Annotated Term Rewriting for Deciding Observation Congruence[1]

Raúl Monroy[2] and **Alan Bundy**[3] and **Ian Green**[3]

Abstract. The use of term-rewriting systems (TRSs) is at the heart of automated theorem proving (ATP). There are a number of methods whereby we can build terminating TRSs with full deduction power, with respect to some theory. Yet, there exist theories these techniques cannot handle, giving rise to a trade-off between deduction power and termination. One such theory is the theory of observation congruence in CCS [9]. One approach to overcome the problem is to use heuristic to control search; it, however, introduces the problem of finding and capturing such meta-level control. This paper advocates the use of annotated TRSs (ATRSs) to tackle such problems. Annotations are used to capture meta-level information that helps direct proof search; they give an intuitive account of why and how rewriting is expected to work. We introduce an ATRS called Observant and show how it can be used to build a decision procedure for observation congruence in CCS; furthermore, Observant can be used to prove CCS equations involving finite-state systems and infinite-state systems. We prove Observant terminating, sound, and complete – over the normalisation of finite processes, and so argue that Observant surpasses rival term-rewriting strategies.

1 Introduction

The use of term-rewriting systems (TRSs) is at the heart of automated theorem proving.

Desirable properties of TRSs include *termination* and *confluence*. A TRS is terminating if there is no infinite sequence of rewriting steps in it. A TRS is confluent if whenever[4] $s \overset{*}{\Leftarrow}: t \overset{*}{:\Rightarrow} q$, then there exists a term t' such that $s \overset{*}{:\Rightarrow} t' \overset{*}{\Leftarrow}: q$ [3]; moreover, a TRS is *canonical* if it is both terminating and confluent. If $\mathcal{R}$ is also terminating and finite, then $\mathcal{R}$ constitutes a decision procedure over the equality of two terms in Γ.

Some theories give rise to no canonical TRS. On the one hand, termination may fail to hold when the TRS includes axioms like commutativity $X + Y :\Rightarrow Y + X$, or when two rules come from an equation used both ways round, e.g. $X + (Y + Z) :\Rightarrow (X + Y) + Z$ and $(X + Y) + Z :\Rightarrow X + (Y + Z)$. On the other hand, the use of completion strategies [8] to some theories may result in an infinite TRS; for example, the superposition of rules may give rise to infinitely many critical pairs that do not reduce to the identity relation [6]. The theory of observation congruence in CCS, a Calculus of Communicating Systems [9], has a divergent completion [7].

Experience has shown that, often, it is possible to use meta-level information to overcome the above problem. So, [3] has suggested to extend the rewriting rule of inference with a meta-level applicability condition. R_OBS [7], a canonical TRS for normalising CCS terms in the theory of observation congruence, adopts this approach. The Dershowitz's approach, however, exhibits 3 main disadvantages. First, inefficiency: rewrite rule application depends on global conditions; the overall state must be therefore tested, updated and validated. Second, specialisation: the output TRS is geared towards a theory, constraining its applicability in other domains. Third, specialists are required for discovering, formalising, and mechanising the meta-level control information.

Another approach to capture meta-level control in TRSs is the use of annotations. An annotated TRS is a TRS in which the (object-level) terms are provided with (meta-level) decorations. The aim of the annotations are twofold: i) to give an account as to why and how the ATRS should work (*expectancy*), and ii) to direct the search for a solution, since matching includes annotations (*efficiency*). ATRSs are, thus, goal-oriented: they often involve little search; moreover, due to the expectancy property, it is possible to analyse failure of proof attempts and patch them. Rippling [1] is one such annotated TRS; it has been successfully and extensively applied in several domains, including automated inductive theorem proving, summing series [11], limit theorems [12], etc. This range of applications might provide evidence that lifting meta-level control information to annotations can extend the applicability of a search procedure (*generality*); hence, avoiding the specialisation problem inherent in Dershowitz's approach.

This paper reports on an experiment on the use of ATRSs: we present an ATRS called Observant to control proof search in the theory of observation congruence. Observant can be made to play a chief role in both process normalisation (over observation congruence), and equation solving (over various behavioural classes, including infinite state processes). Here, we will see that rewriting with Observant is terminating and carefully guided, pruning significantly the proof search space.

[1] The research reported in this paper was supported by both EPSRC grant GR/L/11724, and CONACyT grant 64745.

[2] Departamento de Ciencias Computacionales, ITESM – Campus Estado de México; Apdo. Postal 50, Módulo de Servicio Postal Campus Edo. de México del ITESM; 52926 Atizapán, México; raulm@campus.cem.itesm.mx

[3] Department of Artificial Intelligence, The University of Edinburgh; 80 South Bridge, EH1 1HN, Edinburgh, UK; bundy, img@dai.ed.ac.uk

[4] The symbol $:\Rightarrow$ denotes a term-rewriting relation; furthermore, let $\mathcal{R}$ be a relation, then $\mathcal{R}^*$ stands for the transitive, reflexive closure of $\mathcal{R}$.

© 1998 R. Monroy and A. Bundy and I. Green
ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

Furthermore, we will see that Observant is sound and complete over the normalisation of finite processes, and argue that Observant surpasses rival term-rewriting strategies.

The rest of the paper is organised as follows: In §2, we discuss CCS and the theory of observation congruence; there, we shall look into the absorption lemma, a key result for process normalisation. §3 presents Observant, while §4 shows its properties. In §5, we summarise results, while, in §6, we compare with related work, and draw conclusions.

2 CCS

Terms of CCS represent processes; processes have their own identity, circumscribed by their entire capabilities of interaction with both other agents, and the environment. Such interactions are communicating activities; they are referred to as *actions*. An action is said to be *observable*, if it denotes an interaction between one agent and its environment; otherwise, it is said to be *silent*, or *unobservable*. This interpretation of observation underlies a precise and amenable theory of behaviour: whatever is observable is regarded as the behaviour of a system; two agents are concluded equivalent, if their behaviour is indistinguishable to an external observer.

Syntax The set of actions, $\mathcal{A}ct$, ranged over by meta-variables $\alpha, \beta, \ldots$, contains names (e.g. in), co-names (e.g. $\overline{in}$) and the unobservable action τ, which denotes internal communication. The set of processes, $\mathcal{P}$, ranged over by meta-variables $P, Q, \ldots$, contains expressions with the abstract syntax equation: $P ::= \mathbf{0} \mid \alpha.P \mid P+P$. $\mathbf{0}$ denotes the deadlock process, capable of no actions whatever. The combinator *Prefix*, $(.)$, is used to convey the discrete actions an agent may perform; for example, $\alpha.P$ denotes a process capable of executing the action α, and then evolving into the process P. The combinator *Summation*, $(+)$, disjoins the capabilities of the agents that appear either side of it; as soon as one performs any action, the other is dismissed.

Semantics CCS is given meaning by means of the labelled transition system $(\mathcal{P}, \mathcal{A}ct, \{\xrightarrow{\alpha} : \alpha \in \mathcal{A}ct\})$, where $\xrightarrow{\alpha}$ is the smallest transition relation satisfying the rules below:

Act $\alpha.P \xrightarrow{\alpha} P$, for every $\alpha \in \mathcal{A}ct$; and
Sum If $P_1 \xrightarrow{\alpha} P'$ or $P_2 \xrightarrow{\alpha} P'$ then $P_1 + P_2 \xrightarrow{\alpha} P'$

Let $P \stackrel{\alpha}{\Rightarrow} P'$ be $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* P'$; then, whenever $P \stackrel{\alpha}{\Rightarrow} P'$, P' is called an α-*descendant* of P.

Process Equivalence Different interpretations of what is observable give rise to different equivalence relations. *Observation congruence*, $\approx^c$, is the only relation that supplies two essential ingredients for process analysis: the abstraction of internal actions, and the property of being a congruence. $\approx^c$ has been characterised as the maximum congruence defined by $\mathcal{A}$, the axiom system below.[5] $\mathcal{A}$ has been proved sound and complete for $\approx^c$, i.e. $P \approx^c Q$ iff $\mathcal{A} \vdash P = Q$ [5].

Definition 1 (The Axiom System $\mathcal{A}$)

A1: $P + Q = Q + P$
A2: $P + (Q + R) = (P + Q) + R$
A3: $P + P = P$
A4: $P + \mathbf{0} = P$
A5: $\alpha.\tau.P = \alpha.P$
A6: $\tau.P + P = \tau.P$
A7: $\alpha.(\tau.Q + P) + \alpha.Q = \alpha.(\tau.Q + P)$

[5] We omit the rules for reflexivity, symmetry and substitutivity of equality.

$\mathcal{A}$ does not give rise to a canonical TRS for process normalisation; the superposition of the rules below, derived from **A3**, **A6** and **A7**, gives rise to an infinite number of critical pairs, that do not reduce to the identity. Moreover, process normalisation involves rewriting with the following rules:

$r1$: $P :\Rightarrow P + P$
$r2$: $\tau.P :\Rightarrow \tau.P + P$
$r3$: $\alpha.(\tau.Q + P) :\Rightarrow \alpha.(\tau.Q + P) + \alpha.P$

and, hence, is essentially non-terminating.

To see why the use of $r1$, $r2$, and $r3$ is mandatory, consider the following proof search example. To prove that $\tau.(P+Q)+P = \tau.(P+Q)$, we have to "saturate" the term $\tau.(P+Q)$, using $r2$, leaving $\tau.(P+Q)+(P+Q)+P = \tau.(P+Q)$; then "re-group" the term $(P+Q)+P$ to $(P+P)+Q$, leaving $\tau.(P+Q)+((P+P)+Q) = \tau.(P+Q)$; then "absorb" P into P yielding $\tau.(P+Q)+(P+Q) = \tau.(P+Q)$; and, finally, absorb $P+Q$ into $\tau.(P+Q)$, using the rule $\tau.P+P :\Rightarrow \tau.P$.

These subtleties will be all handled by Observant, which we introduce in §3.

2.1 Deciding Observation Congruence

When proving the completeness of $\mathcal{A}$, [5] hinted how to construct a method for deciding observation congruence, based on process normalisation. Roughly, the normal form of a process corresponds to a process which contains no summand that can be "absorbed" by another; put in another way, normal form means zero redundancies, in terms of process summands and external aspects of behaviour. For example, $P+P$ is not a normal form, as $P+P=P$. More precisely,

Definition 2 (Normal Form) *P is a* sumform *if it is of the form, written $\equiv$, $\sum_{i=1}^{n} \mu_i.P_i$, where each P_i is a sumform.*

A sumform $P \equiv \sum_{i=1}^{n} \mu_i.P_i$ is a proper normal form *if i) $P \not\equiv \tau.P'$, ii) each P_i is a proper normal form; and iii) for $i \neq j$, P_i is not congruent to any μ_i-descendant of $\mu_j.P_j$; that is, it is* not *the case that $P_i \stackrel{\mu_j}{\Rightarrow} P_j$, for $j \neq i$.*

$\tau.P$ is an improper normal form, *if P is a proper normal form.*

A normal form *is either a proper or an improper normal form.*

The key condition in the definition of proper normal form is iii); it states the absorption above mentioned, and formally captured by the following lemma, which is slightly stronger to that given in [5]:

Lemma 3 (absorption) *Let P be a sumform, then*

$$P \stackrel{\alpha}{\Rightarrow} Q \text{ implies } \mathcal{A} \vdash P + \alpha.Q = P.$$

Whenever $P \stackrel{\alpha}{\Rightarrow} Q$, we call P *solvent*; reciprocally, we call $\alpha.Q$ the *excess*. Naturally, by the transition semantics, i.e. the rules **Act** and **Sum**, $P \stackrel{\alpha}{\Rightarrow} Q$ implies both $(P + \mu.R) \stackrel{\alpha}{\Rightarrow} Q$, and $(\mu.R + P) \stackrel{\alpha}{\Rightarrow} Q$. Whenever $P \stackrel{\alpha}{\Rightarrow} Q$ and P is just a single process, i.e. not of the form $P_1 + P_2$, we call P *absorbent*.

Thus, by means of absorption, we can eliminate excesses; moreover, if applied recursively on the depth of the term structure – starting from the deepest levels, absorption produces

the normal form of the input term (nf) [5]. The aim of Observant is to drive an application of absorption; so, to normalise P, we use the following procedure:

$$\begin{aligned}&\mathrm{nf}(P) = \\ &\quad \textbf{if } P \equiv \mathbf{0} \textbf{ then } \mathbf{0};\\ &\quad \textbf{if } P \equiv \alpha.P' \textbf{ then } \mathrm{Observant}^{\star}(\alpha.\mathrm{nf}(P'))\\ &\quad \textbf{if } P \equiv (\sum_{i=1}^{n} P_i) \textbf{ then } \mathrm{Observant}^{\star}(\sum_{i=1}^{n} \mathrm{nf}(P_i))\end{aligned}$$

nf hints at this paper's ultimate goal: Given $P, \alpha.Q \in \mathcal{P}$, such that $P \stackrel{\alpha}{\Rightarrow} Q$, drive the use of $\mathcal{A}$, by means of the rewrite-rule of inference, to absorb $\alpha.Q$ into P.

3 Observant: The Rewriting Strategy

3.1 Annotations

Observant operates on annotated terms; these are as follows:

1. unannotated terms are Observant terms;
2. Take P and Q such that $P \stackrel{\alpha}{\Rightarrow} Q$ and assume they contain no redundant $\mathbf{0}$ terms, then:
   - the excess, $\alpha.Q$, is marked with a dotted circle, $\overset{\cdot}{\alpha.Q}$; and
   - using the transition rules, the solvent, P, is marked with solid circles, so as $P \stackrel{\alpha}{\Rightarrow} Q$ is annotated as $P' \overset{\alpha}{\Rightarrow} \overset{\cdot}{Q}$, where P' is as P except that it contains no annotations;
3. annotated expressions yielded by Observant rewritings are Observant expressions; and
4. only annotated expressions formed using these rules are Observant expressions.

To rewrite $P + \alpha.Q$ into P, provided that $P \stackrel{\alpha}{\Rightarrow} Q$, first we identify and annotate both the excess and the absorbent, and, then, we apply Observant, defined below.

3.2 Annotated Rewrite-Rules

Observant uses annotated rewrite-rules; these rules are classified according to the task they perform: tidying/merging, attraction, saturation and absorption, and applied following this order, until no rule of any class is applicable. After each rule application, this order is restarted; this is the so-called *waterfall approach*. The rules should be applied using a call-by-value search strategy to guarantee completeness.

3.2.1 Tidying/Merging

Tidying eliminates internal τ actions: $\alpha\ .\tau.\ P \ :\Rightarrow\ \alpha.P$,

while merging glues the annotated term $\alpha\ .\ Q$ into $\alpha.Q$.

Merging is a pure meta-level step, i.e. there is no corresponding object-level proof step.

3.2.2 Attraction

Attraction moves the excess next to the absorbent.[6] It applies the following rules, which assume that the absorbent appears before the excess, when the expression is read left to right.[7]

$$\begin{aligned}&(C_1(\ P\ ) + Q) + C_2(\ R\ ) :\Rightarrow (C_1(\ P\ ) + C_2(\ R\ )) + Q\\ &(Q + C_1(\ P\ )) + C_2(\ R\ ) :\Rightarrow Q + (C_1(\ P\ ) + C_2(\ R\ ))\\ &C_1(\ P\ ) + (Q + C_2(\ R\ )) :\Rightarrow (C_1(\ P\ ) + C_2(\ R\ )) + Q\\ &C_1(\ P\ ) + (C_2(\ R\ ) + Q) :\Rightarrow (C_1(\ P\ ) + C_2(\ R\ )) + Q\end{aligned}$$

3.2.3 Saturation

The aim of Saturation is to enable the absorption of the excess and to keep track of the absorbent movements.[8]

$$\begin{aligned}&\tau.C(\ P\ ) + \ Q\ :\Rightarrow \lfloor \tau.C(P) \rfloor + (C(\ P\ ) + \ Q\ )\\ &\alpha\ .(\tau.C(\ Q\ ) + P) + \ R\ :\Rightarrow\\ &\qquad \lfloor \alpha.(\tau.C(Q) + P) \rfloor + (\ \alpha\ .C(\ Q\ ) + \ R\ )\end{aligned}$$

Saturation pops up a copy of the absorbent, P , up the term structure into the next level; furthermore, it marks the terms onto which it applies with this piece of notation $\lfloor \ldots \rfloor$; such a mark serves two purposes: i) it gives a means of undoing the saturation steps in order to obtain the original solvent, and ii) it avoids the saturation of an already saturated term.

3.2.4 Absorption

The aim of the absorption rules is twofold, according to the sort of annotation. First, it absorbs the excess into the absorbent:

$$\begin{aligned}P\ +\ P\ &:\Rightarrow\ P\\ \tau.\ P\ +\ P\ &:\Rightarrow\ \tau.P\\ \alpha\ .(\tau.\ Q\ + P) + \ \alpha.Q\ &:\Rightarrow\ \alpha.(\tau.Q + P)\end{aligned}$$

Second, it undoes previous saturation steps, if any, by getting rid of the absorbent:

$$\begin{aligned}\lfloor \tau.P \rfloor + P\ &:\Rightarrow\ \tau.P\\ \lfloor \alpha.(\tau.Q + P) \rfloor + \alpha.Q\ &:\Rightarrow\ \alpha.(\tau.Q + P)\end{aligned}$$

The $\lfloor\rfloor$ markers cause absorption to unwind the saturation steps in the right order; this is because the solvent is left with no circles after saturation and, so, there should be no markers in the absorbent for absorption to apply.

---

[6] N.B. the absorbent is static.
[7] C_i ($i \in \{1, 2\}$) stands for a second-order variable.
[8] C is other than the identity function.

3.2.5 The Waterfall

Procedurally, the steps are interpreted as follows:

- After tidying/merging, use attraction to move the excess close to the summand that contains the absorbent subcomponents. Then apply absorption.
- If absorption is successful, then terminate;
- Otherwise, proceed to saturate. Saturation will eventually enable absorption, and, then, absorption will get rid of the excess and apply the saturation steps in reverse order, using the tracks left by saturation.

An example is illustrative. Consider that we wish to absorb $\mu.R$ into $\tau.(\tau.Q + \mu.R)$. The absorption is possible, since, by operational semantics, we know that $\tau.(\tau.Q+\mu.R) \stackrel{\mu}{\Rightarrow} R$. Take the following context:[9]

$$\ldots + \tau.(\tau.Q + \mu.R) + \mu.R + \ldots$$

Term annotation yields:

$$\ldots + \tau.(\tau.Q + \ \mu.R\ ) + \ \mu.R\ + \ldots$$

Attraction is inapplicable, because the excess and the absorbent are next to each other; absorption is also inapplicable, because the expression does not match any of the absorption rules. Only saturation can be used, it yields

$$\ldots + (\lfloor \tau.(\tau.Q + \mu.R) \rfloor + (\tau.Q + \ \mu.R\ )) + \ \mu.R\ + \ldots$$

Then, the waterfall starts again. With this formula, the use of attraction is suggested. Because attraction moves the excess close to the absorbent, the structure of the saturated term is preserved; this will allow us to go back along the path marked by saturation.

The absorption rules will complete the job now. First, the excess is removed:

$$\ldots + \lfloor \tau.(\tau.Q + \mu.R) \rfloor + (\tau.Q + \mu.R) + \ldots$$

then, the saturation step is undone, leaving:

$$\ldots + \tau.(\tau.Q + \mu.R) + \ldots$$

just as required.

4 Properties of Observant

4.1 Termination

We analyse each rewriting class first.

Proposition 4 *Tidying and Merging are terminating*

Proof. Trivial: the number of τs and circles are reduced. □

Proposition 5 *Absorption terminates*

Proof. The measure is the number of annotated summands. Each rule eliminates both a summand, and the annotations of the absorbent; absorption is measure decreasing. □

[9] Henceforth, ellipsis will be used to denote expressions that remain unchanged.

Proposition 6 *Attraction terminates*

Proof. The measure for attraction rules is the distance between the terms $C(\ P\ )$ and $\ R\ $, defined as the size of the smallest subexpression that contains all occurrences. Since the variable Q in the attraction rules contains no annotations, the rules for attraction are measure decreasing. □

Proposition 7 *Saturation terminates*

Proof. The measure is the number of τs on the path from the root to the occurrence of $C(\ P\ )$; this is reduced by both saturation rules. □

Theorem 1 *Observant terminates*

Proof. No rewriting step can increase the measure of absorption, saturation, or attraction. By Proposition 7, the absorbent is moved up the term structure, into the level where the excess is, in a finite number of steps, and because no rewriting step introduces any further excess, it follows that attraction is used only a finite number of times each pass through the waterfall. Then, by Proposition 6, each time that attraction is used, it terminates. Finally, by Proposition 5, each time that absorption is used, it terminates. It then follows that Observant terminates. □

4.2 Soundness and Completeness

Theorem 2 *Observant is sound: if it rewrites* $P + \ \alpha.Q$ *into* P*, then* $P + \alpha.Q = P$

Proof. Trivial: annotated rewrite-rules are decorated object-level rewrite rules, each preserves equivalence; so, it follows that $P + \ \alpha.Q\ :\Rightarrow^* P$ implies $P + \alpha.Q = P$. □

Theorem 3 *Observant is complete: if* $P \stackrel{\alpha}{\Rightarrow}\ Q\ $*, then Observant is guaranteed to rewrite* $P + \ \alpha.Q\ $ *into* P'*, where* P' *is as* P *except that it has no annotations.*

Proof. As in [5], we proceed by induction on the structure of an α-descendant, i.e. $P \stackrel{\alpha}{\Rightarrow} Q$ iff $P(\stackrel{\tau}{\rightarrow})^* \stackrel{\alpha}{\rightarrow} (\stackrel{\tau}{\rightarrow})^* Q$; so, we consider three cases:

Case i) if $P \stackrel{\alpha}{\rightarrow}\ Q\ $, then $P + \ \alpha.Q\ :\Rightarrow^* P'$, and P' is the result of deleting the annotations from P. Since P is a sumform, then, by transition semantics, $P \stackrel{\alpha}{\rightarrow}\ Q\ $ implies that P is of the form: $S + \ \alpha.Q\ + T$, where S and T stand for some other, possibly empty, context. In that case, we prove that

$$(S + \ \alpha.Q\ + T) + \ \alpha.Q\ :\Rightarrow^* S + \alpha.Q + T$$

using Observant as follows: first, using the attraction rules, we move the excess next to the absorbent:

$$S + (\ \alpha.Q\ +\ \alpha.Q\ ) + T$$

and second, using the absorption rules, we eliminate the excess: the result follows. Notice how the whole sequence of term transformations has the good quality of leaving the syntactic structure of the solvent, P, unchanged.

Case ii) The induction hypothesis is $P' \overset{\alpha}{\Rightarrow} Q$ implies that $P' + \alpha.Q :\Rightarrow^* P''$, where P'' is as P' except that it has no annotations; we prove

$$P \xrightarrow{\tau} P' \overset{\alpha}{\Rightarrow} Q \quad \text{then } P + \alpha.Q :\Rightarrow^* P'''$$

where P''' is the result of deleting the annotations from P. By transition semantics, $P \xrightarrow{\tau} P'$ implies that P is of the form $S + \tau.P' + T$, and, in that case, we have to prove that

$$(S + \tau.P' + T) + \alpha.Q :\Rightarrow^* P'''$$

Since $P' \overset{\alpha}{\Rightarrow} Q$, the attraction rules will place $\alpha.Q$ close to P', the absorbent; then, saturation becomes applicable, leaving:

$$S + (\lfloor \tau.P'' \rfloor + (P' + \alpha.Q\ )) + T$$

Then, by hypothesis, the excess is absorbed and the circled terms disappear, and so we are left with:

$$S + (\lfloor \tau.P'' \rfloor + P'') + T$$

In that case, using the absorption rules, we eliminate the absorbent, yielding:

$$S + \tau.P'' + T$$

which is P''', just as required.

Case iii) Take $P' \overset{\tau}{\Rightarrow} Q$ then $P' + \tau.Q :\Rightarrow^* P'$ to be the induction hypothesis, and $P \xrightarrow{\alpha} \overset{\tau}{\Rightarrow} Q$ then $P + \alpha.Q :\Rightarrow^* P$ to be the induction conclusion. Proof omitted as it is similar to case ii). □

5 Results

Observant is both sound and complete with respect to an application of absorption; hence, it can be used to build a decision procedure for process normalisation. Furthermore, Observant directs proof search, hence, avoiding the combinatorial explosion. Observant requires only the knowledge of the excess and the absorbent to perform $P \overset{\alpha}{\Rightarrow} Q$; its behaviour does not depend on the syntax, but the associated transition semantics. So, as shown by [10], Observant can be incorporated to verification tools that can deal with problems outside the scope of state-space tools, such as the Edinburgh Concurrency Workbench (CWB) [2]. What is more, Observant never over-saturates; terms are only saturated if that means an absorbent is popped out so as to eliminate an excess.

6 Comparison and Conclusions

R_OBS is a complete normalisation procedure over divergence-free finite-state serial agents. Since it uses no absorbability measure, it over-saturates, hence, increasing the search space. Moreover, R_OBS, same as other class-rewriting systems, may be prohibitively expensive, if equality over canonical systems of recursive equations, though computable, is impractically expensive [4]. This makes R_OBS incompetent, when compared against state-space search based procedures. Observant therefore surpasses R_OBS.

Annotated TRSs can be applied to guide the search for a solution. Often, the search space of ATRSs is smaller than those associated with non-annotated TRSs. ATRSs give an account as to why, and how they achieve a task.

REFERENCES

[1] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill, 'Rippling: A heuristic for guiding inductive proofs', *Artificial Intelligence*, **62**, 185–253, (1993). Also available from Edinburgh as DAI Research Paper No. 567.

[2] R. Cleaveland, Parrow J., and B. Steffen, 'The concurrency workbench: A semantics-based verification tool for finite-state systems', in *Proceedings of the Workshop on Automated Verification Methods for Finite-State Systems*. Springer-Verlag, (1989).

[3] N. Dershowitz, 'Completion and its applications', in *Resolution of Equations in Algebraic Structures, Volume 2: Rewriting Techniques*, eds., H. Aït-Kaci and M. Nivat, pp. 31–86. Academic Press, (1989).

[4] N. Dershowitz and J. P. Jouannaud, 'Rewrite systems', in *Handbook of Theoretical Computer Science*, ed., J. van Leeuwen, volume B: Formal Methods and Semantics, pp. 243–320, Amsterdam, (1990). Elsevier.

[5] M. Hennessy and R. Milner, 'Algebraic Laws for Nondeterminism and Concurrency', *Journal of the Association for Computing Machinery*, **32**(1), 137–161, (1985).

[6] G. Huet and D. C. Oppen, 'Equations and rewrite rules: a survey', in *Formal languages: perspectives and open problems*, ed., R. Book. Academic Press, (1980). Presented at the conference on formal language theory, Santa Barbara, 1979. Available from SRI International as technical report CSL-111.

[7] P. Inverardi and M. Nesi, 'Deciding observational congruence of finite-state ccs expressions by rewriting', *Theoretical Computer Science*, **139**, 315–354, (1995).

[8] J.-P. Jouannaud and H. Kirchner, 'Completion of a set of rules modulo a set of equations', *SIAM J. Comput.*, **15**(4), 1155–194, (1986).

[9] R. Milner, *Communication and Concurrency*, Prentice Hall, London, 1989.

[10] R. Monroy, *Planning Proofs of Correctness of CCS Systems*, Ph.D. dissertation, Edinburgh University, 1998.

[11] T. Walsh, A. Nunes, and A. Bundy, 'The use of proof plans to sum series', in *11th Conference on Automated Deduction*, ed., D. Kapur, pp. 325–339. Springer Verlag, (1992). Lecture Notes in Computer Science Vol. 607. Also available from Edinburgh as DAI Research Paper No. 563.

[12] Tetsuya Yoshida, Alan Bundy, Ian Green, Toby Walsh, and David Basin, 'Coloured rippling: an extension of a theorem proving heuristic', in *In proceedings of ECAI-94*, ed., A. G. Cohn, pp. 85–9. John Wiley, (1994).