Inductionless Induction
Lecture Notes

Hubert Comon*

October 30, 1995

*CNRS and LRI, Bat. 490, Université de Paris Sud, 91405 ORSAY cedex, France. E-mail comon@lri.lri.fr

Summary of the lecture

The course relies on the work which has been done for 15 years in rewriting theory on inductive proofs. The aim here is to give a general introduction encompassing all works which are relevant to "inductionless induction" [21] or "proof by consistency" [17]. For an excellent introduction on the techniques and motivations of these works, see L. Bachmair's book [2].

The original problem was to automatically prove (or disprove) a formula ϕ in the *initial models* of a set of equations E (or Horn clauses with equality). Basically, inductionless induction is a "proof by consistency" method (although we will see other developments).The idea is to use a (first-order) axiomatization $\mathcal{A}$ of the initial models, then add ϕ to $\mathcal{A} \cup E$ and try to derive an inconsistency. In case of success, ϕ is disproved. On the other hand, if $E \cup \{\phi\} \vdash E'$ where E' is *saturated* and $E' \cup \mathcal{A}$ is consistent, then ϕ is an inductive theorem. Three issues have to be discussed:

1. How can we build an axiomatization $\mathcal{A}$ which has the desired properties ?
2. How is it possible to get a saturated set E' from $E \cup \{\phi\}$?
3. How is it possible to decide the consistency of $E' \cup \mathcal{A}$?

The main contribution of rewrite theory was first to explicitly give some axiomatizations $\mathcal{A}$ for which the above method is indeed correct and for which the consistency of $E' \cup \mathcal{A}$ is decidable. Several such sets of axioms $\mathcal{A}$ have been successively (and successfully) proposed, each of them assuming some strong properties of E. For example, D. Musser [22] assumes that E contains an equational axiomatization of equality, and $\mathcal{A}$ is the set $\{true \neq false\}$. G. Huet and J.-M. Hullot [15] assume a *constructor theory* and $\mathcal{A}$ expresses that two pure constructor terms are distinct (see also [21]). J.-P. Jouannaud and E. Kounalis [16] and L. Bachmair [1] assume that E is given by a ground-convergent rewrite system (see also [17]), then $\mathcal{A}$ expresses that two equal terms should be *ground reducible*. This last notion plays a central role in further developments of inductionless induction, as will be explained later.

Now, the second issue above (computing a saturated set of formula) refers to some notion of *redundancy*: a set of formulas is saturated when every inference leads to some redundancy. A typical redundancy criterion is given by *orderings on proofs* [3, 1]: A formula ϕ is redundant in $\Phi \cup \{\phi\}$ when there is a proof $\Phi \vdash \phi$ which is smaller than the proof $\phi \vdash \phi$. Originally, this notion is used in term rewriting system: when Φ is a convergent rewrite system, then every equation that can be deduced from Φ is redundant, according to an adequate ordering on equational proofs [2]. In the case of proofs by consistency, the deduction system consists in some restrictions of the Knuth-Bendix completion procedure, as explained in [2, 13]. This corresponds actually to apply an (implicit) induction scheme.

Besides the proof methods, several works have been devoted to the decision of ground reducibility (e.g. [16, 23, 19, 18], [20, 5, 8]). For arbitrary rewrite systems, the problem has been shown decidable by D. Plaisted [23]. However, the complexity of the decision algorithm is so high (a tower of 7 exponentials) that several other methods have been proposed afterwards. In the general case, the problem, including the computation of a "counter-example" in case of non-ground reducibility, is at least doubly exponential. When we stick to the decision problem, it is at least exponential [18]. This is probably the exact complexity of the problem [8]. Finally, ground reducibility tests can also be used to compute directly a conservative extension of the original set E for which Huet and Hullot's method can be used [7].

All these techniques have been actually developed in the framework of equational theories. They carry over Horn clauses with equality, with the provision that, in general, there is no known axiomatization $\mathcal{A}$ for which the consistency of $E' \cup \mathcal{A}$ can be decided. Nevertheless, it is possible to design a technique which may fail to prove consistency, but always succeeds in detecting inconsistencies; we get refutationally complete proof techniques. (Of course, they may also prove sometimes the consistency of $E' \cup \mathcal{A}$). However, in the conditional case, it is not only hard to prove consistency, but also to get a saturated set. That is why other methods have been designed, which can be seen as intermediate between explicit induction and proofs by consistency. In these methods, the inductive scheme is replaced with so-called *test sets* which are computed from the specification. This is the basis of recent works on inductive proofs in Horn clauses theories [4].

There are several working systems based on all these techniques: RRL, ReDuX, SPIKE and many others.

Here is the summary of the course:

1. Introduction of the problem: statement and examples

2. Rewrite systems

3. General setting of the inductionless induction method.

1 Introduction of the problem: statement and examples

Let me start with a very very classical and simple example.

Example 1.1 Assume that E is the set of equations (implicitly universally quantified):

$$\begin{cases} 0 + x &= x \\ s(x) + y &= s(x + y) \end{cases}$$

Assume moreover that our set of function symbols only consists of $0, s, +$. Hence the above set of equations can be viewed as a definition of $+$ on the natural numbers in base 1.

For example, $0+(s(0)+0) = s(0)$ is a logical consequence of E. $x+0 = x$ is not, however, a logical consequence of E, as there are models of E which do not satisfy this equation.

Exercise 1.

Build a 2 elements model of E which does not satisfy $x+0 = x$. This can be done by explicitly giving the domain (say: $\{0, a\}$) and the functions s and $+$ by their values on these two elements. Equations of E must be satisfied whereas $a + 0 \neq a$.

However, if we only consider natural numbers (i.e. assuming that $x = s^n(0)$ for some n), then it is possible to prove $x + 0 = x$ by induction on n: $s^{n+1}(0) + 0 = s(s^n(0) + 0)$ by the second equation. This is well-known: the inductive theory contains more formulas than the equational theory. Our purpose here is to prove formulas which indeed hold in the inductive theory ($x+0 = x$ in our example). One way to prove such inductive consequences of E is to use (as we did) an explicit structural induction. For, we need to know e.g. that natural numbers is the smallest set containing 0 and closed under successor. In the method we are going to introduce, we will not assume such a knowledge. Let me give another example to illustrate this aspect.

Example 1.2 Assume that there are three function symbols: 0 (constant), s (unary) and $+$ (binary, used in infix notation). Assume moreover that E

consists of the equations:

$$\begin{cases} s(s(0)) & = & 0 \\ x+x & = & 0 \\ x+0 & = & x \\ 0+x & = & x \end{cases}$$

Trying to prove the inductive theorem $s(s(x)) = x$, we run into trouble. Let us try an explicit inductive proof. If $x = 0$, then the equation is in E. If $x = s(y)$, then we have, by induction hypothesis $s(s(y)) = y$, hence $s(s(s(y))) = s(y)$. Now, if $x = y+z$, we don't know how to go on because no equation applies to $s(s(y+z))$. Actually, we need the fact that all ground terms (i.e. terms without variables) containing $+$ are equal, modulo E, to a term which does not contain $+$. (This is the *sufficient completeness* property which we introduce later). Now, showing (automatically) such a property is not straightforward.

More generally, the problem of automating inductive proofs becomes harder when there are no known *constructors* or when there are relations between the constructors. Let us precise these notions.

1.1 Notations

We assume in the following that a finite set of *function symbols* F is given, each function symbol being associated with a fixed *arity* (the number of its arguments). Then $T(F)$ is the set of terms built on that alphabet F. Let also X be an (infinite) set of *variable symbols* which are viewed as function symbols of arity 0. $T(F \cup X)$ is also written $T(F, X)$. Identity on terms is written $\equiv$.

A *substitution* is a mapping from X into $T(F, X)$. A substitution is *ground* when all images belong to $T(F)$. When $x_1, ..., x_n$ are variables, $\{x_1 \mapsto t_1; ...; x_n \mapsto t_n\}$ is the substitution mapping each x_i to the corresponding t_i and all other variables to themselves (or to any ground term in case of ground substitutions). Substitutions can be extended in a unique way into an endomorphism of $T(F, X)$ (w.r.t. the structure of F-algebra): $f(t_1, \ldots, t_n)\sigma \stackrel{\text{def}}{=} f(t_1\sigma, \ldots, t_n\sigma)$. (We use here and in the following substitutions such as σ in postfix notation).

If E is a set of equations between terms of $T(F, X)$, $=_E$ is the smallest congruence on $T(F, X)$ such that $s\sigma =_E t\sigma$ for all equations $s = t \in E$ and for all substitutions σ.

A *position* is a string of natural numbers. The empty string is Λ and the concatenation of strings is written $\cdot$. As usual, a term can be viewed as a mapping from a set $Pos(t)$ of positions into F. If $p \in Pos(t)$, then $t|_p$ is the subterm of t at position p and $t[u]_p$ is the term obtained by replacing $t|_p$ with u in t. $\mathcal{V}ars(t)$ is the set of variables occurring in t and $\mathcal{F}Pos(t)$ is the set of non-variable positions of t.

1.2 Equational Deduction

An equation is an unordered pair of terms. ($s = t$ and $t = s$ are considered as being the same equation). Given a set of equations E, $s = t$ is a *consequence* of E if $s =_E t$. Equivalently $s = t$ is a consequence of E if it can be proved using a finite sequence of *replacement of equals by equals*: $u \underset{E}{\longleftrightarrow} v$ if there is an equation $l = r \in E$, a substitution σ and a position $p \in Pos(u)$ such that $u|_p \equiv l\sigma$ and $v \equiv u[r\sigma]_p$. If we want to make precise the equation and/or the position and/or the substitution, we can use them as indices of the relation as in: $u \overset{p,\sigma}{\underset{l=r}{\longleftrightarrow}} v$.

1.3 Inductive theory

Let $\mathcal{I}$ be the quotient algebra $T(F)/_{=_E}$ (or the least fixed point in case of conditional equations), i.e. the set of equivalence classes on ground terms.

Definition 1.3 *An* inductive consequence *of E is an equation $s = t$ such that $\mathcal{I} \models s = t$.*

Equivalently, $s = t$ is an inductive consequence of E iff for every ground substitution σ, $s\sigma =_E t\sigma$.

In example 1.1, $x+0 = x$ is not a consequence of E, but it is an inductive consequence as for all ground terms t, $t + 0 =_E t$.

1.4 Constructors and sufficient completeness

A subset C of F is called a set of *constructors* (w.r.t. E) if, for every term t in $T(F)$, there is a term u in $T(C)$ such that $t =_E u$

Exercise 2.
Prove that $\{0, s\}$ is a set of constructors in example 1.1.

Definition 1.4 *E is* sufficiently complete *with respect to a subset D of F if $C = F \setminus D$ is a set of constructors w.r.t. E.*

Example 1.5 Assume that we have, in addition to example 1.1, a new function symbol p and the three additional equations:

$$\left\{ \begin{array}{rcl} p(s(x)) & = & x \\ s(p(x)) & = & x \\ p(x) + y & = & p(x+y) \end{array} \right.$$

Then E is sufficiently complete w.r.t. $\{+\}$.

Exercise 3.
Show this property.

If we know a set of constructors, then we may use a structural induction over this set of constructors in order to prove inductive consequences of E. However, on one hand it is not always easy to determine a set of constructors and, on the other hand, the structural induction on this set of constructors may not be sufficient. Indeed, F itself is always a set of constructors. Even if we assume that we have been able to find a minimal set of constructors, there are still problems because there might be *relations between constructors*. In example 1.1, the set $C = \{0, s\}$ of constructors is *free* because, for any two terms $s, t \in T(C)$, $s \neq_E t$. However, in examples 1.2,1.5, the set of constructors is not free.

1.5 Our goal

Our aim here is to explain a technique for proving or disproving inductive consequences of E, without any assumption on the constructors. We insist also on *disproving* conjectures. Indeed, there is an obvious way for disproving a conjecture $s = t$: exhibit a counter example, i.e. a ground substitution σ such that $s\sigma \neq_E t\sigma$. (This assumes that the relation $=_E$ is decidable). However, it should be clear that it is not a realistic method to try every ground substitution. The "inductionless induction method" is a "proof by consistency technique" which, roughly, searches such counter-examples in an adequate way. Hence, we will be interested in *refutation completeness*, i.e. finding such counter-examples each time a conjecture is false. Also, of course, the method may succeed in proving the conjecture, but not always as the inductive theory is not recursive in general. (Actually, the inductive theory, which consists of all inductive consequences of E is co-recursively enumerable but not recursively enumerable).

2 Term Rewriting and Completion

This section is devoted to a necessary detour through term rewriting. We only recall very briefly the basic background. More informations and devel+opments can be found in e.g. [2, 11].

2.1 Rewrite systems

A *rewrite rule* is an oriented equation. A *rewrite system* is a set of rewrite rules. Rewriting a term t into a term u using a rule $l \to r$ at position $p \in Pos(t)$ with the substitution σ is defined as the replacement of equals by equals and written $t \xrightarrow[l\to r]{p,\sigma} u$.

Given a rewrite relation $\xrightarrow[\mathcal{R}]{}$ w.r.t a rewrite system $\mathcal{R}$, the corresponding *reduction relation* is the reflexive transitive closure $\xrightarrow[\mathcal{R}]{*}$ of $\xrightarrow[\mathcal{R}]{}$. $\mathcal{R}$ is *confluent* if, for every terms s, t, u such that $s \xrightarrow[\mathcal{R}]{*} t$ and $s \xrightarrow[\mathcal{R}]{*} u$, there is a term v such that $t \xrightarrow[\mathcal{R}]{*} v$ and $u \xrightarrow[\mathcal{R}]{*} v$. $\mathcal{R}$ is *terminating* iff there is no infinite sequence $t_1 \xrightarrow[\mathcal{R}]{} \ldots \xrightarrow[\mathcal{R}]{} t_n \xrightarrow[\mathcal{R}]{} \ldots$. $\mathcal{R}$ is *convergent* when it is both terminating and confluent. For convergent rewrite systems $\mathcal{R}$, the equational theory is decidable: $s =_{\mathcal{R}} t$ iff the (unique) normal form of s w.r.t. $\mathcal{R}$ is identical to the (unique) normal form of t.

2.2 Orderings

A *reduction ordering* is a well founded ordering $\geq$ on terms which is *monotonic*, i.e. such that, for any s, t, u and any position p of s, if $t > u$, then $s[t]_p > s[u]_p$. The reduction relation of a terminating rewrite system is a reduction ordering.

There are several (a lot of) reduction orderings that can be found in the literature (see [11]). The most popular one in rewriting is the *recursive path ordering*. Defining this ordering requires lexicographic and/or multiset extensions which we will use also for other purposes.

If $R_1, \ldots, R_n$ are relations on $A_1, \ldots, A_n$, let, for each i, $\approx_i$ be the relation defined by $x \approx_i y$ if xR_iy and yR_ix. The *lexicographic combination* of $R_1, \ldots, R_n$ (from left to right) is the relation R defined on $A_1 \times \ldots A_n$ by:

$$(a_1, \ldots, a_n)R(b_1, \ldots, b_n) \text{ iff } \exists 0 \leq i < n, a_1 \approx_1 b_1 \wedge \ldots \wedge a_i \approx_i b_i \wedge a_{i+1}R_{i+1}b_{i+1}$$

Multisets over a set A are applications from A into the natural numbers whose value is 0, but on a finite subset of A. Mulstisets can also be viewed as finite sets with repetitions and are written accrodingly. For example, a multiset M such that $M(a) = 2, M(b) = 1, M(c) = 3$ and $M(x) = 0$ for all other values can be displayed $M = \{a, a, b, c, c, c\}$ or $\{c, b, a, c, c, a\}$. Operations are defined as expected. For example, u nion of multisets is defined by $M \cup M'(a) = M(a) + M'(a)$.

Proposition 2.1 *Lexicographic combinations of well-founded (quasi)-orderings is a well-founded (quasi)-ordering.*

For all (quasi)-orderings $\geq$ whose associated equivalence relation $(\geq \cap \leq)$ is $\approx$, we write $>$ the strict ordering, i.e. the relation $\geq \cap \not\approx$.

Given a (quasi)-ordering $\geq$ on A, its *multiset extension* is the smallest relation $\ggeq$ on the multisets over A which satisfies:

- $M \ggeq \emptyset$ for all M.
- If $M \gg N$ and $M' \ggeq N'$ then $M \cup M' \gg N \cup N'$
- If $a > b_1, \ldots, a > b_n$, then $\{a\} \gg \{b_1, \ldots, b_n\}$
- If $a \geq b$, then $\{a\} \ggeq \{b\}$.

Theorem 2.2 *[12] If $\geq$ is a well-founded (quasi)-ordering, then $\ggeq$ is a well founded (quasi)-ordering.*

Given a (quasi)-ordering $\geq_F$ on the set F of function symbols, and given, for each function symbol, a *status* which is either lex or mult. Two symbols f, g such that $f \approx g$ are assumed here to have the same arity and the same status.

The *recursive path ordering* is defined on $T(F)$ (or $T(F, X)$) by:

$$s \equiv f(s_1, \ldots, s_n) \geq_{rpo} g(t_1, \ldots, t_m) \equiv t \text{ iff}$$

- either there is an index i such that $s_i \geq_{rpo} t$
- **Or** $f >_F g$ and, for all i, $s >_{rpo} t_i$
- **Or** $f \approx_F g$ and f has the status lex and
  - For all i, $s > t_i$

– **And** $(s_1, \ldots, s_n) \geq^{lex} (t_1, \ldots, t_n)$ where $\geq^{lex}$ is the lexicographic combination of n times $\geq_{rpo}$.

- **Or** $f \approx g$ and f has the multiset status and

$$\{s_1, \ldots, s_n\} \gg_{rpo} \{t_1, \ldots, t_n\}$$

$\geq_{rpo}$ is a quasi-ordering on terms (it is an ordering when $\geq_F$ is an ordering and all symbols have the lex status). $\gg_{rpo}$ is a reduction ordering as soon as $\geq_F$ is well-founded on F (which is always the case when F is finite). $\gg_{rpo}$ is stable by instanciation. It is total on $T(F)$ iff $\geq_F$ is total.

2.3 Standard Completion

Completion processes (which will be explained in more details in section 5) consist basically in orienting equations according to a given reduction ordering and computing equational consequences. They, roughly, aim at computing from a set of equations an equivalent convergent rewrite system.

More precisely, a completion procedure takes as input a set of equations E and and a reduction ordering $\geq$ and applies successively one of the rules of figure 1, leading to a sequence of pairs E_i, R_i.

Starting from $E_0; \emptyset$, the rules of figure 1 produce a sequence $E_i; R_i$ called a *completion sequence*. The limit sets E_∞ and R_∞ of such a sequence are defined by:

$$E_\infty = \bigcup_i \bigcap_{j \geq i} E_j \qquad\qquad R_\infty = \bigcup_i \bigcap_{j \geq i} R_j$$

A completion sequence is *fair* if all equations that can be deduced from R_∞ using the **Deduce** rule belong to $\bigcup_i E_i$. (In other words: all overlap between persistent rules are eventually considered).

Exercise 4.

Show that the following completion procedure is fair (we specify here how to derive $E_i; R_i$ from $E; R$. All sequences obtained in the way specified below are fair): repeat

1. Select an equation $s = t$ in E and apply the **Orient** rule, leading to $s \rightarrow t$.

2. Apply the **Compose** rule as long as possible.

Orient

$$E \cup \{l = r\}; R \;\Rightarrow\; E; R \cup \{l \to r\}$$

If $l > r$.

Deduce

$$E; R \cup \{l \to r;\; g \to d\} \;\Rightarrow\; E \cup \{l[d]_p\sigma = r\sigma\}; R \cup \{l \to r;\; g \to d\}$$

If p is a non-variable position of l and σ is a most general unifier of $l|_p$ and g.

Compose

$$E; R \cup \{l \to r\} \;\Rightarrow\; E; R \cup \{l \to r'\}$$

If $r \xrightarrow[R]{+} r'$.

Simplify

$$E \cup \{s = t\}; R \;\Rightarrow\; E \cup \{s = t'\}; R$$

If $t \xrightarrow[R]{+} t'$.

Delete

$$E \cup \{s = s\}; R \;\Rightarrow\; E; R$$

Collapse

$$E; R \cup \{l \to r\} \;\Rightarrow\; E \cup \{l' = r\}; R$$

If $l \xrightarrow[g \to d]{p,\sigma} l'$ with $g \to d \in R$ and either $p \neq \Lambda$, or g is different from l up to renaming, or else $r > d$.

Figure 1: Completion Rules

3. Apply the **Deduce** rule to $s \to t$ (or its simplified form). i.e. overlap $s \to t$ with all other rules in R. This computation excludes the overlap of a rule on itself at the root position.

4. Apply **Simplify** as long as possible

5. Apply **Collapse** as long as possible

6. Apply **Delete** as long as possible

The completion *terminates* when $E_i = E_{i+1}$ and $R_i = R_{i+1}$ for some i. It *fails* if $E_\infty \neq \emptyset$.

When the completion doesn't fail, then R_∞ is a convergent rewrite system (see [10, 2]). This result is actually a consequence of the *proof ordering technique*. Let us summarize very briefly this technique.

Given a pair (E, R), an *equational proof* $\mathcal{P}$ of $s = t$ is a sequence $s_0 = s, ..., s_n = t$ where each s_i is obtained from s_{i-1} using an equation of E or a rule of R. The *complexity* of such a proof $\mathcal{P}$ is the multiset of the complexities of all proof steps. Now, the complexity of a single proof step is defined by:

- $c(s \underset{u=v}{\longleftrightarrow} t) = (\{s,t\}, \bot, \bot, \bot)$ if $u = v \in E$.

- $c(s \xrightarrow[l \to r]{p,\sigma} t) = (\{s\}, s|_p, l, t)$ if $l \to r \in R$.

The ordering on such complexity measures is defined using multiset extensions and lexicographic combinations of three orderings: the reduction ordering $\geq$ which is used in the **Orient** rule of the completion, the subterm ordering, the *encompassment ordering* $\trianglelefteq$ which is defined by: $s \trianglelefteq t$ iff there is a substitution σ such that $s\sigma$ is a subterm of t.

More precisely, the ordering $\succeq^c$ on proofs complexities is the multiset extension of the lexicographic combination of the following orderings:

- (For the first component of the quadruple), the multiset extension of $\geq$

- (Second component of the quadruple) subterm ordering

- (Third component of the quadruple) encompassment ordering

- (Fourth component of the quadruple) $\geq$.

The quasi-ordering on proofs $\succeq^{\mathcal{P}}$ is defined according to the complexities: $\gamma \succ^{\mathcal{P}} \gamma'$ iff $c(\gamma) \succ^c c(\gamma')$ and $\gamma \approx \gamma'$ when $c(\gamma) = c(\gamma')$. $\succeq^{\mathcal{P}}$ is is well-founded since $\succeq^c$ is a combination using multiset and lexicographic extensions of well-founded orderings.

Exercise 5.
Show the following fundamental lemma on the simplification of equational proofs: If $E, R \vdash E', R'$, and $\mathcal{P}$ is a proof of $s = t$ in E, R, then either $\mathcal{P}$ is a proof of $s = t$ in E', R', or else there is a proof $\mathcal{P}'$ of $s = t$ in E', R' such that $\mathcal{P} \succ^{\mathcal{P}} \mathcal{P}'$.

According to the result of this exercise, the proofs are decreasing along the completion sequence. Now, the next fundamental result is the strict decreasingness of the proofs for non-failing (fair) completions. Let a *rewrite proof* be an equational proof of the form $s \to \ldots \to \leftarrow \ldots \leftarrow t$.

Exercise 6.
Show that if E_i, R_i is a fair completion sequence which does not fail and if, for some index i, there is a proof $\mathcal{P}$ of $s = t$ in $E_i; R_i$ which is not a rewrite proof, then there is some index $j \geq i$ and a proof $\mathcal{P}' \prec^{\mathcal{P}} \mathcal{P}$ in $E_j; R_j$.

Hence, for all non-failing fair completion sequences, each provable equation is eventually provable using a rewrite proof (because $\succ^{\mathcal{P}}$ is well-founded).

Another consequence of this result is a simple *redundancy criterion*: we call an equation (or a rule) $s = t$ *redundant* in $E; R$ if there is a proof in $E; R$ of $s = t$ which is smaller than the proof consisting of the single step $s \underset{s=t}{\longleftrightarrow} t$. **Deduce** in the completion process can be restricted to non-redundant equations (or rules), which is fundamental w.r.t. termination of the completion.

The definition of fairness has to be modified according to the redundancy criterion.

Exercise 7.
A very classical example: complete the following set of equations (a presentation of group theory) into a convergent rewrite system:

$$\left\{ \begin{array}{rcl} e * x & = & x \\ I(x) * x & = & e \\ (x * y) * z & = & x * (y * z) \end{array} \right.$$

3 General Setting of the Inductionless Induction Method

The method roughly works as follows: given a set E of equations, and a conjecture $s = t$, we add $s = t$ to E and try to derive an inconsistency. If this turns out to be impossible, then $s = t$ is an inductive consequence of E. "Inconsistency" here has to be understood w.r.t. some axiomatization of $T(F)/_{=_E}$.

Example 3.1 Let us consider the simplest example (example 1.1): $\{0, s\}$ is a set of free constructors, hence the set $\mathcal{A}$ of (universally quantified) formulas:

$$\begin{cases} s(x) \neq 0 \\ s(x) = s(y) \quad \Rightarrow \quad x = y \end{cases}$$

holds in $T(F)/_{=_E}$. Now, assume that we conjecture $s(x) + 0 = s(0)$. Adding this equation to E, we can derive

$$s(0) \underset{E'}{\longleftrightarrow} s(x) + 0 \underset{E'}{\longleftrightarrow} s(x + 0) \underset{E'}{\longleftrightarrow} s(x).$$

Using $\mathcal{A}$, we may then derive (for all x): $s(0) = s(x) \vdash x = 0$, which contradicts the first formula of $\mathcal{A}$.

More generally, if $\mathcal{A}$ is a set of formulas such that $T(F)/_{=_E} \models \mathcal{A}$ and if $\mathcal{A} \cup E \cup \mathcal{C}$ is inconsistent, then $\mathcal{C}$ cannot be a set of inductive consequences of E.

Now we want to be able (at least sometimes) to conclude that some conjecture is indeed an inductive consequence of E. Let us call an *F-generated algebra*, any quotient of $T(F)$. We first assume the following properties of $\mathcal{A}$:

1. $\mathcal{A}$ is a finite set of purely universal first-order sentences
2. $T(F)/_{=_E} \models \mathcal{A}$
3. For every F-generated model $\mathcal{M}$ of E (or, equivalently, for any congruence $\sim$ on $T(F)$ which contains $=_E$),

$$\mathcal{M} \models \mathcal{A} \text{ implies } \mathcal{M} \approx T(F)/_{=_E}$$

Let us call *I-axiomatization* of $T(F)/_{=_E}$ a finite set of sentences which satisfies the three above properties. The set $\mathcal{A}$ of example 3.1 is an I-axiomatization of the corresponding algebra $T(F)/_{=_E}$. Of course such axiomatizations are, in general, incomplete. However, they have a nice property which is the basis of the method. In what follows, $\mathcal{C}$ is a set of equations (but it can be generalized to a set of purely universal formulas).

Lemma 3.2 *Let $\mathcal{A}$ be an I-axiomatization of $T(F)/_{=_E}$. Assume that $E \cup \mathcal{C} \cup \mathcal{A}$ is consistent then $\mathcal{C}$ is a set of inductive consequences of E.*

Proof: $E \cup \mathcal{C} \cup \mathcal{A}$ has an F-generated model (by property 1 of $\mathcal{A}$ and because any consistent set of purely universal formulae has such a model). Now, by property 3, $E \cup \mathcal{C} \cup \mathcal{A}$ has at most one F-generated model, up to isomorphism. Hence $E \cup \mathcal{C} \cup \mathcal{A}$ has exactly one F-generated model, which is $T(F)/_{=_E}$ by property 2. $\Box$

This is a key lemma since, roughly, it reduces (via $\mathcal{A}$) inductive consequences to equational consequences. It remains, however, to clarify several problems. First, how is it possible to check consistency (or inconsistency) of $E \cup \mathcal{C} \cup \mathcal{A}$? Then how is it possible to find I-axiomatizations?

We delay the problem of finding I-axiomatizations until the next sections (section 4 is dedicated to a number of examples for $\mathcal{A}$). With respect to the first issue, term rewriting plays a central role, but we also need a fourth property of $\mathcal{A}$ which, roughly, expresses some proof-normalization property:

An I-axiomatization $\mathcal{A}$ is *normal* if for any inconsistent set $E \cup \{s = t\} \cup \mathcal{A}$, there is an inconsistency proof consisting in two parts:

1. $E \cup \{s = t\} \vdash u = v$
2. $u = v \cup \mathcal{A} \vdash \Diamond$.

This property expresses that, in order to prove inconsistency, we only need to compute equational consequences, checking for each new equation which is inferred whether it is inconsistent with $\mathcal{A}$ indeed.

Exercise 8.
Give an I-axiomatization of the quotient $T(F)/_{=_E}$ of example 1.1 which is not normal.

Exercise 9.

Show that, if $\mathcal{A}$ is a normal I-axiomatization of $T(F)/_{=_E}$, then $E \cup \mathcal{C} \cup \mathcal{A}$ is inconsistent iff there is an equational consequence $s = t$ of $E \cup \mathcal{C}$ such that $\{s = t\} \cup \mathcal{A}$ is inconsistent.

Now, the set of equational consequences of $E \cup \mathcal{C}$ may be infinite. However, there are some ways of preventing an infinite behaviour of the method. For example, if there is an equivalent convergent rewrite system for $E \cup \mathcal{C}$, then only consistency w.r.t. the rules of this rewrite system have to be considered. This problem will be addressed in section 5. Yet another problem is the decision, for each single equation, of its consistency with $\mathcal{A}$. In several cases, this is quite straightforward (as in example 3.1). For some other examples, it relies on a crux property called *ground reducibility* whose decidability is studied in section 6.

4 Examples of Axiomatizations $\mathcal{A}$ that can be found in the Literature

Up to know, we have set up a very general framework for the proof by consistency (or *inductionless induction*) method. However, historically, this was not explained in this way. The method was actually designed by implicitly (I mean: this was hidden in the inconsistency detection rules) assuming some specific I-axiomatization. In this section, I want to give some examples of proof by consistency techniques which can be found in the literature. For each of these examples, we will precise the (normal) I-axiomatization $\mathcal{A}$. We delay more details on the proof engine itself until next section.

Each example is characterized by

1. the (normal) I-axiomatization
2. the equational deduction method
3. the consistency of $\{s = t\} \cup \mathcal{A}$ decision technique

4.1 Musser's approach [22]

At about the same time (1980), D. Musser [22] and J. Goguen [14] proposed the following method.

F is assumed to contain a particular function symbol eq and two constants $true$ and $false$. (Actually, we would need a particular *sort* for the Booleans, but we skip this irrelevant aspect here). E is a finite set of equations and it is assumed that

$$\begin{array}{rcl} s =_E t & \Leftrightarrow & eq(s,t) =_E true \\ s \neq_E t & \Leftrightarrow & eq(s,t) =_E false \end{array}$$

Which means that eq is "completely defined".

Then, running a completion procedure, a contradiction (and therefore a disproof) is detected when an equation $true = false$ is generated. On the other hand, $s = t$ is an inductive consequence of E if $E \cup \{s = t\}$ can be completed into a canonical rewrite system without generating the inconsistent equation $true = false$.

Actually, these results are straightforward. Indeed, choosing $\mathcal{A}$, the equational deduction method and the inconsistency detection as in figure 2 leads to the following results:

1. $\mathcal{A} = \{true \neq false\}$

2. Equational deduction is given by the Knuth-Bendix completion

3. Inconsistency detection is given by the only rule:

$$true = false, true \neq false \vdash \diamond$$

Figure 2: Musser's Approach

Proposition 4.1 *Under the above assumptions, $\mathcal{A}$ is a normal I-axiomatization.*

Proof: Property 1 is obvious. By hypothesis, $true \neq_E false$ (as soon as $T(F)/_{=_E}$ contains at least two elements, which we actually assume trough all the course), therefore $\mathcal{I} \models \mathcal{A}$. (hence property 2). Now, let $\mathcal{M}$ be an F-generated model of E and s, t be two ground terms. There are four cases:

1. $\mathcal{M} \models s = t$ and $s =_E t$
2. $\mathcal{M} \not\models s = t$ and $s \neq_E t$
3. $\mathcal{M} \not\models s = t$ and $s =_E t$. In this case $\mathcal{M} \not\models E$.
4. $\mathcal{M} \models s = t$ and $s \neq_E t$. In this case, $eq(s,t) =_E false$. Therefore, either $\mathcal{M} \not\models E$ or $\mathcal{M} \models eq(s,t) = false$. In the latter case, $\mathcal{M} \models true = eq(s,s) = eq(s,t) = false$ and therefore $\mathcal{M} \not\models \mathcal{A}$.

As a conclusion, if $\mathcal{M} \models E \cup \mathcal{A}$, then, for every ground terms s, t. $s =_E t$ iff $\mathcal{M} \models s = t$. Hence property 3.

Now, $\mathcal{A}$ is normal since the inconsistency of a set S of equations means that it is possible to derive $true = false$ from S. $\square$

As suggested at the end of section 3, we need not to compute all consequences of $E \cup \mathcal{C}$, hence it is (sometimes) possible to prove inductive theorems:

Proposition 4.2 *If $E \cup \mathcal{C}$ can be completed into a convergent rewrite system for which $true$ and $false$ are irreducible, then $\mathcal{C}$ is a set of inductive consequences of E.*

Proof: It is sufficient to notice that $true =_{E\cup\mathcal{C}} false$ iff they have the same normal form w.r.t the convergent rewrite system. $\square$

Exercise 10.
Consider again example 1.1. Design a set of equations which completely define eq. Then show how the method disproves e.g. $s(x) + 0 = x$. Show as well how the method allows to prove $x + 0 = x$. (Use the completion rules of figure 1).

Now, of course, this approach is much too restrictive in many respects. For example, the requirement that an equality predicate is completely specified is much too strong:

Exercise 11.
Assume that F consists of 0 (constant), s, p (unary) and E contains the two equations

$$\left\{ \begin{array}{lcl} s(p(x)) & = & x \\ p(s(x)) & = & x \end{array} \right.$$

Show that it is not possible to design a finite set of equations E' which completely defines eq (in the above sense). For, assume a sort structure which forbids, e.g. to build $s(true)$ or $eq(true, 0)$,...). Complementary question: which predicates over the integers can be completely defined? (**Note**: there is currently no available solution to this question).

4.2 Huet and Hullot's method [15]

In their 1982 paper Huet and Hullot proposed another variant of the method. They assume that the set of function symbols F contains a subset C of free constructors. The main components of the method are summarized in figure 3.

For example, compare the set $\mathcal{A}$ above with the set given in example 3.1.

Proposition 4.3 *Under the above hypotheses, $\mathcal{A}$ is a normal I-axiomatization.*

$$
\begin{aligned}
1. \mathcal{A} = \quad & \{\forall \vec{x}, \vec{y}, \\
& f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \Rightarrow x_1 = y_1 \wedge \ldots \wedge x_n = y_n \\
& \qquad \mid f \in C\} \\
& \cup \{\forall \vec{x}, \vec{y}, \\
& f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m) \\
& \qquad \mid f, g \in C, f \neq g\}
\end{aligned}
$$

2. equational deduction is given by the Knuth-Bendix completion procedure in which constructor (ground) terms are assumed to be smaller than any non-constructor term.

3. Inconsistency detection consists of using $\mathcal{A}$ to decompose equations headed with constructor symbols.

Figure 3: Huet and Hullot's approach

Proof: Property 1 is obvious. Assume that $x_1, \ldots, x_n.y_1, \ldots, y_m \in T(F)$ and $f(x_1, \ldots, x_n) =_E f(y_1, \ldots, y_n)$ for some $f \in C$. We may actually assume that $x_1, \ldots, x_n, y_1, \ldots, y_n \in T(C)$ because, by hypothesis, each term in $T(F)$ can be proved to be equal to a term in $T(C)$. Then, $f(x_1, \ldots, x_n)$ and $f(y_1, \ldots, y_n)$ are in $T(C)$. Since the constructor are free, they are equal iff they are identical. This shows that the first set of axioms in $\mathcal{A}$ is satisfied in $T(F)/_{=_E}$. The verification is similar for the second set of axioms: $\mathcal{I} \models \mathcal{A}$. Hence property 2.

Assume now that $\mathcal{M}$ is an F-generated model of $\mathcal{A} \cup E$. Let $s, t \in T(F)$. Since $\mathcal{M} \models E$, and by definition of a set of constructors, there are two terms $s' \equiv f(s_1, \ldots, s_n)$ and $t' \equiv g(t_1, \ldots, t_m)$ such that $s', t' \in T(C)$ and $\mathcal{M} \models s = s' \wedge t = t'$. And, since $\mathcal{M} \models \mathcal{A}$, s' and t' must be identical as soon as s and t are equal. This means that $\mathcal{M} \models s = t$ implies $E \vdash s = t$: $\mathcal{M}$ is isomorphic to $\mathcal{I}$.

To prove that $\mathcal{A}$ is normal, the only difficulty is to prove that, when $E \cup \{u = v\}$ is completed into a canonical rewrite system $\mathcal{R}$ without generating an equation $s = t$ between two terms in $T(C, X)$, then $\mathcal{A} \cup \{u = v\} \cup E$ is consistent. This is indeed the case because constructor terms are irreducible w.r.t. $\mathcal{R}$. □

Exercise 12.

Show how $x + 0 = x$ is proved to be an inductive theorem using Huet and Hullot's approach in example 1.1.

Exercise 13.

Show that Huet and Hullot's conditions are equivalent to Musser's conditions. In other words, show any specification containing a set of free constructors can be completed in a conservative extension in which the equality predicate is completely defined.

Remarks

1. Though not necessary, it is also possible to use $\mathcal{A}$ in the completion process. This remark applies for other approaches as well. Actually, in their paper, Huet and Hullot add the following rule to the completion rules:

$$E \cup \{c(t_1, \ldots, t_n) = c(u_1, \ldots, u_n)\}; R \quad \Rightarrow \quad E \cup \{t_1 = u_1, \ldots, t_n = u_n\}; R$$

If $c \in C$. Which correspond to the use of an axiom of $\mathcal{A}$.

2. In view of the exercises 13 and 11, this approach is still too restrictive. We will see in the next section how it can be extended.

3. On the other hand, when compared with the explicit induction techniques, too many (useless) consequences might be deduced by the completion procedure. As shown in e.g. [13], strong restrictions to the completion method can be applied. This will also be discussed in section 5.

4.3 Jouannaud and Kounalis approach [16]

We show here a slight extension of Jouannaud and Kounalis approach. Now, we do not assume any set of free constructors nor any equality predicate. Instead, we assume that the axioms of the specifications are oriented into a finite (ground) convergent rewrite system $\mathcal{R}_0$. Examples 1.2 and 1.5 are easy to complete in order to satisfy this condition.

Exercise 14.
Show that these two examples can be completed to fit with this condition indeed.

Now, the set of axioms $\mathcal{A}$ relies on a *ground reducibility* predicate *Red* whose interpretation is given by:

Definition 4.4 *A term t is* ground reducible *(w.r.t. $\mathcal{R}_0$) if all its ground instances are reducible.*

Red is recursive indeed, but this is quite difficult to prove and section 6 is entirely devoted to this question and the feed-back on some (other) possible axiomatizations. For the moment being, we see *Red* as a black box, but note that, in any case it can be defined by a finite purely universal set of formulas.

Exercise 15.
Give a set of Horn clauses defining *Red*. (i.e. whose minimal interpretation in $T(F)$ is the set of reducible terms).

$\mathcal{A}$ also uses the syntactic equality predicate $\equiv$ (identity on terms), which is easily defined using a purely universal set of formulas. Then, the main components of the methods are described in figure 4.

1. $\mathcal{A} = \{\forall \vec{x}.\ s = t \Rightarrow Red(s) \vee Red(t) \vee s \equiv t\}$

where $\vec{x}$ is the set of variables of s, t.

2. equational deduction is given by the Knuth-Bendix completion procedure.

3. Inconsistency detection is given by the (co)-ground reducibility test.

Figure 4: Jouannaud and Kounalis approach

Actually, $\mathcal{A}$ can be replaced with $s = t \Rightarrow Red'(EQ(s,t))$ where Red' is the ground reducibility predicate w.r.t. $\mathcal{R}_0 \cup \{EQ(x,x) \rightarrow \top\}$ and EQ is a new function symbol. Red' is also called the *co-inductive reducibility* property in [16]. Hence, inconsistency detection amounts to decide ground reducibility and is delayed until section 6.

Proposition 4.5 *$\mathcal{A}$ is a normal I-axiomatization.*

<u>Proof:</u> Property 1. is obvious. Now, let us consider property 2. Assume that for all ground substitution σ, $s\sigma =_E t\sigma$. Then, because $\mathcal{R}_0$ is convergent, there is a rewrite proof of $s\sigma = t\sigma$: $s \rightarrow \ldots \leftarrow t$. This proof contains at least two steps when s is not identical to t. Hence, either $s\sigma \equiv t\sigma$ or else $s\sigma$ or $t\sigma$ is reducible.

Let us consider property 3. Let E be the set $\mathcal{R}_0$ where the rules are considered as (unoriented) equations. Assume $T(F)/_{\sim}$ satisfies $\mathcal{A} \cup E$. Let s, t be two ground terms such that $s \sim t$. We prove by induction w.r.t. the reduction ordering $\rightarrow^*_{\mathcal{R}_0}$ on s, t that $s =_E t$. If s, t are irreducible w.r.t $\rightarrow_{\mathcal{R}_0}$, then, by $\mathcal{A}$, s and t must be identical, hence $s =_E t$. Now, assume that s or t is reducible. For example $s \rightarrow_{\mathcal{R}_0} s_1$. Then $s_1 \sim t$ since $T(F)/\sim$ satisfies E. By induction hypothesis, $s_1 =_E t$, hence $s =_E t$.

Let us finally consider the normality of $\mathcal{A}$. If $s = t$ does not hold in $T(F)/_{=_E}$, then there is a ground instance $s\sigma \neq_E t\sigma$. Let s' be the normal form of $s\sigma$ w.r.t. $\mathcal{R}_0$ and t' be the normal form of $t\sigma$ w.r.t. $\mathcal{R}_0$. $s' = t'$ is an equational consequence of $E \cup \{s = t\}$, $s' \not\equiv t'$ and neither s' nor t' are (ground) reducible. Hence $s' = t' \cup \mathcal{A} \vdash \diamond$, which shows the normality. $\square$

ReDuX is based on Jouannaud and Kounalis approach with many slight improvements. It can be used to try the following examples:

Example 4.6 The following example cannot be proved directly by explicit induction without introducing new lemmas. It can however be automatically proved by the inductionless induction method. Let the specification be

```
TYPE int2.
CONSTS
   0 - int
VARS
   x,y -int
OPS
   +(int,int) - int.
   s(int) - int
AXIOMS
   1) +(0,x) == x
   2) +(s(x),y) == s(+(x,y))
   3) s(s(0)) == 0
END
```

Prove that `+(x,x) == 0`.

Exercise 16.

Consider example 1.2. Prove or disprove the following equations which are conjectured as inductive consequences of E:

1. $s(s(x)) = x$
2. $x + s(0) = s(x)$
3. $s(x) + x = s(0)$
4. $s(x) + s(x) = s(x)$

Consider now the following example:

Example 4.7

```
TYPE Intlists.
CONSTS
     0 - int.
   nil - list
VARS
   n,m - int.
   x,y,z - list
OPS
   s (int) - int.
   + (int,int) -int.
   L (list) - int.
   cons (int,list) - list.
   rev (list) - list.
   @ (list,list) - list
AXIOMS
   1) +(0,n) == n
   2) +(s(n),m) == s(+(n,m))
   3) @(nil,x) == x
   4) @(cons(n,x),y) == cons(n,@(x,y))
   5) L(nil) == 0
   6) L(cons(n,x)) == s(L(x))
   7) rev(nil) == nil
   8) rev(cons(n,x)) == @(rev(x),cons(n,nil))
END
```

Exercise 17.

Prove (or disprove) the following conjectures:

1. `L(@(x,y)) == +(L(x),L(y))`
2. `L(rev(x)) == L(x)`
3. `rev(rev(x)) == x`

The following example is a bit more complex. It describes a group generated by a, b where a has order 3 and b order 2. It is not possible to define a set of free constructors, since $*$ is not free on one hand and it occurs in all terms of some equivalence classes.

Example 4.8 `TYPE group3.`

```
CONSTS
    a,b,e - s3
VARS
    x,y,z - s3
OPS
    * (s3,s3) - s3.
    I(s3) - s3
AXIOMS
 1) I(e) == e
12) I(a) == *(a,a)
 2) *(e,x) == x
 3) *(x,e) == x
 4) I(I(x)) == x
 5) *(I(x),x) == e
 6) *(x,I(x)) == e
 7) I(*(x,y)) == *(I(y),I(x))
 8) *(*(x,y),z) == *(x,*(y,z))
11) *(a,*(a,a)) == e
 9) *(I(x),*(x,z)) == z
10) *(x,*(I(x),z)) == z
13) *(a,*(a,*(a,z))) == z
14) *(b,b) == e
END
```

Exercise 18.

Prove (or disprove) the following conjectures:

1. `*(a,*(a,x)) == *(b,x)`
2. `*(x,*(x,*(x,*(x,*(x,x))))) == e` (i.e. each element has order 6).

More examples will be given in what follows.

5 Inductive Completion

The aim of this section is to refine the completion procedure of section 2, dedicating it to inductive theorem proving. We first present the basic inductive completion procedure, as introduced by [16]. This work is the theoretical foundations of the ReDuX system.

Then, we will detail further refinements which include a *linear strategy*; actually only deductions between the original rewrite systems and conjectures have to be considered. From this original observation of L. Fribourg [13], it is possible to avoid all failure situations and get a refutationally complete completion procedure [2]. Note however that using the linear strategies, it is not possible to simplify original axioms using conjectures, which was possible in Jouannaud and Kounalis's inductive completion technique. Hence both methods are actually incomparable.

In all cases we stick in the following to Jouannaud and Kounalis's approach described in the last section.

5.1 Jouannaud and Kounalis inductive completion

The deduction system is very simple: add the set of conjectures $\mathcal{C}$ to $\mathcal{R}_0$ and complete it using a standard Knuth-Bendix completion (as described in section 2). For each new conjecture $s = t$, the ground reducibility of $s = t$ is checked. If a non-ground reducible equation is generated, then the completion stops with $DISPROOF$. Moreover:

Theorem 5.1 *If the completion of $\mathcal{R}_0 \cup \mathcal{C}$ terminates, leading to a (ground) convergent system R without generating $DISPROOF$, then all equations in $\mathcal{C}$ are inductive consequences of $\mathcal{R}_0$.*

Proof: According to the results of the last section, we only have to show that, for any consequence $s = t$ with $s \not\equiv t$, $s = t$ is ground reducible w.r.t. $\mathcal{R}_0$. Since $\mathcal{R}$ is convergent, we know that the irreducible forms s' and t' of s and t respectively are identical: $s \xrightarrow[R]{*} s' \xleftarrow[R]{*} t$. Assume for example that $s \xrightarrow[l\to r]{p,\sigma} s_1$. Then, by hypothesis, l is ground reducible w.r.t. $\mathcal{R}_0$. Hence $l\sigma \equiv s|_p$ is ground reducible by $\mathcal{R}_0$. $\square$

5.2 Linear restrictions on the inductive completion technique

The main differences with the completion procedures of section the last section are:

1. There is no need of deductions between conjectures: only superpositions between conjectures and the original set of rules need to be computed (this result is due to Fribourg [13]).
2. Among these superpositions only few have to be considered: those which form a *covering set* [13]. This notion of covering set will be explained below; it corresponds, in the case of free constructors, to the induction schemas in the classical explicit inductive proofs.
3. It is possible (though never necessary) to use lemmas which are inductively valid in order to speed up the deductions
4. The procedure is refutationally complete: it cannot fail (this is a consequence of 1.)
5. The procedure can easily be combined with any (other) inductive theorem prover.
6. It is not possible to simplify axioms using conjectures. (At least, it is not known to be possible).

Deduction Rules Let $CP(\mathcal{R}_0, \mathcal{C})$ be the set of *critical pairs* obtained by overlapping a rule of $\mathcal{R}_0$ into an equation of $\mathcal{C}$: if p is a non variable position of s, $s = t \in \mathcal{C}$, $l \to r \in \mathcal{R}_0$ and σ is a m.g.u of $s|_p$ and l, then $t\sigma = s[r\sigma]_p$ is a critical pair.

Figure 5 summarizes the deduction rules of inductive completion. It assumes a reduction ordering $>$ containing $\underset{\mathcal{R}_0}{\longleftrightarrow}$. L stands for *lemmas*, i.e. equations which are indeed inductive consequences of $\mathcal{R}_0$ and $\mathcal{C}$ is a set of conjectures as before.

The purpose of lemmas is to capture all inductive theorems which might have been proved by other means, or to re-use previous deductions. Hence, this part is not necessary here, but it might be useful when combining different theorem provers.

The deduction rules are sound:

Proposition 5.2 *If $L;\mathcal{C} \Rightarrow L';\mathcal{C}'$, then $\mathcal{R}_0 \cup \mathcal{C} \cup \mathcal{A}$ is consistent iff $\mathcal{R}_0 \cup \mathcal{C}' \cup \mathcal{A}$ is consistent.*

Deduce

$$L; \mathcal{C} \;\Rightarrow\; L; \mathcal{C} \cup \{s = t\}$$

If $s = t \in CP(\mathcal{R}_0, \mathcal{C})$.

Deletion

$$L; \mathcal{C} \cup \{s = t\} \;\Rightarrow\; L \cup \{s = t\}; \mathcal{C}$$

If $\mathcal{I} \models s = t$.

Induction

$$L; \mathcal{C} \;\Rightarrow\; L \cup \{s = t\}; \mathcal{C}$$

If $\mathcal{I} \models s = t$.

Simplification

$$L; \mathcal{C} \cup \{s = t\} \;\Rightarrow\; L; \mathcal{C} \cup \{u = t\}$$

If $s \xleftrightarrow[\mathcal{R}_0 \cup L]{+} u$ and $s > u$, or else $s \xleftrightarrow[v=w]{p} u$ for some equation $v = w \in \mathcal{C}$ such that $v \lhd s$ and $v > w$.

Inconsistency

$$L; \mathcal{C} \cup \{s = t\} \;\Rightarrow\; DISPROOF$$

If $s = t$ is not ground reducible (w.r.t. $\mathcal{R}_0$.)

Figure 5: Inductive Completion Rules [2]

As we see in this set of deduction rules, the conjectures need not to be oriented. An advantage is, for example, that we can prove non-orientable inductive consequences of E (see example 5.3 below). Using a classical completion procedure leads to a failure in such a situation: on the opposite the application of rules of figure 5 cannot lead to a failure and actually they are refutationally complete as we will see in the next section.

Example 5.3 Consider again the specification of example 1.1 and the conjecture $x + y = y + x$. The specification can be easily oriented into the following convergent rewrite system:

$$\left\{ \begin{array}{rcl} 0 + x & \rightarrow & x \\ s(x) + y & \rightarrow & s(x + y) \end{array} \right.$$

using e.g. a recursive path ordering with $+ > s$.

Now, consider the conjecture $x + y = y + x$. Overlapping this equation with $\mathcal{R}_0$, we derive the two new equations $x + 0 = x$ and $x + s(y) = s(x + y)$. Now, each of these equations is ground reducible (assume this fact for the moment; we will see how to prove it in section 6). Moreover, each new deduced equation is an instance of an (already computed) conjecture. Hence they need not be computed, according to an adequate redundancy criterion which will be precised later. This shows that $x + y = y + x$ (and all new conjectures) are inductive consequences of E indeed.

5.3 Proof ordering and Refutation completeness

As in section 2, an *equational proof* of $s = t$ is a sequence $s \equiv s_0, \ldots, s_n \equiv t$ where, for each $i < n$, either $s_i \underset{L \cup \mathcal{C}}{\longleftrightarrow} s_{i+1}$ or $s_i \underset{\mathcal{R}_0}{\longrightarrow} s_{i+1}$ or $s_i \underset{\mathcal{R}_0}{\longleftarrow} s_{i+1}$. We define a complexity measure on equational proofs as the multiset of the complexities of each single proof step. The latter being defined by:

$$\text{If } l = r \in \mathcal{C}, \quad c(s \overset{p}{\underset{l=r}{\longleftrightarrow}} t) = \left\{ \begin{array}{rl} (\{s\}, l, t, \perp) & \text{If } l > r \\ (\{t\}, r, s, \perp) & \text{If } r > l \\ (\{s, t\}, \perp, \perp, \perp) & \text{otherwise} \end{array} \right.$$

$$\begin{array}{rl} \text{If } l = r \in L, \quad c(s \underset{l=r}{\longleftrightarrow} t) = & (\perp, \perp, \perp, \{s, t\}) \\ \text{If } l = r \in \mathcal{R}_0, \quad c(s \underset{l \rightarrow r}{\longrightarrow} t) = & (\perp, \perp, \perp, \{s\}) \end{array}$$

Let $\lhd_R$ be the transitive closure of the union of $>$ and the proper subterm ordering. $\lhd_R$ is well-founded. The complexities of single step proofs are compared according to the lexicographic combination of

1. the multiset extension of $\lhd_R$,
2. the encompassment ordering $\lhd$
3. $\lhd_R$
4. the multiset extension of $>$

Equational proofs are compared according to the multiset extension of the above ordering on their complexities. $\perp$ is supposed to be an added as the smallest element w.r.t. all orderings. We write $\gamma \succ^P \gamma'$ when a proof γ is strictly larger than a proof γ'. $\succ^P$ is well-founded by multiset and lexicographic combination of well-founded relations.

Lemma 5.4 *Assume that $L;\mathcal{C} \vdash L';\mathcal{C}'$. Let γ be a proof of $s = t$ in $\mathcal{R}_0 \cup L \cup \mathcal{C}$. Then either γ is a proof of $s = t$ in $\mathcal{R}_0 \cup L \cup \mathcal{C}$ or else there is a proof γ' of $s = t$ in $\mathcal{R}_0 \cup L \cup \mathcal{C}$ such that $\gamma \succ^P \gamma'$.*

Proof: We have to consider here the two rules which may remove equations from one of the set of equations: **Deletion** and **Simplification**.

Consider first **Deletion**. Assume that $u \xleftrightarrow[s=t]{p} v$ and $s = t \in \mathcal{C} \setminus \mathcal{C}'$, $s = t \in L' \setminus L$. Then the complexity of the proof which was originally starting with a non-bottom value, is, after the inference rule has been applied, starting with $\perp$.

Now, consider **Simplification**. Assume that $s \xleftrightarrow[l=r]{p} t$ where $l = r \in \mathcal{C}$ and $l = r$ is simplified into $l' = r$. We have to investigate three situations: $l > r$, $r > l$ or l, r incomparable. Similarly, we have to consider two situations for the simplification, depending on whether l is simplified by another conjecture, or simplified by $L \cup \mathcal{R}_0$. This gives rise to six cases four of which are summarized in the array of figure 6 (the cases where l, r are incomparable are left as exercises).

It is a routine work to check that the complexity of the new proof is smaller than the complexity of the original one. □

Consider a derivation $L_0, \mathcal{C}_0 \vdash \ldots \vdash L_n; \mathcal{C}_n \vdash \ldots$. The set of *persisting conjectures* is defined by:

$$\mathcal{C}_\infty = \bigcup_i \bigcap_{j \geq i} \mathcal{C}_j$$

We call a derivation *fair* when $CP(\mathcal{R}_0, \mathcal{C}_\infty) \subseteq \bigcup_i \mathcal{C}_i$. A refined notion of fairness will be defined in the next section.

	$c(s \xleftrightarrow[l=r]{p,\sigma} t)$	new proof of $s = t$	$c(s \xleftrightarrow[L\cup\mathcal{R}_0\cup\mathcal{C}']{*} t)$
$l > r$ $l \xleftrightarrow[\mathcal{R}_0\cup L]{} l'$	$\{(\{s\}, l, t, \bot)\}$	$s \xleftrightarrow[\mathcal{R}_0\cup L]{+} s[l'\sigma]_p \xleftrightarrow[l'=r]{p,\sigma} t$	$\{(\bot, \bot, \bot, m_1), \ldots,$ $(\{s[l'\sigma]_p\}, l', r, \bot)\}$ If $l' > r$ $\{(\bot, \bot, \bot, m_1), \ldots,$ $(\{t\}, r, l', \bot)\}$ If $r > l'$ $\{(\bot, \bot, \bot, m_1), \ldots,$ $(\{s[l'\sigma]_p, t\}, \bot, \bot, \bot)\}$ otherwise
$l > r$ $l \xleftrightarrow[g=d]{q,\theta} l'$	$\{(\{s\}, l, t, \bot)\}$	$s \xleftrightarrow[g=d]{pq,\theta\sigma} s[d\theta\sigma]_{pq} \xleftrightarrow[l'=r]{p,\sigma} t$	$\{(\{s\}, g, s[d\theta\sigma]_{pq}, \bot),$ $(\{s[d\theta\sigma]_{pq}, l', r, \bot)\}$ If $l' > r$ $\{(\{s\}, g, s[d\theta\sigma]_{pq}, \bot),$ $(\{t\}, g, s[d\theta\sigma]_{pq}, \bot)\}$ If $r > l'$ $\{(\{s\}, g, s[d\theta\sigma]_{pq}, \bot),$ $(\{t, s[d\theta\sigma]_p\}, \bot, \bot, \bot)\}$ otherwise
$r > l$ $l \xleftrightarrow[L\cup\mathcal{R}_0]{+} l'$	$\{(\{t\}, r, s, \bot)\}$	$s \xleftrightarrow[L\cup\mathcal{R}_0]{+} s[l'\sigma]_p \xleftrightarrow[l'=r]{p,\sigma} t$	$\{(\bot, \bot, \bot, m_1), \ldots$ $(\{t\}, r, s[d\theta\sigma]_{pq}, \bot)\}$
$r > l$ $l \xleftrightarrow[g=d]{q,\theta} l'$	$\{(\{t\}, r, s, \bot)\}$	$s \xleftrightarrow[g=d]{pq,\theta\sigma} s[d\theta\sigma]_{pq} \xleftrightarrow[l'=r]{p,\sigma} t$	$\{(\{s\}, g, s[d\theta\sigma]_{pq}, \bot),$ $(\{t\}, r, s[d\theta\sigma]_{pq}, \bot)\}$

Figure 6: Proofs transformations

There are always fair derivation sequences. The only requirement is to avoid delaying indefinitely the computation of some critical pair.

Lemma 5.5 *Let $L_i;\mathcal{C}_i$ be a fair derivation. Then, for any ground terms s, t and for any proof $s \underset{l=r}{\longleftrightarrow} t$ with $l = r \in \mathcal{C}_i$, either $l = r$ is not ground reducible, or else there is an index $j \geq i$ and a proof γ in $L_j;\mathcal{C}_j$ which is strictly smaller (w.r.t $\succ^P$) than $s \underset{l=r}{\longleftrightarrow} t$.*

<u>Proof:</u> Let $s|_p \equiv l\sigma$ and $t \equiv s[r\sigma]_p$. Since $l = r$ is ground reducible, either $l\sigma$ or $r\sigma$ is reducible. Assume e.g. that $l\sigma$ is reducible. If $x\sigma$ itself is reducible for some variable x occurring in l, then we can find a simpler proof by first reducing $x\sigma$ (below each occurrence of x in l) and then using $l = r$ with the reduced substitution: we get a proof

$$s \xrightarrow[\mathcal{R}_0]{+} s' \overset{p,\sigma'}{\underset{l=r}{\longleftrightarrow}} t$$

where σ' is identical to σ, except on x where $x\sigma'$ is the normal form of $x\sigma$. This proof is smaller since all $\mathcal{R}_0$ steps are smaller on one hand and

$$c(s' \overset{p,\sigma'}{\underset{l=r}{\longleftrightarrow}} t) = \begin{cases} (\{s'\}, l, t, \bot) & < (\{s\}, l, t, \bot) = c(s \overset{p,\sigma}{\underset{l=r}{\longleftrightarrow}} t) & \text{If } l > r \\ (\{t\}, r, s', \bot) & < (\{t\}, r, s, \bot) = c(s \overset{p,\sigma}{\underset{l=r}{\longleftrightarrow}} t) & \text{If } r > l \\ (\{s', t\}, \bot, \bot, \bot) & < (\{s, t\}, \bot, \bot, \bot) = c(s \overset{p,\sigma}{\underset{l=r}{\longleftrightarrow}} t) & \text{Otherwise} \end{cases}$$

on the other hand.

Now, assume that σ is irreducible. This means that there is a non-variable position q of l such that $l|_q\sigma \equiv g$ for some left hand side g of a rule $g \to d$ in $\mathcal{R}_0$. Let θ be a m.g.u of $l|_p$ and g. Then σ is an instance of θ: $\sigma = \theta\zeta$. Moreover, by the fairness assumption, the critical pair $l[d]_q\theta = r\theta$ belongs to $\bigcup_i \mathcal{C}_i$. Let $l[d]_q\theta = r\theta \in \mathcal{C}_{i_0}$. We claim that there is a simpler proof of $s = t$ in $L_{i_0};\mathcal{C}_{i_0}$:

$$s \underset{\mathcal{R}_0}{\overset{+}{\longleftrightarrow}} s[l[d]_q\theta\zeta]_p \overset{p,\zeta}{\underset{l[d]_q\theta=r\theta}{\longleftrightarrow}} t$$

Again, we have to compare the complexities of the last proof steps: the relation $s[l[d]_q\theta\zeta]_p < s$ gives the desired relation.

Let $j = \max(i, i_0)$. Now, by lemma 5.4, there is a proof of $s = t$ in $L_j;\mathcal{C}_j$ which is again smaller (or equal) to the above one. Hence strictly smaller than the original proof. □

Now, we are in position to state the main results:

Theorem 5.6 (Refutation Completeness) *Let $L_i; \mathcal{C}_i$ be a fair derivation. Then $\mathcal{C}$ is not a set of inductive consequences of $\mathcal{R}_0$ iff there is some i and some non-ground reducible equation $s = t \in \mathcal{C}_i$.*

Proof: Assume that $s = t \in \mathcal{C}$ is not an inductive consequence of $\mathcal{R}_0$. Then, $\mathcal{A}$ being an I-axiomatization, there is some equational consequence $u = v$ of $\mathcal{R}_0 \cup \{s = t\}$ which is inconsistent. Which means here that $u = v$ is not ground reducible. Let σ be a ground substitution such that both $u\sigma$ and $v\sigma$ are irreducible. By soundness of the deduction procedure, there is a proof of $u\sigma = v\sigma$ in $\mathcal{R}_0 \cup L_\infty \cup \mathcal{C}_\infty$. Consider such a minimal proof of $u\sigma = v\sigma$ (w.r.t. the proof ordering). The proof must contain at least one step $u' \underset{l=r}{\longleftrightarrow} v'$ using a conjecture $l = r \in \mathcal{C}_\infty$. Actually, $l = r \in \mathcal{C}_i$ for some i. By lemma 5.5, fairnes and minimality, $l = r$ is not ground reducible. □

As in section 2 we may stop the completion process when every deduction computes unnecessary equations: a conjecture $s = t$ is *redundant* w.r.t. $\mathcal{R}_0; L; \mathcal{C}$ if, for all irreducible ground substitutions σ, there is a proof in $L; \mathcal{C}$ of $s\sigma = t\sigma$ which is (strictly) smaller than the proof $s\sigma \overset{\Lambda,\sigma}{\underset{s=t}{\longleftrightarrow}} t\sigma$. As a typical example, all critical pairs that are instances of some already computed conjectures are redundant. (This justifies example 5.3). More generally, all critical pairs that can be simplified (using **Simplify**) into trivial equations are redundant.

Only non-redundant critical pairs need to be computed: it is obviously fair to only consider non-redundant critical pairs.

Corollary 5.7 *If all equations in $CP(\mathcal{R}_0, \mathcal{C}_i)$ are redundant and all equations of $\mathcal{C}_i$ are ground reducible, then $\mathcal{C}$ is a set of inductive consequences of $\mathcal{R}_0$.*

In the following exercises, it is not always possible to use the ReDuX system for various reasons:

- ReDuX does not implement the linear strategy, hence there are failure situations

- In case of non-linear left hand sides, the ground reducibility algorithm implemented in ReDuX may fail

Exercise 19.
Prove (using the linear strategy method), in example 1.5 that $x+y = y+x$ is an inductive consequence of the axioms.

Exercise 20.
Consider example 4.7. Disprove the commutativity of `@`.

5.4 Covering sets

It is even possible to restrict further the critical pairs that have to be computed. For, let us introduce some new notions.

The notion of *covering sets* is dual from redundancy: $\mathcal{C}'$ is a covering set for $\mathcal{C}$ (or simply *covers* $\mathcal{C}'$) if each equation of $\mathcal{C}$ is redundant in $\mathcal{C}'$. Or, in other words, if each equation in $\mathcal{C}$ can be proved in $\mathcal{C}'$ using a simpler proof than the trivial one in $\mathcal{C}$.

For example, $CP(\mathcal{R}_0, s = t)$ covers $s = t$ when $s = t$ is ground reducible. This is essentially the result of lemma 5.5. Even more restrictive, if $s > t$, then $CP(\mathcal{R}_0, s \to t)$ (i.e. the set of critical overlaps *in* s *only*) covers $s = t$.

Let us call a position p in a term t *inductively complete* if $t|_p$ is ground reducible. More generally, a set of positions P in t is inductively complete if, for every irreducible ground substitution σ, there is some $p \in P$ such that $t\sigma|_p$ is reducible.

Proposition 5.8 *If s is ground reducible and P is an inductively complete set of positions of s, then the set of overlaps between $s|_p, p \in P$ and $\mathcal{R}_0$ covers $s = t$.*

<u>Proof:</u> It is essentially a remake of the proof of lemma 5.5: consider an irreducible ground substitution σ. $s\sigma$ being ground reducible and P being inductively complete, there is a non variable position $p \in P$ such that $s|_p\sigma$ is reducible by $\mathcal{R}_0$. This means that $s|_p$ and some left hand side l of a rule $l \to r \in \mathcal{R}_0$ are unifiable with m.g.u θ and $\sigma = \theta\zeta$. The overlap of $l \to r$ on s at position p yields the equation $t\theta = s[r\theta]_p$. And the proof

$$s\sigma \xrightarrow[l \to r]{p,\theta\zeta} s[r\theta]_p\zeta \xleftrightarrow[t\theta = s[r\theta]_p]{\Lambda,\zeta} t\sigma$$

is strictly smaller than the proof $s\sigma \overset{\Lambda,\sigma}{\underset{s=t}{\longleftrightarrow}} t\sigma$ $\square$

Corollary 5.9 *When s is ground reducible and P is an inductively complete set of positions of s, then* ***Deduce*** *can be restricted to overlaps of $\mathcal{R}_0$ at a position $p \in P$ of s.*

In practice, it is highly desirable to have handy some inductively complete positions. They can actually be given by sets of terms that are already known to be ground reducible: if such a term occurs at p in s, then p is inductively complete.

Example 5.10 Consider again example 1.1. $x + y$ can be shown to be ground reducible. (This may actually be part of some pre-processing of the original term rewriting system, as we will see in the next section). Now, consider the conjecture $x+(y+z) = (x+y)+z$ and assume that the recursive path ordering extending the precedence $+ > s > 0$ with lexicographic status for $+$. Then position 1 is complete in $(x + y) + z$.

Exercise 21.

Using the specification of example 1.1, show how associativity is proved to be an inductive consequence of E.

Exercise 22.

Let

$$\mathcal{R}_0 = \left\{ \begin{array}{rcl} 0 + x & \rightarrow & x \\ s(x) + y & \rightarrow & s(x + y) \\ 0 * x & \rightarrow & 0 \\ s(x) * y & \rightarrow & y + x * y \end{array} \right.$$

$$L_0 = \left\{ \begin{array}{rcl} x + y & = & y + x \\ x + (y + z) & = & (x + y) + z \end{array} \right.$$

a set of lemmas which have already been proved (see previous examples and exercises).

$$\mathcal{C}_0 = \left\{ \begin{array}{rcl} x * (y + z) & = & (x * y) + (x * z) \\ x * (y * z) & = & (x * y) * z \\ x * y & = & y * x \end{array} \right.$$

Assume moreover that we are using any recursive path ordering extending the precedence $* > + > s > 0$.

Show how the linear strategy succeeds in proving these conjectures.

Exercise 23.

Let the system $\mathcal{R}_0$ of exercise 22 be extended by the two rules

$$= \left\{ \begin{array}{rcl} d(0) & \rightarrow & 0 \\ d(s(x)) & \rightarrow & s(s(d(x))) \end{array} \right.$$

and assume that $d > +$ in the precedence. Show how it is proved that $d(x) = x + x$ is an inductive theorem

Exercise 24.

Let the system $\mathcal{R}_0$ of exercise 22 be extended by the two rules

$$= \left\{ \begin{array}{rcl} x^0 & \rightarrow & s(0) \\ x^{s(y)} & \rightarrow & x * x^y \end{array} \right.$$

The exponential symbol is insterted in the precedence as being larger than $*$. Show how it is proved that $x^{y+z} = x^y * x^z$.

Exercise 25.

```
TYPE fib.
CONSTS
   0 - int
VARS
   n,x,y,z -int
OPS
   +(int,int) - int.
   s(int) - int.
   f(int) - int.
   g(int) - list.
   np(list) - list.
   p(int,int) - list
AXIOMS
   1) +(0,x) == x
   2) +(s(x),y) == s(+(x,y))
```

```
   3) f(0) == 0
   4) f(s(0)) == 0
   5) f(s(s(x))) == +(f(s(x)),f(x))
   6) g(0) == p(s(0),0)
   7) g(s(x)) == np(g(x))
   8) np(p(x,y)) == p(+(x,y),x)
END
```

Prove $g(x) = p(f(s(x), f(x)))$

Exercise 26.

Consider again example 4.7 and the conjecture `rev(rev(x)) == x`. One one hand, Jouannaud and Kounalis inductive completion succeeds in proving the conjecture (this was stated as exercise 17 in section 4). Show, on the other hand, that the linear completion method doesn't terminate on this particular example. (We will analyse this behaviour later).

Remark. As exercise 26 shows, the linear strategy may fail where the classical one succeeds. However, a quick analysis shows that overlapping two conjectures yields the desired simplication. It is possible to allow deductions between conjectures, without requiring their computation. Any of the properties we have seen will still be valid. Hence this example is not really an example showing that the two methods are incomparable.

However, what is really used in the refutation completeness proof is that axioms are not simplified by conjectures. Are there examples for which axioms simplifications would lead a terminating completion whereas without this simplification the completion doesn't terminate? If there are such examples, how is it possible to design a more general completion method which would allow both a linear strategy and to reduce the axioms using conjectures ?

6 Ground Reducibility

The decidability of ground reducibility was shown originally by D. Plaisted [23] and several other algorithms have been proposed since. We sketch here two of them: one is base on *tree automata with constraints* along the lines of [5, 8]. The second one is based on *test sets*, along the lines of [19, 20]. Both of these methods have some feed back on inductive proofs since they provide with some explicit induction scheme. This is exploited for example in ReDuX (ground normal forms grammars computation), but also in SPIKE where, instead of ground normal form grammars, the knowledge of a set covering all irreducible ground terms is exploited. Finally, other consequences will be described in section 6.4.

6.1 Automata techniques

In [5], the authors shown a quite general result. Given a term t, let encomp_t be a unary predicate symbol which is interpreted as the set $[\![\mathrm{encomp}_t]\!]$ of terms encompassing t.

Theorem 6.1 ([5]) *Given n terms $t_1, \ldots, t_n$, the first-order theory of $encompt_1$, $\ldots encompt_n$ is decidable.*

On the other hand, ground reducibility of t w.r.t. R can be expressed by a particular formula in this theory: assuming that $l_1, \ldots, l_n$ are the left hand sides of R, t is ground reducible iff

$$\forall x.\mathrm{encomp}_t(x) \Rightarrow (\mathrm{encomp}_{l_1}(x) \vee \ldots \vee \mathrm{encomp}_{l_n}(x))$$

is valid in the above interpretation.

The proof of this property relies, as in Büchi's theorem on a correspondence between formulas and automata: the authors design a kind of tree automata such that $[\![\mathrm{encomp}_t]\!]$ is accepted by such an automaton. Then the closure properties of the corresponding class of languages and the emptiness decidability leads to the desired property.

It is out of the scope of this lecture to give all details on this result. Nevertheless, we are going to give a flavour of a more restricted result, yet sufficient for ground reducibility, which has some consequences back on inductive proofs. For, we first define *automata with disequality constraints.*

Definition 6.2 *An* automaton with disequality constraints *(over a given finite alphabet F of function symbols) consists in a finite set of states Q, a* *subset* Q_f *of* final states *and a* transition relation*, which is a finite set of constrained rewrite rules of the form*

$$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \ldots, x_n))$$

where $q_1, \ldots, q_n, q$ *are states and* c *is a conjunctions of* disequality constraints *written* $p \neq q$ *where* p, q *are strings of natural numbers.*

The disequality constraints can be viewed as unary predicate symbols and are interpreted as follows: $t \models p \neq q$ iff p, q are positions of t and $t|_p \not\equiv t|_q$. A ground term $t \in T(F \cup Q)$ rewrites to u using the constrained rule $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \ldots, x_n))$ at position p_0 if

$$\begin{cases} t_{p_0} \equiv f(q_1(t_1), \ldots, q_n(t_n)) \\ u \equiv t[q(f(t_1, \ldots, t_n))]_{p_0} \\ t|_{p_0} \models c \end{cases}$$

The reduction relation associated with a production rules P is written (as for rewriting) $\xrightarrow[P]{}$. The variables $x_1, \ldots, x_n$ are always irrelevant when displaying P, hence they are not reported.

Definition 6.3 *A ground term* t *is* accepted *by the automaton* (Q, Q_f, P) *if* $t \xrightarrow[P]{*} q(t)$ *for some* $q \in Q_f$*. The* language $L(\mathcal{A})$ *accepted by an automaton* $\mathcal{A}$ *is the set of ground terms that are accepted by* $\mathcal{A}$*.*

Example 6.4 Let $F = \{0, s, +\}$, $Q = Q_f = \{q_0, q_s, q_+\}$ and

$$P = \begin{cases} 0 & \rightarrow & q_0 \\ s(q_0) & \rightarrow & q_s \\ s(q_s) & \rightarrow & q_s \\ s(q_+) & \rightarrow & q_+ \\ q_s + q_s & \xrightarrow{1 \neq 2} & q_+ \end{cases}$$

The term $s(0) + s(s(0))$ is accepted by the automaton ($s(s(0))$ and $s(0)$ are accepted in state q_s and the last rule can be applied at the root since $s(s(0)) \not\equiv s(0)$), whereas $s(0)+s(0)$ or $(s(0)+s(s(0)))+s(0)$ are not accepted.

These automata may also be non-deterministic (which is not shown on the example).

Now, the main interest of such automata lies in three properties:

Theorem 6.5 *For any rewrite system R, there is an automaton with disequality constraints $\mathcal{A}$ which accepts the set of irreducible ground terms.*

Example 6.6 The automaton of example 6.4 accepts all ground terms that are irreducible w.r.t. the rewrite system

$$\mathcal{R} = \begin{cases} 0 + x & \rightarrow & x \\ x + 0 & \rightarrow & x \\ x + x & \rightarrow & 0 \\ (x + y) + z & \rightarrow & x + (y + z) \\ s(x) + (y + z) & \rightarrow & s(x + (y + z)) \end{cases}$$

The proof of theorem 6.5 is a bit involved. Let us sketch a possible construction. Let $L(R)$ be the set of linear left hand sides of the rewrite system R and $NL(R)$ be the set of non-linear left hand sides. Consider also, for each term $l \in NL(R)$, $\tilde{l}$ be a linearized version of l, i.e the term l in which, at all variable positions distinct new variables are replaced. Let moreover $\hat{l}$ be a renaming of $\tilde{l}$ in such a way that the variables of $\hat{l}$ do not occur anywhere else.

We solve first the following formula:

$$\exists \vec{z}. \forall \vec{y}. \bigwedge_{l \in L(R)} x \neq l \bigwedge_{l \in NL(R)} (x = \hat{l} \vee x \neq \tilde{l})$$

where $\vec{z} = \bigcup_{l \in NL(R)} \mathcal{V}ars(\hat{l})$ and $\vec{y} = \bigcup_{l \in L(R)} \mathcal{V}ars(l) \cup \bigcup_{l \in NL(R)} \mathcal{V}ars(\tilde{l})$.

The purpose of the formula is to express that x is not reducible at the root position by any left linear rule. (This is the first conjunction). The purpose of the second part is to express that either x matches a linearized version of $l \in NL(R)$ or else is not unifiable with it. (This last part could be omitted since it always holds. But we need to guess between these two possibilities).

Such a formula can be reduced[1] to a finite disjunction of *solved forms*:

$$\bigvee_{u \in C(R)} \exists \vec{w}. x = u$$

where, for each $l \in L(R)$, u is not unifiable with l and, for each $l \in NL(R)$, either u is not unifiable with $\tilde{l}$ or else u is an instance of $\tilde{l}$. Moreover u is

[1]We skip here the reduction procedure. See for example [?] where a reduction of general equational formulas is investigated.

linear and its depth is smaller than the maximal depth of a left hand side of R.

Moreover, the disjunction of solved forms is equivalent to the original formula in $T(F)$. (i.e. the same ground substitutions satisfy the two formulas).

Example 6.7 Let $R_1 = \{0 + x \to x; s(x) + y \to s(x + y)\}$ and the alphabet be $\{0, s, +\}$, then $C(R_1) = \{0; s(x); (x + y) + z\}$.

Example 6.8 Let $R_2 = \{x + x \to 0; 0 + x \to x; x + 0 \to x; s(s(0)) \to 0\}$, and the alphabet be $\{0, s, +\}$. Then $C(R_2) = \{s(x) + s(y); s(x + y); s(0); 0; s(s(x + y)); s(s(s(x))); x + (y + z); (x + y) + z\}$.

For each term s in $C(R)$ we build a state q_s. (Two terms which only differ in the names of their variables are considered to be identical).

If s is not a variable (which is always the case when R is non-empty) then, $s \equiv f(s_1, \ldots, s_n)$ and we add the production rules $f(q_{t_1}, \ldots, q_{t_n}) \xrightarrow{c} q_s$ for all $t_1, \ldots, t_n \in C(R)$ which are instances of $s_1, \ldots, s_n$ respectively and where c is the conjunction, for all $l \in NL(R)$ such that s is an instance of $\tilde{l}$ of the disjunction of constraints $p \neq q$ for each pair of positions of identical variables in l.

All states are final states. This gives a construction of the automaton $\mathcal{A}$ which accepts irreducible ground terms.

Example 6.9 Following example 6.7 we get the automaton

$$Q = \{q_0, q_{s(x)}, q_{(x+y)+z}\},$$

with production rules:

$$\begin{array}{rcl}
0 & \to & q_0(0) \\
s(q_0) & \to & q_{s(z)} \\
s(q_{s(z)}) & \to & q_{s(z)} \\
s(q_{(z_1+z_2)+z_3}) & \to & q_{s(z)} \\
q_{(z_1+z_2)+z_3} + q_0 & \to & q_{(z_1+z_2)+z_3} \\
q_{(z_1+z_2)+z_3} + q_{s(x)} & \to & q_{(z_1+z_2)+z_3} \\
q_{(z_1+z_2)+z_3} + q_{(z_1+z_2)+z_3} & \to & q_{(z_1+z_2)+z_3}
\end{array}$$

We may actually simplify the notations, removing the arguments of states which are useless in this context. We may also use, again for sake of simplicity, a state q_x and ϵ-transitions from all states to q_x: then we get the

$$
\begin{array}{rcl@{\qquad}rcl}
q_{s(0)} + q_{s(0)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{(s(0)} + q_{s(x+y)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(0)} + q_{s(s(x+y))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(0)} + q_{s(s(s(x)))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(s(x+y))} + q_{s(0)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(s(x+y))} + q_{s(s(x+y))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(s(x+y))} + q_{s(s(s(x)))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(s(x+y))} + q_{s(x+y)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(s(s(x)))} + q_{s(0)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(s(s(x)))} + q_{s(s(x+y))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(s(s(x)))} + q_{s(s(s(x)))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(s(s(x)))} + q_{s(x+y)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(x+y)} + q_{s(0)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(x+y)} + q_{s(x+y)} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
q_{s(x+y)} + q_{(s(s(x+y))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} & q_{s(x+y)} + q_{s(s(s(x)))} & \xrightarrow{1\neq 2} & q_{s(x)+s(y)} \\
s(q_{s(x)+s(y)}) & \rightarrow & q_{s(x)} & s(q_{x+(y+z)}) & \rightarrow & q_{s(x)} \\
s(q_{(x+y)+z}) & \rightarrow & q_{s(x)} & s(q_{x+(y+z)} & \rightarrow & q_{s(x)} \\
s(q_0) & \rightarrow & q_{s(0)} & 0 & \rightarrow & q_0 \\
s(q_{s(x+y)}) & \rightarrow & q_{s(s(x+y))} & s(q_{s(s(s(x)))}) & \rightarrow & q_{s(s(s(x)))} \\
q_{s(x)+s(y)} + q_x & \xrightarrow{1\neq 2} & q_{(x+y)+z} & q_{x+(y+z)} + q_x & \xrightarrow{1\neq 2} & q_{(x+y)+z} \\
q_{(x+y)+z} + q_x & \xrightarrow{1\neq 2} & q_{(x+y)+z} & q_x + q_{s(x)+s(y)} & \xrightarrow{1\neq 2} & q_{x+(y+z)} \\
q_x + q_{x+(y+z)} & \xrightarrow{1\neq 2} & q_{x+(y+z)} & q_x + q_{(x+y)+z} & \xrightarrow{1\neq 2} & q_{x+(y+z)}
\end{array}
$$

Figure 7: Production rules of example 6.8

rules

$$
\begin{array}{rcl}
0 & \rightarrow & q_0 \\
s(q_x) & \rightarrow & q_{s(z)} \\
q_{(z_1+z_2)+z_3} + q_x & \rightarrow & q_{(z_1+z_2)+z_3}
\end{array}
$$

Note that the last rule is looping and $q_{(z_1+z_2)+z_3}$ does not occur anywhere else in a right hand side. This means that the corresponding language is empty, hence the rule can be removed (but this is part of the cleaning process and we are anticipating).

Example 6.10 Following example 6.8,

$$Q = \{q_{s(x)+s(y)}; q_{s(x+y)}; q_{s(0)}; q_0; q_{s(s(x+y))}; q_{s(s(s(x)))}; q_{x+(y+z)}; q_{(x+y)+z}\}.$$

The production rules are given in figure 7.

The set of production rules is actually quite large, but it can be reduced as we will see later since many production rules are useless.

Exercise 27.
What is the size of $C(R)$ w.r.t. the size of R ?

Exercise 28.
Assume that $\mathcal{A}$ is an automaton with disequality constraints accepting the set of all irreducible ground terms. Show that there is an automaton with disequality constraints $\mathcal{A}'$ which accepts the set of equations $s = t$ such that s, t are distinct irreducible ground terms.

Theorem 6.11 *Given an automaton with disequality constraints $\mathcal{A}$, it is decidable whether $L(\mathcal{A})$ is empty, resp. finite. (And in the latter case, compute a bound on the size of the language).*

The proof of this theorem is based on *tree transformations* (see e.g. [8]), which generalize pumping properties: for a well chosen ordering on trees, there is a bound on the depth of a minimal tree accepted by the automaton. The bound is roughly exponential in the depth of the constraints in the automaton and polynomial in the number of production rules. For example, if the set of rules in R are left linear (then the production rules do not check any constraint at all), then the decision of finiteness (resp. emptiness) are performed in linear time.

Using these techniques, we can simplify the automata of examples 6.9 and 6.10. Let $\mathcal{A}_q$ be the automaton which is obtained by replacing the final states of $\mathcal{A}$ with $\{q\}$.

Example 6.12 Following example 6.9, the automaton can be cleaned: mark first the states that can be reached in one step. (Only q_0 and q_x are marked). Then mark the states that can be reached from marked states using one of the production rules; repeat this operation until no new mark state is produced. In the example, $q_{s(z)}$ is marked at the second step. Then no more state can be marked. The process stops. Now, each non marked state is unreachable, hence the corresponding language is empty. This leads to the simplified automaton accepting ground normal forms:

$$0 \rightarrow q_0 \quad s(q_0) \rightarrow q_{s(z)} \quad s(q_{s(z)}) \rightarrow q_{s(z)}$$

(We expand now q_x because the size became reasonable).

Similarly, let us double mark states q which either can be reached from state q or from a double marked state. Again, after two steps, only one state is double marked: $q_{s(x)}$ and there is no progress. The double marked states q exactly correspond to automata $\mathcal{A}_q$ generating an infinite language. The remaining state q_0 corresponds to a language $L(\mathcal{A}_{q_0})$ containing all possible ways of reaching q_0 (in less steps than the number of steps we needed so far). In the example, $L(\mathcal{A}_{q_0}) = \{0\}$.

Example 6.13 Now, we show how to clean the automaton of example 6.10 using a similar technique. Roughly, the difference is now that we need to go twice through each state, to ensure the satisfiability of constraints. At the first step, we mark q_0. At the second step, we mark $q_{s(0)}$ and next, no state can be marked or double marked. Hence we stop with the clean automaton:

$$0 \to q_0 \quad s(q_0) \to q_{s(0)}$$

All other states correspond to empty languages. We also deduce that the language of ground normal forms is finite and contains only two elements: $0, s(0)$.

Now, let us show how this can be applied to ground reducibility.

Let t be a term. Deciding the ground reducibility of t amounts to decide whether there is an irreducible ground instance of t. Such an instance exists iff there is an irreducible ground substitution σ such that $t\sigma$ is not reducible at any non variable position of t. In other words, t is not ground reducible iff the following formula is satisfiable:

$$\forall \vec{y}. \bigwedge_{l \to r \in R} \bigwedge_{p \in \mathcal{F}Pos(t)} t|_p \neq l \bigwedge_{x \in \mathcal{V}ars(t)} x \in q_f$$

where $\vec{y}$ is the set of variables occurring in all left hand sides l, and which is supposed to be disjoint from the set of variables of t.

$x \in q_f$ is interpreted as the set of all irreducible ground terms, i.e. the set of terms recognized by the automaton $\mathcal{A}$.

Now, this formula is equivalent to a finite disjunction of formulas

$$\exists \vec{z}. \bigwedge_{x \in \mathcal{V}ars(t)} x = s_x \wedge x \in q_f \bigwedge_{i=1}^{m} z_i \neq t_i$$

where each s_x is a subterm of some $t_x \in C(R)$ and each t_i is different from z_i. (See e.g. [6, 9] for more details on the procedure). Now $x = s_x \wedge x \in q_f$ can be turned into the disjunction for all terms u in $C(R)$ that match s_x of

$x \in q_u$. Actually, this can be restricted to the terms $u \in C(R)$ which are still indices of states after the automaton has been cleaned up.

Moreover, we may assume that, for all i, $L(\mathcal{A}_{q_i})$ is infinite: it is sufficient to apply the rule

$$x \in q \wedge P \quad \to \quad (x = t_1 \wedge P\{x \to t_1\}) \vee \ldots \vee (x = t_n \wedge P\{x \to t_n\})$$

If $L(\mathcal{A}_{q_i}) = \{t_1, \ldots, t_n\}$

Finally, we remove trivially unsatisfiable problems, i.e. those containing trivial disequations $u \neq u$. If no problem is remaining, then t is ground reducible. Otherwise, there is a problem of the form

$$\bigwedge_{x \in \mathcal{V}ars(t)} x \in q_{u_x} \bigwedge_{i=1}^{m} u_i \neq v_i$$

where all q_{u_x} correspond to infinite languages. In such a situation, t is not ground reducible since such problems always have solutions (see e.g. [**?**]).

Example 6.14 Let us come back to example 6.13. Assume that we want to check the ground reducibility of $s(s(x))$. We start by solving $s(s(x)) \neq s(s(0)) \wedge s(x) \neq s(s(0))$ (plus several disequalities which immediately yield a clash). We get

$$\exists y, z. x = y + z \vee \exists y. x = s(s(y)) \vee \exists y, z. x = s(y + z)$$

None of the states q_u where u matches $y+z$ or $s(s(y))$ or $s(y+z)$ is remaining in the automaton of example 6.13, hence $s(s(x))$ is ground reducible.

Exercise 29.

Compute the clean automaton corresponding to the specification

```
TYPE Mod2b.
CONSTS
   0 - int
VARS
   x,y -int
OPS
   +(int,int) - int.
   s(int) - int.
   g(int) - int
```

```
AXIOMS
   1) +(0,x) == 0
   2) +(x,0) == 0
   3) +(x,x) == 0
   4) +(g(x),y) == 0
   5) +(x,g(y)) == 0
   6) s(s(0)) == 0
END
```

and check the ground reducibility of the terms $g(g(x)), x + y$.

What is changed if the following rule is added:

```
   7) s(g(x)) = 0
```

6.2 Test sets

The basic idea of test sets is very simple: find a finite set of terms TS such that t is ground reducible iff all its instances obtained by replacing its variables by a term in TS are reducible.

Such test sets TS always exist (this is basically Plaisted's result [23]), but they may be very large. How is it possible to compute the test sets? Again, the naive computation is quite simple. We build a tree (called a *reducibility tree*). We call i-instance of a term t any ground instance $t\sigma$ of t such that, for every variable x of t, $x\sigma$ is irreducible.

The root of the reducibility tree is labeled with $t = x$ (a variable). Then we repeat the following steps:

1. Select leaf of the tree whose label t may have reducible i-instances. (i.e. some of its non-variable subterms is unifiable with a left hand side of a rule). If there is no such leaf, then stop.

2. Select a variable x of t (t is not ground because none of the leaves are reducible). Consider the set $S(x,t)$ obtained by splitting x over the signature. More precisely, $S(x,t) = \{t\{x \to f(\vec{y})\} \mid f \in F\}$. Let $S'(x,t)$ be the subset of irreducible terms in $S(x,t)$.

3. If $S'(x,t)$ is empty, then remove the leaf labeled with t from the tree and remove the parents which have no more child by this transformation.

4. If $S'(x,t)$ is not empty, then grow from the node labeled with t as many child as there are elements in $S'(x,t)$ and label them with the elements of $S'(x,t)$.

A test set is given by the leaves of the reducibility tree.

Of course, this simple way of doing is not terminating in general. It is however possible to give a bound B on the depth of the terms labeling a node of the reducibility tree. If, in the above algorithm, it is not possible to add a node labeled with a term of size smaller than B, then the computation stops.

An invariant of this computation is that the set of irreducible ground terms is always contained in the set of all i-instances of the leaves.

The bound B is really huge (from 5 to 7 exponentials depending on the version). It is however possible to restrict the computation in various ways. One of the main remark is the following: if, at some computation step, all leaves of a subtree labeled with t are labeled with terms strictly containing some instance of t then we can remove the whole subtree labeled with t. Indeed, assume that $t\sigma$ is a minimal (w.r.t. size) irreducible ground instance of t, then it should be an instance of some leave of the subtree labeled with t, say $t\sigma \equiv u\theta$. However, by hypothesis, there is some strict subterm of u containing an instance of t. This means that there is an irreducible ground instance of t which is strictly smaller than $t\sigma$. Hence a contradiction.

Example 6.15 Let us go back again to example 1.1. In order to compute a test set, we start with the variable x. Then split x into 0, $s(x)$ and $x+y$. The two first terms are not unifiable with a left hand side and are thus leaves of the reducibility tree. Remains $x+y$, which is split (say, according to the first variable). We get three terms $0+y$, which is reducible, $s(x)+y$ which is reducible, and $(x+z)+y$. Then all descendants of $(x+y)$ contain an instance of $x+y$: this node can be removed.

The reducibility tree now only contain one node and two leaves labeled respectively by 0 and $s(x)$. We got a test set containing two elements.

Of course test sets correspond to induction schemes and test sets computations can be seen as computing an induction scheme which was not explicit. There are however more complicated situations. Let us give a (still quite simple) example.

Example 6.16 Consider example 1.5. Starting the computation of the reducibility tree, we get first $0, s(x), p(x), x+y$. 0 is irreducible and need not be further considered. The three other leaves are however unifiable with left hand sides. Thus, we have to go on adding new leaves; we get the following tree (after some steps):

$$\begin{array}{llll}
x & 0 & & \\
 & s(x) & s(0) & \\
 & & s(s(x)) & s(s(0)) \\
 & & & s(s(s(x))) \\
 & & & s(s(x+y)) \\
 & & s(x+y) & s((x+y)+z) \\
 & & & s(x+(y+z)) \\
 & p(x) & p(0) & \\
 & & p(p(x)) & p(p(0)) \\
 & & & p(p(p(x))) \\
 & & & p(p(x+y)) \\
 & & p(x+y) & p((x+y)+z) \\
 & & & p(x+(y+z))
\end{array}$$

The node labeled with $x+y$ has been removed according to the criterion: all its descendents contained some instance of it. Then, actually, it is possible to remove in the tree all terms containing $x+y$ (removing a node means that we already proved its ground reducibility). Now, we have the tree

$$\begin{array}{lllll}
x & 0 & & & \\
 & s(x) & s(0) & & \\
 & & s(s(x)) & s(s(0)) & \\
 & & & s(s(s(x))) & s(s(s(0))) \\
 & & & & s(s(s(s(x))))\ldots \\
 & p(x) & p(0) & & \\
 & & p(p(x)) & p(p(0)) & \\
 & & & p(p(p(x))) & p(p(p(0))) \\
 & & & & p(p(p(x)))\ldots
\end{array}$$

whose depth depend on the bound B.

It should be clear from this example that the test set is not "optimal" in the last example. To precise this point and have some feedback on the inductionless induction method, let us study the ground normal form languages.

Exercise 30.

In the last example, what is the automaton which accepts all irreducible ground terms ?

Exercise 31.
Compute a test set for the system of example 1.2 (assuming all equations oriented from left to right).

6.3 Ground normal form languages

In both approaches of the last two sections, the aim is actually to find a description of the irreducible ground terms. In the fist case we got a description via tree automata, while in the test set approach, we got a description using a set of i-instances of a test set.

Actually the test set method does not lead to a real description of irreducible ground terms but to an approximation which is sufficient for ground reducibility purposes only. That is why we stick now to the automata approach.

Assume we have an automaton $\mathcal{A}$ which accepts the set of irreducible ground terms and such that none of the states is useless (i.e. the corresponding language is not empty).

So far, we didn't consider type information. But it is of course possible to generalize the inductionless induction method including typing informations. More precisely, we may, instead of pure equational logic consider a logic in which variables are constrained to range over some tree languages.

Example 6.17 Let us define *bool* and *int* as the tree languages accepted in the corresponding states q_{bool} and q_{int} of the automaton whose rules are

$$\begin{array}{rcl@{\qquad}rcl}
0 & \rightarrow & q_{int} & s(q_{int}) & \rightarrow & q_{int} \\
p(q_{int}) & \rightarrow & q_{int} & q_{int}+q_{int} & \rightarrow & q_{int} \\
E(q_{int}) & \rightarrow & q_{bool} & true & \rightarrow & q_{bool} \\
false & \rightarrow & q_{bool} & & &
\end{array}$$

We will actually write *int* instead of q_{int} and *bool* instead of q_{bool}.

Then the following rules could be a specification of the function symbols:

$$\left\{\begin{array}{rcll} s(p(x)) & \rightarrow & x & \mid\ x \in int \\ p(s(x)) & \rightarrow & x & \mid\ x \in int \\ 0 + x & \rightarrow & x & \mid\ x \in int \\ s(x) + y & \rightarrow & s(x+y) & \mid\ x, y \in int \\ p(x) + y & \rightarrow & p(x+y) & \mid\ x, y \in int \\ E(0) & \rightarrow & true & \\ E(s(0)) & \rightarrow & false & \\ E(s(s(x))) & \rightarrow & E(x) & \mid\ x \in int \end{array}\right.$$

Deduction processes may be affected in general by these extensions. But this would lead us too far to consider again a proof system in this context. For sake of simplicity, we will assume that the sort constraints always satisfy conditions under which exactly the same deduction rules as before are sound. (For example, the *sort decreasingness* property is a sufficient condition; see e.g. [25]).

Now, introducing new sorts, corresponding to the states of the automaton $\mathcal{A}$, we may compute from the original specification, another specification containing new sorts and new function symbols and which is equivalent to the original one, with additional properties. Here, by "equivalent" we mean that equalities built on the original signature are inductive consequences of the original set of axioms iff they are inductive consequences of the new set of axioms.

Let us illustrate the transformation on an example.

Example 6.18 Consider the specification of example 1.5. It is not difficult to see that the automaton accepting the irreducible ground terms computed by the method described in section 6.1 is given by the production rules:

$$\begin{array}{rcl@{\qquad}rcl} 0 & \rightarrow & q_0 & s(q_0) & \rightarrow & q_{s(x)} \\ s(q_{s(x)}) & \rightarrow & q_{s(x)} & p(q_0) & \rightarrow & q_{p(x)} \\ p(q_{p(x)}) & \rightarrow & q_{p(x)} & & & \end{array}$$

(This has been actually set up as an exercise).

We may now call q_0 *zero*, $q_{s(x)}$ *pos* and $q_{p(x)}$ *neg*. We also add a sort *int* containing all terms. We introduce two new function symbols s' and p' and claim that the following specification is equivalent to the original one.

$$
\begin{array}{rclcrcl}
0 & \to & zero & \qquad & s(int) & \to & int \\
s'(zero) & \to & pos & & p(int) & \to & int \\
s'(pos) & \to & pos & & int + int & \to & int \\
p'(zero) & \to & neg & & p'(neg) & \to & neg
\end{array}
$$

assuming moreover for sake of simplicity the ϵ-transitions $zero \to int$, $neg \to int$ and $pos \to int$. And the following set of axioms in addition to the original one:

$$
\left\{
\begin{array}{rcll}
s(x) & \to & s'(x) & \mid \; x \in pos \vee x \in zero \\
p(x) & \to & p'(x) & \mid \; x \in neg \vee x \in zero \\
s(p'(x)) & \to & x & \mid \; x \in neg \vee x \in zero \\
p(s'(x)) & \to & x & \mid \; x \in pos \vee x \in zero \\
s'(x) + y & \to & s(x+y) & \mid \; (x \in pos \vee x \in zero) \wedge y \in int \\
p'(x) + y & \to & p(x+y) & \mid \; (x \in neg \vee x \in zero) \wedge y \in int
\end{array}
\right.
$$

How to compute this new set of axioms is out of the scope of these notes (see [7] instead).

Now, the main property of this new specification is that $\{0, s', p'\}$ is a set of free constructors! (We leave the proof as an exercise).

Hence it is possible to compute an equivalent (typed) specification for which there are now free constructors. This is not really surprising: the automaton provides the induction schemes that have to be followed in proving inductive properties.

But we cannot conclude directly that only the free constructor case has to be considered. Indeed, when the rewrite systems contain non linear left hand sides, the typing rules may involve disequality constraints. What does equational deduction become in such typed systems is unknown.

Exercise 32.
Find an example of a rewrite system containing non-linear left hand sides which are not instances of any other left hand sides and such that the set of irreducible ground terms is still accepted by a tree automaton without constraints.

6.4 Sufficient Completeness

Sufficient completeness is used in other areas than automated deduction, for example to check that a function defined by pattern matching is indeed completely defined. We have also seen that in some situations, it is important to decide whether some subset of function symbols is a set of constructors indeed.

Actually, sufficient completeness is strongly related to ground reducibility and that is why we add some words on this topics here. First, this property is undecidable, even in restricted situations:

Proposition 6.19 ([19]) *It is undecidable whether a subset C of F is a set of constructors w.r.t E, even when E is a finite convergent string rewrite system.*

But this can be recovered as follows:

Proposition 6.20 ([16]) *Let $\mathcal{R}$ be a convergent rewrite system for which normal forms of terms in $T(C)$ are in $T(C)$. Then C is a set of constructors iff, for each $f \in F \setminus C$, $f(x_1, \ldots, x_n)$ is ground reducible.*

Exercise 33.
Prove the above proposition.

Hence techniques that have been developed for ground reducibility apply for sufficient completeness as well: there is no need of further developments.

7 Extensions: limitations of the method

In all previous sections, we only considered equational axioms. Everything can actually be extended to conditional equations (i.e. Horn clauses with equality). The only main provision is that ground reducibility is undecidable for conditional equations. Hence we must know, by other means, a test set, or an automaton accepting irreducible ground terms.

The linear strategy described in section 5 is also refutationally complete for conditional equations (see [13]). Deductions have of course to include superpositions (paramodulation) in the conditions of the rules. From a practical point of view, the main problem is the termination of the completion process!

Another possible extension would be to consider equational rewriting or ordered rewriting (see [2]). This would be useful in order to handle, e.g. associative and commutative symbols. However, ground reducibility is again undecidable in presence of associative commutative symbols [18]. Ground reducibility is also undecidable for ordered rewriting. Again, assuming some black box issuing a test set or an automaton, it is possible to consider these extensions. But the inductionless induction method can probably be competitive only in case of equational specifications.

Anyway, ordered rewriting and rewriting modulo seem to be of little use in the framework of inductive proofs since unorientable equations may be often considered as lemmas. That is what we did for example in exercise 22.

On the other hand, if we try to compare with explicit induction methods, the main advantages of inductionless induction seem to be :

- The power of the induction which is related to the ordering used to prove the termination of the rewrite system.
- The refutation completeness and hence the ability to disprove conjectures.
- The automatic discovery of intermediate lemmas (see e.g. example 4.6)
- The ability to discover induction schemes: no explicit constructors are needed.

On the other hand, there are some drawbacks:

- the original presentation has to be a convergent rewrite system (a restriction which is droped in [24])

- So far, only pure equational theories can be handled. But intermediate methods as in [4] seem to work quite well.

References

[1] L. Bachmair. Proof by consistency in equational theories. In *Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh*, July 1988.

[2] L. Bachmair. *Canonical Equational Proofs*. Birkhäuser, Boston, 1991.

[3] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*, pages 346–357, June 1986.

[4] A. Bouhoula. *Preuves Automatiques par Récurrence dans les Théories Conditionnelles*. PhD thesis, University of Nancy 1, France, 1994.

[5] A.-C. Caron, J.-L. Coquidé, and M. Dauchet. Encompassment properties and automata with constraints. In C. Kirchner, editor, *Proc. 5th. Int. Conf. on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, vol. 690, Montreal, Canada, 1993. Springer-Verlag.

[6] H. Comon. Unification et disunification: Théorie et applications. Thèse de Doctorat, Institut National Polytechnique de Grenoble, France, 1988.

[7] H. Comon. Inductive proofs by specifications transformation. In *Proc. 3rd Rewriting Techniques and Applications, Chapel Hill, LNCS 355*, pages 76–91. Springer-Verlag, Apr. 1989.

[8] H. Comon and F. Jacquemard. Ground reducibility and automata with disequality constraints. In P. Enjalbert, editor, *Proc. 11th Symp. on Theoretical Aspects of Computer Science*, Caen, 1994. Springer-Verlag.

[9] H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.

[10] N. Dershowitz. Canonical sets of horn clauses (extended abstract). University of Illinois, Nov. 1990.

[11] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[12] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, Aug. 1979.

[13] L. Fribourg. A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8:253–276, 1989.

[14] J. A. Goguen. How to prove inductive hypothesis without induction. In *Proc. 5th Conf. on Automated Deduction, Les Arcs, France, LNCS 87*, July 1980.

[15] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *J. Comput. Syst. Sci.*, 25(2), 1982.

[16] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1), July 1989.

[17] D. Kapur and D. Musser. Proof by consistency. *Artificial Intelligence*, 31(2), Feb. 1987.

[18] D. Kapur, P. Narendran, D. Rosenkrantz, and H. Zhang. Sufficient completeness, ground reducibility and their complexity. *Acta Inf.*, 28:311–350, 1991.

[19] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Inf.*, 24(4):395–415, 1987.

[20] E. Kounalis. Testing for the ground (co)-reducibility in term rewriting systems. *Theoretical Comput. Sci.*, 106(1):87–117, 1992.

[21] D. S. Lankford. A simple explanation of inductionless induction. Technical Report MTP-14, Mathematics Department, Louisiana Tech. Univ., 1981.

[22] D. Musser. Proving inductive properties of abstract data types. In *Proc. 7th ACM Symp. on Principles of Programming Languages, Las Vegas*, 1980.

[23] D. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65:182–215, 1985.

[24] U. Reddy. Term rewriting induction. In *Proc. 10th Int. Conf. on Automated Deduction, Kaiserslautern, LNCS 449*, 1990.

[25] U. Waldmann. Semantics of order-sorted specifications. *Theoretical Comput. Sci.*, 94:1–35, 1992.

8 Solutions of Exercises

Exercise 1 Let s be the identity. Then $s(x)+y = s(x+y)$ is automatically satisfied, independently of the definition of $+$. Now, let $0 + 0 = 0, 0 + a = a, a + a = a, a + 0 = 0$.