A Case Study in Parallel Verification of Component-Based Systems

N. Beneš, I. Černá, J. Sochor, P. Vařeková and B. Zimmerova

Faculty of Informatics, Masaryk University
Brno, Czech Republic

PDMC’08

March 29, 2008
1 Introduction
   Motivation
   Contribution

2 Foundations
   CoCoME modelling contest
   Modelling language
   Logic for properties specification

3 Verification
   Model of the system
   Verification of the model
   Overview of the results

4 Conclusion
   Summary of the talk
1 Introduction
   Motivation
   Contribution

2 Foundations
   CoCoME modelling contest
   Modelling language
   Logic for properties specification

3 Verification
   Model of the system
   Verification of the model
   Overview of the results

4 Conclusion
   Summary of the talk
Component-based development as a current trend in SE

- Systems assembled from third-party components
  → issue of correct interaction among components

Parallel verification becomes a need

- Concurrency of a large number of components
  → applicability of sequential verification techniques becomes challenging
Contribution

1 Identification of the **component-specific properties** defining correctness issues in component-based systems, and their formalization

- expressed in the logic CI-LTL – an extended version of the action-based Linear Temporal Logic
- check the **validity of the model** and the **correctness of the system**

2 Experimental **application of parallel model checking** to the model of a real system

- CoCoME modelling example
- parallel model-checking tool DiVinE
1 Introduction
   Motivation
   Contribution

2 Foundations
   CoCoME modelling contest
   Modelling language
   Logic for properties specification

3 Verification
   Model of the system
   Verification of the model
   Overview of the results

4 Conclusion
   Summary of the talk
CoCoME modelling contest

The contest

- **The aim** to evaluate and compare the existing component modelling approaches on a common modelling example
- 18/13 modelling teams, URL http://www.cocome.org
  Organized by A. Rausch, R. Reussner, R. Mirandola, F. Plasil

The Common Component Modelling Example

- A trading system handling sales in a chain of supermarkets
  - interaction with the cashier, inventory, ordering goods, generating reports
- Specification in terms of Java source code (125 classes)
  - to prevent ambiguities in the interpretation of the specification by the teams
Common Component Modelling Example
Component-Interaction automata language (or CI automata for short)

- Automata-based language
  - finite state model, infinite executions/traces

- Three types of actions
  - input, output and internal

- Captures important interaction information
  - participants of communication, hierarchy of components

- Flexible composition
  - can be parametrized by architectural assembly, communication strategy

- General to meet various component models
  - by fixing the composition operator and semantics of actions
A component-Interaction automaton

- Finite set of states
- Labeled transitions with structured labels (component names, actions)
- Hierarchy of component names

Example:
Composition of CI automata

\[ C = \bigotimes F \{ C_1, C_2, C_3 \} \]
Logic for properties specification

CI-LTL

- extended version of the action based LTL
- motivated by CBSE requirements
- expresses properties about both
  - occurring component interaction (i.e. labels in automata)
  - possible component interaction (i.e. label enabledness)

Operators

- possible interaction (in a state) \( E(l) \)
- actual interaction (on a path) \( P(l) \)
- LTL operators – next \( X \Phi \), until \( \Phi U \Psi \), and \( \Phi \land \Psi \), not \( \neg \Phi \)

Example \( G (E(1, a, 2) \Rightarrow F P(1, a, 2)) \)
1 Introduction
   Motivation
   Contribution

2 Foundations
   CoCoME modelling contest
   Modelling language
   Logic for properties specification

3 Verification
   Model of the system
   Verification of the model
   Overview of the results

4 Conclusion
   Summary of the talk
Model of the system

Structure of the model

- 140 primitive CI automata,
- composed hierarchically – 34 composite automata, 6 levels
- complemented with a usage profile of the Cashier

State space of the model

- Without usage profile
  - over 322 millions of states (60 GB of memory)
- With usage profile
  - 749,340 reachable states, 3,181,473 reachable transitions
Correctness of the system

- Local deadlocks of components
  - one of the components cannot move further

- Blocking of components
  - temporary blocking of a component

- Infinite loops in the model
  - finite for/while cycles traversed infinitely-many times

- Use-case scenarios
  - presence of a particular scenario (sequence of actions) in the system
Validity of the model

- checking safety of considered abstractions
- simplification of the communicational scheme
- internal behaviour of primitive components
- checked via a number of test cases translated into CI-LTL
Property 4 (Local deadlock on one action). It cannot happen that the *CashDeskApplication* (100) is ready to send a notification to the *CashDeskChannel* (200) saying that it received the *SaleStartedEvent*, but the *CashDeskChannel* is never ready to accept the notification.

\[
[F \neg P(100, oESS'', -)] \lor G[E(100, oESS'', -) \Rightarrow F E(100, oESS'', 200)]
\]

<table>
<thead>
<tr>
<th>property</th>
<th>states</th>
<th>transitions</th>
<th>memory</th>
<th>time</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>prop4</td>
<td>749 343</td>
<td>3 181 479</td>
<td>532 MB</td>
<td>67 s</td>
<td>holds</td>
</tr>
</tbody>
</table>
Property 7 (Blocking of a component). It cannot happen that the *CashDeskApplication* (100) is ready to send a notification to the *CashDeskChannel* (200) saying that it received the *SaleStartedEvent*, but the *CashDeskChannel* is not right in the current state ready to accept the notification.

\[ \text{FP}(100, \text{oESS}''', -)) \lor \neg \text{E}(100, \text{oESS}''', -) \land \neg \text{E}(100, \text{oESS}''', 200) \]

<table>
<thead>
<tr>
<th>property</th>
<th>states</th>
<th>transitions</th>
<th>memory</th>
<th>time</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>prop7</td>
<td>1 498 671</td>
<td>6 362 935</td>
<td>688 MB</td>
<td>534 s</td>
<td>holds not</td>
</tr>
</tbody>
</table>
Use Case scenarios

1. Cash Payment
2. Unsuccessful Card Payment
3. Successful Card Payment

<table>
<thead>
<tr>
<th>property</th>
<th>states</th>
<th>transitions</th>
<th>memory</th>
<th>time</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>uc1</td>
<td>19 362 460</td>
<td>81 959 821</td>
<td>4 204 MB</td>
<td>5 141 s</td>
<td>found</td>
</tr>
<tr>
<td>uc2</td>
<td>11 670 924</td>
<td>49 165 124</td>
<td>2 694 MB</td>
<td>3 203 s</td>
<td>found</td>
</tr>
<tr>
<td>uc3</td>
<td>11 680 736</td>
<td>49 202 320</td>
<td>2 698 MB</td>
<td>3 098 s</td>
<td>found</td>
</tr>
</tbody>
</table>
Effect of the parallelization

Memory consumption depending on the number of computers

![Graph showing memory consumption vs. number of computers for different properties and scenarios.]

- Property 4
- Property 7
- UC scenario 1
- UC scenario 2
Effect of the parallelization

**Time consumption** depending on the number of computers

![Graph showing time consumption depending on the number of computers]

- Property 4
- Property 7
- UC scenario 1
- UC scenario 2
Effect of the parallelization

**Time consumption** depending on the number of computers

![Graph showing time consumption vs. number of computers](image-url)
Lessons learned

Experience and discussion

- Characteristics of the model
  - state space explosion, irregular changes

- Deadlocks in the model
  - unrealistic behaviour

- Local deadlocks and component-blocking properties
  - enabledness operator

- Parallelization
  - effect of parallelization
1 Introduction
   Motivation
   Contribution

2 Foundations
   CoCoME modelling contest
   Modelling language
   Logic for properties specification

3 Verification
   Model of the system
   Verification of the model
   Overview of the results

4 Conclusion
   Summary of the talk
Conclusion

Summary of the talk

- Practical application of the parallel verification to a real component-based system
- Identification of a number of typical component specific properties
- Formalization of the properties using the logic CI-LTL
- Demonstration of the parallel CI-LTL verification

Future work

- Reduction techniques
  - the partial-order and the symmetry reduction
Thank you for your attention