Brief Announcement: Branching Time Refinement

Panagiotis Manolios
College of Computing, CERCS Lab, Georgia Institute of Technology
801 Atlantic Drive, Atlanta, Georgia, 30332, USA
http://www.cc.gatech.edu/~ manolios
manolios@cc.gatech.edu

I develop notions of refinement for branching time based on stuttering simulation and bisimulation and I show that if one system refines another, then a refinement map always exists, without relying on any of the conditions present in the approach taken by Abadi and Lamport, *e.g.*, machine closure, finite invisible nondeterminism, internally continuity, the use of history and prophecy variables, etc. The added generality is obtained by using relations (instead of functions) to relate implementation states with specification states and by requiring the use of well-founded relations.

There are two reasons why I chose to work in the branching-time framework. Firstly, I am primarily interested in mechanical verification where algorithmic issues are paramount: simulation and bisimulation can be decided in polynomial time, whereas trace containment and trace equivalence are PSPACE-complete problems. Secondly, when one proves refinement theorems about infinite-state systems, one uses inductive arguments which depend on the structure of the systems in question. The branching-time notions are structural and local. The linear-time notions are not; as a result, the proof rules are more complicated.

Refinement is used to reason about distributed systems as follows: we specify a sequence of related systems, starting with an abstract system, the specification, and ending with a concrete system, the implementation. We then prove that every pair of adjacent systems is related, via a suitable, compositional notion of refinement.

A *transition system* (TS) $\mathcal{M}$ is a tuple $\langle S, \dashrightarrow, L\rangle$, where S is the set of states of $\mathcal{M}$, $\dashrightarrow \subseteq S \times S$ is the *transition relation* of $\mathcal{M}$, and L is the *labeling function*: its domain is S and it tells us what is observable at a state. A relation on $B \subseteq S \times S$ is a stuttering simulation, if for any s, w such that sBw, s and w are identically labeled and any infinite path starting at s can be matched by some infinite path starting at w. By "matched" I mean that it is possible to partition the infinite paths into finite, non-empty segments such that states in related segments are related by B. I now show how to relate transition systems by defining simulation refinement. Let $\mathcal{M} = \langle S, \dashrightarrow, L\rangle$, $\mathcal{M}' = \langle S', \dashrightarrow', L'\rangle$, and $r : S \rightarrow S'$. We say that $\mathcal{M}$, the implementation, is a simulation refinement of $\mathcal{M}'$, the specification, with respect to refinement map r if there exists a relation, B, such that $\langle \forall s \in S :: sB(r.s)\rangle$ and B is a stuttering simulation on the $\langle S \uplus S', \dashrightarrow \uplus \dashrightarrow', \mathcal{L}\rangle$, where $\uplus$ denotes disjoint union and $\mathcal{L}.s = L'(r.s)$ for s an S state and $\mathcal{L}.s = L'(s)$ otherwise.

That $\mathcal{M}$ is a simulation refinement of $\mathcal{M}'$ with respect to r implies that every visible behavior of $\mathcal{M}$ (where what is visible depends on r) is a behavior of $\mathcal{M}'$. In order to check this, we must show that infinite sequences match, but this requires a *global* analysis and we would much rather reason *locally*, about states and their successors. To this end, I introduce a local proof rule called *well-founded simulation*. $B \subseteq S \times S$ is a well-founded simulation on $\mathcal{M} = \langle S, \dashrightarrow, L\rangle$ iff: (1) $\langle \forall s, w \in S :: sBw \;\Rightarrow\; L(s) = L(w)\rangle$ and (2) There exist functions, $rankt : S \times S \rightarrow W$, $rankl : S \times S \times S \rightarrow \mathbb{N}$, s.t. $\lessdot$ is a well-founded relation on W and for all $s, u, w \in S$ such that sBw and $s \dashrightarrow u$ we have: $\langle \exists v :: w \dashrightarrow v \;\wedge\; uBv\rangle \vee (uBw \;\wedge\; rankt(u,w) \lessdot rankt(s,w)) \vee \langle \exists v :: w \dashrightarrow v \;\wedge\; sBv \;\wedge\; rankl(v,s,u) < rankl(w,s,u)\rangle$.

Theorem 1 *B is a well-founded simulation iff it is a stuttering simulation.*

Theorem 1 shows that we have a sound and complete proof rule. After developing some theory we can show the following compositionality result.

Theorem 2
If $\mathcal{M} \sqsubseteq_r \mathcal{M}'$ and $\mathcal{M}' \sqsubseteq_q \mathcal{M}''$ then $\mathcal{M} \sqsubseteq_{r;q} \mathcal{M}''$.

There are temporal logic implications, which I now outline. Informally, a typed TS is one where the labeling function applied to a state returns the values of a set of variables, each of a particular type, for the state. In the case of refinement maps that hide some of the implementation variables, we can show that the implementation satisfies every $\mathrm{ACTL}^* \setminus \mathsf{X}$ formula that the specification satisfies.

Theorem 3 *If $\mathcal{M} = \langle S, \dashrightarrow, L\rangle \sqsubseteq_r \mathcal{M}' = \langle S', \dashrightarrow', L'\rangle$, both $\mathcal{M}$ and $\mathcal{M}'$ are typed TSs, and $L'(r.s) = L.s|_V$, then for any pair of states $s, r.s$ such that $s \in S$, and any $\mathrm{ACTL}^* \setminus \mathsf{X}$ formula, f, built out of expressions that only depend on variables in V, we have $\mathcal{M}', r.s \models f \;\Rightarrow\; \mathcal{M}, s \models f$.*

I carry out a similar program with stuttering bisimulation. In addition, I have added mechanical support to the ACL2 theorem proving system for proving stuttering simulations and stuttering bisimulations and have conducted several verification case studies.

To appear in the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003) July 13-16, 2003, Boston, Massachusetts.