

Sufficient Completeness Checking with Propositional Tree Automata*

Joe Hendrix¹, José Meseguer¹, and Hitoshi Ohsaki²

¹ University of Illinois at Urbana-Champaign
{jhendrix,meseguer}@uiuc.edu

² National Institute of Advanced Industrial Science and Technology
ohsaki@ni.aist.go.jp

Abstract. A specification is sufficiently complete when its functions are fully defined on all relevant data. This is an important property to verify for both debugging and formal reasoning. Although sufficient completeness is well-studied on unsorted and unconditional specifications, it has not been extensively studied in the more expressive logics supported by advanced equational specification and programming languages. In this work, we extend sufficient completeness methods to handle more expressive specifications involving: (i) partiality; (ii) conditional equations; and (iii) deduction *modulo* axioms. Specifically, we give useful characterizations of the sufficient completeness property for membership equational logic (MEL) specifications having features (i)–(iii). This characterization of sufficient completeness allows us to extend the soundness of the method in [10] to a larger class of specifications. Furthermore, this characterization opens the door to new tree automata-based methods for checking order-sorted specifications with rewriting modulo equations. Using recent results on *Propositional Tree Automata*, a class of tree automata closed under Boolean operations and an equational theory, we are able to check the sufficient completeness of many order-sorted and linear specifications with rewriting modulo any combination of associativity, commutativity, and identity. The methods presented here can serve as a basis for building a next-generation sufficient completeness tool for MEL specifications having features (i)–(iii). These features are widely used in practice, and are supported by languages such as Maude and other advanced specification and equational programming languages.

Keywords: sufficient completeness, equational logic, tree automata

1 Introduction

Sufficient completeness of an equational specification means that enough equations have been specified, so that the functions of the algebraic data type defined by the specification are fully defined on all relevant data elements. This is an important property to check, both to debug and formally reason about specifications and equational programs. For example, many inductive theorem proving

* Research supported by ONR Grant N00014-02-1-0715.

techniques are based on the constructors building up the data and crucially depend on the specification being sufficiently complete.

In practice, there is a need to have expressive equational specification formalisms that match the needs of real applications, and correspondingly a need to extend sufficient completeness methods to handle such formalisms. This work presents new contributions extending sufficient completeness methods in several useful directions, namely: (i) to handle partiality; (ii) to allow conditional specifications; and (iii) to support equational reasoning *modulo* axioms. These extensions are needed in practice because: (i) functions defined on data types are often *partial*; (ii) many languages support conditional specifications; and (iii) functions often assume algebraic properties of their underlying data. For example, functions on sets or multisets are much more simply specified using the fact that set and multiset union are associative and commutative.

Of course, there is tension between expressiveness of specifications and decidability of sufficient completeness. It is well-known that checking sufficient completeness is in general undecidable even for unconditional specifications [8, 9]. Partiality makes decidability even harder to get, and in the presence of associativity axioms decidability is lost for non-linear confluent and terminating equations [17]. In our view, the best way to deal with this tension is *not to give up* on the expressive specifications that are often needed in practice and for which sufficient completeness is undecidable. We advocate a two-pronged approach. First, the sufficient completeness problem should be studied for increasingly more expressive formalisms, and the set of decidable specifications should likewise be expanded as much as possible. Second, sufficient completeness checking algorithms should be coupled with inductive theorem proving techniques that can discharge proof obligations generated when the input specification falls outside of the decidable classes. We refer the reader to [10, 15] for ideas on coupling sufficient completeness and inductive theorem proving.

In this paper, we focus on advancing the first prong in several ways. Our first contribution is to characterize sufficient completeness in a more general setting to support the extensions (i)–(iii) mentioned above. For this purpose, we use membership equational logic (MEL) [3, 20] to allow conditional specification of partial functions (see [20] for a survey of partial specification formalisms and the use of MEL in this context). In MEL atomic sentences are either equations $t = t'$, or memberships $t : s$ stating that a term t has a sort s , where t having a sort is equivalent to t being *defined*. The key idea is that a partial function's domain is axiomatized by conditional membership axioms. We precisely define the sufficient completeness property for conditional MEL specifications which can have extra variables in their conditions and can be defined modulo a set E of unconditional equations. We define sufficient completeness for both MEL theories of this kind and their corresponding conditional rewrite theories when the equations are used as rewrite rules. We also characterize specifications for which both notions coincide. Finally for a large class of specifications, we give conditions equivalent to sufficient completeness which can be the basis of a checking algorithm. These theoretical developments directly apply to the analysis of functional modules

in the Maude language [4], which are MEL specifications supporting deduction modulo axioms such as associativity, commutativity, and identity.

Our second contribution is to show how sufficient completeness for left-linear, unconditional, order-sorted specifications with rewriting modulo axioms can be cast as an equational tree automata decision problem. Although sufficient completeness as a tree automata decision problem has been studied before in the unsorted free case, this has not been done before in studying sufficient completeness with rewriting modulo axioms. This decision problem may still not be decidable. However recent work has led to a decision procedure when considering specifications with associative and commutative (AC) symbols, and a semi-decision procedure for associative symbols (A). The A case is undecidable, but the semi-decision procedure will always find sufficient-completeness counterexamples if they exist, and it can show sufficient completeness if certain regularity conditions are met. In fact, every Maude specification we have analyzed has satisfied these regularity conditions. The algorithm can be further extended to support identity axioms. As associativity, commutativity, and identity are the main equational axioms that Maude is capable of rewriting modulo, these theoretical results form the basis of a new sufficient completeness checker [11] for order-sorted Maude specifications.

Related Work

Instead of a comprehensive survey on sufficient completeness, we mention some related work to place things in context. Sufficient completeness of MEL specifications was first studied in [3]. The definition and methods presented in this paper substantially extend and generalize those in [10], which in turn had generalized those in [3], allowing a much wider class of MEL specifications to be checked. Sufficient completeness itself goes back to Guttag's thesis [8] (but see [9] for a more accessible treatment). An early algorithm for handling unconditional linear specifications is due to Nipkow and Weikum [21]. For a good review of the literature up to the late 80s, as well as some important decidability/undecidability and complexity results, see [16, 17]. A more recent development is the casting of sufficient completeness as tree automata (TA) decision problems: see Chapter 4 of [5] and references there. Our work can be seen as a further contribution to the TA approach to sufficient completeness, particularly in proposing PTA as a new tree automata framework ideally suited for sufficient completeness checking and in extending TA methods and results to the equational cases.

In addition to the TA approach, a widely used alternative approach to sufficient completeness is based on the incremental constructor-based narrowing of patterns. Sufficient completeness checkers in this category include Spike [1], the RRL [15] theorem prover, and an earlier Maude Sufficient Completeness Checker (SCC), which was integrated with the Maude inductive theorem prover [10]. Although constructor-based narrowing methods have been extended to the AC case in the context of unsorted specifications by Jouannaud and Kounalis [14], it seems much harder to apply this extension in the context of order-sorted or MEL specifications. For this reason, an important practical motivation of this work

has been to generalize the earlier narrowing-based Maude SCC, which cannot handle equational axioms other than commutativity, by a next-generation tool based on the PTA methods presented here.

Another recent approach [2] attempts to combine the tree automata and narrowing approaches with integration to a theorem prover to handle conditional, constrained rewrite specifications. This work is able to check the sufficient completeness of many-sorted rewrite specifications for which there may be conditional rules involving defined symbols and *constrained* rules involving constructor symbols. It accomplishes this by first constructing a constrained tree grammar to recognize irreducible constructor terms, and using the tree grammar during narrowing. Like the work in [10], this work targets the case of syntactic rewriting, but does not address specifications with rewriting modulo axioms.

The paper is organized as follows. In sections 2 and 3, we recall membership equational logic, and introduce a very expressive class of rewrite systems called conditional equational rewriting/membership (CERM) systems. We define sufficient completeness in both these contexts, and give conditions under which these definitions coincide. In Section 4, we give conditions on which to base methods for checking sufficient completeness. Finally, in sections 5, we show how the sufficient completeness problem for left-linear, order-sorted specifications can be formulated as a Propositional Tree Automata (PTA) decision problem. This allows us to check sufficient completeness automatically using recent tree automata results in [13].

2 Preliminaries

Membership equational logic (MEL) is essentially Horn logic with equality and unary predicates. The logic has two levels of typing : *kinds* type operator declarations in a signature, and *sorts* type terms using membership axioms in a particular theory. We review the basic terminology of MEL below (see [3] for a more comprehensive introduction).

A *membership equational logic* (MEL) *signature* Σ is a triple $\Sigma = (K, F, S)$, in which: K is a set of *kinds*; $F = \{F_{\vec{k},k}\}_{(\vec{k},k) \in K^* \times K}$ is a K -kinded family of function symbols such that $F_{\vec{k},k}$ and $F_{\vec{k},k'}$ are disjoint for distinct $k, k' \in K$; and $S = \{S_k\}_{k \in K}$ is a disjoint K -kinded family of sets of *sorts*. A K -kinded list of variables $\vec{x} = x_1 : k_1, \dots, x_n : k_n$ is Σ -*distinct* when the x_1, \dots, x_n are pairwise disjoint and also distinct from any constant in F , but the kinds k_1, \dots, k_n in the list can be repeated. For each $k \in K$, $T_\Sigma(\vec{x})_k$ denotes the set of well-kinded terms with kind k formed from the function symbols in F and variables in \vec{x} . $T_\Sigma(\vec{x}) = \bigcup_{k \in K} T_\Sigma(\vec{x})_k$ denotes the set of all well-kinded terms. The set of all *ground terms*, T_Σ , and the set of all ground terms with kind k , $T_{\Sigma,k}$, are sets of terms without variables. The function $\text{vars} : T_\Sigma(\vec{x}) \rightarrow \mathcal{P}(\vec{x})$ is the mapping from a term $t \in T_\Sigma(\vec{x})$ to the variables appearing in t . Let $\square : k$ be a distinguished variable called a *hole*. A *k-context* is a term $C \in T_\Sigma(\vec{x}, \square : k)$ with a single occurrence of the variable $\square : k$. For any $t \in T_\Sigma(\vec{x})_k$, we denote by $C[t] \in T_\Sigma(\vec{x})$ the term obtained by replacing \square by t .

A $\Sigma(\vec{x})$ -equation is a formula $t = u$ with $t, u \in T_\Sigma(\vec{x})_k$ for some $k \in K$. A $\Sigma(\vec{x})$ -membership is a formula $t : s$ with $t \in T_\Sigma(\vec{x})_k$ and $s \in S_k$. Σ -sentences are universally quantified Horn clauses of the form $(\forall \vec{x}) A$ **if** $A_1 \wedge \dots \wedge A_n$ where A and A_i ($1 \leq i \leq n$) are either $\Sigma(\vec{x})$ -equations or $\Sigma(\vec{x})$ -memberships. If A is a $\Sigma(\vec{x})$ -equation, the sentence is a *conditional equation*; if A is a $\Sigma(\vec{x})$ -membership, the sentence is a *conditional membership*. A MEL theory is a pair $\mathcal{E} = (\Sigma, \Gamma)$ with Σ a MEL signature and Γ a set of Σ -sentences. As explained in [3, 20], there is a sound and complete inference system to derive all theorems of a MEL theory (Σ, Γ) .

3 Conditional Equational Rewrite/Membership Systems

In equational specification languages such as Maude, the equations are interpreted in two different ways: most of the equations are treated as oriented rewrite simplification rules, but the matching engine has been extended to match terms modulo common equations like associativity, commutativity, and identity. Our first goal in this paper is to formalize a rewrite system in this paper that captures the operational semantics of rewriting with memberships. This is because sufficient completeness checking techniques require considering the rewrite system associated to an equational specification. Our approach likewise requires considering a rewrite system that supports memberships and conditional rewriting modulo axioms.

Standard conditional term rewriting systems, also called *join* term rewriting systems, interpret equations $t = u$ appearing in a condition as join pairs $t \downarrow u$, in which t and u are considered equal if they rewrite to the same term, i.e. $\mathcal{R} \vdash t \downarrow u$ iff there exists a $v \in T_\Sigma(\vec{x})$ such that $\mathcal{R} \vdash t \rightarrow v$ and $\mathcal{R} \vdash u \rightarrow v$, where $t \rightarrow^1 t'$ denotes rewriting one step, and $t \rightarrow t'$ denotes rewriting in 0, 1, or more steps. However, in order to consider a larger class of term rewriting systems, we use *oriented* systems, in which conditions may contain formulas of the form $t \rightarrow u$. An oriented term rewriting system can simulate a join term rewriting system by introducing a new kind B , a new constant $tt : \rightarrow B$, and for each $k \in K$, a binary operator $eq : k k \rightarrow B$ with a new rule $(\forall x : k) eq(x, x) \rightarrow tt$. Join conditions of the form $t \downarrow u$ can then be expressed as oriented conditions $eq(t, u) \rightarrow tt$.

Definition 1. A conditional equational rewrite/membership (CERM) system is a tuple $\mathcal{R} = (\Sigma, E, <, \Gamma)$ in which:

- $\Sigma = (K, F, S)$ is a MEL signature where the set of terms $T_{\Sigma, k}$ with kind k is nonempty for each $k \in K$;
- E is a set of unconditional Σ -equations;
- $< = \{<_k\}_{k \in K}$ is a K -indexed family of binary relations collectively called the subsort relation such that $<_k$ is a strict order over S_k for each $k \in K$; and
- Γ is a set containing two types of rules:

$$\begin{array}{cc}
 (\forall \vec{y}) t : s \text{ **if** } \bigwedge u_i \rightarrow v_i \wedge \bigwedge w_j : s_j & (\forall \vec{y}) t \rightarrow t' \text{ **if** } \bigwedge u_i \rightarrow v_i \wedge \bigwedge w_j : s_j \\
 \text{MEMBERSHIP RULE} & \text{REWRITE RULE}
 \end{array}$$

Equivalence	$\frac{t =_E u}{t \rightarrow u}$
Replacement	$\frac{t =_E C[l\theta] \quad u_1\theta \rightarrow v_1\theta \dots u_m\theta \rightarrow v_m\theta \quad w_1\theta : s_1 \dots w_n\theta : s_n}{t \rightarrow^1 C[r\theta]}$ <p style="text-align: center; margin: 0;">with $(\forall \vec{y}) l \rightarrow r$ if $\bigwedge_{i=1}^m u_i \rightarrow v_i \wedge \bigwedge_{j=1}^n w_j : s_j$ in Γ.</p>
Transitivity	$\frac{t \rightarrow^1 u \quad u \rightarrow v}{t \rightarrow v}$
Membership	$\frac{t =_E l\theta \quad u_1\theta \rightarrow v_1\theta \dots u_m\theta \rightarrow v_m\theta \quad w_1\theta : s_1 \dots w_n\theta : s_n}{t :^0 s}$ <p style="text-align: center; margin: 0;">with $(\forall \vec{y}) l : s'$ if $\bigwedge_{i=1}^m u_i \rightarrow v_i \wedge \bigwedge_{j=1}^n w_j : s_j$ in Γ and $s' < s$</p>
Subject Reduction	$\frac{t \rightarrow u \quad u :^0 s}{t : s}$ <p style="text-align: center; margin: 0;">with $t, u, v \in T_\Sigma(\vec{x})$ and $\theta : \vec{y} \rightarrow T_\Sigma(\vec{x})$</p>

Fig. 1. Inference rules for $\mathcal{R} = (\Sigma, E, <, \Gamma)$ with respect to Σ -distinct variables \vec{x}

with \vec{y} a set of Σ -distinct variables, $t, t' \in T_\Sigma(\vec{y})_k$ and $s \in S_k$ with $k \in K$, each $u_i, v_i \in T_\Sigma(\vec{y})_{k_i}$ with $k_i \in K$, and each $w_j \in T_\Sigma(\vec{y})_{k_j}$ and $s_j \in S_{k_j}$ with $k_j \in K$.

In the rest of this section, $\mathcal{R} = (\Sigma, E, <, \Gamma)$ is a CERM system where $\Sigma = (K, F, S)$ and \vec{x} is a list of Σ -distinct variables. $\Sigma(\vec{x})$ -atomic formulas are expressions of the form $t \rightarrow u$ and $t : s$, where $t, u \in T_\Sigma(\vec{x})_k$ and $s \in S_k$ with $k \in K$.

When A is a $\Sigma(\vec{x})$ -atomic formula, $\mathcal{R} \vdash A$ denotes that A can be derived from the inferences rules in Fig. 1 for \mathcal{R} . This inference system refines and extends the one used in Fig. 7 of [3]. Specifically, we define refined notions of rewriting and membership directly in the proof theory, combine several rules into a single rule to allow simpler proofs later on, and allow rewrites to take place *modulo* the axioms E . Note that \rightarrow^1 corresponds to the notion of a one-step rewrite modulo the equations E , and $:^0$ is a more restricted notion of membership in which we disallow subject reduction.

A term $t \in T_\Sigma(\vec{x})$ is \mathcal{R} -reducible when there is a $u \in T_\Sigma(\vec{x})$ where $\mathcal{R} \vdash t \rightarrow^1 u$ and \mathcal{R} -irreducible when it is not \mathcal{R} -reducible. \mathcal{R} is *pattern-based* when for each rule in \mathcal{R} of the form $(\forall \vec{y}) A$ if $\bigwedge_{1 \leq i \leq m} u_i \rightarrow v_i \wedge \bigwedge_{1 \leq j \leq n} w_j : s_j$ with A of the form $l \rightarrow r$ or $l : s$, we have for each $v_i \in \{v_1, \dots, v_m\}$:

- The variables in v_i are fresh³, i.e.

$$\text{vars}(v_i) \cap \left(\text{vars}(l) \cup \bigcup_{1 \leq k \leq i} \text{vars}(u_k) \cup \bigcup_{1 \leq k < i} \text{vars}(v_k) \right) = \emptyset.$$

- Any rewriting of a term $v_i\theta$ occurs below the pattern v_i , i.e., for Σ -distinct variables \vec{x} and substitution $\theta : \vec{y} \rightarrow T_\Sigma(\vec{x})$, if there is a term $t \in T_\Sigma$ such that $\mathcal{R} \vdash v_i\theta \rightarrow t$, then there is a substitution $\tau : \vec{y} \rightarrow T_\Sigma(\vec{x})$ where $t =_E v_i\tau$.

\mathcal{R} is *confluent* relative to variables \vec{x} when for every $t, u, v \in T_\Sigma(\vec{x})$, if $\mathcal{R} \vdash t \rightarrow u$ and $\mathcal{R} \vdash t \rightarrow v$, then $\mathcal{R} \vdash u \downarrow v$. \mathcal{R} is *sort-decreasing* relative to variables \vec{x} when for every $t, u \in T_\Sigma(\vec{x})$ and $s \in S$ if $\mathcal{R} \vdash t \rightarrow u$ and $\mathcal{R} \vdash t : s$, then $\mathcal{R} \vdash u : s'$ for some sort $s' < s$. In particular, when \mathcal{R} is confluent (resp. sort-decreasing) relative to \emptyset , the empty set of variables, we call \mathcal{R} *ground confluent* (resp. *ground sort-decreasing*). \mathcal{R} is *weakly-convergent* relative to variables \vec{x} when \mathcal{R} is pattern-based, confluent, and sort-decreasing relative to \vec{x} . \mathcal{R} is *ground weakly-convergent* when \mathcal{R} is weakly-convergent relative \emptyset .

Each CERM system represents the operational description of an underlying MEL specification defined below.

Definition 2. Let $\mathcal{R} = (\Sigma, E, <, \Gamma)$ be a CERM system with $\Sigma = (K, F, S)$. $\mathcal{E}_\mathcal{R} = (\Sigma, E \cup \Gamma' \cup M_<)$ denotes the underlying MEL specification of \mathcal{R} where Γ' contains the same axioms as Γ , but with each oriented atomic formula $t \rightarrow u$ replaced with an equality $t = u$, and $M_<$ is a set of conditional memberships which axiomatize the subsort relation $<$, i.e.

$$M_< = \{(\forall x : k) x : s \text{ if } x : s' \mid s, s' \in S_k : s' < s\}.$$

There is an equivalence between formulas provable in a weakly-convergent system \mathcal{R} and its underlying specification $\mathcal{E}_\mathcal{R}$.

Theorem 1 (Inference Equivalence). Let $\mathcal{R} = (\Sigma, E, <, \Gamma)$ be a CERM system that is weakly-convergent relative to Σ -distinct variables \vec{x} . For all $t, u \in T_\Sigma(\vec{x})$ and $s \in S$,

$$\mathcal{E}_\mathcal{R} \vdash t = u \iff \mathcal{R} \vdash t \downarrow u \quad \text{and} \quad \mathcal{E}_\mathcal{R} \vdash t : s \iff \mathcal{R} \vdash t : s.$$

The inference system of $\mathcal{E}_\mathcal{R}$ can be found in Fig. 4 of [3]. □

A model theoretic definition of sufficient completeness for MEL specifications was given in [10]. We provide here proof theoretic definitions of sufficient completeness for both CERM systems and their underlying MEL specifications, and show under what conditions these two definitions coincide.

Definition 3. Let $\mathcal{R} = (\Sigma, E, <, \Gamma)$ be a CERM system in which $\Gamma = R \cup M$, where R contains the rewrite rules and M contains the membership rules.

Given a subset of memberships $M_\Omega \subseteq M$, let $\mathcal{R}_\Omega = (\Sigma, E, <, R \cup M_\Omega)$ be the associated CERM system. \mathcal{R} is sufficiently complete with respect to \mathcal{R}_Ω when for all $t, u \in T_\Sigma$ and $s \in S$:

$$\mathcal{R} \vdash t \downarrow u \iff \mathcal{R}_\Omega \vdash t \downarrow u \quad \text{and} \quad \mathcal{R} \vdash t : s \iff \mathcal{R}_\Omega \vdash t : s.$$

³ Without the additional requirement that no extra variables are introduced in r or any u_i , i.e., $\text{vars}(r) \subseteq l \cup \bigcup_{1 \leq k \leq m} v_k$ and each $\text{vars}(u_i) \subseteq l \cup \bigcup_{1 \leq k < i} v_k$, pattern-based systems are not directly executable, unless a strategy to instantiate the additional free variables in r and each u_i is given. Nevertheless, our results hold in generality without this extra requirement.

Additionally, $\mathcal{E}_{\mathcal{R}}$ is sufficiently complete with respect to $\mathcal{E}_{\mathcal{R}_\Omega}$ when for all $t, u \in T_\Sigma$ and $s \in S$:

$$\mathcal{E}_{\mathcal{R}} \vdash t = u \iff \mathcal{E}_{\mathcal{R}_\Omega} \vdash t = u \quad \text{and} \quad \mathcal{E}_{\mathcal{R}} \vdash t : s \iff \mathcal{E}_{\mathcal{R}_\Omega} \vdash t : s.$$

Moreover, if $\mathcal{E}_{\mathcal{R}}$ is sufficiently complete with respect to $\mathcal{E}_{\mathcal{R}_\Omega}$, M_Ω are its constructor memberships, $\mathcal{E}_{\mathcal{R}_\Omega}$ is a constructor subspecification of $\mathcal{E}_{\mathcal{R}}$, and $M_\Delta = M_\Sigma - M_\Omega$ are its defined memberships.

Theorem 2 (Suff. Comp. Equivalence). *Let $\mathcal{R} = (\Sigma, E, <, R \cup M)$, and let $M_\Omega \subseteq M$ be a subset of memberships in \mathcal{R} .*

- If \mathcal{R} is ground weakly-convergent and sufficiently complete with respect to \mathcal{R}_Ω , then $\mathcal{E}_{\mathcal{R}}$ is sufficiently complete with respect to $\mathcal{E}_{\mathcal{R}_\Omega}$.
- If \mathcal{R}_Ω is ground weakly-convergent, and $\mathcal{E}_{\mathcal{R}}$ is sufficiently complete with respect to $\mathcal{E}_{\mathcal{R}_\Omega}$, then \mathcal{R} is sufficiently complete with respect to \mathcal{R}_Ω . \square

4 Sufficient Completeness Checking

Techniques for checking sufficient completeness of unconditional specifications typically require that the specification is weakly normalizing, i.e., every term t rewrites to an irreducible term u . In the context of conditional rewriting, this condition is not strong enough. We need to develop a stronger notion of weak normalization in the context of CERM systems. Recall that a *reduction order* \succ on $T_\Sigma(\vec{x})$ is a strict order that is noetherian, congruent with $=_E$, and closed with respect to context and substitution.

Definition 4. *Let $\mathcal{R} = (\Sigma, E, <, \Gamma)$, and let \vec{x} be a set of Σ -distinct variables.*

A proof is said to be reductive relative to a reduction order \succ when every use of Replacement and Membership inferences:

$$\frac{t =_E C[l\theta] \quad \bigwedge u_i \theta \rightarrow v_i \theta \quad \bigwedge w_j \theta : s_1}{t \rightarrow^1 C[r\theta]} \quad \frac{t =_E l\theta \quad \bigwedge u_i \theta \rightarrow v_i \theta \quad \bigwedge w_j \theta : s_j \quad s' < s}{t :^0 s}$$

satisfies that $t \succ C[r\theta]$, $t \succ u_i \theta$ for each u_i , and $t \succ w_j \theta$ for each w_j .

We say that \mathcal{R} is weakly normalizing relative to \vec{x} when:

1. *For each $t \in T_\Sigma(\vec{x})$, there is a \mathcal{R} -irreducible $u \in T_\Sigma(\vec{x})$ such that $\mathcal{R} \vdash t \rightarrow u$*
2. *There is a reductive order $\succ_{\mathcal{R}}$ where for every $\Sigma(\vec{x})$ -sentence A derivable from \mathcal{R} , i.e. $\mathcal{R} \vdash A$, there is a proof of A that is reductive relative to $\succ_{\mathcal{R}}$.*

\mathcal{R} is ground weakly normalizing when it is weakly normalizing with $\vec{x} = \emptyset$.

It may not be immediately clear why these two separate conditions are needed, and whether one can be used to show the other. In fact, these conditions are independent: the system with constants a and b and a single sort s with rules $a \rightarrow b$, $b : s$, and $a : s$ if $b : s$, satisfies condition 1, but not condition 2; the system with a single constant a and the rule $a \rightarrow a$ satisfies condition 2, but not condition 1.

For weakly-normalizing and ground sort-decreasing specifications, there is a characterization of sufficient completeness which is often easier to check.

Theorem (Checking Theorem). *Let $\mathcal{R} = (\Sigma, E, <, R \cup M)$ be a ground weakly normalizing and ground sort-decreasing CERM system in which $\Sigma = (K, F, S)$. Given a set of memberships $M_\Omega \subseteq M$, let $\mathcal{R}_\Omega = (\Sigma, E, <, R \cup M_\Omega)$. \mathcal{R} is sufficiently complete with respect to \mathcal{R}_Ω iff for every membership $m \in M_\Delta$,*

$$(\forall \vec{y}) t : s \text{ if } \bigwedge_{i=1}^m u_i \rightarrow v_i \wedge \bigwedge_{j=1}^n w_j : s_j$$

in M_Δ , and each substitution $\theta : \vec{y} \rightarrow T_\Sigma$, if $t\theta$ is \mathcal{R}_Ω -irreducible, $\mathcal{R} \vdash u_i\theta \rightarrow v_i\theta$ and $\mathcal{R} \vdash w_j\theta : s_j$ for each i and j , then $\mathcal{R}_\Omega \vdash t\theta :^0 s$. \square

Proof. (Sketch) It is quite straightforward to show that if \mathcal{R} is sufficiently complete, the condition to check holds. Showing that the condition we would like to check implies sufficient completeness is more complex. We do this by showing that $\mathcal{R} \vdash t \rightarrow u$ implies $\mathcal{R}_\Omega \vdash t \rightarrow u$, and $\mathcal{R} \vdash t : s$ implies $\mathcal{R}_\Omega \vdash t : s$ by inducting on \mathcal{R} proofs using a well-founded order formed from the reduction order \succ guaranteed by our assumption that \mathcal{R} is weakly-normalizing. For the full details of this proof, see [12].

This theorem shows us that we can check the sufficient completeness of CERM systems by only considering \mathcal{R}_Ω -irreducible instances of defined memberships. This criteria provides the formal justification for why the tool presented in [10] is sound for a more general class of specifications that may be weakly-normalizing rather than reductive, and may use oriented formulas $t \rightarrow u$ in conditions in addition to join conditions. This checking theorem also provides much of the formal justification for our equational tree automata based techniques presented below.

5 Sufficient Completeness as a PTA Decision Problem

In this section, we show how sufficient completeness for a common subclass of CERM systems can be cast as decision problem for Propositional Tree Automata (PTA), a tree automata framework which recognizes languages closed modulo an equational theory and Boolean operations. Propositional Tree Automata differ from standard tree automata in that the signature over which elements in the language are constructed is an equational theory rather than a ranked alphabet, and the acceptance condition depends on the complete set of states a term can reach rather than just individual states. Our definition below slightly generalizes that of [13] by using many-kinded signatures rather than untyped signatures, however the results on emptiness checking can be easily generalized to the many-kinded setting.

Definition 5. *A propositional tree automaton (PTA) \mathcal{A} is a tuple $(\mathcal{E}, Q, \Phi, \Delta)$ where:*

- $\mathcal{E} = (\Sigma, E)$ is a many-kinded equational theory with $\Sigma = (K, F)$;

- $Q = \{Q_k\}_{k \in K}$ is a K -indexed family of sets of states disjoint from the functions symbols in F ;
- $\Phi = \{\phi_k\}_{k \in K}$ is a K -index family of propositional formulas, where the atomic propositions in each ϕ_k are states in Q_k ; and
- Δ contains rewrite rules, called transition rules, each of which is of one of the following forms:

$$\text{(Type 1)} \quad f(p_1, \dots, p_n) \rightarrow q \qquad \text{(Type 2)} \quad p \rightarrow q$$

where for some $k \in K$, $p, q \in Q_k$, $f \in F_{(k_1, \dots, k_n), k}$, and each $p_i \in Q_{k_i}$.

We let $T_\Sigma(Q)$ denote the terms formed from the signature Σ with the states in each set Q_k added as a constant of kind k . The move relation $\rightarrow_{\mathcal{A}}$ is a rewrite relation over the terms in $T_\Sigma(Q)$ formed by rewriting using the rules in Δ modulo the equations in \mathcal{E} , i.e. $s \rightarrow_{\mathcal{A}} t$ if there is a transition rule $l \rightarrow q \in \Delta$ and a context C such that $s =_{\mathcal{E}} C[l]$ and $t =_{\mathcal{E}} C[q]$. The reflexive-transitive closure of $\rightarrow_{\mathcal{A}}$ is denoted by $\rightarrow_{\mathcal{A}}^*$.

A term $t \in T_{\Sigma, k}$ is accepted by \mathcal{A} if the complete set of states reachable from t , $\text{reach}_{\mathcal{A}}(t) = \{q \in Q \mid t \rightarrow_{\mathcal{A}}^* q\}$, is a model of ϕ_k . Boolean formulas are evaluated using their standard interpretations:

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

The class of presentations checkable using Propositional Tree Automata are those that are order-sorted, left-linear, weakly-normalizing, and ground weakly-convergent with rewriting modulo axioms. We call this class of specifications *PTA checkable*, and define it precisely below:

Definition 6. A CERM system $\mathcal{R} = (\Sigma, E, <, \Gamma)$ is PTA checkable when:

- \mathcal{R} is ground weakly normalizing and ground weakly-convergent;
- Each equation in E is linear;
- Every axiom in Γ is of one of the forms:

$$(\forall \vec{y}) f(t_1, \dots, t_n) \rightarrow r \text{ if } \bigwedge_{y_i \in \vec{y}} y_i : s_i \quad (\forall \vec{y}) f(x_1, \dots, x_n) : s \text{ if } \bigwedge_{y_i \in \vec{y}} y_i : s_i$$

where each variable in \vec{y} appears exactly once in the left-hand-side of the axiom.

As syntactic sugar, we will use $f(x_1 : s_1, \dots, x_n : s_n) : s$ to denote a membership $(\forall \vec{x}) f(x_1, \dots, x_n) \text{ if } s \bigwedge_{x_i \in \vec{x}} x_i : s_i$, and will use with the syntax $(\forall \vec{x} : \vec{s}) l \rightarrow r$ to denote a rewrite rule $(\forall \vec{x}) l \rightarrow r \text{ if } \bigwedge_{x_i \in \vec{x}} x_i : s_i$. Since every PTA-checkable system satisfies the conditions in Theorem 4, we can use the characterization given in that theorem to check sufficient completeness.

Although the class of PTA specifications is smaller than the class of general CERM systems, it nevertheless contains many interesting specifications that have not been checkable before. As a simple example of such a CERM system, we give a specification of non-empty multisets of natural numbers in the following Maude functional module:

```

fmod MSET is
  sorts Nat MSet .                subsort Nat < MSet .
  op 0 : -> Nat [ctor].          op s : Nat -> Nat [ctor].
  op -- : MSet MSet -> MSet [ctor assoc comm].
  op |_| : MSet -> Nat .
  var N : Nat                    var M : MSet .
  eq | N | = s(0) .              eq | N M | = s(| M |) .
endfm

```

This module defines the sorts `Nat` and `MSet` with `Nat < MSet`. Because of this subsort declaration, `Nat` and `MSet` are implicitly in the same kind, denoted `[MSet]`. The operators `0` and `s` denote zero and successor respectively. The subsort declaration specifies that every natural number is also a multiset, and the `--` operator denotes the union of two multisets. These operators are labeled with the `ctor` attribute to identify them as constructors, and `--` is additionally labeled with the `assoc` and `comm` attributes to introduce associativity and commutativity axioms for it, i.e., $(xy)z = x(yz)$ and $xy = yx$. A single defined operation is introduced `|_|` which returns the number of elements in a multiset. The variable declarations associate the variable `N` with the the sort `Nat`, and the variable `M` with the sort `MSet`. By this, `M` and `N` are declared to have kind `[MSet]`, and when either variable appears in a rule, an implicit condition is added to the rule that the variable only ranges over terms with the associated sort. The other equations, `| N | = s(0)` and `| N M | = s(| M |)`, are interpreted as rewrite rules. In the CERM system corresponding to `MSET`, E only contains the associativity and commutativity axioms of `--`. Γ contains the memberships and other equations.

The specification contains membership rules as well, but these are implicit in the operator declaration section. In fact, the previous operation and equation declarations are just syntactic sugar for:

```

op 0 : -> [MSet] .                op s : [MSet] -> [MSet] .
op -- : [MSet] [MSet] -> [MSet] [assoc comm].
op |_| : [MSet] -> [MSet] .
mb 0 : Nat .                      cmb s(X) : Nat if X : Nat .
cmb X Y : MSet if X : MSet /\ Y : MSet .
cmb | X | : Nat if X : MSet .
ceq | N | = s(0) if N : Nat .
ceq | N M | = s(| M |) if N : Nat /\ M : MSet .

```

We can reduce the sufficient completeness of this specification and other PTA checkable specifications to the emptiness problem for a PTA.

Theorem 3. *Given a PTA-checkable CERM system $\mathcal{R} = (\Sigma, E, <, R \cup M)$ with $\Sigma = (K, F, S)$, and a set of constructor memberships $M_\Omega \subseteq M$, let $\mathcal{R}_\Omega = (\Sigma, E, <, R \cup M_\Omega)$.*

One can effectively construct a PTA $\mathcal{A}_\mathcal{R}$ such that \mathcal{R} is sufficiently complete relative to constructor memberships M_Ω iff $\mathcal{L}(\mathcal{A}_\mathcal{R}) = \emptyset$.

Proof. (Sketch) We sketch the construction of $\mathcal{A}_{\mathcal{R}}$ here. See [12] for full details.

The PTA $\mathcal{A}_{\mathcal{R}}$ checks sufficient completeness using the characterization given in Theorem 4. The basic idea is that specific states in the automaton recognize terms that satisfy properties mentioned in Theorem 4: (1) for each kind $k \in K$ and sort $s \in S_k$, $\mathcal{A}_{\mathcal{R}}$ contains a state q_s such that $t \rightarrow_{\mathcal{A}_{\mathcal{R}}}^* q_s$ iff $\mathcal{R}_{\Omega} \vdash t : s$ without using Subject Reduction in the proof; (2) for each kind $k \in K$ and sort $s \in S_k$, $\mathcal{A}_{\mathcal{R}}$ contains a state d_s such that $t \rightarrow_{\mathcal{A}_{\mathcal{R}}}^* d_s$ iff there is a membership $f(x_1 : s_1, \dots, x_n : s_n) : s \in M - M_{\Omega}$ such that $t =_E f(t_1, \dots, t_n)$ and $M_{\Omega} \vdash t_i : s_i$ without using Subject Reduction for each $i \in \{1, \dots, n\}$; and (3) for each kind $k \in K$, $\mathcal{A}_{\mathcal{R}}$ contains a state r_k such that for each term $t \in T_{\Sigma, k}$, $t \rightarrow_{\mathcal{A}_{\mathcal{R}}} r_k$ iff t is \mathcal{R} -reducible. Using these states it is easy to show that if $t \in T_{\Sigma, k}$ is a sufficient completeness counterexample, then $r_k \notin \text{reach}_{\mathcal{A}_{\mathcal{R}}}(t)$ and there exists a sort $s \in S_k$ such that $d_s \in \text{reach}_{\mathcal{A}_{\mathcal{R}}}(t)$ and $q_s \notin \text{reach}_{\mathcal{A}_{\mathcal{R}}}(t)$.

The rules for the states q_s and d_s can be immediately derived from the memberships in M . The rules for r_k require auxiliary states. For each kind $k \in K$, we add a q_k that recognizes every term with kind k . We also need a state for each non-variable subterm appearing in the left-hand side of a rewrite rule. We let $\mathbf{I}_{\mathcal{R}}$ denote the set of quantified non-variable strict subterms appearing in the left-hand side of a rule in R , i.e.,

$$\mathbf{I}_{\mathcal{R}} = \{ t[\vec{x} : \vec{s}] \mid (\forall \vec{x} : \vec{s}) (C[t] \rightarrow u \mid C[t] : s) \in \Gamma \wedge t \notin \vec{x} \wedge C \neq \square \}.$$

For each quantified term $u[\vec{x} : \vec{s}] \in \mathbf{I}_{\mathcal{R}}$, $\mathcal{A}_{\mathcal{R}}$ contains a state $q_{u[\vec{x} : \vec{s}]}$, and rules such that $t \rightarrow_{\mathcal{A}_{\mathcal{R}}} q_{u[\vec{x} : \vec{s}]}$ iff $t =_E u\theta$ with $\mathcal{R}_{\Omega} \vdash \theta(x_i) : s_i$ for each $i \in \{1, \dots, n\}$ without using Subject Reduction.

To simplify the definition of $\mathcal{A}_{\mathcal{R}}$, we let $q_{x_i[\vec{x} : \vec{s}]} = q_{s_i}$. Formally, we define $\mathcal{A}_{\mathcal{R}} = (\mathcal{E}, \{Q_k\}_{k \in K}, \{\phi_k\}_{k \in K}, \Delta)$ where $\mathcal{E} = ((K, F), E)$ and

$$Q_k = \{q_s, d_s \mid s \in S_k\} \cup \{q_k, r_k\} \cup \{q_{t[\vec{x} : \vec{s}]} \mid t[\vec{x} : \vec{s}] \in \mathbf{I}_{\mathcal{R}}\},$$

$$\phi_k = \neg r_k \wedge \bigvee_{s \in S_k} (d_s \wedge \neg q_s),$$

$$\Delta = \{f(q_{s_1}, \dots, q_{s_n}) \rightarrow q_s \mid f(x_1 : s_1, \dots, x_n : s_n) : s \in M_{\Omega}\} \quad (1)$$

$$\cup \{q_s \rightarrow q_{s'} \mid s < s'\} \quad (2)$$

$$\cup \{f(x_i : s_i, \dots, x_i : s_i) \rightarrow d_s \mid f(x_1 : s_1, \dots, x_n : s_n) : s \in M_{\Delta}\} \quad (3)$$

$$\cup \{f(q_{t_1[\vec{x} : \vec{s}]}, \dots, q_{t_n[\vec{x} : \vec{s}]}) \rightarrow q_{f(t_1, \dots, t_n)[\vec{x} : \vec{s}]} \mid f(t_1, \dots, t_n)[\vec{x} : \vec{s}] \in \mathbf{I}_{\mathcal{R}}\} \quad (4)$$

$$\cup \{f(q_{t_1[\vec{x} : \vec{s}]}, \dots, q_{t_n[\vec{x} : \vec{s}]}) \rightarrow r_k \mid (\forall \vec{x} : \vec{s}) f(t_1, \dots, t_n) \rightarrow u \in R\} \quad (5)$$

$$\cup \{f(q_{k_1}, \dots, q_{k_n}) \rightarrow q_k \mid f \in \Sigma_{k_1 \dots k_n, k}\} \quad (6)$$

$$\cup \{f(q_{k_1}, \dots, q_{k_{i-1}}, r_{k_i}, q_{k_{i+1}}, \dots, q_{k_n}) \rightarrow r_k \mid f \in F_{k_1, \dots, k_n, k} \wedge 1 \leq i \leq n\}. \quad (7)$$

The rules in (1) and (2) are used to recognize constructor terms. The rules in (3) are used to recognize the defined terms. The rules in (4) and (5) are used so that each term in $T_{\Sigma, k}$ matching a rule in R reaches r_k . Finally, the rules in (6) and (7) are used to close the set of terms reaching r_k under context. The acceptance condition guarantees that $\mathcal{A}_{\mathcal{R}}$ only accepts irreducible terms that match a defined symbol with sort s that are not a constructor term with sort s . It follows that $t \in \mathcal{L}(\mathcal{A}_{\mathcal{R}})$ iff t is a sufficient completeness counterexample. \square

As an example, the sufficient completeness automaton for the MSET specification is the automaton $\mathcal{A}_{\text{MSET}} = (\mathcal{E}, Q, \Phi, \Delta)$ with $\mathcal{E} = ((K, F), E)$ and

- K is the single kind $[\text{MSet}]$.
- F contains 0 , \mathfrak{s} , $_ _$, and $| _ |$.
- E contains the associativity and commutativity axioms for $_ _$.
- $Q_{[\text{MSet}]} = \{\mathfrak{q}_{[\text{MSet}]}, \mathfrak{r}_{[\text{MSet}]}, \mathfrak{q}_{\text{Nat}}, \mathfrak{q}_{\text{MSet}}, \mathfrak{d}_{\text{Nat}}, \mathfrak{d}_{\text{MSet}}, \mathfrak{q}_{\text{NM}}\}$.
- $\Phi_{[\text{MSet}]} = \neg \mathfrak{r}_{[\text{MSet}]} \wedge ((\mathfrak{d}_{\text{Nat}} \wedge \neg \mathfrak{q}_{\text{Nat}}) \vee (\mathfrak{d}_{\text{MSet}} \wedge \neg \mathfrak{q}_{\text{MSet}}))$.
- Δ contains the following rules:

$$0 \rightarrow \mathfrak{q}_{[\text{MSet}]} \qquad \mathfrak{s}(\mathfrak{q}_{[\text{MSet}]}) \rightarrow \mathfrak{q}_{[\text{MSet}]} \qquad (8)$$

$$\mathfrak{q}_{[\text{MSet}]} \mathfrak{q}_{[\text{MSet}]} \rightarrow \mathfrak{q}_{[\text{MSet}]} \qquad | \mathfrak{q}_{[\text{MSet}]} | \rightarrow \mathfrak{q}_{[\text{MSet}]} \qquad (9)$$

$$0 \rightarrow \mathfrak{q}_{\text{Nat}} \qquad \mathfrak{s}(\mathfrak{q}_{\text{Nat}}) \rightarrow \mathfrak{q}_{\text{Nat}} \qquad (10)$$

$$\mathfrak{q}_{\text{MSet}} \mathfrak{q}_{\text{MSet}} \rightarrow \mathfrak{q}_{\text{MSet}} \qquad \mathfrak{q}_{\text{Nat}} \rightarrow \mathfrak{q}_{\text{MSet}} \qquad (11)$$

$$| \mathfrak{q}_{\text{MSet}} | \rightarrow \mathfrak{d}_{\text{Nat}} \qquad \mathfrak{q}_{\text{Nat}} \mathfrak{q}_{\text{MSet}} \rightarrow \mathfrak{q}_{\text{NM}} \qquad (12)$$

$$| \mathfrak{q}_{\text{Nat}} | \rightarrow \mathfrak{r}_{[\text{MSet}]} \qquad | \mathfrak{q}_{\text{NM}} | \rightarrow \mathfrak{r}_{[\text{MSet}]} \qquad (13)$$

$$\mathfrak{s}(\mathfrak{r}_{[\text{MSet}]}) \rightarrow \mathfrak{r}_{[\text{MSet}]} \qquad \mathfrak{q}_{[\text{MSet}]} \mathfrak{r}_{[\text{MSet}]} \rightarrow \mathfrak{r}_{[\text{MSet}]} \qquad (14)$$

$$\mathfrak{r}_{[\text{MSet}]} \mathfrak{q}_{[\text{MSet}]} \rightarrow \mathfrak{r}_{[\text{MSet}]} \qquad | \mathfrak{r}_{[\text{MSet}]} | \rightarrow \mathfrak{r}_{[\text{MSet}]} \qquad (15)$$

Due to (8) and (9), for every term t , we have $t \rightarrow \mathfrak{q}_{[\text{MSet}]}$. Due to (10) for every natural number n , we have $t \rightarrow \mathfrak{q}_{\text{Nat}}$. Due to (11) for every multiset m , we have $m \rightarrow \mathfrak{q}_{\text{MSet}}$. Due to the first rule on (12). $|m| \rightarrow \mathfrak{d}_{\text{Nat}}$ for each multiset m , and due to the second on (12), we have $nm \rightarrow \mathfrak{q}_{\text{NM}}$ for each natural number n and multiset m . Due to (13). terms matching equations are rewritten to $\mathfrak{r}_{[\text{MSet}]}$, and finally due to (14) and (15), the set of terms rewritable to $\mathfrak{r}_{[\text{MSet}]}$ is closed under context.

5.1 PTA Emptiness Testing

For an arbitrary theory \mathcal{E} , the emptiness problem for a PTA \mathcal{A} over that theory is undecidable. However, in practice we do not need to consider arbitrary equational theories, but rather we can focus on specific axioms like associativity, commutativity, and identity with matching algorithms that are supported by rewriting languages like Maude. A decision procedure for PTA with AC symbols was first described in [12]. This decision procedure was later extended to support associative symbols in [13]. In this case, decidability is lost. Nevertheless by using techniques from machine learning, we are always able to find counterexamples if they exist, and show sufficient completeness when the associative symbols are used to define regular structures such as lists, and queues.

The semi-algorithm essentially is a generalization of the subset construction algorithm used on standard tree automata [5], but extended to support reasoning modulo the axioms. The idea is that given the PTA $\mathcal{A} = (\mathcal{E}, Q, \Phi, \Delta)$ we incrementally compute the set of reachable sets of states $R_{\mathcal{A}} = \{R_{\mathcal{A},k}\}_{k \in K}$ such that $R_{\mathcal{A},k} = \{\text{reach}_{\mathcal{A}}(t) \mid t \in T_{\Sigma,k}\}$. If $\mathcal{L}(\mathcal{A}) \neq \emptyset$, then there will be a set of states $P \in R_{\mathcal{A},k}$ such that $P \models \phi_k$.

6 Conclusions

We have presented two contributions advancing methods for proving sufficient completeness to handle conditional specifications involving partial functions and where deduction is performed modulo axioms. Specifically, we have studied the sufficient completeness of such specifications and their associated rewriting systems in greater generality than it had been done up to now, and have identified a subclass of specifications whose sufficient completeness problem is equivalent to the emptiness problem for their associated PTA. This characterization allows us to us to build a checker when the rewriting is modulo any combination of associativity, commutativity, and identity.

A number of further extensions of this work seem worth investigating. A first extension is to continue to push our characterization to consider checking specifications with context-sensitive rewriting and parameterised specifications. A second important topic is the generation of counterexamples to show lack of sufficient completeness: ground term counterexamples are practical and easy to generate, but investigating ways of symbolically describing sets of counterexamples may be quite useful for other purposes, such as generating induction schemes for theorem provers. A third topic worth investigating is what we called the “second prong” in the introduction, namely, integrating sufficient completeness checking and inductive theorem proving in order to handle specifications outside the decidable subclasses. Further advances in these three areas should provide both foundations and algorithms in which to build a next-generation PTA-based sufficient completeness tool for MEL specifications modulo axioms. This would make sufficient completeness checking available for a very wide class of specifications in Maude and other equational languages for specification and programming with advanced features.

References

1. A. Bouhoula and M. Rusinowitch: *SPIKE: A System for Automatic Inductive Proofs*, Proc. of 4th AMAST, Montreal (Canada), LNCS 936, pp. 576–577, 1995.
2. A. Bouhoula and F. Jacquemard: *Automating Sufficient Completeness Check for Conditional and Constrained TRS*, Proc. of UNIF, 2006, Seattle, WA, Available at URL: <http://www.easychair.org/FLoC-06/UNIF.html>
3. A. Bouhoula, J.P. Jouannaud and J. Meseguer: *Specification and Proof in Membership Equational Logic*, TCS 236, pp. 35–132, 2000.
4. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and J.F. Quezada: *Maude Specification and Programming in Rewriting Logic*, TCS 285, pp. 187–243, 2002.
5. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi: *Tree Automata Techniques and Applications (TATA)*, draft, 2002. Available at URL: <http://www.grappa.univ-lille3.fr/tata>
6. F. Gécseg and M. Steinby: *Tree Languages*, Handbook of Formal Languages 3, pp. 1–68 (Chap. 1), Springer-Verlag, 1996.
7. S. Ginsburg: *The Mathematical Theory of Context-Free Languages*, McGraw-Hill, 1966.

8. J.V. Guttag: *The Specification and Application to Programming of Abstract Data Types*, Ph.D. thesis, Computer Science Department, University of Toronto, 1975.
9. J.V. Guttag and J.J. Horning: *The Algebraic Specification of Abstract Data Types*, Acta Inf. 10, pp. 27–52, 1978.
10. J. Hendrix, M. Clavel and J. Meseguer: *A Sufficient Completeness Reasoning Tool for Partial Specifications*, Proc. of 16th RTA, Nara (Japan), LNCS 3467, pp. 165–174, 2005.
11. J. Hendrix, J. Meseguer and H. Ohsaki: *A Sufficient Completeness Checker for Linear Order-Sorted Specifications Modulo Axioms*, Proc. of IJCAR 2006, Seattle, Washington, LNCS 4130, pp. 151–155, 2006.
12. J. Hendrix, H. Ohsaki and J. Meseguer: *Sufficient Completeness Checking with Propositional Tree Automata*, technical report UIUCDCS-R-2005-2635, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005. Available at <http://texas.cs.uiuc.edu/>
13. J. Hendrix, H. Ohsaki and M. Viswanathan: *Propositional Tree Automata*, Proc. of 17th RTA, Seattle, Washington, LNCS 4098, pp. 50–65, 2006.
14. J.P. Jouannaud and E. Kounalis: *Automatic Proofs by Induction in Equational Theories without Constructors*, Inf. Comput. 81(1), pp. 1–33, 1989.
15. D. Kapur: *An Automated Tool for Analyzing Completeness of Equational Specifications*, Proc. of ISSTA'94, Seattle (USA), pp. 28–43, ACM Press, 1994.
16. D. Kapur, P. Narendran and H. Zhang: *On Sufficient-Completeness and Related Properties of Term Rewriting Systems*, Acta Inf. 24, pp. 395–415, 1987
17. D. Kapur, P. Narendran, D.J. Rozenkrantz and H. Zhang: *Sufficient-Completeness, Ground-Reducibility and Their Complexity*, Acta Inf. 28, pp. 311–350, 1991.
18. D. Lugiez and S. Dal Zilio: *XML Schema, Tree Logic and Sheaves Automata*, Proc. of 14th RTA, Valencia (Spain), LNCS 2706, pp. 246–263, 2003.
19. D. Lugiez, S. Dal Zilio and C. Meyssonier: *A Logic You can Count on*, Proc. of 31st POPL, Venice (Italy), pp. 135–146, ACM Press, 2004.
20. J. Meseguer: *Membership Algebra as a Logical Framework for Equational Specification*, Proc. of WADT'97, Tarquinia (Italy), LNCS 1376, pp. 18–61, 1997.
21. T. Nipkow and G. Weikum: *A Decidability Result about Sufficient Completeness of Axiomatically Specified Abstract Data Types*, Proc. of 6th GI Conference, Dortmund (Germany), LNCS 145, pp. 257–268, 1983,
22. H. Ohsaki: *Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories*, Proc. of 15th CSL, Paris (France), LNCS 2142, pp. 539–553, 2001.
23. H. Ohsaki and T. Takai: *Decidability and Closure Properties of Equational Tree Languages*, Proc. of 13th RTA, Copenhagen (Denmark), LNCS 2378, pp. 114–128, 2002.
24. R.J. Parikh: *On Context-Free Languages*, JACM 13(4), pp. 570–581, 1966.
25. H. Seidl, T. Schwentick and A. Muscholl: *Numerical Document Queries*, Proc. of 22nd PODS, San Diego (USA), pp. 155–166, ACM Press, 2003.
26. K.N. Verma: *Two-Way Equational Tree Automata for AC-Like Theories: Decidability and Closure Properties*, Proc. of 14th RTA, Valencia (Spain), LNCS 2706, pp. 165–179, 2003.