Electronic Notes in Theoretical Computer Science 86 No. 1 (2003)
URL: `http://www.elsevier.nl/locate/entcs/volume86.html` 14 pages

Reachability in Conditional Term Rewriting Systems

Guillaume Feuillade [1] Thomas Genet [2]

IRISA
Université de Rennes 1 & ENS Cachan
Campus de Beaulieu
F-35042 Rennes

Abstract

In this paper, we study the reachability problem for conditional term rewriting systems. Given two ground terms s and t, our practical aim is to prove $s \not\rightarrow^*_{\mathcal{R}} t$ for some join conditional term rewriting system $\mathcal{R}$ (possibly not terminating and not confluent). The proof method we propose relies on an over approximation of reachable terms for unrestricted join conditional term rewriting systems. This approximation is computed using an extension of the tree automata completion algorithm to the conditional case.

Introduction

In [8], we proposed a technique for approximating the set of reachable terms: given a Term Rewriting System (TRS for short) $\mathcal{R}$ and a regular set of terms E recognized by a tree automaton $\mathcal{A}$, we compute another tree automaton $\mathcal{A}'$ recognizing a super set of terms reachable by rewriting terms of E with $\mathcal{R}$, i.e. $\mathcal{R}^*(E)$. Then, given two terms $s, t \in \mathcal{T}(\mathcal{F})$ and $s \in E$, if t is not recognized by $\mathcal{A}'$ then we have a proof that $s \not\rightarrow^*_{\mathcal{R}} t$. This technique is implemented in the Timbuk tool [11] and have some direct applications in verification where $\mathcal{R}$ is used to model a program behavior, E a set of initial configurations and the super set of reachable terms represent an approximation of every possible execution. An interesting aspect w.r.t. verification is that no assumption is made over $\mathcal{R}$, in particular termination and confluence are not needed. This approach proved to be successful for the verification of cryptographic protocols [9] and was recently applied to verify a cryptographic protocol for pay TV developed by Thomson Multimedia [10].Cryptographic protocol verification

[1] Email: {`Guillaume.Feuillade`}`@irisa.fr`
[2] Email: {`Thomas.Genet`}`@irisa.fr`

©2003 Published by Elsevier Science B. V.

consists in proving that for a fixed set of possible intruder actions, whatever the combination of those actions maybe, the intruder is not able to obtain any secret information or to break authentication between agents. In our setting, we model the protocol by a term rewriting system $\mathcal{R}$ where for every rule, the left-hand side represents the message expected by an agent and the right-hand side models its answer. This term rewriting system is extended by a set of rules describing the intruder actions: usually listening to any message, encrypting and decrypting with any key he has, building and deconstructing messages, etc. The set E contains the initial messages of the protocol and the set $\mathcal{R}^*(E)$ contains every possible message exchanged during the protocol with an intruder listening, replaying, encrypting, decrypting, etc. With Timbuk we can automatically build an over-approximation of $\mathcal{R}^*(E)$ provided that the user gives some approximation rules by hand. Using user defined approximation rules rather than a fixed automatic approximation methodology permits in particular to adapt the precision of the approximation to the property to be proven. Then, we define a set F of forbidden messages representing flaws in the protocol and prove that the intersection between the over-approximation of $\mathcal{R}^*(E)$ and F is empty, proving that no flaw can occur from an initial message whatever the combination of intruder action may be. This technique allowed us to prove automatically security properties on cryptographic protocols under complex verification assumptions: sessions interleaving, unbounded number of sessions, unbounded number of agents.

Now, our aim is to provide in Timbuk some approximation of reachable terms for an extended specification language: Conditional Term Rewriting Systems (CTRS for short). The first motivation for this extension is that CTRS provide a more 'user-friendly' specification language for programs and protocols. A second motivation is that programming languages based on rewriting like Elan [2] or Maude [3] do integrate conditions. Thus to prove properties on Elan and Maude programs using approximations, it is necessary to take conditions into account. Note that these languages also rely on strategies and that there are already some approaches for dealing with reachability under some strategy for some restricted classes of TRS [13].

A first and natural idea to approximate reachability for conditional term rewriting systems is to encode CTRS into TRS and thus reduce the problem of reachability for a conditional term rewriting system to the problem of reachability for a non conditional one. Suprisingly, this is not the more easy and natural way. Thus, in this paper, we propose an extension of the tree automata completion algorithm of [8] to the conditional case.

In section 1, we define TRS, CTRS and tree automata. In section 2, we shortly recall the tree automata completion algorithm for TRS and the approximation construction. In section 3, we define the extension of this algorithm to the CTRS case and show that it produces a tree automaton recognizing an over approximation of reachable terms for join CTRSs. In section 4, we show that the extended algorithm should give better results than using an encoding of CTRS into TRS and the existing algorithm. Finally, we conclude in section 5.

1 TRS, CTRS and tree automata

In this section we shortly present definitions and tools used in this paper. For details about TRS and CTRS one can refer to [5,6] and to [4] for tree automata.

Let $\mathcal{F}$ be a set of function symbols with an arity in $\mathbb{N}$, $\mathcal{X}$ be a set of variables. $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the set of terms over $\mathcal{F}$ and $\mathcal{X}$, $\mathcal{T}(\mathcal{F})$ is its subset of ground terms. $Var(t)$ designates the set of variables of a term t. The set of *positions* of a term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ is a word over $\mathbb{N}$ defined by:

(i) $\mathcal{P}os(t) = \{\epsilon\}$ if $t \in \mathcal{X}$,

(ii) $\mathcal{P}os(f(t_1,\ldots,t_n)) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq n \text{ and } p \in \mathcal{P}os(t_i)\}$ if $f \in \mathcal{F}$, $arity(f) = n$ and $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{X})$, where ϵ is the empty sequence of integers.

If $p \in \mathcal{P}os(t)$ then the subterm of t at position p is denoted by $t|_p$. The term obtained by replacing $t|_p$ in t at position p by the term s is denoted by $t[s]_p$. $\mathcal{P}os(t)$ is a partially ordered set whose order is defined by $p \leq p' \Leftrightarrow \exists q \in \mathcal{P}os(t|_p)$ s.t. $p' = p.q$.

Definition 1.1 A *substitution* is an application $\sigma : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F},\mathcal{X})$ that one can extend to $\mathcal{T}(\mathcal{F},\mathcal{X})$ in an endomorphism: $\sigma : \mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{T}(\mathcal{F},\mathcal{X})$. The result of the *application* of a substitution σ to a term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ is the term denoted by $t\sigma$.

Definition 1.2 A context is a term $C[\,]$ in $\mathcal{T}(\mathcal{F} \cup \{\square\},\mathcal{X})$ where the new constant symbol $\square \notin \mathcal{F}$ appears only once. For all context $C[\,]$ and all term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$, $C[t]$ is the term obtained replacing $\square$ by t in $C[\,]$.

1.1 TRS and CTRS

A term rewriting system (TRS) over a set of ground terms $\mathcal{T}(\mathcal{F})$ is a set $\mathcal{R}$ of pairs $(t_l, t_r) \in \mathcal{T}(\mathcal{F},\mathcal{X})$ called *rules* (and denoted by $t_l \to t_r$) such that $Var(t_r) \subseteq Var(t_l)$. In the following (r) will designate a rule in $\mathcal{R}$. A term $t \in \mathcal{T}(\mathcal{F})$ can be rewritten by the rule $(r) : t_l \to t_r$ at position $p \in \mathcal{P}os(t)$ iff there exists a substitution σ such that $t|_p = t_l\sigma$ and the result of the rewriting is $t[t_r\sigma]_p$. We denote this rewriting by $t \xrightarrow{(r)}_{\mathcal{R}} t[t_r\sigma]_p$. Thus a rewriting system $\mathcal{R}$ defines a *rewriting relation* which is a binary relation $\to_{\mathcal{R}}$ between terms in $\mathcal{T}(\mathcal{F})$ such that $t \to_{\mathcal{R}} t'$ iff there exists a rule (r), a position $p \in \mathcal{P}os(t)$ and a substitution σ such that $t \xrightarrow{(r)} t'$. $\to_{\mathcal{R}}^*$ is the transitive closure of $\to_{\mathcal{R}}$.

A conditional term rewriting system (CTRS) over a set of ground terms $\mathcal{T}(\mathcal{F})$ is a set $\mathcal{R}$ of conditional rules $(r)t_l \to t_r$ if *cond*, where $t_l, t_r \in \mathcal{T}(\mathcal{F},\mathcal{X})$

and *cond* designates a conjunction of conditions that must be checked before rewriting. In this paper, conditions are pairs of terms denoted by $c_1 \downarrow c_2$ where $c_1, c_2 \in \mathcal{T}(\mathcal{F}, \mathcal{X}), (Var(c_1) \cup Var(c_2)) \subseteq Var(t_l)$; these are join conditions. Such a condition is said to be true for a substitution σ if there exists a term $u \in \mathcal{T}(\mathcal{F})$ such that $c_1\sigma$ and $c_2\sigma$ can be both rewritten by the CTRS $\mathcal{R}$ into u in a finite number of steps. Then, the rule $t_l \to t_r$ if $c_1 \downarrow c_2$ can be applied to the term $t \in \mathcal{T}(\mathcal{F})$ at position p as for a TRS. $\to_{\mathcal{R}}$ also defines a rewriting relation on $\mathcal{T}(\mathcal{F})$.

For a TRS or a CTRS $\mathcal{R}$ and a set of terms $E \subseteq \mathcal{T}(\mathcal{F})$, we define the set of $\mathcal{R}$-descendants of E, denoted by $\mathcal{R}^*(E)$ which is the set of reachable terms from E by $\to_{\mathcal{R}}^*$, i.e. $\mathcal{R}^*(E) = \{t \in \mathcal{T}(\mathcal{F}) \mid s \in E \text{ and } s \to_{\mathcal{R}}^* t\}$.

1.2 Tree automata

Tree automata are tools to represent finitely sets of trees and then of terms. A *tree automaton* is a tuple $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, where $\mathcal{Q}$ is a finite set of symbols with arity 0 that are the *states* of the automaton. $\mathcal{Q}_f$ is the subset of $\mathcal{Q}$ of final states and Δ is a set of *normalized transitions*. A normalized transition is a rewriting rule where the left term is $f(q_1, q_2, \ldots, q_n)$ where $f \in \mathcal{F}$ s.t. $arity(f) = n$, $\{q_1, \ldots, q_n\} \subseteq \mathcal{Q}$ and the right term is a state $q \in \mathcal{Q}$. The rewriting relation induced by Δ is denoted by $\to_\Delta$. The language recognized by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists q \in \mathcal{Q}_f \text{ s.t. } t \to_\Delta^* q\}$. The language recognized by a state $q \in Q$ is the set of terms that rewrite into q and is denoted by $\mathcal{L}(\mathcal{A}, q)$. Note that $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in \mathcal{Q}_f} \mathcal{L}(\mathcal{A}, q)$. A set of terms is said to be *regular* if there exists a tree automaton that recognize it.

2 The tree automaton completion algorithm

We first shortly present the algorithm used to approximate $\mathcal{R}^*(E)$ when $\mathcal{R}$ is a TRS [8]. Informally, the algorithm consists in a completion of the tree automaton recognizing E w.r.t the rules of $\mathcal{R}$ to get -step by step- the $\mathcal{R}$-reachable terms from E. Let $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ be a tree automaton over $\mathcal{T}(\mathcal{F})$.

Definition 2.1 A *regular language substitution over* $\mathcal{A}$ is an application $\sigma_{\mathcal{L}} : \mathcal{X} \mapsto \mathcal{Q}$. We can extend this definition to a morphism $\sigma_{\mathcal{L}} : \mathcal{T}(\mathcal{F}, \mathcal{X}) \mapsto \mathcal{T}(\mathcal{F}, \mathcal{Q})$.

For $E = \mathcal{L}(\mathcal{A})$ and for all $q \in \mathcal{Q}$, the completion consists in extending $\mathcal{L}(\mathcal{A}, q)$, the language recognized by q, to the transitive closure of $\mathcal{L}(\mathcal{A}, q)$ by $\mathcal{R}$. To do so, we have to find $s, t \in \mathcal{T}(\mathcal{F})$ and $s \in \mathcal{L}(\mathcal{A}, q)$ such that $(l \to r) \in \mathcal{R}$, $s = l\sigma \to_{\mathcal{R}} r\sigma = t$, and $t \notin \mathcal{L}(\mathcal{A}, q)$. Then, we can add t to $\mathcal{L}(\mathcal{A}, q)$ by adding a set of transitions to Δ s.t. $t \to_\Delta^* q$.

In practice, considering every possible $s \in \mathcal{T}(\mathcal{F})$ such that $s \in \mathcal{L}(\mathcal{A}, q)$ is not possible since it can be infinite. However, it is equivalent to consider every rule $l \to r$ of $\mathcal{R}$ and every possible regular language substitution $\sigma_{\mathcal{L}}$ (which are in finite number) over $\mathcal{A}$ such that $l\sigma_{\mathcal{L}} \to^*_{\Delta} q$ and $r\sigma_{\mathcal{L}} \not\to^*_{\Delta} q$. In this case, $l\sigma_{\mathcal{L}} \to^*_{\Delta} q$ and $l\sigma_{\mathcal{L}} \to^*_{\mathcal{R}} r\sigma_{\mathcal{L}}$ is what we call a *critical pair*. Note that it is equivalent to consider every possible $l\sigma_{\mathcal{L}}$ instead of every possible $l\sigma \in \mathcal{T}(\mathcal{F})$ because if $l\sigma \to^*_{\Delta} q$ then there exists a particular $\sigma_{\mathcal{L}}$ s.t. $l\sigma \to^*_{\Delta} l\sigma_{\mathcal{L}} \to^*_{\Delta} q$. When such a critical pair is found, adding a set of normalized transitions to perform the rewriting $r\sigma_{\mathcal{L}} \to^*_{\Delta} q$ is equivalent to adding a set of normalized transitions to perform the rewriting $r\sigma \to^*_{\Delta} q$ in $\mathcal{A}$ for the same reason. Non left or right linear rules may cause problems due to the filtering process to get the regular language substitution (a same term can be recognized by two different states, then the non-linearity should not apply to states); however there exist a method to go trough this issue described in [11]. When it terminates, this algorithm produces a tree automaton $\mathcal{A}'$ such that for every state q, $\mathcal{L}(\mathcal{A}, q)$ contains $\mathcal{L}(\mathcal{A}, q)$ as well as an *approximation* of its successors by $\mathcal{R}$.However, in many cases the completion produces an infinite number of new states and thus may not terminate. This is due to the fact that transitions added during completion need to be normalized. Normalization of transitions is necessary in order to be able to perform complementation and intersection operations to exploit the result of the completion. Adding transitions to the automaton adds states to the automaton: the left-hand side of $r\sigma \to q$ may be of depth 2 or more, thus it may be necessary to normalize it before adding it to $\mathcal{A}$. In that case, we have to create some new states to recognize subterms of $r\sigma$. Repeatedly adding new states may lead completion to diverge. On the opposite, it is very easy to force termination by limitating the number of new states that can be used and re-use existing states.

Example 2.2 Let us consider the following transition to be normalized in a automaton $\mathcal{A}$ where $q_1 \in \mathcal{Q}$, q_2 is a new state and there is no transition with q_1 in its left hand side:

$$g(f(q_1)) \to q_2$$

The usual way to normalize it is to create a new state, say q_3, and to add the transitions $f(q_1) \to q_3$ and $g(q_3) \to q_2$. An approximation would be, for example, to add these rules instead: $f(q_1) \to q_1$ and $g(q_1) \to q_2$. This would produce, in the new automaton $\mathcal{A}'$ the language $\mathcal{L}(\mathcal{A}', q_2) = \{g(f^n(\mathcal{L}(\mathcal{A}, q_1))) \mid n \in \mathbb{N}\}$ which is a superset of the exact one which is $\{g(f(\mathcal{L}(\mathcal{A}, q_1)))\}$

The approximation we obtain are very close to the widening operations in abstract interpretation. In such techniques, the widening is defined once and it ensures termination in any case. However, if a widening is not precise enough for a given property nothing can be done. Our setting is different, the user is able to give some approximation rules well adapted to its model and to the property he wants to prove. The approximation rules are rewrite rules used to merge together terms or states in order to minimize the number of new states. Standard widening operations on languages can easily be described using a small number of approximation rules [11].

3 Reachability in CTRS by tree automata completion

The purpose of this section is to adapt the existing algorithm of approximation of reachable terms [8] to CTRS. Let $E \subseteq \mathcal{T}(\mathcal{F})$ be a regular entry set of ground terms. We want to get an over approximation of the set $\mathcal{R}^*(E)$ for a given CTRS $\mathcal{R}$. Note that we only focus on left-linear CTRS, since every non left-linear CTRS can be transformed into an equivalent left-linear CTRS: every rule of the form $C[x, x, x] \to \ldots$ can be transformed into $C[x, y, z] \to \ldots$ if $\downarrow (x, y, z)$ where $\downarrow (c_1, \ldots, c_n)$ is an extended joinability condition. Given a substitution σ, $\downarrow (c_1, \ldots, c_n)$ is true if there exists a term u s.t. $c_1\sigma \to_{\mathcal{R}}^* u, \ldots,$ $c_n\sigma \to_{\mathcal{R}}^* u$. This technique extends to any number of non linear occurrence of variables and the extended joinability condition can easily be integrated in the following completion algorithm. Similarly, for sake of simplicity we chose to present the algorithm on rules with a unique joinability condition but it can straightforwardly be lifted to any conjunction of joinability conditions.

3.1 Reachable terms for conditions in a CTRS

Let us consider a conditional-join rule of type $l \to r$ if $s \downarrow t$. For a substitution σ, the term $l\sigma$ rewrites into $r\sigma$ if and only if exists a term u such that $s\sigma \to^* u$ and $t\sigma \to^* u$ with the considered CTRS. Note that this is similar to $l \to r$ if and only if $\mathcal{R}^*(s\sigma) \cap \mathcal{R}^*(t\sigma) \neq \emptyset$. This condition becomes $\mathcal{R}^*(c_1\sigma) \cap \ldots \cap \mathcal{R}^*(c_n\sigma) \neq \emptyset$ for an extended joinability condition $\downarrow (c_1, \ldots, c_n)$. Similarly for any conjunction of conditions $c_1 \downarrow c'_1 \wedge \ldots c_n \downarrow c'_n$, it becomes $\mathcal{R}^*(c_1\sigma) \cap \mathcal{R}^*(c'_1\sigma) \neq \emptyset \wedge \ldots \wedge \mathcal{R}^*(c_n\sigma) \cap \mathcal{R}^*(c'_n\sigma) \neq \emptyset$. Consequently conditions can be evaluated using reachable terms. The way to verify the truth of a condition should be to build the sets $\mathcal{R}^*(s\sigma)$ and $\mathcal{R}^*(t\sigma)$ in order to check if they have common reachable terms or not. Let us remark that performing such a computation with an over-approximation may lead to assign true for a condition that may be false. However this is coherent with an over-approximation of $\mathcal{R}^*(E)$; informally, since conditional rules may apply more often in an over approximation than in the exact case, more descendants are produced, which is still an over approximation of $\mathcal{R}^*(E)$.

Lemma 3.1 *Let $\mathcal{R}$ be a left-linear TRS, $\mathcal{A}'$ be the result of computation of the completion algorithm applied to a set $E = \mathcal{L}(\mathcal{A})$, then $\mathcal{A}'$ is closed by rewriting w.r.t $\mathcal{R}$, i.e: if $l \to_{\mathcal{R}}^* r$ and $\exists q \in \mathcal{Q}_{\mathcal{A}}, l \in \mathcal{L}(\mathcal{A}, q)$ then $r \in \mathcal{L}(\mathcal{A}', q)$.*

The proof of this corollary is described in [8] and is part of the proof for the correctness of the completion method over a TRS. We intend to reuse this intermediate result to compute separately the reachable terms from useful conditional terms.

3.2 Completion over regular set of terms for a CTRS

We first define a rewriting relation $t \xrightarrow{\downarrow n}_{\mathcal{R}} s$ meaning that to rewrite t into s, it is necessary to evaluate at most n conditions (n is called the depth of the derivation in [6]).

Definition 3.2 For a CTRS $\mathcal{R}$ with a subset $\mathcal{R}_{nc}$ of non conditional rules, we note $\xrightarrow{\downarrow n}_{\mathcal{R}}$ the relation defined by:

- $\xrightarrow{\downarrow 0}_{\mathcal{R}} = \rightarrow_{\mathcal{R}_{nc}}$
- $a \xrightarrow{\downarrow n+1}_{\mathcal{R}} b \Leftrightarrow a \xrightarrow{\downarrow 0}_{\mathcal{R}} b$ or $\exists \sigma$ substitution, $p \in \mathcal{P}os(a)$ and $(l \rightarrow r \text{ if } s \downarrow t) \in \mathcal{R}$ such that $a|_p = l\sigma, b = a[r\sigma]_p$ and $\exists u \in \mathcal{T}(\mathcal{F})$ such that $s\sigma \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} u$ and $t\sigma \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} u$.

Note that $l \rightarrow^*_{\mathcal{R}} r$ means that $\exists n \in \mathbb{N}$ s.t. $l \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} r$.

Let $\mathcal{A}_0$ be the tree automaton whose language E is the entry set of terms for the left-linear CTRS R. Let us consider the following algorithm, where we complete at each step the automaton $\mathcal{A}_i$ to an automaton A_{i+1}. The set of state Q_i is partitioned into three set of states: $Q_0 \cup Q_{i,new} \cup Q_{i,cond}$. Q_0 is the set of states of $\mathcal{A}_0$, $Q_{i,new}$ is a set of states produced by transition normalization and indexed by naturals, $Q_{i,cond}$ is a set of conditional states indexed by terms of $\mathcal{T}(\mathcal{F}, Q_i)$.

(i) from $\mathcal{A}_i = \langle \mathcal{F}, \mathcal{Q}_i, \mathcal{Q}_f, \Delta_i \rangle$, the i^{th} step of completion, we compute the automaton $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_f, \Delta_{i+1} \rangle$ with the initialization: $\mathcal{Q}_{i+1} = \mathcal{Q}_i, \Delta_{i+1} = \Delta_i$.

(ii) Let us consider each *critical pair* without considering the condition of the rule. A pair (q, r) of $\mathcal{Q} \times \mathcal{T}(\mathcal{F})$ is said to be critical for a rule (α) either non conditional $l \rightarrow r$, or conditional $l \rightarrow r$ if $c_1 \downarrow c_2$ where $\sigma_{\mathcal{L}}$ is a regular language substitution $\sigma_{\mathcal{L}} = \{x_1 \rightarrow q_{i_1}, x_2 \rightarrow q_{i_2}, \ldots, x_n \rightarrow q_{i_n}\}$, where $\{x_1, x_2, \ldots, x_n\} = var(l)$, if $l\sigma_{\mathcal{L}} \rightarrow^*_{\Delta_i} q$ and $r\sigma_{\mathcal{L}} \not\rightarrow^*_{\Delta_i} q$.

(iii) for all of these critical pairs, (α) is either:
- a conditional rule: $l \rightarrow r$ if $c_1 \downarrow c_2$. There are two possibilities :
  - · there are no state indexed by $c_1\sigma_{\mathcal{L}}$ or $c_2\sigma_{\mathcal{L}}$ in the conditional subset of states of $\mathcal{Q}_i$ ($q_{c_1\sigma_{\mathcal{L}}} \notin \mathcal{Q}_{i,cond}$ or $q_{c_2\sigma_{\mathcal{L}}} \notin \mathcal{Q}_{i,cond}$), then we create these two states (or the one missing) and we add to the automaton $\mathcal{A}_{i+1}$ the following transitions:

$$c_1\sigma_{\mathcal{L}} \rightarrow^*_{\Delta_{i+1}} q_{c_1\sigma_{\mathcal{L}}} \text{ and } c_2\sigma_{\mathcal{L}} \rightarrow^*_{\Delta_{i+1}} q_{c_2\sigma_{\mathcal{L}}}$$

  - · there exists two states $q_{c_1\sigma_{\mathcal{L}}}$ and $q_{c_2\sigma_{\mathcal{L}}}$ in $\mathcal{Q}_i$. We have to calculate $\mathcal{L}(\mathcal{A}_i, q_{c_1\sigma_{\mathcal{L}}}) \cap \mathcal{L}(\mathcal{A}_i, q_{c_2\sigma_{\mathcal{L}}})$. If this set is empty, the condition is, for this completion step, considered as false. If it is not empty, then the condition is true and we go on processing the critical pair as if the rule were not conditional.

- a non conditional one (or it is conditional and the condition has been found true in the previous step), then we add to the automaton the transition $r\sigma_{\mathcal{L}} \rightarrow^*_{\Delta_{i+1}} q$.

(iv) the new automaton $\mathcal{A}_{i+1} = \langle \mathcal{F}, \mathcal{Q}_{i+1}, \mathcal{Q}_{f,i+1}, \Delta_{i+1} \rangle$ is the result of one step of completion of $\mathcal{A}_i$.

If there exists $i \in \mathbb{N}$ such that $\mathcal{A}_i = \mathcal{A}_{i+1}$, then $\mathcal{A}_i$ is the result. Remember that each time we add a transition to the automaton, we have to normalize it with new states (index by naturals and added in $Q_{i,new}$) and then the opportunity to make an approximation in order to limit the number of new states created for the normalization. As in the non conditional case, this completion may not have a fixed point: we may produce infinitely many new states. However, approximation techniques similar to those of section 2 apply: let Q_{cond} be the set of new states $q_{c_1\sigma_{\mathcal{L}}}$ and $q_{c_1\sigma_{\mathcal{L}}}$ produced by conditions, $\mathcal{Q}_{new}$ the set of new states used to normalize the transitions, one may restrict in any way the set $\mathcal{Q}_{new}$ to force completion to terminate. Note that there is no need to limit the number of states of Q_{cond}, since the number of possible conditions c_1, c_2 is finite and the number of possible $\sigma_{\mathcal{L}}$ is finite if $\mathcal{Q}_{new}$ is.

Theorem 3.3 *Let A_0 be a tree automaton such that $\mathcal{L}(\mathcal{A}_0) \supseteq E$ and $\mathcal{R}$ a left linear CTRS. If $\mathcal{A}'$ is the result of the completion of $\mathcal{A}_0$ w.r.t $\mathcal{R}$, then $\mathcal{L}(\mathcal{A}')$ is closed with respect to $\mathcal{R}$ and $\mathcal{R}^*(E) \subseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0)) \subseteq \mathcal{L}(\mathcal{A}')$*

Proof. Let $\mathcal{A}' = \langle \mathcal{F}, \mathcal{Q}', \mathcal{Q}_f, \Delta' \rangle$. We prove that $\forall t \in \mathcal{T}(\mathcal{F})$ s.t. $\exists q \in \mathcal{Q}', t \in \mathcal{L}(\mathcal{A}', q)$, $\forall u \in \mathcal{T}(\mathcal{F})$ s.t. $t \rightarrow^*_{\mathcal{R}} u$, we have $u \in \mathcal{L}(\mathcal{A}', q)$. We prove by induction that $\forall n \in \mathbb{N}, q \in \mathcal{Q}', t \in \mathcal{L}(\mathcal{A}', q)$, u s.t. $t \xrightarrow{\downarrow n}{}^*_{\mathcal{R}} u$, then $u \in \mathcal{L}(\mathcal{A}', q)$

- If $q \in \mathcal{Q}', t \in \mathcal{L}(\mathcal{A}', q)$, and $t \xrightarrow{\downarrow 0}{}^*_{\mathcal{R}} u$ then we trivially have $u \in \mathcal{L}(\mathcal{A}', q)$. Indeed, $\xrightarrow{\downarrow 0}{}^*_{\mathcal{R}}$ means that we consider non conditional R_{nc} subset of rules of $\mathcal{R}$ and then the proof follows from lemma 3.1.

- now suppose that for a given n: $\forall k \leq n, t \xrightarrow{\downarrow k}{}^*_{\mathcal{R}} u$ and $t \rightarrow^*_{\Delta'} q \Rightarrow u \rightarrow^*_{\Delta'} q$. We want to show that:

$$t \xrightarrow{\downarrow n+1}{}^*_{\mathcal{R}} u \text{ and } t \rightarrow^*_{\Delta'} q \Rightarrow u \rightarrow^*_{\Delta'} q$$

$t \xrightarrow{\downarrow n+1}{}^*_{\mathcal{R}} u$ means that exists $\{t_1, t_2, \ldots, t_j\} \subset \mathcal{T}(\mathcal{F})$ such that

$$t_0 = t \xrightarrow{\downarrow n+1}_{\mathcal{R}} t_1 \xrightarrow{\downarrow n+1}_{\mathcal{R}} t_2 \xrightarrow{\downarrow n+1}_{\mathcal{R}} \ldots \xrightarrow{\downarrow n+1}_{\mathcal{R}} t_{j-1} \xrightarrow{\downarrow n+1}_{\mathcal{R}} t_j = u$$

Now we show that for every t_i, if $t_i \rightarrow^*_{\Delta'} q$ then $t_{i+1} \rightarrow^*_{\Delta'} q$, this leads to two cases:

· $t_i \xrightarrow{\downarrow n}_{\mathcal{R}} t_{i+1}$, then using the induction hypothesis, $t_{i+1} \rightarrow^*_{\Delta'} q$.

· $t_i \not\xrightarrow{\downarrow n}_{\mathcal{R}} t_{i+1}$ and $t_i \xrightarrow{\downarrow n+1}_{\mathcal{R}} t_{i+1}$, so there exists a rule $(k)\ l \rightarrow r$ if $c_1 \downarrow c_2 \in \mathcal{R}$ a closed context $C[]$, and a substitution σ such that:

$$t_i = C[l\sigma] \rightarrow_{\mathcal{R}} C[r\sigma] = t_{i+1} \text{ if } c_1\sigma \downarrow c_2\sigma$$
$$\text{and } \exists c \text{ s.t. } c_1\sigma \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} c, \text{ and } c_2\sigma \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} c$$

Since no critical pair between $\mathcal{R}$ and Δ' exists, the automaton is a fixed point for the completion and we necessarily have that $\exists q_{c_1\sigma}, q_{c_2\sigma} \in \mathcal{Q}'$. Thus, we have:

$$c_1\sigma \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} c \text{ and } c_1 \rightarrow_{\Delta'} q_{c_1\sigma}$$
$$c_2\sigma \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} c \text{ and } c_1 \rightarrow_{\Delta'} q_{c_2\sigma}$$

The induction hypothesis leads to $c \in \mathcal{L}(\mathcal{A}', q_{c_1\sigma})$ and $c \in \mathcal{L}(\mathcal{A}', q_{c_2\sigma})$. Consequently, since $t_i \rightarrow^*_{\Delta'} q$, since the condition $\mathcal{L}(\mathcal{A}', q_{c_1,\sigma}) \cap \mathcal{L}(\mathcal{A}', q_{c_2,\sigma}) \neq \emptyset$ is true, and since $\mathcal{A}$ is a fixed point for the completion for automaton $\mathcal{A}$, we necessarily have $t_{i+1} \rightarrow^*_{\Delta'} q$.

We have $t = t_0 \in \mathcal{L}(\mathcal{A}', q)$, so by induction $\forall i \leq n, t_i \in \mathcal{L}(\mathcal{A}', q)$, in particular $u = t_j$.

We get the result that $t \xrightarrow{\downarrow n+1}{}^{*}_{\mathcal{R}} u$ and $t \rightarrow^*_{\Delta} q$ implies $u \rightarrow^*_{\Delta} q$

So $\forall n \in \mathbb{N}, t \xrightarrow{\downarrow n}{}^{*}_{\mathcal{R}} u$ and $t \rightarrow^*_{\Delta'} q$ implies $u \rightarrow^*_{\Delta'} q$, then $t \rightarrow^*_{\mathcal{R}} u$ and $t \rightarrow^*_{\Delta'} q$ implies $u \rightarrow^*_{\Delta'} q$. This leads us to $\forall q \in \mathcal{Q}', \mathcal{L}(\mathcal{A}', q)$ is closed under rewriting by $\mathcal{R}$, in particular for $q \in \mathcal{Q}_f$, thus $\mathcal{L}(\mathcal{A}')$ is closed under rewriting by $\mathcal{R}$. Since completion is incremental, we have the inequalities $\Delta \subseteq \Delta'$ and thus $E \subseteq \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$, and finally $\mathcal{R}^*(E) \subseteq \mathcal{L}(\mathcal{A}')$.

□ □

4 Regular approximations of conditional descendants

The purpose of this section is to compare the automaton we obtain using the completion algorithm on CTRS w.r.t. the automaton obtained by the initial completion algorithm over a TRS encoding a CTRS. We first give an encoding of CTRS into TRS.

4.1 Encoding a CTRS into a TRS

Many encodings of CTRS into TRS were proposed (See for instance [12,1,14]). All those encodings are based on the extension or transformation of the alphabet $\mathcal{F}$: some symbols are added to model conditions or the arity of symbols of $\mathcal{F}$ is extended in order to store some conditions. We require the encoded CTRS to have the same reachable terms on $\mathcal{T}(\mathcal{F})$ though it rewrites over $\mathcal{T}(\mathcal{F}')$, i.e. if $\mathcal{R}$ is a CTRS over $\mathcal{F}$, and $\mathcal{R}'$ is its TRS encoding over $\mathcal{F}'$ (with a function Π mapping (encoded) terms of $\mathcal{T}(\mathcal{F})'$ to terms of $\mathcal{T}(\mathcal{F})$) then for all terms $s, t \in \mathcal{T}(\mathcal{F})$ and $s', t' \in \mathcal{T}(\mathcal{F})'$ such that $s = \Pi(s')$ and $t = \Pi(t')$, we have: $s \rightarrow^*_{\mathcal{R}} t$ if and only if $s' \rightarrow^*_{\mathcal{R}'} t'$. As far as we know the encoding of P. Viry [14] is the only one to preserve reachability. This is mainly due to the fact encodings are generally designed for a different purpose in a different context: most of them are designed to transform a confluent and terminating conditional term rewriting system into a terminating and confluent non conditional term rewriting system having the same normal forms. This is rather different from our setting where systems are not necessarily confluent or terminating.

With regards to approximation construction, we will also require the encoding to minimize the number of 'new' terms (representing condition evaluation and verification). We will also try to minimize their depth as well as the arity of their symbols, since each of these elements increases the complexity of the approximation construction. Intuitively, a 'new' term is a term that is not already recognized by the approximation automaton. Each subterm of a 'new' term requires one state of the automaton to be recognized. Thus minimizing new terms and their depth minimizes the size of the automaton and the complexity – in time and space – of approximation construction. This is why, although Viry's encoding is preserving reachability, it is not fully satisfactory for our purpose. In [7], we propose another encoding of CTRS into TRS:

$$(i) l \to r \text{ if } c_1 \downarrow c_2$$

is transformed into:

$$\begin{aligned} &(i_1)\ l && \to c_i(l, r, c_1, c_2) \\ &(i_2)\ c_i(x, y, z, z) && \to y \end{aligned}$$

where $x, y, z \in \mathcal{X}$. Since this encoding is dedicated to reachability proof for CTRS (in particular it does not have to preserve termination), it is very simple and has good properties for approximations: no arity extension and fewer new terms. For instance, in our setting, it is more interesting to store l itself in $c_i(l, ...)$ than every variable of l – say $c_i(x_1, \ldots, x_n, \ldots)$ for instance – like in [1,14] which increases the arity of c_i. Similarly, since l is the left-hand side of the rule, then the subterm l of $c_i(l, \ldots)$ will be shared in the approximation and thus will not be constructed twice: in $c_i(l, \ldots)$, l is not a 'new' term, it will be recognized by the same state. Note that termination of the system is not required for the approximation, and we will show that recursive applications of the (i_1) rule can easily be solved.

Since this encoding of a CTRS $\mathcal{R}$ into a TRS $\mathcal{R}'$ (over an alphabet $\mathcal{F}' = \mathcal{F} \cup \{c_i | i = 1 \ldots n\}$) preserves reachability, one can apply the existing completion algorithm for unrestricted TRS [11]. The completion algorithm constructs a tree automaton $\mathcal{A}$ whose language is $\mathcal{L}(\mathcal{A}) \supseteq \mathcal{R}'^*(E)$. To approximate $\mathcal{R}^*(E)$, it is then necessary to project the language from $\mathcal{T}(\mathcal{F})'$ into $\mathcal{T}(\mathcal{F})$. This can easily be done by removing all the transitions of the automaton with a c_i as top symbol.

4.2 Example

In order to avoid technical details about approximation rules (see [11]) and to focus on the comparison between the extended completion algorithm and the encoding approach, we choose an example where the completion terminates without approximation.

Example 4.1 For this example, we consider an automaton recognizing the membership of 'b' in lists of 'a' and 'b'. mb designates membership. Let $\mathcal{F} = \{tt, nil, a, b : 0, cons, mb : 2\}$, $\mathcal{X} = \{x, y, l\}$ and $A = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, where $\mathcal{Q} = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, $\mathcal{Q}_f = \{q_5\}$, and

$$\Delta = \begin{cases} nil \to q_0, b \to q_1, a \to q_2, cons(q_2, q_0) \to q_3 \\ cons(q_2, q_3) \to q_3, cons(q_1, q_3) \to q_4, cons(q_2, q_4) \to q_4 \\ cons(q_1, q_4) \to q_4, mb(q_1, q_3) \to q_5, mb(q_1, q_4) \to q_6 \end{cases}$$

$$\mathcal{R} = \begin{cases} mb(x, cons(y, l)) & \to tt \text{ if } mb(x, l) \downarrow tt \\ mb(x, cons(y, l)) & \to tt \text{ if } x \downarrow y \\ mb(x, nil) & \to \perp \end{cases}$$

$$\mathcal{R}' = \begin{cases} mb(x, cons(y, l)) & \to c(mb(x, cons(y, l)), tt, mb(x, l), tt) \\ mb(x, cons(y, l)) & \to c(mb(x, cons(y, l)), tt, x, y) \\ c(x, y, z, z) & \to y \\ mb(x, nil) & \to \perp \end{cases}$$

Note that state q_5 tests membership of 'b' in lists of 'a', and q_6 tests membership of 'b' in lists of 'a' and 'b', with 'a' for last elements. Completion over $\mathcal{A}$ with the CTRS completion method on the conditional system gives:

applied rule	**state**	**conditions**	**conditions state**	**new rules**
$mb(q_1, cons(q_2, q_0)) \to tt$	q_5	$mb(q_1, q_0) \downarrow tt$	new	$mb(q_1, q_0) \to q_{c1}, tt \to q_{c2}$
$mb(q_1, cons(q_2, q_0)) \to tt$	q_5	$q_1 \downarrow q_2$	$false$	
$mb(q_1, cons(q_2, q_3)) \to tt$	q_5	$mb(q_1, q_3) \downarrow tt$	$false$	
$mb(q_1, cons(q_2, q_3)) \to tt$	q_5	$q_1 \downarrow q_2$	$false$	
$mb(q_1, cons(q_2, q_4)) \to tt$	q_6	$mb(q_1, q_4) \downarrow tt$	$false$	
$mb(q_1, cons(q_2, q_4)) \to tt$	q_6	$mb(q_1, q_4) \downarrow tt$	$false$	
$mb(q_1, cons(q_1, q_3)) \to tt$	q_6	$mb(q_1, q_3) \downarrow tt$	$false$	
$mb(q_1, cons(q_1, q_3)) \to tt$	q_6	$q_1 \downarrow q_1$	$true$	$tt \to q_6$
$mb(q_1, nil) \to \perp$	q_{c1}			$\perp \to q_{c1}$

Applied rule and state are the elements of the considered critical pair, conditions are the conditions for the rule to be effectively applicable, condition state is the current state of the condition regarding the existence of the corresponding conditional states in the automaton (*new* if there are not both present), and their intersection (*false* if it is the empty set, *true* otherwise), and new rules refer to the rules added in the automaton after normalization.
Completion over $\mathcal{A}$ with the TRS method on the translated system:

applied rule	**state**	**transition to normalize**	**new rules**
$mb(q_1, cons(q_2, q_0)) \to tt$	q_5	$c(mb(q_1, cons(q_2, q_0)), tt, mb(q_1, q_0), tt)$	$mb(q_1, q_0) \to q_7$
			$tt \to q_8$
			$c(q_5, q_8, q_7, q_8) \to q_5$
$mb(q_1, cons(q_2, q_0)) \to tt$	q_5	$c(mb(q_1, cons(q_2, q_0)), tt, q_1, q_2)$	$c(q_5, q_8, q_1, q_2) \to q_5$
$mb(q_1, cons(q_2, q_3)) \to tt$	q_5	$c(mb(q_1, cons(q_2, q_3)), tt, mb(q_1, q_3), tt)$	$c(q_5, q_8, q_5, q_8) \to q_5$
$mb(q_1, cons(q_1, q_3)) \to tt$	q_6	$c(mb(q_1, cons(q_1, q_3)), tt, mb(q_1, q_3), tt)$	$c(q_6, q_8, q_4, q_8) \to q_6$
$mb(q_1, cons(q_1, q_3)) \to tt$	q_6	$c(mb(q_1, cons(q_1, q_3)), tt, q_1, q_1)$	$c(q_6, q_8, q_1, q_1) \to q_6$
$mb(q_1, cons(q_2, q_4)) \to tt$	q_6	$c(mb(q_1, cons(q_2, q_4)), tt, mb(q_1, q_4), tt)$	$c(q_6, q_8, q_6, q_8) \to q_6$
$mb(q_1, cons(q_2, q_4)) \to tt$	q_6	$c(mb(q_1, cons(q_2, q_4)), tt, q_1, q_2)$	$c(q_6, q_8, q_1, q_2) \to q_6$
$mb(q_1, nil) \to \perp$	q_7	$\perp \to q_7$	$\perp \to q_7$
$c(q_6, q_8, q_1, q_1) \to q_8$	q_6	$q_8 \to q_6$	$q_8 \to q_6$

Applied rule, state and new rules denote the same informations as in the previous completion, while transition to normalize is given in order to show the complexity of the rule to be added with this method and to insist on the approximation. Note that the construction of $\mathcal{A}$ reflect an abstract interpretation choice (every list of 'a' is recognized by the same state q_3, etc). The second method uses approximation to converge while the first one does not need to. The TRS method over the translation produces deep terms to normalize (such as $c(mb(q_1, cons(q_2, q_0)), tt, mb(q_1, q_0), tt)$) we have chosen to normalize using existing states whenever possible. Even with this approximation, the completion produces odd terms such as $c(q_6, q_8, q_6, q_8)$ which are meaningless (q_6 rewrites into q_8 if q_6 rewrites into q_8). For both examples, we obtain $tt \in \mathcal{L}(\mathcal{A}, q_6)$ and $tt \notin \mathcal{L}(\mathcal{A}, q_5)$, and then prove that $mb(b, l) \not\to_{\mathcal{R}}^{*} tt$ when l is a list of $'a'$.
This example also shows that we can replace the two rules:

$$mb(x, cons(y, l)) \to tt \ \text{ if } mb(x, l) \downarrow tt$$

$$mb(x, cons(y, l)) \to tt \ \text{ if } x \downarrow y$$

by one rule :

$$mb(x, cons(y, l)) \to tt \text{ if } mb(x, l) \downarrow tt \text{ or } x \downarrow y$$

and test both conditions in the same step, i.e allow disjunctions of conditions with the CTRS method, while this is not easily possible with a translation.

5 Conclusion

In this paper, we tackle the problem of approximating reachable terms for any join Conditional Term Rewriting System. As far as we know, this is the first time that this problem is addressed. We proposed an algorithm extending the existing automata completion algorithm – implemented in the Timbuk tool – for dealing with CTRS. This extension is rather natural w.r.t. the existing algorithm and uses similar techniques, in particular for approximation construction.

We compared the automata produced by this algorithm with what could be obtained using an encoding of CTRS into TRS and the completion algorithm on TRS. This comparison is in favor of the first one. Even if the encoding was chosen so as to limit the number of new terms, limit their depth as well as their arity, the automaton produced by completion on the encoding is bigger: it recognizes more terms (right-hand side of the rules even if the condition is not true) and contains some alien information (c_i). One should note that each of these additional terms is likely to make the completion diverge and thus needs to be approximated. Hence, the extension of the completion algorithm to CTRS seems to give a more convincing answer to approximate reachable terms for CTRS and should be chosen soon for implementation in Timbuk.

References

[1] I. Alouini and C. Kirchner. Toward the concurrent implementation of computational systems. In *Algebraic and Logic Programming*, pages 1–31, 1996.

[2] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and M. Vittek. ELAN: A logical framework based on computational systems. In *Proc. 1st WRLA*, volume 4 of *ENTCS*, Asilomar (California), 1996.

[3] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 2001.

[4] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. `http://www.grappa.univ-lille3.fr/tata/`, 2002.

[5] N. Dershowitz and J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990. Also as: Research report 478, LRI.

[6] N. Dershowitz, M. Okada, and G. Sivakumar. Canonical Conditional Rewrite Systems. In *Proc. 9th CADE Conf., Argonne (Ill., USA)*, volume 310 of *LNCS*, pages 538–549. Springer-Verlag, 1988.

[7] G. Feuillade and T. Genet. Reachability in conditional term rewriting systems. Technical report, Irisa, Apr. 2003. `http://www.irisa.fr/lande/genet/publications.html`.

[8] T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. 9th RTA Conf., Tsukuba (Japan)*, volume 1379 of *LNCS*, pages 151–165. Springer-Verlag, 1998.

[9] T. Genet and F. Klay. Rewriting for Cryptographic Protocol Verification. In *Proc. 17th CADE Conf., Pittsburgh (Pen., USA)*, volume 1831 of *LNAI*. Springer-Verlag, 2000.

[10] T. Genet, Y.-M. Tang-Talpin, and V. Viet Triem Tong. Verification of Copy Protection Cryptographic Protocol using Approximations of Term Rewriting Systems. Workshop on Issues in the Theory of Security, 2003.

[11] T. Genet and V. Viet Triem Tong. Reachability Analysis of Term Rewriting Systems with timbuk. In *Proc. 8th LPAR Conf., Havana (Cuba)*, volume 2250 of *LNAI*, pages 691–702. Springer-Verlag, 2001.

[12] C. Hintermeier. A transformation of canonical conditional trs's into equivalent canonical trs's. In *Proceedings of the 4th International Workshop on Conditional Rewriting Systems, Jerusalem (Israel)*, June 1994.

[13] P. Réty and J. Vuotto. Regular Sets of Descendants by some Rewrite Strategies. In *Proc. 13th RTA Conf., Copenhagen (Denmark)*, volume 2378 of *LNCS*. Springer-Verlag, 2002.

[14] P. Viry. Elimination of conditions. *Journal of Symbolic Computation*, 28(3):381–401, 1999.