Constraint-Based Abstract Semantics for Temporal Logic: A Direct Approach to Design and Implementation (extended abstract) ⋆ ⋆⋆

Gourinath Banda[1] and John P. Gallagher[1,2]

[1] Roskilde University, Denmark
[2] IMDEA Software, Madrid
Email: {gnbanda,jpg}@ruc.dk

Introduction

In this work we apply the framework of abstract interpretation [5] to design and implement an abstraction of temporal logic. Then we apply this approach to verify temporal properties of real-time system models (expressed as CLP programs), using a domain based on linear constraints, and an implementation in CLP interfaced to the PPL library and an SMT solver.

Some previous work in temporal abstraction handles only universal formulas (e.g. [4]) or use extra conceptual apparatus in the semantics such as mixed or modal transition systems (e.g. [6, 9]) in order to approximate the meaning of arbitrary formulas. By contrast our approach is able to approximate safely the meaning of any formulas, with no restriction on the occurrence of existential and universal quantifiers.

The present work is part of an attempt to develop a uniform constraint-based formal modelling and verification framework for verifying infinite state reactive systems. The modelling part of this framework was considered in [3, 2] where it was shown how to model linear hybrid automata (LHA) specifications as constraint logic programs.

Temporal logics and finite model checking

The use of temporal languages such as CTL, LTL and the μ-calculus for expressing properties of reactive, real-time systems has become widespread. Their semantics is based on Kripke structures, which are transition systems with a property-labelling function returning the set of propositions that are true in each state.

Given a Kripke structure K modelling some system of interest with states S, functions $pre : 2^S \to 2^S$, $\widetilde{pre} : 2^S \to 2^S$ and $\mathsf{states} : \mathcal{P} \to 2^S$ express essential properties of the structure.

⋆ Work partly supported by the Danish Natural Science Research Council project *SAFT: Static Analysis Using Finite Tree Automata*.

⋆⋆ The full version of this work will appear in the Proceedings of the 16th LPAR conference, Dakar, Senegal, May 2010.

- $pre(S') = \{s \mid \exists s^{'} \in S^{'} : (s, s^{'}) \in \Delta\}$ returns the set of states having at least one of their successors in the set $S' \subseteq S$;
- $\widetilde{pre}(S') = \mathsf{compl}(pre(\mathsf{compl}(S')))$ returns the set of states all of whose successors are in the set $S' \subseteq S$; the function $\mathsf{compl}(X) = S \setminus X$.
- $\mathsf{states}(p) = \{s \in S \mid p \in L(s)\}$ returns the set of states where $p \in \mathcal{P}$ holds.

Using these functions, a function $[\![.]\!]$ is defined that evaluates the set of states where a temporal logic formula holds. The expression $[\![\phi]\!]$ returns the set of states at which formula ϕ holds. Model checking consists of checking whether the Kripke structure K possesses a property ϕ, written $K \models \phi$. This is defined to be true iff $I \subseteq [\![\phi]\!]$, where I is the set of initial states, or equivalently, that $I \cap [\![\neg\phi]\!] = \emptyset$.

When the state-space S is infinite, the evaluation of $[\![.]\!]$ might not terminate and hence the model checking of infinite state systems becomes undecidable. In this case we try to approximate $[\![.]\!]$ using the theory of *abstract interpretation*.

Abstract interpretation and infinite state model checking

In abstract interpretation we develop an abstract semantic function systematically from the standard ("concrete") semantics. The formal framework is based on a Galois connection $\langle L, \sqsubseteq_L\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq_M\rangle$ between the lattices $\langle L, \sqsubseteq_L\rangle$ and $\langle M, \sqsubseteq_M\rangle$ which are the concrete and abstract semantic domains respectively. The functions α and γ are known as the abstraction and concretisation functions respectively.

For our purposes we only consider abstractions based on Galois connections $\langle 2^S, \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle 2^A, \subseteq\rangle$, where the concrete domain 2^S consists of sets of concrete states and the abstract domain 2^A consists of sets of abstract states. (In fact the abstract domain could be any lattice but the presentation become somewhat more complex).

Given such a Galois connection, the concrete functions pre, $\widetilde{pre}$, and states defined earlier are abstracted by their best possible counterparts over the abstract domain, yielding the following abstract functions.

$$apre = \alpha \circ pre \circ \gamma \qquad \widetilde{apre} = \alpha \circ \widetilde{pre} \circ \gamma \qquad \mathsf{astates} = \alpha \circ \mathsf{states}$$

We simply substitute $apre$, $\widetilde{apre}$ and $\mathsf{astates}$ for their concrete counterparts in the concrete semantic function to obtain abstract semantics $[\![.]\!]^a$ for the temporal logic. The properties of Galois connections ensure that $\alpha([\![\phi]\!]) \subseteq [\![\phi]\!]^a$ and $\gamma([\![\phi]\!]^a) \supseteq [\![\phi]\!]$.

The main result

As a consequence, for any Kripke structure K and ϕ a temporal formula, if $\gamma([\![\neg\phi]\!]^a) \cap I = \emptyset$ then $K \models \phi$.

This result provides us with a sound abstract model checking procedure for any temporal logic formula ϕ. Namely, given a formula ϕ to check, we evaluate the abstract semantics of its negation, $[\![\neg\phi]\!]^a$. If this represents a set of states that is disjoint from the initial states, then the property is satisfied. Of course, if $\gamma([\![\neg\phi]\!]^a) \cap I \supset \emptyset$ nothing can be concluded.

Experiments with a constraint-based abstraction

We consider transition systems derived from Linear Hybrid Automata (LHAs) whose states are n-tuples of real numbers, and abstractions of the state space based on a finite partition, which are also represented using linear constraints. In the paper it is shown how constraint representation of all the semantic functions $pre, \widetilde{pre}$ and states and the abstraction functions α and γ are constructed. Hence the abstract functions $apre, \widetilde{apre}$ and $\mathsf{astates}$ can also be constructed by composing the basic functions. A CLP implementation, interfaced to powerful tools, namely the Parma Polyhedra Library (PPL) [1] and the SMT solver Yices [7], was constructed, yielding an effective abstract model checker. This was applied successfully to a number of LHAs models, proving safety, liveness and progress properties.

A critical area of future research is the selection of abstractions (such as a partition of the state space) and strategies for refining the abstraction if it turns out to be too coarse to prove the required property. Constructing the abstract semantics completely using the standard techniques of abstract interpretation provides a general framework for refinement, for example building on the techniques of Ganty [8].

References

1. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
2. G. Banda. *Modelling and Analysis of Real Time Systems with Logic Programming and Constraints*. PhD thesis, Roskilde University, 2010.
3. G. Banda and J. P. Gallagher. Analysis of Linear Hybrid Systems in CLP. In M. Hanus, editor, *LOPSTR 2008*, volume 5438 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 2009.
4. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.
5. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles*, pages 238–252, 1977.
6. D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.
7. B. Dutertre and L. M. de Moura. A fast linear-arithmetic solver for DPLL(T). In T. Ball and R. B. Jones, editors, *CAV 2006*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2006.
8. P. Ganty. *The Fixpoint Checking Problem: An Abstraction Refinement Perspective*. PhD thesis, Université Libre de Bruxelles, Département d'Informatique, 2007.
9. P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In K. G. Larsen and M. Nielsen, editors, *CONCUR 2001*, volume 2154 of *Lecture Notes in Computer Science*, pages 426–440. Springer, 2001.