Symbolic Trajectory Evaluation - A Survey

by

Mihaela Gheorghiu

Department of Computer Science
University of Toronto

Instructor: Prof. Marsha Chechik

January 31, 2004
Motivation

- Simulation vs. verification

- Multi-valued vs. symbolic simulation

- Symbolic Trajectory Evaluation (STE) - a multi-valued symbolic verification method based on simulation

- STE vs. model-checking
Basic STE Theory

Model

- Complete lattice \((S, \leq)\) of states
- Monotonic next-state function \(Y : S \rightarrow S\)

Behaviors

- Infinite sequences \(\sigma_1, \sigma_2, \ldots \in S^\omega\)
- Infinite trajectories - sequences obeying next-state function,
  \[ Y(\sigma_i) \leq \sigma_{i+1}, \text{ for all } i \]
- Lattice order extended to sequences and trajectories pointwise,
  \[ \sigma_1, \sigma_2, \ldots \leq \gamma_1, \gamma_2, \ldots \text{ iff } \sigma_i \leq \gamma_i, \text{ for all } i \]
Example (I)

Inverter circuit

Information partial order
Example (II)

- Each circuit node has *excitation*, constraint on its next value for all states $s_1 s_2$, $y_i(s_1 s_2) = X$, $y_o(s_1 s_2) = \neg s_1$, so $Y = y_i y_o$

- Sequence $1X$, $X0$, $\perp$, $\perp$, ... *is a trajectory*

- Sequence $00$, $\perp$, $\perp$, ... *is not*
Logic

Trajectory formulas

- Atom - *simple state predicate*, monotonic, and having unique lowest state, *defining state*, where true
e.g., \((i \text{ is } 1)\) with defining state \(1X\)
  - true of a trajectory iff true of its initial state

- Conjunction of trajectory formulas \(\varphi \land \psi\) - usual semantics

- Next-time formula \(\mathbf{N}\varphi\) - true of trajectory \(\sigma_1, \sigma_2, \ldots\) iff
  \(\varphi\) true of \(\sigma_2, \sigma_3, \ldots\)

- Nothing else

Exemple for inverter: \((i \text{ is } 1) \land \mathbf{N}(o \text{ is } 0)\)
Specification

Assertions

\[ A \Rightarrow C \]

with \( A, C \) trajectory formulas

- True of a model iff for every trajectory \( \sigma \)
  
  \[ \text{if} \quad \sigma \models A \quad \text{then} \quad \sigma \models C \]

- Better: set of trajectories satisfying \( A \) contained in that of those satisfying \( C \)

Inverter specification:

\[(i \text{ is } 1) \Rightarrow \text{N}(o \text{ is } 0)\]

\[(i \text{ is } 0) \Rightarrow \text{N}(o \text{ is } 1)\]

In LTL:

\[(i \rightarrow \Diamond \neg o) \land (\neg i \rightarrow \Diamond o)\]
Verification (I)

To check \((i \text{ is } 1) \Rightarrow \text{N}(o \text{ is } 0)\)

- Consider *defining sequence*
  \[1X, \bot, \bot, \ldots\]
  for \((i \text{ is } 1)\)
  all trajectories satisfying \((i \text{ is } 1)\) are those above this sequence

- Make it into *defining trajectory*
  \[\tau = 1X, X0, \bot, \bot, \ldots\]
  the lowest trajectory satisfying \((i \text{ is } 1)\)

- Consider defining sequence
  \[\sigma = \bot, X0, \bot, \bot, \ldots\]
  for \(\text{N}(o \text{ is } 0)\)

- Check \(\sigma \leq \tau\)
In general, to check \( A \Rightarrow C \) check \( \sigma_C \leq \tau_A \)

where \( \sigma_C, \tau_A \) are the defining sequence for \( C \) and defining trajectory for \( A \), respectively

Justification
Allow Boolean variables

- Inverter specification becomes

\[(i \text{ is } x) \Rightarrow \neg (o \text{ is } \neg x)\]

- Checked by

\[\bot, X\neg x, \bot, \bot, \ldots \leq xX, X\neg x, \bot, \bot, \ldots\]

- True iff inequality holds for all possible interpretations of the variables

- There are symbolic states, sequences, trajectories, formulas

- Symbolic means *parameterized* by Boolean variables

Implemented using BDDs!
A Few Points

- \( \neg(o \text{ is } 0) \implies (i \text{ is } 1) \) fails:
  \[ 1X, \bot, \bot, \ldots \not\subseteq \bot, X0, \bot, \ldots \]
  simulation only works forward

+ \((i \text{ is } 0) \land (i \text{ is } 1) \implies (o \text{ is } 0)\) succeeds, but vacuity detected
  defining sequence for \((i \text{ is } 0)\): \[0X, \bot, \bot, \ldots\]
  defining sequence for \((i \text{ is } 1)\): \[1X, \bot, \bot, \ldots\]
  defining trajectory for their conjunction \[\top, \bot, \bot, \ldots\]
  by pointwise lub

+ Verification does not depend on the size of the state space

- Four-valued state space, but two-valued verification answer

- Very restricted verification capabilities - only over finite sequences,
  cannot reason about eventuality, or support disjunction, etc.
Beyond Basics

- Fixpoint computations for checking assertions of type $(A \Rightarrow C)^*; G$
- Using enriched syntax and a four-valued information + truth lattice

\[ \begin{array}{c}
0 \quad 1 \\
\downarrow \quad \downarrow \\
X \quad Z
\end{array} \quad \begin{array}{c}
\perp \\
\downarrow \\
\text{false} \quad \text{true}
\end{array} \]

- Generalized STE for checking *assertion graphs* representing all $\omega$-regular properties

\[
\begin{array}{c}
\text{true} \quad \text{true}
\end{array} \quad \begin{array}{c}
\text{no}\_\text{overwrite} \\
\text{data\_correct}
\end{array}
\]

\[
\begin{array}{c}
i \text{ is } x
\end{array} \quad \begin{array}{c}
o \text{ is } \neg x
\end{array}
\]

\[
\begin{array}{c}
\text{write} \quad \text{read}
\end{array}
\]
Forte Tool

- Forte is a formal verification environment implementing STE, used at Intel
- First and current restricted academic version released January 2003
- Essentially performs symbolic simulation, not verification
- Example of STE invocation

```haskell
let ant = [(T, "out[1]", T, 0, 1),
          (T, "out[0]", T, 0, 1),
          (T, "c", F, 0, 1),
          (T, "c", T, 1, 2)];
let cons = [(T, "out[1]", F, 1, 2),
           (T, "out[0]", F, 1, 2)];
STE "" model [] ant cons trace;
```
STE is a special-purpose model-checking method

Successfully used in industry (Intel, IBM, Motorola) to verify large memories and datapath circuits

Relationship to standard model-checking still unclear

Formally shown to be a form of data-flow analysis and its multi-valued models of circuits to be over-approximations of concrete ones

To do: prove a direct relationship with multi-valued abstraction and model-checking as we know them

To do: see how standard model-checking can benefit from STE