typical for term generated algebras, whereas (b) is typical for free algebras. This is justified by the fact that for all four classes of models — all models, term generated models, free models and initial models — the sets of valid clauses are different.

One interesting point about (b) is that sophisticated completion procedures for horn clauses already have to take this into account. A (non-ground) completion procedure generates a set of rewrite rules representing a free algebra. In order to avoid orienting a clause into a rewrite rule, for instance because it is not reductive, it has to prove that the clause is valid anyway. At some point in this proof induction must be used; otherwise, all possible solutions of the condition will be enumerated and termination can only be achieved in trivial cases. These clauses have been called “nonoperational” in the work of Ganzinger [Gan87]. We show that this feature can be used as an explicit proof procedure for validity of clauses in the free models. The procedure is refutationally complete and linear.

We will further exploit this by showing that induction on the term structure can be replaced by induction on possible proofs of conditions. We do this via a specification transformation that enables us to express the fact that a term t is ground as the provability of an atom $\mathrm{gnd}(t)$. By adding such atoms to the conditions of a clause for all variables in the clause, the condition is provably true for ground substitutions only. In conjunction with the proof procedure for the free models we obtain a proof procedure for validity in the initial model, which is again refutationally complete and linear.

By restricting attention to the conditions of form $\mathrm{gnd}(t)$ when we prove that a transformed clause is valid in the free model, we can disable induction of type (b) on proofs of the original conditions. This corresponds to using induction on the term structure only; therefore, this variant of the procedure proves validity in the class of term generated models.

Another benefit of the specification transformation is that it enables us to express sufficient completeness of a specification. We can then use a similar procedure as above to prove this property. Sufficient completeness is important as a correctness criterium of specifications, especially for proving persistency of parameterized specifications.

Completion. Our work is based on horn clause completion, which has been developed by Ganzinger [Gan87]. In the past two years, Bachmair and Ganzinger [BG90a, BG90b], have extended it to first order clauses. Recently, Nieuwenhuis [Nie91] gives a greatly simplified version of their work; he gives a general framework for completion, together with an inference system and a simple completeness proof. We base our presentation of completion mainly on this work. Some bits and pieces not covered by Nieuwenhuis were adapted from Ganzinger and Bachmair's work; for instance, the parts on literal selection strategies and computing in complete specifications.

Proof by consistency. Our proof methods can also be viewed as "proof by consistency" methods. We say a set of clauses H is *consistent* to a specification D if the addition of H to D does not change the corresponding class of models. Then the set of clauses H is valid in this class of models. We show that this is true, whether one chooses the class of all models, term generated models, free models or initial models.

One nice property of many proof by consistency procedures is that they exhibit a particular false instance of a clause they prove to be false. Such an *inconsistency witness* is a clause which is valid in the class of models associated with $D \cup H$, but not in the class of models associated with D. If such a clause exists H is clearly not consistent to D. All proof by consistency methods work by trying to discover such a clause, although it is sometimes left implicit. A proof by consistency procedure is called *refutationally complete* if for any false clause it will eventually refute that clause.

Most proof by consistency methods in the literature have been devised for the case of unconditional equations. They vary both in their notion of inconsistency witness and in the type of inferences computed. We first show how the detection of inconsistencies evolved.

- Musser [Mus80] requires D to contain an equality predicate whose specification is sufficiently complete with respect to $\{\mathrm{true}, \mathrm{false}\}$. An inconsistency witness is an equation $\mathrm{true} \approx \mathrm{false}$.

- Huet and Hullot [HH82] require D to be *constructor complete*, meaning that the operators are partitioned into *constructors* and *defined operators* such that D is sufficienty complete with respect to constructors and no equation between constructor terms is a consequence of D. An inconsistency witness is an equation $u \approx v$ where u, v are constructur terms.

- Jouannaud and Kounalis [JK86] introduce the notion of *inductive reducibility.* A term t is inductively reducible by a rewrite system R if every ground instance of t is reducible. An inconsistency is detected if during completion a rule $l \approx r$ is generated such that $l \succ r$ and l is not inductively reducible by D. Then some ground instance $l\sigma$ is irreducible and greater than $r\sigma$; thus, $l\sigma \approx r\sigma$ is not a consequence of D. The method requires a ground convergent rewrite system R for D. Plaisted [Pla85] proves the decidability of inductive reducibility for finitely many unconditional equations.

- Bachmair [Bac88] refines the method by extending inductive reducibility to equations. An equation $u \approx v$ is inductively reducible if for any nontrivial ground instance $u\sigma \approx v\sigma$ one of $u\sigma$ or $v\sigma$ is reducible.

The methods of Musser or Huet and Hullot can be used for horn clauses without modification. However, they are undesirable since they impose severe restrictions on the specification. The methods that rely on inductive reducibility have the problem that this property is generally undecidable in the horn clause case. This can be demonstrated by a simple example (cf. [KC86]):

Example 1.1
Let D be any specification. Add new operators f and a and the clause

$$u \approx v \rightarrow \mathrm{f}(x_1, \ldots, x_n) \approx \mathrm{a}$$

where $\{x_1, \ldots, x_n\}$ are the variables occurring in $u \approx v$. Then the term $\mathrm{f}(x_1, \ldots, x_n)$ is inductively reducible if and only if $u \approx v$ is an inductive theorem of D. This is undecidable in general, for instance if we choose D to be a specification for natural numbers with addition and multiplication. $\diamond$

The other line of development was the reduction of the number of inferences that are necessary to prove an inductive theorem.

- Musser [Mus80], Huet and Hullot [HH82] and Jouannaud and Kounalis [JK86] use the unmodified completion procedure and compute all critical pairs.

- Fribourg [Fri89] notices that this is not necessary; it is sufficient to use a linear strategy where only rules in the original specification are superposed on rules in H. Moreover, not all such critical pairs need to be computed. Superpositions may be restricted to a complete position in a term t. A position is called complete if $t\sigma$ is reducible at this position for any ground substitution σ. This corresponds to the selection of an induction term in classical inductive proofs.

- Bachmair [Bac88] refines the method of Fribourg such that it is no longer necessary to orient equations in H; as a result his method does not fail. He also generalizes the notion of complete position to complete set of positions.

Numerous other people have worked on this subject [Gog80, Gra89, Gra90, HK88, KM86, KM87, KNZ86, Küc87]. Recently, there have been some approaches to the conditional case, using constructor complete specifications [Ore87] or inductive reducibility [BL90].

The proof procedure for the initial model which we present is refutationally complete and linear. Moreover, by appropriately selecting equations in the condition of a clause, superposition can be restricted to a complete set of positions.

2 Basic Notions

We give only a brief introduction to the basic notions we will need. More elaborate presentations can be found for category theory in [Mac72, HS73], for algebraic specification in [EM85, EGL89], for rewrite systems and orderings in [DJ89] and for horn clauses in [Pad88].

2.1 Preliminaries

Sequences. A^* denotes the free monoid generated by a set A, an $\bar{a} \in A^*$ is called *list*, *string*, *sequence* or *tuple*. The composition is usually denoted by commas or no operator at all. ε denotes the empty list, which will be omitted where possible. We assume $\bar{a} = a_1 \dots a_n$ if $\bar{a}$ is not defined explicitly.

Functions. The set of functions from A to B is denoted by B^A. The composition of functions f and g is denoted by $f\,;g$, where $(f\,;g)(x) = g(f(x))$.

Orderings An *ordering* is a pair $\langle S, > \rangle$ such that $>$ is an irreflexive and transitive binary relation. A *quasi-ordering* is a pair $\langle S, \geq \rangle$ such that $\geq$ is a reflexive and transitive binary relation. By abuse of notation we often just write $>$ for $\langle S, > \rangle$.

2.2 Induction

Definition 2.1 (Well-founded)
An ordering $>$ on a set S is called *well-founded* if every non-empty subset of S contains a minimal element with respect to $>$. $\Diamond$

Definition 2.2 (Induction principle)
Let $>$ be a well-founded ordering on S, and let P be some property on S. If for all x the fact that $P(y)$ holds for all y which are smaller than x implies that $P(x)$ holds then $P(x)$ holds for all x in S. $\Diamond$

In a more concise form this is expressed by the following inference rule:

Induction: $$\frac{\forall x.\ (\forall y.\, y < x \ \Rightarrow\ P(y)) \ \Rightarrow\ P(x)}{\forall x.\, P(x)}$$

We will use this principle of induction heavily. A proof using it will start with "let x be a minimal element in S such that $P(x)$ does not hold" and derive a contradiction by showing that there exists an even smaller y such that $P(y)$ does not hold.

2.3 Multisets

A *multiset* (over a set S) is a function $M : S \to \mathbf{N}$. The set of multisets over S is denoted by $\mathbf{N}^S$. The union and intersection of multisets M and N are defined by $(M \cup N)(x) = M(x) + N(x)$ and $(M \cap N)(x) = \min(M(x), N(x))$ for all $x \in S$, respectively. We say x is an *element* of M and write $x \in M$ if $M(x) > 0$. Intuitively, a multiset is like a set but elements may occur more than once in the multiset. For any element x the number of occurrences is specified by $M(x)$. Therefore we treat multisets like sets and for example write $M = \{x, x, x\}$ to denote the multiset defined by $M(x) = 3$ and $M(y) = 0$ for all $y \neq x$.

If $>$ is an ordering on a set S then $>$ can be extended to an ordering $>_{mul}$ on finite[1] multisets over S as follows. $M >_{mul} N$ if and only if

1. $M \neq N$, and
2. whenever $N(x) > M(x)$ there exists an $y > x$ such that $M(y) > N(y)$.

Lemma 2.3 *Let $>$ be an ordering on a set S and let $>_{mul}$ be the multiset extension of $>$.*

1. *If $>$ is total then $>_{mul}$ is total.*
2. *If $>$ is well-founded then $>_{mul}$ is well-founded.*
3. *For any monotonic function $f : \langle S_1, >_1\rangle \to \langle S_2, >_2\rangle$ the multiset extension*

$$f_{mul} : \langle \mathbf{N}^{S_1}, (>_1)_{mul}\rangle \to \langle \mathbf{N}^{S_2}, (>_2)_{mul}\rangle$$

is monotonic.

2.4 Categories

Definition 2.4 (Category)
A *category* is a sextuple $\mathcal{C} = \langle \mathcal{C}_{\mathrm{Obj}}, \mathcal{C}_{\mathrm{Mor}}, \mathrm{dom}, \mathrm{cod}, \_\,;\_, \mathrm{id}\rangle$ where

1. $\mathcal{C}_{\mathrm{Obj}}$ is a class whose members are called *objects*;
2. $\mathcal{C}_{\mathrm{Mor}}$ is a class whose members are called *morphisms*;
3. $\mathrm{dom} : \mathcal{C}_{\mathrm{Mor}} \to \mathcal{C}_{\mathrm{Obj}}$ is a function; $\mathrm{dom}(f)$ is called the *domain* of f;
4. $\mathrm{cod} : \mathcal{C}_{\mathrm{Mor}} \to \mathcal{C}_{\mathrm{Obj}}$ is a function; $\mathrm{cod}(f)$ is called the *codomain* of f;
5. $\mathrm{id} : \mathcal{C}_{\mathrm{Obj}} \to \mathcal{C}_{\mathrm{Mor}}$ is a function; id_A is called the *identity (morphism)* on A;
6. $\_\,;\_$ is a partial function $\mathcal{C}_{\mathrm{Mor}} \times \mathcal{C}_{\mathrm{Mor}} \to \mathcal{C}_{\mathrm{Mor}}$ called *composition*.

For any object A in $\mathcal{C}_{\mathrm{Obj}}$ and morphisms f and g in $\mathcal{C}_{\mathrm{Mor}}$ the following must hold:

1. $f\,;g$ is defined if and only if $\mathrm{cod}(f) = \mathrm{dom}(g)$.
2. If $f\,;g$ is defined then $\mathrm{dom}(f\,;g) = \mathrm{dom}(f)$ and $\mathrm{cod}(f\,;g) = \mathrm{cod}(g)$.
3. If $f\,;g$ and $g\,;h$ are defined then $f\,;(g\,;h)$ and $(f\,;g)\,;h$ are defined, and $f\,;(g\,;h) = (f\,;g)\,;h$.
4. $\mathrm{dom}(\mathrm{id}_A) = \mathrm{cod}(\mathrm{id}_A) = A$.
5. If $\mathrm{id}_A\,;f$ is defined then $\mathrm{id}_A\,;f = f$.
6. If $g\,;\mathrm{id}_A$ is defined then $g\,;\mathrm{id}_A = g$.

We will write $f\,;g = h$, meaning "the composition $f\,;g$ is defined and equal to h". $\diamond$

[1] For multisets containing infinite ascending chains $x_0 < x_1 < x_2 < \ldots$, $>_{mul}$ is only a quasi-ordering. For example, in the multiset ordering the set of even numbers is simultaneously smaller and greater than the set of odd numbers.

To ensure that the multiset extension of a well-founded ordering is well-founded, we must restrict the ordering to finite multisets. For any infinite multiset $\{x_0, x_1, x_2, \ldots\}$ we would get the infinite descending chain $\{x_0, x_1, x_2, \ldots\} >_{mul} \{x_1, x_2, \ldots\} >_{mul} \{x_2, \ldots\} >_{mul} \cdots$

For instance, SET is the category consisting of sets as objects and functions as morphisms.

Definition 2.5 (Functor)
Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A *functor* $F : \mathcal{C} \to \mathcal{D}$ is a pair $\langle F_{\mathrm{Obj}}, F_{\mathrm{Mor}} \rangle$, where

1. $F_{\mathrm{Obj}} : \mathcal{C}_{\mathrm{Obj}} \to \mathcal{D}_{\mathrm{Obj}}$ is a function;
2. $F_{\mathrm{Mor}} : \mathcal{C}_{\mathrm{Mor}} \to \mathcal{D}_{\mathrm{Mor}}$ is a function.

We write $F(A)$ for $F_{\mathrm{Obj}}(A)$ and $F(f)$ for $F_{\mathrm{Mor}}(f)$. For any object A in $\mathcal{C}_{\mathrm{Obj}}$ and morphisms f and g in $\mathcal{C}_{\mathrm{Mor}}$ the following must hold.

1. $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$.
2. $F(f\,;g) = F(f)\,;F(g)$. $\diamond$

Definition 2.6 (Initial object)
Let $\mathcal{C}$ be a category. An object I in $\mathcal{C}_{\mathrm{Obj}}$ is called *initial* (in $\mathcal{C}$) if for any object A in $\mathcal{C}_{\mathrm{Obj}}$ there exists exactly one morphism $\mathrm{init}_A : I \to A$. $\diamond$

Definition 2.7 (Free object)
Let $\mathcal{C}$ be a category, let $V : \mathcal{C} \to \mathrm{SET}$ be a functor and let X be a set. An object F in $\mathcal{C}_{\mathrm{Obj}}$ is called *free over* X (in $\mathcal{C}$) if there exists a morphism $\eta : X \to V(F)$ such that for any object A in $\mathcal{C}$ and morphism $f : X \to V(A)$ there exists a unique morphism $f^\# : F \to A$ such that the diagram

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[dr, "f"'] & V(F) \arrow[d, "V(f^\#)"] & F \arrow[d, "f^\#"] \\
 & V(A) & A
\end{tikzcd}$$

commutes. $\diamond$

2.5 Sorted Sets

Let S be a set of *sorts*. An *S-sorted set* is an S-indexed family of sets $A = (A_s)_{s \in S}$. We say that elements of A_s *have sort* s. A is extended to an S^*-sorted set by defining $A_{\bar{s}} = A_{s_1} \times \ldots \times A_{s_n}$. Sometimes we will also use A to denote the set $\bigcup_{s \in S} A_s$.

Let A, B be S-sorted sets. An *S-sorted binary relation* $R \subseteq A \times B$ is an S-indexed family of binary relations $R = (R_s \subseteq A_s \times B_s)_{s \in S}$. R is extended to an S^*-sorted binary relation by defining $\bar{a}\ R_{\bar{s}}\ \bar{b}$ if and only if $a_1\ R_{s_1}\ b_1 \wedge \ldots \wedge a_n\ R_{s_n}\ b_n$.

An *S-sorted function* $f : A \to B$ is an S-indexed family of functions $(f_s : A_s \to B_s)_{s \in S}$. f is extended to an S^*-sorted function by defining $f_{\bar{s}}(\bar{a}) = f_{s_1}(a_1) \ldots f_{s_n}(a_n)$.

2.6 Specifications

Definition 2.8 (Signature)
A *signature* is a pair $\Sigma = \langle S, \Omega \rangle$ where S is a set of *sorts* and $\Omega = (\Omega_{\bar{s}s})_{\bar{s} \in S^*, s \in S}$ is an $S^* \times S$-sorted set of *operators* We write $\omega : \bar{s} \to s$ for $\omega \in \Omega_{\bar{s}s}$. $\bar{s}$ is called *arity* of ω and s is called *coarity* or *range* of ω. $\diamond$

Definition 2.9 (Variables)
Let $X = (X_s)_{s \in S}$ be an S-sorted set of *variables*. The signature $\Sigma(X) = \langle S, \Omega(X) \rangle$ contains the variables in X as constants, i.e., $\Omega(X)_s = \Omega_{\varepsilon s} \cup X_s$ and $\Omega(X)_{\bar{s}s} = \Omega_{\bar{s}s}$ for nonempty $\bar{s}$. From now on, we will assume that for any signature Σ there exists a fixed set of variables X such that X_s is countably infinite for all sorts s in S. $\diamond$

Definition 2.10 (Terms)
The set of *terms over* $\Sigma(X)$ is the smallest S-sorted set $T_{\Sigma(X)} = (T_{\Sigma(X),s})_{s \in S}$ such that $X \subseteq T_{\Sigma(X)}$ and the term $\omega(\bar{t})$ is in $T_{\Sigma(X),s}$ whenever ω is an operator in $\Omega_{\bar{s}s}$ and $\bar{t}$ are terms in $T_{\Sigma(X),\bar{s}}$. For the constant term $\omega()$ we simply write ω. $\diamond$

Definition 2.11 (Equation, clause)
A Σ-*equation* (over X) is a multiset $\{u, v\}$, written $u \approx v$, where u and v are terms in $T_{\Sigma(X),s}$, and are of the same sort s.

A *(horn) clause* is a pair of a multiset of equations Γ and a multiset Δ, written $\Gamma \to \Delta$, which may either be empty or contain one equation $u \approx v$. We call Γ the *antecedent* and Δ the *succedent* of the clause. A clause with empty succedent is called *goal clause*, a clause with empty antecedent and one equation in the succedent succedent is called *unit clause*. We write $\varphi_1, \ldots, \varphi_n$ for a multiset of equations $\{\varphi_1, \ldots, \varphi_n\}$ and Γ_1, Γ_2 for $\Gamma_1 \cup \Gamma_2$. Clauses of the form $\Gamma, u \approx v \to u \approx v$ or $\Gamma \to t \approx t$ are called *tautologies*. The clause where both antecedent and succedent are empty is called the *empty clause*. It indicates a contradiction. $\diamond$

Definition 2.12 (Ground term)
var(t) is the S-sorted set of variables in a term t. A term t is called *ground* if *var(t)* is empty. Analogously for equations and clauses. T_Σ denotes the S-sorted set of ground terms over Σ. $\diamond$

Definition 2.13 (Inhabited)
We say a signature is *inhabited*, if for any sort s in S there exists a ground term in $T_{\Sigma,s}$. We assume from now on that all signatures are inhabited. $\diamond$

Definition 2.14 (Specification)
A *specification* is a pair $D = \langle \Sigma, N \rangle$, where Σ is a signature and N is a set of clauses. We will also use D to denote the set of clauses in D to avoid notational overhead. $\diamond$

2.7 Algebras

Definition 2.15 (Algebra)
A Σ-*algebra* A is a pair $\langle S_A, \Omega_A \rangle$ where S_A is an S-sorted set called *carrier* of A and Ω_A is a family of sets of functions. Ω_A contains a function $\omega_A : \bar{s}_A \to s_A$ for each operator $\omega : \bar{s} \to s$ in Ω. We write s_A for $S_{A,s}$ and $\bar{s}_A$ for $s_{1,A} \times \cdots \times s_{n,A}$.

Let A and B be Σ-algebras. A Σ-*algebra morphism* $f : A \to B$ is an S-sorted function such that $f_s(\omega_A(\bar{a})) = \omega_B(f_{\bar{s}}(\bar{a}))$ for all operators $\omega : \bar{s} \to s$ in Ω and tuples of elements $\bar{a}$ in $\bar{s}_A$.

The category $\Sigma-$ALG consists of Σ-algebras and Σ-algebra morphisms.

A *subslgebra* of A is a Σ-algebra B such that $s_B \subseteq s_A$ for all sorts S in S, and $\omega_B(\bar{b}) = \omega_A(\bar{b})$ for all operators ω and elements $\bar{b}$ in $\bar{s}_B$. $\diamond$

Definition 2.16 (Assignment)
An *assignment* is an S-sorted function $\alpha : X \to A$. The *evaluation function* in A with respect to α is an S-sorted function $\alpha^{\#} : T_{\Sigma(X)} \to A$ with

$$\begin{aligned} \alpha^{\#}_s(\omega(\bar{t})) = \omega_A(\alpha^{\#}_{\bar{s}}(\bar{t})) \qquad & \text{for } \omega : \bar{s} \to s,\ \bar{t} \in T_{\Sigma(X),\bar{s}} \\ \alpha^{\#}_s(x) = \alpha(x) \qquad & \text{for } x \in X_s. \end{aligned}$$

We write $\#_A(t)$ for $\alpha^{\#}(t)$ if t is a ground term. ◇

Definition 2.17 (Term generated)
Let a be an element in s_A for some sort s in S. a is called *term generated* if there exists a ground term t in T_Σ such that $\#_A(t) = a$. An algebra A is called *term generated*, if all elements of A are term generated. The subalgebra of A consisting of all term generated elements of A is denoted by A_{gen}. ◇

2.8 Validity

Definition 2.18 (Satisfy, valid)
An assignment $\alpha : X \to A$ *satisfies* an equation $u \approx v$, written $A \models_\alpha u \approx v$, if $\alpha^{\#}(u) = \alpha^{\#}(v)$. α satisfies a clause $\Gamma \to \Delta$ if it either does not satisfy all equations in Γ or it satisfies at least one equation in Δ. An equation $u \approx v$ is *valid* in A, written $A \models u \approx v$, if all assignments $\alpha : X \to A$ satisfy $u \approx v$ in A. Analogously for clauses.

Let $\mathcal{D}$ be a class of Σ-algebras. If an equation $u \approx v$ is valid in all algebras in $\mathcal{D}$ we write $\mathcal{D} \models u \approx v$. Analogously for clauses. ◇

Definition 2.19 (Model)
Let N be a set of clauses. A Σ-algebra A is a *model of* N, written $A \models N$, if A satisfies all clauses in N.

Let $D = \langle \Sigma, N \rangle$ be a specification. A Σ-algebra A is a *model of* D or D*-algebra*, written $A \models D$, if A is a model of N. The class of all D-models is denoted by $\mathrm{Mod}(D)$ and the set of all term generated D-models is denoted by $\mathrm{Gen}(D)$.

We may regard $\mathrm{Mod}(D)$ as the category consisting of the models of D together with the Σ-algebra morphisms between them, and $\mathrm{Gen}(D)$ as the category consisting of the term generated models of D together with the morphisms between them. ◇

Definition 2.20 (Consequence)
For $\mathrm{Mod}(D) \models u \approx v$ we write $D \models u \approx v$ and say $u \approx v$ is a (logical) consequence of D, and for $\mathrm{Gen}(D) \models u \approx v$ we write $D \models_I u \approx v$ and say $u \approx v$ is an inductive consequence of D. We use the same notation if instead of $u \approx v$ we have a clause, a set of clauses or a specification. We say two specifications D and D' are *equivalent* if and only if $D \models D'$ and $D' \models D$, and *inductively equivalent* if and only if $D \models_I D'$ and $D' \models_I D$. ◇

Definition 2.21 (Inconsistent)
A specification D is called *inconsistent* if $\mathrm{Mod}(D)$ is empty. Otherwise, D is called *consistent*. ◇

The following lemma shows that for clauses there is only one notion of inconsistency, whether we require the existence of a term generated model or just any model.

Lemma 2.22 *If there exists a model of D then there exists a term generated model of D.*

Proof: The subalgebra A_{gen} of A is a model of D. □

Lemma 2.23 *Let D be a specification and let $u \approx v$ be a ground equation. $D \models u \approx v$ if and only if $D \cup \{u \approx v \rightarrow\}$ is inconsistent.*

Proof: Suppose A is a model of D. A satisfies the antecedent of $u \approx v \rightarrow$, but since the succedent is empty the clause cannot be satisfied. Therefore, the clause $\{u \approx v \rightarrow\}$ is not valid in A. Conversely, let A be any model of D. We know that the clause $u \approx v \rightarrow$ is not true in A; therefore, A satisfies the equation $u \approx v$. □

2.9 Term Models

Definition 2.24 (Term algebra)
The Σ-*term algebra* $T_{\Sigma(X)}$ is an algebra which has the Σ-terms of sort S as carrier and where $\omega_{T_{\Sigma(X)}}(\bar{t}) = \omega(\bar{t})$ for all operators $\omega : \bar{s} \rightarrow s$. ◇

Definition 2.25 (Substitution)
Let X be S-sorted sets of variables. A *substitution* is an assignment $\sigma : X \rightarrow T_{\Sigma(X)}$. The application of a substitution σ to a term t in $T_{\Sigma(X)}$ is defined as $t\sigma = \sigma^{\#}(t)$. The application of σ to equations, multisets of equations and clauses is defined by $(u \approx v)\sigma = u\sigma \approx v\sigma$, $\{\varphi_1, \ldots, \varphi_n\}\sigma = \{\varphi_1\sigma, \ldots, \varphi_n\sigma\}$ and $(\Gamma \rightarrow \Delta)\sigma = \Gamma\sigma \rightarrow \Delta\sigma$, respectively. ◇

We will give concrete substitutions in the form $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$. Then σ is defined by $\sigma(x_i) = t_i$ and $\sigma(y) = y$ for all variables y not in $\{x_1, \ldots, x_n\}$.

Definition 2.26 (Subsumption ordering)
We define a quasi-ordering $\trianglerighteq$ on substitutions by $\sigma \trianglerighteq \sigma'$ if and only if there exists a substitution τ such that $\sigma = \sigma'\tau$, and an ordering $\triangleright$ by $\sigma \triangleright \sigma'$ if and only if $\sigma \trianglerighteq \sigma'$ but not $\sigma' \trianglerighteq \sigma$.

In the same way $\trianglerighteq$ and $\triangleright$ are defined for terms, equations and clauses instead of substitutions. $\triangleright$ is called the *subsumption ordering.* ◇

Lemma 2.27 *$\triangleright$ is well founded.*

Definition 2.28 (Unifier, most general unifier)
Let u and v be terms. A *unifier* of u and v is a substitution μ such that $u\mu = v\mu$. A *most general unifier* of u and v is a substitution μ that is minimal with respect to the subsumption ordering $\triangleright$ in $\{\, \mu \mid \mu$ is a unifier of u and $v \,\}$. ◇

Lemma 2.29 (Substitution lemma) *Let A be a Σ-algebra, let $f : T_{\Sigma(X)} \rightarrow A$ be a Σ-algebra morphism, let t be a term in $T_{\Sigma(X)}$ and let $\sigma : X \rightarrow T_{\Sigma(X)}$ be a substitution. Then $f(t\sigma) = (\sigma\,;f)^{\#}(t)$.*

Proof: The proof is by induction on the term structure. If t is a variable x in X we have $f(x\sigma) = (\sigma\,;f)(x) = (\sigma\,;f)^{\#}(x)$. Otherwise t is a term $t = \omega(\bar{t})$. Then $f(\omega(\bar{t}\sigma)) = \omega_A(f(\bar{t}\sigma)) = \omega_A((\sigma\,;f)^{\#}(\bar{t})) = (\sigma\,;f)^{\#}(\omega(\bar{t}))$. □

Definition 2.30 (Congruence)
Let $\Sigma = \langle S, \Omega \rangle$ be a signature, let A be a Σ-algebra, and let $\sim$ be an S-sorted binary relation on A. $\sim$ is called *compatible with operators* if $\bar{a} \sim_{\bar{s}} \bar{b}$ implies $\omega_A(\bar{a}) \sim_s \omega_A(\bar{b})$ for all operators ω in $\Omega_{\bar{s}s}$ and all tuples of elements $\bar{a}$ and $\bar{b}$ in $\bar{s}_A$. A Σ*-congruence on* A is an S-sorted equivalence relation on A which is compatible with operators. $\diamond$

Definition 2.31 (Quotient algebra)
Let A be a algebra and let $\sim$ be a Σ-congruence on A. The *quotient algebra* $A/\equiv$ *of* A *by* $\equiv$ is defined by

$$\begin{aligned} s_{A/\equiv} &= s_A/\equiv_s \\ \omega_{A/\equiv}([\bar{a}]) &= [\omega_A(\bar{a})] \end{aligned}$$

$[\_] : A \to A/\equiv$ is called the *natural morphism.* $\diamond$

Definition 2.32
A specification $D = \langle \Sigma, N \rangle$ induces a congruence $\equiv_N$ on $T_{\Sigma(X)}$ defined by $u \equiv_N v$ if and only if $D \models u \approx v$ for terms u and v in $T_{\Sigma(X)}$. We define T_D as $T_\Sigma/\equiv_N$ and $T_{D(X)}$ as $T_{\Sigma(X)}/\equiv_N$. $\diamond$

Theorem 2.33 *Let D be a consistent specification. T_D is initial in* $\mathrm{Mod}(D)$ *and $T_{D(X)}$ is free over X in* $\mathrm{Mod}(D)$.

Proof: Ehrig and Mahr [EM85] give proofs for these. $\square$

Since all initial models are isomorphic, we speak of T_D as *the* initial model. The same applies to *the* free model $T_{D(X)}$. We will usually omit the "over X", because we assume a fixed set of variables.

Definition 2.34 (Solution)
A substitution $\sigma : X \to T_{D(X)}$ is called a *solution* of $u \approx v$ in D, if $D \models (u \approx v)\sigma$. $\diamond$

Lemma 2.35 *Let $\alpha : X \to T_{D(X)}$ be an assignment. Then there exists a substitution $\sigma : X \to T_{\Sigma(X)}$ with $\alpha = \sigma\,;[\_]$. For any such σ we have $T_{D(X)} \models_\alpha u \approx v$ if and only if $D \models (u \approx v)\sigma$.*

Proof: Let $X = \{x_1, \ldots, x_n\}$. We first show that such a σ exists for any α. Since $[\_]$ is surjective, there exist terms t_i in $T_{\Sigma(X)}$ such that $\alpha(x_i) = [t_i]$ for $1 \leq i \leq n$. Define σ by $\sigma(x_i) = t_i$, then we have $\alpha = \sigma\,;[\_]$.

Using the substitution lemma we get $\alpha^{\#}(t) = (\sigma\,;[\_])^{\#}(t) = [t\sigma]$ for any term t in $T_{\Sigma(X)}$. Hence $T_{D(X)} \models_\alpha u \approx v$ implies $\alpha^{\#}(u) = \alpha^{\#}(v)$, $[u\sigma] = [v\sigma]$, $u\sigma \equiv_D v\sigma$ and $D \models u\sigma \approx v\sigma$. $\square$

2.10 Hierarchical Specifications

Definition 2.36 (Signature morphism)
Let $\Sigma_1 = \langle S_1, \Omega_1 \rangle$ and $\Sigma_2 = \langle S_2, \Omega_2 \rangle$ be signatures. A *signature morphism* $\delta : \Sigma_1 \to \Sigma_2$ is a pair of functions $\langle \delta_S, \delta_\Omega \rangle$ where $\delta_S : S_1 \to S_2$ is a function and

$$\delta_\Omega = (\delta_{\Omega,\bar{s}s} : \Omega_{\bar{s}s} \to \Omega_{\delta(\bar{s})\delta(s)})_{\bar{s} \in S_1^*, s \in S_1}$$

We write $\delta(s)$ for $\delta_S(s)$, $\delta(\bar{s})$ for $\delta(s_1)\dots\delta(s_n)$ and $\delta(\omega)$ for $\delta_{\Omega,\bar{s}s}(\omega)$. Signatures and signature morphisms form a category SIGN. $\diamond$

A signature morphism translates terms, clauses and specifications in the obvious way by translating sorts and operators.

Definition 2.37 (Specification morphism)
Let $D_1 = \langle \Sigma_1, N_1 \rangle$ and $D_2 = \langle \Sigma_2, N_2 \rangle$ be specifications. A *specification morphism* $\delta : D_1 \to D_2$ is a signature morphism $\delta : \Sigma_1 \to \Sigma_2$ with $N_2 \models \delta(N_1)$. The category SPEC has specifications as objects and specification morphisms as morphisms. $\diamond$

Definition 2.38 (Subsignature, subspecification)
A *subsignature* of a signature $\Sigma = \langle S, \Omega \rangle$ is a signature $\Sigma' = \langle S', \Omega' \rangle$ such that $S' \subseteq S$ and $\Omega'_{\bar{s}s} \subseteq \Omega_{\bar{s}s}$ for all sorts $\bar{s} \in S'^*$ and $s \in S'$. There exists a signature morphism $\delta : \Sigma' \to \Sigma$ which is called a *signature inclusion*.

A *subspecification* of a specification $D = \langle \Sigma, N \rangle$ is a specification $D' = \langle \Sigma', N' \rangle$ such that Σ' is a subsignature of Σ and $N' \subseteq N$. There exists a specification morphism $\delta : D' \to D$ which is called a *specification inclusion*. $\diamond$

Definition 2.39 (Complete, consistent, persistent specification morphism)
Let $\delta : D_1 \to D_2$ be a specification morphism.

1. We say δ is *complete* if for any sort s in S and term t in $T_{\Sigma_2(X),s}$ there exists a term t' in $T_{\Sigma_1(X),s}$ such that $D_2 \models t \approx t'$.
2. We say δ is *consistent* if for any sort s in S and terms u and v in $T_{\Sigma_1(X),s}$ $D_2 \models \delta(u \approx v)$ implies $D_1 \models u \approx v$.
3. We say δ is *persistent* if it is complete and consistent. $\diamond$

Definition 2.40 (Reduct)
Let $\delta : \Sigma_1 \to \Sigma_2$ be a signature morphism. The *δ-reduct* $A|_\delta$ of a Σ_2-algebra A is defined by

$$s_{A|_\delta} = \delta(s)_A$$
$$\omega_{A|_\delta} = \delta(\omega)_A$$

for all sorts s and operators ω in Σ_1. The *δ-reduct* $f|_\delta$ of a Σ_2-algebra morphism $f : A \to B$ is defined by

$$f|_\delta = (f_{\delta(s)})_{s \in S_1}.$$

The *reduct functor* with respect to δ is $V_\delta : \Sigma_2{-}\mathrm{ALG} \to \Sigma_1{-}\mathrm{ALG}$. It maps algebras and algebra morphisms on their respective reducts. $\diamond$

Definition 2.41 (Extension, complete, consistent, persistent, strongly persistent)
Let $\delta : D_1 \to D_2$ be a specification morphism, let A be a D_1-algebra, let B be a D_2-algebra and let $\eta : A \to B|_\delta$ be a D_1-algebra morphism. Then $\langle B, \eta \rangle$ is called *δ-extension* of A. The extension is called *complete* if η is surjective, *consistent* if η is injective, *persistent* if η is bijective and *strongly persistent* if η is the identity on A. $\diamond$

2.11 Canonical Rewrite Systems

Definition 2.42 (Rewrite relation)
A relation R on terms is *stable under contexts* if $R(\bar{t})$ implies $R(u[\bar{t}])$ for all contexts u and tuples of terms $\bar{t}$, and *stable under substitutions* if $R(\bar{t})$ implies $R(\bar{t}\sigma)$ for all substitutions σ and tuples of terms $\bar{t}$. A binary relation $\Rightarrow$ on terms is called a *rewrite relation* if it is stable under contexts and substitutions. $\diamond$

Definition 2.43 (Reflexive-transitive closure, symmetric closure, Church-Rosser)
Let $\Rightarrow$ be a binary relation. By $\overset{*}{\Rightarrow}$ we denote the *reflexive-transitive closure* of $\Rightarrow$ and by $\Leftrightarrow$ the *symmetric closure*. We define the relation $\Downarrow$ by $u \Downarrow v$ if and only if there exists a term t such that $u \overset{*}{\Rightarrow} t$ and $v \overset{*}{\Rightarrow} t$. $\Rightarrow$ is called *Church-Rosser* if the relations $\overset{*}{\Leftrightarrow}$ and $\Downarrow$ are equal. $\diamond$

Definition 2.44 (Reduction ordering)
A *reduction ordering* is a transitive, well-founded rewrite relation $\succ$. $\diamond$

Since we need to order terms of distinct sorts, $\succ$ is not an S-sorted relation but a binary relation on $\bigcup_{s \in S} T_{\Sigma(X),s}$.

In our examples we will use the *lexicographic path ordering (lpo)* as a particular instance of a reduction ordering. Deshowitz and Jouannaud [DJ89] give a definition.

Definition 2.45 (Rewrite system)
A set of equations R is a *rewrite system* with respect to a reduction ordering $\succ$ if for all equations $u \approx v$ in R either $u \succ v$ or $v \succ u$. Equations in R are called *rules*, and for $u \succ v$ we will always write u on the left and v on the right side of the equation. For such an R we denote by $\Rightarrow_{R^\succ}$ (or $\Rightarrow_R$) the smallest rewrite relation such that $u \Rightarrow_R v$ whenever $u \approx v$ is a rule in R and $u \succ v$. We say a term t is *reducible* by R if there exists a term t' such that $t \Rightarrow_R t'$, otherwise we say t is *irreducible*. $\diamond$

Lemma 2.46 *If $u \Downarrow v$ and $u \succ v$ then u is reducible.*

Definition 2.47 (Canonical)
A rewrite system R is called *canonical* if the corresponding rewrite relation $\Rightarrow_R$ is well founded and Church-Rosser. $\diamond$

Lemma 2.48 *Let R be a ground rewrite system with respect to some reduction ordering $\succ$. If there are no two distinct equations $l \approx r$ and $u \approx v$ in R such that $l \succ r$ and $u \succ v$ and l is a subterm of u then R is canonical.*

2.12 Linear Paramodulation

Definition 2.49 (Linear paramodulation)
Let D be a specification. The inference system for linear paramodulation with respect to D has the following inference rules:

Linear paramodulation: $$\frac{\Gamma, u[l\sigma] \approx v \rightarrow}{\Lambda\sigma, \Gamma, u[r\sigma] \approx v \rightarrow}$$

where $\Lambda \to l \approx r$ is a clause in D and σ is a substitution.

Equality resolution: $$\frac{\Gamma, u \approx u \to}{\Gamma \to}$$

A linear paramodulation refutation with respect to D is a sequence G_0, G_1, G_2, $\ldots$ $\Box$ of goal clauses such that G_{i+1} is obtained from G_i by linear paramodulation or equality resolution. $\Diamond$

Lemma 2.50 *Let D be a specification and let $u \approx v$ be an equation. $D \models u \approx v$ if and only if there exists a linear paramodulation refutation with respect to D starting from $u \approx v \to$.*

Proof: The rules are obviously sound provided Σ is inhabited. Completeness results can be found in several textbooks [Pad88, Höl89]. $\Box$

2.13 Predicates

We will represent predicates as operators. For any predicate $P : \bar{s}$ we introduce a new sort pred_P and new operators $P : \bar{s} \to \mathrm{pred}_P$ and $\top_P : \ \to \mathrm{pred}_P$. The atom $P(\bar{t})$ is represented as the equation $P(\bar{t}) \approx \top_P$. We assume that the operators $\top_P$ are minimal in the reduction ordering. We use a dedicated sort for each predicate in order to avoid problems with persistency. We will omit the P-subscript in our notation.

We also get derived inference rules for resolution from the combination of a paramodulation step using an atom and an equality resolution step. For instance, in the linear paramodulation calculus, we get the rule for linear resolution:

Linear resolution: $$\frac{\Gamma, \pi(\bar{t}) \to}{\Lambda\sigma, \Gamma \to}$$

where $\Lambda \to \pi(\bar{t}')$ is a clause in D and σ is a substitution such that $\pi(\bar{t}')\sigma = \pi(\bar{t})$. This is short for

$$\cfrac{\cfrac{\Gamma, \pi(\bar{t}) \approx \top \to}{\Lambda\sigma, \Gamma, \top \approx \top \to}}{\Lambda\sigma, \Gamma \to}$$

We will later get resolution inference rules for all inference rules which paramodulate into the antecedent; these will be called ordered resolution and selective resolution for the rules strict left superposition and selective paramodulation, respectively.

3 An Abstract Framework for Theorem Proving and Completion

Completion can be seen as a means towards two ends. On the one hand, it can be used as a refutation theorem prover. The advantage of completion as a theorem prover is that it is guided by an ordering; therefore, the number of inferences is greatly reduced. On the other hand, if no refutation is found, the resulting set of clauses is complete. The main advantage is that in a complete set of clauses effective computations can be carried out.

In this section we give an abstract framework for completion, which has been adapted from Nieuwenhuis [Nie91]. It was originally developed for first-order clauses, but since it abstracts from the specific structure of clauses it is well suited for our needs. At the end of this section we discuss the relation to various concepts found in the literature on completion.

3.1 Assumptions

In order to isolate this framework from the rest of the work, we first give all assumptions about clauses, inferences, substitutions and the relations $\models_P$, and $\succ^i$ which we need in order to develop it. We regard these as black boxes and will not make use of their internal structure. Since we need both ground completion and completion on general terms, we will replace the notion of ground substitutions by a subset of substitutions called P-substitutions. In the ground case these will be the ground substitutions. The benefits are that we cover both cases without much overhead, and that similarities are pointed out.

Clauses

There exists a set of *clauses* which contains a distinguished clause denoted by $\Box$ and called the *empty clause*.

Substitutions

There exists a monoid of functions, which are called *substitutions*, from the set of clauses onto itself. The identity substitution is denoted by ι, the composition of two substitutions σ, τ is denoted by $\sigma\tau$, and the result of applying a substitution σ to a clause C is denoted by $C\sigma$.

P-Substitutions

Some of the substitutions are called *P-substitutions*.

Instances of clauses

An *instance* of a clause C is a pair $\langle C, \sigma\rangle$, where σ is a substitution. It is written as $C{\cdot}\sigma$. The application of a substitution τ to an instance $C{\cdot}\sigma$ is defined as $C{\cdot}\sigma\tau$. If σ is a P-substitution, we call $C\sigma$ a *P-clause* and $C{\cdot}\sigma$ a *P-instance* of C.

Instance ordering

There exists an ordering $\succ^i$ on instances of clauses which is well-founded and stable under substitutions. The instances of the empty clause are minimal in $\succ^i$.

Consequence

There is a binary relation $\models_P$ on sets of clauses which is transitive, monotonic with respect to set union and contains $\supseteq$. $\models_P$ is stable under substitutions in the sense that $N \models_P M$

implies $N \models_P M\sigma$. A set of clauses N is called *inconsistent* if $N \models_P \Box$. Two sets of clauses N and M are called *P-equivalent* if $N \models_P M$ and $M \models_P N$.

Compatibility of consequence and P-substitutions

If for all P-instances $C{\cdot}\sigma$ of clauses in M there exist P-instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models_P C\sigma$ then $N \models_P M$.

Inferences

An *inference* is a pair $\langle\langle B_1, \ldots, B_m\rangle, C{\cdot}\mu\rangle$ of a tuple of clauses $\langle B_1, \ldots, B_m\rangle$ and an instance $C{\cdot}\mu$ of a clause such that $B_1, \ldots, B_m \models_P C\mu$. The inference is written as

$$\frac{B_1 \quad \ldots \quad B_m}{C{\cdot}\mu}$$

The clauses $B_1, \ldots, B_m$ are called the *premises* of the inference, the clause $C\mu$ is called its *conclusion* and the substitution μ is called its *unifier*. μ is used to access the instance $B_m{\cdot}\mu$, which is needed when defining redundancy of an inference.

Inference system

An *inference system* $\mathcal{R}$ is a set of inferences.

Instance of an inference

Let π be an inference

$$\frac{B_1 \quad \ldots \quad B_m}{C{\cdot}\mu}$$

If the pair

$$\pi\sigma = \frac{B_1\sigma \quad \ldots \quad B_m\sigma}{C\mu\sigma}$$

is in $\mathcal{R}$, we call the pair $\pi{\cdot}\sigma$ an *instance* of π. Note that $\pi\sigma$ need not be in $\mathcal{R}$ for any substitution σ. If σ is a P-substitution, we call an inference $\pi\sigma$ a *P-inference* and an instance $\pi{\cdot}\sigma$ a *P-instance* of π.

3.2 Redundancy and Completeness

Definition 3.1 (Redundant inference)

Let N be a set of clauses.

1. A P-instance $\pi{\cdot}\sigma$ of an inference
$$\pi = \frac{B_1 \quad \ldots \quad B_m}{C{\cdot}\mu}$$
is called *P-redundant* in N if there exist P-instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models_P C\mu\sigma$ and $B_m{\cdot}\mu\sigma \succ^i C_i{\cdot}\sigma_i$ for $1 \leq i \leq n$.
2. An inference π is called *P-redundant* in N if all P-instances of π are P-redundant in N.
3. An inference is called *P-essential* if it is not P-redundant. $\diamond$

Definition 3.2 (Complete)
A set of clauses N is called *P-complete (with respect to $\mathcal{R}$)* if every inference of $\mathcal{R}$ with premises in N is P-redundant in N. $\diamond$

Definition 3.3 (Refutationally completene)
An inference system $\mathcal{R}$ is called *P-refutationally complete* if every P-inconsistent set of clauses that is P-complete with respect to $\mathcal{R}$ contains the empty clause. $\diamond$

In order to obtain a complete sets of clauses we need a *completion procedure* which generates such a set from a given set of clauses N. The description of this procedure will be given on a very abstract level, stating only the minimal requirements it must fulfil. The first requirement is that it may only change the set of clauses in such a way that equivalence is preserved. The second requirement is called *fairness*. It states that in the limit all inferences must become redundant and ensures that the completion procedure does not ignore an inference forever.

The notion of redundant clause provides a basis for deleting clauses during completion. Deleting redundant clauses results in an equivalent set of clauses, and it preserves the redundancy of other clauses and inferences.

Definition 3.4 (Redundant clause)
Let N be a set of clauses.

1. An instance $C{\cdot}\sigma$ of a clause C is called *P-redundant* in N if there exist P-instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models_P C\sigma$ and $C{\cdot}\sigma \succ^i C_i{\cdot}\sigma_i$ for $1 \leq i \leq n$.
2. A clause C is called *P-redundant* in N if all P-instances of C are P-redundant in N.
3. A clause is called *P-essential* (in N) if it is not P-redundant in N. $\diamond$

Example 3.5 (Semigroup)
Suppose we have a specification of semigroups

$$N = \{(x + y) + z \approx x + (y + z)\}$$

containing only the associativity axiom, where the reduction ordering is a lexicographic path ordering. Roughly speaking, in this ordering a term becomes smaller if the left subterm decreases in size. For instance, $(x + y) + z \succ x + (y + z)$ because of $x + y \succ x$. In our inference system we will have the inference

$$\frac{(x + y) + z \approx x + (y + z) \qquad [B =](x' + y') + z' \approx x' + (y' + z')}{(x + (y + z)) + z' \approx (x + y) + (z + z')}$$

where the associated most general unifier is $\mu = \{x'/(x + y), y'/z\}$. The inference is redundant, because the instances

$$\begin{array}{ll} (x + (y + z)) + z' \approx x + ((y + z) + z') & (1) \\ (y + z) + z' \approx y + (z + z') & (2) \end{array}$$

$$\begin{array}{ll} (x + y) + (z + z') \approx x + (y + (z + z')) & (3) \end{array}$$

are all smaller than $B\mu = ((x+y)+z)+z' \approx (x+y)+(z+z')$. (1) and (2) rewrite the left side and (3) the right side of the conclusion to $x+(y+(z+z'))$; therefore we have (1), (2), (3) $\models (x+(y+z))+z' \approx (x+y)+(z+z')$. Since this is the only inference, N is complete. Since (1)–(3) are also smaller than the conclusion itself, the conclusion serves as an example for a redundant clause. Note that we have treated the variables in the conclusion as constants, and avoided any substitution into them. We will show in section 5 that this guarantees redundancy of all instances of the conclusion.The example dates back to Knuth and Bendix [KB70]. $\diamond$

Lemma 3.6 *Let N and N' be sets of clauses, where $N \subseteq N'$.*

1. *Any inference or clause which is P-redundant in N is also P-redundant in N'.*
2. *If all clauses in $N' \setminus N$ are P-redundant in N' then any inference or clause which is P-redundant in N' is also P-redundant in N.*

Proof: (1) This is trivial.

(2) Let C be a clause which is P-redundant in N' and let $C{\cdot}\sigma$ be a P-instance of C. Then there exist P-instances $C_i{\cdot}\sigma_i$ of clauses in N such that $C{\cdot}\sigma \succ^i C_i{\cdot}\sigma_i$ and $C_1\sigma_1, \ldots, C_n\sigma_n \models_P C\sigma$. Assume that the $C_i{\cdot}\sigma_i$ are taken minimal with respect to the multiset extension of $\succ^i$. Then all the C_i are in N. If we had $C_i \in N' \setminus N$ for some i, the clause C_i would itself be P-redundant in N' and there would exist P-instances $D_j{\cdot}\tau_j$ of clauses in N which imply $C_i\sigma_i$. By replacing the $D_j{\cdot}\tau_j$ for $C_i{\cdot}\sigma_i$ we would obtain a smaller multiset that implies $C\sigma$, contradicting the minimality of $\{C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n\}$. For inferences the proof is similar. $\square$

Lemma 3.7 *Let N and N' be sets of clauses, where $N \subseteq N'$.*

1. *If $N \models_P N'$ then N and N' are P-equivalent.*
2. *If all clauses in $N' \setminus N$ are P-redundant in N' then N and N' are P-equivalent.*

Proof: (1) This is trivial.

(2) We only have to show $N \models_P N'$. Let C be any clause in $N' \setminus N$. From lemma 3.6 we know that C is P-redundant in N; therefore, any P-instance of C is implied by P-instances of clauses in N. We conclude that $N \models_P C$. $\square$

3.3 Theorem Proving Derivations

Definition 3.8 (Theorem proving derivation)
A *P-theorem proving derivation* is a sequence of sets of clauses $N_0, N_1, N_2, \ldots$ such that N_{i+1} is obtained from N by using one of the following inference rules:

***P*-Deduction:** $$\frac{N}{N \cup \{C\}} \qquad \text{if } N \models_P C$$

***P*-Deletion:** $$\frac{N \cup \{C\}}{N} \qquad \text{if } C \text{ is } P\text{-redundant in } N \cup \{C\}$$

$\diamond$

We do not need a separate simplification rule, since simplification can be viewed as a combination of deduction and deletion. Adding the simplified clause D causes the original clause C to become redundant, if C is greater than D and all other clauses used during simplification. Hence simplification is a derived inference

P-Simplification: $$\frac{\dfrac{N \cup \{C\}}{N \cup \{C, D\}}}{N \cup \{D\}} \quad \begin{array}{l}\text{if } N \cup \{C\} \models_P D \text{ and} \\ C \text{ is } P\text{-redundant in } N \cup \{C, D\}\end{array}$$

In our examples we number all equations, and present derivations in the following format:

$$\begin{array}{rll} (i) & C_i & \mathit{comment}_i \\ (i+1) & C_{i+1} & \mathit{comment}_{i+1} \\ \vdots & \vdots & \vdots \\ (n) & C_n & \mathit{comment}_n \end{array}$$

We assume that the set N_0 contains the equations (1)–$(i-1)$, and N_{j+1} is obtained by adding clause C_{i-j} to N_i. We rarely need to delete any clauses, since we will only add conclusions of essential inferences. In these cases, we will give the derivation explicitly.

Definition 3.9 (Fair)
Let $N_0, N_1, N_2, \ldots$ be a P-theorem proving derivation. The set of *persisting clauses* of the derivation is $N_\infty = \bigcup_i \bigcap_{j>i} N_j$. The derivation is called *P-fair (with respect to $\mathcal{R}$)* if any inference in $\mathcal{R}$ with premises in N_∞ is P-redundant in N_j for some index j. $\diamond$

Lemma 3.10 *Let $N{=}N_0$, N_1, N_2, $\ldots$ be a P-theorem proving derivation. Then N and N_∞ are P-equivalent.*

Proof: From lemma 3.7 we know that N is P-equivalent to any N_i. Any clause C in N_∞ is contained in some N_i, thus we have $N \models_P N_\infty$.

For the converse note that $\bigcup_i N_i \models_P N$, and we only have to show that $N_\infty \models_P \bigcup_i N_i$. For any clause C in $(\bigcup_i N_i) \setminus N_\infty$ we know that C is P-redundant in some N_i, and by lemma 3.6, also in N_∞. Therefore, N_∞ and $\bigcup_i N_i$ are P-equivalent by lemma 3.7, and we conclude that $N_\infty \models_P N$. $\square$

A completion procedure has to generate fair theorem proving derivations in order to arrive at a complete set of clauses. Observe that the requirements we make are rather liberal, since the procedure may use arbitrary sound methods to derive new clauses and to prove clauses redundant. The inference system $\mathcal{R}$ is only used to enforce a minimal set of required inferences. These inferences correspond to the notion of "critical pairs".

Lemma 3.11 *Let $N_0, N_1, N_2, \ldots$ be a theorem proving derivation that is P-fair with respect to $\mathcal{R}$.*

1. *Any clause C in $(\bigcup_j N_j) \setminus N_\infty$ is P-redundant in N_∞.*
2. *N_∞ is P-complete with respect to $\mathcal{R}$.*

Proof: (1) Any clause C in $(\bigcup_j N_j) \setminus N_\infty$ must have been deleted at some step in the derivation. Hence, it is P-redundant in some N_k and by lemma 3.6 also in $\bigcup_j N_j$. By using lemma 3.6 again C is P-redundant in N_∞.

(2) Since the derivation is P-fair, any inference with premises in N_∞ is P-redundant in some N_k, and using the same argument as in (1), it is also P-redundant in N_∞. □

Theorem 3.12 *Let $\mathcal{R}$ be a P-refutationally complete inference system, and let $N_0, N_1, N_2, \ldots$ be a theorem proving derivation that is P-fair with respect to $\mathcal{R}$. Then N_0 is P-inconsistent if and only if some N_i contains the empty clause.*

Proof: By lemma 3.10 N_0 and N_∞ are P-equivalent. Since $\mathcal{R}$ is P-refutationally complete and N_∞ is P-complete with respect to $\mathcal{R}$, N_∞ is P-inconsistent if and only if it contains the empty clause. This is the case if and only if it is in some N_i. For the "if" direction of the last statement observe that the empty clause cannot become redundant, since its instances are minimal in $\succ^i$.
□

3.4 Existence of Fair Derivations

Up to now we have not discussed whether there always exist fair derivations. The definition of completeness with respect to $\mathcal{R}$ does not mention how we can cause essential inferences between clauses in N to become redundant. We now show that the usual method of adding all conclusions of essential inferences is sufficient to achieve fairness for certain well-behaved inference systems. In the literature a completion procedures that is always able to generate a fair derivation is called *unfailing*.

Definition 3.13 (Strictly decreasing)
Let $\mathcal{R}$ be an inference system.

1. A P-instance $\pi{\cdot}\sigma$ of an inference
$$\pi = \frac{B_1 \quad \ldots \quad B_m}{C{\cdot}\mu}$$
is called *strictly P-decreasing* if $B_m{\cdot}\mu\sigma \succ^i C{\cdot}\mu\sigma$.
2. An inference π in $\mathcal{R}$ is called *strictly P-decreasing* if any P-instance of π is strictly decreasing.
3. $\mathcal{R}$ is called *strictly P-decreasing* if any inference in $\mathcal{R}$ is strictly P-decreasing. ◇

Lemma 3.14 *Let π be a strictly P-decreasing inference with conclusion C. If C is an instance of a clause in N or if C is P-redundant in N then π is P-redundant in N.*

Proof: Let $\pi{\cdot}\sigma$ be any P-instance of π. In the first case use $\{C{\cdot}\sigma\}$ and in the second case use the set of instances rendering $C{\cdot}\sigma$ redundant to prove redundancy of $\pi{\cdot}\sigma$. □

Lemma 3.15 *Let $\mathcal{R}$ be a strictly P-decreasing inference system. A P-theorem proving derivation $N_0, N_1, N_2, \ldots$ is P-fair with respect to $\mathcal{R}$ if for any P-essential inference in $\mathcal{R}$ with conclusion C and premises in N_∞ there exists an index j, such that C is in N_j.*

Proof: This is an immediate consequence of lemma 3.14. □

Definition 3.16 (Completion procedure)
Let $\mathcal{R}$ be an inference system such that for any finite set of clauses N the set of inferences in $\mathcal{R}$ with premises in N is finite and computable. For a finite set of clauses N_0 the *P-completion procedure* generates a theorem proving derivation $N_0, N_1, N_2, \ldots$ by iterating the following steps, starting with N_0:

1. Do a finite number of arbitrary deduction and deletion steps.
2. Compute the set Π of inferences in $\mathcal{R}$ which have premises in N_i and are not known to be redundant in N_i. Since N_i is finite, Π is also finite.
3. Add the conclusions of the inferences in Π using the deduction rule.
4. If Π is empty, terminate.
5. If the empty clause has been added, terminate.
6. If some inference in Π is not known to be strictly decreasing, fail.
7. Mark the inferences in Π as redundant. $\diamond$

Corollary 3.17 *Let $\mathcal{R}$ be a strictly P-decreasing inference system such that for finite N the set of inferences in $\mathcal{R}$ with premises in N is finite and computable. Then the P-completion procedure generates a theorem proving derivation that is fair with respect to $\mathcal{R}$.*

The completion procedure will serve as the basis for the theorem proving procedures we will define later. We will then put additional restrictions on the procedure, i.e., that it "knows" certain decidable redundancy criteria, that it uses deduction and deletion in a certain way and that it fails in cases where the goal of proving a theorem can no longer be achieved. The reason for writing "not known" is that in general it is undecidable whether an inference is redundant or strictly decreasing.

The difference between termination and failure is that in case of termination we know that all inferences from clauses in N_∞ are redundant and that N_∞ is P-complete with respect to $\mathcal{R}$; in case of failure we cannot make redundant the inference which is not strictly decreasing.

We could have relaxed the condition that for a finite set of clauses N the set of inferences with premises in N is finite. This would require a more sophisticated scheduling mechanism for adding the clauses. Since we do not need it for our inference systems, it is not done here.

Observe that there is a difference between always adding the conclusion C and then deleting C if it is redundant, and adding C only if the inference is essential. In both cases we need to find instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ implying $C\sigma$, but in the latter case these need only be smaller than the instance $B_m{\cdot}\mu\sigma$ of the last premise, whereas otherwise they must be smaller than $C{\cdot}\sigma$ itself.

3.5 The Cases of Ground Terms and Terms with Variables

For the ground case, the above framework has to be instantiated such that P-substitutions are ground substitutions. For this case we will name all notions we have defined by replacing the P prefix by "ground", i.e., ground redundant, ground complete etc. The relation $\models_P$ is instantiated by $\models_I$. An ordering $\succ^i$ and two inference systems for this case are described in section 4. We will later show in theorem 6.3 that $\models_I$ has the following property:

Compatibility of consequence and ground substitutions

If for all ground instances $C{\cdot}\sigma$ of clauses in M there exist ground instances $C_1{\cdot}\sigma_1, \ldots,$ $C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models_I C\sigma$ then $N \models_I M$.

For terms with variables P-substitutions are substitutions $\sigma : X \to T_{\Sigma(K)}$, where K contains skolem constants. We will develop this in detail in section 5. In this case we will just drop the P from the names of the notions defined above, i.e., redundant, complete etc. The relation $\models_P$ is instantiated by $\models$.

3.6 Related Notions

Most notions appearing in this framework have analogies in more concrete views of completion. Fairness has always been defined in the same manner as we do it here. Note that there is a difference between the notion of completeness we have and earlier ones which require the existence of certain "simple" proofs, e.g., by rewriting. Here completeness means only being closed up to redundancy under $\mathcal{R}$; the repercussions on theorem proving in complete sets of clauses are studied separately in subsection 5.3.

Inferences correspond to critical pairs, and the inference system may implement any critical pair criterion that is local to the inference, i.e., that does not refer to other clauses in N. A standard example is that no superpositions on variable positions are needed. Inferences which satisfy other non-local critical pair criteria, like local confluence, subconnectedness, etc., are redundant.

The ordering $\succ^i$ corresponds to the ordering on complexities of clauses used in the proof ordering approach [BDH86, Gan87]. The analogue for the proof ordering itself is its multiset extension $\succ^i_{mul}$, and a proof for a clause is replaced by a set of instances implying that clause. Then redundancy of an inference or clause translates to the existence of a proof that is either smaller than the peak of an inferences or than the clause. The instantiated premise $B_m{\cdot}\mu$ of an inference corresponds to the top of the peak.

4 Inference Systems for Horn Clause Completion

We now instantiate the framework given in section 3 for the case of horn clauses and ground completion. Clauses are horn clauses, P-substitutions are ground substitutions and the consequence relation $\models_P$ is instantiated to inductive consequence. We give a clause ordering and two inference systems, which are restrictions to horn clauses of inference systems given by Nieuwenhuis [Nie91] and Bachmair and Ganzinger [BG90a, BG90b]. Some changes were necessary in order to cope with a partial ordering, resulting in inference rules resembling those given by Bertling [Ber90].

Throughout this section we will assume that there is a fixed signature; therefore, we will only be concerned with sets of clauses. Note that the inference systems depend on the underlying signature, since they refer to the existence of a ground substitutions with certain properties.

4.1 The Ordering on Clause Instances

We assume from now on that $\succ$ is a reduction ordering on $T_{\Sigma(X)}$ which has the simplification property on ground terms, i.e., $u \succeq t$ if t is a subterm of u. Moreover, we require that $t \succ t'$ is decidable for any two ground terms t and t'. We will extend this ordering to an ordering on instances of clauses. The clause ordering itself has been adapted from Nieuwenhuis [Nie91].

To have a uniform definition of reductivity for both the ground case and the case of terms with variables, which we will need later, we define reductivity in a style similar to the framework of section 3. In the ground case P-instances are ground instances.

Definition 4.1 (Reductive)
Let $C = \Gamma \rightarrow l \approx r$ be a clause.

1. An instance $C{\cdot}\sigma$ of C is called *reductive (for l)* if
   (a) $l\sigma \succ r\sigma$,
   (b) $l\sigma \succ u\sigma$ and $l\sigma \succ v\sigma$ for all equations $(u \approx v)\sigma$ in $\Gamma\sigma$ and
   (c) $l\sigma$ contains all variables occurring in $C\sigma$.
2. C is called *P-reductive* if all P-instances of C are reductive. $\diamond$

In the ground case we will call this property *ground reductivity* and in the case of general terms just *reductivity*. For ground clauses both coincide and result in the usual definition of reductivity. Bachmair and Ganzinger [BG90b], as well as some other authors, call this notion of reductivity *universal reductivity*. Their notion of reductivity is defined as the existence of a reductive instance; it is used to express that inference rules are applicable if a reductive instance exists. We think that it is better to have universal reductivity as the default, and to refer explicitly to the existence of a reductive instance in the inference rules. Note that (1b) ensures that $l\sigma$ does not occur in $\Gamma\sigma$.

Later, we will need the property that any unit clause that is a consequence of a complete set of clauses N is redundant in N. In the orderings on clauses that are used by Nieuwenhuis [Nie91] and Bachmair and Ganzinger [BG90b] this is not always the case. For instance, if N contains a clause $\Gamma \rightarrow l \approx r$ and $\Gamma\sigma$ is a consequence of N then $l\sigma \approx r\sigma$ is a consequence of N. But as $(l \approx r){\cdot}\sigma$ is smaller than $(\Gamma \rightarrow l \approx r){\cdot}\sigma$, it is not redundant in N. Therefore, we will have to use an approach where there are two kinds of clauses, ordinary clauses and *rewrite clauses*;

these were previously called conditional equations and conditional rewrite rules, respectively. We will consider rewrite clauses as clauses that are marked somehow, but we will not use a special notation for them. Instead, we will explicitly state whether certain clauses are rewrite clauses where it is needed. If nothing is said a clause can be of either kind. All rewrite clauses are required to be ground reductive. Rewrite clauses will be much cheaper to use in redundancy proofs; their complexity will be dominated by the maximal term of the clause.

Definition 4.2 (Complexity of a clause)
We extend the ordering $\succ$ by a new element $\bot$ such that $t \succ \bot$ for all ground terms t.

An occurrence of a term t in a clause $\Gamma \to \Delta$ has the complexity

$$c(t) = \begin{cases} \{t, \bot\} & \text{if the occurrence is in } \Gamma; \\ \{t\} & \text{if the occurrence is in } \Delta. \end{cases}$$

Occurrences of terms in clauses are ordered by the ordering $\succ^t$ defined by $t \succ t'$ if and only if $c(t) \succ_{mul} c(t')$.

An occurrence of an equation $u \approx v$ in a clause C has the complexity $c(u \approx v) = \{c(u), c(v)\}$, where u and v are regarded as occurrences of terms in C. Occurrences of equations in clauses are ordered by the ordering $\succ^e$, which is defined by $\varphi \succ^e \psi$ if and only if $c(\varphi) \succ^t_{mul} c(\psi)$.

The complexity $c(C)$ of a clause C is the multiset of the complexities of the equations occurring in C. Clauses are ordered by the ordering $\succ^c$, which is defined by $C \succ^c C'$ if and only if $c(C) \succ^e_{mul} c(C')$.

The complexity of an instance $C{\cdot}\sigma$ of a clause $C = \Gamma \to u \approx v$ is

$$c(C{\cdot}\sigma) = \begin{cases} \langle \{\{l\sigma\}\}, C\sigma, C\rangle & \text{if } C \text{ is a rewrite clause and reductive for } l; \\ \langle c(C\sigma), C\sigma, C\rangle & \text{otherwise.} \end{cases}$$

Instances of clauses are ordered by the ordering $\succ^i$, which is the lexicographic combination of $(\succ_{mul})_{mul}$, $\succ^c$ and $\rhd$. ◇

The first component is used to give rewrite clauses a smaller complexity, the second component is the usual ordering on the instances of clauses, and the third component is used to delete clauses that are instances of others. Any instance of the empty clause has the complexity $\langle \emptyset, \square, \square\rangle$ and is minimal in $\succ^i$; $\succ^t$, $\succ^e$, $\succ^c$ and $\succ^i$ are well-founded and stable under substitutions, as required by the framework. Since we do not require that $\succ^i$ is total, we have a greater flexibility in choosing the actual ordering. For the purpose of the completeness proofs given here, the first component of the ordering is sufficient.

The ordering $\succ^c$ assigns slightly more weight to terms in the antecedent. This ordering differs from the ordering given by Bachmair and Ganzinger [BG90b] in the following property: Let $u \approx v$ be an equation occurring in the antecedent, and $u \approx w$ an equation in the succedent, with $w \succ v$. Then in our ordering $u \approx v \succ^e u \approx w$, whereas in the other ordering $u \approx w$ is greater than $u \approx v$.[2]

[2] They define the complexity of an equation $u \approx v$ essentially as $\{u, v, \bot\}$ for occurrences in the antecedent, or $\{u, v\}$ for occurrences in the succedent. These complexities are compared using $\succ_{mul}$. Occurrences of a term in the antecedent or the succedent are not distinguished.

4.2 The Inference Rules of $\mathcal{M}$

Definition 4.3
The inference system $\mathcal{M}$ consists of the inference rules given below with the additional restrictions that

1. the premises of an inference rule must not share any variables (if necessary, the variables in one premise are renamed), and
2. no premise or conclusion must be a tautology.

The inference rules are:

Marking:
$$\frac{[B =]\,\Lambda \to l \approx r}{([C =]\,\Lambda \to l \approx r)\cdot\iota}$$

where C is marked as a rewrite clause, B is ground reductive, and B is not marked as a rewrite clause.

Strict right superposition:
$$\frac{[B' =]\,\Lambda \to l \approx r \qquad [B =]\,\Gamma \to u[l'] \approx v}{([C =]\,\Lambda, \Delta \to u[r] \approx v)\cdot\mu}$$

where C is not marked as a rewrite clause, μ is a most general unifier of l and l', l' is not a variable, and there exists a ground substitution τ such that

1. $B'\cdot\mu\tau$ is reductive for l;
2. $B\cdot\mu\tau$ is reductive for u;
3. $(l \approx r)\mu\tau \not\succeq^e (u \approx v)\mu\tau$.

Strict left superposition:
$$\frac{[B' =]\,\Lambda \to l \approx r \qquad [B =]\,\Gamma, u[l'] \approx v \to \Delta}{([C =]\,\Lambda, \Gamma, u[r] \approx v \to \Delta)\cdot\mu}$$

where C is not marked as a rewrite clause, μ is a most general unifier of l and l', l' is not a variable, and there exists a ground substitution τ such that

1. $B'\cdot\mu\tau$ is reductive for l;
2. $B\cdot\mu\tau$ is not reductive;
3. $v\,\mu\tau \not\succeq u\,\mu\tau$;
4. $(u \approx v)\mu\tau$ is maximal with respect to $\succ^e$ in $\Gamma\mu\tau$.

Ordered equality resolution:
$$\frac{[B =]\,\Gamma, u \approx v \to \Delta}{([C =]\,\Gamma \to \Delta)\cdot\mu}$$

where C is not marked as a rewrite clause, μ is a most general unifier of u and v, and there exists a ground substitution τ such that

1. $B\cdot\mu\tau$ is not reductive;
2. $(u \approx v)\mu\tau$ is maximal with respect to $\succ^e$ in $\Gamma\mu\tau$.

$\diamond$

4.3 Construction of Equality Interpretations

An *(equality Herbrand) interpretation* is a congruence on ground terms. An interpretation I corresponds to the quotient algebra T_Σ/I. An equation $u \approx v$ is true in I if and only if $T_\Sigma/I \models u \approx v$. We may consider an interpretation I as a set of ground equations; a ground clause $\Gamma \to \Delta$ is true in I if and only if $\Gamma \not\subseteq I$ or else $\Delta \cap I \neq \emptyset$. To a canonical ground rewrite system R corresponds an interpretation $R^* = \{\, s \approx t \mid s \Downarrow_R t \,\}$.

We give an interpretation for an arbitrary set of clauses by inductively constructing a canonical ground rewrite system. We will later show that for complete N the interpretation is a model of N, provided that a model exists. For this construction it is essential to use a total ordering, since otherwise it cannot be guaranteed that the resulting rewrite system is canonical. Therefore, we assume that $\succ^{it}$ is a well-founded and total ordering on instances of clauses which contains $\succ^{i}$. It is not required to be a reduction ordering.

Definition 4.4
A ground instance $C{\cdot}\sigma$ of a clause $C = \Lambda \to l \approx r$ in N *generates* a rule $l\sigma \approx r\sigma$ if

1. $I_{C\cdot\sigma} \not\models C\sigma$,
2. $C{\cdot}\sigma$ is reductive for l, and
3. $l\sigma$ is irreducible by $R_{C\cdot\sigma}$,

where $R_{C\cdot\sigma}$ is the set of rules generated by all ground instances of clauses in N which are smaller than $C{\cdot}\sigma$ with respect to $\succ^{it}$, and $I_{C\cdot\sigma} = R^*_{C\cdot\sigma}$. R_N is the set of rules generated by all ground instances of clauses in N, and $I_N = R^*_N$. $\Diamond$

This interpretation will be the basis of the completeness proof for the inference systems, but its significance goes beyond that. It represents the rewrite procedure of an actual implementation; therefore, it could be used to prove that such an implementation is correct. Of course, different models correspond to different implementations. This one, for instance, will do innermost rewriting; this is because no rule will be generated if it is possible to rewrite a subterm (3). It will only use reductive instances of clauses for rewriting (2). Since an actual implementation cannot store an infinite number of rewrite rules, it has to use recursive rewriting, checking that the equations in the antecedent are satisfied (1).

Lemma 4.5 *Any of the rewrite systems $R_{C\cdot\sigma}$ and R_N is canonical.*

Proof: Let $u\sigma \approx v\sigma$ be a rule generated by the ground instance $C{\cdot}\sigma$ of the clause $C = \Gamma \to u \approx v$ and let $l\sigma' \approx r\sigma'$ be generated by the instance $C'{\cdot}\sigma'$ of $C' = \Lambda \to l \approx r$. Assume that both rules are in $R_{C''\cdot\sigma''}$ $[R_N]$ and that $l\sigma'$ is a subterm of $u\sigma$, which implies $u\sigma \succeq l\sigma'$. For the case $u\sigma \succ l\sigma'$ we have $C{\cdot}\sigma \succ^{it} C'{\cdot}\sigma'$, even if C is a rewrite clause and C' is not. Therefore, $u\sigma$ is reducible by $R_{C\cdot\sigma}$ and $C{\cdot}\sigma$ does not generate the rule $u\sigma \approx v\sigma$, which is a contradiction. If $u\sigma = l\sigma'$ and $C{\cdot}\sigma \succ^{it} C'{\cdot}\sigma'$, the same argument applies; if $C'{\cdot}\sigma' \succ^{it} C{\cdot}\sigma$, we may swap $C{\cdot}\sigma$ and $C'{\cdot}\sigma'$. $\Box$

Suppose some ground instance of a clause does not generate a rule, because the interpretation constructed so far does not satisfy the antecedent. In this case the antecedent could possibly become true at a later stage, causing the clause to become false. We will show that this is not the case.

Lemma 4.6 *Let N be a set of clauses, let $C{\cdot}\sigma$, $C'{\cdot}\sigma'$ be ground instances of clauses in N with $C{\cdot}\sigma \succeq^i C'{\cdot}\sigma'$, and let $u \approx v$ be an equation in the antecedent of $C'\sigma'$. Then $u \approx v$ is true in $I_{C\cdot\sigma}$ $[(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*]$ $[I_N]$ if and only if it is true in $I_{C'\cdot\sigma'}$.*

Proof: Assume that $u \approx v$ is true in $I_{C\cdot\sigma}$, which is equivalent to $u \Downarrow_{R_{C\cdot\sigma}} v$. If $u = v$ we are done. Therefore, we may assume without loss of generality that u is reducible by some rule $l \approx r$ in $R_{C\cdot\sigma}$. Then l is a subterm of u, and since $\succ$ has the simplification property on ground terms, $u \succeq l$. Since the instance $C''{\cdot}\sigma''$ generating $l \approx r$ is reductive, and since u occurs in the antecedent of $C'\sigma'$, the instance $C''{\cdot}\sigma''$ is smaller than $C'{\cdot}\sigma'$ and we get $u \Downarrow_{R_{C'\cdot\sigma'}} v$. The converse is a consequence of $R_{C'\cdot\sigma'} \subseteq R_{C\cdot\sigma}$. For $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$ and I_N the proof is similar. □

Lemma 4.7 *Let N be a set of clauses and let $C{\cdot}\sigma$, $C'{\cdot}\sigma'$ be ground instances of clauses in N with $C{\cdot}\sigma \succeq^i C'{\cdot}\sigma'$. Then $I_{C\cdot\sigma}[(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*][I_N] \not\models C'\sigma'$ implies $I_{C'\cdot\sigma'} \not\models C'\sigma'$.*

Proof: Let $C' = \Gamma \rightarrow \Delta$. Then $\Delta \cap I_{C\cdot\sigma} = \emptyset$ implies $\Delta \cap I_{C'\cdot\sigma'} = \emptyset$, and $\Gamma \subseteq I_{C\cdot\sigma}$ implies $\Gamma \subseteq I_{C'\cdot\sigma'}$ by lemma 4.6. We conclude that $I_{C'\cdot\sigma'} \not\models C'\sigma'$. The same applies to the cases I_N and $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$. □

Although this version of the lemma suits our proofs better, it is probably more intuitive to state it the other way round:

$$I_{C'\cdot\sigma'} \models C'\sigma' \text{ implies } I_{C\cdot\sigma} \models C'\sigma'.$$

Our rules do not allow paramodulation into variable positions. The completeness proof uses the following lemma to prove that indeed this is not necessary.

Lemma 4.8 *Let N be a set of clauses and let $C{\cdot}\sigma$ be a ground instance of a clause C such that $I_{C\cdot\sigma}[(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*][I_N] \not\models C\sigma$. If there exists some variable x in C such that $x\sigma$ is reducible by $R_{C\cdot\sigma}$ $[R_N]$ then there exists a ground substitution τ such that $I_{C\cdot\tau}$ $[(R_{C\cdot\tau} \cup E_{C\cdot\tau})^*][I_N] \not\models C\tau$ and $C{\cdot}\sigma \succ^i C{\cdot}\tau$.*

Proof: For $x\sigma \Rightarrow_{R_{C\cdot\sigma}} t$ define τ by $x\tau = t$ and $y\tau = y\sigma$ for $y \neq x$. We have $C{\cdot}\sigma \succ^i C{\cdot}\tau$, since replacing any subterm of a term in $C\sigma$ by a smaller term yields a smaller clause $C\tau$. The validity of $C\sigma$ is not affected, because $R_{C\cdot\sigma}$ is canonical. Using lemma 4.7 we get that $I_{C\cdot\tau}[(R_{C\cdot\tau} \cup E_{C\cdot\tau})^*][I_N] \not\models C\tau$. □

4.4 Refutation Completeness of $\mathcal{M}$

Lemma 4.9 *Let N be a set of clauses that is ground complete with respect to $\mathcal{M}$, that does not contain the empty clause and that contains only unit clauses which are ground reductive. Then for every ground instance $C{\cdot}\sigma$ of a clause C in N $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \models C\sigma$.*

Proof: Let N be a set of clauses that is ground complete with respect to $\mathcal{M}$ and that does not contain the empty clause. We assume that there exists some ground instance $C{\cdot}\sigma$ of a clause C in N such that $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models C\sigma$. Since $\succ^i$ is well-founded we may assume that $C{\cdot}\sigma$ is chosen to be a minimal such clause with respect to $\succ^i$.

There exists no variable x which occurs in C such that $x\sigma$ is reducible by $R_{C\cdot\sigma} \cup E_{C\cdot\sigma}$. Otherwise, by lemma 4.8, there exists a ground instance $C{\cdot}\tau$ of C which is smaller than $C{\cdot}\sigma$ such that $C\tau$ is false in $(R_{C\cdot\tau} \cup E_{C\cdot\tau})^*$, contradicting the minimality of $C{\cdot}\sigma$.

1. Assume that $C{\cdot}\sigma$ is reductive for u. Then $C\sigma$ has the form $\Gamma\sigma \to u\sigma \approx v\sigma$. The instance $C{\cdot}\sigma$ does not generate a rule, otherwise it would be true in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$. The only possible cause is that $u\sigma$ is reducible by $R_{C\cdot\sigma}$; hence, there exists a ground instance $C'{\cdot}\sigma'$ of a clause $C' = \Lambda \to l \approx r$ which generates a rule $l\sigma' \approx r\sigma'$ in $R_{C\cdot\sigma}$ such that $l\sigma'$ is a subterm of $u\sigma$. Since $C'{\cdot}\sigma'$ is reductive for l, $\mathcal{M}$ contains the inference by strict right superposition

$$\frac{\Lambda \to l \approx r \qquad \Gamma \to u[l'] \approx v}{(\Lambda, \Gamma \to u[r] \approx v){\cdot}\mu}$$

where μ is a most general unifier of l and l'. By general assumption C and C' have no variables in common; thus, there exists a ground substitution τ such that $C\mu\tau = C\sigma$ and $C'\mu\tau = C'\sigma'$. Moreover, τ satisfies the conditions associated with the inference rule. In the case $(l \approx r)\sigma' = (u \approx v)\sigma$ we would get a contradiction, since $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \models (l \approx r)\sigma'$ but $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models (u \approx v)\sigma$; in the case $(l \approx r)\sigma' \succ^e (u \approx v)\sigma$ the rule $l\sigma' \approx r\sigma'$ would not be in $R_{C\cdot\sigma}$. Then the ground instance $C''\mu\tau = \Lambda\sigma', \Gamma\sigma \to u\sigma[r\sigma'] \approx v\sigma$ of the conclusion C'' is not true in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$.

   Since N is ground complete, the inference is ground redundant in N; therefore, there exist ground instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models C''\mu\tau$ and the premise $C{\cdot}\sigma$ is greater than $C_i{\cdot}\sigma_i$ for $1 \le i \le n$. Since $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models C''\mu\tau$, there must be some $C_i{\cdot}\sigma_i$ with $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models C_i\sigma_i$. By lemma 4.7 this implies that $C_i\sigma_i$ is false in $(R_{C_i\cdot\sigma_i} \cup E_{C_i\cdot\sigma_i})^*$, which contradicts the assumption that $C{\cdot}\sigma$ is minimal.

2. Now assume that $C{\cdot}\sigma$ is not reductive. Since all unit clauses in N are ground reductive, C has the form $\Gamma, u \approx v \to \Delta$, and we may assume that $u\sigma \approx v\sigma$ is maximal in $\Gamma\sigma$.

   (a) If $u\sigma = v\sigma$, $\mathcal{M}$ contains the inference by ordered equality resolution

   $$\frac{\Gamma, u \approx v \to \Delta}{(\Gamma \to \Delta){\cdot}\mu}$$

   The ground instance $C''\mu\tau = (\Gamma \to \Delta){\cdot}\sigma$ of the conclusion is not true in $(R_{C''\cdot\mu\tau} \cup E_{C''\cdot\mu\tau})^*$, and we obtain a contradiction as in the previous case.

   (b) The remaining case is $u\sigma \neq v\sigma$. Since $C\sigma$ is false in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$, we have $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \models u\sigma \approx v\sigma$; therefore, without loss of generality, $v\sigma \not\succeq u\sigma$ and $u\sigma$ is reducible by $R_{C\cdot\sigma}$. Hence, there exists a ground instance $C'{\cdot}\sigma'$ of a clause $C' = \Lambda \to l \approx r$ in N generating a rule $l\sigma' \approx r\sigma'$ such that $l\sigma'$ is a subterm of $u\sigma$. Then $\mathcal{M}$ contains the inference by strict left superposition

   $$\frac{\Lambda \to l \approx r \qquad \Gamma, u[l'] \approx v \to \Delta}{(\Lambda, \Gamma, u[r] \approx v \to \Delta){\cdot}\mu}$$

   where μ is a most general unifier of l and l'. There exists a ground substitution τ such that $C\mu\tau = C\sigma$ and $C'\mu\tau = C'\sigma'$, and τ satisfies the conditions associated with the inference rule. The ground instance $C''\mu\tau = \Lambda\sigma', \Gamma\sigma, u\sigma[r\sigma'] \approx v\sigma \to \Delta\sigma$ of the conclusion C'' is not true in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$, and we obtain a contradiction as in the first case. □

Theorem 4.10 *Let N be a set of clauses that is ground complete with respect to $\mathcal{M}$ and that contains only unit clauses which are ground reductive. Then N is inconsistent if and only if N contains the empty clause. Moreover, if N is consistent then the interpretation I_N satisfies all clauses in N.*

Proof: This is a consequence of lemma 4.9 and lemma 4.7.

The requirement that all unit clauses in N are reductive is always satisfied if the reduction ordering is total on ground terms; therefore, this restriction applies only to the non-ground case, where we cannot obtain such a total ordering.

Corollary 4.11 *If $\succ$ is total on ground terms then the inference system $\mathcal{M}$ is refutationally complete.*

4.5 Literal Selection Strategies

The inference system $\mathcal{M}$ employs a maximal literal strategy, i.e., superposition occurs always in the maximal literal. This fixed strategy is not always appropriate. It is possible to use a more flexible strategy instead, where for arbitrary clauses literals in the antecedent may be selected for superposition. It is then sufficient to paramodulate all clauses into maximal selected literals, instead of using strict superposition. Clauses for which equations have been selected correspond to the so called "nonoperational equations" found in earlier work by Ganzinger [Gan87]. The justification for this terminology is that they do not contribute to the rewrite relation. We will use this feature later as the basis of our proof procedures. The actual definition of selection function which we use here has been taken from Bachmair and Ganzingers work [BG90b].

Definition 4.12
A *selection function* is a set S of pairs $\langle C, \Gamma \rangle$, where C is a clause that is not a rewrite clause and Γ is a nonempty subset of the antecedent of C. For each clause there is at most one such pair in S. $\diamond$

Definition 4.13
The inference system $\mathcal{S}_S$ consists of

1. all inferences in $\mathcal{M}$ which have no premise C such that some $\langle C, \Gamma \rangle$ is in S,
2. the inferences by the rules given below, with the same restrictions regarding tautologies and variables as for $\mathcal{M}$.

Selective paramodulation:
$$\frac{[B' =]\, \Lambda \rightarrow l \approx r \qquad [B =]\, \Gamma, u[l'] \approx v \rightarrow \Delta}{([C =]\, \Lambda, \Gamma, u[r] \approx v \rightarrow \Delta)\cdot\mu}$$

where C is not marked as a rewrite clause, μ is a most general unifier of l and l', l' is not a variable, some pair $\langle B, \Gamma' \rangle$ is in S, no pair $\langle B', Lambda' \rangle$ is in S, and there exists a ground substitution τ such that

1. $B'\cdot\mu\tau$ is reductive for l;
2. $v\,\mu\tau \not\succeq u\,\mu\tau$;
3. $(u \approx v)\mu\tau$ is maximal with respect to $\succ^e$ in $\Gamma'\mu\tau$.

Selective equality resolution:
$$\frac{[B =]\, \Gamma, u \approx v \rightarrow \Delta}{([C =]\, \Gamma \rightarrow \Delta)\cdot\mu}$$

where C is not marked as a rewrite clause, μ is a most general unifier of u and v, some pair $\langle B, \Gamma' \rangle$ is in S, and there exists a ground substitution τ such that

1. $(u \approx v)\mu\tau$ is maximal with respect to $\succ^e$ in $\Gamma'\mu\tau$.

$\Diamond$

The inference system $\mathcal{M}$ can be regarded as the special case of $\mathcal{S}_S$ where S is empty.

Lemma 4.14 *Let N be a set of clauses that is ground complete with respect to $\mathcal{S}_S$, that does not contain the empty clause and that contains only unit clauses which are ground reductive. Then for every ground instance $C{\cdot}\sigma$ of a clause C in N $\;(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \models C\sigma$.*

Proof: This proof is an extension of the proof for $\mathcal{M}$. Let $C{\cdot}\sigma$ be the minimal ground instance of a clause such that $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models C\sigma$. The previous proof covers the case where S does not contain a pair $\langle C, \Gamma' \rangle$. Therefore, assume that $\langle C, \Gamma' \rangle$ is in S.

1. If $u\sigma = v\sigma$, $\mathcal{S}_S$ contains the inference by selective equality resolution

   $$\frac{\Gamma, u \approx v \to \Delta}{(\Gamma \to \Delta){\cdot}\mu}$$

   where μ is a most general unifier of u and v. The ground instance $C''\sigma = \Gamma\sigma \to \Delta\sigma$ of the conclusion C'' is false in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$. The inference is ground redundant in N; therefore, there exist ground instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models C''\mu\tau$ and the premise $C{\cdot}\sigma$ is greater than $C_i{\cdot}\sigma_i$ for $1 \leq i \leq n$. Since $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models C''\mu\tau$ there must be some $C_i{\cdot}\sigma_i$ with $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \not\models C_i\sigma_i$. Using lemma 4.7 we get $(R_{C_i\cdot\sigma_i} \cup E_{C_i\cdot\sigma_i})^* \not\models C_i\sigma_i$, which contradicts the minimality of $C{\cdot}\sigma$.

2. Otherwise $u\sigma \neq v\sigma$. Since $C\sigma$ is false in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$, we have $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^* \models u\sigma \approx v\sigma$; therefore, without loss of generality, $v\sigma \not\succeq u\sigma$ and $u\sigma$ is reducible by $R_{C\cdot\sigma} \cup E_{C\cdot\sigma}$. Hence, there exists a ground instance $C'{\cdot}\sigma'$ of a clause $C' = \Lambda \to l \approx r$ in N generating a rule $l\sigma' \approx r\sigma'$ in $R_{C\cdot\sigma} \cup E_{C\cdot\sigma}$ such that $l\sigma'$ is a subterm of $u\sigma$.

   (a) If there exists some variable x which occurs in u such that $x\sigma$ is reducible by $l\sigma' \approx r\sigma'$ then by lemma 4.8 there exists a ground instance $C{\cdot}\tau$ which is smaller than $C{\cdot}\sigma$ and false in $(R_{C\cdot\tau} \cup E_{C\cdot\tau})^*$, contradicting the minimality of $C{\cdot}\sigma$.

   (b) Otherwise, $\mathcal{S}_S$ contains the inference by selective paramodulation

   $$\frac{\Lambda \to l \approx r \qquad \Gamma, u[l'] \approx v \to \Delta}{(\Lambda, \Gamma, u[r] \approx v \to \Delta){\cdot}\mu}$$

   where μ is a most general unifier of l and l'. Let τ be a ground substitution such that $C\mu\tau = C\sigma$ and $C'\mu\tau = C'\sigma'$. Furthermore, τ satisfies all conditions associated with the inference rule. The ground instance $C''\mu\tau = \Lambda\sigma', \Gamma\sigma, u\sigma[r\sigma'] \approx v\sigma \to \Delta\sigma$ of the conclusion C'' is not true in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$, and we obtain a contradiction as in the first case. $\Box$

Theorem 4.15 *Let N be a set of clauses that is ground complete with respect to $\mathcal{S}_S$ and that contains only unit clauses which are ground reductive. Then N is inconsistent if and only if N contains the empty clause. Moreover, if N is consistent then the interpretation I_N satisfies all clauses in N.*

Proof: This again is a consequence of lemma 4.14 and lemma 4.7.

Lemma 4.16 *Let N be a set of clauses that is ground complete with respect to $\mathcal{S}_S$. If $\langle C, \Gamma \rangle$ is a pair in S then no ground instance of C generates a rule in R_N.*

Proof: Let $C{\cdot}\sigma$ be a minimal ground instance of a clause $C = \Gamma, u \approx v \rightarrow l \approx r$ such that $\langle C, \Gamma' \rangle \in S$, $u\sigma \approx v\sigma$ is maximal in $\Gamma'\sigma$ and $C{\cdot}\sigma$ generates a rule $l\sigma \approx r\sigma$. Then $\Gamma\sigma$ is true in $I_{C\cdot\sigma}$ and $l\sigma$ is irreducible by $R_{C\cdot\sigma}$.

Suppose there exists a variable x such that $x\sigma$ reduces to t. Then let τ be the substitution with $x\tau = t$ and $y\tau = y\sigma$ for $y \neq x$. Hence $C{\cdot}\tau$ is a smaller ground instance of C which produces a rule $l\sigma \approx r\tau$, contradicting the minimality of $C{\cdot}\sigma$. Note that reductivity is preserved since irreducibility of $l\sigma$ prevents x from occuring in l.

1. If $u\sigma = v\sigma$, $\mathcal{S}_S$ contains the inference by selective equality resolution

   $$\frac{\Gamma, u \approx v \rightarrow \Delta}{(\Gamma \rightarrow \Delta){\cdot}\mu}$$

   where μ is a most general unifier of u and v. The ground instance $C''\sigma = \Gamma\sigma \rightarrow \Delta\sigma$ of the conclusion C'' is not true in $I_{C\cdot\sigma}$. The inference is ground redundant in N; therefore, there exist ground instances $C_1{\cdot}\sigma_1, \ldots, C_n{\cdot}\sigma_n$ of clauses in N such that $C_1\sigma_1, \ldots, C_n\sigma_n \models C''\mu\tau$ and the premise $C{\cdot}\sigma$ is greater than $C_i{\cdot}\sigma_i$ for $1 \leq i \leq n$. Since $I_{C\cdot\sigma} \not\models C''\mu\tau$ there must be some $C_i{\cdot}\sigma_i$ with $I_{C\cdot\sigma} \not\models C_i\sigma_i$. Using lemma 4.7 we get $I_{C_i\cdot\sigma_i} \not\models C_i\sigma_i$. But by lemma 4.14 $(R_{C_i\cdot\sigma_i} \cup E_{C_i\cdot\sigma_i})^* \models C_i\sigma_i$, which is only possible if $C_i{\cdot}\sigma_i$ is productive, contradicting the minimality of $C{\cdot}\sigma$.
2. Otherwise $u\sigma \neq v\sigma$. Since $C\sigma$ is false in $I_{C\cdot\sigma}$, we have $I_{C\cdot\sigma} \models u\sigma \approx v\sigma$; therefore, without loss of generality, $v\sigma \not\succeq u\sigma$ and $u\sigma$ is reducible by $R_{C\cdot\sigma}$. Hence, there exists a ground instance $C'{\cdot}\sigma'$ of a clause $C' = \Lambda \rightarrow l \approx r$ in N generating a rule $l\sigma' \approx r\sigma'$ in $R_{C\cdot\sigma}$ such that $l\sigma'$ is a subterm of $u\sigma$. Then $\mathcal{S}_S$ contains the inference by selective paramodulation

   $$\frac{\Lambda \rightarrow l \approx r \qquad \Gamma, u[l'] \approx v \rightarrow \Delta}{(\Lambda, \Gamma, u[r] \approx v \rightarrow \Delta){\cdot}\mu}$$

   where μ is a most general unifier of l and l'. Let τ be a ground substitution such that $C\mu\tau = C\sigma$ and $C'\mu\tau = C'\sigma'$. Furthermore, τ satisfies all conditions associated with the inference rule. The ground instance $C''\mu\tau = \Lambda\sigma', \Gamma\sigma, u\sigma[r\sigma'] \approx v\sigma \rightarrow \Delta\sigma$ of the conclusion C'' is false in $I_{C\cdot\sigma}$, and we obtain a contradiction as in the first case. □

Lemma 4.17 *Let N be a set of clauses that is ground complete with respect to $\mathcal{S}_S$, and let I_N be the interpretation constructed from N. If $C{\cdot}\sigma$ is a ground instance of a clause C in N, and some $\langle C, \Gamma' \rangle$ is in S then $C\sigma$ is true in $I_{C\cdot\sigma}$.*

Proof: By lemma 4.14 $C\sigma$ is true in $(R_{C\cdot\sigma} \cup E_{C\cdot\sigma})^*$, and by lemma 4.16 $E_{C\cdot\sigma}$ is empty. □

Theorem 4.18 *The inference systems $\mathcal{M}$ and $\mathcal{S}_S$ are strictly ground decreasing.*

Proof: We prove this by case analysis on the inference rules. Marking obviously decreases the complexity of the clause. Let

$$\frac{B' \quad B}{C{\cdot}\mu}$$

be an inference. For strict right superposition the maximal term of B is reduced. Even if both premises are rewrite clauses, the conclusion is smaller than B, because its maximal term is smaller. For strict left superposition or selective paramodulation the second premise cannot be a rewrite clause, because B is not reductive or because S contains a pair $\langle B, \Gamma' \rangle$, respectively. Therefore, it suffices to note that some term t in B is reduced in C. In all three cases the maximal term of B' is smaller with respect to $\succ^t$ than the term t it is superposed on. Therefore, all terms in the conclusion which originate from the first premise are strictly smaller than t.

For both variants of equality resolution one equation from the antecedent is eliminated. Note that in this case the premise cannot be a rewrite clause. □

4.6 Minimality of I_N

We have seen that I_N describes how computations with respect to a set of horn clauses are carried out. Normally, the intended semantics for computations is the initial model of the specification; the question arises whether I_N is a representation of the initial model. We will show that this is indeed the case. Bachmair and Ganzinger [BG90c] prove a similar fact, but use a fixpoint characterization of the initial model.

Lemma 4.19 *Let N be a consistent set of clauses, let I_N be the interpretation constructed from N and let $u \approx v$ be a ground equation. Then $I_N \models u \approx v$ implies $N \models u \approx v$.*

Proof: Let $u \approx v$ be a minimal ground equation that is true in I_N, but not a consequence of N. We know that $u \Downarrow_{R_N} v$, and since $u \approx v$ cannot be a tautology, we may assume without loss of generality that u is reducible by R_N and that $v \not\succeq u$. Then there exists a ground instance $C{\cdot}\sigma$ of a clause $C = \Gamma \rightarrow l \approx r$ in N generating a rule $l\sigma \approx r\sigma$ such that $l\sigma$ is a subterm of $u\sigma$. From reductivity of $C{\cdot}\sigma$ we get $l\sigma \succ r\sigma$; therefore, $u[r\sigma] \approx v$ is smaller than $u \approx v$. It is also true in I_N, as well as $l\sigma \approx r\sigma$. From the simplification property of $\succ$ on ground terms we get $u \succeq l\sigma$. All equations in $\Gamma\sigma$ are true in I_N, and because $C{\cdot}\sigma$ is reductive, smaller than $u \approx v$; hence, they are consequences of N. Since C is in N, $l\sigma \approx r\sigma$ is also a consequence of N. Then $u[r\sigma] \approx v$ cannot be a consequence of N, and we obtain a contradiction to the minimality of $u \approx v$. □

Theorem 4.20 *Let N be a consistent set of clauses that is ground complete with respect to $\mathcal{M}$ or $\mathcal{S}_S$. Then T_Σ / I_N is an initial model of N.*

Proof: This is equivalent to I_N being the smallest interpretation, with respect to set inclusion, satisfying the clauses in N. From theorem 4.10 for the case of $\mathcal{M}$ or from theorem 4.15 for the case of $\mathcal{S}_S$ we know that I_N is a model of N, and from lemma 4.19 we know that I_N is minimal. □

5 Extension to General Terms

Up to now we have only covered the case of ground terms. Since we will later need completion techniques for the case of general terms, we have to extend the methods.

In the preceding sections on completion we have assumed a fixed signature; therefore, we have only operated on sets of clauses. Now, we will have to distinguish between different signatures. Therefore, we will follow the convention that a specification may stand for the set of clauses it contains. For the case of theorem proving derivations, where several sets of clauses are involved, we will write D_0, D_1, D_2, $\ldots$; it should be understood that the signature is the same for all specifications in the derivation. We will also have to specify the signature for the inference systems, since the signature affects the existence of certain ground substitutions. We will write $\mathcal{S}_S^\Sigma$ for the inference system that is defined with respect to the signature Σ.

We use the same clause ordering as for ground the ground case. However, since the underlying reduction ordering $\succ$ cannot be total on $T_{\Sigma(X)}$, the orderings $\succ^t$, $\succ^e$, $\succ^c$ and $\succ^i$ are not total on all instances of clauses.

5.1 Skolemization

For any signature Σ we assume the existence of an S-sorted set K of *skolem constants*, and an S-sorted bijection $\kappa : X \to K$, which we extend to a Σ-isomorphism $\kappa : T_{\Sigma(X)} \to T_{\Sigma(K)}$ on the term algebras. We may regard κ as a substitution $\kappa : X \to T_{\Sigma(K)}$. We assume that the skolem constants are distinct from all constants appearing in Σ. In particular, no skolem constant may appear in a clause of a specification. $\Sigma(K)$ is the signature which is obtained by adding all skolem constants in K to Σ. For the signature containing both variables and skolem constants we write $\Sigma(K, X)$. If D is a specification $\langle \Sigma, N \rangle$ then $D(K)$ will denote the specification $\langle \Sigma(K), N \rangle$. There exists a specification inclusion $\xi : D \hookrightarrow D(K)$. Since it is an inclusion, we have $\xi(C) = C$ for any clause C. For the reduction ordering $\succ$ on $T_{\Sigma(K,X)}$ we require $t\kappa \succ t'\kappa$ if and only if $t \succ t'$.

For a specification $D = \langle \Sigma, N \rangle$, we will instantiate the set of P-substitutions of our framework as the set of all substitutions $\sigma : X \to T_{\Sigma(K)}$.

We now establish the effect of skolemization on the consequence relation. If both the clauses at the left and the clauses at the right side are skolemized, this may be viewed as putting a universal quantifier around the whole expression, i.e., as the equivalent of $\forall X.\, M \supset N$.

Lemma 5.1 *Let D be a specification and let C be a clause. The following are equivalent.*

1. $D \models C$.
2. $D(K) \models C$.
3. $D(K) \models_I C$.
4. $D(K) \models C\kappa$.

Proof: (1) $\Rightarrow$ (2) This holds since the ξ-reduct $A|_\xi$ of any model A of $D(K)$ is a model of D, and since $A|_\xi$ differs from A only in the associated signature.

(2) $\Rightarrow$ (3) This is trivial.

(3) $\Rightarrow$ (4) Let A be any model of $D(K)$. The skolemized clause $C\kappa$ contains only ground equations, which are valid in A if and only if they are valid in A_{gen}. By assumption C and also its instance $C\kappa$ are valid in A_{gen}; therefore, $C\kappa$ must be valid in A.

(4) $\Rightarrow$ (1) Let A be a model of D, and let $\alpha : X \to A$ be an assignment. Define the $D(K)$-algebra A' as the strongly persistent ξ-extension of A with $x\kappa_{A'} = \alpha(x)$. It is a direct consequence of the definition of $\models_\alpha$ that $A \models_\alpha C$ if and only if $A' \models C\kappa$, which in turn is true by assumption. $\square$

Lemma 5.2 *Let N and M be sets of Σ-clauses. If $N\kappa \models M\kappa$ then $N\sigma \models M\sigma$ for all substitutions $\sigma : X \to T_{\Sigma(K)}$.*

Proof: Let σ be a substitution and let A be an algebra that satisfies $N\sigma$. Let A' be the strongly persistent ξ-extension of $A|_\xi$, where $x\kappa_{A'} = \#_A(x\sigma)$. By the definition of $\models$, we have $A \models N\sigma$ if and only if $A' \models N\kappa$. This implies $A' \models M\kappa$ and by the same reasoning as before we get $A \models M\sigma$. $\square$

For reference, we restate redundancy, completeness, etc., for the non-ground case in terms of the respective notions for the ground case with respect to a skolemized specification.

Lemma 5.3 *Let D be a specification.*

1. *A clause C is redundant in D if and only if C is ground redundant in $D(K)$.*
2. *An inference π is redundant in D if and only if π is ground redundant in $D(K)$.*

Lemma 5.4 *D_0, D_1, D_2, ... is a theorem proving derivation if and only if $D_0(K)$, $D_1(K)$, $D_2(K)$, ... is a ground theorem proving derivation.*

Lemma 5.5 *A clause is reductive with respect to Σ if and only if it is ground reductive with respect to $\Sigma(K)$.*

To prove redundancy of a clause for the case of terms with variables, it is sufficient to prove that the skolemized clause is redundant. It is not necessary to consider all instances of the clause as in the ground case.

Lemma 5.6 *Let D be a specification.*

1. *The clause C is redundant in D if and only if $C\kappa$ is redundant in D.*
2. *The inference*
$$\pi = \frac{B_1 \quad \ldots \quad B_m}{C{\cdot}\mu}$$
is redundant in D if and only if
$$\pi\kappa = \frac{B_1\mu\kappa \quad \ldots \quad B_m\mu\kappa}{C{\cdot}\mu\kappa}$$
is redundant in $D(K)$.

Proof: (1) If C is redundant then $C\kappa$ is trivially redundant.

For the converse we know that there exist instances $C_1{\cdot}\sigma_1$, ..., $C_n{\cdot}\sigma_n$ of clauses in D such that $C_1\sigma_1, \ldots, C_n\sigma_n \models C\kappa$ and $C{\cdot}\kappa \succ^i C_i{\cdot}\sigma_i$. Moreover, the σ_i are substitutions $\sigma_i : X \to T_{\Sigma(K)}$; therefore, there exist substitutions $\tau_i : X \to T_{\Sigma(X)}$ such that $\sigma_i = \tau_i\kappa$.

Let $C{\cdot}\sigma$ be any instance of C such that $\sigma : X \to T_{\Sigma(K)}$ is a substitution. There exists a substitution τ such that $\sigma = \tau\kappa$, and the instances $C_1{\cdot}\tau_1\tau\kappa$, ..., $C_n{\cdot}\tau_n\tau\kappa$ of clauses in D can be used to prove that $C{\cdot}\sigma$ is redundant. The definition of the reduction ordering ensures that $C{\cdot}\iota \succ^i C_i{\cdot}\tau_i$, and lemma 5.2 implies $C_1\tau_1\tau\kappa, \ldots, C_n\tau_n\tau\kappa \models C\tau\kappa$. The proof for inferences is similar. $\square$

5.2 The Inference System

For the case of terms with variables we will use the inference system $\mathcal{S}_S^{\Sigma(K)}$, which is the same as $\mathcal{S}_S$, but for the skolemized signature. This ensures that ground substitutions exist.

Lemma 5.7 *D_0, D_1, D_2, ... is a theorem proving derivation that is fair with respect to $\mathcal{S}_S^{D(K)}$ if and only if $D_0(K)$, $D_1(K)$, $D_2(K)$, ... is a ground theorem proving derivation that is fair with respect to $\mathcal{S}_S^{D(K)}$.*

Lemma 5.8 *D is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ if and only if $D(K)$ is ground complete with respect to $\mathcal{S}_S^{\Sigma(K)}$.*

Lemma 5.9 *$\mathcal{S}_S^{\Sigma(K)}$ is strictly decreasing.*

Definition 5.10 (Completion procedure)
Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause C that is not reductive. The *completion procedure* for $\mathcal{S}_S$ generates a theorem proving derivation N_0, N_1, N_2, ..., using the ground completion procedure for $\mathcal{S}_S^{\Sigma(K)}$. Additionally, we require that the completion procedure fails if some N_i contains a nonreductive unit clause that is not known to be redundant. $\diamond$

Corollary 5.11 *Let S be a selection function such that S contains a pair $\langle C, \Gamma \rangle$ for any clause C that is not reductive and has a nonempty antecedent. Then the completion procedure for $\mathcal{S}_S$ either generates a theorem proving derivation N_0, N_1, N_2, ..., which is fair with respect to $\mathcal{S}_S$ or the procedure fails and N_∞ contains a nonreductive unit clause.*

5.3 Computing in Complete Specifications

In earlier work on horn clause completion, completeness is defined as confluence of rewriting. The motivation for this definition is that for such a specification it is decidable whether the specification implies an equation. On the other hand, we have used the approach that has been developed for first-order completion, which defines completeness as being closed up to redundancy with respect to some inference system. We will now show that under some reasonable additional assumptions our notion of completeness implies confluence.

Definition 5.12 (Linear inference)
Let N be a set of clauses. A strict superposition or selective paramodulation inference

$$\frac{B' \quad B}{C \cdot \mu}$$

is called *N-linear*, if B' is in N and B is not in N. $\diamond$

Theorem 5.13 *Let $N \cup M$ be a finite set of clauses that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$, where all clauses in N are rewrite clauses and S contains a pair $\langle C, \Gamma \rangle$ for any clause C in M. Furthermore, let G be a goal clause. Then it is decidable by N-linear strict left superposition, N-linear selective paramodulation and ordered/selective equality resolution whether $N \cup M \cup \{G\}$ is inconsistent.*

Proof: Lemma 5.8 and lemma 5.5 ensure that $N \cup M$ is ground complete and that all clauses in N are ground reductive. By lemma 5.1, $D \models G$ if and only if $D(K) \models G\kappa$. Therefore, we will seek a refutation for $N \cup M \cup \{G\kappa\}$.

We first show that the usual method of adding the conclusions of essential inferences leads to a derivation where any clause not in $N \cup M$ has an empty succedent. Since $N \cup M$ is complete, all inferences from clauses in $N \cup M$ are redundant. No strict superposition or selective paramodulation inference is possible between goal clauses. For any equality resolution inference that has goal clause as premise, the conclusion is also a goal clause. The remaining cases are N-linear strict left superposition and N-linear selective paramodulation inferences of the form

$$\frac{B' \quad B}{C{\cdot}\mu}$$

where B' is in N, B is not in $N \cup M$ and C has an empty succedent.

Therefore, the maximal premise of any essential inference is not in $N \cup M$ and is a ground goal clause. In particular, no nonreductive unit clause appears in the derivation. From the restrictions on variable occurrences for reductive clauses we know that the conclusion of such an inference is again a ground goal clause. Since any ground instance of N is reductive and since any clause not in $N \cup M$ is ground we know that in order to check whether an inference rule is applicable we only have to determine a maximal term of the ground clause and match it against the clauses in N. That is decidable. Since $\mathcal{S}_S^{\Sigma(K)}$ is strictly decreasing, the conclusion of any inference is smaller than the maximal premise. As $\succ^i$ is well-founded, the theorem proving derivation terminates. □

In practice, the rewrite clauses in N are used for recursive conditional rewriting; the clauses in M need not be considered. Since any valid equation can be proved by rewriting, the set of rewrite clauses is confluent.

Theorem 5.14 *Let $N \cup M$ be a finite set of clauses that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$, where all clauses in N are reductive and S contains a pair $\langle C, \Gamma\rangle$ for any clause C in M. Furthermore, let $u \approx v$ be an equation. Then $N \cup M \models u \approx v$ is decidable by N-linear strict left superposition, N-linear selective paramodulation and selective equality resolution.*

Proof: By theorem 5.13 it is decidable whether $N \cup M \cup \{u\kappa \approx v\kappa \rightarrow\}$ is inconsistent. If and only if this is the case we have $N \cup M \models u \approx v$. □

Example 5.15
Consider the specification for natural numbers and addition.

natplus =
  sorts
    nat
  ops
    $0 : \ \rightarrow \mathrm{nat}$
    $\mathrm{s} : \mathrm{nat} \times \mathrm{nat} \rightarrow \mathrm{nat}$
    $+ : \mathrm{nat} \times \mathrm{nat} \rightarrow \mathrm{nat}$
  axioms $\forall\, \mathrm{m}, \mathrm{n} : \mathrm{nat}$

$$0 + \mathrm{n} \approx \mathrm{n} \tag{1}$$
$$\mathrm{s(m)} + \mathrm{n} \approx \mathrm{s(m+n)} \tag{2}$$

The specification is complete with respect to $\mathcal{M}$, since no overlaps exist. Suppose we would like to prove $\mathrm{s(0)} + \mathrm{s(n)} \approx \mathrm{s(s(n))}$. We get the following theorem proving derivation, where $N = \{(1), (2)\}$.

$N_0 = N \cup \{(3) \quad \mathrm{s(0)} + \mathrm{s(k)} \approx \mathrm{s(s(k))} \rightarrow\}$	goal clause to refute
$N_1 = N \cup \{(3), (4) \quad \mathrm{s(0} + \mathrm{s(k))} \approx \mathrm{s(s(k))} \rightarrow\}$	strict left superposition (2) on (3)
$N_2 = N \cup \{(4)\}$	(3) redundant because of (2) and (4)
$N_3 = N \cup \{(4), (5) \quad \mathrm{s(s(k))} \approx \mathrm{s(s(k))} \rightarrow\}$	strict left superposition (1) on (4)
$N_4 = N \cup \{(5)\}$	(4) redundant because of (2) and (5)
$N_5 = N \cup \{(5), \Box\}$	equality resolution (5)

Since we were able to refute the goal, the equation is valid. Note that deduction and deletion steps alternate, which we could combine into simplification steps. The derivation looks like $N \cup G_0$, $N \cup G_1$, $N \cup G_2$, $\ldots$, and can be viewed as a sequence of rewrite steps in the goal.

Now we try to prove $s(0)+s(n) \approx s(n)$. The derivation is similar to the previous one, except that the last step by equality resolution is no longer possible.

$$N_0 = N \cup \{s(0) + s(k) \approx s(k) \rightarrow\}$$
$$\vdots$$
$$N_4 = N \cup \{s(s(k)) \approx s(k) \rightarrow\}$$

N_4 is complete with respect to $\mathcal{M}$ and does not contain the empty clause. Therefore, it is not inconsistent and $s(s(k)) \approx s(k)$ is not a consequence of N. $\Diamond$

Lemma 5.16 *Let $N \cup M$ be a set of clauses that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$, where all clauses in N are reductive and S contains a pair $\langle C, \Gamma \rangle$ for any clause C in M. A unit clause $u \approx v$ is redundant in N if and only if it is a consequence of N.*

Proof: Essentially, we will show that a rewrite proof exists for $u\kappa \approx v\kappa$. In this context, a rewrite proof is a multiset of instances of rewrite clauses in N which implies $u\kappa \approx v\kappa$. Because of the low complexity we have assigned to rewrite clauses, all these instances will be smaller than $(u \approx v){\cdot}\kappa$. By lemma 5.6 we conclude that $u \approx v$ is redundant.

Let $u\kappa \approx v\kappa$ be a minimal unit clause that is a consequence of $M \cup N$ but not redundant in N. Because it is not redundant it cannot be trivial. Due to theorem 5.14 there exists a refutation of $M \cup N \cup \{u\kappa \approx v\kappa \rightarrow\}$ using only N-linear strict left superposition, N-linear selective paramodulation and ordered/selective equality resolution. Because of $u \neq v$, the first inference of the refutation has the form

$$\frac{[B' =]\, \Lambda \rightarrow l \approx r \qquad [B =]\, u\kappa[l\mu] \approx v\kappa \rightarrow}{[C =]\, (\Lambda, u\kappa[r] \approx v\kappa \rightarrow){\cdot}\mu}$$

where B' is a clause in N. The term $r\mu$ is equal to $r'\kappa$ for some term r', and $\Lambda\mu$ is equal to $\Lambda'\kappa$ for some Λ'. Since B' is reductive, $u[r'] \approx v$ and all unit clauses corresponding to equations in Λ' are smaller than $u \approx v \rightarrow$. By assumption, they are redundant in N; thus, there exist instances of clauses $C_1 \cdot \sigma_1\kappa, \ldots, C_n \cdot \sigma_n\kappa$ implying those unit clauses which are smaller than $u \approx v \cdot \kappa$. The instances $B' \cdot \mu\kappa, C_1 \cdot \sigma_1\kappa, \ldots, C_n \cdot \sigma_n\kappa$ can be used to prove $u \approx v \cdot \kappa$ redundant, which contradict our assumption. Note that in the case $l = u$ we need the fact that B' is a rewrite clause to ensure that $u\kappa \approx v\kappa$ is greater than $B'\kappa$. $\square$

5.4 Construction of the Free Model

Theorem 5.17 *Let D be a specification that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and does not contain the empty clause. Then $T_{\Sigma(K)}/I_{D(K)}$ is a free model of D.*

Proof: $D(K)$ is ground complete with respect to $\mathcal{S}_S^{\Sigma(K)}$; therefore, $T_{\Sigma(K)}/I_{D(K)}$ is initial in $\mathrm{Mod}(D(K))$ (cf. theorem 4.20). This is equivalent to the fact that $T_{\Sigma(K)}/I_{D(K)}$ is a free model of D over K. And since there exists an isomorphism between $T_{\Sigma(K)}$ and $T_{\Sigma(X)}$, it is also free over X. $\square$

6 Theorem Proving by Consistency

In this section we investigate another way to use the theorem prover introduced in section 3. The basis for developing it was refutation theorem proving, where theorems are proved by showing that their negation is inconsistent to the given specification. We will introduce the method of *proof by consistency*, where theorems are proved by showing that they essentially do not change the given specification.

First we clarify what is meant by inductive theorem, since for clauses validity in the initial model and validity in all term generated models differ. We introduce four classes of models, which differ in the clauses they satisfy: all models, term generated models, free models and initial models. In parallel, there are four theories associated with a specification, which differ in the syntactic form of clauses they contain: all clauses, ground clauses, unit clauses and ground unit clauses. We characterize validity in the four model classes by showing that it is equivalent to consistency. A set of clauses is consistent to a specification if the combined specification has the same models or, equivalently, the same respective theory.

We use this as the basis for our first proof by consistency procedure, which proves validity of a clause in the free model. We show that the procedure is refutationally complete and linear.

By adding type predicates to the specification we are able to encode the condition that a term is ground. We use this to translate validity in the free model to validity in the initial model. The combination of this translation with the proof procedure for the free model yields a method for proving validity in the initial model. Again, the method is refutationally complete and linear.

As another benefit of the translation we can express sufficient completeness of a specification, and prove it by the same method as above.

A refinement of the method which restricts the selection of literals for superposition gives a proof method for theorems valid in all term generated models. This method is still linear but no longer refutationally complete.

6.1 Validity and Consistency

In this section we will discuss the relation between four classes of models for a specification, for the case of horn clauses.

Definition 6.1
Let D be a specification. We define:

1. $\mathrm{Mod}(D) = \{ A \mid A \text{ is a model of } D \}$;
2. $\mathrm{Gen}(D) = \{ A \mid A \text{ is a term generated model of } D \}$;
3. $\mathrm{Free}(D) = \{ A \mid A \text{ is a free model of } D \text{ over } X \}$;
4. $\mathrm{Init}(D) = \{ A \mid A \text{ is an initial model of } D \}$. $\diamond$

Note that for an inconsistent specification D all model classes are empty. If D is consistent, we will use T_D and $T_{D(X)}$ as *the* initial and free models, respectively. This is legitimate, since all initial (free) models are isomorphic. Therefore, we will speak of *the* initial model.

For a single equation being an *inductive theorem* has a well defined meaning, since an equation is valid in the initial model of a specification if and only if it is valid in all term generated models. This is no longer the case if we regard clauses. There exist clauses which are valid in the initial model but not in the term generated models. In an arbitrary model an equation in the antecedent of a clause may become true, even if this is not required by the specification, resulting in the clause being not valid in this model. We give a much simpler example than Padawitz [Pad88] for this difference. The example also shows that the same difference exists between validity in all models and in the free model; the reason for this is that being term generated and being free are orthogonal concepts.

Example 6.2 (Injectivity of successor)
Let D be the specification for natural numbers

natbase =
    sorts
        nat
    ops
        $0 : \ \to \mathrm{nat}$
        $\mathrm{s} : \mathrm{nat} \to \mathrm{nat}$

Consider the clause $C = \ \mathrm{s(m)} \approx \mathrm{s(n)} \ \to \ \mathrm{m} \approx \mathrm{n}$, which describes injectivity of s. C is valid in $\mathrm{Init}(D)$, which is isomorphic to $\mathbf{N}$, since in $\mathbf{N}$ the successor function is injective. $\mathrm{Free}(D)$ is isomorphic to the algebra A defined by

$$\begin{aligned} \mathrm{nat}_A &= \{\, \langle k, n\rangle \mid k \in \{0\} \cup K_{\mathrm{nat}}, n \in \mathbf{N}\,\} \\ \mathrm{s}_A\langle k, n\rangle &= \langle k, n+1\rangle \end{aligned}$$

It is easy to verify that s_A is injective in this case, too. C is not valid in $\mathrm{Mod}(D)$ or $\mathrm{Gen}(D)$. To demonstrate this we define the algebra A' by

$$\begin{aligned} \mathrm{nat}_{A'} &= \{0, 1\} \\ 0_{A'} &= 0 \\ \mathrm{s}_{A'}(0) &= 1 \\ \mathrm{s}_{A'}(1) &= 1 \end{aligned}$$

Then A' is a term generated model of D and the assignment $\alpha : X \to A'$ with $\alpha(\mathrm{m}) = 0$ and $\alpha(\mathrm{n}) = 1$ does not satisfy C. $\diamond$

Therefore, we will reserve "inductive theorem" for unit clauses. For general horn clauses we have to distinguish between the different notions; instead, we will speak of validity in the different model classes. The following diagram summarizes the relations between these:

$$\begin{array}{ccc} D \models C & \underset{(3)}{\overset{(1)}{\substack{\Longrightarrow \\ \not\Longleftarrow}}} & \mathrm{Gen}(D) \models C \\ (1)\Downarrow \ \not\Uparrow(4) & & (1)\Downarrow \ \not\Uparrow(4) \\ \mathrm{Free}(D) \models C & \underset{(3)}{\overset{(2)}{\substack{\Longrightarrow \\ \not\Longleftarrow}}} & \mathrm{Init}(D) \models C \end{array}$$

We have (1) because of set inclusion of the model classes. For (2) note that Init(D) is a subalgebra of Free(D). To show (3) take any unit clause that is an inductive theorem but not a deductive theorem, e.g., commutativity of + for natural numbers. Example 6.2 shows (4).

Now that we know that all four possible semantics we might associate with a specification are different, we will investigate their relation in more detail. First we show that validity in the class of term generated models, validity in the initial model and validity in the free model can all be reduced to validity in the class of all models.

Theorem 6.3 *Let D be a specification and let $C = \Gamma \to \Delta$ be a horn clause.*

1. $\mathrm{Gen}(D) \models C$ *if and only if* $D \models C\sigma$ *for all ground substitutions* σ.
2. $\mathrm{Free}(D) \models C$ *if and only if* $D \models \Gamma\sigma$ *implies* $D \models\ \to \Delta\sigma$ *for all substitutions* $\sigma : X \to T_{\Sigma(K)}$.
3. $\mathrm{Init}(D) \models C$ *if and only if* $D \models \Gamma\sigma$ *implies* $D \models\ \to \Delta\sigma$ *for all ground substitutions* σ.

We write $D \models\ \to \Delta\sigma$ instead of $D \models \Delta\sigma$ to ensure that the case of an empty Δ is handled correctly.

Proof: If D is inconsistent the equivalence obviously holds in all cases. Therefore, let us assume from now on that D is consistent.

(1) Let A be a model of D and let σ be a ground substitution. Being a ground clause $C\sigma$ contains only ground equations, which are valid in A if and only if they are valid in A_{gen}. But C and also its instance $C\sigma$ are valid in A_{gen}.

For the converse let A be a term generated model of D, and let α be an assignment $X \to A$. There exists a ground substitution σ such that $\alpha(x) = \#_A(x\sigma)$ for all $x \in X$. $A \models_\alpha u \approx v$ if and only if $A \models u\sigma \approx v\sigma$ for all equations $u \approx v$. Therefore, $A \models_\alpha C$ if and only if $A \models C\sigma$, which is true by assumption.

(3) Let σ be a ground substitution, A a model of D, and assume that $D \models \Gamma\sigma$. Δ cannot be empty, otherwise D would be inconsistent. Since $\Gamma\sigma$ contains only ground equations, this implies $\mathrm{Init}(D) \models \Gamma\sigma$. Because C is valid in Init(D), this in turn implies $\mathrm{Init}(D) \models \Delta\sigma$ and $D \models\ \to \Delta\sigma$.

For the converse let $\alpha : X \to \mathrm{Init}(D)$ be an assignment that satisfies Γ. By lemma 2.35 there exists a substitution $\sigma : X \to T_\Sigma$ such that $\alpha = \sigma\,;[\_]$, and we have $D \models \Gamma\sigma$. Again, Δ cannot be empty. This implies $D \models \Delta\sigma$, and finally $\mathrm{Init}(D) \models \Delta\sigma$.

The proof of (2) becomes identical to that of (3), if we add skolem constants to the specification. □

We now associate a *theory* to a class of models $\mathcal{D}$, which contains clauses which are valid in $\mathcal{D}$. By restricting the syntactical structure of the clauses we obtain different notions of theory.

Definition 6.4
Let $\mathcal{D}$ be a class of Σ-algebras.
The *theory* of $\mathcal{D}$ is

$$\mathrm{Th}(\mathcal{D}) = \{\, C \mid C \text{ is a clause and } \mathcal{D} \models C \,\}.$$

The *ground theory* of $\mathcal{D}$ is

$$\mathrm{GTh}(\mathcal{D}) = \{\, C \mid C \text{ is a ground clause and } \mathcal{D} \models C \,\}.$$

The *unit theory* of $\mathcal{D}$ is

$$\mathrm{UTh}(\mathcal{D}) = \{\, C \mid C \text{ is a unit clause or the empty clause and } \mathcal{D} \models C \,\}.$$

The *ground unit theory* of $\mathcal{D}$ is

$$\mathrm{GUTh}(\mathcal{D}) = \{\, C \mid C \text{ is a ground unit clause or the empty clause and } \mathcal{D} \models C \,\}.$$

We write [G][U]Th(D) for [G][U]Th(Mod(D)). $\diamond$

We will show that the four theories associated with a specification D are intimately tied to the four semantics we have introduced above. In fact, for most properties of Mod and Th there exist analogous properties of Gen and GTh, Free and UTh, or Init and GUTh. We will just give those properties we will need later on. Note that D is inconsistent if and only if all theories contain the empty clause.

Lemma 6.5 *Let D be a specification.*

1. $\mathrm{Th}(\mathrm{Mod}(D)) = \mathrm{Th}(D)$.
2. $\mathrm{GTh}(\mathrm{Gen}(D)) = \mathrm{GTh}(D)$.
3. $\mathrm{UTh}(\mathrm{Free}(D)) = \mathrm{UTh}(D)$.
4. $\mathrm{GUTh}(\mathrm{Init}(D)) = \mathrm{GUTh}(D)$.

Proof: (1) This is just the definition of Th(D).

(2) We only need to show $\mathrm{GTh}(\mathrm{Gen}(D)) \subseteq \mathrm{GTh}(D)$. Let C be any ground clause in $\mathrm{GTh}(\mathrm{Gen}(D))$ and let A be an algebra in Mod(D). Then C contains only ground equations, which are valid in A if and only if they are valid in A_{gen}. But C is valid in A_{gen}.

(3),(4) These are true because of the construction of Init(D) and Free(D). $\Box$

Theorem 6.6 *Let D be a specification and H a set of clauses.*

1. $D \models H$ *if and only if* $\mathrm{Mod}(D) = \mathrm{Mod}(D \cup H)$.
2. $\mathrm{Gen}(D) \models H$ *if and only if* $\mathrm{Gen}(D) = \mathrm{Gen}(D \cup H)$.
3. $\mathrm{Free}(D) \models H$ *if and only if* $\mathrm{Free}(D) = \mathrm{Free}(D \cup H)$.
4. $\mathrm{Init}(D) \models H$ *if and only if* $\mathrm{Init}(D) = \mathrm{Init}(D \cup H)$.

Proof: If D is inconsistent all four cases hold. Therefore, assume that D is consistent.

(1),(2) Let A be any [term generated] model of D. Then all clauses in H are true in A and A is a [term generated] model of $D \cup H$. For the converse any model A of D is also a model of $D \cup H$; therefore, H is true in A.

(3) If $\mathrm{UTh}(D \cup H)$ contained the empty clause, $D \cup H$ would be inconsistent. Then, no model A of D could satisfy H, in contradiction to $\mathrm{Free}(D) \models H$. Therefore, $\mathrm{UTh}(D \cup H)$ does not contain the empty clause. Let $u \approx v$ be a unit clause in $\mathrm{UTh}(D \cup H)$ that is not in $\mathrm{UTh}(D)$. There exists a linear paramodulation refutation for $u \approx v \rightarrow$ with respect to $D \cup H$. We may assume that $u \approx v$ has the shortest refutation of all such equations.

1. If the first inference of the refutation is an equality resolution of the form

$$\frac{u \approx v \rightarrow}{\Box}$$

then $u \approx v$ is trivial and in $\mathrm{UTh}(D)$.

2. Otherwise the first inference of the refutation is a linear paramodulation inference of the form

$$\frac{u[l\sigma] \approx v \rightarrow}{\Lambda\sigma, u[r\sigma] \approx v \rightarrow}$$

where the clause $C = \Lambda \rightarrow l \approx r$ has been applied. Then the rest of the proof provides shorter refutations for $u[r\sigma] \approx v$ and all equations in $\Lambda\sigma$; therefore, they are in $\mathrm{UTh}(D)$. If C is in D the refutation applies only clauses in D and $u \approx v$ is in $\mathrm{UTh}(D)$. Otherwise, C is in H. Since C and $\Lambda\sigma$ are valid in $\mathrm{Free}(D)$, this implies that $l\sigma \approx r\sigma$ is valid in $\mathrm{Free}(D)$, too. Thus $u \approx v$ is in $\mathrm{UTh}(D)$.

For the converse note that H is valid in $\mathrm{Free}(D \cup H)$ which is equal to $\mathrm{Free}(D)$.

(4) The proof is identical to that for (3), except that $u \approx v$ and σ are ground. □

Theorem 6.7 *Let D be a specification and H a set of clauses.*

1. $\mathrm{Mod}(D) = \mathrm{Mod}(D \cup H)$ *if and only if* $\mathrm{Th}(D) = \mathrm{Th}(D \cup H)$.
2. $\mathrm{Gen}(D) = \mathrm{Gen}(D \cup H)$ *if and only if* $\mathrm{GTh}(D) = \mathrm{GTh}(D \cup H)$.
3. $\mathrm{Free}(D) = \mathrm{Free}(D \cup H)$ *if and only if* $\mathrm{UTh}(D) = \mathrm{UTh}(D \cup H)$.
4. $\mathrm{Init}(D) = \mathrm{Init}(D \cup H)$ *if and only if* $\mathrm{GUTh}(D) = \mathrm{GUTh}(D \cup H)$.

Proof: If D is inconsistent all four cases hold. Therefore, assume that D is consistent and that none of the theories contains the empty clause.

(1) We have $\mathrm{Th}(D \cup H) = \mathrm{Th}(\mathrm{Mod}(D \cup H)) = \mathrm{Th}(\mathrm{Mod}(D)) = \mathrm{Th}(D)$. For the converse let A be any model of D. Then all clauses in $\mathrm{Th}(D) = \mathrm{Th}(D \cup H)$ are valid in A, including H. Therefore, A is a model of $D \cup H$.

(2) We have $\mathrm{GTh}(D \cup H) = \mathrm{GTh}(\mathrm{Gen}(D \cup H)) = \mathrm{GTh}(\mathrm{Gen}(D)) = \mathrm{GTh}(D)$. For the converse let A be any term generated model of D. Then all clauses in $\mathrm{GTh}(D) = \mathrm{GTh}(D \cup H)$ are valid in A, including all ground instances of H. By theorem 6.3, A is a model of $D \cup H$.

(3),(4) These are direct consequences of the construction of $\mathrm{Init}(D)$ and $\mathrm{Free}(D)$, respectively. □

Theorem 6.8 *Let D be a specification and H a set of clauses.*

1. $D \models H$ *if and only if* $\mathrm{Th}(D) = \mathrm{Th}(D \cup H)$.
2. $\mathrm{Gen}(D) \models H$ *if and only if* $\mathrm{GTh}(D) = \mathrm{GTh}(D \cup H)$.
3. $\mathrm{Free}(D) \models H$ *if and only if* $\mathrm{UTh}(D) = \mathrm{UTh}(D \cup H)$.
4. $\mathrm{Init}(D) \models H$ *if and only if* $\mathrm{GUTh}(D) = \mathrm{GUTh}(D \cup H)$.

Proof: Paste together theorem 6.6 and theorem 6.7. □

The last three theorems provide the basis for the proof method called "proof by consistency". We say a set of clauses H is *consistent* to a specification D if the addition of H to D does not change the class of models (or the theory, which is equivalent as we have shown above).

Proof by consistency is not intended for the case of all models, since for this case other proof methods exist. Rather, it should be seen as a convenient means to introduce induction into the framework of theorem proving by completion. There are two different kinds of inductive arguments at work here. For the case of term generated models, it is induction on the term structure, the kind of induction that is most commonly used. On the other hand, for the case of free models, only provable equations are true. Therefore, we may use induction to verify that for all solutions of the antecedent of a clause the succedent is valid. Then the clause is valid in all free models. In this case, induction operates on possible proofs of the antecedent of a clause. The good news is that our completion procedure already uses this kind of induction when it proves that clauses for which literals have been selected are not needed for rewriting.

6.2 The Proof Procedure for $\mathrm{Free}(D)$

Theorem 6.9 *Let D be a specification and let H be a set of horn clauses such that $D \cup H$ is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$. Moreover, suppose that S contains a pair $\langle C, \Gamma\rangle$ for any clause C in H. Then* $\mathrm{Free}(D) \models H$.

Proof: If $D \cup H$ is inconsistent, the empty clause must be in D, since no equations can be selected in its antecedent. Then $\mathrm{Free}(D)$ is empty and the theorem holds. Therefore, assume that $D \cup H$ is consistent. We know that $(D \cup H)(K)$ is ground complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ (cf. lemma 5.8); therefore, $I_{(D\cup H)(K)}$ is a model of $D \cup H$ and also of D. The clauses in H do not generate any rules in $R_{(D\cup H)(K)}$ (cf. lemma 4.16). Whether the instance of a clause generates a rule depends only on the previously generated rules; therefore, $D(K)$ and $(D \cup H)(K)$ generate the same sets of rules. Thus $I_{D(K)}$ and $I_{(D\cup H)(K)}$ are equal and $I_{D(K)}$ satisfies all clauses in D, even though D is not required to be complete. The free model of D is isomorphic to $T_{\Sigma(K)}/I_{D(K)}$; since it remains unchanged by adding H, the conclusion follows from theorem 6.6. □

Example 6.10 (Injectivity of successor, cont.)
We use the criterion of theorem 6.9 to prove that $H = \{\underline{\mathrm{s(m)} \approx \mathrm{s(n)}} \rightarrow \mathrm{m} \approx \mathrm{n}\}$ is valid in the free model of the specification of natural numbers (cf. example 6.2). We will indicate the selected equations in the antecedent of a clause by underlining them. Both D, which contains no clause, and $D \cup H$ are complete with respect to $\mathcal{S}_S^{\Sigma(K)}$. Hence $\mathrm{Free}(D) \models H$. ◇

We are now ready to state the proof procedure for $\mathrm{Free}(D)$. It will serve as the basis for the proof procedures we develop and is refutationally complete and linear. Furthermore, if the clauses are not valid in the free model the procedure generates an inconsistency witness.

Definition 6.11 (Proof procedure for $\mathrm{Free}(D)$)
Let D be finite a specification that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma\rangle$ for any non-unit clause not in D and any non-rewrite clause in D. The *proof procedure for* $\mathrm{Free}(D)$ takes a set of clauses H as input and generates a theorem proving derivation $D \cup H = N_0, N_1, N_2, \ldots$, using the completion procedure for $\mathcal{S}_S^{\Sigma(K)}$. If the completion procedure fails the inductive proof procedure answers "Disproof"; if the completion procedure terminates the inductive proof

procedure answers "Proof". Additionally, the completion procedure has to obey the following restrictions:

1. All inferences are known to be strictly decreasing.
2. All unit clauses which are redundant in D are known to be redundant.
3. Clauses are deleted if they are known to be redundant.
4. If some N_i contains a unit clause that is not redundant in D the completion procedure fails.
5. If some N_i contains the empty clause the completion procedure fails. $\diamond$

Note that all unit clauses in D must be rewrite clauses; therefore, D cannot contain nonreductive unit clauses.

Theorem 6.12 *Let D be a finite specification that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma\rangle$ for any non-unit clause not in D and any non-rewrite clause in D. Let H be a set of horn clauses. Suppose the proof procedure for* $\mathrm{Free}(D)$ *generates a theorem proving derivation $N_0, N_1, N_2, \ldots$ if H is given as input.*

1. *The clauses in H are valid in the free model of D if and only if the proof procedure answers "Proof" or does not terminate.*
2. *If the proof procedure answers "Disproof", N_∞ contains a clause which is in* $\mathrm{UTh}(D \cup H)$ *but not in* $\mathrm{UTh}(D)$.
3. *Any inference from non-unit premises in N_i by strict superposition or selective paramodulation which is not D-linear is redundant in D.*

Proof: Suppose the proof procedure answers "Proof" or does not terminate. Then the completion procedure generates a fair derivation and N_∞ is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$. For any clause in $N_\infty \setminus D$ the selection function S contains a pair $\langle C, \Lambda\rangle$. By theorem 6.9 all clauses in N_∞ are valid in $\mathrm{Free}(D)$.

If the proof procedure answers "Disproof", N_∞ contains a clause C that is either a non-redundant unit clause or the empty clause. In both cases we have $D \not\models C$ (cf. lemma 5.16), but since N_∞ is equivalent to $D \cup H$, we have $D \cup H \models C$. Thus C is contained in $\mathrm{UTh}(D \cup H)$ but not in $\mathrm{UTh}(D)$.

All inferences between clauses in D are redundant, since D is complete. The only inference rules applicable to non-unit clauses in H_i are selective paramodulation and selective equality resolution, where the first premise is in D and the second premise is in H_i, since for any clause C in H_i the selection function S contains a pair $\langle C, \Gamma\rangle$. $\square$

Note that if D is incomplete proofs are still valid; however, since we no longer can decide if a unit clause is redundant in D, we loose refutation completeness.

Example 6.13 (Antisymmetry)
Consider the specification for ordered natural numbers.

natle $=$ **natbase** $+$

ops

$\leq : \mathrm{nat} \times \mathrm{nat}$

axioms $\forall\, \mathrm{m}, \mathrm{n} : \mathrm{nat}$

$$0 \leq \mathrm{n} \tag{1}$$

$$\mathrm{m} \leq \mathrm{n} \rightarrow \mathrm{s(m)} \leq \mathrm{s(n)} \tag{2}$$

The specification is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ if S does not select the literal in the antecedent of (2). As the reduction ordering we use the lpo defined by $\leq \succ s \succ 0$. We will prove that $\leq$ is antisymmetric in the free model of this specification. We get the derivation

(3)	$\underline{\mathrm{m} \leq \mathrm{n}}, \mathrm{n} \leq \mathrm{m} \rightarrow \mathrm{m} \approx \mathrm{n}$	clause to be proved
(4)	$\underline{\mathrm{n} \leq 0} \rightarrow 0 \approx \mathrm{n}$	selective resolution (1) on (3)
(5)	$\mathrm{m} \leq \mathrm{n}, \underline{\mathrm{s(n)} \leq \mathrm{s(m)}} \rightarrow \mathrm{s(m)} \approx \mathrm{s(n)}$	selective resolution (2) on (3)
	$0 \leq 0 \rightarrow 0 \approx 0$	selective resolution (1) on (4), tautology
	$\mathrm{m} \leq \mathrm{n}, \mathrm{n} \leq \mathrm{m} \rightarrow \mathrm{s(m)} \approx \mathrm{s(n)}$	selective resolution (2) on (5), redundant because of (3)

which proves that $\leq$ is indeed antisymmetric in the free model. $\diamond$

6.3 A Specification Transformation for Inductive Proofs

In order to use the method we have available for theorem proving in the free model for inductive theorem proving, we will now introduce a specification transformation that enables us to restrict applicability of a clause to its ground instances. This will reduce the problem of determining validity of a clause in the initial model to validity in the free model. Therefore, the procedure for proving validity in the initial model will inherit refutation completeness and linearity from the proof procedure for Free(D).

The specification transformation which we introduce essentially encodes the induction on term structure into induction on proofs for equations in the antecedent.

Definition 6.14 (Base signature)
Let $D = \langle \Sigma, N \rangle$ be a specification with signature $\Sigma = \langle S, \Omega \rangle$ and let $\Sigma_B = \langle S, \Omega_B \rangle$ be an inhabited subsignature of Σ. We call Σ_B a *base signature* for D. $\diamond$

Since the sorts and the arities of the operators are already specified in the signature of the specification Σ, we will give concrete base signatures as the set of operators they contain, e.g., $\{0, \mathrm{s}\}$.

Definition 6.15 (Specification transformation)
Let $D = \langle \Sigma, N \rangle$ be a specification with signature $\Sigma = \langle S, \Omega \rangle$ and let Σ_B be a base signature for D. We introduce for each sort s in S a new predicate $\mathrm{gnd}_s : s$ and for each operator $\omega : s_1, \ldots, s_n \rightarrow s$ in Ω_B a clause

$$\omega^g = \mathrm{gnd}_{s_1}(x_1), \ldots, \mathrm{gnd}_{s_n}(x_n) \rightarrow \mathrm{gnd}_s(\omega(x_1, \ldots, x_n))$$

where the $x_i \in X_{s_i}$ are distinct variables. The *transformed specification* for D with respect to Σ_B is the specification $D^{g(\Sigma_B)} = \langle \Sigma^g, N^{g(\Sigma_B)} \rangle$ where

$$\begin{aligned} \Sigma^g &= \langle S, \Omega^g \rangle, \\ \Omega^g &= \Omega \cup \{ \mathrm{gnd}_s : s \mid s \in S \} \text{ and} \\ N^{g(\Sigma_B)} &= N \cup \{ \omega^g \mid \omega \in \Omega_B \}. \end{aligned}$$

The *transformed specification* for D will be denoted by D^g if $\Sigma_B = \Sigma$. There is a SPEC-inclusion $\gamma_D : D \hookrightarrow D^{g(\Sigma_B)}$. ◇

We will omit the sort information from $\mathrm{gnd}_s(t)$ where it is not needed and simply write $\mathrm{gnd}(t)$. The same transformation can be used for unsorted specifications; however, if atoms are simulated by equations persistency will be lost.

Example 6.16 (Natural numbers)
Let the following be a specification for natural numbers with addition.

natplus=
  sorts
    nat
  ops
    $0 : \ \to \mathrm{nat}$
    $\mathrm{s} : \mathrm{nat} \to \mathrm{nat}$
    $+ : \mathrm{nat} \times \mathrm{nat} \to \mathrm{nat}$
  axioms $\forall\, \mathrm{m}, \mathrm{n} : \mathrm{nat}$
    $0 + \mathrm{n} \approx 0$ (1)
    $\mathrm{s}(\mathrm{m}) + \mathrm{n} \approx \mathrm{s}(\mathrm{m} + \mathrm{n})$ (2)

Let Σ_B be the subsignature that contains only the operators 0 and s. Then the transformed specification for **natplus** with respect to Σ_B is

$\mathbf{natplus}^{g(\Sigma_B)} = \mathbf{natplus}\ +$
  ops
    gnd : nat
  axioms $\forall\, \mathrm{m} : \mathrm{nat}$
    $\mathrm{gnd}(0)$ (3)
    $\mathrm{gnd}(\mathrm{m}) \to \mathrm{gnd}(\mathrm{s}(\mathrm{m}))$ (4)

It is complete since both clauses are reductive and there are no overlaps. Finally, the transformed specification for D is

$\mathbf{natplus}^{g} = \mathbf{natplus}^{g(\Sigma_B)}\ +$
  axioms $\forall\, \mathrm{m}, \mathrm{n} : \mathrm{nat}$
    $\underline{\mathrm{gnd}(\mathrm{m})}, \mathrm{gnd}(\mathrm{n}) \to \mathrm{gnd}(\mathrm{m} + \mathrm{n})$ (5)

It is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ if S selects the equation indicated by underlining. We will show this later when we discuss sufficient completeness. ◇

We now prove the two main properties of the specification transformation, namely that it does not change the meaning of the original specification and that $D^g \models \text{gnd}(t)$ if and only if t is a ground term in T_{Σ_B}, up to equivalence.

Lemma 6.17 *Let D be a specification and let Σ_B be a base signature for D. The specification inclusion $\gamma_D : D \hookrightarrow D^{g(\Sigma_B)}$ is persistent.*

Proof: We first show that $D^{g(\Sigma_B)}$ is consistent if D is consistent. Let A be an algebra in $\text{Mod}(D)$ and let A' be the strongly persistent γ_D-extension of A where $\text{gnd}_{s,A'}(a) = \top_{A'}$ for all elements a in s_A and sorts s in S. Obviously, A' satisfies $D^{g(\Sigma_B)}$ if A satisfies D.

By definition 2.39, the specification morphism γ_D is persistent if and only if it is complete and consistent.

Complete: No operators with range s in S are added by γ_D, therefore if t is a term in $T_{\Sigma^{g(\Sigma_B)}(X),s}$ then t is either a variable in X_s or has the form $\omega(t_1,\dots,t_n)$ and ω is an operator in Ω. Thus, the t_i are terms in $T_{\Sigma^{g(\Sigma_B)}(X),s_i}$ with sorts s_i in S. By induction t is in $T_{\Sigma(X),s}$.

Consistent: Let $u \approx v$ be an equation such that $D^{g(\Sigma_B)} \models u \approx v$ but not $D \models u \approx v$. There exists a linear paramodulation refutation for $u \approx v \rightarrow$ with respect to $D^{g(\Sigma_B)}$. We may assume that $u \approx v$ has a refutation of minimal length of all such equations.

1. If the first inference of the refutation is an equality resolution of the form

   $$\frac{u \approx v \rightarrow}{\Box}$$

   then $u \approx v$ is trivial and $D \models u \approx v$.

2. Otherwise the first inference of the refutation is a linear paramodulation inference of the form

   $$\frac{u[l\sigma] \approx v \rightarrow}{\Lambda\sigma, u[r\sigma] \approx v \rightarrow}$$

   where the clause $C = \Lambda \rightarrow l \approx r$ has been applied. Since u and v are in $T_{\Sigma(X)}$, the clause C is in D. The rest of the proof provides shorter refutations for $u[r\sigma] \approx v$ and all equations in $\Lambda\sigma$; therefore, they are consequences of D and there exist refutations for them. We can construct a linear paramodulation refutation for $u \approx v$ by using the inference above. Finally, we have $D \models u \approx v$. $\Box$

Lemma 6.18 *Let D be a specification and let t be a term in $T_{\Sigma(X)}$. Then $D^{g(\Sigma_B)} \models \text{gnd}(t)$ if and only if there exists a ground term $t' \in T_{\Sigma_B}$ with $D \models t \approx t'$.*

Proof: If D is inconsistent the lemma holds, since D is inhabited. Otherwise, let t be a term in $T_{\Sigma(X)}$ such that $D^{g(\Sigma_B)} \models \text{gnd}(t)$ but there exists no ground term $t' \in T_{\Sigma_B}$ with $D \models t \approx t'$. There exists a linear paramodulation refutation for $\text{gnd}(t) \rightarrow$ with respect to $D^{g(\Sigma_B)}$. We may assume that t has been selected so that $\text{gnd}(t)$ has a refutation of minimal length. Since $\text{gnd}(t) \approx \top$ is not trivial the first inference of the refutation cannot be by equality resolution.

1. Suppose the first inference of the refutation is a linear paramodulation inference of the form

   $$\frac{\text{gnd}(t[l\sigma]) \rightarrow}{\Lambda\sigma, \text{gnd}(t[r\sigma]) \rightarrow}$$

where the clause $C = \Lambda \to l \approx r$ has been applied. There exists a shorter refutation for $\mathrm{gnd}(t[r\sigma])$ and all equations in $\Lambda\sigma$. Therefore, there exists a ground term t' in T_{Σ_B} that is equivalent to $t[r\sigma]$. Because of $D \models \Lambda\sigma$ we have $D \models l\sigma \approx r\sigma$, and t' is equivalent to t.

2. Otherwise,the first inference of the refutation is a linear resolution inference of the form

$$\frac{\mathrm{gnd}(\omega(t_1, \ldots, t_n)) \to}{\mathrm{gnd}(t_1), \ldots, \mathrm{gnd}(t_n) \to}$$

where the clause $C = \mathrm{gnd}(x_1), \ldots, \mathrm{gnd}(x_n) \to \mathrm{gnd}(\omega(x_1, \ldots, x_n))$ has been applied and ω is an operator in Ω_B. Then the rest of the proof provides shorter refutations for $\mathrm{gnd}(t_i)$; therefore, there exist terms t'_i in T_{Σ_B} such that $D \models t_i \approx t'_i$. Since ω is in Ω_B the term $t' = \omega(t'_1, \ldots, t'_n)$ is in T_{Σ_B}; furthermore, we have $D \models t \approx t'$.

We prove the converse by induction on the structure of a ground term t' in T_{Σ_B} with $D \models t \approx t'$. t' has the form $\omega(t'_1, \ldots, t'_n)$ where the terms t'_i are in T_Σ and ω is an operator in Ω_B. By assumption $D^{g(\Sigma_B)} \models \mathrm{gnd}(t'_i)$ for $1 \leq i \leq n$. Since $D^{g(\Sigma_B)}$ contains the clause

$$\mathrm{gnd}(x_1), \ldots, \mathrm{gnd}(x_n) \to \mathrm{gnd}(\omega(x_1, \ldots, x_n))$$

we have $D^{g(\Sigma_B)} \models \mathrm{gnd}(t')$ and conclude that $D^{g(\Sigma_B)} \models \mathrm{gnd}(t)$. □

6.4 The Proof Procedure for $\mathrm{Init}(D)$

Definition 6.19
Let $C = \Gamma \to \Delta$ be a clause and $X = \{x_1, \ldots, x_n\}$ the set of variables in C. The *transformed clause* for C is

$$C^g = \mathrm{gnd}(x_1), \ldots, \mathrm{gnd}(x_n), \Gamma \to \Delta$$

◇

Lemma 6.20 *Let D be a specification and let C be a clause.*

1. *$D^g \models C^g$ if and only if $D \models C$.*
2. *$\mathrm{Gen}(D^g) \models C^g$ if and only if $\mathrm{Gen}(D) \models C$.*
3. *$\mathrm{Free}(D^g) \models C^g$ if and only if $\mathrm{Init}(D) \models C$.*

Proof: (1) Let A be an algebra in $\mathrm{Mod}(D)$ and let A' be the strongly persistent γ_D-extension of A where $\mathrm{gnd}_{s,A'}(a) = \top_{A'}$ for all elements a in s_A and sorts s in S. Since γ_D is consistent, any such A' is a model of D^g. Then we have $A' \models C^g$, $A' \models C$ and $A \models C$.

For the converse let A' be an algebra in $\mathrm{Mod}(D^g)$ and let A be the γ_D-reduct of A'. Since γ_D is a specification morphism, A is a model of D. We have $A \models C$, $A' \models C$ and $A' \models C^g$ whether $A' \models_\alpha \mathrm{gnd}_{s_i}(x_i)$ or not.

(2) The proof is analogous to the proof for (1). Note that if we require A to be term generated, A' is also term generated because of its construction. For the converse completeness of γ_D implies that the reduct of the term generated model A' is again term generated.

(3) C^g has the form $\mathrm{gnd}(x_1), \ldots, \mathrm{gnd}(x_n), \Gamma \to \Delta$. Let $C = \Gamma \to \Delta$ be a clause such that $\mathrm{Free}(D^g) \models C^g$, and let σ be a ground substitution such that $D \models \Gamma\sigma$. $D^g \models \mathrm{gnd}(x_i\sigma)$ by lemma 6.18 and $D^g \models \Gamma\sigma$ because γ_D is a specification morphism. From theorem 6.3 we get

$D^g \models\ \rightarrow \Delta\sigma$, and since γ_D is consistent $D \models\ \rightarrow \Delta\sigma$. Using theorem 6.3 again we conclude that C is valid in $\mathrm{Init}(D)$.

For the converse let $C = \Gamma \rightarrow \Delta$ be a clause such that $\mathrm{Init}(D) \models C$. Let $\sigma : X \rightarrow T_{\Sigma^g(X)}$ be a substitution such that $D^g \models \Gamma\sigma$. Suppose C contains a variable such that $x\sigma$ is not a ground term. Then $D^g \not\models \mathrm{gnd}(x\sigma)$ and $\mathrm{Free}(D^g) \models C^g\sigma$ is trivially true. Otherwise there exists a ground substitution τ such that $C\tau = C\sigma$. Because γ_D is consistent $D \models \Gamma\tau$, and by theorem 6.3 $D \models\ \rightarrow \Delta\tau$. Then $D^g \models\ \rightarrow \Delta\tau$. Using theorem 6.3 again we get the validity of C^g in $\mathrm{Free}(D)$. □

Lemma 6.20(3) enables us to prove the validity of a clause C in the initial model by first transforming specification and clause, and then using the proof procedure for $\mathrm{Free}(D)$ to prove validity of C^g. (1),(2) show that this method cannot directly be used to reduce validity in term generated models to validity in all models. We will later give a procedure for proving validity in term generated models by restricting the selection function.

Definition 6.21 (Proof procedure for $\mathrm{Init}(D)$)
Let D be a specification, let D' be a finite specification that is equivalent to D^g, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause not in D' and any non-rewrite clause in D'. The *proof procedure for* $\mathrm{Init}(D)$ takes a set of clauses H as input and generates a theorem proving derivation $D' \cup H^g = N_0, N_1, N_2, \ldots,$ using the completion procedure for $\mathcal{S}_S^{\Sigma(K)}$. If the completion procedure fails the inductive proof procedure answers "Disproof"; if the completion procedure terminates the inductive proof procedure answers "Proof". Additionally, the completion procedure has to obey the following restrictions.

1. All inferences are known to be strictly decreasing.
2. All unit clauses which are redundant in D' are known to be redundant.
3. Clauses are deleted if they are known to be redundant.
4. If some N_i contains a unit clause that is not redundant in D' the completion procedure fails.
5. If some N_i contains the empty clause the completion procedure fails. ◇

Theorem 6.22 *Let D be a specification, let D' be a finite specification that is equivalent to D^g, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause or nonreductive unit clauses. Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause not in D' and any non-rewrite clause in D'. Let H be a set of horn clauses. Suppose the proof procedure for* $\mathrm{Init}(D)$ *generates a theorem proving derivation N_0, N_1, N_2, $\ldots$ when H is given as input.*

1. *The clauses in H are valid in the initial model of D if and only if the proof procedure answers "Proof" or does not terminate.*
2. *Any inference from non-unit premises in N_i by strict superposition or selective paramodulation which is not D'-linear is redundant in D'.*

Proof: The proof is an obvious combination of theorem 6.12 and lemma 6.20. □

Example 6.23 (Associativity of addition)
We will now use the proof procedure for Init(**natplus**) to prove that addition of natural numbers is associative (cf. example 6.16). As the reduction ordering we use the lpo defined by $+ \succ s \succ 0$. The specification D' contains the clauses $N = \{(1), \ldots, (5)\}$, and the transformed clause for associativity is

$$H = \{(6) \quad \underline{\mathrm{gnd}(\mathrm{m})}, \mathrm{gnd}(\mathrm{n}), \mathrm{gnd}(\mathrm{n}') \to (\mathrm{m} + \mathrm{n}) + \mathrm{n}' \approx \mathrm{m} + (\mathrm{n} + \mathrm{n}')\}$$

We show that all inferences with premises in $N \cup H$ are redundant.

$\mathrm{gnd}(\mathrm{n}), \mathrm{gnd}(\mathrm{n}') \to (0 + \mathrm{n}) + \mathrm{n}' \approx 0 + (\mathrm{n} + \mathrm{n}')$	selective resolution (3) on (6), redundant because of (1)
$\mathrm{gnd}(\mathrm{m}), \mathrm{gnd}(\mathrm{n}), \mathrm{gnd}(\mathrm{n}') \to (\mathrm{s}(\mathrm{m}) + \mathrm{n}) + \mathrm{n}' \approx \mathrm{s}(\mathrm{m}) + (\mathrm{n} + \mathrm{n}')$	selective resolution (4) on (6), redundant because of (2)

The conclusion of both clauses can be rewritten to a tautology by using instances of the indicated clause. Therefore, $N \cup H$ is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$, the proof procedure for Init(D) answers "Proof" and $+$ is associative. $\diamond$

The proofs we exhibit show a close resemblance to classical induction proofs. The selection of an *induction variable* x is replaced by selecting an equation $\mathrm{gnd}(x)$ in the antecedent. Then selective paramodulation of the clauses added for the operators into the induction hypothesis yields one instance of the clause for each operator. These instances correspond to the reasoning by cases found in classical induction. They can be eliminated if they are redundant, which proves the particular case. When proving redundancy, smaller instances may be used; in classical induction, this corresponds to the assumption that smaller instances are already true.

6.5 Sufficient Completeness

Sufficient completeness is an important property of a specification. For instance, if we know that a specification is sufficiently complete with respect to a base signature, we need only consider terms over the base signature for inductive proofs. Our proof procedure will do this automagically if the inductive proof is preceded by a proof of sufficient completeness. Another important use of sufficient completeness is as a correctness criterium for parameterized specifications.

Definition 6.24 (Sufficiently complete)
Let D be a specification and let Σ_B be a base signature for D. D is called *sufficiently complete* with respect to Σ_B if for all ground terms t in T_Σ there exists a ground term t' in T_{Σ_B} such that $D \models t \approx t'$. $\diamond$

Theorem 6.25 *Let D be a specification, let Σ_B be a base signature for D and let S be a selection function. Suppose there exists a specification D' that is equivalent to D^g, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that contains a subspecification D_B that is equivalent to $D^{g(\Sigma_B)}$. Moreover, assume that S contains a pair $\langle C, \Gamma \rangle$ for all clauses in $D' \setminus D_B$. Then D is sufficiently complete with respect to Σ_B.*

Proof: This is trivially true if D is inconsistent. Otherwise, the clauses in D' are valid in the free model of D_B (cf. theorem 6.9). Since D' is equivalent to D^g and D_B is equivalent to $D^{g(\Sigma_B)}$, all clauses ω^g are valid in Free$(D^{g(\Sigma_B)})$. Since the clause ω^g for an operator is equal to the transformed clause C^g for gnd$(\omega(x_1,\ldots,x_n))$, we have $D^{g(\Sigma_B)} \models \text{gnd}(t)$ for any ground term t in T_Σ. Thus, for any ground term t there exists an equivalent term in T_{Σ_B}; we conclude that D is sufficiently complete with respect to Σ_B. □

Example 6.26 (Natural numbers, cont.)
Example 6.16 is a perfect example for the use of this theorem. There we state that $\mathbf{natplus}^{g(\Sigma_B)}$, where Σ_B contains the operators $\{0, \mathrm{s}\}$, is complete. We now give the promised proof. It remains to show that all inferences between clause (5) and other clauses are redundant.

	$\text{gnd}(\mathrm{n}) \to \text{gnd}(0 + \mathrm{n})$	selective resolution (3) on (5), redundant because of (1)
$(*)$	$\text{gnd}(\mathrm{m}), \text{gnd}(\mathrm{n}) \to \text{gnd}(\mathrm{s}(\mathrm{m}) + \mathrm{n})$	selective resolution (4) on (5), redundant because of (2),(4) and (5)

Note that the hypothesis (5) is used in the redundancy proof of $(*)$. This is allowed, since the instance that is used,

$$\text{gnd}(m), \text{gnd}(n) \to \text{gnd}(m + n),$$

is smaller than the peak of the inference,

$$\text{gnd}(s(m)), \text{gnd}(n) \to \text{gnd}(s(m) + n).$$

Thus, **natplus** is sufficiently complete with respect to $\{0, \mathrm{s}\}$. ◇

Definition 6.27 (Proof procedure for sufficient completeness)
Let D be a specification, let D_B be a finite specification that is equivalent to $D^{g(\Sigma_B)}$, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause not in D_B and any non-rewrite clause in D_B. The *proof procedure for sufficient completeness* (with respect to Σ_B) generates a theorem proving derivation $D_B \cup (D^g \setminus D^{g(\Sigma_B)}) = N_0, N_1, N_2, \ldots$, using the completion procedure for $\mathcal{S}_S^{\Sigma(K)}$. If the completion procedure fails the inductive proof procedure answers "Disproof"; if the completion procedure terminates the inductive proof procedure answers "Proof". Additionally, the completion procedure has to obey the following restrictions.

1. All inferences are known to be strictly decreasing.
2. All unit clauses which are redundant in D_B are known to be redundant.
3. Clauses are deleted if they are known to be redundant.
4. If some N_i contains a unit clause that is not redundant in D_B the completion procedure fails.
5. If some N_i contains the empty clause the completion procedure fails. ◇

Theorem 6.28 *Let D be a specification, let D_B be a finite specification that is equivalent to $D^{g(\Sigma_B)}$, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause not in D_B and any non-rewrite clause in D_B. Suppose the proof procedure for sufficient completeness with respect to Σ_B generates a theorem proving derivation $N_0, N_1, N_2, \ldots$*

1. *D is sufficiently complete with respect to Σ_B if and only if the proof procedure answers "Proof" or does not terminate.*
2. *If the proof procedure answers "Disproof", N_∞ contains a unit clause of the form $\mathrm{gnd}(t)$, where t is a term in $T_{\Sigma(X)}$ which is not equivalent to a ground term in T_{Σ_B}.*
3. *Any inference from non-unit premises in N_i by strict superposition or selective paramodulation which is not D_B-linear is redundant in D_B.*

Proof: Suppose the proof procedure answers "Proof" or does not terminate. Then the completion procedure generates a fair derivation and N_∞ is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$. For any clause in $N_\infty \setminus D_B$ the selection function S contains a pair $\langle C, \Gamma\rangle$. By theorem 6.25 the specification D is sufficiently complete with respect to Σ_B.

If the proof procedure answers "Disproof", N_∞ contains a clause C that is either a non-redundant unit clause or the empty clause. In both cases we have $D^{g(\Sigma_B)} \not\models C$ (cf. lemma 5.16), but since N_∞ is equivalent to D^g, we have $D^g \models C$. Thus C is contained in $\mathrm{UTh}(D^g)$ but not in $\mathrm{UTh}(D^{g(\Sigma_B)})$. Since C is not in $\mathrm{UTh}(D)$ and γ_D is consistent, C must be of the form $\mathrm{gnd}(t)$. Then t is equivalent to some ground term t' in T_Σ, but no ground term t'' in T_{Σ_B} exists that is equivalent to t or t'. Therefore, D is not sufficiently complete with respect to Σ_B.

The linearity of the proof can be established in the same way as for the proof procedure for $\mathrm{Free}(D)$. $\square$

6.6 A Literal Selection Strategy for Term Generated Models

Theorem 6.29 *Let D be a specification, let D' be a finite specification that is equivalent to D^g, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma\rangle$ for any non-unit clause not in D' and for any non-rewrite clause in D'. Furthermore, assume that for clauses C which are not in D' all selected equations in Γ' have the form $\mathrm{gnd}(t)$. Let H and H' be sets of horn clauses such that $D' \cup H^g$ is equivalent to $D' \cup H'$ and $D' \cup H'$ is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$. Then the clauses in H are valid in all term generated models of D.*

Proof: Observe that the construction of the interpretation $I_{D'(K)}$ involves skolemization; we must distinguish between true ground substitutions $X \to T_\Sigma$ and ground substitutions involving skolem constants $X \to T_{\Sigma(K)}$.

We prove that the clauses in H' are valid in all term generated models of D', by showing that $D' \models H'\sigma$ for all ground substitutions $\sigma : X \to T_\Sigma$ (cf. lemma 6.20). Let $C{\cdot}\sigma$ be the smallest ground instance of a clause C in H' not containing skolem variables such that $D' \not\models C\sigma$. Because of the restrictions put on the selection function S, the clause C has the form $\mathrm{gnd}(t), \Gamma \to \Delta$, where $\mathrm{gnd}(t\sigma)$ is the maximal selected literal of $C\sigma$. There exists an inference by selective paramodulation

$$\frac{\mathrm{gnd}(x_1), \ldots, \mathrm{gnd}(x_n) \to \mathrm{gnd}(\omega(x_1, \ldots, x_n)) \qquad \mathrm{gnd}(t), \Gamma \to \Delta}{([C =]\, \mathrm{gnd}(x_1), \ldots, \mathrm{gnd}(x_n), \Gamma \to \Delta)\mu}$$

in $\mathcal{S}_S^{\Sigma(K)}$, where μ is a most general unifier of t and $\omega(x_1, \ldots, x_n)$. There exists a ground substitution $\tau : X \to T_\Sigma$ such that $t\sigma = \omega(x_1, \ldots, x_n)\tau$ and τ satisfies the conditions of the inference rule. For the ground instance $C'\mu\tau = \mathrm{gnd}(x_1\tau), \ldots, \mathrm{gnd}(x_n\tau), \Gamma\sigma \to \Delta\sigma$ of the conclusion we have $D' \not\models C'\mu\tau$, since $D' \models \mathrm{gnd}(x_i\tau)$ for $1 \leq i \leq n$.[3] Since $D' \cup H'$ is complete, this inference is redundant and we get a contradiction as before (cf. proof of theorem 4.10).

Since $D' \cup H'$ is equivalent to $D' \cup H^g$ and in turn to $D^g \cup H^g$, we have $\mathrm{GTh}(D^g \cup H^g) = \mathrm{GTh}(D' \cup H') = \mathrm{GTh}(D') = \mathrm{GTh}(D^g)$. Using theorem 6.8 again we get $\mathrm{Gen}(D^g) \models H^g$ and using lemma 6.20 we conclude that $\mathrm{Gen}(D) \models H$. □

Definition 6.30 (Proof procedure for $\mathrm{Gen}(D)$**)**
Let D be a specification, let D' be a finite specification that is equivalent to D^g, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause not in D' and any non-rewrite clause in D'. The *proof procedure for* $\mathrm{Gen}(D)$ takes a set of clauses H as input and generates a theorem proving derivation $D' \cup H^g = N_0, N_1, N_2, \ldots$, using the completion procedure for $\mathcal{S}_S^{\Sigma(K)}$. The proof procedure answers "Proof" if the completion procedure terminates. Additionally, the completion procedure has to obey the following restrictions.

1. All inferences are known to be strictly decreasing.
2. The completion procedure fails if some N_i contains a clause that is not redundant in D' and that does not contain an equation of form $\mathrm{gnd}(t)$ in its antecedent.
3. If some N_i contains the empty clause the completion procedure fails. ◇

Theorem 6.31 *Let D be a specification, let D' be a finite specification that is equivalent to D^g, that is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$ and that does not contain the empty clause. Let S be a selection function containing a pair $\langle C, \Gamma \rangle$ for any non-unit clause not in D' and any non-rewrite clause in D'. Suppose the proof procedure for* $\mathrm{Gen}(D)$ *generates a theorem proving derivation N_0, N_1, N_2, ... when H is given as input.*

1. *The clauses in H are valid in the initial model of D if the proof procedure answers "Proof" or does not terminate.*
2. *Any inference from non-unit premises in N_i by strict superposition or selective paramodulation which is not D'-linear is redundant in D'.*

Proof: (1) The completion procedure generates a fair derivation and N_∞ is complete with respect to $\mathcal{S}_S^{\Sigma(K)}$. Let $H' = N_\infty \setminus D'$ and apply theorem 6.29.

(2) The same argumentation as in the case of the proof procedure for $\mathrm{Free}(D)$ applies (cf. theorem 6.12). □

Example 6.32 (Injectivity of successor, cont.)
We now try to prove that the successor function is injective in all term generated models of **natbase**. Of course, we will fail since we have already shown in example 6.2 that this is not true. The transformed specification is

natbase^g = **natbase** +
  ops
    gnd : nat

[3] If we would superpose a clause $\Lambda \to l \approx r$ on some other condition, we would not know whether $I_{D'} \models \Lambda\tau$ for the new conditions.

axioms $\forall\,\mathrm{m} : \mathrm{nat}$

$\mathrm{gnd}(0)$ (1)

$\mathrm{gnd}(\mathrm{m}) \to \mathrm{gnd}(\mathrm{s}(\mathrm{m}))$ (2)

We get the derivation

(3)	$\underline{\mathrm{gnd}(\mathrm{m})}, \mathrm{gnd}(\mathrm{n}), \mathrm{s}(\mathrm{m}) \approx (\mathrm{n}) \to \mathrm{m} \approx \mathrm{n}$	transformed clause to be proved
(4)	$\underline{\mathrm{gnd}(\mathrm{n})}, \mathrm{s}(0) \approx \mathrm{s}(\mathrm{n}) \to 0 \approx \mathrm{n}$	selective resolution (1) on (3)
(5)	$\mathrm{gnd}(\mathrm{m}), \mathrm{gnd}(\mathrm{n}), \mathrm{s}(\mathrm{s}(\mathrm{m})) \approx \mathrm{s}(\mathrm{n}) \to \mathrm{s}(\mathrm{m}) \approx \mathrm{n}$	selective resolution (2) on (3)
	$\mathrm{s}(0) \approx \mathrm{s}(0) \to 0 \approx 0$	selective resolution (1) on (4), tautology
(6)	$\underline{\mathrm{gnd}(\mathrm{n})}, \mathrm{s}(0) \approx \mathrm{s}(\mathrm{s}(\mathrm{n})) \to 0 \approx \mathrm{s}(\mathrm{n})$	selective resolution (2) on (4)
(7)	$\mathrm{s}(0) \approx \mathrm{s}(\mathrm{s}(0)) \to 0 \approx \mathrm{s}(0)$	selective resolution (1) on (6), failure

Clause (7) does not contain any predicate of form $\mathrm{gnd}(t)$; therefore, we cannot select one for superposition and we will not be able to generate a fair derivation meeting the requirements of theorem 6.29. Note that clause (7) is exactly the instance we have used to show that injectivity is not valid in all term generated models; hence, it is not in $\mathrm{GTh}(D)$. $\diamond$

Unfortunately we cannot decide whether a non-unit clause C is redundant. Therefore, proof procedure for $\mathrm{Gen}(D)$ is not refutationally complete.

Example 6.33 (Monotony of + with respect to $\leq$)
First, we repeat the full specification for this example:

natplusle =
  nat
 ops
  $0 : \ \to \mathrm{nat}$
  $\mathrm{s} : \mathrm{nat} \to \mathrm{nat}$
  $+ : \mathrm{nat} \times \mathrm{nat} \to \mathrm{nat}$
  $\leq : \mathrm{nat} \times \mathrm{nat}$
 axioms $\forall\,\mathrm{m}, \mathrm{n} : \mathrm{nat}$
  $0 + \mathrm{n} \approx 0$ (1)
  $\mathrm{s}(\mathrm{m}) + \mathrm{n} \approx \mathrm{s}(\mathrm{m} + \mathrm{n})$ (2)
  $0 \leq \mathrm{n}$ (3)
  $\mathrm{m} \leq \mathrm{n} \to \mathrm{s}(\mathrm{m}) \leq \mathrm{s}(\mathrm{n})$ (4)

No inferences with premises in **natplusle** exist; the specification is complete. Moreover, all clauses are reductive. The specification transformation adds the following operators and axioms:

natplusle^g = **natplusle** +
 ops

$\text{gnd} : \text{nat}$
$\text{gnd} : \text{pred}_{\leq}$
axioms $\forall\, \text{m}, \text{n} : \text{nat}$

$$\text{gnd}(0) \tag{5}$$
$$\text{gnd}(\text{m}) \rightarrow \text{gnd}(\text{s}(\text{m})) \tag{6}$$
$$\underline{\text{gnd}(\text{m})}, \text{gnd}(\text{n}) \rightarrow \text{gnd}(\text{m} + \text{n}) \tag{7}$$
$$\text{gnd}(\top_{\leq}) \tag{8}$$
$$\text{gnd}(\text{m}), \text{gnd}(\text{n}) \rightarrow \text{gnd}(\text{m} \leq \text{n}) \tag{9}$$

We have already shown that all inferences between the clauses (1), ...,(7) are redundant. For clauses (8),(9) not all inferences are redundant, but the completion procedure generates the following terminating derivation:

	$\text{gnd}(0), \text{gnd}(\text{n}) \rightarrow \text{gnd}(0 \leq \text{n})$	strict right superposition (3) on (9), redundant because of (3) and (8)
(10)	$\underline{\text{gnd}(\text{s}(\text{m}))}, \text{gnd}(\text{s}(\text{n})) \rightarrow \text{gnd}(\text{s}(\text{m}) \leq \text{s}(\text{n}))$	strict right superposition (4) on (9)
(11)	$\text{gnd}(\text{m}), \underline{\text{gnd}(\text{s}(\text{n}))} \rightarrow \text{gnd}(\text{s}(\text{m}) \leq \text{s}(\text{n}))$	selective resolution (6) on (10)
	$\text{gnd}(\text{m}), \text{gnd}(\text{n}) \rightarrow \text{gnd}(\text{s}(\text{m}) \leq \text{s}(\text{n}))$	selective resolution (6) on (11), redundant because of (4) and (9)

Now we are ready to prove that $+$ is monotonic with respect to $\leq$ in all term generated models of **natplusle**. We prove it for the second argument of $+$; in order to prove it for the first, we would need additional lemmas, but then it is possible in the same way. We have to prove the clause

$$m \leq n \rightarrow k + m \leq k + n$$

We continue the derivation from N_2:

(12)	$\underline{\text{gnd}(\text{k})}, \text{gnd}(\text{m}), \text{gnd}(\text{n}) \rightarrow \text{k} + \text{m} \leq \text{k} + \text{n}$	transformed clause to be proved
	$\text{gnd}(\text{m}), \text{gnd}(\text{n}) \rightarrow 0 + \text{m} \leq 0 + \text{n}$	selective resolution (3) on (12), redundant because of (1)
	$\text{gnd}(\text{k}), \text{gnd}(\text{m}), \text{gnd}(\text{n}) \rightarrow \text{s}(\text{k}) + \text{m} \leq \text{s}(\text{k}) + \text{n}$	selective resolution (4) on (12), redundant because of (2),(4) and (12)

$\diamond$

6.7 The Use of Lemmas

We now introduce a technique for obtaining redundancy proofs called contextual rewriting [BG90a, BG90b].

Definition 6.34 (Contextual rewriting)
Let C be a clause and let N be a set of clauses. $N_{C\cdot\sigma}$ denotes the set of all instances $C'\cdot\sigma'$ of clauses C' in N such that $C\cdot\sigma \succ^i C'\cdot\sigma'$.

Let $C = \Gamma, u[l\mu] \approx v \rightarrow \Delta$ (or $C = \Gamma \rightarrow u[l\mu] \approx v$) be a clause in N. Suppose there exists an instance $C'\cdot\mu$ of a clause $C' = \Lambda \rightarrow l \approx r$ in N such that

1. $l\mu \succ r\mu$;
2. $C'\cdot\mu$ is in $N_{C\cdot\iota}$;
3. $(N_{C\cdot\iota})\kappa \models \Gamma\kappa \to u'\kappa \approx v'\kappa$ for any equation $u' \approx v'$ in $\Lambda\mu$.

Then C can be contextually rewritten to $\Gamma, u[r\mu] \approx v \to \Delta$ (or $\Gamma \to u[r\mu] \approx v$). $\diamond$

The clause C becomes redundant in $N \cup \{\Gamma, u[r] \approx v \to \Delta\}$ (cf. lemma 5.6). Therefore, we have the simplification step

Contextual Rewriting: $$\frac{N \cup \{\Gamma, u[l] \approx v \to \Delta\}}{N \cup \{\Gamma, u[r] \approx v \to \Delta\}}$$

We may also do several steps of contextual rewriting in a row; in this case the bound on the complexity is provided by the first clause. If we eventually arrive at a clause that is redundant, we have proved that the first clause is redundant. Or we may use the method to prove inferences redundant; in this case the bound is provided by the maximal premise of the inference.

Contextual rewriting provides a nice way to integrate lemmas into the proof procedure.

Example 6.35 (Commutativity of addition)
Suppose we have proved that

$$\mathrm{m} + 0 \approx \mathrm{m}$$
$$\mathrm{m} + \mathrm{s}(\mathrm{n}) \approx \mathrm{s}(\mathrm{m} + \mathrm{n})$$

are inductive theorems of **natplus**. If we have used the proof procedure for Init(**natplus**), we will have a complete specification D'

natplus$'$ = **natplus**^{g} +
axioms $\forall\, \mathrm{m}, \mathrm{n} : \mathrm{nat}$

$$\mathrm{gnd}(\mathrm{m}) \to \mathrm{m} + 0 \approx \mathrm{m} \qquad (6)$$
$$\mathrm{gnd}(\mathrm{m}), \mathrm{gnd}(\mathrm{n}) \to \mathrm{m} + \mathrm{s}(\mathrm{n}) \approx \mathrm{s}(\mathrm{m} + \mathrm{n}) \qquad (7)$$

which is consistent to **natplus**^{g}. It has the same initial model; therefore, we can as well take this specification as the starting point for the next proof, and use the clauses (6) and (7) as lemmas. We will prove the commutativity of addition.

(8)	$\underline{\mathrm{gnd}(\mathrm{m})}, \mathrm{gnd}(\mathrm{n}) \to \mathrm{m} + \mathrm{n} \approx \mathrm{n} + \mathrm{m}$	transformed clause to be proved
	$\mathrm{gnd}(\mathrm{n}) \to 0 + \mathrm{n} \approx \mathrm{n} + 0$	selective resolution (3) on (8), redundant because of (1) and (6)
$(*)$	$\mathrm{gnd}(\mathrm{m}), \mathrm{gnd}(\mathrm{n}) \to \mathrm{s}(\mathrm{m}) + \mathrm{n} \approx \mathrm{n} + \mathrm{s}(\mathrm{m})$	selective resolution (4) on (8), redundant because of (2),(7) and (8)

We will investigate the redundancy proof for $(*)$ in more detail.

$(\dagger)$	$\mathrm{gnd}(\mathrm{m}), \mathrm{gnd}(\mathrm{n}) \to \mathrm{s}(\mathrm{m}) + \mathrm{n} \approx \mathrm{s}(\mathrm{n} + \mathrm{m})$	used (7) to contextually reduce right side of conclusion
	$\mathrm{gnd}(\mathrm{m}), \mathrm{gnd}(\mathrm{n}) \to \mathrm{s}(\mathrm{m} + \mathrm{n}) \approx \mathrm{s}(\mathrm{n} + \mathrm{m})$	used (2) to reduce left side of conclusion, redundant because of (8)

The step by contextual reduction (†) involves the instance

$$\mathrm{gnd}(s(n)), \mathrm{gnd}(m) \to n + s(m) \approx s(n+m)$$

of (7). The reduction is possible, since

1. $\mathrm{n} + \mathrm{s}(\mathrm{m}) \succ \mathrm{s}(\mathrm{n} + \mathrm{m})$;
2. the instance of (7) is smaller than $(*)$;
3. $(4)\kappa \models \mathrm{gnd}(m\kappa), \mathrm{gnd}(n\kappa) \to \mathrm{gnd}(s(n\kappa))$ and (4) is smaller than (7);
4. $\emptyset \models \mathrm{gnd}(m\kappa), \mathrm{gnd}(n\kappa) \to \mathrm{gnd}(m\kappa)$. $\diamond$

7 Conclusion

Implementation. We have implemented the proof procedures for initial models and for sufficient completeness in the CEC-system. The selection of equations is done manually, whereas redundancy proofs are generated automatically. Up to now, we have proved the following theorems:

Natural numbers
: Associativity and commutativity of $+$ and $*$, distributivity of $*$ over $+$, totality, reflexivity (trivial), antisymmetry, transitivity of $\leq$ and monotony of $+$ with respect to $\leq$.

Iterative versions
: Iterative versions of factorial and reverse have been proved correct with respect to their recursive definitions.

Lists
: Associativity of append, $\mathrm{rev}(\mathrm{rev}(l)) \approx l$.

Correctness of a compiler for expressions
: This example has been adapted from an example for the TIP-system [Fra89]. It compiles an expression into a sequence of instructions for a stack machine. The expression contains only constants, variables and binary operations.

Sufficient completeness has been proved for all specifications with respect to those operators that actually appear in ground normal forms.

Further Work. The method should be generalized to the case of first-order clauses. In this case, an initial model need not exist, even for consistent specifications. Instead, we have to replace it by a *perfect model* [BG90c], which depends on the reduction ordering. For the inference systems and completeness proofs in Section 4 the generalization should be easy, since they originally were developed for this case.

The four different classes of models and the related theories should be unified under a common notion of semantics. It is probably possible to use the same approach for final semantics and the theory containing ground goal clauses. For the theory corresponding to the perfect model the ordering must be taken into account, but it is at the moment not clear how this should be done.

Acknowledgements. I am grateful to Prof. Harald Ganzinger for supervising my work, to Hubert Bertling for long discussions on completion and proof by consistency, to Ullrich Hustadt for pointing out various errors and weak spots in this thesis, and to all other members of Lehrstuhl Informatik V for their support.

References

[Bac88] L. Bachmair. Proof by Consistency in Equational Theories. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 228–233, Edinburgh, July 1988.

[BDH86] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for Equational Proofs. In [LICS'86], pages 346–357.

[Ber90] H. Bertling. Knuth-Bendix Completion of Horn Clause Programs for Restricted Linear Resolution and Paramodulation. In [CTRS'90]. To appear as Springer LNCS.

[BG90a] L. Bachmair and H. Ganzinger. On Restrictions of Ordered Paramodulation with Simplification. In *Proc. 10th Int. Conf. on Automated Deduction*, Kaiserslautern, 1990. To appear as Springer LNCS.

[BG90b] L. Bachmair and H. Ganzinger. Completion of First-Order Clauses with Equality by Strict Superposition. In [CTRS'90]. To appear as Springer LNCS.

[BG90c] L. Bachmair and H. Ganzinger. Perfect Model Semantics for Logic Programs with Equality. Draft, November 1990.

[BL90] E. Bevers and J. Lewi. Proof by Consistency in Conditional Equational Theories. In [CTRS'90]. To appear as Springer LNCS.

[CTR'87] *Proc. 1st Int. Workshop on Conditional Term Rewriting*, Orsay, 1987.

[CTRS'90] *Proc. 2nd Int. Workshop on Conditional and Typed Rewriting Systems*, Montreal, 1990. To appear as Springer LNCS.

[DJ89] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. Technical Report 478, Unité Associée au CNRS UA410, April 1989. Draft of Chapter 15 of Volume B of *Handbook of Theoretical Computer Science*, North Holland, 1989.

[EGL89] H.-D. Ehrich, M. Gogolla, and U. Lipeck. *Algebraische Spezifikation abstrakter Datentypen.* B. G. Teubner, Stuttgart, 1989.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1.* Springer, 1985.

[Fra89] U. Fraus. Eine interactive Umgebung für Induktionsbeweise über algebraischen Strukturen. MIP-intern 04/89, Universität Passau, 1989.

[Fri89] L. Fribourg. A Strong Restriction of the Inductive Completion Procedure. *Journal of Symbolic Computation*, 8:253–276, 1989.

[Gan87] H. Ganzinger. A Completion Procedure for Conditional Equations. In [CTR'87], pages 62–83.

[Gog80] J. A. Goguen. How to Prove Algebraic Inductive Hypotheses without Induction. In *Proc. 5th Conf. on Automated Deduction*, pages 356–373, Les Arcs, July 1980. Springer LNCS 87.

[Gra89] B. Gramlich. Inductive Theorem Proving Using Refined Unfailing Completion Techniques. SEKI-Report SR-89-14, Universität Kaiserslautern, Fachbereich Informatik, 1989.

[Gra90] B. Gramlich. Completion Based Inductive Theorem Proving — a Case Study in Verifying Sorting Algorithms. SEKI-Report SR-90-04, Universität Kaiserslautern, Fachbereich Informatik, 1990.

[HH82] G. Huet and J.-M. Hullot. Proofs by Induction in Equational Theories with Constructors. *Journal of Computer and System Sciences*, 25:239–266, 1982.

[HK88] D. Hofbauer and R.-D. Kutsche. Proving Inductive Theorems Based on Term Rewriting Systems. In *Proc. 1st Workshop on Algebraic and Logic Programming*, pages 180–190, Gaussig, GDR, 1988. Springer LNCS 343.

[Höl89] S. Hölldobler. *Foundations of Equational Logic Programming.* Springer, 1989.

[HS73] H. Herrlich and G. E. Strecker. *Category Theory.* Allyn and Bacon, Boston, 1973.

[JK86] J.-P. Jouannaud and E. Kounalis. Proofs by Induction in Equational Theories Without Constructors. In [LICS'86], pages 358–366.

[KB70] D. E. Knuth and P. B. Bendix. Simple Word Problems in Universal Algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, U. K., 1970.

[KC86] S. Kaplan and M. Choquer. On the Decidability of Quasi-Reducibility. *EATCS Bulletin*, 28:32–34, 1986.

[KM86] D. Kapur and D. R. Musser. Inductive Reasoning with Incomplete Specifications. In [LICS'86], pages 367–377.

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

[KNZ86] D. Kapur, P. Narendran, and H. Zhang. Proof by induction using test sets. In *Proc. 8th Int. Conf. on Automated Deduction*, Oxford, July 1986. Springer LNCS 230.

[Küc87] W. Küchlin. Inductive Completion by Ground Proof Transformation. Technical Report 87-08, Department of Computer and Information Sciences, University of Delaware, February 1987.

[LICS'86] *Proc. Symp. on Logic in Computer Science*, Cambridge, Mass., June 1986.

[Mac72] S. MacLane. *Categories for the Working Mathematician.* Springer, 1972.

[Mus80] D. R. Musser. On proving inductive properties of abstract data types. In *Proc. 7th Annual ACM Symp. on Principles of Programming Languages*, pages 154–162, Las Vegas, 1980.

[Nie91] R. Nieuwenhuis. First-Order Completion Techniques. Draft, 1991.

[Ore87] F. Orejas. Theorem Proving in Conditional-Equational Theories. In [CTR'87].

[Pad88] P. Padawitz. *Computing in Horn Clause Theories*. Springer, 1988.

[Pla85] D. A. Plaisted. Semantic Confluence Tests and Completion Methods. *Information and Control*, 65:182–215, 1985.