Symbolic Model Checking for LTL using SNF

Daniel Sheridan
Laboratory for the Foundations of Computer Science,
Division of Informatics,
University of Edinburgh. EH9 3JZ
dan.sheridan@contact.org.uk

Introduction

In Frisch et al. (2002) we proposed an adaptation of the Separated Normal Form (Fisher, 1991) to make it more suited to use as an encoding procedure for Bounded Model Checking (Biere et al., 1999). We observe that the effect of this modification — extending the fixpoint characterisation to the least fixpoint operators — is make the specification appear very much like an automata.

Here we take this work one step further by demonstrating a simple way of using this extension to SNF to create an automata that can be used by a symbolic model checker to verify LTL properties.

Model Checking

A model checking problem is a pair $\langle M, f\rangle$ of a model and a temporal logic specification.

A model M is defined as a Kripke structure $\langle S, R, L, I\rangle$ where S is a set of states; $R : S \to S$ is the transition relation; $L : S \to \mathcal{P}(AP)$ is the labelling function, marking each state with the set of atomic propositions (AP) that hold in that state; and I is the set of initial states, which may be equal to S. A path $\pi \in M$ is a sequence of states $s_0, s_1, \ldots \in M$ such that $\forall i.(s_i, s_{i+1}) \in R$. We write $\pi(i)$ to refer to the ith state in the path, and say that a path π is a *k-loop* if the $\pi(k) = \pi(l)$, for some $l, 0 \le l < k$.

The *model checking problem for LTL* is to verify that for an LTL formula f, for all paths $\pi_i \in M$ such that $\pi_i(0) \in I$, $(M, \pi_i) \models f$.

The Separated Normal Form

Fisher (1991) defined a normal form for temporal logic based on the Separation Theorem (Gabbay, 1989) and gave a series of transformations for reaching it. Our previous work on adapting SNF for bounded LTL (Frisch et al., 2002) and bounded model checking is derived from the work by Bolotov and Fisher (Bolotov and Fisher, 1997) on SNF for CTL.

The general form of SNF is $\mathbf{G}\left(\bigwedge_i (P_i \Rightarrow F_i)\right)$ where P_i are (strict) past time formulæ and F_i are (non-strict) future time formulæ. Since LTL (and CTL) are typically used without explicit past-time operators, the $\mathbf{start}$ operator is included which holds only at the beginning of time.

The rules $P_i \Rightarrow F_i$ are of the following form:

$$\mathbf{start} \Rightarrow \bigvee_j l_j \qquad \text{An } \textit{initial} \text{ rule}$$

$$\bigwedge_i l_i \Rightarrow \mathbf{X} \bigvee_j l_j \qquad \text{A } \textit{global } \mathbf{X}\text{-rule}$$

$$\bigwedge_i l_i \Rightarrow \mathbf{F} \bigvee_j l_j \qquad \text{A } \textit{global } \mathbf{F}\text{-rule}$$

where l_i and l_j are literals. Transformations based on the fixpoint characterisations of LTL operators are used to convert a temporal logic formula to SNF. These transformations are based on replacing the greatest fixpoint operators ($\mathbf{G}, \mathbf{R}$) with a recursive defintion and an existentially quantified variable. The least fixpoint operators cannot be expanded in the same way in an infinite-future context, so are simply reduced to the $\mathbf{F}$ operator.

SNF and Bounded LTL

In the case of bounded model checking, the paths, π under consideration adopt the specific form $\pi = ab^\omega$. This construction allows infinite future properties to be determined by considering only $|ab|$ states. We may also regard the path as a finite one: ab, and we can show that by projecting a $\mathbf{F}$-formula evaluated in b to the start of b, it can be evaluated in the finite context without loss of generality.

The result of this is that in Frisch et al. (2002) we construct an SNF-based normal form (the 'fixpoint form') which eliminates all operators except for $\mathbf{X}$. We enforce the finite path requirement with a constraint on the introduced variable at the end of the first occurrence of b: a rule of the form $\mathbf{bound} \Rightarrow \bigvee_j l_j$.

Generating the Automata

The mapping from the fixpoint form to an automata is for the most part very simple. For a given type of rule, we can write down the corresponding property of the automata:

- Propositional rules $\bigwedge_i l_i \Rightarrow \bigvee_j l_j$ become invariants $\forall s \in S, (\exists l \in l_i \cdot l \notin L(s)) \vee (\exists l \in l_j \cdot l \in L(s))$

- Initial rules $\mathbf{start} \Rightarrow \bigvee_j l_j$ become constraints on the set of initial states: $I \subseteq \{s | s \in S \wedge \exists l \in l_j \cdot l \in L(s)\}$
- Global $\mathbf{X}$ rules $\bigwedge_i l_i \Rightarrow \mathbf{X} \bigvee_j l_j$ become constraints over transitions: $R \subseteq \{(s_0, s_1) | s_0, s_1 \in S \wedge (\exists l \in l_i \cdot l \notin L(s_0)) \vee (\exists l \in l_j \cdot l \in L(s_1))\}$

This leaves one rule and one special constraint unhandled: the items which make the fixpoint characterisation of $\mathbf{F}$ possible in the bounded case. Since we are working in the general symbolic model checking case, we cannot appeal to features of a restricted path semantics. Instead we observe that the effect of the constraints in the bounded case is to restrict the evaluation of the $\mathbf{F}$ operator to one occurrence of each state in b; it is convenient in the bounded case to use the first occurrence of b as we have access to the position in the path of its start and end.

For the unbounded case we still need to reduce the number of evaluations, but it is sufficient to ensure that it is finite. Symbolic model checking provides us with directly with a solution in the form of fairness constraints. It is sufficient to encode a final rule of the form $\mathbf{bound} \Rightarrow \bigvee_j l_j$ simply as a restriction to fair paths in which $\bigvee_j l_j$ holds.

In the case of the transformations given in Frisch et al. (2002) we find that the final rules, and hence the fairness constraints, each contain exactly one variable, which is not contained in the original model.

Space limitations do not allow us to include the proof of this approach, but we draw the reader's attention to the pleasing similarity between the expression of fairness and the general form of SNF: a fairness constraint is written as the CTL $\mathbf{AG}\,\mathbf{AF}\, f$ while the LTL expression we are considering becomes, during transformation to SNF, $\mathbf{G}((p \Rightarrow \mathbf{F}\, f) \wedge \ldots)$.

Past LTL

Including past time operators in LTL does not increase the expressivity, but it is gaining some acceptance in model checking as it can be a more natural way of writing specifications. Fisher (1991) considered a logic including non-strict past time operators. The recent work of Benedetti and Cimatti (2003) deals with handling *strict* past-time operators; it turns out to be simple to adapt SNF and the fixpoint form to handle this type of past (although the issues discuseed in Benedetti and Cimatti (2003) complicate its use in BMC).

In fact, by using the past time transformations together with the past to future transformations given below we can extend the system to include past LTL without needing to change the conversion given in the previous section.

$$\mathbf{Z}\, p \Rightarrow F \longrightarrow \begin{cases} \mathbf{start} \Rightarrow F \\ \mathbf{Y}\, p \Rightarrow F \end{cases}$$

$$\mathbf{Y}\, p \Rightarrow F \longrightarrow p \Rightarrow \mathbf{X}\, F$$

Further Work and Conclusions

Having an implementation of LTL symbolic model checking is vital for a proper comparison between symbolic and bounded model checking procedures. We hope that by consructing an LTL symbolic model checking system on the same foundations as the best bounded model checking encoding we will be able to make a more meaningful comparison between the two. In addition, this method provides a simple and convenient way of dealing with LTL that, in a combined model checking system such as NuSMV or Vis, can be coded to be common to both methods of model checking.

Preliminary experimental results indicate that model checking in this way performs at least as well as the current LTL implementation in NuSMV (based on Grumberg et al. (1997)) while the time taken to construct the automata is greatly reduced.

References

Marco Benedetti and Alessandro Cimatti. Bounded model checking for past LTL. In *Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, TACAS'03*, Lecture Notes in Computer Science, Warsaw, Poland, April 2003. Springer.

Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In W.R. Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems. 5th International Conference, TACAS'99*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer-Verlag Inc., July 1999.

Alexander Bolotov and Michael Fisher. A resolution method for CTL branching-time temporal logic. In *Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning (TIME)*. IEEE Press, 1997.

Michael Fisher. A resolution method for temporal logic. In *Proceedings of Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, August 1991.

Alan Frisch, Daniel Sheridan, and Toby Walsh. A fixpoint based encoding for bounded model checking. In M D Aagaard and J W O'Leary, editors, *Formal Methods in Computer-Aided Design; 4th International Conference, FMCAD 2002*, volume 2517 of *Lecture Notes in Computer Science*, pages 238–254, Portland, OR, USA, November 2002. Springer.

Dov Gabbay. The declarative past and imperative future. In H. Barringer, editor, *Proccedings of the Colloquium on Temporal Logic and Specifications*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer-Verlag, 1989.