Application of Rewriting Techniques to Verification Problems

Adam Koprowski

Technical University of Eindhoven
Department of Computer Science
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
A.Koprowski@tue.nl

Abstract. The goal of the project is to employ techniques from term rewriting to verification problems. The relationship between liveness properties and termination of term rewrite systems (TRSs) is of particular interest. The emphasis is on the investigation of such properties for infinite state space systems where standard model checking techniques fail. Next to developing the necessary underlying theory and performing a case study analysis, the possibility to automate this approach is of great importance. In this paper we discuss the motivation of such work, present the results obtained so far, discuss related work and present plans for the further research.

1 Motivation

The problem of proving termination of term rewrite systems has been studied extensively and still attracts a lot of attention of term rewriting community. Although in general undecidable, for a broad spectrum of term rewrite systems encountered in practice termination can be proven and a number of techniques has been developed to serve this goal.[1]

Apart from the theoretical results, possibility to automate the process of proving termination of TRSs was always an important issue. A number of tools for proving termination of TRSs in a fully automated manner has been developed by different authors and an annual termination competition is being organized to stimulate further work and improvement in that area.[2]

Now our main motivation is to make use of the aforementioned results and develop a method to prove liveness properties by means of TRS termination. The important point is that particular infinite state space models can be encoded by means of finite TRSs. Now, since termination proofs do not depend in any way on exploration of a state space, such problems can be tackled by our approach, whereas standard model checking approach fails for them. We will present the motivating example to illustrate our approach in Section 3.4.

[1] For an overview of term rewriting in general, and termination of term rewriting in particular, reader is refereed to, for instance, [5].

[2] See http://www.lri.fr/~marche/termination-competition for more details.

2 Related work

The idea of transforming verification problems to problems of termination of TRSs goes back to Giesl and Zantema. In [1] they presented two such transformations. The first one is sound and complete, that is a termination of the transformed TRS is equivalent to a liveness question at hand. The second one is only sound but significantly simpler. In [2] a slightly different setting has been discussed.

As already remarked in [1] the sound and complete transformation presented there is by far too complicated to be of practical use – even for simple input systems termination of transformed TRSs is difficult to show. On the other hand the preliminary case study conducted by authors revealed that the sound transformation from [1] is often not strong enough and results in non-terminating TRSs for problems where liveness do hold. That was the motivation for seeking an alternative transformation; more powerful, but still suitable for automatic termination provers. We will present such a transformation in Section 3.2.

3 Preliminary results

After giving some preliminaries in Section 3.1 we present the transformation from liveness problems to (relative) termination problems of TRSs (3.2). Then in 3.3 we describe TPA – a tool developed by authors for solving such (relative) termination problems. In Section 3.4 we illustrate our approach with an example.

3.1 Preliminaries

For a signature Σ and a set of variables $\mathcal{V}$, we denote the set of terms over Σ and $\mathcal{V}$ by $\mathcal{T}(\Sigma, \mathcal{V})$. We denote the set of variables occurring in a term t by $\mathsf{Var}(t)$. A *rewrite rule* is a pair (ℓ, r), written $\ell \to r$, with $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V})$, $\ell \notin \mathcal{V}$, $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$. A *term rewriting system* (TRS) is a set of rewrite rules. The *rewrite relation* $\to_R$ for a TRS R is defined by $s \to_R t$ if there exists a rewrite rule $\ell \to r \in R$, a substitution δ and a context C such that $s = C[\ell\delta]$ and $t = C[r\delta]$. A TRS R is called terminating ($\mathrm{SN}(R)$[3]) if there is no infinite reduction $t_1 \to_R t_2 \to_R \ldots$.

For two relations R, S we define $R/S \equiv S^* \cdot R \cdot S^*$. We call an infinite $R \cup S$ reduction *fair* with respect to R if it contains infinitely many R-steps. The *relative termination* problem is to decide given two TRSs R, S whether $\mathrm{SN}(R/S)$. Note that $\mathrm{SN}(R/S)$ is equivalent to lack of infinite $\to_R \cup \to_S$ reductions fair with respect to $\to_R$.

Let top be a fresh unary symbol in Σ ($\mathsf{top} \notin \Sigma$). A term $t \in \mathcal{T}(\Sigma \cup \{\mathsf{top}\}, \mathcal{V})$ is called a *top term* if it contains exactly one instance of the top symbol, at the root of the term. We denote the set of top terms by $\mathcal{T}_{top}(\Sigma, \mathcal{V})$. A TRS over $\Sigma \cup \{\mathsf{top}\}$ is called a *top term rewrite system* (top TRS) if for all its rules $\ell \to r$ either both ℓ and r are top terms (*top rule*) or both ℓ and r do not contain an instance of the top symbol (*non-top rule*).

[3] It is usual to write $\mathrm{SN}(R)$ instead of $\mathrm{SN}(\to_R)$.

3.2 Transformation from liveness problems to termination problems

We make a concise presentation of the underlying theory and then present the transformation. For more elaborate description we refer the reader to [3] and [4].

We extend the notion of liveness as considered in [1] by introducing *fairness*. We define liveness with respect to a set of states S, two relations modelling computations $\to, \xrightarrow{=} \subseteq S \times S$, a set of initial states $I \subseteq S$ and a set of good states $G \subseteq S$ denoted as $\mathrm{Live}(I, \to, \xrightarrow{=}, G)$ to hold iff:

$$\forall t_1, t_2, \ldots : \left\{ \begin{array}{r} t_1 \in I \\ \forall i : t_i \to t_{i+1} \vee t_i \xrightarrow{=} t_{i+1} \\ \forall i \exists j > i : t_j \to t_{j+1} \end{array} \right\} \implies \exists i : t_i \in G$$

Now we represent the computation states by terms, so S becomes $\mathcal{T}(\Sigma, \mathcal{V})$ and $I, G \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. Abstract reduction relations $\to$ and $\xrightarrow{=}$ now correspond to rewrite relations of two TRSs over the same signature Σ: R and $R^=$, respectively. As a shorthand for $\to_R$ we write $\to$ and for $\to_{R^=}$ we simply write $\xrightarrow{=}$. Just like it is usual to write $\mathrm{SN}(R)$ rather than $\mathrm{SN}(\to_R)$, we will write $\mathrm{Live}(I, R, R^=, G)$ rather than $\mathrm{Live}(I, \to_R, \to_{R^=}, G)$.

Given some set of terms P we are going to restrict to the set of good states being terms not containing an instance of some term from P (we will denote this set by $G(P)$). Now we are going to investigate liveness properties of the form: $\mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^=, G(P))$ for some top TRSs R and $R^=$. This is equivalent to proving that every infinite fair reduction of top terms contains a term which does not contain an instance of any of the terms from P. Now we will present a transformation that relates this problem with the (relative) termination of transformed systems.

Definition 1 (LT) *Let R and $R^=$ be top TRSs over $\Sigma \cup \{\mathsf{top}\}$ and $P \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. The transformed systems $\mathrm{LT}(R)$ and $\mathrm{LT}^=(R^=, P)$ over $\Sigma \cup \{\mathsf{top}, \mathsf{ok}, \mathsf{check}\}$ are defined as follows:*

$$\boxed{\mathrm{LT}(R)}$$

$$\begin{array}{rll} \ell \to r & & \textit{for all non-top rules } \ell \to r \textit{ in } R \\ \mathsf{top}(\mathsf{ok}(\ell)) \to \mathsf{top}(\mathsf{check}(r)) & & \textit{for all top rules } \mathsf{top}(\ell) \to \mathsf{top}(r) \textit{ in } R \end{array}$$

$$\boxed{\mathrm{LT}^=(R^=, P)}$$

$$\begin{array}{rll} \ell \to r & & \textit{for all non-top rules } \ell \to r \textit{ in } R^= \\ \mathsf{top}(\mathsf{ok}(\ell)) \to \mathsf{top}(\mathsf{check}(r)) & & \textit{for all top rules } \mathsf{top}(\ell) \to \mathsf{top}(r) \textit{ in } R^= \\ \mathsf{check}(p) \to \mathsf{ok}(p) & & \textit{for all } p \in P \\ \mathsf{check}(f(x_1, \ldots, x_n)) \to f(x_1, \ldots, \mathsf{check}(x_i), \ldots, x_n) & & \\ & & \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1, 1 \leq i \leq n \\ f(x_1, \ldots, \mathsf{ok}(x_i), \ldots, x_n) \to \mathsf{ok}(f(x_1, \ldots, x_n)) & & \\ & & \textit{for all } f \in \Sigma \textit{ of arity } n \geq 1, 1 \leq i \leq n \end{array}$$

The following theorem from [3] relates relative termination of transformed systems with the liveness problem they originated from.

Theorem 2 (Soundness) *Let $R, R^{=}$ be top TRSs over $\Sigma \cup \{\mathsf{top}\}$, let $P \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. Then:*

$$\mathrm{SN}(\mathrm{LT}(R)/\,\mathrm{LT}^{=}(R^{=}, P)) \implies \mathrm{Live}(\mathcal{T}_{top}(\Sigma, \mathcal{V}), R, R^{=}, G(P))$$

It is worth noting that under some mild additional restrictions our transformation is also complete. For details we again refer to [3].

3.3 Proving (relative) termination automatically

In the preceding section we saw how to transform liveness problems to (relative) termination problems. To deal with such problems the first author developed a tool, TPA (Termination Proved Automatically), that aims at solving such problems in an automated way. It is the first tool that also supports relative termination of TRSs, which was one of the main motivations to develop it. It uses a number of termination proving techniques, most notably semantic labelling with natural numbers, which, for the time being, is used by no other tool. It got 3rd place in the aforementioned termination competition in 2005. More information about TPA can be found on its web-page, `http://www.win.tue.nl/tpa`.

3.4 Example

Example 1 (Cars over a bridge). There is a road with cars going in two directions. But on their way there is a bridge which is only wide enough to permit a single lane of traffic. So there are lights indicating which side of the bridge is allowed to cross it. We want to verify the following liveness property: every car will eventually cross the bridge. For that clearly we need some assumptions about the lighting system. We want to be as general as possible so instead of assuming some particular algorithm of switching lights we just require them to change in a fair way, that is in the infinite observation of the system there must be infinitely many light switches. Also we assume that before a light switches at least one car will pass (otherwise liveness is lost as lights can change all the time without any cars passing).

This system can be modelled with a unary *top* symbol whose arguments start with a binary symbol *left* or *right* indicating which side has a green light. The arguments of *left* and *right* start with unary symbols *new* and *old* representing cars waiting to cross the bridge. The constant *bot* stands for the end of the queue. New cars are allowed to arrive at the end of the queue at any time. What we want to prove is that finally no old car remains. The top TRS modelling this system follows:

$$
\begin{array}{llrcl}
(1) & \mathsf{top}(\mathsf{left}(\mathsf{old}(x), y)) & \to & \mathsf{top}(\mathsf{right}(x, y)) \\
(2) & \mathsf{top}(\mathsf{left}(\mathsf{new}(x), y)) & \to & \mathsf{top}(\mathsf{right}(x, y)) \\
(3) & \mathsf{top}(\mathsf{right}(x, \mathsf{old}(y))) & \to & \mathsf{top}(\mathsf{left}(x, y)) \\
(4) & \mathsf{top}(\mathsf{right}(x, \mathsf{new}(y))) & \to & \mathsf{top}(\mathsf{left}(x, y)) \\
(5) & \mathsf{top}(\mathsf{left}(\mathsf{bot}, y)) & \to & \mathsf{top}(\mathsf{right}(\mathsf{bot}, y))
\end{array}
\qquad
\begin{array}{llrcl}
(6) & \mathsf{top}(\mathsf{right}(x, \mathsf{bot})) & \to & \mathsf{top}(\mathsf{left}(x, \mathsf{bot})) \\
(7) & \mathsf{top}(\mathsf{left}(\mathsf{old}(x), y)) & \xrightarrow{=} & \mathsf{top}(\mathsf{left}(x, y)) \\
(8) & \mathsf{top}(\mathsf{left}(\mathsf{new}(x), y)) & \xrightarrow{=} & \mathsf{top}(\mathsf{left}(x, y)) \\
(9) & \mathsf{top}(\mathsf{right}(x, \mathsf{old}(y))) & \xrightarrow{=} & \mathsf{top}(\mathsf{right}(x, y)) \\
(10) & \mathsf{top}(\mathsf{right}(x, \mathsf{new}(y))) & \xrightarrow{=} & \mathsf{top}(\mathsf{right}(x, y)) \\
(11) & \mathsf{bot} & \xrightarrow{=} & \mathsf{new}(\mathsf{bot})
\end{array}
$$

We have the following semantics of the rules: $(1) - (4)$ car passes and the light changes; $(5) - (6)$: no car waiting, light can change; $(7) - (10)$ car passes, light remains the same; (11) New car arriving.

By using our approach, in a way described in the preceding sections, the liveness property stating that every old car can eventually cross the bridge, can be transformed to a question of relative termination of TRS. This question, in turn, can be positive answered in a fully automated way by the use of TPA .

4 Conclusions and further research

We presented a framework for verification of liveness properties, that can work also for infinite state space systems. This method requires the model of the system to be given as TRS but then the proof that liveness property holds is delivered automatically by TPA by first transforming the TRS and then proving termination of the transformed TRS.

Clearly this is just the beginning of the journey and a lot of extensions is possible, among which the following ones we find particularly interesting and worth further investigation:

- Clearly our definition of fairness and of liveness problems we are aiming at could enjoy some generalization. Of particular interest would be the direction allowing us to deal not only with liveness but also with safety properties.
- Once the framework is mature enough it would be interesting to perform a case study analysis, on real-life examples.
- Some techniques for proving termination generalize to relative termination and as such they are used in TPA. But, since relative termination plays an important role in dealing with liveness with fairness, we believe that further development of relative termination techniques would be of great interest.
- As soon as such techniques are available implementing them in TPA would be a natural next step, increasing the applicability of the tool.

References

1. Jürgen Giesl and Hans Zantema. Liveness in rewriting. In *Proc. 14th RTA, LNCS 2706*, pages 321–336, 2003.
2. Jürgen Giesl and Hans Zantema. Simulating liveness by reduction strategies. *Electr. Notes Theor. Comput. Sci.*, 86(4), 2003.
3. Adam Koprowski and Hans Zantema. Proving liveness with fairness using rewriting. In *Frontiers of Combining Systems, 5th International Workshop, FroCoS 2005, Vienna, Austria, September 19-21, 2005, Proceedings*, volume 3717 of *Lecture Notes in Computer Science*, pages 232–247. Springer, 2005.
4. Adam Koprowski and Hans Zantema. Proving liveness with fairness using rewriting. Technical Report CSR 05-06, Eindhoven University of Technology, Eindhoven, The Netherlands, March 2005.
5. TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.