Introduction to MCRL2 (verification of process properties)

Luís Soares Barbosa

Universidade do Minho

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

UNITED NATIONS UNIVERSITY
UNU-EGOV

Interaction & Concurrency Course Unit (Lcc)

Universidade do Minho, 29.IV.2019

Overview

The verification problem

- Given a specification of the system's behaviour is in MCRL2
- and the system's requirements are specified as properties in a temporal logic,
- a model checking algorithm decides whether the property holds for the model: the property can be verified or refuted;
- sometimes, witnesses or counter examples can be provided

Which logic?

μ-calculus with data, time and regular expressions

From modal logic ...

Hennessy-Milner logic

... propositional logic with action modalities

$$\phi \ ::= \ true \mid false \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi$$

Laws

$$\neg\langle a\rangle\phi = [a]\neg\phi$$
$$\neg[a]\phi = \langle a\rangle\neg\phi$$
$$\langle a\rangle false = false$$
$$[a] true = true$$
$$\langle a\rangle(\phi \vee \psi) = \langle a\rangle\phi \vee \langle a\rangle\psi$$
$$[a](\phi \wedge \psi) = [a]\phi \wedge [a]\psi$$
$$\langle a\rangle\phi \wedge [a]\psi \Rightarrow \langle a\rangle(\phi \wedge \psi)$$

From modal logic ...

Hennessy-Milner logic + regular expressions

ie, with regular expressions within modalities

$$\rho ::= \epsilon \mid \alpha \mid \rho.\rho \mid \rho + \rho \mid \rho^* \mid \rho^+$$

where

- α is an action formula and ϵ is the empty word
- concatenation $\rho.\rho$, choice $\rho + \rho$ and closures ρ^* and ρ^+

Laws

$$\langle\rho_1 + \rho_2\rangle\phi = \langle\rho_1\rangle\phi \vee \langle\rho_2\rangle\phi$$
$$[\rho_1 + \rho_2]\phi = [\rho_1]\phi \wedge [\rho_2]\phi$$
$$\langle\rho_1.\rho_2\rangle\phi = \langle\rho_1\rangle\langle\rho_2\rangle\phi$$
$$[\rho_1.\rho_2]\phi = [\rho_1][\rho_2]\phi$$

From modal logic ...

Action formulas

$$\alpha \ ::= a_1 \mid \cdots \mid a_n \ \mid \ \mathit{true} \ \mid \ \mathit{false} \ \mid \ -\alpha \ \mid \ \alpha \cup \alpha \ \mid \ \alpha \cap \alpha$$

where

- $a_1 \mid \cdots \mid a_n$ is a set with this single multiaction
- *true* (universe), *false* (empty set)
- $-\alpha$ is the set complement

Modalities with action formulas:

$$\langle\alpha\rangle\phi \ = \ \bigvee_{a\in\alpha} \langle a\rangle\phi \qquad [\alpha]\phi \ = \ \bigwedge_{a\in\alpha} [a]\phi$$

... to temporal logic

Examples of properties

- $\langle \epsilon \rangle \phi \ = \ [\epsilon]\phi \ = \ \phi$
- $\langle a.a.b \rangle \phi \ = \ \langle a \rangle \langle a \rangle \langle b \rangle \phi$
- $\langle a.b + g.d \rangle \phi$

Safety

- $[true^*]\phi$
- it is impossible to do two consecutive enter actions without a leave action in between:
  $[true^*.enter. - leave^*.enter]false$
- absence of deadlock:
  $[true^*]\langle true \rangle true$

... to temporal logic

Examples of properties

Liveness

- $\langle \mathit{true}^* \rangle \phi$
- after sending a message, it can eventually be received: $[\mathit{send}]\langle \mathit{true}^*.\mathit{receive}\rangle \mathit{true}$
- after a send a receive is possible as long as it has not happened: $[\mathit{send}.-\mathit{receive}^*]\langle \mathit{true}^*.\mathit{receive}\rangle \mathit{true}$

… to temporal logic

The modal μ-calculus

- modalities with regular expressions are not enough in general
- … but correspond to a subset of the modal μ-calculus [Kozen83]

Add explicit minimal/maximal fixed point operators to Hennessy- Milner logic

$$\phi \ ::= \ X \mid \mathit{true} \mid \mathit{false} \mid \neg\phi \mid \phi\wedge\phi \mid \phi\vee\phi \mid \phi\Rightarrow\phi \mid \langle a\rangle\phi \mid [a]\phi \mid \mu X\,.\,\phi \mid \nu X\,.\,\phi$$

... to temporal logic

The modal μ-calculus (intuition)

- $\mu X \,.\, \phi$ is valid for all those states in the smallest set X that satisfies the equation $X = \phi$ (finite paths, liveness)
- $\nu X \,.\, \phi$ is valid for the states in the largest set X that satisfies the equation $X = \phi$ (infinite paths, safety)

Warning

In order to be sure that a fixed point exists, X must occur positively in the formula, ie preceded by an even number of negations.

... to temporal logic

Laws & Notes (but see the μ-calculus slides!)

$$\mu X\,.\,\phi \;\Rightarrow\; \nu X\,.\,\phi$$

and self-duals:

$$\neg\mu X\,.\,\phi \;=\; \nu X\,.\,\neg\phi$$
$$\neg\nu X\,.\,\phi \;=\; \mu X\,.\,\neg\phi$$

Translation of regular formulas with closure

$$\langle R^*\rangle\phi \;=\; \mu X\,.\,\langle R\rangle X \vee \phi$$
$$[R^*]\phi \;=\; \nu X\,.\,[R]X \wedge \phi$$
$$\langle R^+\rangle\phi \;=\; \langle R\rangle\langle R^*\rangle\phi$$
$$[R^+]\phi \;=\; [R][R^*]\phi$$

Example: The dining philosophers problem

Formulas to verify Demo

- No deadlock (every philosopher holds a left fork and waits for a right fork (or vice versa):

```
[true*]<true>true
```

- No starvation (a philosopher cannot acquire 2 forks):

```
forall p:Phil. [true*.!eat(p)*] <!eat(p)*.eat(p)>true
```

- A philosopher can only eat for a finite consecutive amount of time:

```
forall p:Phil. nu X. mu Y. [eat(p)]Y && [!eat(p)]X
```

- there is no starvation: for all reachable states it should be possible to eventually perform an `eat(p)` for each possible value of `p:Phil`.

```
[true*](forall p:Phil. mu Y. ([!eat(p)]Y && <true>true))
```