A Maude Coherence Checker Tool for Conditional Order-Sorted Rewrite Theories

Francisco Durán[1] and José Meseguer[2]

[1] Universidad de Málaga, Spain
[2] University of Illinois at Urbana-Champaign, IL, USA

Abstract. For a rewrite theory to be executable, its equations E should be (ground) confluent and terminating modulo the given axioms A, and their rules should be (ground) coherent with E modulo A. The correctness of many important formal verification tasks, including search, LTL model checking, and the development of abstractions, crucially depends on the theory being ground coherent. Furthermore, many specifications of interest are typed, have equations E and rules R that are both conditional, have axioms A involving various combinations of associativity, commutativity and identity, and may contain frozenness restrictions. This makes it essential to extend the known coherence checking methods from the untyped, unconditional, and AC or free case, to this much more general setting. We present the mathematical foundations of the Maude ChC 3 tool, which provide such a generalization to support coherence and ground coherence checking for order-sorted rewrite theories under these general assumptions. We also explain and illustrate the use of the ChC 3 tool with a nontrivial example.

1 Introduction

Traditionally, a rewrite system is a set of directed equations used to compute a value by repeatedly replacing subterms of a given formula with equal terms until a (typically unique) simplest possible form is obtained. This interpretation of a rewrite system gives an equational semantics to it, and a way of executing functional programs by rewriting. But rewriting is also useful for specifying non-equational relations, such as transitions between states. Rewriting logic [21] suggests keeping all rules with an equational interpretation as a distinguished set E of equations, and considering the remaining rules R as defining state transition steps over equivalence classes modulo E.

A rewriting logic signature is an equational specification. But, rewriting logic is parameterized by the choice of its underlying equational logic. For example, for Maude [3], the underlying equational logic is membership equational logic, so that signatures are of the form (Ω, E), where $\Omega = (K, \Sigma, S)$ is a membership equational logic signature and E is a set of (conditional) membership axioms and equations. Such a signature (Ω, E) makes explicit the set of equations in order to emphasize that rewriting will operate on congruence classes of terms *modulo* E.

P.C. Ölveczky (Ed.): WRLA 2010, LNCS 6381, pp. 86–103, 2010.

Thus, a rewrite theory has both rules and equations, so that rewriting is performed modulo such equations. However, this does not mean that an implementation of rewriting logic must have an E-matching algorithm for each equational theory E that a user might specify, which is impossible, since matching modulo an arbitrary theory is undecidable. What, e.g., Maude instead requires for rewrite theories in system modules is that:

- The equations are divided into a set A of structural axioms, for which matching algorithms exist and a set E of equations that are (ground) Church-Rosser and terminating modulo A. For some equations E, termination modulo A can be checked using the Maude Termination Tool (MTT) [9,5] and the Church-Rosser property can be checked using a Church-Rosser checker as the one presented in [15,14].
- The rules R in the module are (ground) *coherent* [22,25] with the equations E modulo A. This means that appropriate critical pairs can be filled in between rules and equations, allowing us to intermix rewriting with rules and rewriting with equations without losing completeness of rule computations by failing to perform a rewrite that would have been possible before an equational deduction step was taken. In this way, we get the effect of rewriting modulo $E \cup A$ with just a matching algorithm for A. In particular, a simple strategy available in these circumstances is to always reduce to canonical form using E before applying any rule in R. This is precisely the strategy adopted by the Maude interpreter.

Therefore, it is very important to know whether a given Church-Rosser and terminating specification is indeed ground-coherent. For this purpose, the coherence checking methods proposed by Viry [25] must be substantially generalized because: (i) they are restricted to the AC or free cases; (ii) assume that both the equations and the rules are unconditional; (iii) always require the very restrictive condition that the right-hand and left-hand sides of any equation are both linear; and (iv) are untyped. Instead, what we need to handle more expressive specifications are *generalized rewrite theories* $\mathcal{R} = (\Sigma, E \cup A, R, \phi)$ [2] such that: (i) have an initial model semantics; (ii) the equations E and the rules R can both be *conditional*; (iii) Σ is typed (here we assume Σ order-sorted); (iv) the set A of axioms may involve associativity and/or commutativity and/or identity axioms; and (v) rewriting with rules is restricted by frozenness information ϕ.

At first sight, checking coherence under these more general conditions may appear to be an even more challenging task than in the simpler situations contemplated by Viry in [25]. However, as we show in this paper, some of these more general conditions can make it *much easier* to check coherence. In particular:

(1) frozenness can eliminate many critical pairs and greatly reduce the linearity requirements on variables of equations;
(2) order-sorted type structure can: (i) eliminate many critical pairs, (ii) further relax linearity conditions on variables of equations, and (iii) eliminate many problematic non-overlap situations between equations and rules;

(3) the initial model semantics substantially relaxes the coherence requirement into a *ground coherence* one where: (i) unjoinable critical pairs can be shown ground joinable if some equational inductive proof obligations can be discharged; and (ii) by checking sufficient completeness of the equations with respect to a constructor subsignature, defined function symbols can safely be assumed to be *frozen*, which by (1) can further reduce the number of critical pairs that need to be considered and the linearity requirements on equations.

A further point to emphasize is that the present ChC tool can in principle deal with *any combination* of associativity, commutativity, and identity axioms, including the thorny cases of associativity without commutativity for which no finitary unification algorithms exist. Although in general computing critical pairs for the associativity without commutativity cases may not be possible, in many practical cases our tool can show that the relevant left-hand sides have a finite set of *variants* [6,18] when associativity is used as a rule. This then allows the application of a theory transformation described in [10] thanks to which associativity without commutativity axioms need not be used when computing critical pairs.

Our coherence checker tool (ChC) [13] is particularly well-suited for checking Maude specifications with an initial model semantics whose equations E have already been proved Church-Rosser and terminating modulo A, and now we need to check that its rules R are ground-coherent with E modulo A. Our methods can also be used to check the coherence property of conditional order-sorted specifications that do not have an initial model semantics, such as, for example, those specified in Maude system theories [4]. Since, for the reasons mentioned above, user interaction will typically be quite essential, coherence completion is not attempted. Instead, if the specification cannot be shown to be coherent or ground-coherent by the tool, proof obligations are generated and are given back to the user as a guide in the attempt to establish the ground-coherence property. Since this property is in fact inductive, in some cases the Maude inductive theorem prover can be enlisted to prove some of these proof obligations. In other cases, the user may in fact have to modify the original specification by carefully considering the information conveyed by the proof obligations. We give in Section 3 some methodological guidelines for the use of the tool, and illustrate the use of the tool with some examples.

The present ChC tool only accepts order-sorted conditional specifications, where each of the operation symbols has either no equational attributes, or any combination of associativity/commutativity/identity.[1] Furthermore, it is assumed that such specifications do not contain any built-in function, do not use the `owise` attribute, and that they have already been proved Church-Rosser and terminating. The tool attempts to establish the ground-coherence property

[1] The associativity without commutativity case is handled using a semi-algorithm proposed in [10], which works in many practical situations but not always. We refer the reader to [10] for further details.

modulo the equational axioms specified for each of the operators by checking a sufficient condition. Therefore, the tool's output consists of a set of critical pairs that the tool has not been able to join and must be shown ground-joinable.

As other tools in the Maude formal environment [5], the ChC tool has been implemented as an extension of Full Maude [12,7]. Details on how to extend Full Maude in different forms can be found in, e.g., [17,12,7,8]. Following these techniques, the ChC has been integrated within the Full Maude environment, to allow checking of modules defined in Full Maude and to get a much more convenient user interface. Of course, it would have been possible to define an interface for the tool without integrating it with Full Maude. Since all the infrastructure built for Full Maude can be used by itself, just by selecting functions from that infrastructure in the needed modules, any of the two possibilities can give rise to an interface in a very short time. However, by integrating the specifications of Full Maude and of the ChC we not only have such a needed infrastructure, but in addition we can, for example, check the coherence property of any module in Full Maude's database. We can therefore use the tool on any module accepted by Full Maude, including structured modules, parameterized modules, etc. We still have, of course, the restrictions mentioned above.

The rest of the paper is structured as follows. Section 2 introduces the notion of coherent order-sorted specification modulo axioms. Section 3 presents some directions on how to use the tool and illustrates it with an example. Section 4 concludes and presents some future work. Proofs of technical results are not included here for space reasons. They can be found in [16]. We assume that the reader is familiar with basic rewriting terminology and notations. Although we have tried to make the paper self contained, we refer the interested reader to [23] for additional details on rewriting techniques.

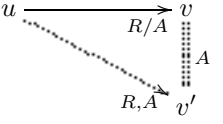
2 Coherent Order-Sorted Specifications Modulo Axioms

This section presents the theoretical foundations of the ChC.

2.1 Conditional Rewriting Modulo Linear and Regular Axioms A

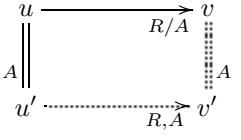
Given an order-sorted rewrite theory $\mathcal{R} = (\Sigma, A, R)$, where A is a collection of unconditional equational axioms of the form $u = v$ that are *linear* (no repeated variables in either u or v), and *regular* ($vars(u) = vars(v)$), we define the relation $\rightarrow_{R/A}$, either by the inference system of rewriting logic (see [2]), or by the usual inductive description: $\rightarrow_{R/A} = \bigcup_n \rightarrow_{R/A,n}$, where $\rightarrow_{R/A,0} = \emptyset$, and for each $n \in \mathbb{N}$, we have $\rightarrow_{R/A,n+1} = \rightarrow_{R/A,n} \cup \{(u, v) \mid u =_A l\sigma \rightarrow r\sigma =_A v \wedge l \rightarrow r \text{ if } \bigwedge_i u_i \rightarrow v_i \in R \wedge \forall i, u_i\sigma \rightarrow^*_{R/A,n} v_i\sigma\}$. In general, of course, given terms t and t' with sorts in the same connected component, the problem of whether $t \rightarrow_{R/A} t'$ holds is undecidable.

Even if there is an effective A-matching algorithm, the relation $u \to_{R/A} v$ still remains undecidable in general, since to see if $u \to_{R/A} v$ involves searching through the possibly infinite equivalence classes $[u]_A$ and $[v]_A$ to see whether an A-match is found for a subterm of some $u' \in [u]_A$ and the result of rewriting u' belongs to the equivalence class $[v]_A$. For this reason, a much simpler relation $\to_{R,A}$ is defined, which becomes decidable if an A-matching algorithm exists. We define (see [24]) $\to_{R,A} = \bigcup_n \to_{R,A,n}$ where $\to_{R,A,0} = \emptyset$, and for each $n \in \mathbb{N}$ and any terms u, v with sorts in the same connected component the relation $u \to_{R,A,n+1} v$ holds if either $u \to_{R,A,n} v$, or there is a position p in u, a rule $l \to r$ if $\bigwedge_i u_i \to v_i$ in R, and a substitution σ such that $u|_p =_A l\sigma$, $v = u[r\sigma]_p$, and $\forall i, u_i\sigma \to^*_{R,A,n} w_i$ with $w_i =_A v_i\sigma$. Of course, $\to_{R,A} \subseteq \to_{R/A}$. The important question is the *completeness* question: can any $\to_{R/A}$-step be simulated by a $\to_{R,A}$-step? We say that $\mathcal{R}$ satisfies the A-*completeness* property if for any u, v with sorts in the same connected component we have:



where here and in what follows dotted lines indicate existential quantification.

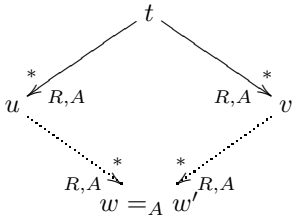
It is easy to check that A-completeness is equivalent to the following (strong) A-coherence[2] (or just *coherence* when A is understood) property:



If a theory $\mathcal{R}$ is not coherent, we can try to make it so by completing the set of rules R to a set of rules $\widetilde{R}$ by a Knuth-Bendix-like completion procedure that computes critical pairs between equations in A and rules in R (see, e.g., [20,25] for the *strong* coherence completion that we use here, and [19] for the equivalent notion of extension completion). For theories A that are combinations of associativity, commutativity, left identity, and right identity axioms, the coherence completion procedure always terminates and has a very simple description (see [24], and for a more informal explanation [4, Section 4.8]).

We say that $\mathcal{R} = (\Sigma, A, R)$ is A-*confluent*, resp. A-*terminating*, if the relation $\to_{R/A}$ is confluent, resp. terminating. If $\mathcal{R}$ is A-coherent, then A-confluence is equivalent to asserting that, for any $t \to^*_{R,A} u$, $t \to^*_{R,A} v$, we have:

[2] Note that the assumption of A being regular and linear is essential for one $\to_{R/A}$-step to exactly correspond to one $\to_{R,A}$-step. For this reason, some authors (e.g., [20,25]) call conditions as the one above *strong coherence*, and consider also weaker notions of coherence.



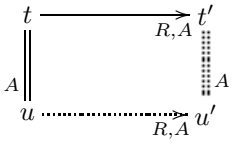
and A-termination is equivalent to the termination of the $\rightarrow_{R,A}$ relation. In what follows, given a rewrite theory $\mathcal{R} = (\Sigma, A, R)$, saying that $\mathcal{R}$ is A-coherent is equivalent to saying that the rules R are A-coherent.

The fact that we are performing *order-sorted* rewriting makes one more requirement necessary. When A-matching a subterm $t|_p$ against a rule's left-hand side to obtain a matching substitution σ, we need to check that σ is well-sorted, that is, that if a variable x has sort s, then the term $x\sigma$ has also sort s. This may however fail to be the case even though there is a term $w \in [x\sigma]_A$ which does have sort s. We call an order-sorted signature *A-preregular* if the set of sorts $\{s \in S \mid \forall w \in \mathcal{T}_\Sigma(\mathcal{X}), \exists w' \in [w]_A \text{ s.t. } w' \in \mathcal{T}_\Sigma(\mathcal{X})_s\}$ has a least upper bound, denoted $ls[w]_A$ which can be effectively computed.[3] Then we can check the well-sortedness of the substitution σ not based on $x\sigma$ above, but, implicitly, on all the terms in $[w]_A$.

Yet another property required for the good behavior of confluent and terminating rewrite theories modulo A is their being *A-sort-decreasing*. This means that $\mathcal{R}$ is A-preregular, and for each term t we have $ls[t]_A \geq ls[t{\downarrow_R}]_A$.

From this, the following lemma follows.

Lemma 1. *For R A-coherent rules, if $t \rightarrow_{R,A} t'$, then*



As mentioned above, for $\rightarrow_{R,A}$ to be decidable we need an A-matching algorithm. Therefore, we will consider the set of equations to be a union $E \cup A$ with A a set of axioms for which there exists a matching algorithm (as associativity, commutativity, and identity), and E the remaining equations.

2.2 Coherence of Conditional Rewrite Theories

A rule $l \rightarrow u_{n+1}$ *if* $\bigwedge_{i=1..n} u_i \rightarrow v_i$ is said to be *deterministic* if $\forall j \in [1..n]$, $\mathcal{V}ar(u_j) \subseteq \mathcal{V}ar(l) \cup \bigcup_{k<j} \mathcal{V}ar(v_k)$. A conditional rewrite theory is *deterministic*

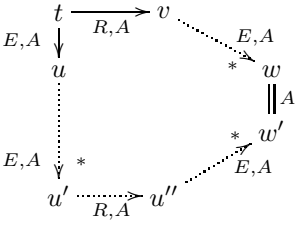
[3] The Maude system automatically checks the A-preregularity of a signature Σ for A any combination of associativity, commutativity, left identity, and right identity axioms (see [4, Chapter 22.2.5]).

if each of its rules is deterministic. Given a rewrite theory $\mathcal{R}$, a term t is called *strongly irreducible* with respect to R modulo A (or *strongly R, A-irreducible*) if $t\sigma$ is a normal form for every normalized substitution σ. A rewrite theory $\mathcal{R}$ is called *strongly deterministic* if for every rule $l \to r$ *if* $\bigwedge_{i=1..n} u_i \to v_i$ in R each v_i is strongly R, A-irreducible.

We assume an order-sorted rewrite theory of the form $\mathcal{R} = (\Sigma, E \cup A, R, \phi)$, where:

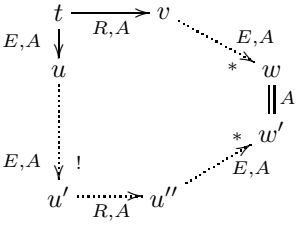
(1) ϕ is the frozenness information [2].
(2) $(\Sigma, E \cup A)$ is an order-sorted equational theory with possibly conditional equations, which can be converted into a strongly deterministic rewrite theory that is *operationally terminating* modulo A. Furthermore, the equations E are *confluent* modulo A. Also, the axioms in A are a collection of regular and linear unconditional equational axioms and are all at the *kind level*, i.e., each connected component in the poset $(S, \leq)$ of sorts has a top sort, and the variables in the axioms A all have such top sorts.
(3) R is a collection of rewrite rules $l \to r$ *if* C, where C is an *equational condition*, which again can be turned into a deterministic rewrite rule of the form $l \to r$ *if* $u_1 \to_E v_1 \wedge \ldots \wedge u_n \to_E v_n$ with the $v_1, \ldots, v_n$ strongly E, A-irreducible.
(4) Both the equations E and the rules R are *A-coherent*. Therefore, the relations $\to_{R/A}$ (resp. $\to_{E/A}$) and $\to_{R,A}$ (resp. $\to_{E,A}$) essentially coincide.

Definition 1. *A rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R, \phi)$ satisfying (1)-(4) above is called* coherent *(resp.* ground coherent*) iff for each Σ-term t (resp. ground Σ-term t) such that $t \to_{E,A} u$, and $t \to_{R,A} v$ we have*



(C)

Likewise, $\mathcal{R}$ is called locally coherent *(resp.* ground locally coherent*) iff for each Σ-term t (resp. ground Σ-term t) such that $t \to_{E,A} u$, and $t \to_{R,A} v$ we have*



(LC)

where dotted arrows are existentially quantified, and $s \to^{!}_{E,A} t$ *iff* $s \to^{*}_{E,A} t$ *and* t *is* E, A*-irreducible.*

Theorem 1. *$\mathcal{R}$ is coherent (resp. ground coherent) iff $\mathcal{R}$ is locally coherent (resp. locally ground coherent).*

Since for all terms t, t is coherent iff t is locally coherent, we can approach the verification of coherence for $\mathcal{R}$ as follows: We can reason by cases on the situations $u \overset{E,A}{\leftarrow} t \overset{R,A}{\rightarrow} v$ depending on whether they are or not *overlap* situations. For this we need the notion of a conditional critical pair, and the notion of conditional critical pair joinability.

Definition 2. *Given conditional rewrite rules with disjoint variables $l \to r$ if C in R and $l' \to r'$ if C' in E, their set of conditional critical pairs modulo A is defined as usual: either we find a non-variable position p in l such that $\alpha \in Unif_A(l|_p, l')$ and then we form the conditional critical pair*

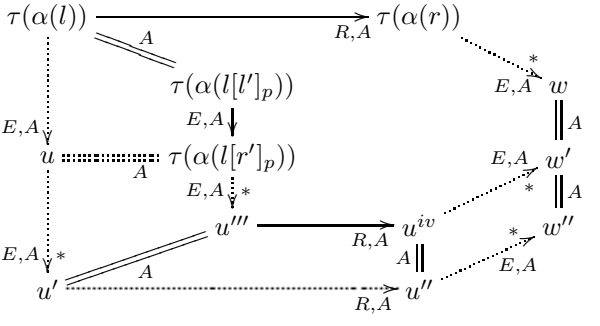
$$\alpha(C) \wedge \alpha(C') \quad \Rightarrow \quad \begin{array}{ccccc} \alpha(l[l']_p) & \overset{A}{=\!=} & \alpha(l) & \underset{R}{\longrightarrow} & \alpha(r) \\ \downarrow{\scriptstyle E} & & & & \\ \alpha(l[r']_p) & & & & \end{array} \qquad (I)$$

or we have a non-variable and nonfrozen *position p' in l' such that $\alpha \in Unif_A(l'|_{p'}, l)$ and we form the conditional critical pair:*

$$\alpha(C) \wedge \alpha(C') \quad \Rightarrow \quad \begin{array}{ccccc} \alpha(l') & \overset{A}{=\!=} & \alpha(l'[l]_{p'}) & \underset{R}{\longrightarrow} & \alpha(l'[r]_{p'}) \\ \downarrow{\scriptstyle E} & & & & \\ \alpha(r') & & & & \end{array} \qquad (II)$$

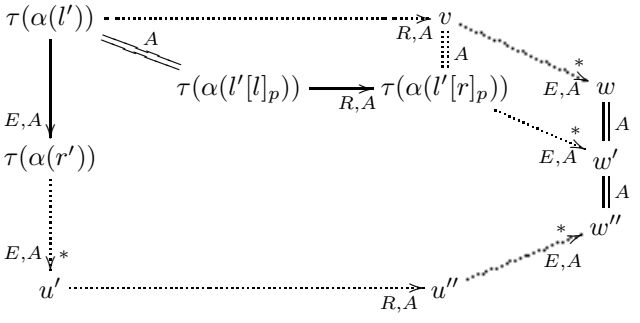
We typically write these critical pairs as $\alpha(C) \wedge \alpha(C') \Rightarrow \alpha(l[r']_p) \to \alpha(r)$ and $\alpha(C) \wedge \alpha(C') \Rightarrow \alpha(r') \to \alpha(l'[r]_{p'})$.

We say that a critical pair of type (I) is *joinable* iff for any substitution τ such that $E \cup A \vdash \tau\alpha(C) \wedge \tau\alpha(C')$ we then have



Of course, by $(C) \Leftrightarrow (LC)$ *it is enough to make this check* with $u''' = u'''{\downarrow_{E,A}}$.

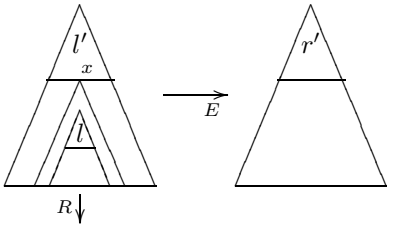
Similarly, we say that a critical pair of type (II) is *joinable* iff for any substitution τ such that $E \cup A \vdash \tau\alpha(C) \wedge \tau\alpha(C')$ we then have



where, again, by $(C) \Leftrightarrow (LC)$ it is enough to perform the check with $u' = u'{\downarrow_{E,A}}$.

Of course, joinability of all conditional critical pairs is a *necessary* condition for coherence. The challenge now is to find a set of *sufficient conditions* for coherence that includes the joinability of conditional critical pairs.

Specifically, non-overlapping situations between equations and rules require additional conditions. In the case of *coherence* checking, we need to worry about non-overlapping of R under E, that is, for $l' \rightarrow_E r'$ *if* C' in E and $l \rightarrow_R r$ *if* C in R we need to worry about situations of the form:



This situation can be problematic in two related ways: (1) when $l' \rightarrow_E r'$ is unconditional but not linear, or (2) when $l' \rightarrow_E r'$ *if* C' is conditional. The problem with (1) is well-understood since [25]. The problem with (2) was also mentioned by Viry in [25]; it has to do with the fact that the satisfiability of the condition C' in an equation $l' \rightarrow_E r'$ *if* C' *depends* on the substitution θ (may hold or not depending on the given θ). But since R *rewrites* the substitution θ, we do *not know if* C' *will hold anymore after a one-step rewrite with the rule* $l \rightarrow_R r$ *if* C.[4]

Theorem 2. *Given* $\mathcal{R}$ *as above, then if:*

(i) all conditional critical pairs are joinable and

[4] Note that we can view cases of unconditional $l \rightarrow r$ with l *non-linear* as special cases of (2), since we can *linearize* l, and give an explicit equality condition instead. E.g., `x + x = x` becomes `x + y = x if x = y`.

(ii) for any equation $l' \to r'$ *if* C' *in* E*, for each* $x \in \mathcal{V}ar(l')$ *such that* x *is non-frozen in* l'*, then* either
   (a) x *is such that* $x \notin vars(C')$*,* x *is also non-frozen in* r'*, and* x *is linear in both* l' *and* r'*, or*
   (b) the sort s *of* x *is such that* no rewriting with $\to_{R,A}$ is possible for terms of such sort s*,*

then $\mathcal{R}$ *is coherent.*

Condition (ii)-(b) of Theorem 2 requires a fixpoint calculation. An algorithm that checks that situations where a non-frozen variable x in a left-hand side of an equation fails to satisfy (ii)-(a) or (ii)-(b) is impossible is provided in [13].

2.3 Context-Joinability and Unfeasibility of Conditional Critical Pairs

From those conditional critical pairs which cannot be joined, the tool can currently automatically discard those that are *context-joinable* or *unfeasible*, based on a result by Avenhaus and Loría-Sáenz [1], which we generalize here to the order-sorted case and modulo A. Let us first introduce some notation.

Let a *context* $C = \{u_1 \to_E v_1, \ldots, u_n \to_E v_n\}$ be a set of oriented equations. We denote by $\overline{C}$ the result of replacing each variable x by a new constant $\overline{x}$, and by $\overline{X}$ the set of such new constants. Given a term t, $\overline{t}$ results from replacing each variable $x \in \mathcal{V}ar(C)$ by the constant $\overline{x}$.

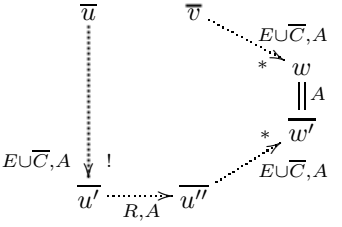
We denote by $\rhd$ the proper subterm relation. Then, given an order $\succ$, we denote by $\succ_{st} = (\succ \cup \rhd)^+$ the smallest ordering that contains $\succ$ and $\rhd$. A partial ordering $\succ$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is *well founded* if there is no infinite sequence $t_0 \succ t_1 \succ \ldots$. A partial ordering $\succ$ is *compatible with substitutions* if $u \succ u'$ implies $u\sigma \succ u'\sigma$ for any substitution σ. A partial ordering $\succ$ is *compatible with the term structure* if $u \succ u'$ implies $t[u]_p \succ t[u']_p$ for any term t and position p in t. A partial ordering $\succ$ is *compatible with the axioms* A if $v =_A u \succ u' =_A v'$ implies $v \succ v'$ for all terms u, u', v, and v' in $\mathcal{T}_\Sigma(\mathcal{X})$. A partial ordering $\succ$ is *A-compatible* if it is compatible with substitutions, compatible with the term structure, and compatible with the axioms A. Then, a *reduction ordering* is a partial ordering that is well founded and A-compatible.

A deterministic rewrite theory $\mathcal{R}$ is *quasi-reductive* w.r.t. a reduction ordering $\succ$ on $\mathcal{T}_\Sigma(\mathcal{X})$ if for every substitution σ, every rule $l \to u_{n+1}$ *if* $u_1 \to v_1 \wedge \ldots \wedge u_n \to v_n$ in R , and every $i \in [1..n]$, $u_j\sigma \succeq v_j\sigma$ for every $j \in [1..i]$ implies $l\sigma \succ_{st} u_{i+1}\sigma$.

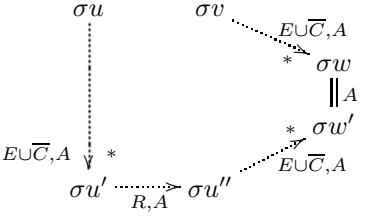
Definition 3. *Let* E *be an order-sorted deterministic term rewrite systems that is quasi-reductive modulo* A *w.r.t. an* A*-compatible order* $\succ$*, and let* $C \Rightarrow s \to t$ *be a conditional critical pair resulting from* $l \to r$ *if* C_1 *in* R *and* $l' \to r'$ *if* C_2 *in* E*, and* $\sigma \in Unif_A(l|_p, l')$ *(resp.* $\sigma \in Unif_A(l'|_q, l)$*). We call* $C \Rightarrow s \to t$ unfeasible *if there are terms* t_0*,* t_1*,* t_2 *such that* $\sigma(l) \succ_{st} t_0$ *(resp.* $\sigma(l') \succ_{st} t_0$*),* $\overline{t_0} \to^*_{E \cup \overline{C},A} t_1$*,* $\overline{t_0} \to^*_{E \cup \overline{C},A} t_2$*, and* t_1*,* t_2 *are not unifyable and strongly* $E \cup \overline{C}$*,* A*-irreducible.*

A Maude order-sorted conditional specification can be converted into an order-sorted deterministic rewrite theory with a simple procedure (see, e.g., [13]). Maude checks that the conditional equational specifications entered are deterministic (c.f. [4]), and we assume it is operationally terminating, and therefore there exists a well-founded A-compatible order $\succ_{st}$ such that we can use the results in [1] and their extension to the Maude case [15], to discard those conditional critical pairs generated that are unfeasible.

Definition 4. *Given a rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R)$, a non-joinable conditional critical pair $C \Rightarrow u \to v$ (coming from a conditional critical pair $C \Rightarrow {}_{E,A}u \swarrow t \searrow {}_{R,A}v$) is* context-joinable *if and only if in the extended rewrite theory $\mathcal{R}_C = (\Sigma \cup \overline{X}, E \cup \overline{C} \cup A, R)$ we have:*



Lemma 2. *If the conditional critical pair $C \Rightarrow u \to v$ is context joinable, then for all substitutions σ such that σC holds we have*



and therefore, the coherence property holds for the conditional critical pair $C \Rightarrow {}_{E,A}u \swarrow t \searrow {}_{R,A}v$.

2.4 The Ground Coherence Case

Assume that Σ has a sub-signature of constructors Ω that has been verified to be *sufficiently complete* with respect to the equations E modulo A. Then, we can view each $f \in \Sigma$ with a different syntactic form from Ω as a *frozen* operator, since any ground term in E, A-canonical form will *not* contain the symbol f. This automatically excludes all problematic non-overlaps with R below E except for:

(i) constructor equations, and

(ii) equations $f(t_1, \ldots, t_n) \to r$ *if* C in E with $f \in \Sigma - \Omega$, and with f having the identity, left identity, or right identity attributes, and such that the left-hand side of the equation resulting from the variant-based transformation to remove the identity attributes has a *non-frozen variable* (see [10] for details on the variant-based transformation).

Therefore, for ground coherence under the assumption of frozenness of defined symbols, we only have to check condition (ii) in Theorem 2 on equations of types (i) and (ii) above.

Furthermore, for a conditional critical pair $\alpha(C) \wedge \alpha(C') \Rightarrow u \to v$ which we have not been able to show context-joinable, we can use constructor-based inductive methods to try to prove its *inductive one-step reachability*, that is, that $\mathcal{R} \vdash_{ind} \alpha(C) \wedge \alpha(C') \Rightarrow u \to v$. We can illustrate such inductive proof methods with a simple example of a rewrite theory operating on cells containing numbers modulo 4, where the rules double the cell's contents each time.

```
(mod DOUBLE is
   sorts Nat/4 State .
   op 0 : -> Nat/4 [ctor] .
   op s : Nat/4 -> Nat/4 [ctor] .
   op `[_`] : Nat/4 -> State [ctor] .
   op _+_ : Nat/4 Nat/4 -> Nat/4 .

   vars N M : Nat/4 .

   eq s(s(s(s(N)))) = N .
   eq N + 0 = N .
   eq N + s(M) = s(N + M)  .

   rl [double-0]: [0] => [0] .
   rl [double-s]: [s(N)] => [s(N) + s(N)] .
endm)
```

The equations in this theory can be proved terminating using Maude's MTT, Church-Rosser using Maude's CRC, and sufficiently complete using Maude's SCC. The ChC gives a nontrivial critical pair:

```
Maude> (check ground coherence .)

Coherence checking of DOUBLE
Coherence checking solution:
The following critical pairs cannot be rewritten:
  cp [#1:Nat/4]
    => [#1:Nat/4 + #1:Nat/4].
```

However, a simple constructor-based induction on `N` proves that

$$\texttt{DOUBLE} \vdash_{ind} \texttt{[N]} \to \texttt{[N + N]}.$$

Indeed, using rule `double-0` we can reduce the base case to checking

$$\texttt{DOUBLE} \vdash_{ind} \texttt{[0]} = \texttt{[0 + 0]},$$

which can be trivially discharged; and the inductive step can be proved by using rule `double-s` and discharging the trivial equality

$$\texttt{DOUBLE} \vdash_{ind} \texttt{[s(N) + s(N)]} = \texttt{[s(N) + s(N)]}.$$

In general, the inductive equational goals generated this way may not be so trivial as in this example, and may require additional steps of inductive equational reasoning. Also, it may be the case that more than one rule can be applied to rewrite the constructor-instantiated lefthand side of a critical pair, so that we get a disjunctive proof obligation. For example, if we had added to the `DOUBLE` module a rule to reset any cell contents to 1, namely,

```
rl [reset]: [N] => [s(0)] .
```

then, in the induction step of the inductive proof for the same critical pair `[N]` $\rightarrow$ `[N + N]`, we would now get the (still equally trivial in this case) disjunctive goal

$$\texttt{DOUBLE} \vdash_{ind} \texttt{[s(N) + s(N)]} = \texttt{[s(N) + s(N)]} \vee \texttt{[s(0)]} = \texttt{[s(N) + s(N)]}.$$

3 How to Use the Maude Coherence Checker

This section illustrates the use of the Maude ChC tool, and suggests some methods that—using the feedback provided by the tool—can help the user establish that his/her specification is ground-coherent.

We assume a context of use in which the user has already developed an *executable specification* of his/her intended system with an initial model semantics, and that this specification has already been checked to have confluent and terminating equations and to have been *tested* with examples, so that the user is in fact confident that the specification is *ground-coherent*, and wants only to check this property with the tool.

The ChC tool not generating any proof obligations is only a *sufficient* condition: in some cases of interest the specification may be *ground* coherent, but not coherent; or may be coherent but the ChC cannot check this automatically because of conditional rules or equations. Then, a collection of critical pairs will be returned by the tool as proof obligations. The ChC does *not* attempt an automatic completion process to add new rules to the user's specification. In many cases this could easily lead to a nonterminating process. Even if such a completion were to terminate, it could easily modify and enlarge the user's specification in undesirable ways. Instead, the feedback of the ChC tool should be used as a guide for *careful analysis* about one's specification. By analyzing the critical pairs returned, the user can understand why they could not be joined. In any case, it is the user himself/herself who must study where the coherence problems come from, and how to fix them by modifying the specification. Interaction with the tool then provides a way of modifying the original specification and ascertaining whether the new version passes the test or is a good step towards that goal.

We present in the following section a simple example that illustrates the use of the tool for different combinations of the associativity, commutativity, and identity axioms. The interested reader can find in [14] additional examples in which conditional equations and rules are used, cases in which conditional critical pairs are discarded using inductive proofs, and so on.

3.1 An Unordered Communication Channel

Consider a communication channel in which messages can get out of order. There is a sender and a receiver. The sender is sending a sequence of data items, for example numbers. The receiver is supposed to get the sequence in the exact same order in which they were in the sender's sequence. To achieve this in-order communication in spite of the unordered nature of the channel, the sender sends each data item in a message together with a sequence number; and the receiver sends back an `ack` message indicating that has received the item. The Full Maude specification of the protocol is as follows:

```
(mod UNORDERED-CHANNEL is
   sorts Nat NatList Msg Conf State .
   subsort Msg < Conf .
   op 0 : -> Nat [ctor] .
   op s : Nat -> Nat [ctor] .
   op nil : -> NatList [ctor] .
   op _;_ : Nat NatList -> NatList [ctor] .    *** list constructor
   op _@_ : NatList NatList -> NatList .       *** list append
   op `[_`,_`] : Nat Nat -> Msg [ctor] .
   op ack : Nat -> Msg [ctor] .
   op null : -> Conf [ctor] .
   op __ : Conf Conf -> Conf [ctor assoc comm id: null] .
   op `{_`,_|_|_`,_`} : NatList Nat Conf NatList Nat -> State [ctor] .

   vars N M J K : Nat .          var  C : Conf .
   vars L P Q : NatList .

   eq nil @ L = L .              eq (N ; L) @ P = N ; (L @ P) .

   rl [snd]: {N ; L, M | C | P, K} => {N ; L, M | [N, M] C | P, K} .
   rl [rec]: {L, M | [N, J] C | P, J}
     => {L, M | ack(J) C | P @ (N ; nil), s(J)}  .
   rl [rec-ack]: {N ; L, J | ack(J) C | P, M}  => {L, s(J) | C | P, M} .
endm)
```

The contents of the unordered channel is modeled as a *multiset* of messages of sort `Conf`. The entire system state, involving the sender, the channel, and the receiver is a 5-tuple of sort `State`, where the components are:

- a buffer for the sender containing the current list of items to be sent,
- a counter for the sender keeping track of the sequence number for items to be sent,
- the contents of the unordered channel,
- a buffer for the receiver storing the sequence of items already received, and
- a counter for the receiver keeping track of the sequence number for items received.

One essential property of this protocol is of course that it achieves *in-order communication* in spite of the unordered communication medium. We can specify this in-order communication property as an *invariant* in Maude. We will assume that all initial states are of the form:

```
  {n1 ; ... ; nk ; nil, 0 | null | nil, 0}
```

That is, the sender's buffer contains a list of numbers `n1 ; ... ; nk ; nil` and has the counter set to `0`, the channel is empty, and the receiver's buffer is also empty. Also, the receiver's counter is initially set to `0`.

In specifying the invariant, the auxiliary notion of a list prefix may be useful. Given lists L and L' we say that L is a *prefix* of L' iff either: (1) $L = L'$, or (2) there is a nonempty list L'' such that $L \ @ \ L'' = L'$.

```
(mod UNORDERED-CHANNEL-INVARIANT is inc UNORDERED-CHANNEL .
   sort Truth .
   ops tt ff : -> Truth [ctor] .
   op _~_ : Nat Nat -> Truth [comm] . *** equality predicate
   op _and_ : Truth Truth -> Truth [assoc comm id: tt] .

   vars M N K P : Nat .          var  C : Conf .
   vars L L' L'' : NatList .     var  B : Truth .

   eq 0 ~ 0 = tt .
   eq 0 ~ s(N) = ff .
   eq s(N) ~ s(M) = N ~ N .
   eq ff and ff = ff .

   op prefix : NatList State -> Truth .
   eq [I1]: prefix(M ; L, {L', N | C | K ; L'', P})
     = (M ~ K) and  prefix(L, {L', N | C | L'', P}) .
   eq [I3]: prefix(L, {L, N | C | nil, K}) = tt .
   eq [I4]: prefix(nil, {L', N | C | M ; L'', K}) = ff .
endm)
```

The equational part of the specification can be checked terminating and Church-Rosser using the MTT [9] and the CRC [14]. And the rules can be shown to be ground coherent with the equations by using the ChC tool.

```
Maude> (check ground coherence .)

Coherence checking of UNORDERED-CHANNEL
Coherence checking solution:
All critical pairs have been rewritten and all equations are non-
    ↪constructor.
The specification is ground coherent.
```

The problem with this simple example is that one cannot verify the invariant using the `search` command in Maude, because, due to the `snd` rule, the number of messages that can be present in the channel is unbounded, so that there is an infinite number of reachable states. One should therefore use an *abstraction*.

```
(mod UNORDERED-CHANNEL-ABSTRACTION is
   pr UNORDERED-CHANNEL-INVARIANT .
   vars M N P K : Nat .
   vars L L' L'' : NatList .
   var  C : Conf .

   eq [A1]: {L, M | [N, P] [N, P] C | L', K}
     = {L, M | [N, P] C | L', K} .
endm)
```

There are of course several key properties that such an abstraction should satisfy:

(1) the set of states reachable from any initial state should be finite,
(2) the equational theory should be confluent and terminating,
(3) the rules should be coherent with the equations, and
(4) the abstraction should preserve the invariant.

Properties (1), (2) and (4) can easily be checked. For (3) we can use the ChC.

```
Maude> (check ground coherence .)

Coherence checking of UNORDERED-CHANNEL-ABSTRACTION
Coherence checking solution:
The following critical pairs cannot be rewritten:
  cp for A1 and rec
    {L:NatList, M:Nat | #3:Conf [N:Nat, J:Nat] | P:NatList, J:Nat}
    => {L:NatList, M:Nat | #3:Conf ack(J:Nat) [N:Nat, J:Nat]
                         | P:NatList ; N:Nat, s(J:Nat)}.
  cp for A1 and rec
    {L:NatList, M:Nat | [N:Nat, J:Nat] | P:NatList, J:Nat}
    => {L:NatList, M:Nat | ack(J:Nat) [N:Nat, J:Nat]
                         | P:NatList ; N:Nat, s(J:Nat)}.
```

These critical pairs indicate that a rule is missing. We can add the rule:

```
(mod UNORDERED-CHANNEL-ABSTRACTION-2 is
   inc UNORDERED-CHANNEL-ABSTRACTION .
   vars M N K : Nat . vars L L' : NatList . var  C : Conf .

   rl [rec2]: {L, M | [N, K] C | L', K}
     => {L, M | [N, K] ack(K) C | L' ; N, s(K)} .
endm)
```

After checking properties (1), (2) and (4) above, we can check also he coherence of the specification.

```
Maude> (check ground coherence .)

Coherence checking of UNORDERED-CHANNEL-ABSTRACTION-2
Coherence checking solution:
All critical pairs have been rewritten, and no rule can be applied
below non-frozen and non-linear variables of equations.
```

4 Conclusions and Future Work

We have presented the theoretical foundations and design of the Maude Coherence Checker. This tool addresses an important need of rewriting logic specifications, namely, checking coherence and ground coherence for very general order-sorted rewrite theories whose equations and rules can be conditional and can be applied modulo various combinations of associativity and/or commutativity and/or identity axioms, and whose operators may have frozenness restrictions. As we have shown, some of these more general requirements, plus the initial model semantics of rewrite theories, can make it in fact *easier* to check coherence and ground coherence than in the much more restrictive untyped, unconditional, and unfrozen case considered by Viry [25]. The tool, together with its documentation, is available at `http://maude.lcc.uma.es/CRChC`.

More work remains ahead. An important issue is that of formal tool integration. The ChC and the CRC are already integrated within a single tool; but as we have explained, the checking of ground coherence can generate inductive equational goals that should be discharged by the Maude ITP. Therefore, a closer integration between the ChC and the ITP would be highly desirable.

Acknowledgements. F. Durán was supported by Spanish Research Projects TIN2008-03107 and P07-TIC-03184. J. Meseguer was partially supported by NSF Grants CCF-0905584, CNS-07-16038, CNS-09-04749, and CNS-08-34709.

References

1. Avenhaus, J., Loría-Sáenz, C.: On conditional rewrite systems with extra variables and deterministic logic programs. In: Pfenning, F. (ed.) LPAR 1994. LNCS, vol. 822, pp. 215–229. Springer, Heidelberg (1994)
2. Bruni, R., Meseguer, J.: Semantic foundations for generalized rewrite theories. Theoretical Computer Science 351(1), 286–414 (2006)
3. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Quesada, J.: Maude: Specification and programming in rewriting logic. Theoretical Computer Science 285, 187–243 (2002)
4. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007)
5. Clavel, M., Durán, F., Hendrix, J., Lucas, S., Meseguer, J., Ölveczky, P.: The Maude formal tool environment. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) CALCO 2007. LNCS, vol. 4624, pp. 173–178. Springer, Heidelberg (2007)
6. Comon-Lundh, H., Delaune, S.: The finite variant property: How to get rid of some algebraic properties. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 294–307. Springer, Heidelberg (2005)
7. Durán, F.: A Reflective Module Algebra with Applications to the Maude Language. PhD thesis, U. de Málaga, Spain (June 1999), `http://maude.csl.sri.com/papers`
8. Durán, F.: The extensibility of Maude's module algebra. In: Rus, T. (ed.) AMAST 2000. LNCS, vol. 1816, pp. 422–437. Springer, Heidelberg (2000)
9. Durán, F., Lucas, S., Meseguer, J.: MTT: The Maude termination tool (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 313–319. Springer, Heidelberg (2008)
10. Durán, F., Lucas, S., Meseguer, J.: Termination modulo combinations of equational theories. In: Ghilardi, S. (ed.) FroCoS 2009. LNCS, vol. 5749, pp. 246–262. Springer, Heidelberg (2009)
11. Durán, F., Meseguer, J.: A Church-Rosser checker tool for Maude equational specifications. Technical Report ITI-2000-5, Dpto. de Lenguajes y Ciencias de la Computación, U. de Málaga (October 2000), `http://maude.cs.uiuc.edu`
12. Durán, F., Meseguer, J.: Maude's module algebra. Science of Computer Programming 66(2), 125–153 (2007)
13. Durán, F., Meseguer, J.: ChC 3: A coherence checker tool for conditional order-sorted rewrite Maude specifications (2009), `http://maude.lcc.uma.es/CRChC`
14. Durán, F., Meseguer, J.: CRC 3: A Church-Rosser checker tool for conditional order-sorted equational Maude specifications (2009), `http://maude.lcc.uma.es/CRChC`
15. Durán, F., Meseguer, J.: A Church-Rosser checker tool for conditional order-sorted equational Maude specifications. In: Ölveczky, P.C. (ed.) 8th Intl. Workshop on Rewriting Logic and its Applications (2010)
16. Durán, F., Meseguer, J.: A Maude coherence checker tool for conditional order-sorted rewrite theories, long version (2010), `http://maude.lcc.uma.es/CRChC`
17. Durán, F., Ölveczky, P.C.: A guide to extending Full Maude illustrated with the implementation of Real-Time Maude. In: Roşu, G. (ed.) Proceedings 7th Intl. Workshop on Rewriting Logic and its Applications (WRLA 2008). Electronic Notes in Theoretical Computer Science. Elsevier, Amsterdam (2008)

18. Escobar, S., Meseguer, J., Sasse, R.: Variant narrowing and equational unification. In: Rosu, G. (ed.) Proc. 7th Intl. Workshop on Rewriting Logic and its Applications (WRLA 2008). Electronic Notes in Theoretical Computer Science, vol. 238, pp. 103–119. Elsevier, Amsterdam (2008)
19. Giesl, J., Kapur, D.: Dependency pairs for equational rewriting. In: Middeldorp, A. (ed.) RTA 2001. LNCS, vol. 2051, pp. 93–108. Springer, Heidelberg (2001)
20. Jouannaud, J.-P., Kirchner, H.: Completion of a set of rules modulo a set of equations. SIAM Journal of Computing 15(4), 1155–1194 (1986)
21. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96(1), 73–155 (1992)
22. Meseguer, J.: A logical theory of concurrent objects and its realization in the Maude language. In: Agha, G., Wegner, P., Yonezawa, A. (eds.) Research Directions in Concurrent Object-Oriented Programming, pp. 314–390. The MIT Press, Cambridge (1993)
23. Ohlebusch, E.: Advanced Topics in Term Rewriting. Springer, Heidelberg (2002)
24. Peterson, G., Stickel, M.: Complete sets of reductions for some equational theories. Journal of ACM 28(2), 233–264 (1981)
25. Viry, P.: Equational rules for rewriting logic. Theoretical Computer Science 285(2), 487–517 (2002)