Formalizing Knuth-Bendix Orders and Knuth-Bendix Completion*

Christian Sternagel[1] and René Thiemann[2]

1 **School of Information Science**
Japan Advanced Institute of Science and Technology, Japan
c-sterna@jaist.ac.jp

2 **Institute of Computer Science**
University of Innsbruck, Austria
rene.thiemann@uibk.ac.at

Abstract

We present extensions of our *Isabelle Formalization of Rewriting* that cover two historically related concepts: the Knuth-Bendix order and the Knuth-Bendix completion procedure.

The former, besides being the first development of its kind in a proof assistant, is based on a generalized version of the Knuth-Bendix order. We compare our version to variants from the literature and show all properties required to certify termination proofs of TRSs.

The latter comprises the formalization of important facts that are related to completion, like *Birkhoff's theorem*, the *critical pair theorem*, and a soundness proof of completion, showing that the strict encompassment condition is superfluous for finite runs. As a result, we are able to certify completion proofs.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases certification, completion, confluence, termination

Category Regular Research Paper

1 Introduction

In their seminal paper [10], Knuth and Bendix introduced two important concepts: a procedure that allows us to solve certain instances of the word problem – (Knuth-Bendix) completion – as well as a specific order on terms that is useful to orient equations in the aforementioned procedure – the Knuth-Bendix order (or KBO, for short).

Our main contributions around KBO and completion are:

- There are several variants of KBO (e.g., incorporating quasi-precedences, infinite signatures, subterm coefficient functions, and generalized weight functions). In fact, not for all of these variants well-foundedness has been proven. We give the first well-foundedness proof for a variant of KBO that combines infinite signatures, quasi-precedences, and subterm-coefficient functions. Our proof is direct, and we illustrate why we could employ Kruskal's tree theorem only for a less powerful variant of KBO.
- We dropped the strict encompassment condition in the inference rules of completion and present a new proof which shows that the modified rules are still sound for finite runs.

* This research is supported by the Austrian Science Fund (FWF): P22767, J3202.

- All our results have been formalized with the proof assistant Isabelle/HOL [17] as part of our library IsaFoR [25]. That is, we developed the first mechanized proofs of KBO, Birkhoff's theorem [3], and completion.
- We extended our certifier CeTA [25] (whose soundness is formally verified) by the facility to check completion proofs. For accepted proofs, CeTA provides a decision procedure for the word problem.

Note that most termination techniques, besides KBO, that are used by modern termination tools were already formalized in IsaFoR before. Our extensions further increase the percentage of termination proofs (generated by some termination tool) that can be certified and also enables the certification of completion proofs. Moreover, in its current state, IsaFoR covers most of the theorems in the first seven chapters of [1] (only confluence beyond weak orthogonality and decision procedures for special cases like right-ground TRSs are missing), and hence significantly contributes to a full formalization of standard rewriting techniques for first-order rewrite systems.

Both IsaFoR and CeTA are freely available from `http://cl-informatik.uibk.ac.at/software/ceta/`. The results of this paper are available in IsaFoR/CeTA version 2.10.

The paper is organized as follows: In Section 2, we present some preliminaries on term rewriting. Our results on KBO and its formalization are discussed in Section 3. Afterwards, in Section 4, we present our formalization of the critical pair theorem and the algorithm for certifying confluence. These results are required in Section 5, where we prove soundness of completion and illustrate how we certify completion proofs. Our formalization of Birkhoff's theorem is the topic of Section 6, before we conclude and discuss related work in Section 7.

2 Preliminaries

We assume familiarity with rewriting, equational logic, and completion [1].

In the sequel, let R denote an arbitrary binary relation. We say that R is *well-founded* if and only if there is no infinite sequence a, s.t., $a_1\ R\ a_2\ R\ a_3\ R \cdots$.

We call R *almost-full* if and only if all infinite sequences a contain indices $j > i$, s.t., $a_j\ R\ a_i$. The same property is sometimes expressed by saying that all infinite sequences are *good*. (Note that every almost-full relation is reflexive.)

A binary relation $\succsim$ that is reflexive and transitive, is a *quasi-order* (or *preorder*). The *strict part* $\succ$ of $\succsim$ is defined by $x \succ y = x \succsim y \wedge y \not\succsim x$. Two elements x and y are *equivalent* if $x \succsim y$ and $y \succsim x$. A quasi-order whose strict part is well-founded is called a *quasi-precedence*. (The "quasi" part of the name, stems from the fact that different elements may be equivalent.)

A *partial order*, denoted $\succ$, is a binary relation that is transitive and irreflexive. Its reflexive closure is denoted by $\succeq$. If $\succsim$ is a quasi-order, then its strict part $\succ$ is a partial order. In that case, $\succsim\ =\ \succeq$ if and only if $\succsim$ is antisymmetric. (This is in contrast to alternative definitions of partial orders that start from a quasi-order $\succsim$ and additionally require antisymmetry; then we always have $\succsim = \succeq$.)

A *well-partial-order* (*well-quasi-order*) is a partial order $\succ$ (quasi-order $\succsim$), s.t., $\succeq$ ($\succsim$) is almost-full. For every well-partial-order $\succ$ (well-quasi-orders $\succsim$), $\succ$ ($\succ$) is well-founded.

A non-strict order $\succsim$ is *compatible* with a strict order $\succ$ if and only if $\succsim \circ \succ \circ \succsim \subseteq \succ$.

We say that a list $x_1, \ldots, x_m$ is a *superlist* of the list $y_1, \ldots, y_n$ (or equivalently that $y_1, \ldots, y_n$ is embedded in $x_1, \ldots, x_m$), written $x_1, \ldots, x_m \trianglerighteq^* y_1, \ldots, y_n$, if $y_1, \ldots, y_n$ is obtained from $x_1, \ldots, x_m$ by deleting elements (but not changing their order).

A *signature* $\mathcal{F}$ is a set of *function symbols* (denoted by f, g, ... for non-constants and a, b, c, ... for constants). The set of *terms* over a signature $\mathcal{F}$ and a set of variables $\mathcal{V}$ is

denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. We write $\mathcal{V}_{\mathsf{mul}}(t)$ for the multiset of variables occurring in a term t. Let p be a position in a term t. Then $t|_p$ denotes the subterm of t at position p and $t[s]_p$ denotes the result of replacing the subterm of t at position p by the term s. We write $s \trianglerighteq t$ ($s \triangleright t$) if t is a (proper) subterm of t. A *rewrite order* is an irreflexive and transitive order that is closed under contexts and closed under substitutions. For terms s and t from $\mathcal{T}(\mathcal{F},\mathcal{V})$ we call $s \approx t$ an *equation*. An *equational system* (ES) $\mathcal{E}$ is a set of equations.

Sometimes we orient an equation $s \approx t$, write $s \to t$, and call it a *rule*. Then for an ES $\mathcal{E}$ we denote by $\to_{\mathcal{E}}$ the smallest relation that contains $\mathcal{E}$ and is closed under contexts and substitutions. Let $\to$ be a relation. We write $(\to)^{-1}$ (or simply $\leftarrow$) for the inverse of $\to$, $\leftrightarrow$ for $\to \cup \leftarrow$, and $\to^*$ for the reflexive and transitive closure of $\to$.

An ES $\mathcal{E}$ is called a *term rewrite system* (TRS) if all equations are rules. A TRS $\mathcal{R}$ is *terminating* if $\to_{\mathcal{R}}$ is well-founded, and *(locally) confluent* if $({}_{\mathcal{R}}\!\leftarrow \cdot \to_{\mathcal{R}} \subseteq \to_{\mathcal{R}}^* \cdot {}_{\mathcal{R}}^*\!\leftarrow)$ ${}_{\mathcal{R}}^*\!\leftarrow \cdot \to_{\mathcal{R}}^* \subseteq \to_{\mathcal{R}}^* \cdot {}_{\mathcal{R}}^*\!\leftarrow$. A term s is in *($\mathcal{R}$-)normal form* if there is no term t with $s \to_{\mathcal{R}} t$. We write $s{\downarrow_{\mathcal{R}}}$ for an $\mathcal{R}$-normal form of a term s, i.e., some term t such that $s \to_{\mathcal{R}}^* t$ and t is in normal form. Terms s and t are *($\mathcal{R}$-)joinable* if $s \to_{\mathcal{R}}^* \cdot {}_{\mathcal{R}}^*\!\leftarrow t$, and *($\mathcal{E}$-)convertible* if $s \leftrightarrow_{\mathcal{E}}^* t$. A TRS $\mathcal{R}$ and an ES $\mathcal{E}$ are *equivalent* if $\leftrightarrow_{\mathcal{E}}^* = \leftrightarrow_{\mathcal{R}}^*$, i.e., their respective equational theories coincide.

We call (s,t) a *critical pair* of a TRS $\mathcal{R}$ if and only if there are two (not necessarily distinct) variable renamed rules $\ell_i \to r_i$, $i = 1,2$ (without common variables) and a position p in ℓ_1 such that θ is an mgu of $\ell_1|_p$ and ℓ_2, $\ell_1|_p$ is not a variable, $s = r_1\theta$, and $t = \ell_1\theta[r_2\theta]_p$.

An *$\mathcal{F}$-algebra* $\mathcal{A}$ consists of a non-empty set A (the *carrier*) and an *interpretation* $I : \mathcal{F} \to A^* \to A$. For each $\mathcal{F}$-algebra $\mathcal{A} = (A, I)$, and each *variable assignment* $\alpha : \mathcal{V} \to A$, we define the *term evaluation* $[\cdot]_\alpha^{\mathcal{A}} : \mathcal{T}(\mathcal{F},\mathcal{V}) \to A$ as $[x]_\alpha^{\mathcal{A}} = \alpha(x)$ and $[f(t_1,\dots,t_n)]_\alpha^{\mathcal{A}} = I(f)([t_1]_\alpha^{\mathcal{A}},\dots,[t_n]_\alpha^{\mathcal{A}})$. We drop $\mathcal{A}$ whenever it is clear from the context.

An $\mathcal{F}$-algebra $\mathcal{A}$ is a *model* of an equation $s \approx t$ if for each variable assignment α the equality $[s]_\alpha = [t]_\alpha$ is holds. If every $\mathcal{F}$-algebra that is a model of all equations in $\mathcal{E}$ also is a model of $s \approx t$, we write $\mathcal{E} \models s \approx t$ and say that $s \approx t$ *follows from* $\mathcal{E}$. The relation $\models$ is called *semantic entailment*.

3 Knuth-Bendix Order

We start by presenting our definition of KBO as it is formalized in IsaFoR in the files `KBO.thy` and `KBO_Impl.thy`. It comes in the form of two mutually recursive functions which define the strict order $\succ_{\mathsf{kbo}}$ and the compatible non-strict order $\succsim_{\mathsf{kbo}}$. The mutual dependency is created by the lexicographic extension of two compatible orders: each of $\succ_{\mathsf{kbo}}^{\mathsf{lex}}$ and $\succsim_{\mathsf{kbo}}^{\mathsf{lex}}$ require both $\succ_{\mathsf{kbo}}$ and $\succsim_{\mathsf{kbo}}$ in their definition.

▶ **Definition 3.1** (Knuth-Bendix Order). Let $\succsim$ be a quasi-precedence, $w_0 \in \mathbb{N} \setminus \{0\}$, and $w : \mathcal{F} \to \mathbb{N}$. Further, let w be *admissible*, i.e., $w(c) \geq w_0$ for all constants c and, if $w(f) = 0$ and f is unary, then f is of largest precedence, i.e., $f \succsim g$ for all g.

The weight function w is lifted to terms as follows.

$$w(x) = w_0$$
$$w(f(t_1,\dots,t_n)) = w(f) + \sum_{1 \leq i \leq n} w(t_i)$$

We define $s \succ_{\mathsf{kbo}} t$ if and only if $\mathcal{V}_{\mathsf{mul}}(s) \supseteq \mathcal{V}_{\mathsf{mul}}(t)$ and $w(s) \geq w(t)$

- and $w(s) > w(t)$ (weight)
- or $s = f(s_1, \ldots, s_m)$
  - and t is a variable (fun-var)
  - or $t = g(t_1, \ldots, t_n)$
    - and $f \succ g$ (prec)
    - or $f \succsim g$ and $(s_1, \ldots, s_m) \succ^{\mathsf{lex}}_{\mathsf{kbo}} (t_1, \ldots, t_n)$ (lex)

and $s \succsim_{\mathsf{kbo}} t$ if and only if $\mathcal{V}_{\mathsf{mul}}(s) \supseteq \mathcal{V}_{\mathsf{mul}}(t)$ and $w(s) \geq w(t)$

- and $w(s) > w(t)$ (weight)
- or s is a variable
  - and t is a variable (var-var)
  - or t is a constant and $\mathit{least}(t)$ (least)
- or $s = f(s_1, \ldots, s_m)$
  - and t is a variable (fun-var)
  - or $t = g(t_1, \ldots, t_n)$
    - and $f \succ g$ (prec)
    - or $f \succsim g$ and $(s_1, \ldots, s_m) \succsim^{\mathsf{lex}}_{\mathsf{kbo}} (t_1, \ldots, t_n)$ (lex-eq)

Here, *least* is a predicate that checks whether a constant c has weight w_0 and is of least precedence among all constants with weight w_0, i.e., $\forall d.\, w(d) = w_0 \longrightarrow d \succsim c$.

The lexicographic extension, $(s_1, \ldots, s_m) \succ^{\mathsf{lex}}_{\mathsf{kbo}} (t_1, \ldots, t_n)$ is defined by

$$(\exists i \leq \min(m,n).\, s_i \succ_{\mathsf{kbo}} t_i \wedge (\forall j < i.\, s_j \succsim_{\mathsf{kbo}} t_j)) \vee (m > n \wedge (\forall j \leq n.\, s_j \succsim_{\mathsf{kbo}} t_j)), \quad \text{(lex-def)}$$

and its non-strict part $(s_1, \ldots, s_m) \succsim^{\mathsf{lex}}_{\mathsf{kbo}} (t_1, \ldots, t_n)$ is defined similarly, except that $m > n$ is replaced by $m \geq n$.

Note that due to the condition $\mathcal{V}_{\mathsf{mul}}(s) \supseteq \mathcal{V}_{\mathsf{mul}}(t)$, we have that s and t are the same variable in (var-var), and t is a variable occurring in s in (fun-var).

Before we give details of our formalization of KBO, we relate our definition to other definitions of KBO as in [6, 10, 14, 19, 30].

Non-strict Part. In [6, 10, 14, 19, 30] only the strict order $\succ_{\mathsf{kbo}}$, without accompanying $\succsim_{\mathsf{kbo}}$, is defined. In lexicographic comparisons $=$ is used instead of $\succsim_{\mathsf{kbo}}$. However, the usage of $\succsim_{\mathsf{kbo}}$ leads to a more powerful variant of KBO, since $\succsim_{\mathsf{kbo}}$ is reflexive and thus, replacing $=$ by $\succsim_{\mathsf{kbo}}$ can only enlarge $\succ_{\mathsf{kbo}}$. Moreover, our (syntactic) definition of KBO can treat examples that have been used before to illustrate the power of non-syntactic definitions, where $s \succ_{\mathsf{kbo}} t$ for non-ground terms s and t is defined by $s\sigma \succ_{\mathsf{kbo}} t\sigma$ for all ground instances. Using our definition, we can orient the leading example $\mathsf{g}(x, \mathsf{a}, \mathsf{b}) \succ_{\mathsf{kbo}} \mathsf{g}(\mathsf{b}, \mathsf{b}, \mathsf{a})$ of [11] by choosing $w_0 = w(\mathsf{g}) = w(\mathsf{b}) = 1$, $w(\mathsf{a}) = 2$, and a precedence that makes all symbols equivalent.

Quasi-Precedence. In [6, 10, 14] no quasi-precedences are allowed, i.e., $f \succsim g$ is replaced by $f = g$ in (lex) and (lex-eq), which is clearly less powerful.

Lexicographic-Extension. Our definition of lexicographic extension allows comparisons where the decrease is due to a decrease in the lengths of the lists. Of course, this is only relevant for quasi-precedences [19, 30], since otherwise the argument lists always have the same length. Whereas this kind of definition is also used in [19], in [30], a lexicographic decrease is only possible if at some position $i \leq \min(m, n)$ there is a strict decrease.

However, this small change has an important impact: KBO as defined in [30] is neither closed under substitutions nor does it have the subterm property; taking $w_0 = w(\mathsf{a}) = 1$, $w(\mathsf{f}) = 0$ and a precedence where all symbols are equivalent, we have $\mathsf{f}(x) \succ_{\mathsf{kbo}} x$, but not $\mathsf{f}(\mathsf{a}) \succ_{\mathsf{kbo}} \mathsf{a}$, since $(\mathsf{a}) \not\succ^{\mathsf{lex}}_{\mathsf{kbo}} ()$ for the definition of lexicographic extension of [30].

Infinite Signatures. Our definition as well as [19, 30] admit arbitrary, possibly infinite, signatures, whereas in [6, 10, 14], only finite signatures are considered. However, neither [19] nor [30] mention a solution to the problem that Kruskal's tree theorem cannot be easily applied: the result that every rewrite order containing the subterm relation is well-founded, only holds for finite signatures. Hence, a well-foundedness proof for their versions of KBO using infinite signatures is missing.

And indeed, it is questionable whether our definition of KBO on infinite signatures has all the desired properties: the lexicographic extension as defined in (lex-def) does not preserve well-foundedness in general. Consider unbounded arities and take the lexicographic extension of the standard order on natural numbers. Then, we have $(1) >^{\mathsf{lex}} (0,1) >^{\mathsf{lex}} (0,0,1) \ldots$. If we bound arities for lexicographic comparisons, we do not achieve the subterm property: if $w(\mathsf{f}) = 0$ and all symbols have equal precedence, then $\mathsf{f}(\mathsf{g}(\mathsf{a},\ldots,\mathsf{a})) \succ_{\mathsf{kbo}} \mathsf{g}(\mathsf{a},\ldots,\mathsf{a})$ can only be shown by proving $(\mathsf{g}(\mathsf{a},\ldots,\mathsf{a})) \succ^{\mathsf{lex}}_{\mathsf{kbo}} (\mathsf{a},\ldots,\mathsf{a})$, i.e., for each g of arity n we require a successful comparison of lists of length 1 with lists of length n.

Status Functions. In contrast to [19], we do not integrate a status function which decides how to compare argument lists. Thus, our definition of KBO is incomparable to the one in [19]. The reason for this omission is simple: We are not aware of any termination tool that uses KBO with status. Nevertheless, we do not expect severe difficulties for adding a status function, as we already did this in a formalization of the recursive path order [24].

Ordinal Weights. Another generalization that we do not consider are weight functions over ordinal numbers [14].

To summarize, we are not aware of any proof of well-foundedness for a KBO variant that allows quasi-precedences and infinite signatures. Moreover, the subterm property and closure under substitutions look interesting as they are not satisfied by the definition in [30].

We now state all the desired properties that have been formalized for our version of KBO. Incidentally, we proved all these properties for a variant of KBO that also incorporates the subterm coefficient functions from [14]. That is, for each argument, we can specify how often it should be counted when computing weights and multisets of variables. E.g., for $\Psi(f) = [2,3]$ we compute the weight function w.r.t. Ψ as $w^\Psi(f(t_1,t_2)) = w(f) + 2w^\Psi(t_1) + 3w^\Psi(t_2)$, and the multiset of variables w.r.t. Ψ as $\mathcal{V}^\Psi_{\mathsf{mul}}(f(t_1,t_2)) = \mathcal{V}^\Psi_{\mathsf{mul}}(t_1) \cup \mathcal{V}^\Psi_{\mathsf{mul}}(t_1) \cup \mathcal{V}^\Psi_{\mathsf{mul}}(t_2) \cup \mathcal{V}^\Psi_{\mathsf{mul}}(t_2) \cup \mathcal{V}^\Psi_{\mathsf{mul}}(t_2)$. Since, for the proofs of the following properties, subterm coefficient functions did not pose any severe challenges, we omit them in the remainder to simplify our presentation.

▶ **Lemma 3.2** (Properties of $\succ_{\mathsf{kbo}}$ and $\succsim_{\mathsf{kbo}}$).

1. *Every term has a weight not less than w_0, i.e., $w(t) \geq w_0$ for all t.*
2. *Strict KBO is irreflexive and non-strict KBO reflexive, i.e., $t \not\succ_{\mathsf{kbo}} t$ and $t \succsim_{\mathsf{kbo}} t$ for all t.*
3. *The non-strict part is an extension of the strict part, i.e., $\succ_{\mathsf{kbo}} \subseteq \succsim_{\mathsf{kbo}}$.*
4. *KBO is closed under contexts, i.e., $s \mathrel{(\succsim_{\mathsf{kbo}})} t \longrightarrow C[s] \mathrel{(\succsim_{\mathsf{kbo}})} C[t]$ for all s, t, and C.*
5. *A weak decrease from an argument implies a strict decrease from the overall term, i.e., $s \succsim_{\mathsf{kbo}} t \longrightarrow f(\ldots,s,\ldots) \succ_{\mathsf{kbo}} t$ for all f, s, and t.*
6. *KBO has the* subterm property*, i.e., $\rhd \subseteq \succ_{\mathsf{kbo}}$. (We also have $\trianglerighteq \subseteq \succsim_{\mathsf{kbo}}$.)*

7. *Every term is larger than a least constant, i.e., least*$(c) \longrightarrow t \succsim_{\mathsf{kbo}} c$ *for all* c *and* t.
8. *KBO is closed under substitutions, i.e.,* $s \mathrel{\tilde{\succsim}}_{\mathsf{kbo}} t \longrightarrow s\sigma \mathrel{\tilde{\succsim}}_{\mathsf{kbo}} t\sigma$ *for all* s, t, *and* σ.
9. *All combinations of* $\succ_{\mathsf{kbo}}$ *and* $\succsim_{\mathsf{kbo}}$ *are transitive, i.e.,* $\succsim_{\mathsf{kbo}} \circ \succsim_{\mathsf{kbo}} \subseteq \succsim_{\mathsf{kbo}}$, $\succsim_{\mathsf{kbo}} \circ \succ_{\mathsf{kbo}} \subseteq \succ_{\mathsf{kbo}}$, $\succ_{\mathsf{kbo}} \circ \succsim_{\mathsf{kbo}} \subseteq \succ_{\mathsf{kbo}}$, *and* $\succ_{\mathsf{kbo}} \circ \succ_{\mathsf{kbo}} \subseteq \succ_{\mathsf{kbo}}$

Proof. We only present the proofs of the most interesting properties. All other proofs are provided in IsaFoR, file `KBO.thy`.

5. This is the main auxiliary fact for proving the subterm property. We prove it by induction on t. If t is a variable x, then we have $\mathcal{V}_{\mathsf{mul}}(f(\dots,s,\dots)) \supseteq \mathcal{V}_{\mathsf{mul}}(s) \supseteq \mathcal{V}_{\mathsf{mul}}(x)$, since $s \succsim_{\mathsf{kbo}} t = x$, and $w(f(\dots,s,\dots)) \geq w_0 = w(x)$ follows from **1**. Hence, $f(\dots,s,\dots) \succ_{\mathsf{kbo}} x$ by (fun-var). Otherwise, t is a term of the form $g(t_1,\dots,t_n)$ and $s \succsim_{\mathsf{kbo}} g(t_1,\dots,t_n)$. Using $s \succsim_{\mathsf{kbo}} g(t_1,\dots,t_n)$ we conclude that the conditions on the variables and weights are satisfied when comparing $f(\dots,s,\dots)$ and $g(t_1,\dots,t_n)$, e.g., $w(f(\dots,s,\dots)) \geq w(s) \geq w(g(t_1,\dots,t_n))$. If $w(f(\dots,s,\dots)) > w(g(t_1,\dots,t_n))$ or $f \succ g$ then we are done. Otherwise, $w(f(\dots,s,\dots)) = w(s) = w(g(t_1,\dots,t_n))$ which can only be the case when $w(f) = 0$ and f is unary, i.e., $f(\dots,s,\dots) = f(s)$. By admissibility, we conclude $f \succsim g$ and since $f \not\succ g$ we know $g \succsim f$, and hence, by transitivity of $\succsim$ and admissibility, we know that $g \succsim h$ for all symbols h. Then we consider the following cases:

- **case** $(n = 0)$. Then $f(s) \succ_{\mathsf{kbo}} g = g(t_1,\dots,t_n)$ by (lex), since $(s) \succ^{\mathsf{lex}}_{\mathsf{kbo}} ()$. This is the part of the proof where the length comparisons of lists in $\succ^{\mathsf{lex}}_{\mathsf{kbo}}$ play a crucial role.
- **case** $(n \geq 1)$. We know from $s \succsim_{\mathsf{kbo}} g(t_1,\dots,t_n)$ that s cannot be a variable. So let $s = h(s_1,\dots,s_m)$ for some h and $s_1,\dots,s_m$. Since $w(s) = w(g(t_1,\dots,t_n))$ and $g \succsim h$ we conclude $s_1,\dots,s_m \succsim^{\mathsf{lex}}_{\mathsf{kbo}} t_1,\dots,t_n$. Using $n \geq 1$ and **3** we know that $m \geq 1$ and $s_1 \succsim_{\mathsf{kbo}} t_1$. Using the induction hypothesis, we conclude $s = h(s_1,\dots,s_m) \succ_{\mathsf{kbo}} t_1$ and thus, $(s) \succ^{\mathsf{lex}}_{\mathsf{kbo}} (t_1,\dots,t_n)$ – and in this last comparison one can observe why we must not bound the lengths of the lists in (lex-def). In combination with $f \succsim g$ we conclude $f(s) \succ_{\mathsf{kbo}} g(t_1,\dots,t_n)$ by (lex).

6. From $s \rhd t$, we first obtain a non-empty context such that $s = C[t]$. Then $C[t] \succ_{\mathsf{kbo}} t$ is shown by induction on C, with the help of **2**, **3**, and **5**. Afterwards, the result for $s \trianglerighteq t$ follows by **2** and **3**.

8. We show $(s \succsim_{\mathsf{kbo}} t \longrightarrow s\sigma \succsim_{\mathsf{kbo}} t\sigma) \wedge (s \succ_{\mathsf{kbo}} t \longrightarrow s\sigma \succ_{\mathsf{kbo}} t\sigma)$ to ensure closure under substitutions. The proof works by induction on s followed by a case analysis on t. Most of the proof is straight-forward, just note that we require **7** (if the decrease is due to (least)) and **6** (if the decrease is due to (fun-var)). ◀

It remains to investigate well-foundedness of KBO. For finite signatures, this property follows from Kruskal's tree theorem, as $\succ_{\mathsf{kbo}}$ is a rewrite order which contains the strict subterm relation (and thus is a simplification order for finite signatures). Therefore, the definitions of KBO in [6, 10, 14] are well-founded.

As shown in [16], also for infinite signatures the tree theorem can be employed to show well-foundedness. However, this time the subterm property alone is not enough. Instead we need to regard homeomorphic embedding in the definition of simplification order. Given a relation $\succ$, *homeomorphic embedding* on terms, written $\succ_{\mathsf{he}}$, is defined inductively by the rules

$$\frac{t \in t_1,\dots,t_n}{f(t_1,\dots,t_n) \succ_{\mathsf{he}} t} \qquad \frac{s \succ_{\mathsf{he}} t \quad t \succ_{\mathsf{he}} u}{s \succ_{\mathsf{he}} u} \qquad \frac{s \succ_{\mathsf{he}} t}{C[s] \succ_{\mathsf{he}} C[t]} \qquad \frac{f \succ g \quad s_1,\dots,s_m \trianglerighteq^* t_1,\dots,t_n}{f(s_1,\dots,s_m) \succ_{\mathsf{he}} g(t_1,\dots,t_n)}$$

▶ **Theorem 3.3** (Kruskal's Tree Theorem). *If $\succ$ is a well-partial-order on the signature $\mathcal{F}$, then $\succ_{\mathsf{he}}$ is a well-partial-order on the set of ground terms $\mathcal{T}(\mathcal{F})$.*

Proof. A formalization of Theorem 3.3 is given in [21] and the proofs presented in [20]. ◀

▶ **Definition 3.4** (Simplification Order). A *simplification order* is a rewrite order R, s.t., $\succ_{\mathsf{he}}$ is contained in R, for some well-partial-order $\succ$.

Every simplification order is well-founded. Thus, instead of proving well-foundedness of R directly, it suffices to show that R is a rewrite order that contains $\succ_{\mathsf{he}}$ for some well-partial-order $\succ$.

We were able to show (and formalize in Isabelle/HOL) the following theorem.

▶ **Theorem 3.5.** *If $\succ$ is a well-partial-order, then $\succ_{\mathsf{kbo}}$ is a simplification order (and thus well-founded).*

Proof. The proof of [16, Theorem 7.10] works also for our definition. ◀

For definitions of KBO that are not based on a quasi-precedence (e.g., [16]), the above theorem is enough to show that any KBO $\succ_{\mathsf{kbo}}$ over a well-founded partial order $\succ$ is a simplification order. This can be seen as follows: every well-founded partial order $\succ$ can be extended to a total well-order $\succ'$ (i.e., a well-founded partial order that satisfies $\forall x\, y.\, x = y \vee x \succ' y \vee y \succ' x$); moreover, every total well-order is a well-partial-order.

▶ **Lemma 3.6.** *Every total well-order is a well-partial-order.*

Proof. Let $\succ$ be a total well-order. Assume that it is not a well-partial-order (i.e., not almost-full). Then there is an infinite sequence a such that $a_j \not\succ a_i$ for all $j > i$. By totality, we have $a_i \succ a_j$ for all $j > i$ and thus $a_1 \succ a_2 \succ a_3 \succ \cdots$, contradicting well-foundedness. ◀

Furthermore, KBO is monotone w.r.t. the given precedence (i.e., $\succ \subseteq \succ'$ implies $\succ_{\mathsf{kbo}} \subseteq \succ'_{\mathsf{kbo}}$); thus, we can apply Theorem 3.5 to obtain well-foundedness of $\succ'_{\mathsf{kbo}}$ and then, by $\succ_{\mathsf{kbo}} \subseteq \succ'_{\mathsf{kbo}}$, conclude well-foundedness of $\succ_{\mathsf{kbo}}$.

Unfortunately, the same procedure is not applicable with our definition of KBO, since it does not seem to be monotone w.r.t. the given quasi-precedence, unless $\succsim$ is antisymmetric and thus $\succsim = \succeq$ (and we are back at not having a quasi-precedence).

Unfortunately, that means that Theorem 3.5 is not relevant for termination tools, since those typically provide some quasi-precedence, but do not care, whether its strict part is a well-partial-order.

Besides the problems with quasi-precedences, further note that the proof of Theorem 3.5 does not apply in the presence of subterm coefficient functions. Therefore, we also formalized a direct well-foundedness proof which has some similarities to [16] and integrates all extensions: arbitrary signatures, quasi-precedences, and subterm coefficient functions (again, the last extension is only visible in IsaFoR).

▶ **Theorem 3.7** (Well-Foundedness). *KBO is well-founded, i.e.,* $\mathrm{SN}(\succ_{\mathsf{kbo}})$.

Proof. Let *strong normalization of a term t*, written $\mathrm{SN}(t)$, be the property that there is no infinite sequence $t \succ_{\mathsf{kbo}} t_1 \succ_{\mathsf{kbo}} t_2 \succ_{\mathsf{kbo}} \ldots$. Then it suffixes to show that for all terms s, we have $\mathrm{SN}(s)$. Let s be an arbitrary but fixed term. We perform four inductions, labeled (I), (II), (III), and (IV).

The outermost induction (I) is on s. The variable case is easy as variables are normal forms w.r.t. $\succ_{\mathsf{kbo}}$. For the non-variable case we have to show $\mathrm{SN}(s_1, \ldots, s_m) \longrightarrow$

$\mathrm{SN}(f(s_1,\ldots,s_m))$ where $\mathrm{SN}(s_1,\ldots,s_m)$ abbreviates $\forall s \in s_1,\ldots,s_m.\,\mathrm{SN}(s)$. This property is proven by induction (II) on f and $s_1,\ldots,s_m$ where the induction relation compares the pairs $(w(f(s_1,\ldots,s_m)),f)$ lexicographically, using the standard order $>$ on natural numbers for the first component, and $\succ$ to compare the function symbols in the second component.

In principle, we would like to add a third component to the lexicographic comparisons, where we compare the argument lists $s_1,\ldots,s_m$ by $\succ_{\mathsf{kbo}}^{\mathsf{lex}}$. But $\succ_{\mathsf{kbo}}^{\mathsf{lex}}$ is not necessarily well-founded, as we did not bound the lengths of these lists. We bound the lengths in a further induction (III). To be more precise, once, we have to prove the property of induction (II) for some given f and $s_1,\ldots,s_m$, we define $\succ_{\mathsf{kbo}}^{\mathsf{lexb}}$ as lexicographic extension of $\succ_{\mathsf{kbo}}$ where the lengths of all lists are bounded by $w(f(s_1,\ldots,s_m))$ and where it is assumed that all elements in the lists are already strongly normalizing. Then indeed, $\succ_{\mathsf{kbo}}^{\mathsf{lexb}}$ is strongly normalizing, so we can prove the following property by induction (III) on $t_1,\ldots,t_n$ for all g w.r.t. the induction relation $\succ_{\mathsf{kbo}}^{\mathsf{lexb}}$.

$$f \succsim g \longrightarrow w(f(s_1,\ldots,s_m)) \geq w(g(t_1,\ldots,t_n)) \longrightarrow \mathrm{SN}(t_1,\ldots,t_n) \longrightarrow \mathrm{SN}(g(t_1,\ldots,t_n)) \quad (\star)$$

Clearly, once this property is proven, we can finish induction (II) by choosing $g = f$ and $t_1,\ldots,t_n = s_1,\ldots,s_m$.

To prove $(\star)$, we assume the preconditions of $(\star)$ for some g and $t_1,\ldots,t_n$ and have to show $\mathrm{SN}(g(t_1,\ldots,t_n))$. To this end, we prove that all successors of $g(t_1,\ldots,t_n)$ are strongly normalizing, i.e., for all u: $g(t_1,\ldots,t_n) \succ_{\mathsf{kbo}} u \longrightarrow \mathrm{SN}(u)$. This property we show by induction (IV) which is a structural induction on u.

There is nothing to show if u is a variable, so let $u = h(u_1,\ldots,u_k)$. Since $g(t_1,\ldots,t_n) \succ_{\mathsf{kbo}} h(u_1,\ldots,u_k)$ we also know that $g(t_1,\ldots,t_n)$ is larger than every element of $u_1,\ldots,u_k$ using the subterm property and transitivity. Hence, using induction hypothesis (IV) we know $\mathrm{SN}(u_1,\ldots,u_k)$.

Now, if $(w(f(s_1,\ldots,s_m)),f)$ is larger than $(w(h(u_1,\ldots,u_k),h)$ w.r.t. induction relation (II), then we are done using the induction hypothesis (II). Otherwise, in combination with the preconditions of $(\star)$ and $g(t_1,\ldots,t_n) \succ_{\mathsf{kbo}} h(u_1,\ldots,u_k)$ we conclude $w(f(s_1,\ldots,s_m)) \geq w(h(u_1,\ldots,u_k))$, $f \succsim h$, and $(t_1,\ldots,t_n) \succ_{\mathsf{kbo}}^{\mathsf{lex}} (u_1,\ldots,u_k)$. Since both $w(h(u_1,\ldots,u_k))$ and $w(g(t_1,\ldots,t_n))$ are smaller than $w(f(s_1,\ldots,s_m))$ in particular this shows, that the lengths of $u_1,\ldots,u_k$ and $t_1,\ldots,t_n$ are smaller than $w(f(s_1,\ldots,s_m))$. As additionally both $\mathrm{SN}(u_1,\ldots,u_k)$ and $\mathrm{SN}(t_1,\ldots,t_n)$ we can derive $(t_1,\ldots,t_n) \succ_{\mathsf{kbo}}^{\mathsf{lexb}} (u_1,\ldots,u_k)$ from $(t_1,\ldots,t_n) \succ_{\mathsf{kbo}}^{\mathsf{lex}} (u_1,\ldots,u_k)$. But then induction hypothesis (III) can be applied to derive the desired property $\mathrm{SN}(h(u_1,\ldots,u_k))$. ◀

4 Confluence

In this section we first discuss challenges in the formalization of the critical pair theorem (which is essential to prove soundness of completion), and afterwards present our algorithm to actually certify confluence proofs (which is reused later to certify completion proofs). The corresponding formalizations are provided in files `Critical_Pairs.thy` and `Critical_Pairs_Impl.thy`.

4.1 Critical Pair Theorem

Recall that in the context of completion, we are interested in terminating and confluent TRSs. Using our formalization of KBO (and other termination criteria that are available in IsaFoR), we can already certify termination proofs. Hence, we only require a criterion

to check for confluence. By Newman's lemma (which is trivially formalized), it suffices to analyze local confluence. Here, the key technique is the critical pair theorem of Huet [9] – making a result by Knuth and Bendix [10] explicit. It states that $\mathcal{R}$ is locally confluent if and only if all critical pairs of $\mathcal{R}$ are joinable.

By definition, every critical pair (s, t), gives rise to a peak

$$s = r_1\theta \;{}_{\mathcal{R}}\!\leftarrow \ell_1\theta = \ell_1\theta[\ell_1|_p\theta]_p = \ell_1\theta[\ell_2\theta]_p \rightarrow_{\mathcal{R}} \ell_1\theta[r_2\theta]_p = t.$$

Hence, for local confluence all critical pairs must be joinable. Huet, Knuth, and Bendix have shown that for local confluence it is indeed sufficient to show joinability of critical pairs: whenever $t_1 \;{}_{\mathcal{R}}\!\leftarrow s \rightarrow_{\mathcal{R}} t_2$ where t_1 and t_2 are not joinable, then there are C, σ, t_1', and t_2' such that $t_1 = C[t_1'\sigma]$, $t_2 = C[t_2'\sigma]$, and (t_1', t_2') or (t_2', t_1') is a critical pair of $\mathcal{R}$.

In order to formalize the notion of critical pairs in Isabelle, we need a unification algorithm. To this end, we have formalized the algorithm of [1, Figure 4.5]. In contrast to [1] we directly proved all properties (termination, soundness, and completeness) for this concrete algorithm (as opposed to the abstract algorithm of [1, Chapter 4.6], which transforms unification problems). As a result, the termination argument is a bit easier (it does not rely upon the notion of solved variable), whereas the soundness proof becomes a bit harder (as we had to prove that every result of the unification algorithm is in solved form), cf. `Substitution.thy`.

Having unification, a problem when formalizing the definition of critical pairs occurred. Notice that in IsaFoR, terms are polymorphic in the set of function symbols (all elements of type α) and the set of variables $\mathcal{V}$ (all elements of type β). Hence, to prove the critical pair theorem in a generic way we cannot assume anything on the set of variables. In particular, $\mathcal{V}$ might be finite and thus, we might not even have enough variables for building the critical pairs (as their definition requires variable renamed rules) and thus cannot even formulate the critical pair theorem for arbitrary $\mathcal{V}$.[1]

Since the definition of critical pairs should be executable, we actually do not only want that there are enough variables for the renaming, but we want some executable algorithm which actually performs such a renaming. Therefore, we did not formalize the critical pair theorem for arbitrary infinite sets $\mathcal{V}$, but for strings. For this type of variables it was easy to define an efficient and executable renaming algorithm: it just prefixes all variables by x in the one rule, and by y in the other rule.

▶ **Theorem 4.1.** *A TRS $\mathcal{R}$ over $\mathcal{T}(\mathcal{F}, String)$ is locally confluent if and only if all critical pairs of $\mathcal{R}$ are joinable.*

Note that the theorem does not require any variable-condition for $\mathcal{R}$. Hence, $\mathcal{R}$ may, e.g., contain left-hand sides which are variables or free variables in right-hand sides.

Further note that there are prior formalizations of the critical pair theorem in other theorem provers. The first one is described in [18] using ACL2, and another one is presented in [7] using PVS. Our own formalization is similar to the one in [7], as both are based on the higher-order structure of Huet's proof. However, there are some differences to [7], e.g., in the definition of critical pairs. Whereas in [7] arbitrary renaming substitutions and most general unifiers are allowed for building critical pairs, our definition uses specific renaming and unification algorithms. As a consequence, for a finite TRS we only get finitely many critical pairs, whereas in [7] one usually has to consider infinitely many critical pairs (which arise from using different variable names in the substitutions).

[1] Also other well-known results are problematic if there are not enough variables. For example, during our formalization we figured out that infinitely many variables are also essential for Toyama's modularity result on confluence [27].

4.2 Certifying Joinability

To actually certify local confluence proofs, we have to ensure that all critical pairs are joinable. There are two possibilities.

- The certificate provides the joining rewrite sequences for each critical pair.
- The certifier itself computes joining sequences.

The first possibility has the advantage that it is easy to formalize. However, it will make the certificates bulky. Even more important, this approach cannot be used to prove non-confluence, since it does not allow to certify that a critical pair is not joinable.

Both disadvantages are resolved in the second approach, where due to termination, for each critical pair (s, t) we just have to check $s{\downarrow_{\mathcal{R}}} = t{\downarrow_{\mathcal{R}}}$. Hence, in IsaFoR we integrated an automatic check which computes normal forms.

This sounds like a trivial task, where we just define:

$$\textit{compute-NF}\ \mathcal{R}\ s \equiv \mathsf{if}\ \{t \mid s \to_{\mathcal{R}} t\} = \varnothing\ \mathsf{then}\ t\ \mathsf{else}\ \textit{compute-NF}\ \mathcal{R}\ (\mathsf{SOME}\ t.\, s \to_{\mathcal{R}} t)$$

The problem is that *compute-NF* is not a total function (since the parameter $\mathcal{R}$ might be a nonterminating TRS). Hence, the function package of Isabelle [13] accepts *compute-NF* only as partial function, where the defining equation of *compute-NF* is guarded by a condition (if the input belongs to the domain, i.e., $\mathcal{R}$ is terminating, then the equation is valid). However, conditional equations prevent to use the code generation facility of Isabelle [8]. Thus, the above definition is not suitable for our setting.

The solution is to use the partial function command of [12]. It allows to define partial functions which can be code generated, if the functions have a specific shape. In our case, we just have to wrap the result of *compute-NF* in an option-type (where *None* represents nontermination) and everything works fine. For the generated code this means that a call to *compute-NF* may diverge in general. However, we invoke *compute-NF* only if termination of $\mathcal{R}$ has already been ensured.

5 Knuth-Bendix Completion

As in the previous section on confluence, we start with the formalization of the general soundness theorem of completion, and afterwards discuss the certification algorithm for checking completion proofs. The formalizations are available in files `Completion.thy` and `Check_Completion_Proof.thy`.

5.1 Soundness of Completion

Having formalized the critical pair theorem, we proceed to also formalize soundness of Knuth-Bendix completion. Since in the end, we are only able to certify finite completion runs (as the input cannot be infinite), we only formalized soundness for finite runs. To this end, we used the completion rules from [1] as illustrated in Figure 1. However, the restriction to finite runs allowed us to drop the requirement of strict encompassment in the collapse-rule, cf. Theorem 5.2.

We write $(\mathcal{E}_i, \mathcal{R}_i) \rightsquigarrow (\mathcal{E}_{i+1}, \mathcal{R}_{i+1})$ for the application of an arbitrary inference rule to the pair $(\mathcal{E}_i, \mathcal{R}_i)$ resulting in $(\mathcal{E}_{i+1}, \mathcal{R}_{i+1})$.

▶ **Definition 5.1.** A completion *run* for $\mathcal{E}$ is a finite sequence $(\mathcal{E}_0, \mathcal{R}_0) \rightsquigarrow^n (\mathcal{E}_n, \mathcal{R}_n)$ of rule applications, where $\mathcal{E}_0 = \mathcal{E}$ and $\mathcal{R}_0 = \varnothing$. A run is *successful* if $\mathcal{E}_n = \varnothing$ and every critical pair of $\mathcal{R}_n$ is contained in $\bigcup_{i \le n} \mathcal{E}_i$ or is joinable by $\mathcal{R}_n$.

$$\frac{(\mathcal{E},\mathcal{R})}{(\mathcal{E}\cup\{s\approx t\},\mathcal{R})}\ (\mathsf{deduce})\quad \text{if } s\ {}_{\mathcal{R}}\!\leftarrow u\rightarrow_{\mathcal{R}} t \qquad \frac{(\mathcal{E}\cup\{s\approx s\},\mathcal{R})}{(\mathcal{E},\mathcal{R})}\ (\mathsf{delete})$$

$$\frac{(\mathcal{E}\cup\{s\approx t\},\mathcal{R})}{(\mathcal{E},\mathcal{R}\cup\{s\to t\})}\ (\mathsf{orient_l})\quad \text{if } s\succ t \qquad \frac{(\mathcal{E}\cup\{s\approx t\},\mathcal{R})}{(\mathcal{E},\mathcal{R}\cup\{t\to s\})}\ (\mathsf{orient_r})\quad \text{if } t\succ s$$

$$\frac{(\mathcal{E}\cup\{s\approx t\},\mathcal{R})}{(\mathcal{E}\cup\{u\approx t\},\mathcal{R})}\ (\mathsf{simplify_l})\quad \text{if } s\rightarrow_{\mathcal{R}} u \qquad \frac{(\mathcal{E}\cup\{s\approx t\},\mathcal{R})}{(\mathcal{E}\cup\{s\approx u\},\mathcal{R})}\ (\mathsf{simplify_r})\ \text{if } t\rightarrow_{\mathcal{R}} u$$

$$\frac{(\mathcal{E},\mathcal{R}\cup\{s\to t\})}{(\mathcal{E},\mathcal{R}\cup\{s\to u\})}\ (\mathsf{compose})\ \text{if } t\rightarrow_{\mathcal{R}} u \qquad \frac{(\mathcal{E},\mathcal{R}\cup\{s\to t\})}{(\mathcal{E}\cup\{u\approx t\},\mathcal{R})}\ (\mathsf{collapse})\ \text{if } s\rightarrow_{\mathcal{R}} u$$

Figure 1 The inference rules of completion.

If $(\mathcal{E},\varnothing)\rightsquigarrow^n(\varnothing,\mathcal{R}_n)$ is a successful run for $\mathcal{E}$, then $\mathcal{R}_n$ is confluent, terminating, and equivalent to $\mathcal{E}$ (cf. Theorem 5.2). Note that the requirement on critical pairs for a successful run can be replaced by one of the two weaker criteria: either local confluence of $\mathcal{R}_n$ (all critical pairs are joinable) or generation of all critical pairs of $\mathcal{R}_n$ during the run (all critical pairs are contained in $\bigcup_{i\le n}\mathcal{E}_i$).

We have formally proven soundness of completion in IsaFoR.

▶ **Theorem 5.2** (Soundness of Completion). *If $(\mathcal{E},\varnothing)\rightsquigarrow^n(\varnothing,\mathcal{R})$ is a successful run of completion then $\mathcal{R}$ is confluent, terminating, and equivalent to $\mathcal{E}$.*

Proof. Let $(\mathcal{E},\varnothing)\rightsquigarrow^n(\varnothing,\mathcal{R})$ be a successful run. Then there are $\mathcal{E}_i$ and $\mathcal{R}_i$ with $(\mathcal{E}_i,\mathcal{R}_i)\rightsquigarrow(\mathcal{E}_{i+1},\mathcal{R}_{i+1})$ for all $0\le i<n$, $\mathcal{E}_0=\mathcal{E}$, $\mathcal{R}_0=\varnothing$, $\mathcal{E}_n=\varnothing$, and $\mathcal{R}_n=\mathcal{R}$. In total, we have to prove that $\mathcal{R}$ and $\mathcal{E}$ are equivalent, that $\mathcal{R}$ is terminating, and that $\mathcal{R}$ is confluent. Here, for equivalence and termination we just formalized the proofs which are provided in [1] since they do not depend on the encompassment condition in the original collapse rule.

The real interesting part is to prove confluence of $\mathcal{R}$, where we use proof orders [2], which are also utilized in [1]. More specifically, we formalized the proof using the same structure as in the proof of [1, Lemma 7.3.4] and just list the most important differences.

In our case, the persistent rules $\mathcal{R}_\omega$ are $\mathcal{R}_n$, the set $\mathcal{R}_\infty$ is $\bigcup_{i\le n}\mathcal{R}_i$, the persistent equations are $\mathcal{E}_\omega=\mathcal{E}_n=\varnothing$, and $\mathcal{E}_\infty=\bigcup_{i\le n}\mathcal{E}_i$.

The cost of a proof step (s_{i-1},s_i) is just a pair and not a triple as in [1], where we keep the first component of the original triple, but drop the part that is concerned with strict encompassment. Instead, we use another well-founded order on rules. To this end, we define the latest occurrence function as $o(\cdot)$ where $o(\ell\to r)=\max\{i\mid i\le n,\ell\to r\in\mathcal{R}_i\}$. Then we define the cost $c(\cdot)$ of a proof step as follows: if $s_{i-1}\leftrightarrow_{\mathcal{E}_\infty}s_i$ then $c(s_{i-1},s_i)=(\{s_{i-1},s_i\},-)$ where the first component is a multiset of terms and the second component is irrelevant; if $s_{i-1}\overset{\ell\to r}{\rightarrow}_{\mathcal{R}_\infty}s_i$ then $c(s_{i-1},s_i)=(\{s_{i-1}\},n-o(\ell\to r))$; if $s_{i-1}\overset{\ell\to r}{\leftarrow}_{\mathcal{R}_\infty}s_i$ then $c(s_{i-1},s_i)=(\{s_i\},n-o(\ell\to r))$.

As in the original proof, these pairs are compared lexicographically, where the first component is compared via the multiset extension of the reduction order $\succ$, and the second using the standard greater-than order on natural numbers. Hence, the order on the cost of proof steps is well-founded. Finally, as in the original proof, the cost of a conversion proof $s\leftrightarrow^*_{\mathcal{E}_\infty\cup\mathcal{R}_\infty}t$ is the multiset of costs of each single proof step, and these conversion costs are compared via the multiset extension of the cost of a single proof step. This defines the relation $\succ_{\mathcal{C}}$ on conversion proofs which is well-founded.

In [1, Lemma 7.3.4] it is stated that whenever there is a conversion $s \leftrightarrow^*_{\mathcal{E}_\infty \cup \mathcal{R}_\infty} t$ which is not a rewrite proof of $\mathcal{R}_\omega$, i.e., a conversion of the form $s \to^*_{\mathcal{R}_\omega} \cdot {}^*_{\mathcal{R}_\omega}\!\leftarrow t$, then there is another conversion between s and t with less cost. This is performed via a large case analysis (cases 1.1, 1.2, 1.3, 2.1, 2.2, 3.1, and 3.2). This proof can almost completely be formalized in our setting with the new collapse-rule and the different cost function. The reason is that in cases 1.1, 1.2, 1.3, 3.1, and 3.3, collapse does not occur and moreover, there is a decrease of cost due to the first component in the lexicographic combination – and as we have the identical first component, we obtain a decrease of cost by the same reasoning.

Hence, it remains to investigate cases 2.1 and 2.2 where the precondition of both cases is that $s_{i-1} \to_{\mathcal{R}_\infty} s_i$ at position p using a rule $s \to t \in \mathcal{R}_\infty - \mathcal{R}_\omega$ and substitution σ. Let $k = o(s \to t)$. Thus, $c(s_{i-1}, s_i) = (\{s_{i-1}\}, n - k)$. By definition of o and the fact that $s \to t \notin \mathcal{R}_\omega = \mathcal{R}_n$, we conclude $k < n$. Hence, in the step from $(\mathcal{E}_k, \mathcal{R}_k)$ to $(\mathcal{E}_{k+1}, \mathcal{R}_{k+1})$ the rule $s \to t$ is removed from $\mathcal{R}_k$, so let $\mathcal{R}_k = \mathcal{R}' \cup \{s \to t\}$.

Now we consider both cases how $s \to t$ has been removed in the k-th step.

If we used compose (case 2.1) then $s \to t$ has been replaced by $s \to u$ since $t \overset{\ell \to r}{\to}_{\mathcal{R}'} u$ for some $\ell \to r \in \mathcal{R}'$. As $s \to u \in \mathcal{R}_{k+1}$ we know that $o(s \to u) > k$ and thus, $n - k > n - o(s \to u)$. We follow the original proof by replacing the one proof step $s_{i-1} \to_{\mathcal{R}_\infty} s_i$ by the two proof steps $s_{i-1} = s_{i-1}[s\sigma]_p \overset{s \to u}{\to}_{\mathcal{R}_\infty} s_{i-1}[u\sigma]_p \overset{\ell \to r}{\leftarrow}_{\mathcal{R}'} s_{i-1}[t\sigma]_p = s_i$. Since the cost of the original proof step $c(s_{i-1}, s_i) = (\{s_{i-1}\}, n - k)$ is strictly larger than the cost of both new proof steps – $c(s_{i-1}, s_{i-1}[u\sigma]_p) = (\{s_{i-1}\}, n - o(s \to u))$ yields a decrease in the second component and $c(s_{i-1}[u\sigma]_p, s_i) = (\{s_i\}, n - o(\ell \to r))$ yields a decrease in the first component – we have constructed a conversion with smaller cost.

Similarly, if we used collapse (case 2.2) then $s \to t$ has been replaced by $u \approx t$ since $s \overset{\ell \to r}{\to}_{\mathcal{R}'} u$ for some $\ell \to r \in \mathcal{R}'$. As $\ell \to r \in \mathcal{R}' = \mathcal{R}_{k+1}$ we know that $o(\ell \to r) > k$ and thus, $n - k > n - o(\ell \to r)$. We follow the original proof by replacing the one proof step $s_{i-1} \to_{\mathcal{R}_\infty} s_i$ by the two proof steps $s_{i-1} = s_{i-1}[s\sigma]_p \overset{\ell \to r}{\to}_{\mathcal{R}_\infty} s_{i-1}[u\sigma]_p \overset{u \approx t}{\leftrightarrow}_{\mathcal{E}_\infty} s_{i-1}[t\sigma]_p = s_i$. Since the cost of the original proof step $c(s_{i-1}, s_i) = (\{s_{i-1}\}, n-k)$ is strictly larger than the cost of both new proof steps – $c(s_{i-1}, s_{i-1}[u\sigma]_p) = (\{s_{i-1}\}, n - o(\ell \to r))$ yields a decrease in the second component and $c(s_{i-1}[u\sigma]_p, s_i) = (\{s_{i-1}[u\sigma], s_i\}, -)$ yields a decrease in the first component – we have constructed a conversion with smaller cost.

In total, also for our completion rules, we can prove the major lemma that whenever there is a conversion $s \leftrightarrow^*_{\mathcal{E}_\infty \cup \mathcal{R}_\infty} t$ which is not a rewrite proof of $\mathcal{R}_\omega$, then there is another conversion between s and t with less cost w.r.t. $\succ_{\mathcal{C}}$. Using this result it is straightforward to show confluence of $\mathcal{R}_n = \mathcal{R}_\omega$.

Since $\mathcal{R}_n$ is terminating it suffices to prove local confluence. By the critical pair theorem we only have to prove joinability of all critical pairs. So, let (s, t) be some critical pair of $\mathcal{R}_n$. If it is not joinable, then by the definition of a successful run we know that $(s, t) \in \mathcal{E}_\infty$ and thus, $s \leftrightarrow^*_{\mathcal{E}_\infty \cup \mathcal{R}_\infty} t$. Since $\succ_{\mathcal{C}}$ is well-founded and by using the major lemma we know that there also is a conversion of s and t which is a rewrite proof of $\mathcal{R}_n$. But this is the same as demanding that s and t are joinable via $\mathcal{R}_n$. ◀

5.2 Certification of Completion Proofs

For checking completion proofs, although Theorem 5.2 has been formalized, it is not checked whether the completion rules are applied correctly. This allows us to also certify proofs from completion tools which are based on other inference rules than those in Figure 1. Instead it is just verified whether the result of the completion procedure is a confluent and terminating TRS that is equivalent to the initial set of equations. How to ensure confluence was already

described in Section 4.2, and for termination we use the existing machinery of IsaFoR [26].

It remains to check that $\mathcal{R}$ is equivalent to $\mathcal{E}$ which is done by showing $\ell \leftrightarrow^*_{\mathcal{E}} r$ for all $\ell \to r \in \mathcal{R}$ and $s \leftrightarrow^*_{\mathcal{R}} t$ for all $s \approx t \in \mathcal{E}$.

For the latter, since we already know that $\mathcal{R}$ is confluent and terminating, we just compare the normal forms of s and t in the same way as we check joinability of critical pairs in Section 4.2.

For the former, we demand a step-wise conversion from ℓ to r as input. Such a conversion can be obtained by storing additional information during the completion run, e.g., a history of all applied inference rules.

Given some successful completion proof from $\mathcal{E}$ to $\mathcal{R}$, we can of course also easily implement the decision procedure for $s \leftrightarrow^*_{\mathcal{E}} t$. In combination with a formalization of Birkhoff's theorem in the upcoming section, we obtain a formalized decision procedure for $\mathcal{E} \models s \approx t$. To be more precise, in the file `Check_Equational_Proof.thy` we provide two means to prove and disprove $\mathcal{E} \models s \approx t$, namely by checking a completion proof and afterwards check whether the normal forms of s and t coincide or not. For ensuring $\mathcal{E} \models s \approx t$ one can alternatively provide a derivation $s \leftrightarrow^*_{\mathcal{E}} t$, or a proof tree using the inference rules of equational logic.

6 Birkhoff's Theorem

Birkhoff's theorem [3] states that semantic entailment and convertibility are equivalent: $\mathcal{E} \models s \approx t$ if and only if $s \leftrightarrow^*_{\mathcal{E}} t$. To show this theorem we had to formalize semantic entailment. Already at this point we encountered the first problem. Whereas it was easy to formalize term evaluation and whether an $\mathcal{F}$-algebra (A, I) is a model of an equation (written $I \models s \approx t$), a problem occurred in the definition of semantic entailment.

Before describing the problem, we show how $\mathcal{F}$-algebras are formalized in IsaFoR, within the file `Equational_Reasoning.thy`. As shown in Section 2 we consider $\mathcal{F}$-algebras consisting of a non-empty carrier A and an interpretation I. In Isabelle, a (non-dependently) typed higher-order logic, we represent such interpretations by functions of type $\alpha \Rightarrow \gamma\ list \Rightarrow \gamma$ (where lower-case Greek letters denote type variables and $\gamma\ list$ is the type of finite lists having elements of type γ) which corresponds to $\mathcal{F} \to A^* \to A$ from Section 2. More specifically, the carrier A as well as the signature $\mathcal{F}$ are implicit in the type. In Isabelle it is not possible to quantify over types. Hence, we cannot define semantic entailment by

$$\forall \gamma.\, \forall I :: \alpha \Rightarrow \gamma\ list \Rightarrow \gamma.\, ((\forall s' \approx t' \in \mathcal{E}.\, I \models s' \approx t') \longrightarrow I \models s \approx t) \tag{1}$$

To solve this problem we just omit the outer quantifier and define $\mathcal{E} \models s \approx t$ by

$$\forall I :: \alpha \Rightarrow \gamma\ list \Rightarrow \gamma.\, ((\forall s' \approx t' \in \mathcal{E}.\, I \models s' \approx t') \longrightarrow I \models s \approx t) \tag{2}$$

where γ, the type of carrier elements, is a type variable. To make this dependence more prominent, we write $\mathcal{E} \models_\gamma s \approx t$ in the following.

Using (2) we proved one direction of Birkhoff's theorem – whenever $s \leftrightarrow^*_{\mathcal{E}} t$ then $\mathcal{E} \models_\gamma s \approx t$ – via induction over the length of the conversion $s \leftrightarrow^*_{\mathcal{E}} t$. Since in this theorem γ is arbitrary, we have indeed formalized that provability implies semantic entailment as defined in Section 2.

For the other direction, we shortly recall the main ideas of the proof of Birkhoff's theorem as presented in [1]. We construct an $\mathcal{F}$-algebra where the carrier is the set of equivalence classes of $\leftrightarrow^*_{\mathcal{E}}$, i.e., $A = \mathcal{T}(\mathcal{F}, \mathcal{V})/{\leftrightarrow^*_{\mathcal{E}}}$ and every symbol is interpreted by itself, i.e., $I(f)([t_1]_{\leftrightarrow^*_{\mathcal{E}}}, \ldots, [t_n]_{\leftrightarrow^*_{\mathcal{E}}}) = [f(t_1, \ldots, t_n)]_{\leftrightarrow^*_{\mathcal{E}}}$. Note that I is well-defined, since $\leftrightarrow^*_{\mathcal{E}}$ is a congruence. It can be shown that (A, I) is a model of all equations in $\mathcal{E}$. Hence, whenever $\mathcal{E} \models s \approx t$, then $[s]_{\leftrightarrow^*_{\mathcal{E}}} = [t]_{\leftrightarrow^*_{\mathcal{E}}}$ and thus, $s \leftrightarrow^*_{\mathcal{E}} t$.

The problem in the formalization of this proof is that the carrier $A = \mathcal{T}(\mathcal{F}, \mathcal{V})/{\leftrightarrow_{\mathcal{E}}^*}$ is not expressible as type in Isabelle. The reason is that quotient types can only be built statically, and may not depend on a dynamic input like an arbitrary set of equations $\mathcal{E}$. To this end, in the formalization we replaced (2) by

$$\forall A :: \gamma\ set.\, \forall I.\, \mathit{well\text{-}defined}_A(I) \wedge (\forall s' \approx t' \in \mathcal{E}.\, I \models s' \approx t') \longrightarrow I \models s \approx t \tag{3}$$

where the carrier A is not a type, but an arbitrary set having elements from the parametric type γ. Here, the predicate $\mathit{well\text{-}defined}_A(I)$ checks whether the interpretation I is well-defined w.r.t. the carrier A, i.e., for every f it must be ensured that $I(f)(a_1, \ldots, a_n) \in A$ whenever each $a_i \in A$. In (1) and (2) this was automatically enforced by the type system, but this is no longer true in (3). With this definition we finally write $\mathcal{E} \models_\gamma s \approx t$ to denote that the considered carriers are of type $\gamma\ set$.

By (3) it is straightforward to formalize Birkhoff's theorem using the proof ideas above. For the direction from semantic to syntactic entailment we choose γ to be the type of sets of terms, written $(\alpha, \beta)\ term\ set$ in Isabelle, where α is the type of function symbols and β the type of variables.

▶ **Theorem 6.1** (Birkhoff). *Let $\mathcal{E}$ be a set of equations over terms of type (α, β) term. Let α, β, and γ be arbitrary types. Then we have:*

- *Whenever $s \leftrightarrow_{\mathcal{E}}^* t$ then $\mathcal{E} \models_\gamma s \approx t$.*
- *Whenever $\mathcal{E} \models_{(\alpha,\beta)\ term\ set} s \approx t$ then $s \leftrightarrow_{\mathcal{E}}^* t$.*
- *$\mathcal{E} \models_{(\alpha,\beta)\ term\ set} s \approx t$ if and only if $s \leftrightarrow_{\mathcal{E}}^* t$.*

7 Conclusion

In this paper we formalized several properties of KBO including well-foundedness, where we used a variant with quasi-precedences, subterm coefficient functions, and infinite signatures. For well-foundedness we used a direct proof, and we illustrated why Kruskal's tree theorem could only be applied in a less powerful variant of KBO. Moreover, we have formalized the critical pair theorem, Birkhoff's theorem, and soundness of Knuth-Bendix completion for finite runs, where we could drop the strict encompassment condition. The latter result was already mentioned in a preliminary paper [22] and it has been extended in [28, Sections 5.1.2 and 6.1.2] to other versions of completion: Winkler showed that the strict encompassment condition can also be dropped for ordered completion [2] and normalized completion [15], if one considers finite runs, although without any formalization.

As far as we know, our results are the first formalizations of KBO, completion, and Birkhoff's theorem. Besides the generic theorems, we also developed executable criteria which allow us to certify completion proofs and (non-)derivability proofs, e.g., we can certify proofs from KBCV [23] and MKBTT [29].

Related work in Coq also allows checking a TRS for local confluence and termination, where C*i*ME3 [5] generates proof scripts for Coq. Although C*i*ME3 contains functionality to check derivability, we did not find any possibility to certify completion proofs or non-derivability proofs.

We also mention that resolution based ATPs are already used for finding and certifying proofs, but in a different way than we have presented: Sledgehammer [4] extracts the axioms used in a refutation proof (found by an external ATP) and then tries to find a certified proof using a variant of the Metis ATP whose inferences go through Isabelle's kernel.

An interesting future work would be to integrate our findings into Sledgehammer.

Acknowledgments

The authors are listed in alphabetical order regardless of individual contributions or seniority. We thank the anonymous referees for their helpful comments and remarks.

References

1 F. Baader and T. Nipkow. *Term Rewriting and All That*. New York, USA, paperback edition, Aug. 1999. doi:10.2277/0521779200.

2 L. Bachmair and N. Dershowitz. Equational inference, canonical proofs, and proof orderings. *J. ACM*, 41(2):236–276, 1994. doi:10.1145/174652.174655.

3 G. Birkhoff. On the structure of abstract algebras. *Math. Proc. Cambridge*, 31(4):433–454, 1935. doi:10.1017/S0305004100013463.

4 S. Böhme and T. Nipkow. Sledgehammer: Judgement day. In *International Joint Conference on Automated Reasoning, IJCAR'10*, volume 6173 of *Lecture Notes in Computer Science*, pages 107–121. doi:10.1007/978-3-642-14203-1_9.

5 É. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Automated certified proofs with C*i*ME3. In *Rewriting Techniques and Applications, RTA'11*, volume 10 of *Leibniz International Proceedings in Informatics*, pages 21–30. doi:10.4230/LIPIcs.RTA.2011.21.

6 J. Dick, J. Kalmus, and U. Martin. Automating the Knuth-Bendix ordering. *Acta Inform.*, 28(2):95–119, 1990. doi:10.1007/BF01237233.

7 A. L. Galdino and M. Ayala-Rincón. A formalization of the Knuth-Bendix-Huet critical pair theorem. *J. Automat. Reason.*, 45(3):301–325, 2010. doi:10.1007/s10817-010-9165-2.

8 F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In *Functional and Logic Programming, FLOPS'10*, volume 6009 of *Lecture Notes in Computer Science*, pages 103–117. doi:10.1007/978-3-642-12251-4_9.

9 G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980. doi:10.1145/322217.322230.

10 D. E. Knuth and P. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. 1970.

11 K. Korovin and A. Voronkov. Orienting rewrite rules with the Knuth-Bendix order. *Inform. and Comput.*, 183(2):165–186, 2003. doi:10.1016/S0890-5401(03)00021-X.

12 A. Krauss. Recursive definitions of monadic functions. In *Workshop on Partiality and Recursion in Interactive Theorem Proving, PAR'10*, volume 43 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–13. doi:10.4204/EPTCS.43.1.

13 A. Krauss. Partial and nested recursive function definitions in higher-order logic. *J. Automat. Reason.*, 44(4):303–336, 2010. doi:10.1007/s10817-009-9157-2.

14 M. Ludwig and U. Waldmann. An extension of the Knuth-Bendix ordering with LPO-like properties. In *LPAR'07*, volume 4790 of *Lecture Notes in Computer Science*, pages 348–362. doi:10.1007/978-3-540-75560-9_26.

15 C. Marché. Normalized rewriting: An alternative to rewriting modulo a set of equations. *J. Symb. Comp.*, 21(3):253–288, 1996. doi:10.1006/jsco.1996.0011.

16 A. Middeldorp and H. Zantema. Simple termination of rewrite systems. *Theor. Comput. Sci.*, 175(1):127–158, 1997. doi:10.1016/S0304-3975(96)00172-7.

17 T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. 2002. doi:10.1007/3-540-45949-9.

18 J. Ruiz-Reina, J. Alonso, M. Hidalgo, and F.-J. Martín-Mateos. Formal proofs about rewriting using ACL2. *Ann. Math. Artif. Intel.*, 36(3):239–262, 2002. doi:10.1023/A:1016003314081.

19 J. Steinbach. Extensions and comparison of simplification orders. In *Rewriting Techniques and Applications, RTA'89*, volume 355 of *Lecture Notes in Computer Science*, pages 434–448. doi:10.1007/3-540-51081-8_124.

20 C. Sternagel. A formal proof of Kruskal's tree theorem in Isabelle/HOL. draft.

21 C. Sternagel. Well-Quasi-Orders. *Archive of Formal Proofs*, Apr. 2012. http://afp.sourceforge.net/entries/Well_Quasi_Orders.shtml, Formal proof development.

22 T. Sternagel, R. Thiemann, H. Zankl, and C. Sternagel. Recording completion for finding and certifying proofs in equational logic. In *International Workshop on Confluence, IWC'12*, pages 31–36. Available at http://arxiv.org/abs/1208.1597.

23 T. Sternagel and H. Zankl. KBCV – Knuth-Bendix completion visualizer. In *International Joint Conference on Automated Reasoning, IJCAR'12*, volume 7364 of *Lecture Notes in Artificial Intelligence*, pages 530–536. doi:10.1007/978-3-642-31365-3_41.

24 R. Thiemann, G. Allais, and J. Nagele. On the formalization of termination techniques based on multiset orderings. In *Rewriting Techniques and Applications, RTA'12*, volume 15 of *Leibniz International Proceedings in Informatics*, pages 339–354. doi:10.4230/LIPIcs.RTA.2012.339.

25 R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Theorem Proving in Higher Order Logics, TPHOLs'09*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468. doi:10.1007/978-3-642-03359-9_31.

26 R. Thiemann and C. Sternagel. Loops under strategies. In *Rewriting Techniques and Applications, RTA'09*, volume 5595 of *Lecture Notes in Computer Science*, pages 17–31. doi:10.1007/978-3-642-02348-4_2.

27 Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *J. ACM*, 34(1):128–143, 1987. doi:10.1145/7531.7534.

28 S. Winkler. *Termination Tools in Automated Reasoning*. PhD thesis, University of Innsbruck, 2013.

29 S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Optimizing mkbTT (system description). In *Rewriting Techniques and Applications, RTA'10*, volume 6 of *Leibniz International Proceedings in Informatics*, pages 373–384. doi:10.4230/LIPIcs.RTA.2010.373.

30 H. Zankl, N. Hirokawa, and A. Middeldorp. KBO orientability. *J. Automat. Reason.*, 43(2):173–201, 2009. doi:10.1007/s10817-009-9131-z.