Integrating Model-Checking and Theorem Proving for Automating the Generation of Abstractions

Shiva Nejati and Mehrdad Sabetzadeh

Department of Computer Science

University of Toronto

{shiva,mehrdad}@cs.toronto.edu

January 31, 2003

Introduction and Motivation

- Model-checkers are efficient tools for verifying finite-state systems, *however* ...
  - suffer from the state explosion problem;
  - cannot handle parameterized/unbounded systems.
- Theorem provers are very general, *however* ...
  - require detailed guidance;
  - working with theorem-provers requires a great deal of expertise.
- In principle, we would like to combine the two approaches:
  - Model-checkers handle decision procedures.
  - Theorem provers handle proofs in undecidable logics.

Introduction and Motivation (Cnt'd)

- We need to verify systems that
  - are very large and complicated;
  - have unbounded state-spaces.
- Combining model-checking and theorem proving facilitates
  - automating the process of constructing finite-state abstractions.
- **Example**:
  - A BDD-based model-checker has been integrated with the PVS theorem prover.

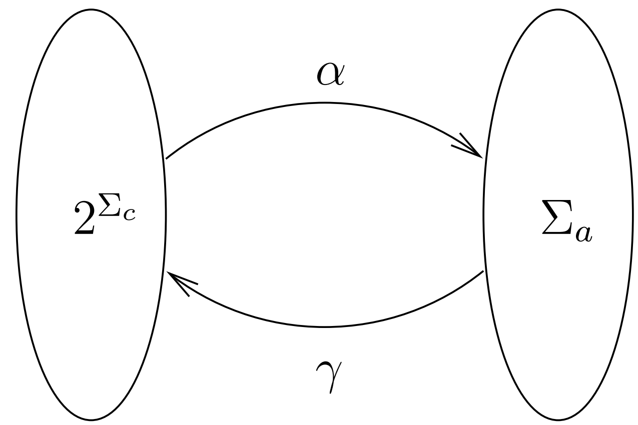
Outline

- What is Abstraction?
- Predicate Abstraction
- Abstraction in PVS
- 3-Valued (Mixed) Abstraction
- Optimizations to Under-Approximation Abstraction
- Conclusions

What is Abstraction?

- Reducing a large model to a smaller one while preserving the desired properties.
- To verify a concrete model C using abstraction:
  1. generate an abstract model A either *manually* or *automatically*;
  2. check the soundness of abstraction:
  $$\forall\varphi \in L \cdot A \models \varphi \Rightarrow C \models \varphi;$$
  3. check the properties of interest over A:
  $$A \overset{?}{\models} \varphi.$$

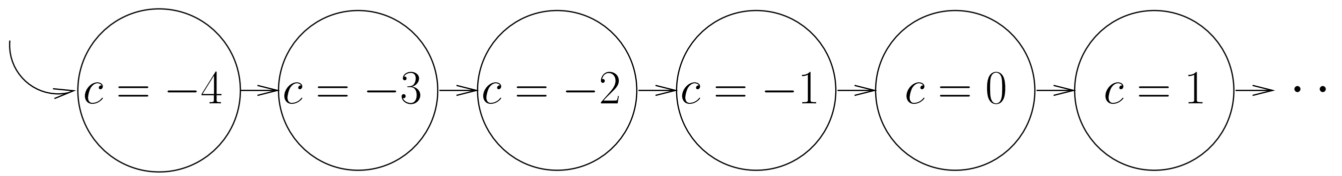
Computing Abstractions



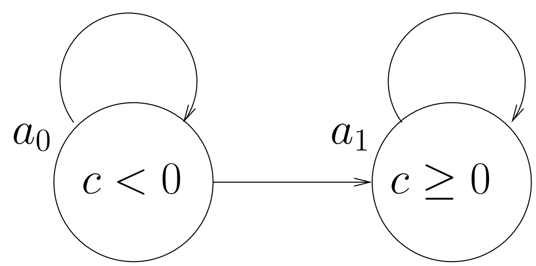
- Σ_c is the concrete state-space;
- Σ_a is the abstract state-space;
- $\alpha : 2^{\Sigma_c} \rightarrow \Sigma_a$ is an *abstraction* function;
- $\gamma : \Sigma_a \rightarrow 2^{\Sigma_c}$ is a *concretization* function;
- Properties of α and γ:
  - ▷ $\forall S \subseteq \Sigma_c .\, S \subseteq \gamma\big(\alpha(S)\big)$;
  - ▷ $\forall s \in \Sigma_a .\, \alpha\big(\gamma(s)\big) = s$.

Abstraction: an Example

- The concrete model:



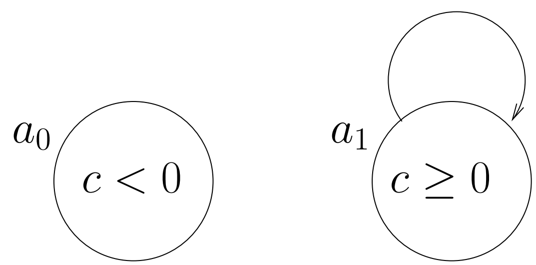
- Abstraction criteria: $\varphi_1 = (c < 0)$ and $\varphi_2 = (c \geq 0)$
- Over-approximation:



✎ The abstract model *simulates* the concrete model.

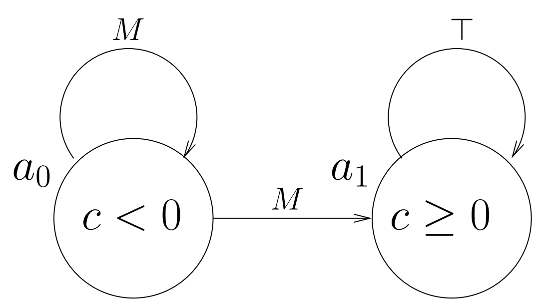
Abstraction: an Example (Cnt'd)

- Under-approximation



✎ The abstract model is *simulated* by the concrete model.

- 3-val (mixed) Abstraction:



Manual Generation of Abstractions

- Let A and C be Kripke structures.
- A is an over-approximation abstraction of C iff

$$\begin{array}{ll} (1) & \forall s \in I_c \cdot \alpha(s) \in I_a \\ (2) & \forall s_0, s_1 \in \Sigma_c \cdot R_c(s_0, s_1) \Rightarrow R_a(\alpha(s_0), \alpha(s_1)) \end{array}$$

```
init_simulation: THEOREM
 init(s) IMPLIES a_init(abst(s))

may_next_simulation: THEOREM
 next(s0, s1) IMPLIES a_next(abst(s0), abst(s1))
```

✔ The above soundness criteria were proven by PVS's `(grind)` rule without need for human guidance.

Predicate Abstraction

- Let $\varphi_1, \ldots, \varphi_n$ be a set of abstraction predicates, and let $b_1, \ldots, b_n$ be abstract boolean variables.

$$\gamma(f(b_1, \ldots, b_n)) = f(\varphi_1/b_1, \ldots, \varphi_n/b_n)$$
$$\alpha(\psi) = \bigwedge\{f(b_1, \ldots, b_n) \mid \psi \Rightarrow f(\varphi_1/b_1, \ldots, \varphi_n/b_n)\}$$

- It is hard to compute α because
  - there are 2^{2^n} distinct boolean functions $f(b_1, \ldots, b_n)$.
- A simplified version of α is:

$$\alpha(\psi) = \bigwedge\{b_i \mid \psi \Rightarrow b_i\}$$

- Let $\varphi_1 = (c < 0)$ and $\varphi_2 = (c \geq 0)$. Then, $\alpha(c > 5) = \neg\varphi_1 \wedge \varphi_2$, but what about $\alpha(c \leq 0)$?!

Abstraction in PVS

- A conservative abstraction scheme has been implemented as a proof rule in PVS.
- They improve the abstraction function α:

$$\alpha(\psi) = \bigwedge\{f(b_1, \ldots, b_n) \mid \psi \Rightarrow f(\varphi_1/b_1, \ldots, \varphi_n/b_n)\}$$

  where $f(b_1, \ldots, b_n)$ are all possible disjunctions of variables $b_1, \ldots, b_n$.
- This reduces the number of functions from 2^{2^n} to 3^n.
- Let $\varphi_1 = (c < 0)$ and $\varphi_2 = (c \geq 0)$. Then, $\alpha(c \leq 0) = (\varphi_1 \vee \varphi_2) \wedge (\neg\varphi_1 \vee \neg\varphi_2)$.
- Under-approximation of the abstract function α:

$$\alpha_-(\psi) = \bigvee\{f(b_1, \ldots, b_n) \mid f(\varphi_1/b_1, \ldots, \varphi_n/b_n) \Rightarrow \psi\}$$

Abstraction in PVS: Over-Approximation

- There is an over-approximation transition from a_0 to a_1 iff

$$\exists s, s' \cdot s \in \gamma(a_0) \wedge s' \in \gamma(a_1) \wedge R_c(s, s')$$

- We need to express it as a quantifier-free formula. If $a_0, \ldots, a_k$ are the only successors of a then

$$\forall s, s' \cdot s \in \gamma(a) \wedge R_c(s, s') \Rightarrow s' \in \big(\gamma(a_0) \vee \ldots \vee \gamma(a_k)\big)$$

- Let $\varphi_1 = (c < 0)$ and $\varphi_2 = (c \geq 0)$. Then,

$$\begin{aligned} \forall c, c' \cdot \varphi_1(c) \wedge (c' = c + 1) &\quad \Rightarrow \quad \varphi_2(c') \vee \varphi_1(c') \\ \forall c, c' \cdot \varphi_2(c) \wedge (c' = c + 1) &\quad \Rightarrow \quad \varphi_2(c') \end{aligned}$$

```
(abstract-and-mc ("lambda(s):c(s) >= 0" "lambda(s):c(s) < 0"))
```

Abstraction in PVS: Under-Approximation

- There is an under-approximation transition from a_0 to a_1 iff

$$\forall s \cdot \exists s' \cdot s \in \gamma(a_0) \Rightarrow s' \in \gamma(a_1) \wedge R_c(s, s')$$

- This formula is not in quantifier-free form.
- However, in our example:

$$\begin{aligned} & \forall c \cdot \exists c' \cdot (c \geq 0) \Rightarrow (c' \geq 0) \wedge R_c(c, c') \\ \Rightarrow \quad & (\text{Since } R_c(c, c') \Leftrightarrow (c' = c + 1)) \\ & \forall c \cdot (c \geq 0) \Rightarrow (c + 1 \geq 0) \end{aligned}$$

✎ If the `pre-` or `post-`image function can be written in quantifier-free form, the under-approximation formula can be written in quantifier-free form, as well.

Abstraction in PVS: Under-approximation Cnt'd

- `pre`- and `post`-image functions:

$$\begin{array}{lcl} \texttt{post}(\psi) & \triangleq & \{s' \in S \mid \exists s \in S \cdot R(s,s') \wedge s \models \psi\} \\ \texttt{pre}(\psi) & \triangleq & \{s \in S \mid \forall s' \in S \cdot R(s,s') \Rightarrow s' \models \psi\} \end{array}$$

- Conventional programming languages can be translated into guarded command form,
  - this makes it possible to have q-free `pre` and `post` functions:

$$g(\bar{x}) \wedge \bar{x}' := assign(\bar{x})$$

$$\texttt{pre}(\varphi) = g(\bar{x}) \wedge \varphi(assign(\bar{x})/\bar{x})$$

Example:

$$(x > 5) \wedge x' := x - 10$$

$$\texttt{pre}(x > 0) = (x > 5) \wedge (x - 10 > 0) = (x > 10)$$

3-Valued (or Mixed) Abstraction

- Let $\varphi_1, \ldots, \varphi_n$ be a set of abstraction predicates, and let $b_1, \ldots, b_n$ be the corresponding *3-valued* abstract variables.
- The abstract domain is all possible *3-valued* states:

$$\Sigma_a = \{\bigwedge_{1 \leq i \leq n}(b_i = l) \mid l \in \{\top, M, \bot\}\}$$

- The number of states is 3^n.
- Let $\varphi_1 = (c < 0)$ and $\varphi_2 = (c \geq 0)$. Then,

$$\begin{aligned} \alpha(c \leq 0) &= ((\varphi_1 = M) \wedge (\varphi_2 = M)) \\ &= (\neg\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \neg\varphi_2) \\ &= (\varphi_1 \vee \varphi_2) \wedge (\neg\varphi_1 \vee \neg\varphi_2) \end{aligned}$$

3-Valued Abstraction: the Transition Relation

- There is an over-approximation transition from φ_1 to φ_2 iff $\varphi_1 \wedge \neg\texttt{pre}(\neg\varphi_2)$ is ***satisfiable***.
- There is an under-approximation transition from φ_1 to φ_2 iff $\varphi_1 \wedge \texttt{pre}(\neg\varphi_2)$ is ***unsatisfiable***.
- Let $\varphi_1 = (c < 0)$, $\varphi_2 = (c \geq 0)$, and $R(c, c') \Leftrightarrow (c' = c + 1)$:

(1) $\varphi_1 \xrightarrow{M} \varphi_2$ because $\varphi_1(c) \wedge \varphi_2(c+1)$ is $\textsc{Sat}$:

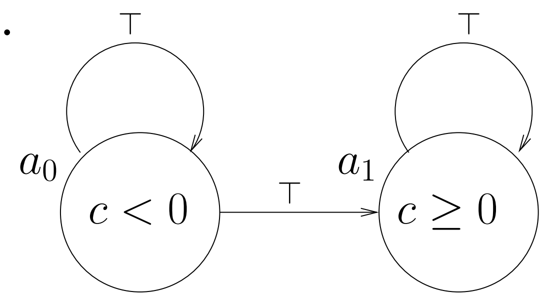
$$\exists c \cdot (c < 0) \wedge (c + 1 \geq 0)$$

(2) $\varphi_2 \xrightarrow{\top} \varphi_2$ because $\neg(\varphi_2(c) \wedge \neg\varphi_2(c+1))$ is a $\textsc{Tautology}$:

$$\forall c \cdot (c \geq 0) \Rightarrow (c + 1 \geq 0)$$

Optimizations to Under-Approximation

- Using a distance-bounded reachability (instead of the immediate-successor) relation for computing the under-approx. transition relation.



- $EF(c \geq 0)$ is conclusive for this abstract model.
- Let $\varphi_1 = (c < 0)$, $\varphi_2 = (c \geq 0)$, and $R(c, c') \Leftrightarrow (c' = c + 1)$, and let $init(c) = -2$:

(1) $\varphi_1 \xrightarrow{\top} \varphi_2$ because

$$\begin{aligned} \forall c \cdot (-2 \leq c < 0) \Rightarrow \quad & (c + 1 \geq 0) \bigvee \\ & (c + 1 \leq 0) \wedge (c + 2 \geq 0) \end{aligned}$$

Concluding Remarks

- **What we did**:
  - surveyed some of the approaches to automated generation of abstract models;
  - demonstrated how distance-bounded reachability can be employed to make an under-approximation transition relation more precise.
- **What we learned**:
  - gained hands-on experience with the PVS toolkit;
    - in particular, we implemented a number of abstraction examples.
  - We found PVS very useful for generating abstract models; *however*, the PVS specification language does not provide a convenient means for describing state transition systems.

Future Work

- **How this work can be pursued**:
  - Finding out whether `pre`-image functions can be used for the elimination of quantifiers in general;
  - Using `pre`-image functions for quantifer elimination in other contexts e.g. SAT-based model-checking.
  - Using fairness assumptions for sharpening the results of abstraction (for some preliminary work in this direction, see the report.)