An Unfolding Approach to Model Checking

Javier Esparza

Laboratory for Foundations of Computer Science
University of Edinburgh
Concurrent programs

Program: a tuple \( P = (T_1, \ldots, T_n) \) of finite labelled transition systems

\[
T_i = (A_i, S_i, \Delta_i, s_{0i}), \quad 1 \leq i \leq n
\]

where

- \( A_i \) is an alphabet of \textit{actions},
- \( S_i \) is a finite set of \textit{(local) states},
- \( \Delta_i \subseteq S_i \times A_i \times S_i \) is a \textit{transition relation}, and
- \( s_{0i} \in S_i \) is the \textit{initial state}. 
Semantics

The behaviour of $P$ is defined by the (reachable subset of) the global transition system

$$T_P = (A, S, \Delta, s_0)$$

where

- $A = A_1 \cup \ldots \cup A_n$
  
  (A partitioned into visible and invisible actions),

- $S = S_1 \times \ldots \times S_n$
  
  ($s(i)$ denotes the $i$th component of $s \in S$),

- $s_0 = (s_{01}, \ldots, s_{0n})$,

- $(s, a, s') \in \Delta$ iff for every $1 \leq i \leq n$
  
  - $a \in A_i \implies (s(i), a, s'(i)) \in \Delta_i$, and
  
  - $a \notin A_i \implies s(i) = s'(i)$.
Reducing the model checking problem

The model checking problem for a program $P = (T_1, \ldots, T_n)$ can be reduced to (several instances of) the following problems:

**The forbidden trace problem (FT)**
Given: Program $P$, action $a$.
To decide: Does $T_P$ exhibit a forbidden trace, i.e., a trace $a_0a_1a_2 \ldots a_n \in A^*$ such that $a_n = a$?

**The forbidden infinite trace problem (FIT)**
Given: Program $P$, action $a$.
To decide: Does $T_P$ exhibit a forbidden infinite trace, i.e., an infinite trace $a_0a_1a_2 \ldots \in A^\omega$ such that $a_i = a$ for infinitely many $i \geq 0$?

**The livelock problem (L)**
Given: Program $P$, action $a$.
To decide: Does $T_P$ exhibit a livelock, i.e., an infinite trace $a_0a_1a_2 \ldots \in A^\omega$ such that $a_i = a$ for some $i$, and $a_j$ is invisible for every $j > i$?
A first analysis

- Complexity of FT, FIT and L: PSPACE-complete.
  - Standardsolution: computetheglobaltransitionsystem $T$ and use well-known graph algorithms. Time and space complexity $O(|T|)$.
  - Problem: exponential in the size of $T$ for very easy instances, e.g. for completely independent processes.
  - In practice often $O(|T|^2)$.

Compact "proof objects", exponentially smaller than $|T|$.

- Our solution: work on the unfolding of the system.

- Proof objects of size $O(|T|)$ for FT, and of size $O(|T|^2)$ for FIT and L.

In this talk: only FT and FIT (L more technical).
The unfolding
Second tree
In this case \( \mathcal{L} = \mathcal{L}^p \).

An example with \( u = 1 \).
**Solving FT for** $n = 1$

Proof tree: Prefix of the unfolding of $P$.

A node $n$ is a **terminal** if

(I) it is reached by an event labelled by $a$, or
(II) the proof tree constructed so far contains a node $n'$ labelled by the same state as $n$.

A **tableau** is a (minimal) proof tree such that all leaves are terminals.

A terminal $n$ is **successful** if it is of type I.
A proof tree is **successful** if it has at least one successful terminal.

**Theorem**: $(P,a)$ exhibits a forbidden trace iff $(P,a)$ has a successful proof tree.
Problem: Terminals are local states, but a terminal’s definition must refer to global states.

Idea [McMillan 92, 95]: Associate to each node \( n \) of the unfolding a global state \( GS(n) \) as follows:

- let \( Hist(n) \) be the “history” of \( n \);
- let \( GS(n) \) be the result of “executing” \( Hist(n) \).

New definition of terminal:

A node \( n \) is a **terminal** if

(I) it is reached by an event labelled by \( a \), or

(II) the proof tree constructed so far contains a node \( n' \) such that \( GS(n) = GS(n') \).

A terminal \( n \) is **successful** if it is of type I.
The attempt fails!
Adequate orders

$GS(n) = GS(n')$ too weak for a terminal

An order $\preceq$ on histories is adequate if it

- is well-founded,
- preserves causality: if $h$ prefix of $h'$ then $h \preceq h'$, and
- is preserved by finite extensions: if $h \preceq h'$, then $h \cdot h'' \preceq h' \cdot h''$ for all $h''$.

We say $n \preceq n'$ if $Hist(n) \preceq Hist(n')$.

New definition of terminal:

A node $n$ is a terminal if

(I) it is reached by an $a$-transition, or
(II) the proof tree constructed so far contains a node $n' \preceq n$ such that $GS(n) = GS(n')$.

A terminal $n$ is successful if it is of type I.

Theorem: $(P,a)$ has a forbidden trace iff it has a successful tableau.
Problem: \( n' \preceq n \) is an additional condition
\[ \Rightarrow \text{ less places are terminals} \]
\[ \Rightarrow \text{ proof trees can be bigger!} \]

- In [McMillan 92]:
  \( h \preceq h' \) if size of \( h \) smaller than size of \( h' \).
  Tableaux can be \textit{exponentially} bigger than \( T_P \)

- In [E.,Römer,Vogler 96, E., Römer 99]:
  \textbf{Total} orders \( \preceq \).

  \textbf{Theorem}: Any tableaux in which the events are added in \( \preceq \)-order has size \( O(T_P) \).
A solution to FIT for $n = 1$

A node $n$ is a terminal if $Hist(n)$ contains a node $n'$ labelled with the same state as $n$.

A terminal $n$ is successful if $Hist(n) - Hist(n')$ (the path from $n'$ to $n$) contains some $a$-labelled transition.

**Theorem:** $(P,a)$ exhibits a forbidden infinite trace iff $(P,a)$ has a successful tableau.

**Problem:** tableaux can be exponentially larger than $T_P$
New definition of terminal:

A node \( n \) is a terminal if there is a node \( n' \) labelled with the same state as \( n \) such that

- \( n' \) belongs to \( Hist(n) \), or
- \( n' \) does not belong to \( Hist(n) \), and \( Hist(n') \) contains at least as many \( a \)-labelled transitions as \( Hist(n) \).

A terminal \( n \) is successful if \( n' \) belongs to \( Hist(n) \) and \( Hist(n) \setminus Hist(n') \) contains some \( a \)-labelled transition.

**Theorem:** \((P, a)\) has a forbidden infinite trace iff it has a successful tableau. The size of any tableau in which events are added in \( \leq \)-order is \( O(|T_P|^2) \).
Definition of terminal:

A node \( n \) is a terminal if there is a node \( n' \leq n \) such that \( GS(n') = GS(n) \), and

- \( n' \) belongs to \( Hist(n) \), or
- \( n' \) does not belong to \( Hist(n) \), and \( Hist(n') \) contains at least as many \( a \)-labelled transitions as \( Hist(n) \).

A terminal \( n \) is successful if \( n' \) belongs to \( Hist(n) \) and \( Hist(n) \setminus Hist(n') \) contains some \( a \)-labelled transition.

**Theorem:** \((T_P, a)\) has a forbidden infinite trace iff it has a successful tableau.

With the adequate orders of [E., Römer, Vogler 96] and [E., Römer 99] the size of any tableau is at most \( O(|T_P|^2) \).
A successful tableau
<table>
<thead>
<tr>
<th>System</th>
<th>Structured Petri net</th>
<th>Prefix</th>
<th>Places</th>
<th>Traps</th>
<th>Places</th>
<th>Traps</th>
<th>Global States</th>
<th>Places</th>
<th>Traps</th>
<th>Places</th>
<th>Traps</th>
<th>BDD size</th>
<th>(scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW(10)</td>
<td>7576</td>
<td>15974</td>
<td>29132</td>
<td>1.6 \times 10^6</td>
<td>96</td>
<td>86</td>
<td>(rw(10))</td>
<td>257</td>
<td>150</td>
<td>45460</td>
<td>256</td>
<td>385</td>
<td>2.8 \times 10^7</td>
</tr>
<tr>
<td>DME(64)</td>
<td>210249</td>
<td>780</td>
<td>1671</td>
<td>2.8 \times 10^7</td>
<td>140</td>
<td>150</td>
<td>DME(64)</td>
<td>736</td>
<td>736</td>
<td>40188</td>
<td>231</td>
<td>241</td>
<td>&lt; 3.1 \times 10^6</td>
</tr>
<tr>
<td>C(7)</td>
<td>40188</td>
<td>1035</td>
<td>2164</td>
<td>&lt; 3.1 \times 10^6</td>
<td>202</td>
<td>150</td>
<td>C(7)</td>
<td>736</td>
<td>736</td>
<td>40188</td>
<td>231</td>
<td>241</td>
<td>3235</td>
</tr>
<tr>
<td>SR(10)</td>
<td>40188</td>
<td>16935</td>
<td>32354</td>
<td>4.3440</td>
<td>99</td>
<td>736</td>
<td>SR(10)</td>
<td>736</td>
<td>736</td>
<td>40188</td>
<td>231</td>
<td>241</td>
<td>119450</td>
</tr>
<tr>
<td>DPD(7)</td>
<td>40188</td>
<td>119450</td>
<td>86180</td>
<td>8.1 \times 10^7</td>
<td>100</td>
<td>100</td>
<td>DPD(7)</td>
<td>736</td>
<td>736</td>
<td>40188</td>
<td>231</td>
<td>241</td>
<td>109965</td>
</tr>
<tr>
<td>CY(12)</td>
<td>40188</td>
<td>86310</td>
<td>74264</td>
<td>&lt; 3.1 \times 10^6</td>
<td>100</td>
<td>100</td>
<td>CY(12)</td>
<td>736</td>
<td>736</td>
<td>40188</td>
<td>231</td>
<td>241</td>
<td>8.1 \times 10^7</td>
</tr>
</tbody>
</table>
 RW: Readers and Writers (Hellwagner)

DME: STG specification of a circuit for distributed mutual exclusion

CP: Concurrent Pushers (Heimer)

PC: Production cell

EL: Elevator

SR: Slotted ring protocol

DPP: Philosophers with deadlock avoidance

CY: Cycler (Milner)
Unfoldings vs. BDDs

Conceptual similarities and differences:

- Both techniques *compress* the state space
- BDDs exploit *regularity*
  Unfoldings exploit *concurrency*

Consequences:

- **Robustness**: Unfoldings less sensitive to changes in the system
- **Compression**: Prefix smaller for loosely coupled systems, BDDs smaller for tightly coupled systems
<table>
<thead>
<tr>
<th>BDD size</th>
<th>Nr. of phil.</th>
<th>Min.</th>
<th>Max.</th>
<th>St. Dev.</th>
<th>Average</th>
<th>St. Dev.</th>
<th>Average</th>
<th>BDD size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.54</td>
<td>4</td>
<td>85798</td>
<td>429903</td>
<td>9742</td>
<td>33742</td>
<td>4855</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1.76</td>
<td>6</td>
<td>85389</td>
<td>47032</td>
<td>632</td>
<td>4855</td>
<td>3140</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1.48</td>
<td>8</td>
<td>4637</td>
<td>27516</td>
<td>510</td>
<td>3140</td>
<td>340</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>0.92</td>
<td>10</td>
<td>1437</td>
<td>9678</td>
<td>800</td>
<td>1555</td>
<td>583</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0.52</td>
<td>12</td>
<td>305</td>
<td>1716</td>
<td>248</td>
<td>583</td>
<td>583</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>14</td>
<td>52</td>
<td>455</td>
<td>94</td>
<td>583</td>
<td>94</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

BDD for the set of reachable states (Petri net)

100 random tables with right-handed, left-handed, and ambidextrous philosophers
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.06</td>
<td>9.25</td>
<td>185</td>
<td>140</td>
<td>161</td>
<td>14</td>
<td>4</td>
<td>117</td>
<td>0.09</td>
<td>5.13</td>
<td>0.10</td>
<td>5</td>
</tr>
<tr>
<td>0.05</td>
<td>0.74</td>
<td>160</td>
<td>120</td>
<td>141</td>
<td>12</td>
<td>6</td>
<td>117</td>
<td>0.09</td>
<td>5.99</td>
<td>0.09</td>
<td>6</td>
</tr>
<tr>
<td>0.07</td>
<td>0.78</td>
<td>135</td>
<td>100</td>
<td>117</td>
<td>10</td>
<td>8</td>
<td>95</td>
<td>0.07</td>
<td>6.92</td>
<td>0.07</td>
<td>8</td>
</tr>
<tr>
<td>0.07</td>
<td>0.692</td>
<td>110</td>
<td>85</td>
<td>90</td>
<td>8</td>
<td>6</td>
<td>70</td>
<td>0.05</td>
<td>5.99</td>
<td>0.05</td>
<td>6</td>
</tr>
<tr>
<td>0.09</td>
<td>0.99</td>
<td>58</td>
<td>60</td>
<td>70</td>
<td>6</td>
<td>4</td>
<td>46</td>
<td>0.07</td>
<td>5.13</td>
<td>0.07</td>
<td>4</td>
</tr>
<tr>
<td>0.10</td>
<td>3.13</td>
<td>60</td>
<td>40</td>
<td>40</td>
<td>4</td>
<td>4</td>
<td>46</td>
<td>0.05</td>
<td>4.13</td>
<td>0.05</td>
<td>4</td>
</tr>
</tbody>
</table>

100 random tables with right-handed, left-handed, and ambidextrous philosophers.
Checking deadlock-freedom with BDDs

100 random tables with right-handed, left-handed, and ambidextrous philosophers

<table>
<thead>
<tr>
<th>Phil.</th>
<th>Average</th>
<th>Min.</th>
<th>Max.</th>
<th>Std. Dev.</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.45</td>
<td>0.13</td>
<td>0.05</td>
<td>0.08</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>1.18</td>
<td>0.20</td>
<td>14.60</td>
<td>2.45</td>
<td>0.16</td>
</tr>
<tr>
<td>6</td>
<td>3.36</td>
<td>0.20</td>
<td>15.80</td>
<td>4.60</td>
<td>0.46</td>
</tr>
<tr>
<td>8</td>
<td>4.14</td>
<td>0.14</td>
<td>1.18</td>
<td>1.45</td>
<td>0.36</td>
</tr>
<tr>
<td>10</td>
<td>0.80</td>
<td>0.05</td>
<td>0.36</td>
<td>0.16</td>
<td>0.08</td>
</tr>
<tr>
<td>12</td>
<td>0.29</td>
<td>0.05</td>
<td>0.13</td>
<td>0.02</td>
<td>0.05</td>
</tr>
</tbody>
</table>

SMV on a SUN Ultra 60, 2 processors, 640 MB

100 random tables with right-handed, left-handed, and ambidextrous philosophers

Chekking deadlock-freedom with BDDS
Checking deadlock-freedom with unfoldings

100 random tables with right-handed, left-handed, and ambidextrous philosophers

PEP + stable models on a SUN Ultra 60, 2 processors, 640 MB

<table>
<thead>
<tr>
<th>Nr. of phil.</th>
<th>Time in seconds</th>
<th>Min</th>
<th>Max</th>
<th>St. Dev</th>
<th>Ave.</th>
<th>St. Dev</th>
<th>Ave.</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.17</td>
<td>0.05</td>
<td>0.03</td>
<td>0.007</td>
<td>0.03</td>
<td>0.016</td>
<td>0.17</td>
</tr>
<tr>
<td>10</td>
<td>0.17</td>
<td>0.02</td>
<td>0.04</td>
<td>0.007</td>
<td>0.02</td>
<td>0.016</td>
<td>0.17</td>
</tr>
<tr>
<td>12</td>
<td>0.20</td>
<td>0.04</td>
<td>0.03</td>
<td>0.007</td>
<td>0.03</td>
<td>0.016</td>
<td>0.17</td>
</tr>
<tr>
<td>14</td>
<td>0.22</td>
<td>0.07</td>
<td>0.07</td>
<td>0.009</td>
<td>0.07</td>
<td>0.016</td>
<td>0.17</td>
</tr>
<tr>
<td>16</td>
<td>0.28</td>
<td>0.04</td>
<td>0.04</td>
<td>0.007</td>
<td>0.04</td>
<td>0.016</td>
<td>0.17</td>
</tr>
<tr>
<td>18</td>
<td>0.30</td>
<td>0.05</td>
<td>0.05</td>
<td>0.007</td>
<td>0.05</td>
<td>0.016</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Checking deadlock-freedom with unfoldings
Unfoldings vs. stubborn sets

Conceptual similarities and differences:

- Both techniques exploit concurrency
- Stubborn sets discard information
  Unfoldings compress information
- Stubborn sets are conservative: small overhead is guaranteed, at the price of a suboptimal reduction
  Unfoldings “take risks”: large overhead is possible, but optimal compression

Consequences:

- No loss of information \(\rightarrow\) All reachability properties checkable on the same prefix
- Causality information available
  (ex. alarm patterns)
- Larger overheads for tightly coupled systems
Checking deadlock freedom with stubborn sets and unfolding

<table>
<thead>
<tr>
<th>Nr. of phil.</th>
<th>Time in seconds</th>
<th>PROD</th>
<th>PEP + smodels</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>18</td>
<td>PROD</td>
<td>PEP + smodels</td>
</tr>
<tr>
<td>12</td>
<td>69</td>
<td>PROD</td>
<td>PEP + smodels</td>
</tr>
<tr>
<td>14</td>
<td>834</td>
<td>PROD</td>
<td>PEP + smodels</td>
</tr>
<tr>
<td>16</td>
<td>5003</td>
<td>PROD</td>
<td>PEP + smodels</td>
</tr>
<tr>
<td>18</td>
<td>29257</td>
<td>PROD</td>
<td>PEP + smodels</td>
</tr>
</tbody>
</table>

SUN Ultra 60, 2 processors, 640 MB

- SUN Ultra 60, 2 processors, 640 MB
- 1 left-handed, 1 right-handed, and (n - 2) ambidextrous philosophers
Tentative rules of thumb

- Unfoldings more suitable for highly concurrent systems
- Stubborn sets and unfoldings more suitable for irregular but concurrent systems
- Stubborn sets more suitable for systems with little concurrency
- BDDs more suitable for very regular systems

- BDDs more suitable for very regular systems