Verifying Incomplete and Evolving Specifications

Claudio Menghi
menghi@elet.polimi.it
Verifying Specifications

Requirements $\phi$

Specification M

Verification

yes

no + counterexample
Evolving
Evolving
Incomplete
Incomplete and Evolving

Incompleteness

Evolution
Verifying Incomplete and Evolving Specifications

Requirements

Specification

Verification

yes

no + count

maybe + constraints
Constraints

Constraints are conditions (necessary and sufficient) that, if satisfied, guarantee that $\phi$ holds for M.
The idea in practice

\[ E(\neg(\bullet) \cup (\star)) \]
The idea in practice

$E(\neg( \text{??} ) \cup ( \text{??} ))$
The idea in practice

\[ E(\neg(\text{??})) \cup (\text{??}) \]

\[ E(\neg(\text{??})) \cup (\text{??}^\text{\(^\neg\)(??)}) \]

\[ E(\neg(\text{??})) \cup (\text{??}) \]
The idea in practice

\[ E(\neg(???))U(???)^{\wedge(\neg???)} \]

\[ E(\neg(???))U( ))) \]

\[ E(\neg(???))U( )) ) \]
The idea in practice

$E(\neg(\bullet)) \cup (\bullet)$

$E(\neg(\bullet)) \cup (\wedge(\neg\bullet))$

$E(\neg(\bullet)) \cup (\bullet)$
Consideration of our Approach

The efficiency of the approach strongly depends on the number and on the type of the constraints generated.
Consideration of our Approach

(1) it is possible to check whether an initial, incomplete and high level description of the system (such as the one produced during earlier stages of the software design) satisfies a given property.

\[ E(\neg(\mathit{\text{Wario}}))U(\mathit{Princess Peach}) \]
Consideration of our Approach

(2) the verification of incomplete models supports the replacement of complex parts of the design with incomplete parts to reduce the verification time.
Consideration of our Approach

(3) in the **adaptive system** case, the verification of incomplete models allows to understand whether the system satisfies its requirements when the different components are removed (plugged) into the running system
## Formalisms

<table>
<thead>
<tr>
<th>Specification</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled Transition Systems (LTS)(^1)</td>
<td>Computation Tree Logic (CTL)</td>
</tr>
<tr>
<td>Buchi automata</td>
<td>Linear-Time Temporal Logic (LTL)</td>
</tr>
<tr>
<td>Statechart(^2,3)</td>
<td>Computation Tree Logic (CTL)</td>
</tr>
<tr>
<td>Markov Chains</td>
<td>Probabilistic CTL</td>
</tr>
<tr>
<td>Ambient Calculus</td>
<td>Ambient Logic</td>
</tr>
</tbody>
</table>


Research Progress

2013

2014

2015

LTS vs CTL

Statecharts vs CTL

Buchi automata vs LTL

PhD Starting Point
Sharifloo, Amir Molzam, and Paola Spoletini.
"Lover: light-weight formal verification of adaptive systems at run time."
Formal Aspects of Component Software.
Thanks to

Carlo Ghezzi

Paola Spoletini

Amir Molzam Sharifloo
Questions ?