Mu-Calculus Property-Dependent Reductions for Divergence-Sensitive Branching Bisimilarity

Radu Mateescu\textsuperscript{1} and Anton Wijs\textsuperscript{2}

\textsuperscript{1}Inria Grenoble – Rhône-Alpes / Convecs
\textsuperscript{2}Eindhoven University of Technology
Context

- **Concurrent systems**
  - Process algebraic languages (LNT)
  - Value-passing communication
  - Interleaving semantics, action-based setting (LTSs)
  - Equivalence relations (e.g., bisimulations)
  - Branching-time temporal logics (e.g., $\mu$-calculus)

- **Explicit-state verification**
  - Enumeration of individual states and transitions
  - Forward and backward exploration
  - Diagnostic generation

- **CADP toolbox:** [http://cadp.inria.fr](http://cadp.inria.fr)
Labeled Transition Systems

\[ M = (S, A, T, s_0) \]

- Two-place FIFO lossy buffer
- Stream of 0/1 messages

WS-FMDS 2014
Behavioural specification

compilation

implicit LTS

on-the-fly verification

verdict & diagnostic

explicit LTS

compilation

global verification

verdict & diagnostic

minimisation
(modulo a relation preserving the formula)
Adequacy of Temporal Logics with Equivalence Relations

Logic \( L \) is \textit{adequate} with equivalence relation \( \approx_R \) iff for any LTSs \( M_1, M_2 \) and formula \( \varphi \) of \( L \):

\[
M_1 \approx_R M_2 \iff (M_1 \models \varphi \iff M_2 \models \varphi)
\]

Examples of adequacy results:

<table>
<thead>
<tr>
<th>Temporal logic</th>
<th>Equivalence relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>modal ( \mu )-calculus (( L_\mu ))</td>
<td>strong bisimulation</td>
</tr>
<tr>
<td>ACTL ( \setminus X )</td>
<td>divergence-sensitive branching bisimulation</td>
</tr>
<tr>
<td>weak ( L_\mu )</td>
<td>weak bisimulation</td>
</tr>
<tr>
<td>selective ( L_\mu )</td>
<td>( \tau^*.a ) bisimulation</td>
</tr>
<tr>
<td>BSL</td>
<td>safety equivalence</td>
</tr>
</tbody>
</table>
Using Adequacy to Improve Model Checking

Theoretical interest:

- Reason using either logic, or equivalence
  • Characteristic formulas for equivalences

Practical interest:

- Reduce the LTS modulo $\equiv_R$ before checking $\varphi$
  • Improve verification performance for complex formulas
  • Reduce once, then check several formulas of $L$
  • If $\approx_R$ is a congruence for $||$, use compositional LTS generation

→ Objective: improve this approach further by specializing it for a given formula
Model Checking Language
(dataless fragment)

Action formulas:
\[ \alpha ::= \text{false} \mid \tau \mid a \mid \neg \alpha \mid \alpha_1 \lor \alpha_2 \]

Regular formulas:
\[ \beta ::= \alpha \mid \beta_1 \cdot \beta_2 \mid \beta_1 | \beta_2 \mid \beta^* \]

State formulas:
\[ \varphi ::= \text{false} \mid \neg \varphi \mid \varphi_1 \lor \varphi_2 \]
\[ \mid < \beta > \varphi \mid [ \beta ] \varphi \mid \\
\mid < \beta > @ \mid [ \beta ] - \mid \\
\mid Y \mid \mu Y . \varphi \mid \nu Y . \varphi \]
Property-Dependent Reduction

[Mateescu-Wijs-14]

**Input:** LTS $M = (S, A, T, s_0)$ and $L\mu$ formula $\varphi$

**Step 1:** Maximal hiding modulo $\varphi$

- Determine $h(\varphi) =$ set of actions that can be hidden in $M$ without changing the interpretation of $\varphi$ on $M$
- Hide $h(\varphi)$ in $M$

**Step 2:** Reduction of $M$ preserving $\varphi$

- strong bisimulation: full $L\mu$
- ds-branching bisimulation: $L\mu$-dsbr fragment

**Step 3:** Verification of $\varphi$ on reduced $M$
Lossy Buffer
(hide “PUT !1” and “GET !1”)

Minimized modulo strong bisimulation:
Lossy Buffer
(hide “PUT !1” and “GET !1”)

Minimized modulo ds-branching bisimulation:
Formula $\varphi_1$
(nested regular modalities – response)

$[\text{true}^* \cdot \text{"PUT !0"}] < \text{true}^* \cdot \text{"GET !0"} > \text{true}$

Witness in LTS minimized modulo $\approx_{\text{dsbr}}$: 
Formula $\varphi_2$
(fairness operators – cycle)

$\diamond < \text{true}^* \cdot "\text{PUT !0"} \cdot \text{true}^* \cdot "\text{GET !0"} > @$

Witness in LTS minimized modulo $\approx_{\text{dsbr}}$:

![Diagram](image-url)
Formula $\varphi_3$
(fixed point operators - inevitability)

\[ \text{true}^* \ . \ "\text{PUT !0}" \]
\[
\mu Y . (\langle \text{true} \rangle \text{true} \text{and} [\text{not } \"\text{GET !0}\"] Y)
\]

Counterexample in LTS minimized modulo $\cong_{\text{dsbr}}$:
What Actions Can I Hide for my Formula?

\[
\begin{align*}
\text{Rule of Thumb \#1: For an } L\mu \text{ formula } \varphi \text{ without occurrences of } \tau, \text{ hide all actions but those occurring in } \varphi.
\end{align*}
\]
What about Constant Action Formulas?

\[ [\text{true}] \langle \text{true} \rangle > \text{true} \]

Actions formulas “false” can be eliminated:

\[ < \text{false} \rangle \varphi = \text{false} \]
\[ [\text{false}] \varphi = \text{true} \]

Rule of Thumb #2: For an Lμ formula \( \varphi \) with only constant action formulas, hide all actions.
Lμ-dsbr Fragment

[Mateescu-Wijs-14]

Replace strong modalities of Lμ with:

- \(< \alpha_1^* > \varphi > ultra weak modality\)
- \(< \alpha_1^*. \alpha_2 > \varphi > weak modality\)
- \(< \alpha_1 > @ > weak infinite looping\)

where \(\alpha_1 \vdash \tau\) and \(\alpha_2 \nvdash \tau\)

Lμ-dsbr is adequate with \(\cong_{dsbr}\)
Formulas $\varphi_1$, $\varphi_2$, $\varphi_3$ Revisited

\[ [ \text{true}^* \cdot "\text{PUT} \, !0" ] < [ \text{true}^* \cdot "\text{GET} \, !0" ] > \text{true} \]

\[ < [ \text{true}^* \cdot "\text{PUT} \, !0" ] \cdot \text{true}^* \cdot "\text{GET} \, !0" ] > @ \]

\[ = \mu Y \cdot < [ \text{true}^* \cdot "\text{PUT} \, !0" ] > < [ \text{true}^* \cdot "\text{GET} \, !0" ] > Y \]

\[ [ \text{true}^* \cdot "\text{PUT} \, !0" ] \]

\[ \nu Y \cdot ( < \text{true} > \text{true} \text{ and } [ \text{not} "\text{GET} \, !0" ] \, Y ) \]

\[ = [ \text{true}^* \cdot "\text{PUT} \, !0" ] \]

\[ ([ (\text{not} "\text{GET} \, !0")^* ] \, \text{not deadlock} \]

\[ \text{and } [ \text{not} "\text{GET} \, !0" ] \, -| ) \]

\[ \text{deadlock} = [ \text{true}^*. \text{not} \, \tau ] \, \text{false} \text{ and } [ \tau ] \, -| \]
Why I Can’t Use Strong Modalities?

Rule of Thumb #3: Any strong modality in the formula $\varphi$ must be preceded by a weak modality capturing a sequence of 0 or more $\tau$-transitions.
ACTL (Action-Based CTL)

[DeNicola-Vaandrager-92]

\[ E \left[ \varphi_{1\alpha_1} \cup \varphi_2 \right] \]

\[ A \left[ \varphi_{1\alpha_1} \cup \varphi_2 \right] \]

\[ \alpha_1 \models \tau \]

\[ \alpha_2 \not\models \tau \]
\textbf{Lμ-dsbr and μ-ACTL\textbackslash X}

\begin{itemize}
  \item \textbf{μ-ACTL [Fantechi-Gnesi-Ristori-94]}
  \begin{itemize}
    \item Extension of ACTL with fixed point operators
    \item Adequate with \textit{strong} bisimulation
  \end{itemize}

  \item \textbf{Lμ-dsbr is equally expressive to μ-ACTL\textbackslash X}
  \begin{align*}
    <\alpha_1^*\!>\varphi &= E\left[\text{true}_{\alpha_1}\cup\varphi\right]
    \\
    <\alpha_1^*\cdot\alpha_2\!>\varphi &= E\left[\text{true}_{\alpha_1}\cup_{\alpha_2}\varphi\right]
    \\
    <\alpha_1\!>@ &= \nu Y . E\left[\text{true}_{false}\cup_{\alpha_1}Y\right]
  \end{align*}

  \item \textbf{Lμ-dsbr adequate with} $\simeq_{\text{dsbr}}$
  \textbf{μ-ACTL\textbackslash X adequate with} $\simeq_{\text{dsbr}}$
\end{itemize}
Lμ-dsbr and Selective Lμ

Selective Lμ [Barbuti-et-al-96]
- Special modalities indexed by sets of visible actions
- For a formula ϕ, hide all actions but those in ϕ
- Minimize the LTS modulo τ*.a
- Selective Lμ equally expressive to Lμ
  \[ but reductions only when hiding is possible!

Selective Lμ modalities translated in Lμ-dsbr
\[< \alpha_1 >_{\alpha_2} \phi = < (\neg(\alpha_1 \lor \alpha_2))^* \cdot \alpha_1 > \phi \]
when \( \alpha_1 \lor \alpha_2 \neq \text{true} \)
Lμ-dsbr and Selective Lμ

Advantages of Lμ-dsbr w.r.t. selective Lμ:

– Allows one to use τ in action formulas (more flexible)

– Adequate with ≈_dsbr
  • Stronger than τ*.a bisimulation (captures deadlocks and livelocks)
  • Suitable for compositional LTS construction (≈_dsbr is a congruence w.r.t. parallel composition, whereas τ*.a not)

– Lμ-dsbr subsumes the interesting fragment of selective Lμ (formulas which make hiding possible)
**Lµ-dsbr and Weak Lµ**

- **Weak Lµ [Stirling-01]**
  - Lµ fragment adequate with weak bisimulation
  - Weak modalities, no τ actions in formulas
  - Does not express inevitability properties

- Weak Lµ modalities translated in Lµ-dsbr

\[ \lll \alpha \rrr \varphi = < \tau^* . \alpha > < \tau^* > \varphi \]

\[ \llll \varphi = < \tau^* > \varphi \]

- Weak modalities (over regular formulas) are directly available in MCL
Operators Adequate with $\approx_{\text{dsbr}}$

**$\mu$ fixed point operators**

- $\mu Y . \varphi$
- $\nu Y . \varphi$

**ACTL/$\mathcal{X}$ operators**

- $E [\varphi_1 U_{\alpha_1} \varphi_2]$
- $A [\varphi_1 U_{\alpha_1} \varphi_2]$
- $E [\varphi_1 U_{\alpha_1} U_{\alpha_2} \varphi_2]$
- $A [\varphi_1 U_{\alpha_1} U_{\alpha_2} \varphi_2]$

**$L\mu$-dsbr modalities**

- $< \alpha_1^* > \varphi$
- $< \alpha_1^* . \alpha_2 > \varphi$
- $< \alpha_1 > @$
- $[ \alpha_1^* ] \varphi$
- $[ \alpha_1^* . \alpha_2 ] \varphi$
- $[ \alpha_1 ] - |$

**Boolean operators**

- true
- false
- not
- or
- and

WS-FMDS 2014
Experimental Results
(strong bisimulation minimization)

- Alternating Bit Protocol
- Characteristic property: maximal hiding $\Rightarrow$ strong bisimulation minimization using BCG_MIN $\Rightarrow$ model checking using EVALUATOR 4.0
Experimental Results
(on-the-fly $\tau$-confluence reduction)

- Erathosthene’s sieve
- Characteristic property: maximal hiding $\Rightarrow$ on-the-fly $\tau$-confluence reduction $\Rightarrow$ model checking using EVALUATOR 4.0
Experimental Results
(ds-branching bisimulation reduction)

- DTD (Dynamic Task Dispatcher) [Lantreibeck-Serwe-13]
- Property P_2: maximal hiding \(\Rightarrow\) reduction modulo \(\approx_{\text{dsbr}}\)
  using BCG_MIN \(\Rightarrow\) model checking using EVALUATOR 4.0
Ongoing and Future Work

- Develop an MCL library containing all operators adequate with $\approx_{dsbr}$

- Automate maximal hiding:
  - Option `-labels` of EVALUATOR 4.0 (rules of thumb 1+2)
  - Extend to the general case

- Automate adequacy detection:
  - Determine the weakest equivalence relation adequate with an MCL formula (e.g., rule of thumb 3)
  - Integrate within SVL

- Handle MCL formulas with data
Thank you

For more information:
http://cadp.inria.fr