Discrete Mathematics and Theoretical Computer Science **6**, 2004, 143–162

Towards automated proofs of observational properties

Narjes Berregeb[1] and Riadh Robbana[2] and Ashish Tiwari[3]

[1]*Laboratoire d'Informatique de Productique et de Parallélisme, Institut National des Sciences Appliquées et de Technologie, Tunis.*
e-mail: narjes.robbana@insat.rnu.tn
[2]*Laboratoire d'Informatique de Productique et de Parallélisme, Ecole Polytechnique de Tunisie.*
e-mail: riadh.robbana@fst.rnu.tn
[3]*SRI International.*
e-mail: tiwari@csl.sri.com

Observational theories are a generalization of first-order theories where two objects are observationally equal if they cannot be distinguished by experiments with observable results. Such experiments, called contexts, are usually infinite. Therefore, we consider a special finite set of contexts, called cover-contexts, "covering" all the observable contexts. Then, we show that to prove that two objects are observationally equal, it is sufficient to prove that they are equal (in the classical sense) under these cover-contexts. We give methods based on rewriting techniques, for constructing such cover-contexts for interesting classes of observational specifications.

Keywords: observational, contexts, rewriting

1 Introduction

A fundamental aim of formal specifications is to provide a rigorous basis to establish software correctness. Intuitively, a program is correct w.r.t its initial specification if it satisfies all the properties required by this specification. *Behavioural* abstraction provides a suitable basis for a more adequate notion of correctness. In a behavioural or observational theory, two objects are viewed as being identical if they cannot be ditinguished by observable experiments. Thus, for proving the correctness of a program, behavioural (or *observational*) concepts allow to abstract away from internal implementation details, and to focus only on its observable behaviour.

For instance, in the field of object-oriented programming, an observable experiment consists of an application of a *method* that returns certain visible attributes of that object. The actual implementation of the object is not crucial as long as the visible attributes returned by certain methods (and the iterative application of these methods) satisfy the program specification. When formalized using algebraic specifications, method names map to a sorted signature and the observable experiments map to contexts (terms over the signature with a "hole" for the invisible object) with visible sorts.

The idea that the semantics of a specification must describe the behaviour of an abstract datatype is due to [Gut75]. A lot of work has been devoted to the semantical aspects of observability and provability of observational equivalence, see for example [BBK94, Rei95, JR97, HB99, GM99]. In the framework of algebraic specifications, experiments with observable results are represented by particular terms called *observable contexts*. The main difficulty when dealing with proofs of observational properties is that the number of observable contexts is often infinite, and it had been shown in [Sch92] that there is no finite axiomatization of the behavioural equality in first-order logic.

In the framework of initial algebras, an algorithm was given in [BBR98] for constructing a finite set of contexts, that allows to describe the whole set of observable contexts. It applies to specifications represented by a left-linear system. We consider this construction in a more general framework (the semantic we use is not limited to only initial algebra), give more intuitions about it, and show that it can be used by any first order theorem prover for proving observational properties. We call the contexts obtained *cover-contexts*. Then, to prove that two terms are observationally equal, it is sufficient to prove that they are equal (in the classical sense) under these cover-contexts. We also show that the algorithm applies to an interesting class of non left-linear (equational) specifications with observable sorts. Finally, we consider the general case of non left-linear rewriting systems, we propose a procedure which outputs a cover-context set when it terminates.

2 Related work

Hennicker [Hen91] has proposed an induction principle, called *context induction*, which is a proof principle for behavioural abstractions. A context c is viewed as a particular term containing exactly one variable; therefore, the subterm ordering defines a Noetherian relation on the set of observable contexts. Consequently, the principle of structural induction induces a proof principle for properties of contexts of observable sort, which is called *context induction*. This approach provides a uniform proof method for the verification of behavioural properties. It has been implemented in the system ISAR [BH93]. However in concrete examples, this verification is a non trivial task and requires human guidance: the system often needs a generalization of the current induction assertion before each nested context induction, so as to achieve the proof.

Malcolm and Goguen [GM00] suggested doing coinduction proofs by first defining a relation, showing it is a behavioural or hidden congruence, and then showing behavioural equivalence of two terms by showing that they are congruent. This technique, which they call "hidden coinduction," is easily automated only in certain cases where the specification satisfies additional strong restrictions. Note that checking if a relation is a hidden congruence involves establishing observational equivalence. Moreover, if the candidate relation is not a hidden congruence, then users have to find another candidate to complete the proof. The same problems appear in the approach of Bidoit and Hennicker [BH96] where users have to provide partial congruences.

Several other proof tools have been developed to aid coinductive proofs, but all of them require the user to supply an appropriate relation which the system can then prove to be a bisimulation. In the work of [DBG96], an automatic method is given to construct a bisimulation relation, however it uses heuristics and can fail on some examples.

In [MF98], an algorithm is proposed to generate the contextual equality, which coincides with the behavioural equivalence, by eliminating the redundant observational contexts using rewriting techniques. However, this algorithm applies to specifications with only one hidden sort.

3 Basic notions

We assume that the reader is familiar with the basic concepts of algebraic specifications [Wir90], term rewriting and equational reasoning. A many sorted signature Σ is a pair (S,F) where S is a set of sorts and F is a set of function symbols. We assume a partition of F into two subsets C and D of *constructors symbols* and *defined function symbols*. Let X be a family of sorted variables and let $T(F,X)$ be the set of sorted terms. Let $var(t)$ denote the set of variables appearing in t. A term is linear if all its variables occur only once. If $var(t)$ is empty then t is a ground term. The set of all ground terms is also denoted by $T(F)$. A term in $T(C,X)$ is called a *constructor term*. Let A be an arbitrary non empty set, and let $F_A = \{f_A | f \in F\}$ such that if f is of arity n, then f_A is a function from A^n to A. The pair (A,F_A) is called a Σ-algebra, and A is the carrier of the algebra. For sake of simplicity, we will write A to denote the Σ-algebra when F and F_A are non ambiguous. A substitution η assigns terms of appropriate sorts to variables. The domain of η is denoted by $dom(\eta)$. If t is a term, then $t\eta$ denotes the application of η to t. If η replaces every variable by a ground term, then η is a ground substitution. We denote by $\equiv$ the syntactic equivalence between objects.

Let N^* be the set of finite sequences of positive integers. For any term t, $Pos(t) \subseteq N^*$ denotes the set of positions of t, and the expression t/u denotes the subterm of t at position u. The root position is denoted by ε. The depth of a position u, denoted by $|u|$, is the length of the corresponding sequence. We denote by $t(u)$ the symbol of t at position u. A position u in a term t is said to be a strict position if $t(u) \in F$. Let u and v two positions, if there exists a position w such that uw=v then $u \prec v$. We write $t[s]_u$ to indicate that s is a subterm of t at position u. We use also the notation $t[s_1,\ldots,s_n]$ to indicate that the term t contains the subterms $s_1,\ldots,s_n$. The depth of a term t is defined as follows: $|t| = 0$ if t is a constant or a variable, otherwise, $|f(t_1,\ldots,t_n)| = 1 + max_i|t_i|$. An equation is a formula of the form $l = r$. It will be called a rewrite rule and written $l \rightarrow r$ if interest is in the left to right use of this equation. The term l is the left-hand side of the rule. A rewrite rule $l \rightarrow r$ is left-linear if l is linear. A rewrite system is a set of rewriting rules. Let t be a term and u a position in t. We write: $t \rightarrow_R s$ if there exists a rule $l \rightarrow r$ in R and a substitution σ such that $t/u = l\sigma, s/u = r\sigma$ and $t/v = s/v$ for any position v such that $u \not\prec v$. A term t is irreducible (or in normal form) if there is no term s such that $t \rightarrow_R s$. A rewrite system R is said to be terminating if there is no infinite derivation $t_1 \rightarrow_R t_2 \rightarrow_R \cdots$ starting from any term t_1. If $\succ$ is a reduction ordering[†] on terms such that $l \succ r$ for every $l \rightarrow r \in R$, then R is terminating. A term t is ground reducible if all its ground instances are reducible. A valid property $t_1 = t_2$ in R, is denoted by $R \models t_1 = t_2$. A theorem $t_1 = t_2$ of R, is denoted by $R \vdash t_1 = t_2$. An operator $f \in D$ is *sufficiently complete* iff for all $t_1,\ldots,t_n \in T(C)$, there exists $t \in T(C)$ such that $f(t_1,\ldots,t_n) \overset{*}{\rightarrow} t$. A rewriting system R is sufficiently complete if each $f \in D$ is sufficiently complete.

4 Observational Semantics

The notion of observations has been introduced as a means for describing what is observed in a given algebra. Various techniques have been proposed: observations based on sorts, operators, terms or formula (see [BBK94] for a survey). The semantics we choose is based on a relaxing of the satisfaction relation. The notion of context is fundamental in all approaches based on such observational semantics. An observational property is obtained by taking into account only observable information. To show that it is valid, one has to show its validity in all observable contexts.

[†] A reduction ordering is a well-founded ordering that is closed under contexts and substitutions.

specification: STACK
sorts: nat, stack
observable sorts: nat
constructors:
0: →nat
s: nat→ nat
Nil: → stack
push: nat × stack → stack
defined operators
top: stack → nat
pop: stack → stack
axioms:
top(push(x,y))=x
pop(push(x,y))=y

Fig. 1: Stack specification

Definition 1 *(Context) Let $T(F,X)$ be a term algebra and (S,F) be its signature.*

- *a context over F is a non ground term $c \in T(F,X)$ with one distinguished occurrence of a variable called the contextual variable of c. To indicate the contextual variable z_s occurring in c, we often write $c[z_s]$ instead of c, where s is the sort of z_s.*
- *a context reduced to a variable z_s of sort s is called an empty context variable of sort s.*
- *the application of a context $c[z_s]$ to a term $t \in T(F,X)$ of sort s, denoted by $c[t]$, is defined by the substitution of z_s by t in $c[z_s]$. In this case, the context c is said to be applicable to t.*
- *by assumption, $var(c)$ will denote the set of variables occurring in c except the contextual variable of c. A context c is ground if $var(c) = \emptyset$. We denote by $|c|$ the depth of c.*
- *a subcontext (resp. strict subcontext) of c, is a context which is a subterm (resp. strict subterm) of c, having the same contextual variable as c.*

Notations Let $c[z_s]$ and $c'[z'_{s'}]$ be contexts such that c' is of sort s, let t be a term and σ be a substitution such that $z_s \notin dom(\sigma)$. We use the following notations:

- $c[(c'[t])] = (c[c'])[t] = c[c'[t]]$
- $(c[t])\sigma = (c\sigma)[t\sigma] = c[t]\sigma$

Definition 2 *(Specification, Observable specification) A specification (or equational specification) SP is a triple (S,F,E) where (S,F) is a signature and E is a set of equations. An observational specification SP_{obs} is a couple (SP,S_{obs}) such that $SP = (S,F,E)$ is a specification and $S_{obs} \subseteq S$ is the set of observable sorts.*

In the following we denote by $SP_{obs} = (SP,S_{obs})$ an observational specification, where $SP = (S,F,E)$. We denote by R the rewriting system associated with SP.

Example 1 *The Stack specification in Figure 1 is an observational specification where $S_{obs} = \{nat\}$.*

Definition 3 *(Observable context) An observable context is a context whose sort belongs to S_{obs}. The set of observable contexts is denoted by C_{obs}.*

Example 2 *Consider the specification in Figure 1, there are infinitely many observable contexts:*

- $top(z_{stack})$,
- $top(pop(z_{stack})), \dots,$
- $top(pop(\dots(pop(z_{stack}))\dots)), \dots,$
- $top(push(i, z_{stack}))$,
- $top(push(i, pop(z_{stack}))), \dots.$

The notion of observational validity is based on the idea that two objects are equal if they cannot be distinguished by observable contexts.

Definition 4 *(Observational validity) Let t_1, t_2 be two terms. We say that $t_1 = t_2$ is observationally valid, and denote it by $E \models_{obs} t_1 = t_2$, iff for all ground $c_{obs} \in C_{obs}, E \models c_{obs}[t_1] = c_{obs}[t_2]$. We say that t_1 and t_2 are observationally equal and denote it by $t_1 =_{obs} t_2$, iff $E \models_{obs} t_1 = t_2$.*

Note that if $E \models t_1 = t_2$ then $E \models_{obs} t_1 = t_2$, but the converse may not be true. Observational theories generalize first-order theories: if $S_{obs} = S$ then the satisfaction relation $\models_{obs}$ is equal to $\models$.

Example 3 *Consider the Stack specification in Figure 1. It is easy to see that $push(top(s), pop(s)) = s$ is not satisfied (in the classical sense), because $push(top(Nil), pop(Nil)) = Nil$ is not valid. However, it is observationally satisfied if we just observe the elements of the sequences $push(top(s), pop(s))$ and s. This can be formally shown by considering all observable ground contexts.*

5 Cover-contexts

The main problem of proving observational properties is that the number of observable ground contexts is often infinite. We introduce in this section the notion of *cover-contexts* which allow to describe finitely the often infinite set of observable ground contexts.

Definition 5 *(Cover-tree) A cover-tree T_s is a tree such that:*

- *The root is a contextual variable z_s with $s \notin S_{obs}$*
- *The successors of a node which is a non observable context $c[z_s]$ of sort s_i, are all the contexts of the form $f(x_1, \dots, x_{i-1}, c, x_{i+1}, \dots, x_n)$, where:*
  - $f \in F, f: s_1 \times \dots \times s_{i-1} \times s_i \times s_{i+1} \times \dots \times s_n \rightarrow s'$
  - *c is of sort $s_i, (s_i \notin S_{obs})$.*
  - *$x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ are new variables not occuring in $c[z_s]$.*

A path from a node n_1 to a node n_p is a sequence of contexts $(n_1, n_2, \dots, n_p)$ such that each n_i is a successor of n_{i-1}, for $2 \leq i \leq p$. The depth of T, denoted by $depth(T)$ is the length of the longest path of T.

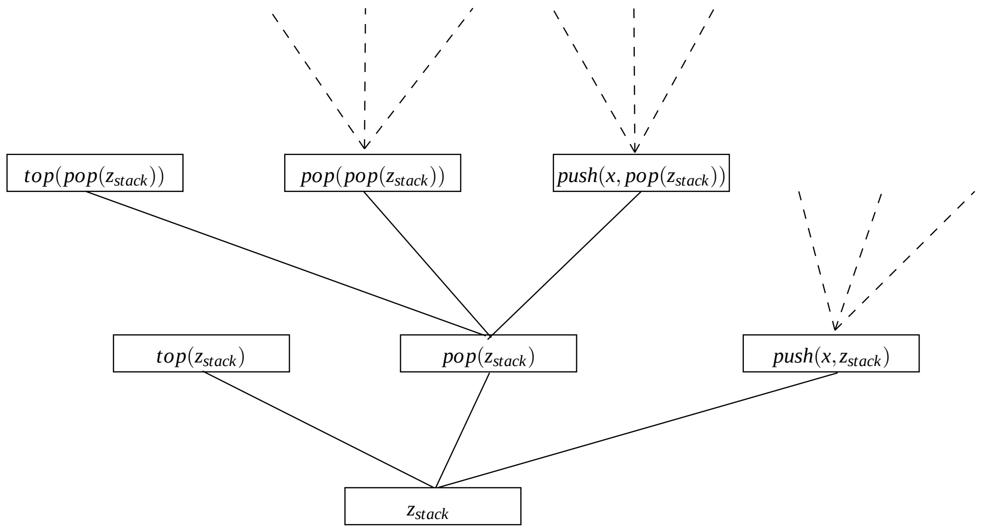

Fig. 2: A cover-tree for the stack specification

Example 4 *A cover-tree T_{stack} for the stack specification is given in Figure 2.*

Note that such cover-tree is often infinite. Each node is a context embedding its predecessor contexts in the same branch, and each leaf is an observable context. An interesting point is that, for proving that $a =_{obs} b$ where a and b have a sort $s \notin S_{obs}$, it is sufficient to consider a finite set of contexts $\{c_1, c_2, \ldots, c_n\}$ such that at least one context c_i occurs in each path starting from the root z_s. In fact, if we have $c_i[a] = c_i[b]$ for each $c_i, 1 \leq i \leq n$, then we can deduce $c_{obs}[a] = c_{obs}[b]$ for each observable context c_{obs}, since by construction of T_s, the contexts $c_i(1 \leq i \leq n)$ are subcontexts of all possible observable contexts. For example, if t_1 and t_2 are terms of sort *stack*, we can show if $t_1 =_{obs} t_2$ by considering only the set of contexts $\{top(z_{stack}), pop(z_{stack}), push(x, z_{stack})\}$. We can also consider the set $\{top(z_{stack}), top(pop(z_{stack})), pop(pop(z_{stack})), push(x, pop(z_{stack})), push(x, z_{stack})\}$. Note that there usually are infinitely many possibilities for choosing a set of contexts like $\{c_1, c_2, \ldots, c_n\}$. An interesting refinement is to consider in T_s, not the whole set of contexts $c[z_s]$, but only contexts that can be embedded in an observable ground irreducible context. We call such contexts *cover-contexts*. In other terms, instead of reasoning on the set of all observable contexts, we consider just an equivalent subset which is the set of ground observable and irreducible contexts.

Definition 6 *(Quasi ground reducibility) A context c is quasi ground reducible if for all ground substitutions τ such that $dom(\tau) = var(c)$, $c\tau$ is reducible.*

Example 5 *The context $top(push(x, z_{stack}))$ is quasi ground reducible. But the context $top(z_{stack})$ is not quasi ground reducible.*

Definition 7 *(Cover-context set) A cover-context set is a finite set of contexts $CC = \{c_1, c_2, \ldots, c_n\}$ such that:*

i/ for each $c_i \in CC$, there exists an observable context c_{obs} such that $c_{obs}[c_i]$ is not quasi ground reducible.

ii/ for each ground irreducible observable context c_{obs}, there exists $c_i \in CC, c'_{obs} \in C_{obs}$ and a ground substitution τ, such that $c_{obs} = c'_{obs}[c_i]\tau$.

Property i expresses the minimality of CC: only contexts that can be embedded in an observable ground irreducible context are retained. Property ii expresses the completeness of CC: each ground observable context is "covered", in some sense, by a context c of CC.

The following lemma shows that cover-contexts are sufficient for proving an observational property in first-order logic.

Lemma 1 *Let R be a terminating rewriting system such that $var(r) \subseteq var(l)$ for each rule $l \to r$ in R, and let S_{obs} denotes the observable sorts.*
Then, $R \models_{obs} t_1 = t_2$ if $R \models c[t_1] = c[t_2]$ for all $c \in CC$.

Proof:
Let c_{obs} be an observable ground context. If c_{obs} is irreducible, then there exists a context $c \in CC$, a context $c'_{obs} \in C_{obs}$ and a ground substitution τ such that $c_{obs} = c'_{obs}[c]\tau$ (by second property of a cover-context set). We have $R \models c[t_1] = c[t_2]$ (by hypothesis), and therefore, $R \models (c'_{obs}[c]\tau)[t_1] = (c'_{obs}[c]\tau)[t_2]$. Thus, $R \models c_{obs}[t_1] = c_{obs}[t_2]$. If c_{obs} is reducible, then there exists an irreducible term t such that $R \models c_{obs}[z] = t$ (since R is terminating). There are three cases here:

(i) If the variable z does not occur in t, then clearly $R \models c_{obs}[t_1] = t$ and $R \models c_{obs}[t_2] = t$, and hence $R \models c_{obs}[t_1] = c_{obs}[t_2]$.

(ii) If the variable z occurs exactly once in t, then $t = c'_{obs}[z]$, where c'_{obs} is an observable irreducible ground context. In this case, the argument given above shows that $R \models c'_{obs}[t_1] = c'_{obs}[t_2]$, and hence $R \models c_{obs}[t_1] = c_{obs}[t_2]$.

(iii) Finally, if the variable z occurs more than once in t, then we consider the context t^1 obtained by replacing all but one occurrences of z by t_1. Then we have: $c_{obs}[t_1] \to_R^* t^1[t_1]$. If t^1 is reducible, we normalize it by R to t' and if z occurs more than once in t', we consider the term t^2 obtained by replacing all but one occurrences of z by t_1. We repeat the process. If this process does not terminate, then we have an infinite derivation $c_{obs}[t_1] \to_R^* t^1[t_1] \to_R^* t^2[t_1] \to_R^* \cdots$, which is impossible since R is terminating. Therefore, the above process terminates with an irreducible term u which contains at most one occurrence of z. Thus, this reduces to either case i or case ii above.

□

6 Computation of cover-context sets for left-linear systems

Let us first introduce some useful definitions. We define $sdepth(R)$ as the maximal depth of strict positions in left-hand sides of R. We define $depth(R)$ as the maximal depth of positions in left-hand sides of R.

The idea of the computation is the following: for all non observable sorts s, we construct a cover-tree T_s of depth equal to $sdepth(R)$. Then we consider the set L of all the leaves of all T_s. Starting from the non quasi ground reducible observable contexts of L, we add all contexts of L that can be embedded in one of those observable contexts, to give a non quasi ground reducible and observable context. The algorithm is given in Figure 3. This computation terminates since the trees T_s are finite, for all $s \notin S_{obs}$.

Since the depth of the cover-trees T_s is $sdepth(R)$, the cover-contexts constructed will have a depth smaller than or equal to $sdepth(R)$. However, we can reduce the number of cover-contexts by considering only subcontexts (for example of depth smaller than or equal to 1) of the constructed cover-contexts. Example 7 illustrates this idea.

The Ground reducibility is decidable for equational rewriting systems [Pla85]. The test of ground reducibility relies on a special finite set of terms called *test set*. A test set is defined as a finite set $TS(R)$ of irreducible terms, such that a term is ground reducible **iff** all its instances by substitutions in $TS(R)$ are reducible. Several algorithms have been proposed for computing a test set, for left-linear rewrite systems (for example see [JK89, SF95]). For non left-linear systems, the computation is more complex than the left-linear case but applies to any rewrite system (for example see [KNZ86, Kou92, SF95]). The constructed test set verifies some important properties (see [Kou92]):

1/ (finiteness): $TS(R)$ is a finite set.

2/ (minimality): For any term t in $TS(R)$, there exists a ground substitution τ such that $t\tau$ is ground and irreducible.

3/ (completeness): For any ground irreducible term s, there exists a term t in $TS(R)$ and a ground substitution τ such that $s = t\tau$.

4/ (transnormality): Every non ground term in $TS(R)$ has an infinite number of ground irreducible instances.

5/ (coverage): Any non ground term t in $TS(R)$ is of depth equal or greater than $depth(R)$.

In the case of a left-linear rewrite system, only conditions 1, 2, 3 and 5 are necessary, besides condition 5 is simplified as follows: Any non ground term t in $TS(R)$ is of depth equal to $sdepth(R)$.

Test substitutions allow to instantiate variables of a context c (or a term t) by elements of the test set whose variables are renamed in order to check the existence of a ground irreducible instance of the context c (or the term t).

Definition 8 *(Test substitution) A test substitution for a context c (resp. a term t) is a substitution that instantiates all the variables in $var(c)$ (resp. the variables in $var(t)$) by terms taken from the test set (whose variables have been renamed).*

Test sets are used to test ground reduciblity, this property can be expressed as follows: Let t be a term and σ be a test substitution such that $t\sigma$ is irreducible, then there exists a ground substitution ρ such that $t\sigma\rho$ is ground and irreducible. The proof of this property in the left-linear case is based on the fact that for all variable x in t, $|x\sigma| = sdepth(R)$, so, no subterm of $t\sigma\rho$ can match a left-hand-side of a rule in R. In the non left-linear case, the proof uses the transnormality property to build an irreducible instance ρ such that $t\sigma\rho$ is ground and irreducible (see for example [Kou92]).

The Quasi ground reducibility is decidable for equational rewriting systems [KC86]. To test whether a context $c[z_s]$ is quasi ground reducible, we apply all test substitutions to $c[z_s]$ and show that they are reducible.

Lemma 2 *A context $c[z_s]$ is quasi ground reducible iff for all test substitutions σ such that $dom(\sigma) = var(c)$, $c[z_s]\sigma$ is reducible.*

```
**for each** non observable sort s, construct a cover-tree $T_s$ of depth $sdepth(R)$.

**let** $LV$ be the set of leaves of all cover-trees $T_s$.

$L := \bigcup_{c \in LV} expand(c)$ where $expand(c)$ is $c$ if $c$ is observable, otherwise $expand(c)$ is obtained from
$c$ by instantiating its variables (except the contextual variable) in all possible ways by terms,
such that the new obtained contexts have all their variables at the same depth $sdepth(R)$.

$CC_0 := \{c \in L | c$ is observable and not quasi ground reducible$\} \cup \{z_s |\ s\ is\ observable\}$
$L_0 := \{c \in L | c$ is not observable$\}$

**repeat**
$CC_{i+1} := CC_i \cup \{c \in L_i | \exists c_i \in CC_i$ such that $c_i[c]$ is not quasi ground reducible$\}$
$L_{i+1} := L_i \setminus CC_{i+1}$
**until** $CC_{i+1} = CC_i$

**output** $CC_i$
```

Fig. 3: Computation of cover-contexts for a left-linear system

Proof:

$\Leftarrow$

Suppose for all test substitutions σ, $c[z_s]\sigma$ is reducible. By property of test sets, we deduce that for all ground substitutions τ such that for all $x \in var(c), x\tau$ is ground and irreducible: there exists a ground substitution ρ such that for all $x \in var(c), x\sigma\rho$ is ground and irreducible. Then, $c[z_s]\sigma\rho$ is reducible. Thus, for all ground substitutions τ such that $dom(\tau) = var(c)$, $c[z_s]\tau$ is reducible.

$\Rightarrow$

Suppose that there exists a test substitution σ such that $c[z_s]\sigma$ is irreducible. We have to show that we can build a ground substitution ρ such that $\forall x \in var(c), x\sigma\rho$ is ground and irreducible, and $c[z_s]\sigma\rho$ is irreducible, and Thus, $c[z_s]$ is not quasi ground reducible. The proof uses the same arguments than for showing that test sets allow to test for ground reducibility. Let us detail the linear case:

From condition 2/ of test sets, there exists a ground substitution ρ such that $\sigma\rho$ is ground and irreducible. Suppose that $c[z_s]\sigma\rho$ is reducible. Then there exists a strict position u, a rule $g \rightarrow d$ and a substitution θ such that $c[z_s]\sigma\rho/u = g\theta$. Note that the position u necessarily occurs in $c[z_s]$ since $\sigma\rho$ is irreducible. Since g is linear and for all x in $c[z_s]$ $|x\sigma| = sdepth(R)$, we can build a substitution α such that for all variable x occuring at a position v of g, $x\alpha = c[z_s]\sigma/uv$. Then, $c[z_s]\sigma/u = g\alpha$. Thus, $c[z_s]\sigma$ is reducible, contradiction.

□

Theorem 1 *Let R be a left-linear rewriting system. Then, the set of contexts CC output by the computation described in Figure 3 is a cover-context set.*

Proof: We have to show that CC is a cover-context set w.r.t Definition 7

i/ Let $c \in CC$. Then there exists i such that $c \in CC_i$. Let us show that there exists an observable context c_{obs} such that $c_{obs}[c]$ is not quasi ground reducible. The proof is by induction on i:

$i = 0$**:** in this case $c \in CC_0$. We set $c_{obs} = z_s$ where s is the sort of c.

$i > 0$**:** there exists $c_{i-1} \in CC_{i-1}$ such that $c_{i-1}[c]$ is not quasi ground reducible. If $c_{i-1} \in C_{obs}$, then we set $c_{obs} = c_{i-1}$. Otherwise, $|c_{i-1}| = sdepth(R)$. By induction hypothesis, there exists an observable context c_{obs} such that $c_{obs}[c_{i-1}]$ is not quasi ground reducible. Let us show that $c_{obs}[c_{i-1}[c]]$ is not quasi ground reducible.

$c_{obs}[c_{i-1}]$ is not quasi ground reducible, therefore, there exists a test substitution σ such that $dom(\sigma) = var(c_{obs}[c_{i-1}])$ and $(c_{obs}[c_{i-1}])\sigma$ is irreducible (by Lemma 2). Let σ_1 be the restriction of σ to $var(c_{obs})$, then $(c_{obs}\sigma_1)[c_{i-1}]$ is still irreducible.
$c_{i-1}[c]$ is not quasi ground reducible, therefore, there exists a test substitution σ_2 such that $dom(\sigma_2) = var(c_{i-1}) \cup var(c)$ and $c_{i-1}[c]\sigma_2$ is irreducible (by Lemma 2).
Let us show that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2$ is irreducible. We can then deduce, by property of test sets, a ground substitution τ such that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2\tau$ is ground and irreducible.
Suppose that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2$ is reducible, then there exists a rule $g \to d$, a substitution θ and a position u such that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2/u = g\theta$. The position u cannot occur in c_{i-1}, otherwise $c_{i-1}[c]\sigma_2$ would be reducible. Then, necessarily, u occurs in c_{obs}. Since g is linear, we can build a substitution α such that for each variable x appearing at a position w of g, $x\alpha = (c_{obs}\sigma_1)[c_{i-1}]/uw$. Then $(c_{obs}\sigma_1)[c_{i-1}]/u = g\alpha$, which contradicts the fact that $(c_{obs}\sigma_1)[c_{i-1}]$ is irreducible

ii/ Suppose that there exists an observable ground irreducible context c_o such that there does not exist $c \in CC$ and a ground substitution τ such that $c\tau$ is a subcontext of c_o. Let us first define the top of a term t as follows:

- $top(t,d) = t$, if $|t| \leq d$
- $top(f(t_1,\ldots,t_n),0) = f(x_1,\ldots,x_n)$, where $x_i (i \in [1..n])$ are fresh variables.
- $top(f(t_1,\ldots,t_n),d) = f(top(t_1,d-1),\ldots,top(t_n,d-1))$ otherwise.

Let d be the depth of the position of the contextual variable of c_o. Let us choose c_o such that d is minimal. Let $c_{obs} = top(c_o,d)$. If $d \leq sdepth(R)$ then $c_{obs} \in CC$, contradiction. Otherwise, let c be a subcontext of c_{obs} of depth $sdepth(R)$, and let c'_{obs} be an observable context such that $c_{obs} = c'_{obs}[c]$. Note that $c'_{obs}[c]$ is not quasi ground reducible. By hypothesis, $c \notin CC$. Let d' be the depth of the contextual variable of c'_{obs}. We have, $d' < d$, necessarily there exists $c' \in CC, \tau', c''_{obs}$ such that $c'_{obs} = c''_{obs}[c']\tau'$. In this case, $c' \in CC$ since $c'[c]$ is not quasi ground reducible, contradiction.

□

Example 6 *Consider the specification in Figure 1. We have:* $sdepth(R) = 1$*. A test set for R is:*

$$\{0, s(x), Nil, push(x,y)\}.$$

Applying the computation principle described in Figure 3, we get:

$$\begin{aligned} CC_0 &= \{z_{nat}, top(z_{stack})\} \\ L_0 &= \{pop(z_{stack}), push(i, z_{stack})\} \end{aligned}$$

In the next iteration, we add the context $pop(z_{stack})$ since $top(pop(z_{stack}))$ is ground and irreducible.

$$\begin{aligned} CC_1 &= \{z_{nat}, top(z_{stack})\} \cup \{pop(z_{stack})\} \\ L_1 &= \{push(i, z_{stack})\} \end{aligned}$$

$CC_2 = CC_1$ is a cover-context set for R.

Definition 9 *Let $f \in D$. We say that f is strongly complete if $f(t_1, \ldots, t_n) \xrightarrow{*} t$ and $t \in T(C,X)$, for all $t_i (i \in [1..n]) \in (T(C,X) \setminus X)$. A rewriting system R is strongly complete if for all $f \in D$, f is strongly complete.*

Note that: if f is strongly complete, then f is sufficiently complete. The converse may not be true. The following theorem states that we can compute a cover-context based on the algorithm given in Figure 3, for a non left-linear rewriting systems provided that it is strongly complete and that the relations between defined functions are left-linear.

Theorem 2 *Let R be a rewriting system strongly complete, such that relations between defined functions are left-linear. We also assume that if a constructor $cons : s_1 \times s_2 \ldots s_n \to s$ is such that if s is observable then $s_1, s_2 \ldots s_n$ are also observable. Then, the set of contexts CC output by the computation described in Figure 3 is a cover-context set.*

Proof:

i/ Let $c \in CC$. Then there exists i such that $c \in CC_i$. Let us show that there exists an observable context c_{obs} such that $c_{obs}[c]$ is not quasi ground reducible. The proof is by induction on i:

$i = 0$**:** in this case $c \in CC_0$. We set $c_{obs} = z_s$ where s is the sort of c.

$i > 0$**:** there exists $c_{i-1} \in CC_{i-1}$ such that $c_{i-1}[c]$ is not quasi ground reducible. If $c_{i-1} \in C_{obs}$, then we set $c_{obs} = c_{i-1}$. Otherwise, $|c_{i-1}| = sdepth(R)$. By induction hypothesis, there exists an observable context c_{obs} such that $c_{obs}[c_{i-1}]$ is not quasi ground reducible. Thanks to the assumptions on constructors, we can choose c_{obs} such that $c_{obs}(\varepsilon) \in D$ or $c_{obs}(\varepsilon) \in X$. If $c_{obs}(\varepsilon) \in X$ (empty context), then we have $c_{obs}[c_{i-1}[c]]$ not quasi ground reducible. Otherwise, we have $c_{obs}(\varepsilon) \in D$. Let us show then, that $c_{obs}[c_{i-1}[c]]$ is not quasi ground reducible.

$c_{obs}[c_{i-1}]$ is not quasi ground reducible, therefore, there exists a test substitution σ_1 such that $dom(\sigma_1) = var(c_{obs})$ and $(c_{obs}\sigma_1)[c_{i-1}]$ is irreducible.

$c_{i-1}[c]$ is not quasi ground reducible, therefore, there exists a test substitution σ_2 such that $dom(\sigma_2) = var(c_{i-1}) \cup var(c)$ and $c_{i-1}[c]\sigma_2$ is irreducible.

Consider all subcontexts $f(t_1, \ldots, t_{k-1}, c'[z_s], t_{k+1}, \ldots, t_n)$ of $c_{obs}[c_{i-1}]\sigma_1\sigma_2$, where t_j is a term for all $j \in \{1, \ldots, k-1, k+1, \ldots, n\}$, and c' is a context. Since $(c_{obs}\sigma_1)[c_{i-1}]$ is irreducible and R is strongly complete, then $t_j \in T(C,X)$. Besides $c_{obs}(\varepsilon) \in D$.

specification: STACK
sorts: nat, stack
observable sorts: nat
constructors:
0: $\rightarrow$nat
s: nat$\rightarrow$ nat
Nil: $\rightarrow$ stack
push: nat $\times$ stack $\rightarrow$ stack
defined operators
top: stack $\rightarrow$ nat
pop: stack $\rightarrow$ stack
axioms:
push(x,push(x,Nil))=push(x,Nil)
top(Nil)=0
top(push(x,y))=x
pop(Nil)=Nil
pop(push(x,y))=y

Fig. 4: A non left-linear specification

Let u_z be the position of z_s in $(c_{obs}\sigma_1)[c_{i-1}[z_s]]$, and u be a position in $(c_{obs}\sigma_1)[c_{i-1}[z_s]]$ such that $u \prec u_z$. Necessarily, $(c_{obs}\sigma_1)[c_{i-1}](u) \in D$, otherwise $(c_{obs}\sigma_1)[c_{i-1}]$ would be reducible since R is strongly complete.

Now, let us show that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2$ is irreducible. We can then deduce, by property of test sets, a ground substitution τ such that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2\tau$ is ground and irreducible. Suppose that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2$ is reducible, then there exists a rule $g \rightarrow d$, a substitution θ and a position u such that $c_{obs}[c_{i-1}[c]]\sigma_1\sigma_2/u = g\theta$. The position u cannot occur in c_{i-1}, otherwise $c_{i-1}[c]\sigma_2$ would be reducible. Then, necessarily, u occurs in c_{obs}.

$u \not\prec u_z$**,** then $(c_{obs}\sigma_1)[c_{i-1}]$ would be reducible. Contradiction.

$u \prec u_z$ then $c_{obs}(u) \in D$. In this case, g is linear since the relations between defined symbols are left-linear. Then, we can build a substitution α such that for each variable x appearing at a position w of g, $x\alpha = (c_{obs}\sigma_1)[c_{i-1}]/uw$. Then $(c_{obs}\sigma_1)[c_{i-1}]/uw = g\alpha$, which contradicts the fact that $(c_{obs}\sigma_1)[c_{i-1}]$ is irreducible.

ii/ The proof of the second property of cover-contexts is similar to the case where R is left-linear.

□

Example 7 *Consider an example of a non left-linear rewriting system given in Figure 4. The rewriting system is strongly complete and the relations between defined operators (top and pop) are left-linear. We have $sdepth(R) = 2$. A test set for R is:*

$$\{0, s(0), s(s(x)), Nil, push(0, Nil), push(s(x), Nil), push(0, push(x,y)), push(s(x), push(x,y))\}$$

Applying the computation principle described in Figure 3, we get:

$$\begin{array}{rcl} CC_0 & = & \{z_{nat}, top(z_{stack}), top(pop(z_{stack}))\} \\ L_0 & = & \{pop(pop(z_{stack})), push(0, pop(z_{stack}), push(s(x), pop(z_{stack}), \\ & & push(0, push(x,z)), push(s(y), push(x,z))\} \end{array}$$

In the next iteration, we add the context $pop(pop(z_{stack}))$ since $top(pop(pop(z_{stack})))$ is ground and irreducible.

$$\begin{array}{rcl} CC_1 & = & \{z_{nat}, top(z_{stack}), top(pop(z_{stack}))\} \cup \{pop(pop(z_{stack}))\} \\ L_1 & = & \{push(0, pop(z_{stack}), push(s(x), pop(z_{stack}), push(0, push(x,z)), \\ & & push(s(y), push(x,z))\} \end{array}$$

$CC_2 = CC_1$ is a cover-context set for R.

We can refine CC_1 by considering only subcontexts of depth smaller than or equal to 1*. This leads to the cover-context set $\{z_{nat}, top(z_{stack}), pop(z_{stack})\}$.*

7 Computation of cover-context sets for non left-linear systems

The algorithm for computing a cover-context set for left-linear rewriting systems does not work for the non-linear case. For example, consider the following rewrite system R:

$$\begin{array}{rcl} g(g(x)) & \rightarrow & x \\ f(g(x),x) & \rightarrow & x \\ f(x,g(x)) & \rightarrow & x \\ f(x,x) & \rightarrow & x \\ f(x,f(x,z)) & \rightarrow & x \\ f(g(x),f(x,z)) & \rightarrow & x \end{array}$$

A test set for R is $TS(R) = \{a, g(a)\}$. Consider the contexts $c_1 = f(x_1,z), c_2 = f(x_2,z), c_3 = f(x_3,z)$. The context $c_1[c_2]$ has an irreducible ground instance which is $f(a, f(g(a),z))$. The context $c_2[c_3]$ has also an irreducible ground instance which is $f(a, f(g(a),z))$. However, the context

$$c_1[c_2[c_3]] = f(x_1, f(x_2, f(x_3,z)))$$

has not an irreducible ground instance. Therefore, the cover-contexts computation for left-linear systems (see Figure 3), does not hold for non left-linear systems, since it is based on the idea that: if there exists $c_i \in CC_i$ such that $c_i[c]$ is a non quasi ground reducible context, then c can be embedded in an observable ground irreducible context. For non left-linear systems, we use a stronger condition; we show that if there exists $c_i \in CC_i$ such that $c_i[c]$ has an infinite number of irreducible instances, then c can be embedded in an observable ground irreducible context. We present in this section a procedure for computing a cover-context set for non left-linear rewriting systems, but which can diverge in some cases. Let us first introduce some useful definitions.

```
for each non observable sort s, construct a cover-tree $T_s$ of depth $depth(R)$.

let $LV$ be the set of leaves of all cover-trees $T_s$.

$L := \bigcup_{c \in LV} expand(c)$ where $expand(c)$ is $c$ if $c$ is observable, otherwise $expand(c)$ is obtained from
$c$ by instantiating its variables (except the contextual variable) in all possible ways by terms,
such that the new obtained contexts have all their variables at the same depth $depth(R)$.

$CC_0 := \{c \in L \mid c$ is observable and not quasi ground reducible$\} \cup \{z_s \mid$ s is observable$\}$
$L_0 := \{c \in L \mid c$ is not observable$\}$

repeat
for each $c \in L_i$ do
    if there exists $c_i \in CC_i$ such that $c_i[c]$ is observable
            and not quasi ground reducible
    then $CC_{i+1} := CC_i \cup \{c\}$
         $L_{i+1} := L_i \setminus \{c\}$
    else if there exists $c_i \in CC_i$ such that $c_i[c]$ is quasi infinitary
              or ground and irreducible
        then $CC_{i+1} := CC_i \cup \{c\}$
             $L_{i+1} := L_i \setminus \{c\}$
        else for each $c_i \in CC_i$ such that $c_i[c]$ is not quasi ground reducible
              and not quasi infinitary do
              extend $c_i[c]$ by all possible ground substitutions $\rho_j$
              such that $c_i[c]\rho_j$ is either ground irreducible or quasi infinitary
              $L_{i+1} := (L_i \setminus \{c\}) \cup (\cup_j c_i[c]\rho_j)$
until $L_{i+1} = L_i$ and $CC_{i+1} = CC_i$
output $CC = CC_i$
```

Fig. 5: Computation of cover-contexts for a non left-linear system

Definition 10 *(Quasi infinitary) A context $c[z_s]$ is quasi infinitary iff there exists a test substitution σ such that $dom(\sigma) = var(c)$, for all $x \in dom(\sigma)$, $x\sigma$ is not ground, and $c[z_s]\sigma$ is irreducible.*

If c is a quasi infinitary context, then c has an infinite number of irreducible instances, thanks to the test set properties (transnormality).

Example 8 *Consider the specification in Figure 4. The context $push(x, z_s)$ is quasi-infinitary since $push(s(y), z)$ is irreducible for all y.*

Theorem 3 *Let R be a non left-linear rewriting system. Then, if the computation described in Figure 5 terminates, the set of contexts CC output is a cover-context set.*

Proof: We have to show that CC is a cover-context set w.r.t Definition 7.

Let $c \in CC$. Then there exists i such that $c \in CC_i$. Let us show that there exists an observable context c_{obs} such that $c_{obs}[c]$ is not quasi ground reducible. The proof is by induction on i:

$i = 0$: in this case $c \in CC_0$. We set $c_{obs} = z_s$ where s is the sort of c.

$i > 0$: there exists $c_{i-1} \in CC_{i-1}$ such that $c_{i-1}[c]$ is not quasi ground reducible. If $c_{i-1} \in C_{obs}$, then we set $c_{obs} = c_{i-1}$. Otherwise, $|c_{i-1}| = depth(R)$. By induction hypothesis, there exists an observable context c_{obs} such that $c_{obs}[c_{i-1}]$ is quasi infinitary or ground irreducible. Let us show that $c_{obs}[c_{i-1}[c]]$ is not quasi ground reducible.

$c_{obs}[c_{i-1}]$ is not quasi ground reducible, therefore, there exists a ground substitution τ_1 such that $dom(\tau_1) = var(c_{obs})$ and $(c_{obs}\tau_1)[c_{i-1}]$ is irreducible.

Suppose $c_{i-1}[c]$ is quasi infinitary (the proof of the case where it is ground and irreducible is a subcase), therefore, there exists a ground substitution τ_2 such that $dom(\tau_2) = var(c_{i-1}) \cup var(c)$, $c_{i-1}[c]\tau_2$ is irreducible and for all $x, y \in var(c_{i-1}[c])$:

$$\begin{array}{rcl} |x\tau_2| & > & |c_{obs}[c_{i-1}]\tau_1| \\ \big||x\tau_2| - |y\tau_2|\big| & > & |c_{obs}[c_{i-1}]\tau_1| \end{array}$$

Let us show that $c_{obs}[c_{i-1}[c]]\tau_1\tau_2$ is irreducible.

Suppose that $c_{obs}[c_{i-1}[c]]\tau_1\tau_2$ is reducible, then there exists a rule $g \to d$, a substitution θ and a position u such that $c_{obs}[c_{i-1}[c]]\tau_1\tau_2/u = g\theta$. The position u cannot occur in c_{i-1}, otherwise $c_{i-1}[c]\tau_2$ would be reducible. Then, necessarily, u occurs in c_{obs}.

- If g is linear, we can build a substitution α such that for each variable x appearing at a position w of g, $x\alpha = (c_{obs}\tau_1)[c_{i-1}]/uw$. Then $(c_{obs}\tau_1)[c_{i-1}]/uw = g\alpha$, which contradicts the fact that $(c_{obs}\tau_1)[c_{i-1}]$ is irreducible.

- If g is not linear, suppose there exists two positions u_1 and u_2 corresponding to a variable x in g such that
$$c_{obs}[c_{i-1}]\tau_1/uu_1 \neq c_{obs}[c_{i-1}]\tau_1/uu_2.$$
but
$$c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1 = c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2$$
Let u_z be the postion of the contextual variable of c_{i-1} in $c_{obs}[c_{i-1}]$.

case 1: u_1 occurs in $c_{obs}\tau_1$. u_2 occurs in c_{i-1}.

case 1.1: $u_2 \prec u_z$ and $u_1 \not\prec u_z$. In this case: $c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2$ is not ground, but
$$c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1$$
is ground, contradiction.

case 1.2: $u_2 \prec u_z$ and $u_1 \prec u_z$. In this case:
$$|c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1| \neq |c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1|,$$
contradiction.

case 1.3: $u_1 \not\prec u_z$ and $u_2 \not\prec u_z$.

- if $c_{obs}[c_{i-1}]/uu_2$ is ground, then $c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2 = c_{obs}[c_{i-1}]\tau_1/uu_2$.
  Besides:
  $$c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2 = c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1.$$
  Therefore
  $$c_{obs}[c_{i-1}]\tau_1/uu_2 = c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1 = c_{obs}[c_{i-1}]\tau_1/uu_1,$$
  contradiction.
- if $c_{obs}[c_{i-1}]/uu_2$ is not ground. Let x be a variable occuring in $c_{obs}[c_{i-1}]/uu_2$. We have:
  $$|x\tau_2| > |c_{obs}[c_{i-1}]\tau_1|.$$
  Therefore:
  $$|c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1| = |c_{obs}[c_{i-1}]\tau_1/uu_1| < |c_{obs}[c_{i-1}]\tau_1\tau_2/uu_2| = |c_{obs}[c_{i-1}]\tau_1\tau_2/uu_1|,$$
  contradiction.

case 2: u_1 occurs in c_{i-1}. u_2 occurs in c_{i-1}.

case 2.1: $u_2 \prec u_z$ and $u_1 \not\prec u_z$ In this case $c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2$ is not ground and $c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1$ is ground, contradiction.

case 2.2: $u_2 \prec u_z$ and $u_1 \prec u_z$. Similar to **case 1.2**

case 2.3: $u_2 \not\prec u_z$ and $u_1 \not\prec u_z$.

- if $c_{obs}[c_{i-1}]/uu_1$ is ground and $c_{obs}[c_{i-1}]/uu_2$ is ground. In this case
  $$c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1 = c_{obs}[c_{i-1}]\tau_1/uu_1$$
  and
  $$c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2 = c_{obs}[c_{i-1}]\tau_1/uu_2.$$
  Therefore:
  $$c_{obs}[c_{i-1}]\tau_1/uu_1 = c_{obs}[c_{i-1}]\tau_1/uu_2,$$
  contradiction
- if $c_{obs}[c_{i-1}]/uu_1$ is ground and $c_{obs}[c_{i-1}]/uu_2$ is not ground. Let x be a variable occuring in $c_{obs}[c_{i-1}]/uu_2$. We have
  $$|x\tau_2| > |c_{obs}[c_{i-1}]\tau_1| \geq |c_{obs}[c_{i-1}]\tau_1/uu_1|.$$
  Therefore:
  $$|c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2| > |c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1|,$$
  contradiction.

– if $c_{obs}[c_{i-1}]/uu_1$ is not ground and $c_{obs}[c_{i-1}]/uu_2$ is not ground. Let x be a variable in $c_{obs}[c_{i-1}]/uu_1$ such that

$$|x\tau_1| = max_{x_i \in c_{obs}[c_{i-1}]/uu_1} |x_i\tau_2|.$$

Let y be a variable in $c_{obs}[c_{i-1}]/uu_2$ such that

$$|y\tau_2| = max_{y_i \in c_{obs}[c_{i-1}]/uu_2} |y_i\tau_2|.$$

Suppose that $|x\tau_2| > |y\tau_2| + |c_{obs}[c_{i-1}]\tau_1|$. Therefore:

$$|c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_1| > |c_{obs}[c_{i-1}[c]]\tau_1\tau_2/uu_2|,$$

contradiction.

The second part of the proof is similar to that of Theorem 1. □

8 Conclusion

We have presented an algorithm for computing a *cover-context set* which is a finite description of the often infinite set of observable contexts. We have shown that this computation applies to left-linear rewriting systems as well as an interesting class of non left-linear rewriting system, with any number of observable sorts. In the general case of a non left-linear system, we have proposed a procedure for computing a cover-context set. Once a cover-context set is computed, it is possible to use any first order theorem prover to prove observational properties, provided we use the *context induction rule* given below:

$$\frac{\forall c[z_s] \in CC, \quad c[t_1] = c[t_2]}{t_1 = t_2} \quad \text{where } t_1, t_2 \text{ are terms of non observable sort. } s$$

We plan to extend the computation of cover-context sets for a more general class of conditional specifications, and to use more refined observations based on operators or terms.

Acknowledgements

We would like to thank the referees for their valuable comments.

References

[BBK94] G. Bernot, M. Bidoit, and T. Knapik. Behavioural approaches to algebraic specifications: a comparative study. *Acta Informatica*, 31(7):651–671, 1994.

[BBR98] N. Berregeb, A. Bouhoula, and M. Rusinowitch. Observational proofs with critical contexts. In *Fundamental Approaches to Software Engineering*, volume 1382 of *Lecture Notes in Computer Science*. Springer Verlag, 1998.

[BH93] B. Bauer and R. Hennicker. Proving the correctness of algebraic implementations by the isar system. In *DISCO'93*. Springer-Verlag, 1993.

[BH96] M. Bidoit and R. Hennicker. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science*, 165(1):3–55, 1996.

[DBG96] L. Dennis, A. Bundy, and I. Green. Using a generalisation critic to find bisimultations for coinductive proofs. In *14th Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Computer Science*, pages 276–290, 1996.

[GM99] J. Goguen and G. Malcolm. Hidden coinduction: Behavioral correctness proofs for objects. *Mathematical Structures in Computer Science*, 9(3):287–319, 1999.

[GM00] J. Goguen and G. Malcolm. A hidden agenda. *Theoretical Computer Science*, 245(1):55–101, 2000.

[Gut75] J. Guttag. *The specification and application to programming of abstract data types*. PhD thesis, University of Toronto, 1975.

[HB99] R. Hennicker and M. Bidoit. Observational logic. In *Algebraic Methodology and Software Technology*, volume 1548 of *Lecture Notes in Computer Science*, pages 263–277, 1999.

[Hen91] R. Hennicker. Context induction: a proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing*, 3(4):3–55, 1991.

[JK89] J. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82:1–33, 1989.

[JR97] B. Jacobs and J. Rutten. A tutorial on coalgebras and coinduction. *EATCS Bulletin*, 62:222–259, 1997.

[KC86] S. Kaplan and M. Choquer. On the decidability of quasi-reducibility. *Bulletin of European Association for Theoretical Computer Science*, 1986.

[KNZ86] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1986.

[Kou92] E. Kounalis. Testing for the ground (co-)reducibility property in term-rewriting systems. *Theoretical Computer Science*, 106:87–117, 1992.

[MF98] M. Matsumoto and K. Futatsugi. Test set coinduction: Toward automated verification of behavioural properties. In *2nd International Workshop on Rewriting Logic and its Applications*, Electronic Notes in Theoretical Computer Science. Elsevier Science, 1998.

[Pla85] D. Plaisted. Semantic confluence and completion method. *Information and Control*, 1985.

[Rei95] H. Reichel. An approach to object semantics based on terminal co-algebras. *Mathematical Structures in Computer Science*, 5:129–152, 1995.

[Sch92] O. Schoett. Two impossibly theorems on behavioural specifications. *Acta Informatica*, 29:595–621, 1992.

[SF95] K. Shmid and R. Fettig. Towards an efficient construction of test sets for deciding ground reducibility. In *6th Conference on Rewriting Techniques and Applications*, volume 914 of *Lecture Notes in Computer Science*, 1995.

[Wir90] M. Wirsing. *Algebraic specifications*, chapter 13. MIT press, 1990.