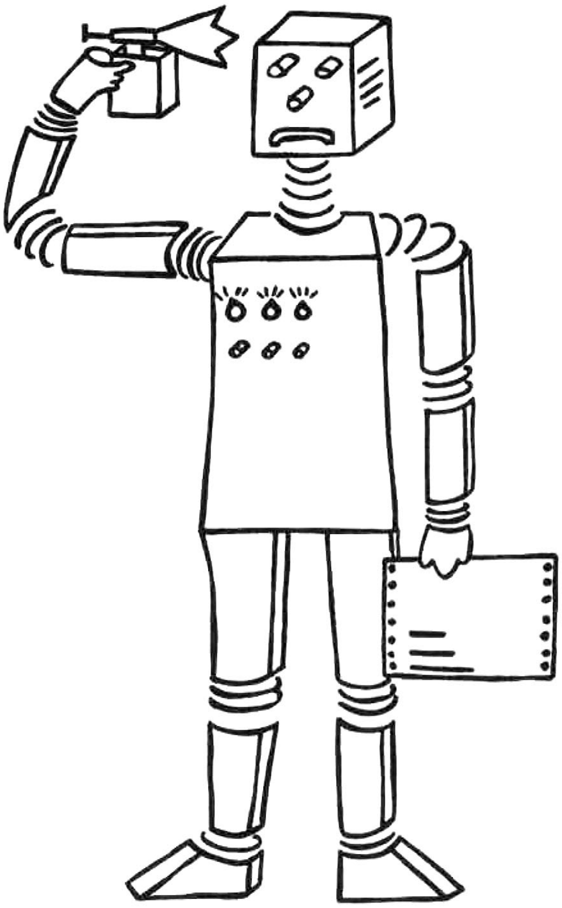
SEKI-REPORT

Artificial
Intelligence
Laboratories

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



Basic Narrowing Revisited

Werner Nutt, Pierre Réty and Gert Smolka

July 1987 SEKI-REPORT SR-87-07

SEKI Report SR-87-07, Universität Kaiserslautern, West Germany, July 1987

Basic Narrowing Revisited

Werner Nutt
Universität Kaiserslautern

Pierre Réty
Centre de Recherche en Informatique de Nancy

Gert Smolka
Universität Kaiserslautern

Abstract

In this paper we study basic narrowing as a method for solving equations in the initial algebra specified by a ground confluent and terminating term rewriting system. Since we are interested in equation solving, we don't study basic narrowing as a reduction relation on terms but consider immediately its reformulation as an equation solving rule. This reformulation leads to a technically simpler presentation and reveals that the essence of basic narrowing can be captured without recourse to term unification.

We present an equation solving calculus that features three classes of rules. Resolution rules, whose application is don't care nondeterministic, are the basic rules and suffice for a complete solution procedure. Failure rules detect inconsistent parts of the search space. Simplification rules, whose application is don't care nondeterministic, enhance the power of the failure rules and reduce the number of necessary don't know steps.

Three of the presented simplification rules are new. The rewriting rule allows for don't care nondeterministic rewriting and thus yields a marriage of basic and normalizing narrowing. The safe blocking rule is specific to basic narrowing and is particulary useful in conjunction with the rewriting rule. Finally, the unfolding rule allows for a variety of search strategies that reduce the number of don't know alternatives that need to be explored.

Keywords: Equation Solving, Universal Unification, Narrowing, Basic Narrowing, Normalizing Narrowing, Rewriting.

Acknowledgments: Nutt and Smolka are funded by the Bundesminister für Forschung und Technologie under grant ITR8501A.

Addresses of Authors: Werner Nutt, Gert Smolka, FB Informatik, Universität Kaiserslautern, 6750 Kaiserslautern, West Germany. Pierre Réty, CRIN, Campus Scientifique, BP 239, 54506 Vandœuvre les Nancy Cedex, France. UUCP: nutt@uklirb, rety@crin, smolka@uklirb.

1 Introduction

Narrowing first appeared in the context of resolution based theorem proving as an adaption of the paramodulation rule [Robinson/Wos 69] to canonical term rewriting systems [Slagle 74, Lankford 75]. Fày [78] realized that narrowing can be employed as a universal unification procedure that solves equations in the theory defined by a canonical rewriting system. Hullot [80] continued Fay's [78] work and devised a new narrowing strategy called basic narrowing. Kirchner [85] extended narrowing to rewriting modulo equations. Kaplan [84] and Hußmann [85] investigated narrowing for conditional term rewriting systems. The recent interest in logic programming with equations [Dershowitz/Plaisted 85, Goguen/Meseguer 86] has generated much work on universal unification (often called E-unification) [Gallier/Snyder 87, Hölldobler 87, Martelli et al. 86] and narrowing [Bosco et al. 87, Fribourg 85, Josephson/Dershowitz 86, Réty et al. 85, You/Subrahmanyam 86] in particular.

Technically, narrowing combines term unification and rewriting. To perform a narrowing step on a term t means to replace t by $\theta(t[\pi \leftarrow v])$, where t/π is a nonvariable subterm of t, $u \rightarrow v$ is a variable disjoint copy of a rule, and θ is the most general unifier of the subterm t/π and the left hand side u of the rule. The thus obtained narrowing relation extends the rewriting relation since every rewriting step is also a narrowing step.

Fay's [78] unification procedure employs a normalizing narrowing strategy, where a proper narrowing step is only performed if no rewriting step is possible. In other words, after every proper narrowing step the obtained term is rewritten to normal form. While the application of a rewriting step is don't care nondeterministic (that is, it doesn't matter which rewriting step is applied next), the application of a narrowing step is don't know nondeterministic (that is, it matters which narrowing step is applied next). The advantage of normalizing narrowing over pure narrowing is that it yields a unification procedure with a smaller search space.

Hullot's [80] basic narrowing strategy obtains a search space reduction by restricting narrowing steps to subterms that were not introduced by instantiation. The drawback of this stragtegy is that the application of a narrowing step that is actually a rewriting step is no longer don't care nondeterministic. Recently, the authors [Réty 87, Smolka/Nutt 87] devised special rewriting rules that are compatible with the basic narrowing strategy and whose application is still don't care nondeterministic. This present paper combines and simplifies our results.

We study basic narrowing and its optimizations as a method for solving equations in the initial algebra specified by a ground confluent and terminating term rewriting system. Since we are interested in equation solving, we don't study basic narrowing as a reduction relation on terms but consider immediately its reformulation as an equation solving rule. This reformulation leads to a technically simpler presentation and reveals that the essence of basic narrowing can be captured without recourse to term unification.

There are several advantages gained from weakening the usual confluence requirement to ground confluence. Applications in algebraic specification and logic programming usually employ initial algebra semantics, which means that ground confluence rather than full confluence is the natural requirement. A typical example is the specification of the integers shown in Figure 1. This specification is a terminating and ground confluent rewriting system, which is not confluent since, for instance, $x * y$ and $((x * y) + y) + (-y)$ are two distinct normal forms of $p(s(x)) * y$. An automatic completion of this system seems to be difficult if not impossible. Réty et al. [85] give a confluent extension of this system by adding thirteen

$$
\begin{array}{llll}
(1) & p(s(x)) \to x & & \\
(2) & s(p(x)) \to x & & \\
(3) & 0 + y \to y & & \\
(4) & s(x) + y \to s(x+y) & & \\
(5) & p(x) + y \to p(x+y) & & \\
(6) & -0 \to 0 & (9) & 0 * y \to 0 \\
(7) & -s(x) \to p(-x) & (10) & s(x) * y \to (x * y) + y \\
(8) & -p(x) \to s(-x) & (11) & p(x) * y \to (x * y) + (-y)
\end{array}
$$

Figure 1.1. A specification of the integers as a ground confluent and terminating rewriting system.

inductive consequences. This more than doubles the original rules and thus increases the search space of a narrowing based unification procedure. To be able to weaken the usual confluence requirement to ground confluence, completeness must be defined with respect to solutions, which map variables into irreducible ground terms, rather than unifiers, which map variables to terms possibly containing variables.

Our equation solving calculus employs three classes of rules: *resolution rules* whose application is don't know nondeterministic, *simplification rules* whose application is don't care nondeterministic, and *failure rules* allowing to prune inconsistent parts of a search tree. The resolution rules are the basic rules and suffice for a complete solution procedure. The purpose of the simplification rules is to reduce the search space. In some cases, the use of simplification rules can cut down an infinite search space to a finite one.

Three of the presented simplification rules are new. The rewriting rule allows for don't care nondeterministic rewriting and thus yields a marriage of basic and normalizing narrowing that enjoys the advantages of both approaches. The safe blocking rule is specific to basic narrowing and is particulary useful in conjunction with the rewriting rule. Finally, the unfolding rule allows for a variety of search strategies that reduce the number of don't know alternatives that need to be explored.

Our equation solving calculus is the basis for a class of solution procedures, where the don't know application of a resolution step is followed by the don't care application of finitely many simplification steps. The completeness of these procedures is shown with a new proof technique yielding a scheme that is easily applied to additional or alternative rules. As an application of our proof scheme, we show the completeness of an innermost constructor strategy similar to the one proposed by Fribourg [85].

The paper is organized as follows. In Section 2 we fix our notation for equations and rewriting systems. In Section 3 we present two resolution rules that yield a complete but very inefficient solution procedure. In Section 4, which is the heart of the paper, we extend the equation solving calculus with failure and simplification rules, thus obtaining a far more efficient solution procedure. In Section 5 we show the completeness of a solution procedure that uses inductive consequences for rewriting and prove the completeness of an innermost constructor strategy.

For most applications the use of many-sorted or even order-sorted (many-sorted with subsorts) equational logic is essential. Nevertheless, in this paper we consider only unsorted logic since it suffices to demonstrate our ideas. The generalization of our results to the many-sorted case without subsorts is straightforward. The generalization to the order-sorted case is also not difficult if sort-decreasing rewriting systems [Smolka et al. 87] are employed.

2 Equations and Rewriting Systems

In this section we review the necessary notations for equations and rewriting systems. The reader not familiar with the theory of term rewriting systems may consult [Huet 80, Huet/Oppen 80].

We assume that a set of **function symbols** (ranged over by f, g, and h) and an infinite set of **variables** (ranged over by x, y, z) are given. Every function symbol comes with an **arity**, which is a nonnegative integer.

Terms (ranged over by s, t, u, and v) and **occurrences** of terms (ranged over by π) are defined as usual. We use s/π to denote the **subterm of** s **at occurrence** π and $s[\pi \leftarrow t]$ to denote the term obtainable from s by replacing the subterm at occurrence π with t. An **equation** $s \doteq t$ is an ordered pair consisting of two terms s and t. The letter P will always range over equations. An **equation system** is a bag P_1 & $\cdots$ & P_n of equations; we use $\emptyset$ to denote the **empty equation system**. The letter E will always range over equation systems. An equation is called **trivial** if it has the form $s \doteq s$; an equation system is called **trivial** if each of its equations is trivial.

A **syntactical object** is either a term, an equation, or an equation system. A syntactical object is called **ground** if it does not contain variables. We use $\mathcal{V}(O)$ to denote the set of variables occurring in a syntactical object O.

A **signature** is a set of function symbols. The letter Σ will always range over signatures. A syntactical object is called a Σ-object if every function symbol occurring in it is in Σ.

Let Σ be a signature. A **Σ-substitution** is a function from Σ-terms into Σ-terms such that $\theta f(s_1, \ldots, s_n) = f(\theta s_1, \ldots, \theta s_n)$ and $\mathcal{D}\theta := \{x \mid \theta x \neq x\}$ is finite. In abuse of notation, $\mathcal{D}\theta$ is called the **domain of** θ and $\mathcal{C}\theta := \{\theta x \mid x \in \mathcal{D}\theta\}$ is called the **codomain of** θ. Furthermore, $\mathcal{I}\theta := \mathcal{V}(\mathcal{C}\theta)$ is called the set of **variables introduced by** θ. The letters θ, ψ, and ϕ will always range over substitutions. The composition of Σ-substitutions is again a Σ-substitution. Σ-substitutions are extended to syntactical Σ-objects as usual. A substitution θ is **ground** if θx is a ground term for all $x \in \mathcal{D}\theta$. A substitution θ is **idempotent** if $\theta\theta = \theta$. Note that θ is idempotent if and only if $\mathcal{D}\theta$ and $\mathcal{I}\theta$ are disjoint.

The **equational representation** $[\theta]$ of a substitution θ is the equation system

$$x_1 \doteq \theta x_1 \;\&\; \cdots \;\&\; x_n \doteq \theta x_n$$

where $\{x_1, \ldots, x_n\} = \mathcal{D}\theta$. Two substitutions are equal if and only if their equational representations are equal. Conversely, every Σ-equation system $x_1 \doteq s_1 \;\&\cdots\&\; x_n \doteq s_n$ such that $x_1, \ldots, x_n$ are distinct variables is the equational representation of some Σ-substitution, which we denote with $\langle x_1 \doteq s_1 \;\&\; \cdots \;\&\; x_n \doteq s_n\rangle$. Note that $\theta = \langle[\theta]\rangle$ for every substitution θ.

Let θ be a substitution and V be a set of variables. The **restriction** $\theta|_V$ of θ to V is defined by: $\theta|_V(x) := \theta x$ if $x \in V$, otherwise $\theta|_V(x) := x$. Furthermore, the **update** $\theta[y \leftarrow s]$ of θ at y with s is defined by: $\theta[y \leftarrow s](x) := s$ if $x = y$, otherwise $\theta[y \leftarrow s](x) := \theta x$.

A syntactical object O is called an **instance** of a syntactical object O' if there is a substitution θ such that $O = \theta O'$. A syntactical object O is called a **variant** of a syntactical object O' if O is obtainable from O' by consistent variable renaming, that is, there exist substitutions θ and ψ such that $O' = \theta O$ and $O = \psi O'$.

Let $\rightarrow$ be a binary relation on a set M. Then we use $\rightarrow^*$ to denote the reflexive and transitive closure of $\rightarrow$. The relation $\rightarrow$ is called **confluent** if for all a, b, and c in M such that $a \rightarrow^* b$ and $a \rightarrow^* c$ there exists a d in M such that $b \rightarrow^* d$ and $c \rightarrow^* d$. Furthermore, $\rightarrow$ is called **terminating** if there is no infinite chain $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots$.

A **Σ-rewriting rule** $s \rightarrow t$ is an equation $s \doteq t$ such that s isn't a variable and every variable occurring in the right hand side t occurs in the left hand side s. A **rewriting system** $\mathcal{R} = (\Sigma, \mathcal{E})$ consists of a signature Σ and a set $\mathcal{E}$ of Σ-rewriting rules. A rewriting system $\mathcal{R} = (\Sigma, \mathcal{E})$ defines a binary relation $\xrightarrow{\mathcal{R}}$ called the **rewriting relation** of $\mathcal{R}$ on the set of all Σ-terms as follows: $s \xrightarrow{\mathcal{R}} t$ if and only if there exists an occurrence π of s and an instance $u \rightarrow v$ of a rule of $\mathcal{R}$ such that $s/\pi = u$ and $t = s[\pi \leftarrow v]$. A term s is **$\mathcal{R}$-normal** if there is no term t such that $s \xrightarrow{\mathcal{R}} t$. A term t is an **$\mathcal{R}$-normal form** of a term s if $s \xrightarrow{\mathcal{R}}{}^* t$ and t is $\mathcal{R}$-normal. An **$\mathcal{R}$-value** is an $\mathcal{R}$-normal ground term. A rewriting system $\mathcal{R} = (\Sigma, \mathcal{E})$ is **ground confluent** if the restriction of $\xrightarrow{\mathcal{R}}$ to the set of all ground Σ-terms is confluent.

The **initial algebra** $\mathcal{I}(\mathcal{R})$ specified by a ground confluent and terminating rewriting system $\mathcal{R} = (\Sigma, \mathcal{E})$ can be defined as follows:

- The carrier of $\mathcal{I}(\mathcal{R})$ is the set of all $\mathcal{R}$-values.
- The denotation $f_{\mathcal{I}(\mathcal{R})}$ of a function symbol in Σ is given by $f_{\mathcal{I}(\mathcal{R})}(s_1, \ldots, s_n) = s$, where s is the $\mathcal{R}$-normal form of $f(s_1, \ldots, s_n)$.

A ground Σ-equation $s \doteq t$ is **valid** in (the initial algebra of) $\mathcal{R}$ if s and t have the same $\mathcal{R}$-normal form. We write $\mathcal{R} \models s \doteq t$ or $s =_{\mathcal{R}} t$ if $s \doteq t$ is valid in $\mathcal{R}$. A ground Σ-equation system is **valid** in (the initial algebra of) $\mathcal{R}$ if each of its equations is valid in $\mathcal{R}$. We write $\mathcal{R} \models E$ if E is valid in $\mathcal{R}$. A Σ-equation $s \doteq t$ is an **inductive consequence** of $\mathcal{R}$ if every ground instance of $s \doteq t$ is valid in $\mathcal{R}$. Two ground Σ-substitutions θ and ψ are **equal in** $\mathcal{R}$ (write $\theta =_{\mathcal{R}} \psi$) if $\mathcal{D}\theta = \mathcal{D}\psi$ and $\theta x =_{\mathcal{R}} \psi x$ for every $x \in \mathcal{D}\theta$.

Let $\mathcal{R} = (\Sigma, \mathcal{E})$ be a ground confluent and terminating rewriting system. Then we have:

- "$s =_{\mathcal{R}} t$" is a congruence on the set of all ground Σ-terms, that is, "$s =_{\mathcal{R}} t$" is an equivalence relation satisfying
$$s_1 =_{\mathcal{R}} t_1 \ \wedge \ \cdots \ \wedge \ s_n =_{\mathcal{R}} t_n \quad \Rightarrow \quad f(s_1, \ldots, s_n) =_{\mathcal{R}} f(t_1, \ldots, t_n).$$
- "$\theta =_{\mathcal{R}} \psi$" is an equivalence relation on the set of all ground Σ-substitutions.
- If $\theta =_{\mathcal{R}} \psi$, then $\theta s =_{\mathcal{R}} \psi s$ for every term s such that $\mathcal{V}(s) \subseteq \mathcal{D}\theta = \mathcal{D}\psi$.

3 The Basic Resolution Rules

In this section we develop a simple equation solving calculus that captures the essence of Hullot's [80] basic narrowing method. This calculus is the basis for a simple solution procedure whose soundness and completeness we will prove. In the next section we will present several extensions for this calculus, thus obtaining a refined solution procedure with a much smaller search space. In particular, the basic calculus to be presented in this section does not yet incorporate term unification, which will only be added in the next section.

General Assumption. *In the rest of this paper we assume that $\mathcal{R} = (\Sigma, \mathcal{E})$ is a ground confluent and terminating rewriting system; furthermore, we assume that there is at least one ground Σ-term.*

We start by defining the solutions of an equation system in the initial algebra of $\mathcal{R}$. A substitution θ is an **$\mathcal{R}$-assignment** if θx is an $\mathcal{R}$-value for all $x \in \mathcal{D}\theta$. We use $\mathrm{ASS}_{\mathcal{R}}$ to denote the set of all $\mathcal{R}$-assignments. With that we define the set of all $\mathcal{R}$-solutions of an equation system E as

$$\mathrm{SOL}_{\mathcal{R}}(E) := \{\theta \in \mathrm{ASS}_{\mathcal{R}} \mid \mathcal{D}\theta = \mathcal{V}(E) \;\wedge\; \mathcal{R} \models \theta E\}.$$

An equation solving procedure for $\mathcal{R}$ is a procedure that enumerates $\mathrm{SOL}_{\mathcal{R}}(E)$.

For technical reasons that will become apparent soon, we need to relativize the solutions of an equation system with respect to a set of "primary variables". The **$\mathcal{R}$-solutions** of a Σ-equation system E with respect to a set V of variables are defined as follows:

$$\mathrm{SOL}_{\mathcal{R}}^{V}(E) := \{\theta|_V \mid \theta \in \mathrm{ASS}_{\mathcal{R}} \;\wedge\; \mathcal{D}\theta = V \cup \mathcal{V}(E) \;\wedge\; \mathcal{R} \models \theta E\}.$$

Note that $\mathrm{SOL}_{\mathcal{R}}(E) = \mathrm{SOL}_{\mathcal{R}}^{\mathcal{V}(E)}(E)$. For convenience, we write $\mathrm{SOL}_{\Sigma}^{V}(E)$ for $\mathrm{SOL}_{(\Sigma,\emptyset)}^{V}(E)$, where $(\Sigma, \emptyset)$ is the rewriting system with signature Σ and no rules. Note that $\mathrm{SOL}_{\Sigma}^{V}(E)$ can be represented rather explicitly by the most general unifier of E, which can be computed using term unification. This will be discussed in Subsection 4.2.

In the literature, narrowing is usually presented for confluent rewriting systems and completeness is shown with respect to all unifiers, which include nonground substitutions. Since we have weakened the confluence requirement to ground confluence, we have to restrict our attention to ground substitutions. Nevertheless, the ground confluence approach subsumes the conventional approach. To see this, assume a confluent rewriting system is given. We can extend this system by adding infinitely many constants to its signature, one for each variable. Then the solutions with respect to the extended system, which is still ground confluent, exactly correspond to the unifiers with respect to the original system.

The rules of our equation solving calculus, which are given in Figure 3.1, apply to pairs $C.\, E$ consisting of two equation systems C and E; C is called the **constraint part** and E is called the **unsolved part**. The division of $C\&E$ into two parts is needed to express the basic narrowing strategy. The calculus will allow us to reduce an **initial pair** $\emptyset.\, E$ to **solved pairs** $C_1.\, \emptyset$, $C_2.\, \emptyset$, $\ldots$ such that

- *(Soundness)* $\;\forall\, i.\;\; \mathrm{SOL}_{\Sigma}^{V}(C_i) \subseteq \mathrm{SOL}_{\mathcal{R}}^{V}(C_i) \subseteq \mathrm{SOL}_{\mathcal{R}}^{V}(E)$
- *(Completeness)* $\;\forall\, \theta \in \mathrm{SOL}_{\mathcal{R}}^{V}(\emptyset.\, E)\;\; \exists\, i.\;\; \theta \in \mathrm{SOL}_{\Sigma}^{V}(C_i)$.

Thus, our calculus "solves" by reducing $\mathcal{R}$-solutions to Σ-solutions. The two rules given in Figure 3.1 are called *resolution rules* because they are the primary rules for solving equation

Blocking

$$(B) \quad C\,.\,P \,\&\, E \quad \xrightarrow{r}_{\mathcal{R},V} \quad C \,\&\, P\,.\,E$$

Application

$$(A) \quad C\,.\,P \,\&\, E \quad \xrightarrow{r}_{\mathcal{R},V} \quad C \,\&\, (P/\pi \doteq u)\,.\,P[\pi \leftarrow v] \,\&\, E$$

if P/π isn't a variable and $u \rightarrow v$ is a variant of a rule of $\mathcal{R}$ having no variables in common with $C\,.\,P \,\&\, E$ or V

Figure 3.1. The basic resolution rules.

systems and because we want to distinguish them from the failure and simplification rules to be presented in the next section. With Robinson's [65] resolution rule our resolution rules have only in common that they resolve something—in our case equations.

The application rule in Figure 3.1 has to introduce new variables to obtain a renamed variant of the employed rewriting rule. The following assumption makes sure that there are always enough new variables left.

General Assumption. *In the rest of this paper we assume that V is a finite set of variables.*

Example. Let $\mathcal{R}$ be the system in Figure 1.1, $V = \{y\}$, and consider the equation $0 + y \doteq 0$, which has the unique solution $\langle y \doteq 0\rangle$. Then:

$$\begin{array}{lll} & \emptyset\,.\,0+y \doteq 0 & \\ \xrightarrow{r}_{\mathcal{R},V} & 0+y \doteq 0+y'\,.\,y' \doteq 0 & \text{by a resolution step using rule (3)} \\ \xrightarrow{r}_{\mathcal{R},V} & 0+y \doteq 0+y' \,\&\, y' \doteq 0\,.\,\emptyset & \text{by a blocking step.} \end{array}$$

Theorem 3.1. **(Soundness)** *If $C\,.\,E \xrightarrow{r}_{\mathcal{R},V} C'\,.\,E'$ by a blocking or an application step, then* $\mathrm{SOL}^V_{\mathcal{R}}(C' \,\&\, E') \subseteq \mathrm{SOL}^V_{\mathcal{R}}(C \,\&\, E)$.

Proof. Let $C\,.\,E \xrightarrow{r}_{\mathcal{R},V} C'\,.\,E'$ and let θ be an assignment such that $V \cup \mathcal{V}(C'\,.\,E') = \mathcal{D}\theta$ and $\theta(C' \,\&\, E')$ is valid in $\mathcal{R}$. It suffices to show that $\theta(C \,\&\, E)$ is valid in $\mathcal{R}$.

If $C' \,\&\, E'$ has been obtained from $C\,.\,E$ by a blocking step, then the claim is trivial. If an application step has been performed, then $C\,.\,E = (C\,.\,P \,\&\, E_1)$, $C'\,.\,E' = (C \,\&\, P/\pi \doteq u\,.\, P[\pi \leftarrow v] \,\&\, E_1)$, and $u \rightarrow v$ is a rule of $\mathcal{R}$. It suffices to show that θP is valid in $\mathcal{R}$. Since $\theta(P/\pi) =_{\mathcal{R}} \theta u$ and $u \rightarrow v$ is a rule of $\mathcal{R}$, we have $\theta(P/\pi) =_{\mathcal{R}} \theta v$. Since $\theta(P[\pi \leftarrow v]) = (\theta P)[\pi \leftarrow \theta v]$ is valid in $\mathcal{R}$, we know that $(\theta P)[\pi \leftarrow \theta(P/\pi)] = \theta P$ is valid in $\mathcal{R}$. □

The nondeterministic solution procedure in Figure 3.2 is an operational formulation of the equation solving calculus in Figure 3.1. The procedure can be explained as a two person game played by a don't care player who makes the don't care choices and a don't know player who makes the don't know choices. Given $\mathcal{R}$, a pair $\emptyset\,.\,E$, $V := \mathcal{V}(E)$, and solve($C.\ E$) is

1. if E is empty, then return C;
2. choose **don't care** an equation P in E;
3. choose **don't know** $C'.\ E'$ such that $C.\ E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by a step on P;
4. solve($C'.\ E'$)

Figure 3.2. The basic solution procedure.

a solution $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$, the don't care player wins if the procedure terminates with an equation system C such that $\theta \notin \mathrm{SOL}_{\Sigma}^{V}(C)$; the don't know player wins if the procedure terminates with an equation system C such that $\theta \in \mathrm{SOL}_{\Sigma}^{V}(C)$. We say that the procedure is complete if the don't know player can always win if he makes the right choices. In the following we will show the completeness of the procedure.

An implementation of the basic solution procedure has to explore all alternatives of a don't know choice. In fact, the procedure generates a huge number of don't know alternatives in step 3. One alternative is to block P; the other alternatives are obtained by applying a rule to P, where every nonvariable occurrence of P and every rule of $\mathcal{R}$ have to be considered. To be efficient, it is crucial to eliminate redundant or inconsistent don't know alternatives as early as possible. This will be the theme of the next section.

The application rule needs to introduce new variables to obtain a renamed variant of a rewriting rule. The choice of the new variables is obviously a don't care nondeterminism, but making this fact explicit is technically very tedious. For this reason the choice of new variables appears as a don't know nondeterminism in the procedure in Figure 3.2. This problem will be solved in the next section by the introduction of a simplification rule that can be used to rename variables not occurring in V.

The basic idea behind the completeness proof is a lifting argument. If $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$, then this fact can be verified by rewriting θE into a trivial equation system. Now the idea is that a blocking step corresponds to the deletion of a trivial equation in θE and an application step corresponds to an innermost rewriting step on θE.

We start by setting up a calculus for verifying that a ground equation system is valid in $\mathcal{R}$. The two rules of the verification calculus correspond to the blocking and the application rule of the equation solving calculus:

- *(VB)* $P\ \&\ E \xrightarrow{vr}_{\mathcal{R}} E$ if P is a trivial equation
- *(VA)* $P\ \&\ E \xrightarrow{vr}_{\mathcal{R}} P[\pi \leftarrow v]\ \&\ E$ if $P/\pi \rightarrow v$ is an instance of a rule of $\mathcal{R}$.

The rule *(VB)* deletes a trivial equation and the rule *(VA)* applies a rewriting step.

Proposition 3.2.

- *(Invariance) If $E \xrightarrow{vr}_{\mathcal{R}} E'$, then E is valid in $\mathcal{R}$ if and only if E' is valid in $\mathcal{R}$.*

- *(Termination) The relation "$E \xrightarrow{vr}_{\mathcal{R}} E'$" is terminating.*

- *(Completeness) E is valid in $\mathcal{R}$ if and only if $E \xrightarrow{vr}{}^{*}_{\mathcal{R}} \emptyset$.*

An **$\mathcal{R}$-triple** θ. C. E consists of an $\mathcal{R}$-assignment θ and two equation systems C and E such that $\mathcal{V}(C.\ E) \subseteq \mathcal{D}\theta$, θC is trivial, and θE is valid in $\mathcal{R}$. The assignment θ should be thought of as the solution one wants to find by applying resolution steps to the pair $C.\ E$.

Proposition 3.3. *If $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$, then $\theta.\ \emptyset.\ E$ is an $\mathcal{R}$-triple. Furthermore, if $\theta.\ C.\ \emptyset$ is an $\mathcal{R}$-triple and $V \subseteq \mathcal{V}(C)$, then $\theta|_V \in \mathrm{SOL}^V_\Sigma(C)$.*

We now define a reduction relation on $\mathcal{R}$-triples that links resolution steps with their corresponding verification steps. We write $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ if $\theta.\ C.\ E$ and $\theta'.\ C'.\ E'$ are both $\mathcal{R}$-triples and

- θ and θ' agree on V
- $C.\ E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by a resolution rule σ
- $\theta E \xrightarrow{vr}_{\mathcal{R}} \theta' E'$ by the verification rule corresponding to σ.

Proposition 3.4. (Termination) *The triple relation "$\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$" is terminating.*

A term is called **$\mathcal{R}$-innermost** if each of its proper subterms is $\mathcal{R}$-normal. The proof of the following theorem rests on the idea that for a triple $\theta.\ C.\ E$ a verification step that rewrites an innermost term of θE can be "pushed up" to an application step on $C.\ E$.

Theorem 3.5. (Push Up) *If $\theta.\ C.\ E$ is an $\mathcal{R}$-triple and P is an equation in E, then there exists a triple $\theta'.\ C'.\ E'$ such that $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a resolution step on P.*

Proof. Let $\theta.\ C.\ P\&E$ be an $\mathcal{R}$-triple. Then θP is valid in $\mathcal{R}$. Thus θP is either trivial or can be rewritten.

1. Suppose θP is a trivial equation. Then $C.\ P\ \&\ E \xrightarrow{r}_{\mathcal{R},V} C\ \&\ P.\ E$ by the blocking rule and $\theta(P\ \&\ E) \xrightarrow{vr}_{\mathcal{R}} \theta E$ by the verification rule *VB*. Since $\theta.\ C\&P.\ E$ is an $\mathcal{R}$-triple, this yields the claim.

2. Suppose θP can be rewritten. Then there exist a nonvariable occurrence π of P such that $(\theta P)/\pi$ is $\mathcal{R}$-innermost, a variant $u \to v$ of a rule of $\mathcal{R}$, and a substitution ϕ such that $\phi u = (\theta P)/\pi$. Without loss of generality we can assume that $\mathcal{D}\phi = \mathcal{V}(u \to v)$ and $u \to v$ has no variables in common with V, $C.\ P\&E$, and $\mathcal{D}\theta$. Define $C' := (C\ \&\ P/\pi \doteq u)$ and $E' := (P[\pi \leftarrow v]\ \&\ E)$. Since $\theta P\ \&\ \theta E_1 \xrightarrow{vr}_{\mathcal{R}} (\theta P)[\pi \leftarrow \phi v]\ \&\ \theta E_1$ by the verification rule *VA* and $C.\ P\&E \xrightarrow{r}_{\mathcal{R},V} C'.\ E'$ by the resolution rule A, it suffices to show that there exists an $\mathcal{R}$-assignment θ' such that $\mathcal{D}\theta' = \mathcal{D}\theta \cup \mathcal{V}(u \to v)$, θ' agrees with θ on $\mathcal{D}\theta$, $\theta'(P/\pi) = \theta' u$, and $\theta'(P[\pi \leftarrow v])$ is valid in $\mathcal{R}$.

To show this, define θ' as follows: if $x \in \mathcal{D}\phi$ then $\theta' x := \phi x$, otherwise $\theta' x := \theta x$. To show that θ' is an $\mathcal{R}$-assignment, it suffices to show that ϕ is an $\mathcal{R}$-assignment, which holds since $\mathcal{D}\phi = \mathcal{V}(u \to v) = \mathcal{V}(u)$, $\phi u = \theta(P/\pi)$ is $\mathcal{R}$-innermost and ground, and u isn't a variable.

Since $\mathcal{D}\phi = \mathcal{V}(u \to v)$, we have $\mathcal{D}\theta' = \mathcal{D}\theta \cup \mathcal{D}\phi = \mathcal{D}\theta \cup \mathcal{V}(u \to v)$ as required. Since $\mathcal{D}\theta$ and $\mathcal{D}\phi = \mathcal{V}(u \to v)$ are disjoint, θ' and θ agree on $\mathcal{D}\theta$. Furthermore, $\theta'(P/\pi) = \theta' u$ since $\theta(P/\pi) = \phi u$.

Finally, $\theta'(P[\pi \leftarrow v]) = (\theta P)[\pi \leftarrow \phi v]$ is valid in $\mathcal{R}$ since $\phi v =_{\mathcal{R}} \phi u$, $\phi u = \theta(P/\pi)$, and $\theta P = (\theta P)[\pi \leftarrow \theta(P/\pi)]$ is valid in $\mathcal{R}$. □

Corollary 3.6. *For every $\mathcal{R}$-triple $\theta .\, C .\, E$ there exist θ' and C' such that $\theta .\, C .\, E \xrightarrow{r}{}^{*}_{\mathcal{R},V} \theta' .\, C' .\, \emptyset$.*

Proof. Suppose that $\theta .\, C .\, E$ is an $\mathcal{R}$-triple. If E is empty, then the claim is trivial. Otherwise, the push up theorem applies and yields $\theta .\, C .\, E \xrightarrow{r}_{\mathcal{R},V} \theta' .\, C' .\, E'$ for some triple $\theta' .\, C' .\, E'$. Thus, using the termination property of the triple reduction relation, the claim follows by induction. □

Corollary 3.7. **(Completeness)** *Let $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$. Then there exists an equation system C such that $\emptyset .\, E \xrightarrow{r}{}^{*}_{\mathcal{R},\mathcal{V}(E)} C .\, \emptyset$ and $\theta \in \mathrm{SOL}_{\Sigma}^{\mathcal{V}(E)}(C)$.*

Proof. Let $\theta \in \mathrm{SOL}_{\mathcal{R}}(E)$. Then $\theta .\, \emptyset .\, E$ is an $\mathcal{R}$-triple. By the preceding corollary we know that there exist θ' and C' such that $\theta .\, \emptyset .\, E \xrightarrow{r}{}^{*}_{\mathcal{R},\mathcal{V}(E)} \theta' .\, C' .\, \emptyset$. Thus, we know that $\theta = \theta'|_{\mathcal{V}(E)} \in \mathrm{SOL}_{\Sigma}^{\mathcal{V}(E)}(C)$. □

4 Failure and Simplification Rules

In this section we present several optimizations for the basic solution procedure that was discussed in the last section. An implementation of this procedure must explore all alternatives of a don't know choice in step 3, which generates a huge search space. To reduce this search space, it is crucial to detect as early as possible whether a pair $C .\, E$ is consistent, that is, whether there is an assignment that extends it to an $\mathcal{R}$-triple. This is accomplished by so-called failure rules, which are decidable sufficient criteria for the inconsistency of a pair. The second method for cutting down the search space is the addition of so-called simplification rules whose application, in contrast to the application of resolution rules, is don't care nondeterministic. By simplifying a pair with the simplification rules before the application of a resolution step it is often possible to reduce the number of don't know resolution steps needed to reach a solved pair. Furthermore, often a failure rule applies to an inconsistent pair only after it has been simplified. Figure 4.1 shows the extension of the basic solution procedure to failure and simplification rules.

4.1 The Failure Rules

The following definitions are needed to formulate the failure rules.

An equation system E is Σ-**consistent** if there is a substitution θ such that θE is trivial. The Σ-consistency of an equation system can be decided by a term unification algorithm.

A pair $C .\, E$ is **consistent** in $\mathcal{R}$ if there exists a substitution θ such that $\theta .\, C .\, E$ is an $\mathcal{R}$-triple.

A function symbol is called **generating** in $\mathcal{R}$ if it occurs in at least one $\mathcal{R}$-value. A function symbol is called **completely defined** in $\mathcal{R}$ if it is not generating in $\mathcal{R}$. In the rewriting system in Figure 1.1 the functions 0, s and p are generating and the functions $+$, $-$ and $*$ are completely defined.

Two function symbols f and g are **disjoint** in $\mathcal{R}$ if no ground equation of the form $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)$ is valid in $\mathcal{R}$.

A function symbol f is called **reducible** in $\mathcal{R}$ if there is a rule ρ in $\mathcal{R}$ such that f is the top symbol of the left hand side of ρ. A function symbol is called **irreducible** in $\mathcal{R}$ if it isn't reducible in $\mathcal{R}$. The constant 0 is the only irreducible function symbol in the rewriting system in Figure 1.1.

Proposition 4.1. *If a function symbol is irreducible in $\mathcal{R}$, then it is generating in $\mathcal{R}$. Furthermore, if f and g are distinct function symbols that are both irreducible in $\mathcal{R}$, then f and g are disjoint in $\mathcal{R}$.*

Proposition 4.2. (Failure Rules) *A pair $C \bullet E$ is inconsistent in $\mathcal{R}$ if one of the following conditions holds:*

1. *C is not Σ-consistent.*
2. *C contains an equation $x \doteq t$ such that t is not $\mathcal{R}$-normal.*
3. *C contains an equation $x \doteq t$ such that t contains a completely defined function symbol.*
4. *$C = [\psi]$ for some substitution ψ and ψE contains an equation $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)$ such that f and g are disjoint.*

The requirement that the constraint part of a pair is the equational representation of a substitution is not a real restriction since we will introduce a simplification rule that replaces the constraint part by its most general unifier.

The concept of a completely defined function symbol is of little use for unsorted rewriting systems. For instance, if we add to the system in Figure 1.1 the constants *true* and *false*, the functions $+$, $-$ and $*$ are no longer completely defined. This problem can be avoided by working with many-sorted rewriting systems. Since the power of the failure rule (3) increases with the number of completely defined functions, the presence of sorts, even without subsorts, can lead to smaller search spaces.

4.2 Term Unification and Solved Equation Systems

Term unification will be an important part of our optimized solution procedure. After every resolution step the computation of the most general unifier of the constraint part of the obtained pair is attempted. If this attempt fails, we know by the failure rule (1) that the obtained pair is inconsistent. Otherwise, the constraint part can be replaced with the equational representation of its most general unifier, an optimization that will be expressed by a simplification rule. If no other failure and simplication rules are employed, the thereby obtained solution procedure performs essentially basic narrowing as described in [Hullot 80].

In this subsection we review the necessary notations and results for term unification.

An equation system S is called **solved** if it has the form $x_1 \doteq s_1 \;\&\; \ldots \;\&\; x_n \doteq s_n$ where the variables $x_1, \ldots, x_n$ occur only once. Note that an equation system is solved if and only if it is the equational representation of an idempotent substitution. The letter S will always range over solved systems.

solve($C \centerdot E$) is

1. choose **don't care** $C' \centerdot E'$ such that $C \centerdot E \xrightarrow{s}{}^{*}_{\mathcal{R},V} C' \centerdot E'$ by simplification steps;
2. if a failure rule applies to $C' \centerdot E'$, then fail;
3. if E' is empty, then return C';
4. choose **don't care** an equation P in E';
5. choose **don't know** $C'' \centerdot E''$ such that $C' \centerdot E' \xrightarrow{r}_{\mathcal{R},V} C'' \centerdot E''$ by a resolution step on P;
6. solve($C'' \centerdot E''$)

Figure 4.1. The extended solution procedure.

The next theorem is the adaption of Robinson's [65] unification theorem to our framework.

Theorem 4.3. *A Σ-equation system E is Σ-consistent if and only if there exists a solved Σ-equation system S such that $\mathcal{D}\langle S\rangle \subseteq V$ and $\mathrm{SOL}_\Sigma^V(E) = \mathrm{SOL}_\Sigma^V(S)$.*

For an example, consider $\mathrm{SOL}_\Sigma^{\{x\}}(x + s(0) \doteq s(0) + y) = \mathrm{SOL}_\Sigma^{\{x\}}(x \doteq s(0))$. The next proposition says that the solved system S is a fairly explicit representation of the solution set $\mathrm{SOL}_\Sigma^V(S)$.

Proposition 4.4. *If $\mathcal{D}\langle S\rangle \subseteq V$, then $\mathrm{SOL}_\Sigma^V(S) = \{(\theta\langle S\rangle)|_V \mid \forall x \in V.\ \ \theta\langle S\rangle x$ is ground$\}$.*

4.3 The Simplification Rules

Figure 4.2 and 4.3 show the simplification rules we will discuss in this paper. Three of these rules—the rewriting rule, the unfolding rule and the safe blocking rule *SB1*—did not appear in the literature so far. In conjunction with the don't care selection of the equation to be resolved upon next, the unfolding rule can drastically reduce the don't know alternatives our solution procedure has to explore. The rewriting rule, if used together with the unfolding rule and the safe blocking rule *SB1*, results in a marriage of basic and normalizing narrowing that enjoys the advantages of both approaches.

The key property of the simplification rules is that their application preserves the reachable solutions, that is, if $C \centerdot E \xrightarrow{s}_{\mathcal{R},V} C' \centerdot E'$ by a simplification step, then every solution that can be reached from $C \centerdot E$ can also be reached from $C' \centerdot E'$. We postpone the proof of this claim to the next subsection. As a consequence of this preservation property, a pair $C \centerdot E$ is inconsistent if it is inconsistent after it has been simplified. This fact greatly enhances the power of the failure rules.

The following definition is needed for the decomposition rule. A function symbol f is **decomposable** in $\mathcal{R}$ if for every ground equation $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ that is valid in $\mathcal{R}$ the equations $s_1 \doteq t_1$, …, $s_n \doteq t_n$ are valid in $\mathcal{R}$. In the rewriting system in Figure 1.1 the function symbols s and p are decomposable.

Subsumption

$(S)\quad S\,.\,P \;\&\; Q \;\&\; E \;\xrightarrow{s}_{\mathcal{R},V}\; S\,.\,Q \;\&\; E$

if $\langle S\rangle P = \langle S\rangle Q$

Permutation

$(P1)\quad C\,.\,s \doteq t \;\&\; E \;\xrightarrow{s}_{\mathcal{R},V}\; C\,.\,t \doteq s \;\&\; E$

$(P2)\quad S\,.\,x \doteq s \;\&\; t \doteq u \;\&\; E \;\xrightarrow{s}_{\mathcal{R},V}\; S\,.\,x \doteq s \;\&\; x \doteq u \;\&\; E$

if $\langle S\rangle s = \langle S\rangle t$

$(P3)\quad C\,.\,E \;\xrightarrow{s}_{\mathcal{R},V}\; C'\,.\,E'$

if $C'\,.\,E'$ is obtainable from $C\,.\,E$ by replacing all occurrences of x with y, where $x \notin V$ and $y \notin V \cup \mathcal{V}(C\,.\,E)$

Figure 4.3. The simplification rules, part 2.

list) and '.' (the cons operator) are irreducible and thus generating, decomposable and disjoint. The function symbol *app* (list concatenation) is completely defined.

Example 4.6. (Rewriting) We want to solve the equation $app(app(x,y),\ z) \doteq nil$ in $\mathcal{R}1$ with respect to the variable z. This problem has an infinite search space if only unification is employed for simplification, but it has a finite search space if both the unification and rewriting rule can be used. To see this, consider the derivation

$$
\begin{array}{lll}
 & \emptyset\,.\,app(app(x,y),\ z) \doteq nil & \\
\xrightarrow{r}_{\mathcal{R}1,\{z\}} & app(x,y) \doteq app(x'.y',z')\,.\,app(x'.app(y',z'),\ z) \doteq nil & \text{by } A \\
\xrightarrow{s}_{\mathcal{R}1,\{z\}} & \emptyset\,.\,app(x'.app(y',z'),\ z) \doteq nil & \text{by } Uni,
\end{array}
$$

which can be continued infinitely often by applying rule (2) to the inner occurrence of *app*. However, if the rewriting rule is available for simplification, we can prune this infinite and inconsistent part of the search space by rewriting the above pair to

$$
\xrightarrow{s}_{\mathcal{R}1,\{z\}} \quad \emptyset\,.\,x'.app(app(y',z'),\ z) \doteq nil \qquad \text{by } R.
$$

This pair can now be recognized as inconsistent by the failure rule (4) since the function symbols '.' and *nil* are disjoint in $\mathcal{R}1$.

The following derivation shows how the solution of the system can be computed:

first. This idea can be exploited by unfolding the right inner occurrence of *ap*
the equations

$$z' \doteq app(x', y') \ \& \ app(app(x, y), z') \doteq z$$

and thus eliminates the alternatives (2) and (3) if the left equation is consid

In conjunction with the don't care selection of the next equation to be the unfolding rule can be used to obtain a variety of strategies that reduce t alternatives a solution procedure has to consider. Two examples are the t basic narrowing strategy in [Herold 86] and the selection narrowing strategy i 87]. Another example is the innermost constructor strategy in [Fribourg 85], discuss in the next section.

Bosco et al. [87] present a translation of basic narrowing into SLD-resolut which gives them implicitly the effect we would obtain by using the unfoldin as possible. Complete unfolding, however, has the disadvantage of reducin the rewriting rule. Nevertheless, Bosco et al.'s [87] paper gave us the idea fo rule.

The application conditions of the unfolding rule ensure that it can't pro of the form $x \doteq y$, a restriction that is needed to preserve the completeness o solution procedure.

Example 4.8. (Safe Blocking) As we have seen in Example 3.1, using the for simplification may cut down an infinite search space to a finite one. A d the rewriting rule is, however, that it transfers terms from the constraint par unsolved part, thus increasing the search space again. To see this, let $\mathcal{R}$ be system in Figure 1.1 and consider the rewriting step

$$\begin{array}{lll} & y \doteq s(s(s(z))) \,.\, x + p(y) \doteq 0 & \\ \xrightarrow{s}_{\mathcal{R},\{y\}} & y \doteq s(s(s(z))) \,.\, x + s(s(z)) \doteq 0 & \text{by } R \end{array}$$

which carries the term $s(s(z))$ from the constraint part into the unsolved p advantage can be completely avoided by using the unfolding and the safe bl transfer terms carried over by the rewriting rule back into the constraint par

$$\begin{array}{lll} \xrightarrow{s}_{\mathcal{R},\{y\}} & y \doteq s(s(s(z))) \,.\, x' \doteq s(s(z)) \ \& \ x + x' \doteq 0 & \text{by } U \\ \xrightarrow{s}_{\mathcal{R},\{y\}} & y \doteq s(s(s(z))) \ \& \ x' \doteq s(s(z)) \,.\, x + x' \doteq 0 & \text{by } S \end{array}$$

Example 4.9. (Naive Rewriting) The following restriction of the applicati we will refer to as the *naive rewriting rule*, seems to be a better alternative tc rule in Figure 4.2 since it doesn't transfer terms from the constraint part tc part:

$$S.\, P \ \& \ E \quad \longrightarrow_{\mathcal{R},V} \quad S \ \& \ (P/\pi \doteq u).\, P[\pi \leftarrow v] \ \& \ E$$

if P/π isn't a variable,
$u \rightarrow v$ is a variant of a rule of $\mathcal{R}$ containing only new variables,
and $\langle S\rangle(P/\pi)$ is an instance of u.

However, this rule cannot be used as a simplification rule since, in general, i is not don't care nondeterministic. To see this, consider the rewriting system and the initial pair

$$\emptyset \,.\, s(p(x + 0)) \doteq 0,$$

which has the unique solution $\langle x \doteq 0\rangle$. By applying the naive rewriting rule to s with rule (2) we obtain the pair

$$s(p(x+0)) \doteq s(p(x')) \;.\; x' \doteq 0,$$

which, after a unification step, becomes

$$x' \doteq x + 0 \;.\; x' \doteq 0.$$

The only resolution step that applies to the unsolved equation of this pair is blocking, which yields

$$x' \doteq x + 0 \;\&\; x' \doteq 0 \;.\; \emptyset,$$

a pair whose constraint part is Σ-inconsistent. This shows that the application of the naive rewriting rule is not don't care nondeterministic.

Example 4.10. (Decomposition) Let $\mathcal{R}$ be the rewriting system in Figure 1.1 and consider the equation $s(x) \doteq s(y)$. Since s is decomposable in $\mathcal{R}$ (note that s is not irreducible in $\mathcal{R}$), we know by the decomposition rule that the equation $x \doteq y$, which is in solved form, has the same solutions in $\mathcal{R}$ as the equation $s(x) \doteq s(y)$. Without the decomposition rule, however, our solution procedure cannot avoid to compute a second solved form that is redundant:

	$\emptyset \;.\; s(x) \doteq s(y)$	
$\xrightarrow{r}_{\mathcal{R},\{y\}}$	$s(x) \doteq s(p(x')) \;.\; x' \doteq s(y)$	by *A*
$\xrightarrow{s}_{\mathcal{R},\{y\}}$	$\emptyset \;.\; x' \doteq s(y)$	by *Uni*
$\xrightarrow{r}_{\mathcal{R},\{y\}}$	$s(y) \doteq s(p(y')) \;.\; x' \doteq y'$	by *A*
$\xrightarrow{s}{}^{*}_{\mathcal{R},\{y\}}$	$y \doteq p(y') \;.\; \emptyset$	by *SB2, Uni.*

With the permutation rule *P3* it is possible to rename auxiliary variables, that is, variables that don't occur in V. We have included this rule to show that the introduction of new variables by the application rule (Figure 3.1) is actually a don't care nondeterminism.

4.4 Soundness and Completeness Proofs

Theorem 4.11. (Soundness) *If* $C \,.\, E \xrightarrow{s}_{\mathcal{R},V} C' \,.\, E'$ *by a simplification step, then* $\mathrm{SOL}^V_{\mathcal{R}}(C' \,\&\, E') \subseteq \mathrm{SOL}^V_{\mathcal{R}}(C \,\&\, E)$.

Proof. Let $C \,.\, E \xrightarrow{s}_{\mathcal{R},V} C' \,.\, E'$ by a simplification step and let θ be an assignment such that $\mathcal{D}\theta = V \cup \mathcal{V}(C' \,.\, E')$ and $\theta(C' \,\&\, E')$ is valid in $\mathcal{R}$. We have to show that there exists an assignment θ' such that $V \cup \mathcal{V}(C \,\&\, E) \subseteq \mathcal{D}\theta'$, θ' and θ agree on V, and $\theta'(C \,\&\, E)$ is valid in $\mathcal{R}$. Let the simplification rule employed in $C \,.\, E \xrightarrow{s}_{\mathcal{R},V} C' \,.\, E'$ be:

Uni. Then $C' \,.\, E' = S \,.\, E$, where S is solved, $\mathrm{SOL}^W_\Sigma(C) = \mathrm{SOL}^W_\Sigma(S)$ and $\mathcal{D}\langle S\rangle \subseteq W = V \cup \mathcal{V}(E)$. It suffices to show that there exists a ground substitution $\bar\theta$ such that $V \cup \mathcal{V}(C \,\&\, E) \subseteq \mathcal{D}\bar\theta$, $\theta|_V =_{\mathcal{R}} \bar\theta|_V$, and $\bar\theta(C \,\&\, E)$ is valid in $\mathcal{R}$, since then defining $\theta' x$ as the normal form of $\bar\theta x$ for every $x \in \mathcal{D}\bar\theta$ yields the claim.

Since S is solved, we know that $\langle S\rangle S$ is a trivial equation system, which implies that $\theta\langle S\rangle S$ is a trivial system. This implies $(\theta\langle S\rangle)|_W \in \mathrm{SOL}^W_\Sigma(S) = \mathrm{SOL}^W_\Sigma(C)$. Therefore, there exists a ground substitution $\bar{\theta}$ such that $\mathcal{D}\bar{\theta} = W \cup \mathcal{V}(C) = V \cup \mathcal{V}(E) \cup \mathcal{V}(C)$, $\bar{\theta}|_W = (\theta\langle S\rangle)|_W$, and $\bar{\theta}C$ is trivial. In particular, $\bar{\theta}C$ is valid in $\mathcal{R}$.

Since θS is valid in $\mathcal{R}$, we have $\theta =_\mathcal{R} \theta\langle S\rangle$, which yields $\theta|_W =_\mathcal{R} (\theta\langle S\rangle)|_W = \bar{\theta}|_W$. Since $W = V \cup \mathcal{V}(E)$, this yields that $\bar{\theta}E$ is valid in $\mathcal{R}$ and $\theta|_V =_\mathcal{R} \bar{\theta}|_V$.

R. Then $C.\, E = (S.\, P \,\&\, E_1)$ and $C'.\, E' = (S.\, P[\pi \leftarrow v] \,\&\, E_1)$, where $\langle S\rangle(P/\pi) \rightarrow v$ is an instance of a rule of $\mathcal{R}$. It suffices to show that $\theta P = (\theta P)[\pi \leftarrow \theta(P/\pi)]$ is valid in $\mathcal{R}$, which in turn follows from $\theta(P/\pi) =_\mathcal{R} \theta v$, since $(\theta P)[\pi \leftarrow \theta v] = \theta(P[\pi \leftarrow v])$ is valid in $\mathcal{R}$. Since θS is valid in $\mathcal{R}$, we know that $\theta =_\mathcal{R} \theta\langle S\rangle$. Hence, $\theta(P/\pi) =_\mathcal{R} \theta\langle S\rangle(P/\pi) =_\mathcal{R} \theta v$ as required.

Unf. Then $C.\, E = (C.\, P \,\&\, E_1)$ and $C'.\, E' = (C.\, x \doteq P/\pi \,\&\, P[\pi \leftarrow x] \,\&\, E_1)$, where x is a new variable. It suffices to show that $\theta P = (\theta P)[\pi \leftarrow \theta(P/\pi)]$ is valid in $\mathcal{R}$, which holds since $\theta x =_\mathcal{R} \theta(P/\pi)$ and $(\theta P)[\pi \leftarrow \theta x] = \theta(P[\pi \leftarrow x])$ is valid in $\mathcal{R}$.

SB1 or *SB2*. Then the claim is trivial.

D. Then the claim follows from the congruence property of the relation "$s =_\mathcal{R} t$".

S. Then $C.\, E = (S.\, P \,\&\, Q \,\&\, E_1)$ and $C'.\, E' = (S.\, Q \,\&\, E_1)$, where $\langle S\rangle P = \langle S\rangle Q$. It suffices to show that θP is valid in $\mathcal{R}$. Since θS is valid in $\mathcal{R}$, we have $\theta =_\mathcal{R} \theta\langle S\rangle$, and since θQ is valid in $\mathcal{R}$, we know that $\theta\langle S\rangle Q$ is valid in $\mathcal{R}$. This yields that θP is valid in $\mathcal{R}$ since $\langle S\rangle Q = \langle S\rangle P$.

P1. Then the claim is trivial.

P2. Then $C.\, E = (S.\, x \doteq s \,\&\, t \doteq u \,\&\, E_1)$ and $C'.\, E' = (S.\, x \doteq s \,\&\, x \doteq u \,\&\, E_1)$, where $\langle S\rangle s = \langle S\rangle t$. It suffices to show that $\theta t =_\mathcal{R} \theta u$. Since θS is valid in $\mathcal{R}$, we know that $\theta =_\mathcal{R} \theta\langle S\rangle$, which yields that $\theta t =_\mathcal{R} \theta\langle S\rangle t = \theta\langle S\rangle s =_\mathcal{R} \theta s =_\mathcal{R} \theta x =_\mathcal{R} \theta u$.

P3. Then $C'.\, E'$ has been obtained from $C.\, E$ by replacing all occurrences of x with y, where $x \notin V$ and $y \notin V \cup \mathcal{V}(C.\, E)$. Thus $\theta' := \theta[x \leftarrow \theta y]$ yields the claim. □

Our next goal is to prove the completeness of the extended solution procedure in Figure 4.1. As before, the proof will be based on the notion of a triple reduction relation, which links steps on the resolution level with steps on the verification level. We start by giving the corresponding verification rule for every simplification rule:

- *(VUni), (VP3)* $E \xrightarrow{vs}_\mathcal{R} E$
- *(VR)* $P \,\&\, E \xrightarrow{vs}_\mathcal{R} P[\pi \leftarrow v] \,\&\, E$ if $P/\pi \rightarrow v$ is an instance of a rule of $\mathcal{R}$
- *(VUnf)* $P \,\&\, E \xrightarrow{vs}_\mathcal{R} s \doteq P/\pi \,\&\, P[\pi \leftarrow s] \,\&\, E$ if s is the $\mathcal{R}$-normal form of P/π
- *(VSB1), (VSB2)* $P \,\&\, E \xrightarrow{vs}_\mathcal{R} E$ if P is a trivial equation
- *(VD)* $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n) \,\&\, E \xrightarrow{vs}_\mathcal{R} s_1 \doteq t_1 \,\&\, \ldots \,\&\, s_n \doteq t_n \,\&\, E$ if f is decomposable
- *(VS)* $P \,\&\, P \,\&\, E \xrightarrow{vs}_\mathcal{R} P \,\&\, E$
- *(VP1)* $s \doteq t \,\&\, E \xrightarrow{vs}_\mathcal{R} t \doteq s \,\&\, E$
- *(VP2)* $v \doteq s \,\&\, s \doteq u \,\&\, E \xrightarrow{vs}_\mathcal{R} v \doteq s \,\&\, v \doteq u \,\&\, E$ if v is $\mathcal{R}$-normal.

Proposition 4.12. **(Invariance)** *Let $E \xrightarrow{vs}_{\mathcal{R}} E'$. Then E is valid in $\mathcal{R}$ if and only if E' is valid in $\mathcal{R}$.*

The **$\mathcal{R}$-complexity** $\|E\|_{\mathcal{R}}$ of an equation system E is defined as the maximal length of an $\mathcal{R}$-rewriting derivation issuing from E.

Proposition 4.13. **(Compatibility)** *If $E \xrightarrow{vs}_{\mathcal{R}} E'$, then $\|E\|_{\mathcal{R}} \geq \|E'\|_{\mathcal{R}}$.*

Next we extend the simplification steps to $\mathcal{R}$-triples. We write $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$ if both $\theta.\ C.\ E$ and $\theta'.\ C'.\ E'$ are $\mathcal{R}$-triples and

- θ and θ' agree on V
- $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ by some simplification rule σ
- $\theta E \xrightarrow{vs}_{\mathcal{R}} \theta' E'$ by the verification rule corresponding to σ.

The next theorem is the counterpart to the push up theorem for the resolution rules. Since the application of the simplification rules is supposed to be don't care nondeterministic, we must be able to push down a simplification step from the resolution level to the verification level.

Theorem 4.14. **(Push Down)** *If $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ by a simplification step and $\theta.\ C.\ E$ is an $\mathcal{R}$-triple, then there exists an assignment θ' such that $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$.*

Proof. Let $\theta.\ C.\ E$ be an $\mathcal{R}$-triple. Then $\mathcal{V}(C.\ E) \subseteq \mathcal{D}\theta$, θC is trivial, and θE is valid in $\mathcal{R}$. We will show that for every simplification step $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ there exists an assignment θ' such that $\mathcal{V}(C'.\ E') \subseteq \mathcal{D}\theta'$, θ and θ' agree on V, $\theta' C'$ is trivial, and $\theta E \xrightarrow{vs}_{\mathcal{R}} \theta' E'$ by the corresponding verification step. Let the simplification rule employed in $C.\ E \xrightarrow{s}_{\mathcal{R},V} C'.\ E'$ be:

Uni. Then $C'.\ E' = S.\ E$, where S is solved, $\mathrm{SOL}_{\Sigma}^{W}(C) = \mathrm{SOL}_{\Sigma}^{W}(S)$, $\mathcal{D}\langle S\rangle \subseteq W$, and $W = V \cup \mathcal{V}(E)$. Since θC is trivial and $W \cup \mathcal{V}(C) \subseteq \mathcal{D}\theta$, we have $\theta|_W \in \mathrm{SOL}_{\Sigma}^{W}(C) = \mathrm{SOL}_{\Sigma}^{W}(S)$. Therefore, there exists a ground substitution θ' such that θ' agrees with θ on W, $\theta' S$ is trivial, and $\mathcal{D}\theta' = W \cup \mathcal{V}(S)$. Since $\mathcal{V}(E) \subseteq W$, we know that $\theta' E = \theta E$ is valid in $\mathcal{R}$. Thus, it suffices to show that $\theta' x$ is $\mathcal{R}$-normal for every $x \in W \cup \mathcal{V}(S) = W \cup \mathcal{I}(S)$.

If $x \in W$, then $\theta' x$ is $\mathcal{R}$-normal, since $\theta' x = \theta x$ and θx is $\mathcal{R}$-normal. If $x \in \mathcal{I}\langle S\rangle$, then there is an equation $y \doteq s$ in S such that x occurs in s and $y \in \mathcal{D}\langle S\rangle \subseteq W$. Hence, $\theta' x$ is a subterm of the term $\theta' s$, which is $\mathcal{R}$-normal since $\theta' s = \theta' y = \theta y$. Thus $\theta' x$ is $\mathcal{R}$-normal.

R. Then $C.\ E = (S.\ P\ \&\ E_1)$ and $C'.\ E' = (S.\ P[\pi \leftarrow v]\ \&\ E_1)$, where $\langle S\rangle(P/\pi) \rightarrow v$ is an instance of a rule of $\mathcal{R}$. It suffices to prove that $\theta P \xrightarrow{vs}_{\mathcal{R}} \theta(P[\pi \leftarrow v])$ by the verification rule *VR*, since then we can define $\theta' := \theta$.

Since $\theta.\ S.\ E$ is an $\mathcal{R}$-triple, θS is trivial. Hence $\theta = \theta\langle S\rangle$, which implies that $(\theta P)/\pi = \theta(P/\pi) = \theta\langle S\rangle(P/\pi)$. Thus $\theta P \xrightarrow{vs}_{\mathcal{R}} (\theta P)[\pi \leftarrow \theta v]$ by the verification rule *VR*, since $\langle S\rangle(P/\pi) \rightarrow v$ is an instance of a rule of $\mathcal{R}$.

Unf. Then $C.\ E = (C.\ P\ \&\ E_1)$ and $C'.\ E' = (C.\ x \doteq P/\pi\ \&\ P[\pi \leftarrow x]\ \&\ E_1)$, where x is a new variable. Defining $\theta' := \theta[x \leftarrow s]$, where s is the $\mathcal{R}$-normal form of $\theta(P/\pi)$, yields the claim.

SB1. Then $C.\ E = (S.\ x \doteq t\ \&\ E_1)$ and $C'.\ E' = (S\ \&\ x \doteq t.\ E_1)$, where S contains an equation $y \doteq s$ such that $\langle S\rangle t$ is a subterm of s. It suffices to show that $\theta x \doteq \theta t$ is a trivial equation, since then we can define $\theta' := \theta$.

Since $\theta.\ C.\ E$ is an $\mathcal{R}$-triple, we have that $\theta = \theta\langle S\rangle$, $\theta y = \theta s$, and $\theta x =_{\mathcal{R}} \theta t$. Since $\theta t = \theta\langle S\rangle t$ and $\langle S\rangle t$ is a subterm of s, we know that θt is a subterm of θs. Since $\theta s = \theta y$ is an $\mathcal{R}$-value, θt is an $\mathcal{R}$-value. Since θx is an $\mathcal{R}$-value and $\theta x =_{\mathcal{R}} \theta t$, we conclude that $\theta x = \theta t$.

SB2. Then $C.\ E = (C.\ P\ \&\ E_1)$ and $C'.\ E' = (C\ \&\ P.\ E_1)$, where every function symbol occurring in P is irreducible. It suffices to prove that θP is trivial, since then we can define $\theta' := \theta$. Since θ is normal and every function symbol occurring in P is irreducible, θP cannot be rewritten. Since θP is valid in $\mathcal{R}$, this yields the claim.

D, S, P1 or *P2*. For these rules $\theta' := \theta$ does the job.

P3. Then $C'.\ E'$ has been obtained from $C.\ E$ by replacing all occurrences of x with y, where $x \notin V$ and $y \notin V \cup \mathcal{V}(C.\ E)$. Defining $\theta' := \theta[y \leftarrow \theta x]$ yields the claim. □

We write $\theta.\ C.\ E \xrightarrow{sr}_{\mathcal{R},V} \theta'.\ C'.\ E'$ if $\theta.\ C.\ E \xrightarrow{s}{}^{*}_{\mathcal{R},V} \theta''.\ C''.\ E'' \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ for some $\mathcal{R}$-triple $\theta''.\ C''.\ E''$. By the push down and the push up theorem we know that the extended solution procedure builds a derivation

$$\theta.\ C.\ E \xrightarrow{sr}_{\mathcal{R},V} \theta'.\ C'.\ E' \xrightarrow{sr}_{\mathcal{R},V} \theta''.\ C''.\ E'' \xrightarrow{sr}_{\mathcal{R},V} \cdots,$$

provided the right don't know choices are made. Thus, we know that the procedure is complete if we can show that the triple reduction relation "$\theta.\ C.\ E \xrightarrow{sr}_{\mathcal{R},V} \theta'.\ C'.\ E'$" is terminating. To do this, we will define a complexity measure on triples that is decreased by resolution steps and not increased by simplification steps. A first attempt to define the complexity of a triple $\theta.\ C.\ E$ could be to use $\|E\|_{\mathcal{R}}$. However, this doesn't work since the resolution step B (blocking) doesn't necessarily decrease $\|E\|_{\mathcal{R}}$.

To define a complexity measure that works, we need a few auxiliary definitions. For a term s, let $|s|$ be the number of function symbols occurring in s. For an equation $s \doteq t$, define $|s \doteq t| := 0$ if s and t are variables and $|s \doteq t| := |s| + |t| - 1$ otherwise. For an equation system E, let $|E| := \sum_{P\in E} |P|$ and $\sharp E$ be the number of equations occurring in E. With that we define the complexity of an $\mathcal{R}$-triple as a triple of nonnegative integers:

$$|\theta.\ C.\ E| := (\|\theta E\|_{\mathcal{R}}, |E|, \sharp E).$$

On these complexities we obtain a well founded ordering "$|\theta.\ C.\ E| \geq |\theta' C'.\ E'|$" by extending the usual ordering on integers lexicographically.

Theorem 4.15. (Compatibility)

1. *If $\theta.\ C.\ E \xrightarrow{r}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a resolution step, then $|\theta.\ C.\ E| > |\theta' C'.\ E'|$.*
2. *If $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a simplification step, then $|\theta.\ C.\ E| \geq |\theta' C'.\ E'|$.*

Proof. *1.* Since application steps decrease $\|\theta E\|_{\mathcal{R}}$, and since blocking steps increase neither $\|\theta E\|_{\mathcal{R}}$ nor $|E|$, but decrease $\sharp E$, resolution steps decrease the complexity of a triple.

2. Let $\theta.\ C.\ E \xrightarrow{s}_{\mathcal{R},V} \theta'.\ C'.\ E'$ by a simplification step. By proposition 4.13, we know that no simplification step increases $\|\theta E\|_{\mathcal{R}}$. Therefore, it suffices to show that if a simplification step increases $|E|$, then it decreases $||\theta E||_{\mathcal{R}}$, and if a simplification step increases $\sharp E$, then it decreases $|E|$. The only rule that can increase $|E|$ is the rewriting rule, which does decrease $||\theta E||_{\mathcal{R}}$. The only rules that can increase $\sharp E$ are the unfolding and the decomposition rule, which do decrease $|E|$. □

Corollary 4.16. *The relation "θ. C. $E \xrightarrow{sr}_{\mathcal{R},V} \theta'$. C'. E'" is terminating.*

Corollary 4.17. *The solution procedure in Figure 4.1 is complete.*

The proof method we have developped in this and the last section can be used to show the completeness of alternative sets of resolution and simplification rules. Given such an alternative set of rules, the first step is to devise for every rule a suitable verification rule. The verification rules are applied to ground equation systems and must leave their validity invariant. The combination of the given rules with their corresponding verification rules then yields a reduction relation on triples. Next one defines a complexity measure on triples that is decreased by resolution steps and not increased by simplification steps. Then one shows with a push up theorem that every unsolved triple can be reduced by a resolution step on any given equation. Finally, one shows with a push down theorem that every triple can be reduced with any given simplification step.

If one uses an alternative set of resolution rules but the same complexity measure we used here, the simplification rules discussed here can be used without reproving anything. If the complexity measure is changed, it is still possible to reuse the push down theorem.

5 Refinements

In this section we discuss two refinements for the extended solution procedure. Both of them depend on additional knowledge about the underlying rewriting system.

5.1 Rewriting with Inductive Consequences

Let $\mathcal{R}$ be the rewriting system in Figure 1.1 and consider the equation $x + 0 \doteq 0$. Although this equation has the unique solution $\langle x \doteq 0 \rangle$ in $\mathcal{R}$, which is easily found, the extended solution procedure nevertheless has an infinite search space for this equation. To see this, consider the derivation steps

$$
\begin{array}{lll}
 & \emptyset \,.\, x + 0 \doteq 0 & \\
\xrightarrow{r}_{\mathcal{R},\{x\}} & x + 0 \doteq s(x') + y' \,.\, s(x' + y') \doteq 0 & \text{by } A \\
\xrightarrow{s}_{\mathcal{R},\{x\}} & x \doteq s(x') \;\&\; y' \doteq 0 \,.\, s(x' + y') \doteq 0 & \text{by } \mathit{Uni},
\end{array}
$$

which can be continued infinitely often by applying rule (4) to the occurrence of $+$. The obtained pair is actually inconsistent, but our simplification and failure rules are too weak to detect this inconsistency.

We can get rid of this annoying problem if we add the rule $x + 0 \to x$ to the rewriting system. Then the extended solution procedure can find the solution of $x + 0 \doteq 0$ by using simplification steps only. Since the equation $x + 0 \doteq x$ is an inductive consequence of $\mathcal{R}$ and the extended rewriting system still terminates, adding this rule doesn't change the solutions of an equation. We will show that the solution procedure stays complete if the new rule is used for simplification with the rewriting rule but is not used for resolution with the application rule.

Two ground confluent and terminating rewriting systems are **equivalent** if they have the same signature and every ground term has the same normal form in both systems. Equivalent rewriting systems define, up to isomorphism, the same initial algebra.

Proposition 5.1. *Let $\mathcal{R}$ and $\mathcal{R}'$ be two equivalent ground confluent and terminating rewriting systems. Then a ground equation is valid in $\mathcal{R}$ if and only if it is valid in $\mathcal{R}'$.*

Proposition 5.2. *Let $\mathcal{R} = (\Sigma, \mathcal{E})$ be a ground confluent and terminating rewriting system and $s \to t$ be a rewriting rule that is an inductive consequence of $\mathcal{R}$. Then $\mathcal{R}' := (\Sigma, \mathcal{E} \cup \{s \to t\})$ is a ground confluent rewriting system. Furthermore, if $\mathcal{R}'$ is terminating, then $\mathcal{R}$ and $\mathcal{R}'$ are equivalent.*

Theorem 5.3. *Let $\mathcal{R} = (\Sigma, \mathcal{E})$ and $\mathcal{R}' = (\Sigma, \mathcal{E} \cup \mathcal{E}')$ be two equivalent ground confluent and terminating rewriting systems. Then the extended solution procedure in Figure 4.1 is complete if the rules in $\mathcal{E}$ are employed for resolution steps and the rules in $\mathcal{E} \cup \mathcal{E}'$ are employed for simplification steps.*

Proof. It suffices to show that the Push Up Theorem still holds if only the rules in $\mathcal{E}$ are available for application steps. This is the case since every ground term that can be rewritten with a rule in $\mathcal{E} \cup \mathcal{E}'$ can also be rewritten with a rule in $\mathcal{E}$. □

The idea to use inductive consequences for rewriting also appears in Fribourg [85].

5.2 Free Rewriting Systems

A ground confluent and terminating rewriting system $\mathcal{R}$ is called **free** if every function symbol that is reducible in $\mathcal{R}$ is completely defined in $\mathcal{R}$. Recall that a function symbol f is reducible in $\mathcal{R}$ if f is the top symbol of the left hand side of at least one rule of $\mathcal{R}$, and that f is completely defined in $\mathcal{R}$ if f occurs in no $\mathcal{R}$-value. The rewriting system $\mathcal{R}1$ in Subsection 4.3 is an example for a free rewriting system. The irreducible function symbols of a free rewriting system are often called **constructors**. Furthermore, a term is called **canonical** in $\mathcal{R}$ if it doesn't contain a function symbol that is reducible in $\mathcal{R}$.

Proposition 5.4. *Let $\mathcal{R}$ be a free rewriting system. Then a ground term is an $\mathcal{R}$-value if and only if it is canonical.*

The reason we discuss free rewriting systems here is that for these systems the number of don't know alternatives our solution procedure has to explore can be significantly reduced. Given a free rewriting system $\mathcal{R}$, we call a term $f(s_1, \ldots, s_n)$ **simple** in $\mathcal{R}$ if its top symbol f is reducible in $\mathcal{R}$ and its arguments $s_1, \ldots, s_n$ are canonical in $\mathcal{R}$. The solution procedure in Figure 5.1 restricts resolution steps to rule applications to don't care chosen simple subterms. To prove that this procedure is complete for free rewriting systems, we have to show two things. First, it must always be possible to simplify a pair $C.\ E$ such that the unsolved part contains only equations that contain at least one simple term. This is the case since an equation that doesn't contain a simple term contains only irreducible function symbols and can thus be blocked with the simplification rule *SB2*. Second, we need a stronger push up theorem:

Theorem 5.5. (Push Up for Free Rewriting Systems) *Let $\mathcal{R}$ be a free rewriting system. Then, if $\theta.\ C.\ E$ is an $\mathcal{R}$-triple, P is an equation in E, and P/π is a simple subterm* solve(C. E) is

1. choose **don't care** C'. E' such that C. $E \xrightarrow{s}{}^{*}_{\mathcal{R},V} C'$. E' by simplification steps and every equation in E' contains at least one simple term;
2. if a failure rule applies to C'. E', then fail;
3. if E' is empty, then return C';
4. choose **don't care** an equation P in E' and a simple subterm P/π in P;
5. choose **don't know** C''. E'' such that C'. $E' \xrightarrow{r}_{\mathcal{R},V} C''$. E'' by an application step on P at π;
6. solve(C''. E'')

Figure 5.1. A solution procedure for free rewriting system.

of P, there exists a triple θ'. C'. E' such that θ. C. $E \xrightarrow{r}_{\mathcal{R},V} \theta'$. C'. E' by an application step on P at π.

Proof. Let θ. C. $P\&E$ be an $\mathcal{R}$-triple and P/π be a simple subterm of P. Then $\theta(P/\pi)$ is an innermost ground term. Thus there exist a variant $u \to v$ of a rule of $\mathcal{R}$ and a substitution ϕ such that $\phi u = (\theta P)/\pi$. From here on the proof is identical with the proof of the push up theorem in Section 3. □

Corollary 5.6. *The solution procedure in Figure 5.1 is complete for free rewriting systems.*

Fribourg [85] discusses a similar solution procedure for free conditional rewriting systems. He has the additional requirement that the left hand sides of all rules be simple terms.

There is actually no need for reproving a stronger version of the push up theorem, since our simplification rules are already strong enough to justify the solution procedure for free rewriting systems. In fact, the solution procedure in Figure 5.1 just realizes one of the many strategies that one can obtain by using the unfolding rule in conjunction with the don't care selection of the next equation to be resolved upon. To see this, first notice that every equation that doesn't contain a simple term can be safely blocked with the simplification rule *SB2*. Secondly, any simple term s contained in an equation can be unfolded into an equation $x \doteq s$, which then can be chosen to be the next equation to be resolved upon. Blocking such an equation immediately yields an inconsistent pair, as we know by failure rule (3) since the top symbol of s is completely defined. Furthermore, any application step to a proper subterm s/π of s yields an inconsistent pair, as we know by failure rule (1) since the top symbol of s/π is irreducible, that is, is different from the top symbol of the left hand side of any rewriting rule. Thus we are left with exactly the don't know alternatives that are considered by the solution procedure for free rewriting systems.

The left-to-right basic narrowing strategy in [Herold 86] and the selection narrowing strategy in [Bosco et al. 87] are two further examples for the strategies that can be obtained by using the unfolding rule.

References

P.G. Bosco, E. Giovannetti, and C. Moiso, Refined Strategies for Semantic Unification. Proc. of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 1987, Springer LNCS 250, 276–290.

N. Dershowitz and D. Plaisted, Logic Programming cum Applicative Programming. Proc. of the 1985 Symposium on Logic Programming, Boston, July 1985, 54–67.

M. Fay, First Order Unification in an Equational Theory. Proc. of the 4th Workshop on Automated Deduction, University of Texas, Austin 1979, 161–167.

L. Fribourg, SLOG: A Logic Programming Language Inerpreter Based on Clausal Superposition and Rewriting. Proc. of the 1985 Symposium on Logic Programming, Boston, July 1985, 172–184.

J. Gallier and W. Snyder, A General Complete E-Unification Procedure. Proc. of the 2nd International Conference on Rewriting Techniques and Applications, Bordeaux, France, May 1987, Springer LNCS 256, 216–227.

J.A. Goguen and J. Meseguer, Eqlog: Equality, Types, and Generic Modules for Logic Programming. In D. DeGroot and G. Lindstrom (ed.), Logic Programming, Functions, Relations, and Equations. Prentice Hall 1986, 179–210.

A. Herold, Narrowing Techniques Applied to Idempotent Unification. Seki Report SR-86-16, Universität Kaiserslautern, West Germany, 1986.

S. Hölldobler, A Unification Algorithm for Confluent Theories. Proc. of the 14th International Conference on Automata, Languages, and Programming, Karlsruhe, Germany, 1987, Springer LNCS 267, 31–41.

G. Huet, Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. Journal of the and Associativity. Journal of the ACM 27,4 (1980), 797–821.

G. Huet and D.C. Oppen, Equations and Rewrite Rules: A Survey. In R. Book (ed.), Formal Languagues: Perspectives and Open Problems, Academic Press 1980, 349–405.

J.-M. Hullot, Canonical Forms and Unification. Proc. of the 5th Conference on Automated Deduction, 1980, Les Arcs, France, Springer LNCS 87, 318–334.

H. Hußmann, Unification in Conditional Equational Theories. Proc. of the EUROCAL '85, Springer LNCS 204, 543–553.

A. Josephson and N. Dershowitz, An Implementation of Narrowing: The RITE Way. Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 187–197.

S. Kaplan, Fair Conditional Term Rewriting Systems: Unification, Termination, and Confluence. Technical Report no. 194, Université de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique, 1984.

C. Kirchner, Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationelles, These d'état de l'Université de Nancy I, 1985.

D.S. Lankford, Canonical Inference, Technical Report ATP-32, Department of Mathematics and Computer Science, University of Texas at Austin, December 1975.

J.W. Lloyd, Foundations of Logic Programming. Springer Verlag, 1984.

A. Martelli, C. Moiso, and G.F. Rossi, An Algorithm for Unification in Equational Theories. Proc. of the 1986 Symposium on Logic Programming, Salt Lake City, 180–186.

P. Réty, C. Kirchner, H. Kirchner, and P. Lescanne, Narrower: A New Algorithm for Unification and its Application to Logic Programming. Proc. of the 1st International Conference on Term Rewriting Techniques and Applications, Dijon, France, May 1985, Springer LNCS 202, 141–157.

P. Réty, Improving Basic Narrowing Techniques. Proc. of the 2nd International Conference on Rewriting Techniques and Applications, Bordeaux, France, May 1987, Springer LNCS 256, 228–241. Also presented at the 1st Workshop on Unification, Val d'Ajol, France, March 1987.

G.A. Robinson and L. Wos, Paramodulation and Theorem-Proving in First-Order Theories with Equality. Machine Intelligence 4, Edinburgh University Press, 1969, 135–150.

J.A. Robinson, A Machine-Oriented Logic Based on the Resolution Principle. Journal of the ACM 12, 1965, 23–41.

J.R. Slagle, Automated Theorem Proving for Theories with Simplifiers, Commutativity, and Associativity. Journal of the ACM, Vol. 21, No. 4, October 1974, 622–642.

G. Smolka and W. Nutt, Lazy Basic Order-Sorted Narrowing. Presented at the 1st Workshop on Unification, Val d'Ajol, France, March 1987.

G. Smolka, W. Nutt, J. Goguen, and J. Meseguer, Order-Sorted Equational Computation. Presented at the Colloquium on the Resolution of Equations in Algebraic Structures, Lakeway, Texas, May 1987.

J.-H. You and P.A. Subrahmanyam, A Class of Confluent Term Rewriting Systems and Unification. Journal of Automated Reasoning 2, 1986, 391–418.