Some Experiments using term rewriting techniques for concurrency

Carron Kirkwood[a] and Kathy Norrie[b]

[a]Department of Computing Science, University of Glasgow
(Funded by SERC grant gr/f 35371/ 4/1/1477, Verification Techniques for LOTOS.)

[b]Department of Computing Science, Royal Holloway and Bedford New College, University of London
(Funded by SERC grant gr/f 35364/ 4/1/1477, Verification Techniques for LOTOS.)

Introduction

The purpose of this short paper is to report on preliminary investigations into the use of term rewriting theorem proving to support verification of formal description techniques for concurrent systems. These experiments are designed to explore contrasting areas in the use of term rewriting for concurrency.

In two distinct sets of experiments, we use the LP theorem prover[1] for equational and inductive proofs in CSP[2], and the RRL system[3], particularly Knuth-Bendix completion and equational proofs, to investigate one of the LOTOS[4] behavioural equivalences.

This work was completed as part of the SERC/IED ERIL project, which is investigating the use of Equational Reasoning for LOTOS. A complete presentation may be found in [5].

The examples presented are all small ones, but are already at the level where hand verification is tedious and error prone. They execute immediately in LP and RRL, and give hope that the same techniques can be used on much larger examples.

Experiments using CSP and LP

The experiments here involve using LP to verify the examples in [2]. Much of the reasoning in these examples is equational, making term rewriting an obvious method of theorem proving. We are also able to perform proofs by induction, since proving the base step and the induction hypothesis may only require equational reasoning. The basic rule set used consists of the CSP laws governing the process operators.

Consider a vending machine, *MACND*, which takes a coin then nondeterministically returns either chocolate or toffee, and three customers: *CUSTT* who always chooses toffee, *CUSTW* who nondeterministically chooses chocolate or toffee, and *CUSTA* who takes what he is given.

$$\begin{aligned}
&MACND = coin \rightarrow ((choc \rightarrow MACND) \sqcap (toffee \rightarrow MACND))\\
&CUSTT = coin \rightarrow toffee \rightarrow CUSTT\\
&CUSTW = coin \rightarrow ((choc \rightarrow CUSTW) \sqcap (toffee \rightarrow CUSTW))\\
&CUSTA = coin \rightarrow ((choc \rightarrow CUSTA) \mid (toffee \rightarrow CUSTA))
\end{aligned}$$

By putting the vending machine in parallel with each of these customers in turn, different behaviours are obtained. The recursive equations are unwound manually, and entered into LP in the form given below, since an infinite rewriting loop will appear if the equations are used in the form given above.

We show, using LP, that the behaviour of two of the processes may lead to deadlock (indicated by the possibility of nondeterministically choosing $STOP$):

$$\begin{aligned}
&\quad (coin \rightarrow ((choc \rightarrow MACND) \sqcap (toffee \rightarrow MACND))\\
&\;\| \; (coin \rightarrow toffee \rightarrow CUSTT)\\
&\equiv (coin \rightarrow (STOP \sqcap (toffee \rightarrow (MACND \parallel CUSTT))))
\end{aligned}$$

$$\begin{aligned}
&\quad (coin \rightarrow ((choc \rightarrow CUSTW) \sqcap (toffee \rightarrow CUSTW))\\
&\;\| \; (coin \rightarrow ((choc \rightarrow MACND) \sqcap (toffee \rightarrow MACND)))\\
&\equiv coin \rightarrow \quad ((choc \rightarrow (CUSTW \parallel MACND))\\
&\qquad\qquad\quad \sqcap (toffee \rightarrow (CUSTW \parallel MACND))\\
&\qquad\qquad\quad \sqcap STOP)
\end{aligned}$$

$$\begin{aligned}
&\quad (coin \rightarrow ((choc \rightarrow CUSTA) \mid (toffee \rightarrow CUSTA))\\
&\;\| \; (coin \rightarrow ((choc \rightarrow MACND) \sqcap (toffee \rightarrow MACND)))\\
&\equiv coin \rightarrow ((choc \rightarrow (CUSTA \parallel MACND)) \sqcap (toffee \rightarrow (CUSTA \parallel MACND))
\end{aligned}$$

The following example illustrates the method of proof by induction: The vending machine $MACND$ is modified to a machine of finite capacity k, in which the number of sweets remaining, n, is known at any stage. The customer can refill the machine at any time. Thus, the adapted machine, ND, is defined by the following mutually recursive processes:

$$\begin{aligned}
ND(0) \quad &= \quad refill \rightarrow ND(k)\\
ND(n + 1) \quad &= \quad if\ (n < k,\\
&\qquad\quad (coin \rightarrow ((choc \rightarrow ND(n)) \sqcap (toffee \rightarrow ND(n))) \mid refill \rightarrow ND(k)),\\
&\qquad\quad STOP)
\end{aligned}$$

The corresponding customer processes are defined analogously, and in order to reason about the resulting concurrent processes LP uses induction on the natural numbers.

Experiments using LOTOS and RRL

The aim of these experiments was to generate a *complete* set of rewrite rules from the axioms of weak bisimulation congruence shown in appendix B.2.2 of the LOTOS standard.

For a set of rewrite rules to be *complete* two properties are required: the equational system must be *confluent* and *terminating*. These properties ensure that every term has a unique normal form and give us a decision procedure for equality of terms.

The set of axioms used was built up incrementally, adding new axioms one by one, and rejecting those which cause the completion algorithm to fail or diverge. At present only a small subset of the axioms in appendix B.2.2 have been successfully run through the completion procedure. However, even this small set of rewrite rules can be useful.

Equational Proofs With the complete set of rules obtained, simple equational proofs may now be carried out which were not possible before completion. A more complex example showing that a two-way buffer is the same as two one-way buffers in parallel is given in detail in [5]. This example is a little false as recursion is not part of the restricted LOTOS used.

This example also highlights the need for laws relating some of the higher level constructs (such as parallelism) to the basic constructs of action sequencing and choice. These are known as the expansion laws. Any LOTOS process can be written using just action sequencing and choice. The actual expansion law required in this case cannot be entered straight into RRL because it involves generalised choice, which is not a part of the restricted LOTOS used, so two useful instantiations of this expansion law are derived, enabling the proof of weak bisimulation congruence to be carried out.

Axioms causing Infinite Sequences of Rewrite Rules Some axioms of appendix B.2.2 caused the completion procedure to diverge, generating an infinite sequence of rewrite rules. It would be useful if these rules could be generalised by producing a finite set of rules that subsume the infinite sequence of rules and are terminating and confluent.

Some attempts were made to generalise the particular sequences obtained. For example, the law distributing **hide** across $\gg$,

$$\textbf{hide}\ \mathrm{A}\ \textbf{in}\ (B_1 \gg B_2) \equiv (\textbf{hide}\ A\ \textbf{in}\ B_1) \gg (\textbf{hide}\ A\ \textbf{in}\ B_2)$$

produces two distinct infinite sequences of rewrite rules. The sequences generated by this rule can be eliminated by adding two simple new rules which are derived from a law in appendix B.2.2. This particular law was rejected from the rule set because it requires conditional rewriting, which is to be considered as an extension to this work.

Unfortunately most of the infinite sequences generated by our experiments are not so easy to eliminate. For example, the rules in the infinite sequence generated by the addition of the axiom for associativity of choice (A [] B) [] C == A [] (B [] C) follow this basic pattern: $A\,[]\,B \rightarrow B$ This rule is of course undesirable as it makes choice totally unfair (it will always choose the second argument). Also it is not an exact generalisation (i.e. not every instantiation is a rule) and so adding it would imply unsound equivalences. We suspect that A is a subterm of B (by inspection of the rules generated), but there is a more complex relationship between these two terms, which deserves further investigation.

Other Results It is expected that the weak bisimulation congruence laws in appendix B of the LOTOS standard will be sound. It was believed that this was not true and our experiments revealed an inconsistency.

When deciding which laws to use we entered: $A \gg \mathbf{stop} \equiv A \mid\mid\mid \mathbf{stop}$ When taken with other rules in the rule set, this rule causes the following contradiction: $\mathbf{i};\ \mathbf{stop} \equiv \mathbf{stop}$ Under weak bisimulation congruence this is false as the two equivalent processes are required to behave in the same way in all contexts. To test this condition it is necessary only to test how the processes behave when part of a choice statement.

Conclusions and Further Work

We conducted experiments in the application of term rewriting to the concurrency formalisms CSP and LOTOS. While the results we obtained were hopeful, we appreciate that there is still much work to be carried out in this area.

This preliminary investigation has been successful in that not only have we have shown that term rewriting is useful in reasoning about concurrency, we have also uncovered several areas for further development.

Acknowledgements

We should like to thank Steve Garland and John Guttag for letting us have LP, and Deepak Kapur for letting us have RRL. We would especially like to thank Ursula Martin, Dave Cohen, Muffy Thomas and Phil Watson, our colleagues.

References

[1] S.J. Garland, J.V. Guttag: *An Overview of LP, The Larch Prover*, Proc. Rewriting Techniques and Applications, 3rd Intl. Conference, LNCS 355 (1989), 137-151.

[2] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[3] D. Kapur and H. Zhang, RRL : Rewrite Rule Laboratory User's Manual, 1987, Revised May 1989.

[4] ISO 8807 : LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, 1988.

[5] C. Kirkwood and K. Norrie, Some Experiments using Term Rewriting Techniques for Concurrency, Technical Report — Royal Holloway and Bedford New College, 1990.