Game-based and Simulation-based Improvements for LTL Synthesis

Barbara Jobstmann
Roderick Bloem

Graz University of Technology, Austria
August 21st, 2006
Motivation

- Synthesis from specification fascinating
- Correct by construction - no verification
- You say what, it tells how!
- Goes back to Church (1962)
- What has changed since then?
Outline

• Introduction

• Safraless Approach
  • Steps and Improvements

• Lily

• Conclusion
LTL Synthesis

• Automatically build design from properties

• Input
  • Set of LTL properties
  • Partition of the atomic propositions (input/output signals)

• Output
  • Automatically created functionally correct finite-state machine (Moore)

• Proposed for LTL by Pnueli, Rosner (POPL'89)
FSM and Trees

- Moore machine
  - $r=1, r=0$ .... input alphabet
  - $a=1, a=0$ .. output alphabet
  - Input signal $i$, output signal $o$

- Tree (regular)
  - $r=1, r=0$ .... directions $D$
  - $a=1, a=0$ .. alphabet $\Sigma$ (labeling)
FSM and Trees

- **Moore machine**
  - $r=1, r=0$ .... input alphabet
  - $a=1, a=0$ .. output alphabet
  - Input signal $i$, output signal $o$

- **Tree (regular)**
  - $r=1, r=0$ .... directions $D$
  - $a=1, a=0$ .. alphabet $\Sigma$ (labeling)

\[ \Sigma-labeled \ D-tree \]
Idea

1) Build a tree automaton
   - Directions are input values (D=2^I, input signals I)
   - Alphabet are output values (Σ=2^O, output signals O)
   - Automaton accepts all Σ-labeled D-trees where all paths satisfy the given formula

2) Compute language emptiness

3) Build FSM from the witness (a Σ-labeled D-tree)
Necessary Theory

- Infinite game theory
- Automata theory
  - Branching mode (Nondeterministic, Universal, Alternating)
  - Acceptance condition (Büchi, Co-Büchi, Weak, ..)
  - Input element (Word, Tree)
  - Use of KV's abbreviation (e.g., NBW, UCT, ..)
Alternating Word Automata

• N+U branching (edges we can follow and edges we must follow)

• Notation:
  • Circles represent states
  • Boxes represent universal edges
  • Edges are labeled with sets of labels
ABW

\[
\begin{align*}
\text{Input} & \quad r=1 & r=1 & r=0 \\
\text{word:} & \quad a=0 & a=0 & a=1 \\
\end{align*}
\]

...
ABW

Input word: $a=0$  $r=1$  $a=0$  $r=1$  $r=0$  ...
ABW

Input word: a=0

r=1
a=0

r=0
a=1

...
ABW

Input

<table>
<thead>
<tr>
<th>r=1</th>
<th>r=1</th>
<th>r=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=0</td>
<td>a=0</td>
<td>a=1</td>
</tr>
</tbody>
</table>

word: n5, n14, a=0, a=1

Input word: n5, n14, a=0, a=1

Barbara Jobstmann
Seattle, Aug 21\textsuperscript{st}
Improvements for LTL Synthesis
Tree Automata

Universal edges:
Foreach direction, follow only the matching edges

Diagram with transition labels:
- init
- n5
- n14

Transition labels:
- $r=0$
- $a=1$
- $a=0$

Barbara Jobstmann
Seattle, Aug 21st
Improvements for LTL Synthesis
Improvements for LTL Synthesis

Input tree:

\[ a = 0 \]
Input tree:

Universal and tree branching
Improvements for LTL Synthesis

Barbara Jobstmann

Seattle, Aug 21st

Graz University of Technology
Barbara Jobstmann
Seattle, Aug 21st
Improvements for LTL Synthesis
Standard Approach [PR89]

• Construct a nondeterministic Büchi automaton
• Determinize it with Safra’s determinization construction → Rabin word automaton
• Convert to Rabin tree automaton
• Check Language Emptiness
Issues

- 2EXP worst case complexity
- Safra's determinization construction
Solutions

• Concentrate on subsets
  • Alur, Madhusudan, Nam (BMC'03, STTT'05)
  • Wallmeier, Hütter, Thomas (CIAA'03)
  • Harding, Ryan, Schobbens (TACAS'05)
  • Piterman, Pnueli, Sa'ar (VMCAI'06)

• Full LTL (Safraless approach)
  • Kupferman, Vardi (FOCS'05)
  • Kupferman, Piterman, Vardi (CAV'06)
Safraless Approach [KV05]

1) Build a UCT
   - Negate $\varphi$
   - Build an NBW for $\neg \varphi$
   - Invert NBW $\rightarrow$ UCT

2) Convert to AWT

3) Convert to NBT

4) Check Language Emptiness

\begin{align*}
\Phi + \text{i/o} & \rightarrow \text{Build UCT} \\
\text{UCT} & \rightarrow \text{Build AWT} \\
\text{AWT} & \rightarrow \text{Build NBT} \\
\text{NBT} & \rightarrow \text{Lang. Emp.} \\
\end{align*}

$\begin{align*}
\text{L(UCT)} &= \text{L}^T(\varphi) \\
\text{L(AWT)} &\subseteq \text{L}^T(\varphi) \\
\text{L}^T(\varphi) &\neq \emptyset \rightarrow \text{L(AWT)} \neq \emptyset \\
\text{L(NBT)} &\subseteq \text{L}^T(\varphi) \\
\text{L}^T(\varphi) &\neq \emptyset \rightarrow \text{L(NBT)} \neq \emptyset
\end{align*}$
Safrless Approach

\[ \varphi + i/o \]

Build UCT

Build AWT

Build NBT

Lang. Emp.

FSM
Build a UCT for $\varphi$ (1)

- **NBW for $\neg \varphi$**
  - Build a generalized NBW
  - Apply Counting construction $\rightarrow$ Single fairness condition
  - Optimizations of Somenzi, Bloem (CAV'00)

- **Example:**
  - $\varphi = G(F(\text{timer})) \rightarrow G(\text{light} \rightarrow \text{light} U \text{timer}))$
  - NBW for $\neg \varphi$ build with Wring

![Diagram of UCT]
Build a UCT for $\varphi$ (2)

- **Dualize:**
  - Nondeterminism $\rightarrow$ Universality
  - Büchi $\rightarrow$ Co-Büchi (accepting states $\rightarrow$ $\alpha$-states)

- **Word $\rightarrow$ Tree**
  - Inputs $\rightarrow$ Directions (Universality)
  - Outputs $\rightarrow$ Labels

- **Very similar structure**
UCT Example

φ = G(F(timer)) → G(light → light U timer)
Simplify UCT

- **Idea**
  - Approximate states with empty language (accept no tree)
  - Trees with non-accepting path are rejected
  - Environment can force a non-accepting path
  - Sufficient (but not necessary) for language emptiness
Simplify UCT

- **Game**
  - System picks the *label* and the *nondeterminism*
  - Environment picks *direction* and *universality*

- **State** $s$ is winning for environment $\rightarrow L^T(s)$ empty

n3...$\alpha$-state (co-büchi)
Simplify UCT

- **Game**
  - System picks the label and the nondeterminism
  - Environment picks direction and universality

- **State s is winning for environment** → $L^T(s)$ empty

n3...α-state (co-büchi)
Safrless Approach

\[ \varphi + i/o \]

Build UCT

\[ UCT \]

Build AWT

\[ AWT \]

Build NBT

\[ NBT \]

Lang. Emp.

\[ FSM \]
Build AWT

- **Idea**
  - UCT accepts if $\alpha$-states avoided
  - Build layers by copying the UCT
  - Alternate accepting and non-accepting layers
  - Tr of $\alpha$-states in accepting layers is false
  - In accepting layers visit to $\alpha$-states not possible $\rightarrow$ only finitely many visits to $\alpha$-states

- **Example**

  ![Example Diagram](image)

  $n_2, n_4...$ $\alpha$-states (co-büchi)
AWT Example

Layer 2

Layer 1 (accepting)

Layer 0

n2, n4...α-states (co-büchi)
AWT Example

Layer 0

Layer 1 (accepting)

Layer 2

n2, n4...\(\alpha\)-states (co-büchi)

-Release function

\#nd-edges(n1,2) = 9
AWT Example

Layer 0

Layer 1 (accepting)

Layer 2

n2, n4...α-states (co-büchi)

-Release function

#nd-edges(n1,2)=9
Simplify AWT: Construction

- **Simplify release function**
  - cf. Gurumurthy, Kupferman, Somenzi, Vardi (CHARME'03)

- **Ideas**
  - Release just between two layers (from i to i and i-1)
  - Stay in accepting layer if possible
  - Do not distinguish between directions if same state
Simpler Release Function

Layer 5 (accepting)

Layer 4

Layer 3 (accepting)

Layer 2

Layer 1 (accepting)

Layer 0

#edges > #edges of UCT

#nd-edges(n1,2)=2
Simplify AWT: Game-based

- Game-based Simplification like on UCT

A diagram showing a game-based simplification process with nodes labeled as n1,0 to n5,5 and actions a and d.
Simplify AWT: Simulation-based

- Simulation relation for tree automaton
  - cf. Alur, Henzinger, Kupferman, Vardi (CONCUR'98)
  - cf. Fritz, Wilke (FSTTCS'02)

- Idea:
  - State $B$ simulates state $A$ → tree accepted by $A$ is accepted by $B$, $L^T(A) \subseteq L^T(B)$
  - Nondeterministic edges to $A$ and $B$, remove edge to $A$
  - Universal edges to $A$ and $B$, remove edge to $B$

- Use a game to compute simulation relation
Simplify AWT: Simulation-based

- Turn based game between simulated and simulating state
- Game for B simulates A
  - A picks label and his nondeterminism
  - B picks her nondeterminism
  - A picks direction
  - A picks universality of B
  - B picks universality of A
- Winning Condition
  - Player who has to choose from empty set, looses
  - B wins a play, if whenever A is in an accepting state also B is in an accepting state (direct simulation)
Simplify AWT: Simulation-based

Sim. equivalent
\[ n_{4,4} =_{\text{AWT}} n_{4,2} \]
\[ n_{5,4} =_{\text{AWT}} n_{5,2} \]
\[ n_{5,3} =_{\text{AWT}} n_{5,1} \]

A simulated by B
\[ n_{3,4} \leq_{\text{AWT}} n_{3,3} \]
\[ n_{5,2} \leq_{\text{AWT}} n_{5,1} \]
Safralless Approach

\[ \varphi + i/o \]

Build UCT

\[ UCT \]

Build AWT

\[ AWT \]

Build NBT

\[ NBT \]

Lang. Emp.

\[ FSM \]
Build NBT

- Tree-version of Miyano-Hayashi's construction
- Extended subset construction
- NBT-state $\approx$ set of AWT-states
Simplify NBT

- **Use simulation relation from AWT**
  - cf. Fritz (CIAA'03)

- **Idea**
  - NBT-state ≈ set of AWT-states
  - Language is conjunct of all sub-language

- **Simplifications**
  - Two AWT-states A,B in set (NBT-state), B simulates A → B not necessary
  - Two NBT-states S0,S1, if
    - Forall A in S0
    - Exists B in S1 such that A is simulated by B,
      Then S0 is simulated by S1
  - NBT-states S3, edges to S0 and S1 → construct only edge to S1
Safralless Approach

\[ \varphi + i/o \]

Build UCT

\[ UCT \]

Build AWT

\[ AWT \]

Build NBT

\[ NBT \]

Lang. Emp.

\[ FSM \]
Language Emptiness

- Language Emptiness of tree automata is a game
  - Gurevich, Harrington (STOC'82)
- For NBT - Simple Büchi game
  - System picks: Label and transitions
  - Environment picks: direction
  - Winning condition: visit accepting states infinitely often
- Outcome
  - Game lost ≈ not realizable or
  - Winning strategy ≈ (regular) $\Sigma$-labeled D-tree ≈ FSM
Combine MH + Language Emptiness

- **Incremental MH**
- **Idea**
  - We just need one witness
  - Expanded NBT in a breadth-first manner
  - Compute language emptiness on expanded part:
    - If witness found $\rightarrow$ Finished
    - else proceed with expanding
Lily - Linear Logic Synthesizer

- First tool to offer synthesis for full LTL
- Based on Fabio Somenzi's Wring
- Implements all mentioned optimizations
- http://www.ist.tugraz.at/staff/jobstmann/lily/
Conclusion

- Still a long way
- A nice toy to play with
- Example are small but useful for property debugging (or learning LTL)
- Optimizations are enabling factor
- Future?
Thank you for your attention!