On the Completeness of Context-Sensitive Order-sorted Specifications

Joe Hendrix and José Meseguer

University of Illinois at Urbana-Champaign
{jhendrix,meseguer}@uiuc.edu

Abstract. We propose three different notions of completeness for term rewrite specifications supporting order-sorted signatures, deduction modulo axioms, and context-sensitive rewriting relative to a replacement map μ. Our three notions are: (1) an appropriate definition of μ-sufficient completeness with respect to a set of constructor symbols; (2) a definition of μ-canonical completeness under which μ-canonical forms coincide with canonical forms; and (3) a definition of semantic completeness that guarantees that the μ-operational semantics and standard initial algebra semantics are isomorphic. Based on these notions, we use equational tree automata techniques to obtain decision procedures for checking these three kinds of completeness for equational specifications satisfying appropriate requirements such as ground confluence, ground sort-decreasingness, weakly normalization, and left-linearity. Although the general equational tree automata problems are undecidable, our algorithms work modulo any combination of associativity, commutativity, and identity axioms. For all combinations of these axioms except associativity without commutativity, our algorithms are decision procedures. For the associativity without commutativity case, which is undecidable in general, our algorithms use learning techniques that are effective in all practical examples we have considered. We have implemented these algorithms as an extension of the Maude sufficient completeness checker.

1 Introduction

In equational programming there is a relentless drive to increase the expressiveness and generality of programs. This provides a much easier and elegant way of mapping many applications into such languages. For example, the use of sorts, subsorts, and matching modulo axioms like associativity and/or commutativity, makes equational programming much easier and allows very elegant and succinct programming solutions.

Another important dimension, along which program expressiveness can be substantially increased is that of user-programmable evaluation strategies based on *context-sensitive* (CS) rewriting (see, for example [14, 16, 21]). They allow very fine-grained control (at the level of each individual function symbol) on how the rewriting evaluation is performed. Their value and practical importance has been recognized in many equational languages. OBJ2 [6] was the first such language supporting them; and they are, for example, supported in all languages in the OBJ family, including CafeOBJ [5] and Maude [2]. In practice, CS rewriting can be used for two somewhat different purposes:

1. to increase the *efficiency* of a standard equational program without changing its meaning: for example by restricting the evaluation of an if-then-else symbol to its first, boolean argument to avoid wasteful or even nonterminating computations; and
2. as a way to compute with *infinite data structures* such as, for example, the infinite stream of all prime numbers, in a *lazy* way; in this second case, CS rewriting provides an elegant, finitary way of computing with infinite objects.

Expressiveness is substantially increased in both of these ways, since the user can *both* control the efficiency of program execution and map into the language new applications involving infinite data structures.

This is all very well. However, there are a number of open research questions about how to *reason* formally about equational programs supporting CS rewriting for verification purposes. Two areas where important progress has been made are in methods for proving termination, e.g., [7, 18, 21] and confluence [14] of CS equational programs. But other important questions remain unexplored.

Imagine, for example, that you want to use an inductive theorem prover to verify some property about a CS equational program. No inductive theorem prover that we are aware of allows reasoning about CS programs. Is it ok to *ignore* the CS information and just reason about the underlying equational theory? We think that, in general, the answer is: *definitely not!* Why not? Because the *model* on which the inductive reasoning principles are sound and that of the CS program may be quite different.

What models are we talking about? Well, that is, indeed, one of the interesting research questions. For an equational *theory* (Σ, E), the model on which inductive reasoning is sound is obvious, namely, the *initial algebra* $T_{\Sigma/E}$. In fact, *initial algebra semantics* is the standard mathematical semantics of equational programs in languages such as OBJ, CafeOBJ, and Maude. Furthermore, provided that the equational program is weakly normalizing and ground confluent, the initial algebra semantics fully *agrees* with the *operational semantics*, in the precise, mathematical sense that the initial algebra $T_{\Sigma/E}$ and the *canonical term algebra* $\mathrm{Can}_{\Sigma/E}$ obtained by rewriting are *isomorphic*. For CS rewriting the matter is less obvious, since we only have an operational semantics provided by the CS rewriting relation, but as far as we know no mathematical models in the form of *algebras* have been put forward. Therefore, the first thing we do in this work is to put forward such an algebra, namely, the algebra $\mathrm{Can}^{\mu}_{\Sigma/E}$ of *μ-canonical forms*, for μ the replacement map of the given CS program. We do so not just for vanilla-flavored, untyped CS programs, but for the more general and expressive CS programs with other features such as order-sorting and rewriting module axioms that one encounters in actual languages.

The importance of the algebra $\mathrm{Can}^{\mu}_{\Sigma/E}$ is that it makes possible articulating and providing proof methods for three important CS *completeness problems*, namely:

1. *μ-canonical completeness*, which means satisfying the set-theoretic equality $\mathrm{Can}^{\mu}_{\Sigma/E,s} = \mathrm{Can}_{\Sigma/E,s}$ for each sort s in the specification;

2. *μ-semantic completeness*, which model-theoretically corresponds to the case where the surjective Σ-homomorphism $q : \mathrm{Can}^{\mu}_{\Sigma/E} \to T_{\Sigma/E}$, which we show always exists under minimal assumptions, is an *isomorphism*, and proof-theoretically means that the *sound* way of proving ground E-equalities by CS rewriting is also *complete*;
3. *μ-sufficient completeness*, which is in fact a new notion generalizing to the CS case the usual sufficient completeness of equational function definitions with respect to a signature of constructors. The subtlety here is that in general it would be *too strong* to require that constructors appear in all positions of a term t in μ-canonical form: we only make such a requirement for *replacing* positions in t.

Our goal is not only to articulate these notions, but also to provide proof methods for them in the form of *decision procedures* under mild assumptions about the given CS program. Given that the CS programs we consider perform rewriting modulo axioms and are order-sorted, our methods are based on *Propositional Tree Automata* [12], a kind of equational tree automata, that can take into account both sort information and reasoning modulo axioms. These decision procedures have been implemented in an extension of Maude's Sufficient Completeness Checker (SCC) [10], and we use several Maude programs to illustrate both the basic ideas and the use of SCC in verifying CS completeness properties.

The paper is organized as follows. In Section 2, we review basic concepts from order-sorted algebra, and introduce the precise class of CS term-rewrite systems we are considering. In Section 3, we define the canonical term algebra for a CS specification. In Section 4, we define the three notions of CS completeness, and in Section 5 we show how one can use PTA to check these completeness notions under appropriate assumptions. Finally, we discuss related work and suggest future avenues of research in Section 6. Full proofs are in the Appendix.

2 Preliminaries

2.1 Order-Sorted Algebra

Order-sorted algebras are an extension of many-sorted algebras where a partial order $\leq$ is associated to the sorts in order to build a notion of subtype and supertype into the algebra and operators can be overloaded and then must agree on common data.

Definition 1. *An* order-sorted signature *is a tuple $\Sigma = (S, F, \leq)$ where*

- *$(S, \leq)$ is a poset; and*
- *$F = \{F_{w,s}\}_{(w,s)\in S^* \times S}$ is a family of operator symbols such that if $f \in F_{w,s} \cap F_{w',s'}$ then $w \equiv_{\leq} w'$ and $s \equiv_{\leq} s'$ where $\equiv_{\leq}$ denotes the equivalence relation generated by $\leq$ extended to sequences in the usual way.*

We assume the existence of an S-sorted family of variables $X = \{X_s\}_{s\in S}$ distinct from the operators F, where each set X_s is a countably infinite and $X_s \cap X_{s'} = \varnothing$ for distinct sorts $s, s' \in S$. We write x_s if x is a variable in X_s. When the signature $\Sigma = (S, F, \leq)$ is clear, we sometimes write $f : s_1 \dots s_n \to s$

for $f \in F_{s_1 \ldots s_n, s}$. Given a signature Σ, $T_\Sigma(X)_s$ denotes the set of terms with sort s formed by the operators in Σ and variables in X, and $T_{\Sigma,s}$ denotes the set of ground terms with sort s. A substitution $\theta : X \to T_\Sigma(X)$ is a mapping such that $\theta(x_s) \in T_\Sigma(X)_s$ for each $x_s \in X_s$,

Definition 2. *An* order-sorted theory *is a pair $\mathcal{E} = (\Sigma, E)$ where $\Sigma = (S, F, \leq)$ is an order-sorted signature, and E is a set of equations of the form $l = r$ with $l, r \in T_\Sigma(X)$ terms having sorts in the same equivalence class in $S/\equiv_\leq$.*

There are various inference systems in order-sorted logic for deriving equations of the form $t = u$. We use the sound and complete inference system presented in [19]. We also use the definition for order-sorted algebras and homomorphisms found in [19]. See [9, 19] for surveys on order-sorted algebra.

Definition 3. *A Σ-algebra A for an order-sorted signature $\Sigma = (S, F, \leq)$ consists of:*

- *a set A_s for each sort $s \in S$ such that $A_s \subseteq A_{s'}$ for $s \leq s'$;*
- *a function $A_{f:w \to s} : A_w \to A_s$ for each symbol $f \in F_{w,s}$ where $w = s_1 \ldots s_n$, $A_w = A_{s_1} \times \ldots A_{s_n}$, and $A_{f:w \to s}(\bar{a}) = A_{f:w' \to s'}(\bar{a})$ for each $f \in F_{w,s} \cap F_{w',s'}$ and $\bar{a} \in A_w \cap A_{w'}$.*

A *Σ-homomorphism* $h : A \to B$ is a family of functions $\{h_s : A_s \to B_s\}_{s \in S}$ such that: if $s \equiv_\leq s'$ and $a \in A_s \cap A_{s'}$, then $h_s(a) = h_{s'}(a)$; and for $f \in F_{s_1 \ldots s_n, s}$ and $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}$, we have $h_s(A_f(a_1, \ldots, a_n)) = B_f(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$.

Given an order-sorted theory $\mathcal{E} = (\Sigma, E)$, an $\mathcal{E}$-algebra is a Σ-algebra satisfying the equations in E. We let T_Σ denote the term algebra for Σ, and $T_{\Sigma/E}$ denote the $\mathcal{E}$-algebra such that $T_{\Sigma/E,s} = \{[t]_E \mid t \in T_{\Sigma,s}\}$ for each sort $s \in S$, where $[t]_E$ denotes the equivalence class of t under $=_E$. Both T_Σ and $T_{\Sigma/E}$ are *initial* for the categories of Σ-algebras and $\mathcal{E}$-algebras respectively, so there is a unique homomorphism from T_Σ to any Σ-algebra, and a unique homomorphism from $T_{\Sigma/E}$ to any $\mathcal{E}$-algebra. For a Σ-algebra A and term $t \in T_\Sigma$, we let $A(t)$ denote the value of t in the unique homomorphism $A : T_\Sigma \to A$, i.e., $A(f(t_1, \ldots, t_n)) = A_f(A(t_1), \ldots, A(t_n))$.

2.2 Context-sensitive Order-sorted Term Rewrite Systems

In order to execute an equational theory, one typically treats the equations $l = r \in E$ as rewrite rules $l \to r$ and simplifies expressions from left to right. The most advanced rewrite engines today have matching algorithms capable of matching modulo specific equational axioms such as associativity and commutativity, and with such systems we treat those specific axioms as equations while treating the other axioms as rules. When rewriting modulo axioms, the variables in the axioms are typically constrained at the level of the connected component rather than of individual sorts. We include this restriction in the definition below:

Definition 4. *An* Order-sorted Term Rewrite System *(TRS) is a tuple $\mathcal{R} = (\Sigma, A, R)$ where:*

- *$\Sigma = (S, F, \leq)$ is an order-sorted signature where each connected component $[s] \in S/\equiv_\leq$ contains a maximal sort denoted by k_s;*

– *A is a set of unconditional Σ-equations where the variables in each equation are only constrained with the maximal sorts; and*
– *R is a set of rewrite rules of the form $l \to r$ with $l, r \in T_\Sigma(X)_{k_s}$ for some $k_s \in S$, and $\mathrm{vars}(r) \subseteq \mathrm{vars}(l)$;*

Given a TRS $\mathcal{R} = (\Sigma, A, R)$, $\mathrm{lhs}(R)$ denotes the left-hand sides of the rules in R, i.e., $\mathrm{lhs}(R) = \{l \mid l \to r \in R\}$. We say that an order-sorted TRS $\mathcal{R} = (\Sigma, A, R)$ is *left-linear* if each $l \in \mathrm{lhs}(R)$ is linear. Due to the restrictions on the signatures and equations in our term rewrite system we are able to treat our order-sorted axioms as many-sorted axioms for the purposes of matching modulo.

Definition 5. *Given an order-sorted TRS $\mathcal{R} = (\Sigma, A, R)$ with $\Sigma = (S, F, \leq)$, we let $\Sigma^k = (S^K, F^K)$ denote the many-sorted signature where S^K contains exactly the maximal sorts $k_s \in S$, and F^K contains an operator $f : k_{s_1} \dots k_{s_n} \to k_s$ for each $f \in F_{s_1 \dots s_n, s}$.*

Due to our restrictions on the equations in our term-rewrite theories, we can essentially use $=_A$ to denote $=_{(\Sigma,A)}$ and $=_{(\Sigma^K,A)}$ interchangeably on ground terms, as justified by the following lemma which can be easily shown:

Lemma 1. *Given a order-sorted TRS $\mathcal{R} = (\Sigma, A, R)$, for all terms $t, u \in T_\Sigma$, we have $t =_{(\Sigma,A)} u$ iff $t =_{(\Sigma^k A)} u$.*

We are interested in studying and analyzing CS rewriting for order-sorted term rewrite systems. In CS rewriting, there is a function $\mu : F \to \mathcal{P}(\mathbb{N})$, called the *replacement map*, which maps each function symbol $f \in F$ to a set of *replacing positions* $\mu(f) \subseteq \{1, \dots, \mathrm{arity}(f)\}$. The replacement map μ is used for restricting rewriting so that in rewriting a term $f(t_1, \dots, t_n) \in T_\Sigma(X)$, the term t_i can only be rewritten if $i \in \mu(f)$. A *CS term rewrite system* is a pair $(\mathcal{R}, \mu)$ where μ is a replacement map for the signature used in $\mathcal{R}$.

Given a replacement map μ, the set of positions that may be rewritten are called the *μ-replacing positions* and denoted by $\mathrm{pos}^\mu(t)$. Formally, we have:

$$\mathrm{pos}^\mu(x) = \{\epsilon\} \quad \text{and} \quad \mathrm{pos}^\mu(f(t_1, \dots, t_n)) = \{\epsilon\} \cup \bigcup_{i \in \mu(f)} \{iw \mid w \in \mathrm{pos}^\mu(t_i)\}.$$

A context C is *μ-replacing* when $\Box$ appears in a μ-replacing position.

We write $t \to_{\mathcal{R},\mu} u$ if t rewrites to u using the rules in $\mathcal{R}$ and replacement map μ in a *single* rewrite step, i.e., there is a rule $l \to r$ in Γ such that $t =_A C[l\theta]$ and $u =_A C[r\theta]$ for some μ-replacing context C and substitution $\theta : X \to T_\Sigma(X)$. The reflexive and transitive of closure of $\to_{\mathcal{R},\mu}$ is $\to^*_{\mathcal{R},\mu}$. We write $t \downarrow^\mu_{\mathcal{R}} u$ if t and u can be rewritten to the same term, i.e., there is a term $v \in T_\Sigma(X)$ such that $t \to^*_{\mathcal{R},\mu} v$ and $u \to^*_{\mathcal{R},\mu} v$.

A term $t \in T_\Sigma(X)$ is *$(\mathcal{R}, \mu)$-reducible* iff there is a $u \in T_\Sigma(X)$ such that $t \to_{\mathcal{R},\mu} u$, and *$(\mathcal{R}, \mu)$-irreducible* otherwise. We also say that a μ-irreducible term $t \in T_\Sigma(X)$ is in *μ-canonical* form. We write $t \to^!_{\mathcal{R},\mu} u$ if $t \to^*_{\mathcal{R},\mu} u$ and u is $(\mathcal{R}, \mu)$-irreducible. $\mathcal{R}$ is *μ-weakly-normalizing* when for each term $t \in T_\Sigma(X)$ there is a term $u \in T_\Sigma(X)$ such that $t \to^!_{\mathcal{R},\mu} u$. $\mathcal{R}$ is *μ-terminating* if the relation $\to_{\mathcal{R},\mu}$ is Noetherian. $\mathcal{R}$ is *μ-confluent* if for all $t, u, v \in T_\Sigma(X)$, $t \to^*_{\mathcal{R},\mu} u$ and $t \to^*_{\mathcal{R},\mu} v$ implies $u \downarrow^\mu_{\mathcal{R}} v$. $\mathcal{R}$ is *μ-sort-decreasing* if for all terms $t \in T_\Sigma(X)_s$ and

$u \in T_\Sigma(X)_{k_s}$, $t \to^*_{\mathcal{R},\mu} u$ implies that there is a term $v \in T_\Sigma(X)_s$ such that $u \to^*_{\mathcal{R},\mu} v$. When $\mathcal{R}$ is μ-weakly normalizing, μ-confluent, or μ-sort-decreasing on ground terms, we say that it is ground μ-weakly normalizing, ground μ-confluent, or ground μ-sort-decreasing, respectively.

When the replacement map μ allows rewriting at every subterm position, this inference system specializes to rewriting in the ordinary sense. Let $\mu_\top$ be the replacement map $f \mapsto \{1, \ldots, \text{arity}(f)\}$. We write $t \to_{\mathcal{R}} u$ iff $t \to_{\mathcal{R},\mu_\top} u$, and $t \to^*_{\mathcal{R}} u$ iff $t \to^*_{\mathcal{R},\mu_\top} u$. Additionally, we will say that a system $\mathcal{R}$ is *weakly-normalizing* iff it is $\mu_\top$-weakly-normalizing. More generally, we extend this convention to all other properties. For example, we will say that $\mathcal{R}$ is confluent iff it is $\mu_\top$-confluent.

3 Context-Sensitive Canonical Term Algebras

When $\mathcal{R} = (\Sigma, A, R)$ is ground μ-weakly-normalizing and ground μ-confluent, for each term $t \in T_\Sigma$, there is a $(\mathcal{R}, \mu)$-irreducible term, denoted by $t\,!^\mu_{\mathcal{R}}$ such that $t \to^!_{\mathcal{R},\mu} t\,!^\mu_{\mathcal{R}}$, which is unique up to A. When $\mathcal{R}$ is additionally sort-decreasing, we can use this uniqueness to define a (Σ, A)-algebra of $(\mathcal{R}, \mu)$ canonical forms as follows:

Definition 6. *Let $\mathcal{R} = (\Sigma, A, R)$ be a TRS with $\Sigma = (S, F, \leq)$ that is ground μ-weakly-normalizing, ground μ-confluent and ground μ-sort-decreasing. The* canonical term algebra *for $(\mathcal{R}, \mu)$ is the Σ-algebra $\text{Can}^\mu_{\mathcal{R}}$ such that:*

- *for each sort $s \in S$, $\text{Can}^\mu_{\mathcal{R},s} = \{[t\,!^\mu_{\mathcal{R}}]_A \mid \exists u \in [t\,!^\mu_{\mathcal{R}}]_A \cap T_{\Sigma,s}\}$; and*
- *for each $f \in F_{w,s}$, $\text{Can}^\mu_{\mathcal{R},f:w\to s}([t_1]_A, \ldots, [t_n]_A) = [f(u_1, \ldots, u_n)\,!^\mu_{\mathcal{R}}]_A$ where $u_i \in [t_i]_A \cap T_{\Sigma,s}$ for $1 \leq i \leq n$.*

Note that $\text{Can}^\mu_{\mathcal{R}}$ has a strong computation meaning: it is exactly the algebra of *values* (μ-normal forms) that a user interacting with a system that evaluates $(\mathcal{R}, \mu)$ obtains by reduction.[1] It provides, therefore, the perfect algebra for the *operational semantics* of $(\mathcal{R}, \mu)$. This model is in a sense situated *between* the initial algebra T_Σ, and the model for the *mathematical semantics* of $\mathcal{R}$ as an equational theory, namely, the initial algebra $T_{\Sigma/A\cup R}$. On the one hand, by initiality we have a unique homomorphism $\text{Can}^\mu_{\mathcal{R}} : T_\Sigma \to \text{Can}^\mu_{\mathcal{R}}$ which, as shown below, may not be surjective. On the other hand, $\text{Can}^\mu_{\mathcal{R}}$ is *more concrete* than $T_{\Sigma/A\cup R}$, and therefore a *sound*, but not necessarily complete, model for equational computation with $\mathcal{R}$. That is, we have:

Proposition 1. *If $\mathcal{R} = (\Sigma, A, R)$ with $\Sigma = (S, F, \leq)$ is ground μ-weakly normalizing, ground μ-confluent, and ground μ-sort-decreasing, then the family of functions $\{q_s : \text{Can}^\mu_{\mathcal{R},s} \to T_{\Sigma/A\cup R,s}\}_{s\in S}$ with $q_s : [t]_A \mapsto [t]_{A\cup R}$ defines a surjective Σ-homomorphism $q : \text{Can}^\mu_{\mathcal{R}} \to T_{\Sigma/A\cup R}$.*

One typically constructs ground terminating and confluent specifications in order to reason about the equivalence of two terms algebraically, and it is important to be able to reduce the equality problem $t =_{A\cup R} u$ to the convergence

[1] If $(\mathcal{R}, \mu)$ is μ-terminating, this is exactly true; if it is only μ-weakly normalizing, this requires either a μ-normalizing strategy, or the use of breadth-first search.

problem $t \downarrow^{\mu}_{\mathcal{R}} u$. When considering ordinary (not context-sensitive) rewriting, we have $t =_{A \cup R} u$ iff $t \downarrow_{\mathcal{R}} u$ iff $t !_{\mathcal{R}} =_A u !_{\mathcal{R}}$ for terms $t, u \in T_\Sigma$ when $\mathcal{R}$ is ground weakly-normalizing, ground confluent, and ground sort-decreasing. In this case, we are guaranteed that $\text{Can}_{\mathcal{R}} = \text{Can}^{\mu_\top}_{\mathcal{R}}$ is isomorphic to $T_{\Sigma/A \cup R}$, thus obtaining a perfect agreement between the operational semantics of $\mathcal{R}$ and the mathematical, initial algebra semantics. In general, as we show below, this is *not* the case for CS rewriting, even if $\mathcal{R}$ is ground μ-terminating, ground μ-confluent, and ground μ-sort-decreasing. That is $\text{Can}^{\mu}_{\mathcal{R}}$ is *sound*, since $t \downarrow^{\mu}_{\mathcal{R}} u$ implies $t =_{A \cup R} u$, but in general is *not complete*, i.e., $t =_{A \cup R} u \not\Rightarrow t \downarrow^{\mu}_{\mathcal{R}} u$.

Consider the specification $\mathcal{R}$ with single sort s, symbols $a : \rightarrow s$, $b : \rightarrow s$, and $f : s \rightarrow s$, and replacement map μ where $\mu(f) = \varnothing$ with the rules: $a \rightarrow f(a)$ and $b \rightarrow f(a)$. This specification is clearly μ-weakly normalizing, μ-confluent, and μ-sort-decreasing. However, $T_{\Sigma/A \cup R, s} = \{[a]_{A \cup R}\}$ whereas $\text{Can}^{\mu}_{\mathcal{R},s}$ is the infinite set $\{\{f(a)\}, \{f(b)\}, \{f(f(a))\}, \{f(f(b))\}, \dots\}$.

The algebra $\text{Can}^{\mu}_{\mathcal{R}}$ differs from $\text{Can}_{\mathcal{R}}$ in several other properties as well. In general, it is not the case that $\text{Can}^{\mu}_{\mathcal{R}}(t) = t !^{\mu}_{\mathcal{R}}$. In the specification above, $\text{Can}^{\mu}_{\mathcal{R}}(f(a)) = f(\text{Can}^{\mu}_{\mathcal{R}}(a)) !^{\mu}_{\mathcal{R}} = f(f(a))$, whereas $f(a) !^{\mu}_{\mathcal{R}} = f(a)$. Additionally, the unique homomorphism $\text{Can}^{\mu}_{\mathcal{R}} : T_{\Sigma/A} \rightarrow \text{Can}^{\mu}_{\mathcal{R}}$ is neither surjective nor idempotent. For example, there is no canonical term $t \in T_\Sigma$, such that $\text{Can}^{\mu}_{\mathcal{R}}(\{t\}) = \{f(b)\}$, while $\text{Can}^{\mu}_{\mathcal{R}}(\{a\}) = \{f(a)\}$ and $\text{Can}^{\mu}_{\mathcal{R}}(\{f(a)\}) = \{f(f(a))\}$.

4 Completeness in Context Sensitive Rewriting

We have now shown that the usual requirements of μ-termination, μ-confluence, and μ-sort-decreasingness are insufficient to guarantee that the operational semantics of CS term rewriting corresponds to the mathematical semantics of the equational specification. One of the goals of this section is investigating what additional conditions we need to impose to guarantee that CS rewriting serves as a sound and complete technique to deduce ground equalities, i.e., when is the canonical term algebra $\text{Can}^{\mu}_{\mathcal{R}}$ isomorphic to the initial algebra $T_{\Sigma/A \cup R}$.

In this section, we introduce three notions of completeness for CS term rewrite systems $(\mathcal{R}, \mu)$: (1) μ-canonical completeness; (2) μ-semantic completeness; and (3) μ-sufficient completeness. The first two notions of completeness are used to characterize the deductive power of CS rewriting. The third is used to analyze specifications that may not be complete in the first two senses, but may nevertheless represent useful applications of CS rewriting, such as specifying infinite data-structures. Later, in Section 5, we will show how these three completeness properties can be checked for specifications satisfying appropriate requirements such as left-linearity, ground μ-weak normalization, ground μ-confluence, and ground μ-sort-decreasingness.

4.1 μ-Canonical Completeness

The first property we consider is whether the canonical forms of CS rewriting and ordinary rewriting agree:

Definition 7. *A TRS $\mathcal{R} = (\Sigma, A, R)$ is* μ-canonically complete *if every $(\mathcal{R}, \mu)$-irreducible term $t \in T_\Sigma$ is $\mathcal{R}$-irreducible.*

The theorem below shows that, for specifications that are ground μ-weakly normalizing and ground confluent, canonical completeness is enough to imply that CS and ordinary rewriting agree on convergence relations.

Theorem 1. *If a TRS $\mathcal{R}$ is ground μ-weakly normalizing, μ-canonically complete, and ground confluent, then for $t, u \in T_\Sigma$, $t \downarrow_\mathcal{R} u$ iff $t \downarrow^\mu_\mathcal{R} u$.*

As a corollary, we observe that this class of specifications is μ-confluent.

Corollary 1. *If $\mathcal{R}$ is ground μ-weakly normalizing, μ-canonically complete, and ground confluent, then $\mathcal{R}$ is ground μ-confluent.*

In a similar vein, we can show ground μ-sort-decreasingness of $\mathcal{R}$ by showing that $\mathcal{R}$ is ground μ-normalizing, μ-canonically complete, and sort-decreasing.

Theorem 2. *If $\mathcal{R}$ is ground μ-normalizing, μ-canonically complete, and ground sort-decreasing, then $\mathcal{R}$ is ground μ-sort-decreasing.*

Together, Corollary 1 and Theorem 2 provide a means to check μ-confluence and μ-sort-decreasingness for μ-weakly normalizing, μ-canonically complete, confluent, and sort-decreasing specifications. Since one can prove μ-termination with existing tools [4, 8, 17], and check μ-canonical completeness of left-linear specifications with the decision procedure in Section 5.2, this eliminates the need for specialized CS-aware checking procedures for this class of specifications. The case of ground weak-normalizing and ground μ-normalization for μ-canonically complete specification yields a relation in the other direction.

Theorem 3. *If $\mathcal{R}$ is ground μ-weakly normalizing and μ-canonically complete, then $\mathcal{R}$ is ground weakly-normalizing.*

On the other hand, if $\mathcal{R}$ is ground weakly normalizing and μ-canonically complete, it may not be ground μ-weakly-normalizing. Let $\mathcal{R}$ have the rules: $f(x) \to f(x)$, $a \to b$, and $f(b) \to b$. $\mathcal{R}$ is ground weakly-normalizing, because every term can reduce to the $\mathcal{R}$-irreducible term b. Given the replacement map μ with $\mu(f) = \varnothing$, $\mathcal{R}$ is μ-canonically complete, because b is the only $(\mathcal{R}, \mu)$-irreducible term as well. However, $\mathcal{R}$ is not μ-weakly normalizing, because $f(a) \not\to^*_{\mathcal{R},\mu} b$.

As an example of a μ-canonically complete specification, we present a Maude module below for computing the factorial of a natural number.

```
fmod FACTORIAL is  protecting NAT .
  var X Y Z : Nat .
  op p : Nat -> Nat .    eq p(s(X)) = X .   eq p(0) = 0 .
  op if0 : Nat Nat Nat -> Nat [strat(1 0)].
  eq if0(0, Y, Z) = Y .  eq if0(s(X), Y, Z) = Z .
  op fact : Nat -> Nat .
  eq fact(X) = if0(X, s(0), X * fact(p(X))) .
endfm
```

This specification protects the built-in `NAT` specification, which contains constructor operators `0` and `s` for for zero and successor respectively, along with defined operators for plus and times. Predecessor `p` is defined as usual, and the operator `if0` is annotated with a strategy `strat(1 0)`, indicating that only the first argument should be evaluated. Since the other operators are not given a

strategy, Maude uses its default strategy, which evaluates every argument. In effect, these declarations define a replacement map μ where $\mu(\texttt{if0}) = \{1\}$ and $\mu(f) = \{1, \ldots, \text{arity}(f)\}$ for $f \neq \texttt{if0}$. Using `if0`, factorial can be defined with a single equation.

Without the strategy declaration on `if0`, this specification is not terminating, and evaluating $\texttt{fact}(0)$ quickly leads to a segmentation fault in the Maude interpreter. However, with the given replacement map μ, the specification is μ-terminating. Moreover, it is μ-canonically complete, ground μ-confluent, and ground μ-sort-decreasing. Since there is only one sort, μ-sort-decreasingness is obvious. As the specification is left-linear, the decision procedure we introduce in Section 5.2 will allow us to automatically check its μ-canonically completeness. To see that it is ground μ-confluent one can just observe that it is confluent (indeed, orthogonal), and use Corollary 1.

4.2 μ-Semantic Completeness

Canonical completeness means that $\text{Can}_{\mathcal{R},s} = \text{Can}^{\mu}_{\mathcal{R},s}$ for each sort $s \in S$. By itself, this is not enough to immediately imply that $\text{Can}^{\mu}_{\mathcal{R}}$ and $T_{\Sigma/A\cup R}$ are isomorphic. This is implied by another notion of completeness, called *μ-semantic completeness*, which we define below.

Definition 8. *A TRS $\mathcal{R} = (\Sigma, A, R)$ is* μ-semantically complete *iff for all $t, u \in T_\Sigma$, $t \downarrow^{\mu}_{\mathcal{R}} u$ iff $t =_{A\cup R} u$.*

This definition at the syntactic level of terms captures the agreement between operational semantics and mathematical semantics that we want when the canonical algebra $\text{Can}^{\mu}_{\mathcal{R}}$ is well-defined.

Theorem 4. *If $\mathcal{R}$ is ground μ-weakly normalizing, ground μ-confluent, and ground μ-sort-decreasing, then μ-semantically complete iff $\text{Can}^{\mu}_{\mathcal{R}}$ is isomorphic to $T_{\Sigma/A\cup R}$.*

The next question that we address is how to check that a specification is μ-semantically complete. The results in the previous section on μ-canonical completeness lead to the following result:

Theorem 5. *A TRS $\mathcal{R}$ that is ground μ-weakly normalizing, μ-canonically complete, ground confluent, and ground sort-decreasing is μ-semantically complete.*

As a corollary, we can easily obtain checkable conditions under which all three of the algebras $\text{Can}^{\mu}_{\mathcal{R}}$, $\text{Can}_{\mathcal{R}}$ and $T_{\Sigma/A\cup R}$ are isomorphic.

Corollary 2. *If $\mathcal{R}$ is ground μ-weakly normalizing, ground μ-canonically complete, ground confluent, and ground sort-decreasing, then $\text{Can}^{\mu}_{\mathcal{R}}$ and $\text{Can}_{\mathcal{R}}$ are both well-defined and isomorphic to $T_{\Sigma/A\cup R}$.*

When the specification $\mathcal{R}$ is ground μ-weakly normalizing, ground confluent, and ground sort-decreasing, μ-canonical completeness is a *sufficient* condition to show μ-semantic completeness, but it turns out not to be a *necessary* condition. For example, let $\mathcal{R}$ have the rules: $f(f(x)) \to f(x)$, $a \to b$, and $f(b) \to f(a)$, and let μ be the replacement map with $\mu(f) = \varnothing$. The initial algebra contains two

equivalence classes: one with the constants a and b, the other with terms containing f. The $(\mathcal{R}, \mu)$-canonical terms are b and $f(a)$, and it is easy to that $\text{Can}^{\mu}_{\mathcal{R}}$ and $T_{\Sigma/A\cup R}$ are isomorphic. Since $\mathcal{R}$ is also ground μ-weakly normalizing, ground μ-confluent and ground μ-sort decreasing, $\mathcal{R}$ is μ-semantically complete by Theorem 4. However, $f(a)$ is $\mathcal{R}$-reducible, leaving b the only $\mathcal{R}$-irreducible term, and so $\mathcal{R}$ is not μ-canonically complete. In addition to not being μ-canonically complete, $\mathcal{R}$ is not ground weakly-normalizing. It turns out that if $\mathcal{R}$ is ground weakly-normalizing, μ-semantic completeness implies μ-canonical completeness.

Theorem 6. *Let $\mathcal{R}$ be a TRS that is ground weakly-normalizing, if $\mathcal{R}$ is μ-semantically complete, then $\mathcal{R}$ is μ-canonically complete.*

It then follows that if $\mathcal{R}$ is weakly-normalizing and not μ-canonically complete, it is not μ-semantically complete either.

4.3 μ-Sufficient Completeness

Although μ-canonical completeness and μ-semantic completeness are useful notions of completeness in CS rewriting, there are many interesting applications of CS rewriting, especially those involving infinite data structures, that are not μ-semantically complete. As an example, we present a typed version of a specification of infinite lists from [16] in Maude syntax:

```
fmod INF-LIST is protecting NAT .
  sorts Nat? List .    subsort Nat < Nat? .
  op none : -> Nat? [ctor].
  op [] : -> List [ctor].
  op _:_ : Nat List -> List [ctor strat(1 0)].
  vars M N : Nat . var L : List .
  op sel : Nat List -> Nat? .
  eq sel(0, N : L) = N .  eq sel(s(M), N : L) = sel(M, L) .
  op from : Nat -> List .
  eq from(M) = M : from(s(M)) .
  op first : Nat List -> List .
  eq first(0, L) = [] .   eq first(s(M), N : L) = N : first(M, L) .
endfm
```

The term `from(M)` represents the infinite list "$\mathtt{M} : \mathtt{M}+\mathtt{1} : \ldots$", and there are functions for obtaining the ith element in a list and the first n elements in the list. This specification is an interesting use of CS rewriting to obtain a terminating method to execute a non-terminating rewrite system. Although the equation for `from` is non-terminating, it is μ-terminating because of the strategy on ":".

The specification `INF-LIST` is not μ-canonically complete, and its canonical algebra is not isomorphic to the initial algebra of the equational theory given by its axioms. For example $\mathtt{0} : \mathtt{from}(\mathtt{s}(\mathtt{0}))$ and $\mathtt{0} : \mathtt{s}(\mathtt{0}) : \mathtt{from}(\mathtt{s}(\mathtt{s}(\mathtt{0})))$ are distinct μ-canonical terms, but $\mathtt{0} : \mathtt{from}(\mathtt{s}(\mathtt{0})) =_{\mathtt{INF-LIST}} \mathtt{0} : \mathtt{s}(\mathtt{0}) : \mathtt{from}(\mathtt{s}(\mathtt{s}(\mathtt{0})))$. In order to check properties of specifications like `INF-LIST` that are not μ-semantically complete, we therefore need techniques that analyze CS specifications directly. The case of μ-termination is well understood [7, 18, 21], and the case of μ-confluence has already been studied in [14].

Another interesting property that seems not to have been studied for CS specifications is sufficient completeness. Sufficient completeness in term-rewriting specifications means that enough equations have been defined so that all terms reduce to constructor terms. For example, a sufficiently-complete specification involving arithmetic over the natural numbers should reduce every term containing plus and times to a term containing only zero and successor.

Although simple, this definition of sufficient completeness seems *too strong* in the context of CS specifications. The reason is that the non-replacing positions of a symbol intentionally do not reduce their arguments. Accordingly, our definition of μ-sufficient completeness allows defined symbols in the non-replacing positions of canonical terms, provided that all *replacing* positions have constructor symbols.

Definition 9. *Let $\mathcal{R}$ be a ground μ-weakly-normalizing and ground μ-sort-decreasing TRS $\mathcal{R} = (\Sigma, A, R)$ with $\Sigma = (S, F, \leq)$ equipped with a indexed family of constructor symbols $C = \{C_{w,s}\}_{(w,s)\in S^*\times S}$ with each $C_{w,s} \subseteq F_{w,s}$. We say that $\mathcal{R}$ is* μ-sufficiently complete relative to *C iff for all $(\mathcal{R}, \mu)$-irreducible terms $t \in T_\Sigma$, $\mathrm{pos}^\mu(t) \subseteq \mathrm{pos}_C(t)$ where*

$$\mathrm{pos}_C(t) = \{i \in \mathrm{pos}(t) \mid t_{|i} = c(\bar{t}) \wedge c \in C_{w,s} \wedge \bar{t} \in T_{\Sigma,w}\}.$$

Note that our definition of μ-sufficient completeness reduces to the usual definition of sufficient completeness when every position is a replacing position, i.e., $\mu = \mu_\top$. Therefore, we say in this paper that a specification $\mathcal{R}$ is *sufficiently complete* relative to C iff it is $\mu_\top$-sufficiently complete relative to C.

Theorem 7. *If $\mathcal{R}$ is ground μ-weakly normalizing, μ-canonically complete, and ground sort-decreasing, then $\mathcal{R}$ is μ-sufficiently complete relative to C iff it is sufficiently complete relative to C.*

5 Checking μ-Completeness Properties

In the left-linear case, we are able to reduce the μ-canonical completeness and μ-sufficient completeness properties to an emptiness problem for *Propositional Tree Automata*, a class of tree automata introduced in [12] which is closed under Boolean operations and an equational theory. We are further able to use the results of Theorem 5 to have sufficient conditions for showing the μ-semantic completeness of $\mathcal{R}$ when $\mathcal{R}$ is left-linear, μ-weakly normalizing, μ-canonically complete, ground confluent, and ground sort-decreasing.

5.1 Propositional Tree Automata

We now define *Propositional Tree Automata*, first introduced in [12]. We extend the definition of [12] from unsorted signatures to many-sorted signatures. We also use production rules $\alpha := f(\beta_1, \ldots, \beta_n)$ in lieu of rewrite rules $f(\beta_1, \ldots, \beta_n) \rightarrow \alpha$ in the definition to reflect a change in how the rules are interpreted. The definition using rewrite rules in [12] and the definition below are equivalent when the equations in the signature are linear. They are not equivalent, in general, when considering non-linear equations such as idempotence $f(x, x) = x$. We think that the definition given below is more useful in applications involving non-linear equations, and if we did not use this formalization, we would have to restrict later results in this paper involving tree automata to the linear case.

Definition 10. *A* Propositional Tree Automaton *(PTA) is a tuple* $(\mathcal{E}, Q, \Phi, \Delta)$ *in which:*

- $\mathcal{E} = (\Sigma, E)$ *is an many-sorted equational theory with* $\Sigma = (S, F)$*;*
- $Q = \{Q_s\}_{s \in S}$ *is a* S*-indexed family of sets of* states *disjoint from the function symbols in* F*;*
- $\Phi = \{\phi_s\}_{s \in S}$ *is a* S*-indexed family of propositional formulae where the atomic propositions in* ϕ_s *are states in* Q_s*; and*
- Δ *contains* transition rules*, each with one of the following forms: (1)* $\alpha := f(\beta_1, \ldots, \beta_n)$ *where* $f \in F_{s_1 \ldots s_n, s}$*,* $\alpha \in Q_s$*, and each* $\beta_i \in Q_{s_i}$ *for* $1 \leq i \leq n$*; or (2)* $\alpha := \beta$ *where* $\alpha, \beta \in Q_s$*.*

For a term $t \in T_\Sigma$ *and state* $\beta \in Q$*, we write* $\beta :=_{\mathcal{A}} t$ *iff (1)* $t =_E f(u_1, \ldots, u_n)$ *and there is a rule* $\alpha :=_{\mathcal{A}} f(\beta_1, \ldots, \beta_n)$ *in* Δ *such that* $\beta_i :=_{\mathcal{A}} u_i$ *for* $1 \leq i \leq n$*, or (2) there is a rule* $\alpha :=_{\mathcal{A}} \beta$ *in* Δ *and* $\beta :=_{\mathcal{A}} t$*. A term* $t \in T_{\Sigma,s}$ *is* accepted by $\mathcal{A}$ *if the complete set of states that generate* t*,* $\mathrm{gen}_{\mathcal{A}}(t) = \{\, \alpha \in Q_s \mid \alpha :=_{\mathcal{A}} t \,\}$*, is a model of* ϕ_s*, i.e.* $\mathrm{gen}_{\mathcal{A}}(t) \models \phi_s$*. Boolean formulae are evaluated using their standard interpretations:*

$$P \models q \text{ if } q \in P, \quad P \models \phi_1 \vee \phi_2 \text{ if } P \models \phi_1 \text{ or } P \models \phi_2, \text{ and } \quad P \models \neg\, \phi \text{ if } P \not\models \phi.$$

The language *accepted by* $\mathcal{A}$ *is the* S*-index family* $\mathcal{L}(\mathcal{A}) = \{\mathcal{L}_s(\mathcal{A})\}_{s \in S}$ *of set of terms accepted by* $\mathcal{A}$*, i.e.,* $\mathcal{L}_s(\mathcal{A}) = \{t \in T_{\Sigma.s} \mid \mathrm{gen}_{\mathcal{A}}(t) \models \phi_s\}$*.*

Given a PTA $\mathcal{A} = (\mathcal{E}, Q, \Phi, \Delta)$ with $\mathcal{E} = (\Sigma, E)$, we let $\mathcal{A}_\varnothing$ denote the same PTA formed over the free theory with symbols in Σ, i.e., $\mathcal{A}_\varnothing = ((\Sigma, \varnothing), Q, \Phi, \Delta)$. By using grammar rules instead of rewrite rules in the definition of PTA, we are able to show the following for arbitrary equational theories that may be non-linear. This was proven for equational tree automata by Verma in [20] — the proof in this case is identical.

Lemma 2. *Given a PTA* $\mathcal{A} = (\mathcal{E}, Q, \Phi, \Delta)$ *with* $\mathcal{E} = (\Sigma, E)$*, if* $\alpha :=_{\mathcal{A}} t$*, then there must be a term* $u \in T_\Sigma$ *such that* $t =_E u$ *and* $\alpha :=_{\mathcal{A}_\varnothing} u$*.*

The languages recognized by PTA over a theory $\mathcal{E}$ recognize precisely those languages that are in the Boolean closure of regular equational tree automata languages sharing the same theory $\mathcal{E}$. This is an important property, because in general, equational tree automata are not closed under Boolean operations [12].

We can use PTA to check the CS completeness properties for a TRS $\mathcal{R} = (\Sigma, A, R)$ into the emptiness test for a PTA with the same axioms A. When A consists of any combination of associativity, commutativity, and identity axioms, the techniques from [12] can be used to check the emptiness of the corresponding PTA. It is know that the emptiness problem for PTA is decidable A contains any combination of associativity, commutativity, and identity and every associative symbol is also commutative. If however, the PTA does contain an associative symbol that is not commutative, there is a semi-algorithm in [12] that can always show non-emptiness and can often show emptiness using some techniques from machine learning. The algorithms in [12] have been integrated into Maude as part of the Maude Sufficient Completeness Checker (SCC) [10].

5.2 Checking μ-Canonical Completeness

From the definition of μ-canonical completeness, we know that $\mathcal{R}$ is not μ-canonically complete iff there is a term $t \in T_\Sigma$ that is $\mathcal{R}$-reducible and $(\mathcal{R}, \mu)$-irreducible. Therefore, we can reduce the problem of μ-canonical completeness to an emptiness problem of a PTA $\mathcal{A}$ by constructing an automaton that accepts precisely those terms $t \in T_\Sigma$ that are counterexamples.

Theorem 8. *Given a left-linear TRS $\mathcal{R} = (\Sigma, A, R)$, one can effectively construct a PTA $\mathcal{A}_{\mathsf{cc}}$ such that $\mathcal{R}$ is μ-canonically complete iff $\mathcal{L}(\mathcal{A}_{\mathsf{cc}}) = \varnothing$.*

Proof. (Sketch) Let $\Sigma = (S, F, \leq)$. $\mathcal{A}_{\mathsf{cc}}$ is a PTA with signature $\Sigma^K = (S^K, F^K)$ as defined in Definition 5. For each sort $s \in S$, $\mathcal{A}_{\mathsf{cc}}$ contains a state α_s such that for each $t \in T_{\Sigma^K}$, $\alpha_s :=_{\mathcal{A}_{\mathsf{cc}}} t$ iff $t \in T_{\Sigma,s}$. For each $k \in K$, $\mathcal{A}_{\mathsf{cc}}$ contains states r_k, r_k^μ for recognizing $\mathcal{R}$-reducible and $(\mathcal{R}, \mu)$ reducible terms respectively. The acceptance formula for each $k \in S^K$, then just becomes $\phi_k = \alpha_k \wedge r_k \wedge \neg r_k^\mu$. The full construction and correctness proof of $\mathcal{A}_{\mathsf{cc}}$ is in the Appendix. □

We have implemented an algorithm for constructing the tree automaton $\mathcal{A}_{\mathsf{cc}}$ in Maude automatically from a Maude specification, and have integrated it into the Maude Sufficient Completeness Checker [10]. If we ask the tool to check the μ-canonical completeness of the specification `FACTORIAL` given in Section 4.1, we are able to verify that it is μ-canonically complete:

```
Maude> (ccc FACTORIAL .)
Checking canonical completeness of FACTORIAL ...
Success: FACTORIAL is canonically complete.
```

If we ask the tool to check the `INF-LIST` specification, the tool generates a counterexample showing that the specification is not μ-canonically complete:

```
Maude> (ccc INF-LIST .)
Checking canonical completeness of INF-LIST ...
Failure: The term 0 : first(0,[]) is a counterexample that is
    mu-irreducible, but reducible under ordinary rewriting.
```

5.3 Checking μ-Semantic Completeness

Since we were able to check the μ-canonical completeness of a left-linear specification $\mathcal{R}$ using the results in the previous section, using the results in Theorem 5, the μ-semantic completeness of specifications can be mechanically checked by showing: (1) μ-canonical completeness with the checker in the previous section; (2) μ-terminating with a CS termination tool such as [4, 8, 17]; and (3) confluence and sort-decreasingness with a tool such as the Maude Church-Rosser checker. This allows us to show that, for example, the `FACTORIAL` specification is μ-semantically complete.

5.4 Checking μ-Sufficient Completeness

Using our definition of μ-sufficient completeness, we are able to extend the Maude Sufficient Completeness Checker in [11] to the CS case.

Theorem 9. *Given a left-linear TRS $\mathcal{R}$ that is ground μ-weakly normalizing and ground μ-sort-decreasing, one can construct a PTA $\mathcal{A}_{\mathsf{SC}}$ such that $\mathcal{R}$ is μ-sufficiently complete relative to C iff $\mathcal{L}(\mathcal{A}_{\mathsf{SC}}) = \varnothing$.*

We have also implemented an algorithm for constructing the automaton $\mathcal{A}_{\mathsf{SC}}$ from a CS Maude specification automatically. In this case, the checker succeeds on the `FACTORIAL` example, as expected:

```
Maude> (mu-scc FACTORIAL .)
Checking the mu-sufficient completeness of FACTORIAL ...
Success: FACTORIAL is mu-sufficiently complete assuming that it is
    ground mu-weakly normalizing and ground mu-sort-decreasing.
```

Running the checker on the `INF-LIST` example yields an error:

```
Maude> (mu-scc INF-LIST .)
Checking the mu-sufficient completeness of INF-LIST ...
Failure: The term sel(0,[]) is an mu-irreducible term with sort
    Nat? in INF-LIST with defined symbols in replacement positions.
```

It turns out that the rewrite system given in [16] was missing equations for defining `sel` and `first` when the second argument was the empty list. If we add the equations "`sel(M, []) = none`" and "`first(M, []) = []`" to the Maude specification, the μ-sufficient completeness check succeeds.

6 Related Work and Conclusions

An earlier paper by Lucas [14] has a section on relating the $\mathcal{R}$ and $(\mathcal{R}, \mu)$-canonical forms. In one of the results, a replacement map $\mu^B_{\mathcal{R}}$ is constructed from $\mathcal{R}$ and the subset of symbols $B \subseteq F$, and results show that $\mathcal{R}$ is μ-canonically complete if the $(\mathcal{R}, \mu)$-irreducible terms are in T_B, and $\mu \supseteq \mu^B_{\mathcal{R}}$. This condition is sufficient to show that the `FACTORIAL` example is μ-canonically complete. However it is easy to give examples where $\mathcal{R}$ is μ-canonically complete, but $\mu \not\supseteq \mu^B_{\mathcal{R}}$. One aspect of our results is that, since we have a decision procedure for μ-canonical completeness, by varying the replacement map μ, one can find all *minimal* replacement maps μ for which $\mathcal{R}$ is μ-canonically complete.

It would be useful to investigate the relationships between the work we have presented here and infinite rewriting and infinite normal forms, e.g., [1, 3], which has been extended to the CS in [15]. In particular, it seems interesting to investigate the relations between algebras of finite and infinite terms, and the extension of sufficient completeness to infinite normal forms.

We have proposed a new model-theoretic semantics for order-sorted CS specifications in the form of the μ-canonical term algebra $\mathrm{Can}^{\mu}_{\mathcal{R}}$. And we have investigated three notions of CS completeness: (1) canonical μ-completeness with respect to canonical forms; (2) semantic μ-completeness with respect to equational deduction; and (3) sufficient μ-completeness with respect to constructors. We have also proposed and implemented decision procedures based on propositional tree automata that, under reasonable assumptions on the CS specification (which can be discharged by other existing tools), ensure that it satisfies the different μ-completeness properties. These results provide new ways of reasoning formally about CS equational programs, not only to allow a programmer

to check that his/her program behaves as desired, but also to prove properties: for example it *is* sound to use an inductive theorem prover to reason about a μ-semantically complete CS program, whereas in general such reasoning may be unsound, since $\mathrm{Can}^{\mu}_{\mathcal{R}}$ may not satisfy the equations of $\mathcal{R}$ and may have "junk" data outside the image from the initial algebra.

We think that it would be useful to extend the concepts and results presented here to: (1) more general conditional CS specifications in membership equational logic [19]; (2) CS specifications with non-left-linear rules, for which the tree automata techniques proposed in [13] could be quite useful; and (3) infinite μ-normal forms and infinitary rewriting, as discussed above.

References

[1] G. Boudol. Computational semantics of term rewriting systems. *Algebraic methods in semantics*, pages 169–236, 1986.

[2] M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet, and C. Talcott. *All About Maude*. Springer LNCS Vol. 4350, 2007. To appear.

[3] N. Dershowitz, S. Kaplan, and D. A. Plaisted. Rewrite, rewrite, rewrite, rewrite, rewrite, ... *Theor. Comput. Sci.*, 83(1):71–96, 1991.

[4] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving termination of membership equational programs. In *Proc. of PEPM*, pages 147–158. ACM, 2004.

[5] K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.

[6] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Proc. of POPL 1985*, pages 52–66. ACM, 1985.

[7] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *J. Funct. Program.*, 14(4):379–427, 2004.

[8] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. of IJCAR*, volume 4130 of *LNCS*, pages 281–286. Springer, 2006.

[9] J. Goguen and R. Diaconescu. An oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(3):363–392, 1994.

[10] J. Hendrix, J. Meseguer, and H. Ohsaki. A sufficient completeness checker for linear order-sorted specifications modulo axioms. In *Proc. of IJCAR*, volume 4130 of *LNCS*, pages 151–155. Springer, 2006.

[11] J. Hendrix, H. Ohsaki, and J. Meseguer. Sufficient completeness checking with propositional tree automata. Technical Report UIUCDCS-R-2005-2635, University of Illinois, 2005. Available at: `http://maude.cs.uiuc.edu/tools/scc/`.

[12] J. Hendrix, H. Ohsaki, and M. Viswanathan. Propositional tree automata. In *Proc. of RTA*, volume 4098 of *LNCS*, pages 165–174. Springer, 2006.

[13] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Tree automata with equality constraints modulo equational theories. In *Proc. of IJCAR*, volume 4130 of *LNCS*, pages 557–571. Springer, 2006.

[14] S. Lucas. Context-sensitive computations in functionnal and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1), 1998.

[15] S. Lucas. Transfinite rewriting semantics for term rewriting systems. In *Proc. of RTA*, volume 2051 of *LNCS*, pages 216–230. Springer, 2001.

[16] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):294–343, 2002.

[17] S. Lucas. mu-term: A tool for proving termination of context-sensitive rewriting. In *Proc. of RTA*, volume 3091 of *LNCS*, pages 200–209. Springer, 2004.
[18] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782–1846, 2006.
[19] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proc. of WADT*, volume 1376 of *LNCS*, pages 18–61. Springer, 1997.
[20] K. N. Verma. Two-way equational tree automata for AC-like theories: Decidability and closure properties. In *Proc. of RTA*, volume 2706 of *LNCS*, pages 180–196. Springer, 2003.
[21] H. Zantema. Termination of context-sensitive rewriting. In *Proc. of RTA*, volume 1232 of *LNCS*, pages 172–186. Springer, 1997.

Appendix

Proofs for Section 3

Proposition 1. *If $\mathcal{R} = (\Sigma, A, R)$ with $\Sigma = (S, F, \leq)$ is ground μ-weakly normalizing, ground μ-confluent, and ground μ-sort-decreasing, then the family of functions $\{q_s : \mathrm{Can}^{\mu}_{\mathcal{R},s} \to T_{\Sigma/A\cup R,s}\}_{s\in S}$ with $q_s : [t]_A \mapsto [t]_{A\cup R}$ defines a surjective Σ-homomorphism $q : \mathrm{Can}^{\mu}_{\mathcal{R}} \to T_{\Sigma/A\cup R}$.*

Proof. The mapping q is a Σ-homomorphism, because for each $f \in F_{s_1\ldots s_n,s}$ and $[t_1]_A \in \mathrm{Can}^{\mu}_{\mathcal{R},s_1}, \ldots, [t_n]_A \in \mathrm{Can}^{\mu}_{\mathcal{R},s_n}$, we can choose $u_i \in [t_i]_A \cap T_{\Sigma,s_i}$ for $i \in \{1, \ldots, n\}$, and then we have,

$$T_{\Sigma/A\cup R,f}(q([t_1]_A), \ldots, q([t_n]_A)) \\ = [f(u_1, \ldots, u_n)]_{A\cup R} = q(\mathrm{Can}^{\mu}_{\mathcal{R},f}([t_1]_A, \ldots, [t_n]_A))$$

as $\mathrm{Can}^{\mu}_{\mathcal{R},f}([t_1]_A, \ldots, [t_n]_A) \subseteq [f(u_1, \ldots, u_n)]_{A\cup R}$ by the soundness of $\to_{\mathcal{R},\mu}$. □

Theorem 3. *If $\mathcal{R}$ is ground μ-weakly normalizing and μ-canonically complete, then $\mathcal{R}$ is ground weakly-normalizing.*

Proof. If $\mathcal{R} = (\Sigma, A, R)$ is ground μ-weakly normalizing, then for each term $t \in T_\Sigma$, there is a $(\mathcal{R}, \mu)$-irreducible term $u \in T_\Sigma$ such that $t \to^{!}_{\mathcal{R},\mu} u$. Since $\mathcal{R}$ is μ-canonically complete it follows that u is $\mathcal{R}$-irreducible as well, so $t \to^{!}_{\mathcal{R}} u$. □

Proofs for Section 4.1

Theorem 1. *A TRS $\mathcal{R}$ that is ground μ-weakly normalizing, μ-canonically complete, and ground confluent, then for $t, u \in T_\Sigma$, $t \downarrow_{\mathcal{R}} u$ iff $t \downarrow^{\mu}_{\mathcal{R}} u$.*

Proof. Of course $t \downarrow^{\mu}_{\mathcal{R}} u \Rightarrow t \downarrow_{\mathcal{R}} u$. To see $t \downarrow_{\mathcal{R}} u \Rightarrow t \downarrow^{\mu}_{\mathcal{R}} u$, since $\mathcal{R} = (\Sigma, A, R)$ is μ-weakly normalizing, there must be $(\mathcal{R}, \mu)$-irreducible terms $t', u' \in T_\Sigma$ such that $t \to^{!}_{\mathcal{R},\mu} t'$ and $u \to^{!}_{\mathcal{R},\mu} u'$. $\mathcal{R}$ is μ-canonically complete, so both t' and u' must be $\mathcal{R}$-irreducible as well. This implies that $t' =_A u'$ by the ground confluence of $\mathcal{R}$. □

Corollary 1. *If $\mathcal{R}$ is ground μ-weakly normalizing, μ-canonically complete, and ground confluent, then $\mathcal{R}$ is ground μ-confluent.*

Proof. If $t \to^{*}_{\mathcal{R},\mu} u$ and $t \to^{*}_{\mathcal{R},\mu} v$, we have $u \downarrow_{\mathcal{R}} v$ by the ground confluence of $\mathcal{R}$. Thus, $u \downarrow^{\mu}_{\mathcal{R}} v$ by Theorem 1. □

Theorem 2. *If $\mathcal{R}$ is ground μ-normalizing, μ-canonically complete, and ground sort-decreasing, then $\mathcal{R}$ is ground μ-sort-decreasing.*

Proof. Assume $t \in T_{\Sigma,s}$ and $t \to^*_{\mathcal{R},\mu} u$ with $u \notin T_{\Sigma,s}$. Since $\mathcal{R}$ is ground μ-normalizing and μ-canonically complete, there is an $\mathcal{R}$-irreducible term $u' \in T_\Sigma$ such that $u \to^*_{\mathcal{R},\mu} u'$. Because $\mathcal{R}$ is ground sort-decreasing, there must be a term $v \in T_{\Sigma,s}$ such that $u' \to^*_{\mathcal{R}} v$. However, u' is $\mathcal{R}$-irreducible, so $u' =_A v$. It follows that $u \to^*_{\mathcal{R},\mu} v$, and $\mathcal{R}$ is ground μ-sort-decreasing. □

Proofs for Section 4.2

Theorem 4. *If $\mathcal{R}$ is ground μ-weakly normalizing, ground μ-confluent, and ground μ-sort-decreasing, then μ-semantically complete iff $\mathrm{Can}^\mu_\mathcal{R}$ is isomorphic to $T_{\Sigma/A\cup R}$.*

Proof. Let $\mathcal{R} = (\Sigma, A, R)$ with $\Sigma = (S, F, \leq)$. If $\mathrm{Can}^\mu_\mathcal{R}$ is isomorphic to $T_{\Sigma/A\cup R}$, they must satisfy the same equations, so $\mathcal{R}$ is μ-semantically complete. On the other hand, the equivalences $t =_{A\cup R} u \iff t\downarrow^\mu_\mathcal{R} u \iff t\,!^\mu_\mathcal{R} =_A u\,!^\mu_\mathcal{R}$ imply that the surjective map $q_s : [t]_A \mapsto [t]_{A\cup R}$ is injective for each $s \in S$, and therefore bijective. To see that $q : \mathrm{Can}^\mu_\mathcal{R} \to T_{\Sigma/A\cup R}$ is an Σ-isomorphism, we should check that q^{-1} is also a Σ-homomorphism. Consider $[t]_{A\cup R}$, then $q^{-1}([t]_{A\cup R}) = [t\,!^\mu_\mathcal{R}]_A$, and for any $f : s_1 \dots s_n \to s$ in Σ, given $[t_i] \in T_{\Sigma/A\cup R,s_i}$ for $1 \leq i \leq n$, we can choose a term $t_i\,!^\mu_\mathcal{R} \in [t_1]_{A\cup R} \cap T_{\Sigma,s_i}$ for $1 \leq i \leq n$, so that

$$\begin{aligned} q^{-1}(T_{\Sigma/A\cup R,f}([t_1],\dots,[t_n]) &= q^{-1}\left([f(t_1\,!^\mu_\mathcal{R},\dots,t_n\,!^\mu_\mathcal{R})]_{A\cup R}\right) \\ &= [f(t_1\,!^\mu_\mathcal{R},\dots,t_n\,!^\mu_\mathcal{R})\,!^\mu_\mathcal{R}]_A = \mathrm{Can}^\mu_{\mathcal{R},f}\left(q^{-1}([t_1]),\dots,q^{-1}([t_n])\right). \end{aligned}$$

□

Theorem 5. *A TRS $\mathcal{R}$ that is ground μ-weakly normalizing, μ-canonically complete, ground confluent, and ground sort-decreasing is μ-semantically complete.*

Proof. By Theorem 3, $\mathcal{R}$ is ground weakly-normalizing. It follows that $t =_{R\cup A} u$ iff $t\downarrow_\mathcal{R} u$ since $\mathcal{R}$ is ground confluent and sort-decreasing as well. By Theorem 1, we have that $t\downarrow_\mathcal{R} u$ iff $t\downarrow^\mu_\mathcal{R} u$. So $\mathcal{R}$ is μ-semantically complete. □

Corollary 2. *If $\mathcal{R}$ is ground μ-weakly normalizing, ground μ-canonically complete, ground confluent, and ground sort-decreasing, then $\mathrm{Can}^\mu_\mathcal{R}$ and $\mathrm{Can}_\mathcal{R}$ are both well-defined and isomorphic to $T_{\Sigma/A\cup R}$.*

Proof. We know that $\mathrm{Can}_\mathcal{R}$ is well-defined and isomorphic to $T_{\Sigma/A\cup R}$ since $\mathcal{R}$ is weakly normalizing by Theorem 3. We also know by Theorems 5 and 4, that $\mathrm{Can}^\mu_\mathcal{R}$ is isomorphic to $T_{\Sigma/A\cup R}$. Therefore, all three algebras are isomorphic. □

Theorem 6. *Let $\mathcal{R}$ be a TRS that is ground weakly-normalizing, if $\mathcal{R}$ is μ-semantically complete, then $\mathcal{R}$ is μ-canonically complete.*

Proof. Let $\mathcal{R} = (\Sigma, A, R)$. Let $t \in T_\Sigma$ be a $(\mathcal{R}, \mu)$-irreducible term. Because $\mathcal{R}$ is weakly-normalizing, there is a $\mathcal{R}$-irreducible term $u \in T_\Sigma$ such that $t \to_{\mathcal{R}}^* u$. We know that $t =_{A \cup R} u$ by the soundness of rewriting, and since $\mathcal{R}$ is μ-semantically complete, we have $t \downarrow_{\mathcal{R}}^{\mu} u$. However, both t and u are $(\mathcal{R}, \mu)$-irreducible, so $t =_A u$. Since u is $\mathcal{R}$-irreducible as well, it follows that t is $\mathcal{R}$-irreducible. □

Proofs for Section 4.3

Theorem 7. *If $\mathcal{R}$ is ground μ-weakly normalizing, μ-canonically complete, and ground sort-decreasing, then $\mathcal{R}$ is μ-sufficiently complete relative to C iff it is sufficiently complete relative to C.*

Proof. First, it should be noted that our definition of μ-sufficiently completness only applies if $\mathcal{R}$ is μ-sort decreasing. However, given the conditions, we know that it is μ-sort-decreasing by Theorem 2.

If $\mathcal{R}$ is μ-canonically complete, then the $\mathcal{R}$-canonical forms and $(\mathcal{R}, \mu)$-canonical forms are the same. It easily follows then that if $\mathcal{R}$ is sufficiently complete relative to C, $\mathcal{R}$ must be μ-sufficiently complete. On the other hand, suppose that $\mathcal{R}$ is μ-sufficiently complete relative to C. The characterization of sufficient completeness given by Theorem 5 of [11] shows that $\mathcal{R}$ is sufficiently complete iff each term $t \in T_{\Sigma,s}$ with a defined symbol at the root and constructor subterms is $\mathcal{R}$-reducible. However, since $\mathcal{R}$ is μ-sufficiently complete and the root must be a replacing position, we know that each $(\mathcal{R}, \mu)$-irreducible term $t \in T_\Sigma$ must have a constructor at the root. So any term with a defined symbol at the root must be $(\mathcal{R}, \mu)$-reducible, and consequentally, $\mathcal{R}$-reducible due to $\mathcal{R}$ being μ-canonically complete. □

Proofs for Section 5

In order to recognize ground terms that are $\mathcal{R}$ and $(\mathcal{R}, \mu)$-reducible in a tree automata framework, we will additionally need to recognize terms that match subterms appearing in the left-hand side of rules in $\mathcal{R}$.

Definition 11. *Given an order-sorted and left-linear TRS $\mathcal{R} = (\mathcal{E}, R)$, the* intermediate terms *$I_\mathcal{R}$ denote the set of quantified non-variable strict subterms appearing in the left-hand side of a rule in $\mathcal{R}$ along with the sort-constraints on the variables in the terms, i.e.,*

$$I_\mathcal{R} = \{t \mid C[t] \in \mathrm{lhs}(R) \wedge t \notin X \wedge C \neq \Box\}.$$

We now give the construction of $\mathcal{A}_{\mathtt{cc}}$:

Theorem 8. *Given a left-linear TRS $\mathcal{R} = (\Sigma, A, R)$, one can effectively construct a PTA $\mathcal{A}_{\mathsf{cc}}$ such that $\mathcal{R}$ is μ-canonically complete iff $\mathcal{L}(\mathcal{A}_{\mathsf{cc}}) = \varnothing$.*

Proof. Let $\Sigma = (S, F, \leq)$ The basic idea behind the automaton $\mathcal{A}_{\mathsf{cc}}$, is that it contains states that recognize terms with a particular sort in T_Σ/A, terms that match subterms in $I_\mathcal{R}$, and $\mathcal{R}$-reducible and $\mathcal{R}$-reducible terms. To simplify the full definition of $\mathcal{A}_{\mathsf{cc}}$, we let $\alpha_{x_s} = \alpha_s$. Formally, $\mathcal{A}_{\mathsf{cc}} = (\Sigma^K, Q, \Phi, \Delta)$ where:

- $\Sigma^K = (S^K, F^K)$ is defined as in Definition 5;
- $Q = \{Q_k\}_{k \in S^K}$ with $Q_k = \{r_k, r_k^\mu\} \cup \{\alpha_s \mid s \in S\} \cup \{\alpha_u \mid u \in I_\mathcal{R} \cap T_{\Sigma,k}\}$;
- $\Phi = \{\phi_k\}_{k \in S^K}$ with $\phi_k = \alpha_k \wedge r_k \wedge \neg r_k^\mu$;
- The rules in Δ are defined as follows:

$$\begin{aligned}
\Delta = \{&\alpha_s := f(\alpha_{s_1}, \ldots, \alpha_{s_n}) \mid f \in F_{s_1 \ldots s_n, s}\} \\
&\cup \{\alpha_{s'} := \alpha_s \mid s, s' \in S \wedge s < s'\} \\
&\cup \{\alpha_{f(t_1,\ldots,t_n)} := f(\alpha_{t_1}, \ldots, \alpha_{t_n}) \mid f(t_1, \ldots, t_n) \in I_\mathcal{R}\} \\
&\cup \{r_k^\mu := f(\alpha_{t_1}, \ldots, \alpha_{t_n}) \mid f(t_1, \ldots, t_n) \in T_{\Sigma,k} \cap \mathrm{lhs}(R)\} \\
&\cup \{r_k := r_k^\mu \mid k \in S^K\} \\
&\cup \{r_k^\mu := f(\alpha_{k_1}, \ldots, r_{k_i}^\mu, \ldots, \alpha_{k_n}) \mid f \in F^K_{k_1 \ldots k_n, k} \wedge i \in \mu(f)\} \\
&\cup \{r_k := f(\alpha_{k_1}, \ldots, r_{k_i}, \ldots, \alpha_{k_n}) \mid f \in F^K_{k_1 \ldots k_n, k} \wedge i \in \{1, \ldots, n\}\}.
\end{aligned}$$

By induction on $t \in T_\Sigma$, we have $\alpha_s :=_{\mathcal{A}_{\mathsf{cc}\varnothing}} t$ for each term $t \in T_{\Sigma,s}$. A similar inductive argument allows us to show that $\alpha_u :=_{\mathcal{A}_{\mathsf{cc}\varnothing}} t$ iff there is a substitution θ such that $t = u\theta$. This, in turn, allows us to show that $r_k^\mu :=_{\mathcal{A}_{\mathsf{cc}\varnothing}} t$ iff there is a μ-replacing Σ^K-context C, Σ-substitution θ, and rule $l \to r$ in R such that $t = C[l\theta]$, and to show that $r_k :=_{\mathcal{A}_{\mathsf{cc}\varnothing}} t$ iff there is a context C, substitution θ, and rule $l \to r$ in R such that $t = C[l\theta]$. It then follows from Lemmas 1 and 2 that $t \in \mathcal{L}_k(A)$ iff t is a $\mathcal{R}$-reducible and $(\mathcal{R}, \mu)$-irreducible term in $t \in T_{\Sigma,k}$. □

Theorem 9. *Given a left-linear TRS $\mathcal{R} = (\Sigma, A, R)$ that is ground μ-weakly normalizing and ground μ-sort-decreasing, one can construct a PTA $\mathcal{A}_{\mathsf{sc}}$ such that $\mathcal{R}$ is μ-sufficiently complete relative to C iff $\mathcal{L}(\mathcal{A}_{\mathsf{sc}}) = \varnothing$.*

Proof. Let $\Sigma = (S, F, \leq)$. We construct $\mathcal{L}(\mathcal{A}_{\mathsf{sc}})$ so that it accepts a $(\mathcal{R}, \mu)$-irreducible term $t \in T_\Sigma$ with a replacement position $i \in \mathrm{pos}^\mu(t)$ that is not in $\mathrm{pos}_C(t)$. Specifically, $\mathcal{A}_{\mathsf{sc}} = (\Sigma^K, Q, \Phi, \Delta)$ where

- $\Sigma^K = (S^K, F^K)$ is defined as in Definition 5;
- $Q = \{Q_k\}_{k \in S^K}$ with $Q_k = \{r_k^\mu\} \cup \{\alpha_s, c_s \mid s \in S\} \cup \{\alpha_u \mid u \in I_\mathcal{R} \cap T_{\Sigma,k}\}$;
- $\Phi = \{\phi_k\}_{k \in K}$ with $\phi_k = \neg r_k^\mu \wedge \bigvee_{s \in [k]_\leq} \alpha_s \wedge \neg c_s$; and

– The rules in Δ are defined as follows:

$$\begin{aligned}\Delta = &\{\alpha_s := f(\alpha_{s_1}, \ldots, \alpha_{s_n}) \mid f \in F_{s_1 \ldots s_n, s}\} \\ &\cup \{c_s := c(\mathrm{rep}^{\mu}_{c,1}(s_1), \ldots, \mathrm{rep}^{\mu}_{c,n}(s_n)) \mid c \in C_{s_1 \ldots s_n, s}\} \\ &\cup \{\alpha_{s'} := \alpha_s, c_{s'} := c_s \mid s, s' \in S \wedge s < s'\} \\ &\cup \{\alpha_{f(t_1,\ldots,t_n)} := f(\alpha_{t_1}, \ldots, \alpha_{t_n}) \mid f(t_1, \ldots, t_n) \in I_{\mathcal{R}}\} \\ &\cup \{r^{\mu}_k := f(\alpha_{t_1}, \ldots, \alpha_{t_n}) \mid f(t_1, \ldots, t_n) \in T_{\Sigma,k} \cap \mathrm{lhs}(R)\} \\ &\cup \{r^{\mu}_k := f(\alpha_{k_1}, \ldots, r^{\mu}_{k_i}, \ldots, \alpha_{k_n}) \mid f \in F^K_{k_1 \ldots k_n, k} \wedge i \in \mu(f)\} \\ &\quad \text{where } \mathrm{rep}^{\mu}_{c,i}(s) \text{ equals } c_s \text{ if } i \in \mu(c), \text{ and } \alpha_s \text{ otherwise.}\end{aligned}$$

The rest of our proof is similar to that used in Theorem 8. By induction on $t \in T_{\Sigma^K}$, we have $\alpha_s :=_{\mathcal{A}_{\mathrm{sc}\varnothing}} t$ for each term $t \in T_{\Sigma,s}$. This allows us to show that for $t \in T_{\Sigma^K}$, we have $\alpha_u :=_{\mathcal{A}_{\mathrm{cc}\varnothing}} t$ iff there is a substitution θ such that $t = u\theta$. This, in turn, allows us to show that $r^{\mu}_k :=_{\mathcal{A}_{\mathrm{cc}\varnothing}} t$ iff there is a μ-replacing Σ^K-context C, Σ-substitution θ, and rule $l \to r$ in R such that $t = C[l\theta]$. An inductive argument also allows us to show that for $t \in T_{\Sigma^K}$, $c_s := t$ iff $t \in T_{\Sigma,s}$ and $\mathrm{pos}^{\mu}(t) \subseteq \mathrm{pos}_C(t)$. It follows that $t \in \mathcal{L}_k(A)$ iff t is a $(\mathcal{R}, \mu)$-irreducible term in for $T_{\Sigma/A,s}$ with $\mathrm{pos}^{\mu}(t) \not\subseteq \mathrm{pos}_C(t)$. □