Assume-Guarantee Abstraction Refinement for Probabilistic Systems

Anvesh Komuravelli and Corina Păsăreanu

Carnegie Mellon

NSF Expeditions in Computing

The Problem

- A *Labeled Probabilistic Transition System* (LPTS) is a generalization of an MDP allowing non-determinism on actions.
- Verifying LPTSes composed of multiple components suffers from *state explosion*.
- Can we use Assume-Guarantee Compositional Reasoning?

$$\frac{L_1 \parallel A \preceq P \qquad L_2 \preceq A}{L_1 \parallel L_2 \preceq P}\ (\text{ASYM})$$

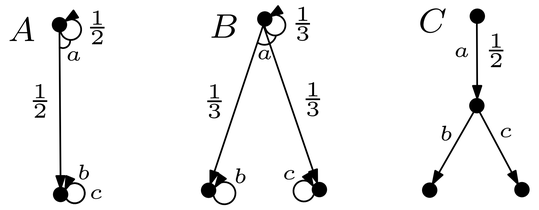
- Many choices for $\preceq$ and P.

Our Choices

- $\preceq$ is the *strong simulation conformance*. $L_1 \preceq L_2$ iff there is a relation $R \subseteq S_1 \times S_2$ such that
  1. $s_1 R s_2$ and $s_1 \xrightarrow{a} \mu_1 \Rightarrow$ there exists $s_2 \xrightarrow{a} \mu_2$ with $\mu_1 \sqsubseteq_R \mu_2$, and
  2. $s_1^0 R s_2^0$.
- P is another LPTS.
- How good are these choices?
  1. $\preceq$ relates specifications to implementations.
  2. sound and complete rule.
  3. can be generalized to multiple components and to also model checking logical properties.
- **No previous algorithm** or tool to generate counterexamples to $\preceq$.

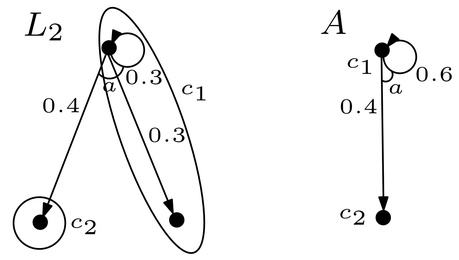
Counterexample to $L_1 \preceq L_2$

- *Sub-Stochastic Tree*, *i.e.* tree-shaped LPTS.
- *Sub-structure* of an unrolling of L_1.
- Traces, Markov Chains, MDPs are insufficient in general.
- Based on the greatest fixed point algorithm for $\preceq$.



Obtaining A automatically

- Maintain A as an *abstraction* of L_2, *i.e.* $L_2 \preceq A$, by partitioning S_2.



- Only need to check Premise 1 of ASYM.
- Use counterexamples to refine A (the partition of S_2).
- Need at most $|L_2|$ steps.

Refining A using C

- C is a sub-structure of an unrolling of A inducing an injection I between S_C and S_2.
- Find the coarsest strong simulation between C and L_2 contained in I using a *bottom-up* algorithm.
- Split abstract states when $R(s_a) = \emptyset$ for some $s_a \in S_A$ or when the initial states are not related.
- Always results in a finer partition.

Experiments

- PRISM's front-end for parsing models and Yices for checking $\preceq$.

Example (*param*)	$\|L\|$	$\|P\|$	ASYM		ASYM-N		MONO	
			Time	*Mem*	*Time*	*Mem*	*Time*	*Mem*
$CS_1(5)$	94	16	7.2	15.6	99.1	15.1	**0.2**	**8.8**
$CS_1(6)$	136	19	11.6	22.7	1*k*	21.4	**0.5**	**12.2**
$CS_1(7)$	186	22	37.7	49.4	*out*	–	**0.8**	**17.9**
$CS_N(2)$	34	15	0.7	7.1	4.9	6.8	**0.1**	**5.9**
$CS_N(3)$	184	54	43.0	63.0	5*k*	110.3	**14.8**	**37.9**
$CS_N(4)$	960	189	*out*	–	*out*	–	**1.8***k*	**667.5**
MER (3)	16*k*	12	**2.6**	19.7	4.3	**14.6**	193.8	458.5
MER (4)	120*k*	15	**15.0**	53.9	28.0	**37.3**	*out*	–
MER (5)	841*k*	18	–	*out*	**170.7**	**88.5**	–	*out*
SN (1)	462	18	**0.2**	**6.2**	3.0	8.5	1.5	27.7
SN (2)	7860	54	**79.5**	**112.9**	679.6	171.6	4.7*k*	1.3*k*
SN (3)	78*k*	162	*out*	–	**6.9***k*	**524.9**	–	*out*

AGAR Algorithm

1. $A \leftarrow$ coarsest abstraction of L_2
2. while $L_1 \parallel A \not\preceq P$
3. obtain a counterexample C
4. obtain projections $C \restriction_{L_1}$ and $C \restriction_A$
5. if $(C \restriction_{L_1})^{\alpha_1} \parallel L_2 \not\preceq P$
6. a counterexample C'
7. else
8. $(\_, A) \leftarrow \mathit{analyzeAndRefine}(C \restriction_A, A, L_2)$
9. return $L_1 \parallel L_2 \preceq P$ holds

Generalizing ASYM

- Generalized to reasoning about $N \geq 2$ components using recursive invocations of the rule for 2 components.

$$\frac{L_1 \parallel A_1 \preceq P \qquad \ldots \qquad L_n \preceq A_{n-1}}{\|_{i=1}^{n} L_i \preceq P}\ (\text{ASYM-N})$$

Need a counterexample which is a sub-structure of an unrolling of L_2 for the recursion - line 5 of the algorithm.

- Can be generalized to *Weak Safety* or richer logics.

$$\frac{L_1 \parallel A \models \phi \qquad L_2 \preceq A}{L_1 \parallel L_2 \models \phi}$$

Future Work

- Exploring *learning* techniques as an alternative.
- Assumptions with smaller alphabets using alphabet refinement and *weak simulation*.

References

1. S.J. Chaki. *A CEGAR Framework for Verifying Concurrent C Programs.* PhD Thesis, CMU, 2005.
2. Bobaru et al. *Automated Assume-Guarantee Reasoning by Abstraction Refinement.* CAV 2008.