Verification Tools for Mobile Processes

Björn Victor
Bjorn.Victor@it.uu.se

Uppsala University
Overview

1. Implementations
   - What has been implemented, and where?
   - The Mobility Workbench - how does it work
   - The MIHDA tool - how does it work
   - The Profundis Web-based Verification Environment
Overview

1. Implementations
   • What has been implemented, and where?
   • The Mobility Workbench - how does it work
   • The MIHDA tool - how does it work
   • The Profundis Web-based Verification Environment

2. Exercises for the course
   • extensions of tools
   • use tools
   • study and present chapters of Sangiorgi & Walker
Implementations

Not many tools available:

- What has been implemented, and where?
- The Mobility Workbench
- The MIHDA tool
- The HAL toolset
- The Profundis Web-based Verification Environment
What has been implemented?

Analysis methods which have been implemented:

- Equivalence checking:
  open and early bisimulation (MWB and MIHDA)
What has been implemented?

Analysis methods which have been implemented:

- Equivalence checking:
  open and early bisimulation (MWB and MIHDA)

- Model checking:
  MWB, HAL, XMC, three-four different logics
What has been implemented?

Analysis methods which have been implemented:

- Equivalence checking:
  open and early bisimulation (MWB and MIHDA)
- Model checking:
  MWB, HAL, XMC, three-four different logics
- Sort inference:
  MWB, “experimental” state
What has been implemented?

Analysis methods which have been implemented:

- Equivalence checking:
  open and early bisimulation (MWB and MIHDA)

- Model checking:
  MWB, HAL, XMC, three-four different logics

- Sort inference:
  MWB, “experimental” state

- Minimization:
  MIHDA, indirect
What has been implemented?

Analysis methods which have been implemented:

- **Equivalence checking:**
  open and early bisimulation (MWB and MIHDA)

- **Model checking:**
  MWB, HAL, XMC, three-four different logics

- **Sort inference:**
  MWB, “experimental” state

- **Minimization:**
  MIHDA, indirect

- **Simulation (“execution”), deadlock detection:**
  MWB
The Mobility Workbench

- Developed 1993-1999 at Uppsala University in collaboration with Edinburgh University and the Swedish Institute of Computer Science. (*Faron Moller, Davide Sangiorgi, Mads Dam, Lars-Henrik Eriksson, Fredrick Beste...*)
- Interactive, similar to CWB
- Open bisimulation equivalence, model checking, sort inference, simulation, deadlock detection, fusion hyperequivalence...
- Used for research and education at > 30 sites.
- On-the-fly algorithms: generate state space while constructing bisimulation relation or deriving proof.
MWB Internals

- Exploits symbolic semantics: no universal quantification
- On-the-fly generation of state space
  - can give negative results even on infinite-state (non finite-control)
  - “dynamic programming”: save computed transition derivations
- Name representation using De Bruijn indices: avoids alpha conversion
- Written in SML/NJ, lots of modules (one by Joachim!)
MWB Efficiency

- Classical GSM handover example (17-page hand proof by Orava & Parrow)
  1993: on SS10 ~100MHz? > 230 CPU hrs (didn’t terminate)
  1995: on SS10: < 11 minutes, 295 Mb memory
  2003: on P4 2GHz: < 10 seconds, 36 Mb memory

- Simplified GSM handover example (also used in MIHDA demo):
  on P4 2GHz: ~ 2.4 seconds, 22 Mb memory.
MWB Terms

- Recursion, not replication ($! P \equiv P \mid ! P$)
- Match: $[x = y]P$
- Abstractions and concretions:

  \[
  a(x_1, \ldots, x_n).P = a.(\lambda x_1) \cdots (\lambda x_n)P \\
  \overline{a}\langle x_1, \ldots, x_n \rangle . P = \overline{a}.[x_1] \cdots [x_n]P \\
  P(x_1, \ldots, x_n) \overset{\text{def}}{=} A = P \overset{\text{def}}{=} (\lambda x_1) \cdots (\lambda x_n)A
  \]

Communication and application the same:

- $a\ x \cdot P \mid \overline{a} \ z \cdot Q \xrightarrow{\tau} (\lambda x)P@[y]Q \equiv P\{y/x\} \mid Q$
- $P\langle y \rangle \equiv (\lambda x)A@[y]0 \equiv A\{y/x\}$ where $P(x) \overset{\text{def}}{=} A$
The MIHDA tool

- Developed in Pisa (Emilio Tuosto, Roberto Raggi)
- Translates $\pi$-calculus terms to History Dependent Automata with Symmetries (HDS)
- Minimizes HDS with respect to early bisimulation
- Uses partition refinement algorithm: create state-space, split in equivalence classes until fixpoint reached
- Equivalence test: same minimal HDS
- No way back to $\pi$-calculus yet (minimal $\pi$ agent wanted)
The HAL toolset

- Developed in Pisa (*Stefania Gnesi, Marco Pistore,...*)
- Translates $\pi$-calculus terms to HDS, which are translated to standard automata
- Standard automata analysed by standard finite-state tools (partition refinement etc)
- Model checking, early equivalence
- Problem of mapping back to $\pi$-calculus
The PWEB

**The PROFUNDIS Web-based Verification Environment**

- Distributed architecture, XML-based communication: less monolithic tools
- Distribution of tools: more accessible
- Combine functionalities at coordination level: instead of incorporating functionality of others
Exercises

Extensions of MWB (or other tools): theory-based programming

- replication operator (not using $! P \equiv P | ! P$)
- late bisimulation (in MWB)
- dynamic binding and blocking operator (Vivas’ thesis)
- sorted/typed equivalence
- late minimization (in MIHDA)
- HD-to-$\pi$ conversion
- ...

Verification Tools for Mobile Processes – p.12/13
Exercises

Use the MWB/PWEB to verify models, e.g:
• Encode spi-calculus in \( \pi \), verify toy examples
• …

Read a chapter in Sangiorgi & Walker, present at a seminar, e.g:
• Asynchronous \( \pi \)
• Typed \( \pi \)-calculi
• Higher-order \( \pi \)-calculus
• Objects in \( \pi \)
• …