Chapter 4

Equational Logic

From now on First-order Logic is considered with equality. In this chapter, I investigate properties of a set of unit equations. For a set of unit equations I write E. Full first-order clauses with equality are studied in Chapter 5. I recall certain definitions from Section 1.6 and Chapter 3.

The main reasoning problem considered in this chapter is given a set of unit equations E and an additional equation $s \approx t$, does $E \models s \approx t$ hold? As usual, all variables are implicitely universally quantified. The idea is to turn the equations E into a convergent term rewrite system (TRS) R such that the above problem can be solved by checking identity of the respective normal forms: $s\downarrow_R = t\downarrow_R$. Showing $E \models s \approx t$ is as difficult as proving validity of any first-order formula, see Section 3.15.

For example consider the euqational ground clauses $E = \{g(a) \approx b, a \approx b\}$ over a signature consisting of the constants a, b and unary function g, all defined over some unique sort. Then for all algebras $\mathcal{A}$ satifying E, all ground terms over a, b, and g, are mapped to the same domain element. In particular, it holds $E \models g(b) \approx b$. Now the idea is to turn E into a convergent term rewrite system R such that $g(b)\downarrow_R = b\downarrow_R$. To this end, the equations in E are oriented, e.g., a first guess might be the TRS $R_0 = \{g(a) \to b, a \to b\}$. For R_0 we get $g(b)\downarrow_{R_0} = g(b)$, $b\downarrow_{R_0} = b$, so not the desired result. The TRS R_0 is not confluent an all ground terms, because $g(a) \to_{R_0} b$ and $g(a) \to_{R_0} g(b)$, but b and $g(b)$ are R_0 normal forms. This problem can be repaired by adding the extra rule $g(b) \to b$ and this process is called *completion* and is studied in this chapter. Now the extended rewrite system $R_1 = \{g(a) \to b, a \to b, g(b) \to b\}$ is convergent and $g(b)\downarrow_{R_1} = b\downarrow_{R_1} = b$. Termination can be shown by using a KBO (or LPO) with precedence $g \succ a \succ b$. Then the left hand sides of the rules are strictly larger than the right hand sides. Actually, R_1 contains some redundancy, even removing the first rewrite rule $g(a) \to b$ from R_1 does not violate confluence. Detecting redundant rules is also discussed in this chapter.

Definition 4.0.1 (Equivalence Relation, Congruence Relation)**.** An *equivalence* relation $\sim$ on a term set $T(\Sigma, \mathcal{X})$ is a reflexive, transitive, symmetric binary relation on $T(\Sigma, \mathcal{X})$ such that if $s \sim t$ then $\mathrm{sort}(s) = \mathrm{sort}(t)$.

Two terms s and t are called *equivalent*, if $s \sim t$.

An equivalence $\sim$ is called a *congruence* if $s \sim t$ implies $u[s] \sim u[t]$, for all terms $s, t, u \in T(\Sigma, \mathcal{X})$. Given a term $t \in T(\Sigma, \mathcal{X})$, the set of all terms equivalent to t is called the *equivalence class of t by $\sim$*, denoted by $[t]_\sim := \{t' \in T(\Sigma, \mathcal{X}) \mid t' \sim t\}$.

If the matter of discussion does not depend on a particular equivalence relation or it is unambiguously known from the context, $[t]$ is used instead of $[t]_\sim$. The above definition is equivalent to Definition 3.2.3.

The set of all equivalence classes in $T(\Sigma, \mathcal{X})$ defined by the equivalence relation is called a *quotient by* $\sim$, denoted by $T(\Sigma, \mathcal{X})|_\sim := \{[t] \mid t \in T(\Sigma, \mathcal{X})\}$. Let E be a set of equations then $\sim_E$ denotes the smallest congruence relation "containing" E, that is, $(l \approx r) \in E$ implies $l \sim_E r$. The equivalence class $[t]_{\sim_E}$ of a term t by the equivalence (congruence) $\sim_E$ is usually denoted, for short, by $[t]_E$. Likewise, $T(\Sigma, \mathcal{X})|_E$ is used for the quotient $T(\Sigma, \mathcal{X})|_{\sim_E}$ of $T(\Sigma, \mathcal{X})$ by the equivalence (congruence) $\sim_E$.

4.1 Term Rewrite System

I instantiate the abstract rewrite systems of Section 1.6 with first-order terms. The main difference is that rewriting takes not only place at the top position of a term, but also at inner positions.

Definition 4.1.1 (Rewrite Rule, Term Rewrite System)**.** A *rewrite rule* is an equation $l \approx r$ between two terms l and r so that l is not a variable and $vars(l) \supseteq vars(r)$. A *term rewrite system* R, or a TRS for short, is a set of rewrite rules.

Definition 4.1.2 (Rewrite Relation)**.** Let E be a set of (implicitly universally quantified) equations, i.e., unit clauses containing exactly one positive equation. The *rewrite relation* $\rightarrow_E \subseteq T(\Sigma, \mathcal{X}) \times T(\Sigma, \mathcal{X})$ is defined by

$$s \rightarrow_E t \quad \text{iff} \quad \begin{array}{l} \text{there exist } (l \approx r) \in E, p \in pos(s), \\ \text{and matcher } \sigma, \text{ so that } s|_p = l\sigma \text{ and } t = s[r\sigma]_p. \end{array}$$

Note that in particular for any equation $l \approx r \in E$ it holds $l \rightarrow_E r$, so the equation can also be written $l \rightarrow r \in E$.

Often $s = t \downarrow_R$ is written to denote that s is a normal form of t with respect to the rewrite relation $\rightarrow_R$. Notions $\rightarrow_R^0, \rightarrow_R^+, \rightarrow_R^*, \leftrightarrow_R^*$, etc. are defined accordingly, see Section 1.6. An instance of the left-hand side of an equation is called a *redex* (reducible expression). *Contracting* a redex means replacing it with the corresponding instance of the right-hand side of the rule. A term rewrite system R is called *convergent* if the rewrite relation $\rightarrow_R$ is confluent and terminating. A set of equations E or a TRS R is terminating if the rewrite relation $\rightarrow_E$ or $\rightarrow_R$ has this property. Furthermore, if E is terminating then it is a TRS. A rewrite system is called *right-reduced* if for all rewrite rules $l \rightarrow r$

in R, the term r is irreducible by R. A rewrite system R is called *left-reduced* if for all rewrite rules $l \to r$ in R, the term l is irreducible by $R\backslash\{l \to r\}$. A rewrite system is called *reduced* if it is left- and right-reduced.

Lemma 4.1.3 (Left-Reduced TRS)**.** Left-reduced terminating rewrite systems are convergent. Convergent rewrite systems define unique normal forms.

Lemma 4.1.4 (TRS Termination)**.** A rewrite system R terminates iff there exists a reduction ordering $\succ$ so that $l \succ r$, for each rule $l \to r$ in R.

4.1.1 E-Algebras

Let E be a set of universally quantified equations. A model $\mathcal{A}$ of E is also called an *E-algebra*. If $E \models \forall\vec{x}(s \approx t)$, i.e., $\forall\vec{x}(s \approx t)$ is valid in all E-algebras, this is also denoted with $s \approx_E t$. The goal is to use the rewrite relation $\to_E$ to express the semantic consequence relation syntactically: $s \approx_E t$ if and only if $s \leftrightarrow_E^* t$. Let E be a set of (well-sorted) equations over $T(\Sigma, \mathcal{X})$ where all variables are implicitly universally quantified. The following inference system allows to derive consequences of E:

Reflexivity $E \;\Rightarrow_{\mathrm{E}}\; E \cup \{t \approx t\}$

Symmetry $E \uplus \{t \approx t'\} \;\Rightarrow_{\mathrm{E}}\; E \cup \{t \approx t'\} \cup \{t' \approx t\}$

Transitivity $E \uplus \{t \approx t', t' \approx t''\} \;\Rightarrow_{\mathrm{E}}\; E \cup \{t \approx t', t' \approx t''\} \cup \{t \approx t''\}$

Congruence $E \uplus \{t_1 \approx t_1', \ldots, t_n \approx t_n'\} \;\Rightarrow_{\mathrm{E}}\; E \cup \{t_1 \approx t_1', \ldots, t_n \approx t_n'\} \cup \{f(t_1, \ldots, t_n) \approx f(t_1', \ldots, t_n')\}$
for any function $f : \mathrm{sort}(t_1) \times \ldots \times \mathrm{sort}(t_n) \to S$ for some S

Instance $E \uplus \{t \approx t'\} \;\Rightarrow_{\mathrm{E}}\; E \cup \{t \approx t'\} \cup \{t\sigma \approx t'\sigma\}$
for any well-sorted substitution σ

Lemma 4.1.5 (Equivalence of $\leftrightarrow_E^*$ and $\Rightarrow_E^*$)**.** The following properties are equivalent:

1. $s \leftrightarrow_E^* t$
2. $E \Rightarrow_E^* s \approx t$ is derivable.

where $E \Rightarrow_E^* s \approx t$ is an abbreviation for $E \Rightarrow_E^* E'$ and $s \approx t \in E'$.

Proof. (i)⇒(ii): $s \leftrightarrow_E t$ implies $E \Rightarrow_E^* s \approx t$ by induction on the depth of the position where the rewrite rule is applied; then $s \leftrightarrow_E^* t$ implies $E \Rightarrow_E^* s \approx t$ by induction on the number of rewrite steps in $s \leftrightarrow_E^* t$.

(ii)⇒(i): By induction on the size (number of symbols) of the derivation for $E \Rightarrow_E^* s \approx t$. □

Corollary 4.1.6 (Convergence of E)**.** If a set of equations E is convergent then $s \approx_E t$ if and only if $s \leftrightarrow^* t$ if and only if $s \downarrow_E = t \downarrow_E$.

Corollary 4.1.7 (Decidability of $\approx_E$)**.** If a set of equations E is finite and convergent then $\approx_E$ is decidable.

The above Lemma 4.1.5 shows equivalence of the syntactically defined relations $\leftrightarrow_E^*$ and $\Rightarrow_E^*$. What is missing, in analogy to Herbrand's theorem for first-order logic without equality Theorem 3.5.5, is a semantic characterization of the relations by a particular algebra.

Definition 4.1.8 (Quotient Algebra)**.** For sets of unit equations this is a *quotient algebra*: Let X be a set of variables. For $t \in T(\Sigma, \mathcal{X})$ let $[t] = \{t' \in T(\Sigma, \mathcal{X})) \mid E \Rightarrow_{\mathrm{E}}^* t \approx t'\}$ be the *congruence class* of t. Define a Σ-algebra $\mathcal{I}_E$, called the *quotient algebra*, technically $T(\Sigma, \mathcal{X})/E$, as follows: $S^{\mathcal{I}_E} = \{[t] \mid t \in T_S(\Sigma, \mathcal{X})\}$ for all sorts S and $f^{\mathcal{I}_E}([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$ for $f : \mathrm{sort}(t_1) \times \ldots \times \mathrm{sort}(t_n) \to T \in \Omega$ for some sort T.

Lemma 4.1.9 ($\mathcal{I}_E$ is an E-algebra)**.** $\mathcal{I}_E = T(\Sigma, \mathcal{X})/E$ is an E-algebra.

Proof. Firstly, all functions $f^{\mathcal{I}_E}$ are well-defined: if $[t_i] = [t_i']$, then $[f(t_1, \ldots, t_n)] = [f(t_1', \ldots, t_n')]$. This follows directly from the Congruence rule for $\Rightarrow^*$.

Secondly, let $\forall x_1 \ldots x_n (s \approx t)$ be an equation in E. Let β be an arbitrary assignment. It has to be shown that $\mathcal{I}_E(\beta)(\forall \vec{x}(s \approx t)) = 1$, or equivalently, that $\mathcal{I}_E(\gamma)(s) = \mathcal{I}_E(\gamma)(t)$ for all $\gamma = \beta[x_i \mapsto [t_i] \mid 1 \le i \le n]$ with $[t_i] \in \mathrm{sort}(x_i)^{\mathcal{I}_E}$. Let $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, with $t_i \in T_{\mathrm{sort}(x_i)}(\Sigma, \mathcal{X})$, then $s\sigma \in \mathcal{I}_E(\gamma)(s)$ and $t\sigma \in \mathcal{I}_E(\gamma)(t)$. By the Instance rule, $E \Rightarrow^* s\sigma \approx t\sigma$ is derivable, hence $\mathcal{I}_E(\gamma)(s) = [s\sigma] = [t\sigma] = \mathcal{I}_E(\gamma)(t)$. □

Lemma 4.1.10 ($\Rightarrow_E$ is complete)**.** Let $\mathcal{X}$ be a countably infinite set of variables; let $s, t \in T_S(\Sigma, \mathcal{X})$. If $\mathcal{I}_E \models \forall \vec{x}(s \approx t)$, then $E \Rightarrow_E^* s \approx t$ is derivable.

Proof. Assume that $\mathcal{I}_E \models \forall \vec{x}(s \approx t)$, i.e., $\mathcal{I}_E(\beta)(\forall \vec{x}(s \approx t)) = 1$. Consequently, $\mathcal{I}_E(\gamma)(s) = \mathcal{I}_E(\gamma)(t)$ for all $\gamma = \beta[x_i \mapsto [t_i] \mid 1 \le i \le n]$ with $[t_i] \in \mathrm{sort}(x_i)^{\mathcal{I}_E}$. Choose $t_i = x_i$, then $[s] = \mathcal{I}_E(\gamma)(s) = \mathcal{I}_E(\gamma)(t) = [t]$, so $E \Rightarrow^* s \approx t$ is derivable by definition of $\mathcal{I}_E$. □

Theorem 4.1.11 (Birkhoff's Theorem)**.** Let $\mathcal{X}$ be a countably infinite set of variables, let E be a set of (universally quantified) equations. Then the following properties are equivalent for all $s, t \in T_S(\Sigma, \mathcal{X})$:

1. $s \leftrightarrow_E^* t$.

2. $E \Rightarrow_E^* s \approx t$ is derivable.

3. $s \approx_E t$, i.e., $E \models \forall \vec{x}(s \approx t)$.

4. $\mathcal{I}_E \models \forall \vec{x}(s \approx t)$.

Proof. (1.)⇔(2.): Lemma 4.1.5.
(2.)⇒(3.): By induction on the size of the derivation for $E \Rightarrow^* s \approx t$.
(3.)⇒(4.): Obvious, since $\mathcal{I}_E = T(\Sigma, \mathcal{X})/E$ is an E-algebra.
(4.)⇒(2.): Lemma 4.1.10. □

Universal Algebra

$T(\Sigma,\mathcal{X})/E = T(\Sigma,\mathcal{X})/{\approx_E} = T(\Sigma,\mathcal{X})/{\leftrightarrow_E^*}$ is called the *free E-algebra* with generating set $\mathcal{X}/{\approx_E} = \{[x] \mid x \in \mathcal{X}\}$: Every mapping $\phi : \mathcal{X}/{\approx_E} \to \mathcal{B}$ for some E-algebra $\mathcal{B}$ can be extended to a homomorphism $\hat{\phi} : T(\Sigma,\mathcal{X})/E \to \mathcal{B}$.

$T(\Sigma,\emptyset)/E = T(\Sigma,\emptyset)/{\approx_E} = T(\Sigma,\emptyset)/{\leftrightarrow_E^*}$ is called the *initial E-algebra.*

$\approx_E = \{(s,t) \mid E \models s \approx t\}$ is called the *equational theory* of E.

$\approx_E^I = \{(s,t) \mid T(\Sigma,\emptyset)/E \models s \approx t\}$ is called the *inductive theory* of E.

Example 4.1.12. Let $E = \{\forall x(x + 0 \approx x),\ \forall x \forall y(x + s(y) \approx s(x + y))\}$. Then $x + y \approx_E^I y + x$, but $x + y \not\approx_E y + x$.

4.2 Critical Pairs

By Theorem 4.1.11 the semantics of E and $\leftrightarrow_E^*$ conincide. In order to decide $\leftrightarrow_E^*$ we need to turn $\to_E^*$ in a confluent and terminating relation. If $\leftrightarrow_E^*$ is terminating then confluence is equivalent to local confluence, see Newman's Lemma, Lemma 1.6.6. Local confluence is the following problem for TRS: if $t_1\ {}_E\!\leftarrow t_0 \to_E t_2$, does there exist a term s so that $t_1 \to_E^* s\ {}_E^*\!\leftarrow t_2$? If the two rewrite steps happen in different subtrees (disjoint redexes) then a repitition of the respective other step yields the common term s. If the two rewrite steps happen below each other (overlap at or below a variable position) again a repetition of the respective other step yields the common term s. If the left-hand sides of the two rules overlap at a non-variable position there is no ovious way to generate s.

More technically two rewrite rules $l_1 \to r_1$ and $l_2 \to r_2$ overlap if there exist some non-variable subterm $l_1|_p$ such that l_2 and $l_1|_p$ have a common instance $(l_1|_p)\sigma_1 = l_2\sigma_2$. If the two rewrite rules do not have common variables, then only a single substitution is necessary, the mgu σ of $(l_1|_p)$ and l_2.

Definition 4.2.1 (Critical Pair)**.** Let $l_i \to r_i$ $(i = 1, 2)$ be two rewrite rules in a TRS R whithout common variables, i.e., $\mathrm{vars}(l_1) \cap \mathrm{vars}(l_2) = \emptyset$. Let $p \in \mathrm{pos}(l_1)$ be a position so that $l_1|_p$ is not a variable and σ is an mgu of $l_1|_p$ and l_2. Then $r_1\sigma \leftarrow l_1\sigma \to (l_1\sigma)[r_2\sigma]_p$. $\langle r_1\sigma, (l_1\sigma)[r_2\sigma]_p\rangle$ is called a *critical pair* of R. The critical pair is *joinable* (or: converges), if $r_1\sigma \downarrow_R (l_1\sigma)[r_2\sigma]_p$.

Recall that $\text{vars}(l_i) \supseteq \text{vars}(r_i)$ for the two rewrite rules by Definition 4.1.1.

Theorem 4.2.2 ("Critical Pair Theorem")**.** A TRS R is locally confluent iff all its critical pairs are joinable.

Proof. ($\Rightarrow$) Obvious, since joinability of a critical pair is a special case of local confluence.

($\Leftarrow$) Suppose s rewrites to t_1 and t_2 using rewrite rules $l_i \to r_i \in R$ at positions $p_i \in \text{pos}(s)$, where $i = 1, 2$. The two rules are variable disjoint, hence $s|_{p_i} = l_i\sigma$ and $t_i = s[r_i\sigma]_{p_i}$. There are two cases to be considered:

1. Either p_1 and p_2 are in disjoint subtrees ($p_1 \,||\, p_2$) or
2. one is a prefix of the other (w.l.o.g., $p_1 \leq p_2$).

Case 1: $p_1 \,||\, p_2$. Then $s = s[l_1\sigma]_{p_1}[l_2\sigma]_{p_2}$, and therefore $t_1 = s[r_1\sigma]_{p_1}[l_2\sigma]_{p_2}$ and $t_2 = s[l_1\sigma]_{p_1}[r_2\sigma]_{p_2}$. Let $t_0 = s[r_1\sigma]_{p_1}[r_2\sigma]_{p_2}$. Then clearly $t_1 \to_R t_0$ using $l_2 \to r_2$ and $t_2 \to_R t_0$ using $l_1 \to r_1$.
Case 2: $p_1 \leq p_2$.
Case 2.1: $p_2 = p_1q_1q_2$, where $l_1|_{q_1}$ is some variable x. In other words, the second rewrite step takes place at or below a variable in the first rule. Suppose that x occurs m times in l_1 and n times in r_1 (where $m \geq 1$ and $n \geq 0$). Then $t_1 \to_R^* t_0$ by applying $l_2 \to r_2$ at all positions $p_1q'q_2$, where q' is a position of x in r_1. Conversely, $t_2 \to_R^* t_0$ by applying $l_2 \to r_2$ at all positions p_1qq_2, where q is a position of x in l_1 different from q_1, and by applying $l_1 \to r_1$ at p_1 with the substitution σ', where $\sigma' = \sigma[x \mapsto (x\sigma)[r_2\sigma]_{q_2}]$.
Case 2.2: $p_2 = p_1p$, where p is a non-variable position of l_1. Then $s|_{p_2} = l_2\sigma$ and $s|_{p_2} = (s|_{p_1})|_p = (l_1\sigma)|_p = (l_1|_p)\sigma$, so σ is a unifier of l_2 and $l_1|_p$.Let σ' be the mgu of l_2 and $l_1|_p$, then $\sigma = \tau \circ \sigma'$ and $\langle r_1\sigma', (l_1\sigma')[r_2\sigma']_p\rangle$ is a critical pair. By assumption, it is joinable, so $r_1\sigma' \to_R^* v \leftarrow_R^* (l_1\sigma')[r_2\sigma']_p$. Consequently, $t_1 = s[r_1\sigma]_{p_1} = s[r_1\sigma'\tau]_{p_1} \to_R^* s[v\tau]_{p_1}$ and $t_2 = s[r_2\sigma]_{p_2} = s[(l_1\sigma)[r_2\sigma]_p]_{p_1} = s[(l_1\sigma'\tau)[r_2\sigma'\tau]_p]_{p_1} = s[((l_1\sigma')[r_2\sigma']_p)\tau]_{p_1} \to_R^* s[v\tau]_{p_1}$. □

Please note that critical pairs between a rule and (a renamed variant of) itself must be considered, except if the overlap is at the root, i.e., $p = \epsilon$, because this critical pair always joins.

Corollary 4.2.3. A terminating TRS R is confluent if and only if all its critical pairs are joinable.

Proof. By the Theorem 4.2.2 and because every locally confluent and terminating relation $\to$ is confluent, Newman's Lemma, Lemma 1.6.6. □

Corollary 4.2.4. For a finite terminating TRS, confluence is decidable.

Proof. For every pair of rules and every non-variable position in the first rule there is at most one critical pair $\langle u_1, u_2\rangle$. Reduce every u_i to some normal form u_i'. If $u_1' = u_2'$ for every critical pair, then R is confluent, otherwise there is some non-confluent situation $u_1' \ {}_R^*\!\!\leftarrow u_1 \leftarrow_R s \to_R u_2 \to_R^* u_2'$. □

$l \to r \in R$. For ensuring confluence the calculus checks whether all critical pairs are joinable.

The completion procedure itself is presented as a set of abstract rewrite rules working on a pair of equations E and rules R: $(E_0;R_0) \Rightarrow_{\mathrm{KBC}} (E_1;R_1) \Rightarrow_{\mathrm{KBC}} (E_1;R_2) \Rightarrow_{\mathrm{KBC}} \ldots$. The initial state is $(E_0, \emptyset)$ where $E = E_0$ contains the input equations. If $\Rightarrow_{\mathrm{KBC}}$ successfully terminates then E is empty and R is the convergent rewrite system for E_0. For each step $(E;R) \Rightarrow_{\mathrm{KBC}} (E';R')$ the equational theories of $E \cup R$ and $E' \cup R'$ agree: $\approx_{E\cup R} = \approx_{E'\cup R'}$. By $\mathrm{cp}(R)$ I denote the set of critical pairs between rules in R.

Orient $\quad (E \uplus \{s \mathrel{\dot{\approx}} t\}; R) \ \Rightarrow_{\mathrm{KBC}} \ (E; R \cup \{s \to t\})$
if $s \succ t$

Delete $\quad (E \uplus \{s \approx s\}; R) \ \Rightarrow_{\mathrm{KBC}} \ (E; R)$

Deduce $\quad (E; R) \ \Rightarrow_{\mathrm{KBC}} \ (E \cup \{s \approx t\}; R)$
if $\langle s, t\rangle \in \mathrm{cp}(R)$

Simplify-Eq $\quad (E \uplus \{s \mathrel{\dot{\approx}} t\}; R) \ \Rightarrow_{\mathrm{KBC}} \ (E \cup \{u \approx t\}; R)$
if $s \to_R u$

R-Simplify-Rule $\quad (E; R \uplus \{s \to t\}) \ \Rightarrow_{\mathrm{KBC}} \ (E; R \cup \{s \to u\})$
if $t \to_R u$

L-Simplify-Rule $\quad (E; R \uplus \{s \to t\}) \ \Rightarrow_{\mathrm{KBC}} \ (E \cup \{u \approx t\}; R)$
if $s \to_R u$ using a rule $l \to r \in R$ so that $s \sqsupset l$, see below.

Trivial equations cannot be oriented and since they are not needed they can be deleted by the Delete rule. The rule Deduce turns critical pairs between rules in R into additional equations. Note that if $\langle s, t\rangle \in \mathrm{cp}(R)$ then $s \,{}_R{\leftarrow} u \to_R t$ and hence $R \models s \approx t$. The simplification rules are not needed but serve as reduction rules, removing redundancy from the state. Simplification of the left-hand side may influence orientability and orientation of the result. Therefore, it yields an equation. For technical reasons, the left-hand side of $s \to t$ may only be simplified using a rule $l \to r$, if $l \to r$ cannot be simplified using $s \to t$, that is, if $s \sqsupset l$, where the *encompassment quasi-ordering* $\underset{\sim}{\sqsupset}$ is defined by $s \underset{\sim}{\sqsupset} l$ if $s|_p = l\sigma$ for some p and σ and $\sqsupset = \underset{\sim}{\sqsupset} \setminus \underset{\sim}{\sqsubset}$ is the strict part of $\underset{\sim}{\sqsupset}$.

Lemma 4.4.1. $\sqsupset$ is a well-founded strict partial ordering.

Lemma 4.4.2. If $(E; R) \Rightarrow_{KBC} (E'; R')$, then $\approx_{E\cup R} = \approx_{E'\cup R'}$.

Lemma 4.4.3. If $(E; R) \Rightarrow_{KBC} (E'; R')$ and $\to_R \subseteq \succ$, then $\to_{R'} \subseteq \succ$.

Proposition 4.4.4 (Knuth-Bendix Completion Correctness)**.** If the completion procedure on a set of equations E is run, different things can happen:

1. A state where no more inference rules are applicable is reached and E is not empty. $\Rightarrow$ Failure (try again with another ordering?)

2. A state where E is empty is reached and all critical pairs between the rules in the current R have been checked.

3. The procedure runs forever.

In order to treat these cases simultaneously some definitions are needed:

Definition 4.4.5 (Run)**.** A (finite or infinite) sequence $(E_0; R_0) \Rightarrow_{KBC} (E_1; R_1) \Rightarrow_{KBC} (E_2; R_2) \Rightarrow_{KBC} \ldots$ with $R_0 = \emptyset$ is called a *run* of the completion procedure with input E_0 and $\succ$. For a run, $E_\infty = \bigcup_{i\geq 0} E_i$ and $R_\infty = \bigcup_{i\geq 0} R_i$.

Definition 4.4.6 (Persistent Equations)**.** The sets of *persistent equations of rules* of the run are $E_* = \bigcup_{i\geq 0} \bigcap_{j\geq i} E_j$ and $R_* = \bigcup_{i\geq 0} \bigcap_{j\geq i} R_j$.

Note: If the run is finite and ends with E_n, R_n then $E_* = E_n$ and $R_* = R_n$.

Definition 4.4.7 (Fair Run)**.** A run is called *fair* if $\mathrm{CP}(R_*) \subseteq E_\infty$ (i.e., if every critical pair between persisting rules is computed at some step of the derivation).

Goal: Show: If a run is fair and E_* is empty then R_* is convergent and equivalent to E_0. In particular: If a run is fair and E_* is empty then $\approx_{E_0} = \approx_{E_\infty \cup R_\infty} = \leftrightarrow^*_{E_\infty \cup R_\infty} = \downarrow_{R_*}$.

From now on, $(E_0; R_0) \Rightarrow_{KBC} (E_1; R_1) \Rightarrow_{KBC} (E_2; R_2) \Rightarrow_{KBC} \ldots$ is a fair run and R_0 and E_* are empty.

A *proof* of $s \approx t$ in $E_\infty \cup R_\infty$ is a finite sequence $(s_0, \ldots, s_n)$ so that $s = s_0, t = s_n$ and for all $i \in \{1, \ldots, n\}$ it holds:

1. $s_{i-1} \leftrightarrow_{E_\infty} s_i$ or

2. $s_{i-1} \rightarrow_{R_\infty} s_i$ or

3. $s_{i-1} \;{}_{R_\infty}\!\!\leftarrow s_i$.

The pairs (s_{i-1}, s_i) are called *proof steps*. A proof is called a *rewrite proof* in R_* if there is a $k \in \{0, \ldots, n\}$ so that $s_{i-1} \rightarrow_{R_*} s_i$ for $1 \leq i \leq k$ and $s_{i-1} \;{}_{R_*}\!\!\leftarrow s_i$ for $k + 1 \leq i \leq n$.

Idea (Bachmair, Derschowitz, Hsiang): Define a well-founded ordering on proofs so that for every proof that is not a rewrite proof in R_* there is an equivalent smaller proof. Consequence: For every proof there is an equivalent rewrite proof in R_*. A *cost* $c(s_{i-1}, s_i)$ is associated with every proof step as follows:

1. If $s_{i-1} \leftrightarrow_{E_\infty} s_i$ then $c(s_{i-1}, s_i) = (\{s_{i-1}, s_i\}, -, -)$ where the first component is a multiset of terms and $-$ denotes an arbitrary (irrelevant) term.

2. If $s_{i-1} \to_{R_\infty} s_i$ using $l \to r$ then $c(s_{i-1}, s_i) = (\{s_{i-1}\}, l, s_i)$.

3. If $s_{i-1} \ {}_{R_\infty}\!\!\leftarrow s_i$ using $l \to r$ then $c(s_{i-1}, s_i) = (\{s_i\}, l, s_{i-1})$.

Proof steps are compared using the lexicographical combination of the multiset extension of the reduction ordering $\succ$, the encompassment ordering $\sqsupset$ and the reduction ordering $\succ$. The cost $c(P)$ of a proof P is the multiset of the cost of its proof steps. The *proof ordering* $\succ_C$ compares the cost of proofs using the multiset extension of the proof step ordering.

Lemma 4.4.8. $\succ_C$ is well-founded ordering.

Lemma 4.4.9. Let P be a proof in $E_\infty \cup R_\infty$. If P is not a rewrite proof in R_* then there exists an equivalent proof P' in $E_\infty \cup R_\infty$ so that $P \succ_C P'$.

Proof. If P is not a rewrite proof in R_* then it contains

1. a proof step that is in E_∞ or

2. a proof step that is in $R_\infty \backslash R_*$ or

3. a subproof $s_{i-1} \ {}_{R_*}\!\!\leftarrow s_i \to s_{i+1}$ (peak).

It is shown that in all three cases the proof step or subproof can be replaced by a smaller subproof:
Case 1.: A proof step using an equation $s \dot{\approx} t$ is in E_∞. This equation must be deleted during the run.

If $s \dot{\approx} t$ is deleted using *Orient*:

$$\ldots s_{i-1} \leftrightarrow_{E_\infty} s_i \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \to_{R_\infty} s_i \ldots$$

If $s \dot{\approx} t$ is deleted using *Delete*:

$$\ldots s_{i-1} \leftrightarrow_{E_\infty} s_{i-1} \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \ldots$$

If $s \dot{\approx} t$ is deleted using *Simplify-Eq*:

$$\ldots s_{i-1} \leftrightarrow_{E_\infty} s_i \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \to_{R_\infty} s' \leftrightarrow_{E_\infty} s_i \ldots$$

Case 2.: A proof step using a rule $s \to t$ is in $R_\infty \backslash R_*$. This rule must be deleted during the run.

If $s \to t$ is deleted using *R-Simplify-Rule*:

$$\ldots s_{i-1} \to_{R_\infty} s_i \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \to_{R_\infty} s' \ {}_{R_\infty}\!\!\leftarrow s_i \ldots$$

If $s \to t$ is deleted using *L-Simplify-Rule*:

$$\ldots s_{i-1} \to_{R_\infty} s_i \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \to_{R_\infty} s' \leftrightarrow_{E_\infty} s_i \ldots$$

Case 3.: A subproof has the form $s_{i-1} \ {}_{R_*}\!\!\leftarrow s_i \to_{R_*} s_{i+1}$.

If there is no overlap or a non-critical overlap:

$$\ldots s_{i-1} \; {}_{R_*}\!\!\leftarrow s_i \rightarrow_{R_*} s_{i+1} \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \rightarrow^*_{R_*} s' \; {}^*_{R_*}\!\!\leftarrow s_{i+1} \ldots$$

If there is a critical pair that has been added using *Deduce*:

$$\ldots s_{i-1} \; {}_{R_*}\!\!\leftarrow s_i \rightarrow_{R_*} s_{i+1} \ldots \quad \Longrightarrow \quad \ldots s_{i-1} \leftrightarrow_{E_\infty} s_{i+1} \ldots$$

In all cases, checking that the replacement subproof is smaller than the replaced subproof is routine. □

Theorem 4.4.10 (KBC Soundness)**.** Let $(E_0; R_0) \Rightarrow_{KBC} (E_1; R_1) \Rightarrow_{KBC} (E_2; R_2) \Rightarrow_{KBC} \ldots$ be a fair run and let R_0 and E_* be empty. Then

1. every proof in $E_\infty \cup R_\infty$ is equivalent to a rewrite proof in R_*,

2. R_* is equivalent to E_0 and

3. R_* is convergent.

Proof. 1. By well-founded induction on $\succ_C$ using the previous lemma.

2. Clearly, $\approx_{E_\infty \cup R_\infty} = \approx_{E_0}$. Since $R_* \subseteq R_\infty$ this yields $\approx_{R_*} \subseteq \approx_{E_\infty \cup R_\infty}$. On the other hand, by 1. it holds that $\approx_{E_\infty \cup R_\infty} \subseteq \approx_{R_*}$.

3. Since $\rightarrow_{R_*} \subseteq \succ$, R_* is terminating. By 1. it holds that R_* is confluent. □

Now using the proof of Theorem 3.15.2 termination of $\Rightarrow_{KBC}$ is undecidable.

Corollary 4.4.11 (KBC Termination)**.** Termination of $\Rightarrow_{KBC}$ is undecidable for some given finite set of equations E.

Proof. Using exactly the construction of Theorem 3.15.2 it remains to be shown that all computed critical pairs can be oriented. Critical pairs corresponding to the search for a PCP solution result in equations $f_R(u(x), v(y)) \approx f_R(u'(x), v'(y))$ or $f_R(u'(x), v'(x)) \approx c$. By chosing an appropriate ordering, all these equations can be oriented. Thus $\Rightarrow_{KBC}$ does not produce any unorientable equations. The rest follows from Theorem 3.15.2. □

4.4.1 Unfailing Completion

Classical completion: Try to transform a set E of equations into an equivalent convergent TRS. Fail, if an equation can neither be oriented nor deleted.

Unfailing completion (from Bachmair, Derschowitz and Plaisted [4]): If an equation cannot be oriented, *orientable instances* can still be used for rewriting. Note: If $\succ$ is total on ground terms, then every *ground instance* of an equation is trivial or can be oriented. The goal is to derive a *ground convergent* set of equations.