Late Choice
Model Checking Asynchronous Systems with Queues using Constraints

Claus Traulsen
ctr@informatik.uni-kiel.de
joint work with Martin Steffen

Department of Computer Science and Applied Mathematics
Christian-Albrechts University of Kiel

October 19, 2005
1. Motivation and Definitions
2. A semantics with constraints
3. Use constraints for model checking
4. Experimental results
5. Conclusion
Problem Setting

Enumerative Model Checking
- enumerate the reachable states
- show that a specification holds for all paths in the state-graph
- main problem: state explosion

Queues
- are often needed to model protocols and distributed systems
- may lead to additional state explosion
State-Explosion caused by Queues
Late Choice

- the value is specified as soon as A or B uses it
- pointers are hard to formalize
  → use constraints instead
Constraints

Definition: subset of $\text{Dom}(x_1) \times \cdots \times \text{Dom}(x_n)$

Sub-constraints: $C \subseteq U \cap D$, $U \subseteq \text{var}(C) \cap \text{var}(D)$

Representation: set of equations and in-equations

- set of variables is implicitly given
- constraint is set of valuations
  $\sigma : \text{Var} \leftrightarrow \text{Dom}$ which solve this equations and in-equations
**Syntax**

- Parallel composition of LTS
- Communication via bounded queues

### Labels

<table>
<thead>
<tr>
<th>Label Type</th>
<th>Label</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>$l \xrightarrow{a?x} l'$</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>$l \xrightarrow{g\triangleright a!e} l'$</td>
<td></td>
</tr>
<tr>
<td>Assign</td>
<td>$l \xrightarrow{g\triangleright x:=e} l'$</td>
<td></td>
</tr>
<tr>
<td>External Input</td>
<td>$l \xrightarrow{ext?_y,z^x} l'$</td>
<td></td>
</tr>
<tr>
<td>Skip</td>
<td>$l \xrightarrow{g} l'$</td>
<td></td>
</tr>
</tbody>
</table>
Defining the semantics: receiving data

- **StdInput:**

\[
\frac{\overrightarrow{I} \xrightarrow{a?x} \overrightarrow{I'}} \quad \text{len}(a_Q) > 0 \quad v = \text{head}(a_Q)}
{(\overrightarrow{I}, \sigma, Q) \rightarrow (\overrightarrow{I'}, \sigma[x \mapsto v], Q[tail(a)/a])}
\]

- **ConInput:**

\[
\frac{\overrightarrow{I} \xrightarrow{a?x} \overrightarrow{I'}} \quad \text{len}(a_Q) > 0 \quad y = \text{head}(a_Q) \quad \text{new } x'}
{(\overrightarrow{I}, C, Q) \rightarrow (\overrightarrow{I'}, C[x'/x] \cup \{x = y\}, Q[tail(a)/a])}
\]
Defining the semantics: external inputs

- **StdExtInput:**

\[
\frac{\vec{l} \overset{\text{ext?}_{[y,z]}^x}{\longrightarrow} \vec{l}'}{\begin{array}{l}
\sigma \leq v \leq [z]_\sigma \\
(\vec{l}, \sigma, Q) \rightarrow (\vec{l}', \sigma[x \mapsto v], Q)
\end{array}}
\]

- **ConExtInput:**

\[
\frac{\vec{l} \overset{\text{ext?}_{[y,z]}^x}{\longrightarrow} \vec{l}'}{\begin{array}{l}
C \cup \{y \leq z\} \not\models \bot \quad \text{new } x'
\end{array}}
\]

\[
(\vec{l}, C, Q) \rightarrow (\vec{l}', C[x'/x] \cup \{y \leq x, x \leq z\}, Q)
\]
How to deal with the new variables?

- **ConEquiv**

\[
C \equiv_{\text{Var}_P \cup \text{Var}_Q} D
\]

\[
(\vec{l}, C, Q) \leadsto (\vec{l}, D, Q)
\]

- **ConSpecialize**

\[
x \in \text{Var}_H \quad u \in [x]_C
\]

\[
(\vec{l}, C, Q) \leadsto (\vec{l}, C[u/x], Q)
\]

When shall we specialize to an exact value?

- **Never →** minimal number of states.
- **Always →** minimal size for every state.
- **After receiving →** might be a good compromise.
Equivalence

**Soundness**
Every state represented by a reachable state of the *semantics with constraints* is reachable in the *standard semantics*.

**Completeness**
Every state reachable in the *standard semantics* is represented by a reachable state of the *semantics with constraints*. 
Checking Arbitrary LTL Formulae

How to decide whether states are equal

1. reduce to same variables
2. test for sub-constraints

Problem

We enumerate the right states, but add spurious transitions.

Solutions

- we do not have false positives, that is enough
- test for equality, not for sub-constraints
- use bounded model checking
Comparison: Time

- Semantics with valuations
- Semantics with constraints
- Spin

Graph showing time in seconds (x-axis) versus number of possible elements in queue (y-axis) for different scenarios.
Comparison: No. States

\[ \text{#states} \quad K(\text{#possible elements in queue}) \]
Conclusion

- Late choice helps to reduce the number of orderings in a queue
- Constraints can be used to encode different states by one
- Checking state properties works
- Checking arbitrary LTL-formulae is not so easy