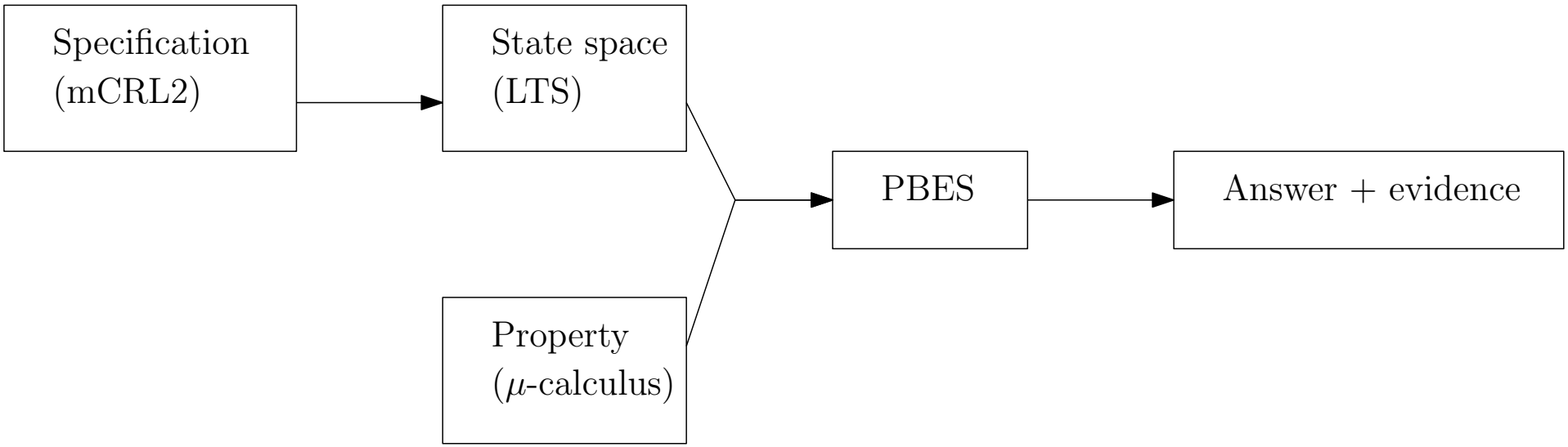
Fast Higher-order Term Rewriting for Model Checking Purposes

Rick Erkens

Eindhoven University of Technology
The Netherlands

July 1st 2019

Model checking with the mCRL2 toolset



State space explosion

- LTS could contain many, many states
- A powerful rewriter is needed
- As of now, the choice of rewriter matters a lot performance-wise

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms?

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

Types?

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

Types? Simple.
$\tau ::= \tau_{basic} \mid \tau \times \cdots \times \tau \to \tau$

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

Types? Simple.
$\tau ::= \tau_{basic} \mid \tau \times \cdots \times \tau \to \tau$

Rewrite rules?

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

Types? Simple.
$\tau ::= \tau_{basic} \mid \tau \times \cdots \times \tau \to \tau$

Rewrite rules? Constraint rewrite rules.
$c : \ell \to r$ with c of type $\mathbb{B}$ and ℓ first-order pattern.

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

Types? Simple.
$\tau ::= \tau_{basic} \mid \tau \times \cdots \times \tau \to \tau$

Rewrite rules? Constraint rewrite rules.
$c : \ell \to r$ with c of type $\mathbb{B}$ and ℓ first-order pattern.

Rewrite step?

The mCRL2 language

mCRL2: Process algebra with higher-order terms as data patameters.
Terms? Higher-order.
Well typed terms of the grammar $t ::= x \mid f \mid t(\vec{t}) \mid \lambda \vec{x}.t$

Types? Simple.
$\tau ::= \tau_{basic} \mid \tau \times \cdots \times \tau \to \tau$

Rewrite rules? Constraint rewrite rules.
$c : \ell \to r$ with c of type $\mathbb{B}$ and ℓ first-order pattern.

Rewrite step? Application of a rule + β-reduction, both closed under contexts.
$C[t] \to_R C[r^\sigma]$ if there exists a substitution σ and a rewrite rule $c : \ell \to r$ such that $t = \ell^\sigma$ and $c^\sigma \to_R^* \text{true}$.
$C[(\lambda \vec{x}.t)(\vec{t})] \to_R C[t[\vec{x} := \vec{t}]]$

The goal

To describe and implement a rewriter

The goal

To describe and implement a rewriter

- that performs well,

The goal

To describe and implement a rewriter

- that performs well,
- that we understand,

The goal

To describe and implement a rewriter

- that performs well,
- that we understand,
- that is relatable to the theory of rewriting,

The goal

To describe and implement a rewriter

- that performs well,
- that we understand,
- that is relatable to the theory of rewriting,
- that has a clear documentation for users on the limitations of the formalism,

The goal

To describe and implement a rewriter

- that performs well,
- that we understand,
- that is relatable to the theory of rewriting,
- that has a clear documentation for users on the limitations of the formalism,
- that allows for optimisation.