2. Rewriting Modulo SMT

Camilo Rocha
Escuela Colombiana de Ingeniería
Joint work with J.Meseguer (UIUC) & C.Muñoz (NASA)

7th International School on Rewriting (ISR2014)
Universidad Técnica Federico Santa María
August 25-29, 2014

Motivation

An *open system* is a concurrent system that interacts with an external, non-deterministic environment and can be specified by rewrite theories of the form $\mathcal{R} = (\Sigma, E, R)$.
When an open system is specified by a rewrite theory $\mathcal{R}$, it has two sources of non-determinism:

internal
: comes from the fact that in a given system state different instances of rules in R may be enabled; these rules have the form
$t \longrightarrow t'$ with $\mathit{vars}(t') \subseteq \mathit{vars}(t)$

external
: comes from the fact that in a given system state changes to the enviroment enable instances of rules in R; these rules have the form
$t(\overrightarrow{x}) \rightarrow t'(\overrightarrow{x}, \overrightarrow{y})$ **if** ϕ, where $\overrightarrow{y}$ are fresh new variables and ϕ is a constraint

The Problem

There are non-trivial **challenges** when modeling and analyzing open systems with rules of the form $t(\overrightarrow{x}) \to t'(\overrightarrow{x}, \overrightarrow{y})$ **if** ϕ:

1. The enormous and possibly infinitary non-determinism due to the environment, which typically renders finite-state model checking impossible or unfeasible
2. The impossibility of executing the rewrite theory $\mathcal{R}$ in the standard sense, due to the non-deterministic choice of substitutions for $\overrightarrow{y}$
3. The, in general, undecidable challenge of checking the rule's condition, since without knowing all instantiations ρ of $\overrightarrow{y}$, the resulting condition $\phi\rho$ may be non-ground, so that its satisfiability may be undecidable

In This Talk (0)

Rewriting modulo SMT, a new symbolic technique that seamlessly combines the powers of rewriting modulo theories, SMT solving, and model checking.
In rewriting modulo SMT, rules for external transitions have the form $t(\overrightarrow{x}) \to t'(\overrightarrow{x}, \overrightarrow{y})$ **if** ϕ, where:

- the constraint ϕ is solvable by an SMT solver
- $\overrightarrow{y}$ has variables with typings in the STM solver
- ϕ may allow the environment to choose an infinite number of substitutions ρ for $\overrightarrow{y}$, but can exclude choices that the environment will never make

In This Talk (1)

Challenges (1)–(3) are all met successfully by rewriting modulo SMT because:

1. states are represented not as concrete states, i.e., ground terms, but as symbolic constrained terms $\langle t\,;\varphi\rangle$ with t a term with variables ranging in the domains handled by the SMT solver and φ an SMT-solvable formula, so that the choice of ρ is avoided
2. rewriting modulo SMT can symbolically rewrite such pairs $\langle t\,;\varphi\rangle$ to other pairs $\langle t'\,;\varphi'\rangle$
3. decidability of $\varphi \wedge \phi\theta$ can be settled by invoking an SMT solver

Outline

Outline

Built-in Subsignature

The notion of *built-in subsignature* (i.e., the one with symbols supported by an SMT solver) in an order-sorted signature Σ is modeled by a many-sorted signature Σ_0 defining the built-in terms $T_{\Sigma_0}(X_0)$.

Definition (Signature with Built-ins)

An order-sorted signature $\Sigma = (S, \leqslant, F)$ is a signature with *built-in subsignature* $\Sigma_0 \subseteq \Sigma$ iff $\Sigma_0 = (S_0, F_0)$ is many-sorted, S_0 is a set of minimal elements in $(S, \leqslant)$, and if $f : w \longrightarrow s \in F_1$, then $s \notin S_0$ and f has no other typing in F_0, where $F_1 = F \backslash F_0$ and $\Sigma_1 = (S, \leqslant, F_1)$.

Matching a Σ-term

Under certain restrictions on axioms, matching a Σ-term t to a Σ-term u, can be decomposed modularly into Σ_1-matching and Σ_0-matching of the built-in subterms.

Lemma

Let $\Sigma = (S, \leqslant, F)$ be a signature with built-in subsignature $\Sigma_0 = (S_0, F_0)$. Let B_0 be a set of Σ_0-axioms and B_1 a set of Σ_1-axioms. For B_0 and B_1 regular, linear, collapse free for any sort in S_0, and sort-preserving, if $t \in T_{\Sigma_1}(X_0)$ is linear with vars$(t) = \{x_1, \ldots, x_n\}$, then for each $\theta : X_0 \longrightarrow T_{\Sigma_0}(X_0)$ if $t\theta =_{B_0 \uplus B_1} t'$, then there exist $v \in T_{\Sigma_1}(X_0)$ and $\theta' : X_0 \longrightarrow T_{\Sigma_0}(X_0)$ such that $t' = v\theta'$, $t =_{B_1} v$, and $\theta =_{B_0} \theta'$ (i.e., $\theta(x) =_{B_0} \theta'(x)$ for each $x \in X_0$).

Rewriting Modulo a Built-in Subtheory

Definition

A *rewrite theory modulo the built-in subtheory* $\mathcal{E}_0$ is a topmost rewrite theory $\mathcal{R} = (\Sigma, E, R)$ with:

(a) $\Sigma{=}(S, \leqslant, F)$ a signature with built-in subsignature $\Sigma_0{=}(S_0, F_0)$ and top sort $\mathit{State}{\in}S$;

(b) $E = E_0 \uplus B_0 \uplus B_1$, where E_0 is a set of Σ_0-equations, B_0 (resp., B_1) are regular, linear, collapse free for any sort in S_0, and sort-preserving Σ_0-axioms (resp., Σ_1-axioms), $\mathcal{E}_0 = (\Sigma_0, E_0 \uplus B_0)$ and $\mathcal{E} = (\Sigma, E)$ are convergent, and the theory inclusion $\mathcal{E}_0 \subseteq \mathcal{E}$ is protecting;

(c) R is a set of *rewrite rules* of the form $l(\overrightarrow{x_1}, \overrightarrow{y}) \to r(\overrightarrow{x_2}, \overrightarrow{y})$ **if** $\phi(\overrightarrow{x_3})$ such that $l, r \in T_\Sigma(X)_{\mathit{State}}$, l is $(S \backslash S_0)$-linear, $\overrightarrow{x_i}{:}\overrightarrow{s_i}$ with $\overrightarrow{s_i} \in S_0^*$, for $i \in \{1, 2, 3\}$, $\overrightarrow{y}{:}\overrightarrow{s}$ with $\overrightarrow{s} \in (S \backslash S_0)^*$, and $\phi \in \mathit{QF}_{\Sigma_0}(X_0)$, where $\mathit{QF}_{\Sigma_0}(X_0)$ denotes the set of quantifier-free Σ_0-formulas with variables in X_0.

The Ground Rewrite Relation

The binary rewrite relation induced by a rewrite theory $\mathcal{R}$ modulo $\mathcal{E}_0$ on $T_{\Sigma,State}$ is called the *ground rewrite relation* of $\mathcal{R}$.

Definition (Ground Rewrite Relation)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory modulo $\mathcal{E}_0$. The relation $\rightarrow_{\mathcal{R}}$ induced by $\mathcal{R}$ on $T_{\Sigma,State}$ is defined for $t, u \in T_{\Sigma,State}$ by $t \rightarrow_{\mathcal{R}} u$ iff there is a rule $l \rightarrow r$ **if** ϕ in R and a ground substitution $\sigma : X \longrightarrow T_\Sigma$ such that

(a) $t =_E l\sigma$, $u =_E r\sigma$, and

(b) $\mathcal{T}_{\mathcal{E}_0} \models \phi\sigma$.

Normal Form of a Rewrite Theory

A rewrite theory $\mathcal{R}$ modulo $\mathcal{E}_0$ always has a canonical representation in which all left-hand sides of rules are S_0-linear Σ_1-terms.

Definition (Normal Form of $\mathcal{R}$ Modulo $\mathcal{E}_0$)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory modulo $\mathcal{E}_0$. Its *normal form* $\mathcal{R}^\circ = (\Sigma, E, R^\circ)$ has rules:

$$R^\circ = \{l^\circ \to r \textbf{ if } \phi \wedge \phi^\circ \mid (\exists l \to r \textbf{ if } \phi \in R)\langle \lambda \overrightarrow{x}.l^\circ\,;\theta^\circ\,;\phi^\circ\rangle = \mathit{abstract}_\Sigma(l, \mathit{vars}(\{l, r, \phi$$

where the expression $\mathit{abstract}_{\Sigma_1}(t, Y)$ denotes the choice of a triple $\langle \lambda x_1 \cdots x_n.t^\circ\,;\theta^\circ\,;\phi^\circ\rangle$ such that the context $\lambda x_1 \cdots x_n.t^\circ$ and the substitution θ° satisfy certain properties, and $\phi^\circ = \bigwedge_{i=1}^{n}(x_i = \theta^\circ(x_i))$

Lemma

$\to_{\mathcal{R}} = \to_{\mathcal{R}^\circ}$.

Matching Lemma

By the properties of the axioms in a rewrite theory modulo built-ins $\mathcal{R} = (\Sigma, E_0 \uplus B_0 \uplus B_1)$, B_1-**matching a term** $t \in T_\Sigma(X_0)$ to a left-hand side l° of a rule in R° provides a **complete unifiability algorithm** for ground B_1-unification of t and l°.

Lemma (Matching Lemma)

Let $\mathcal{R} = (\Sigma, E_0 \uplus B_0 \uplus B_1, R)$ be a rewrite theory modulo $\mathcal{E}_0$. For $t \in T_\Sigma(X_0)_{State}$ and l° a left-hand side of a rule in R° with
$vars(t) \cap vars(l^\circ) = \emptyset$,
$t \ll_{B_1} l^\circ \iff GU_{B_1}(t = l^\circ) \neq \emptyset$,
where $GU_{B_1}(t = l^\circ) = \{\sigma : X \longrightarrow T_\Sigma \mid t\sigma =_{B_1} l^\circ\sigma\}$.

Outline

Constrained Terms

Symbolic states are represented by *symbolic constrained* terms $\langle t\,;\varphi\rangle$ with t a term in $T_\Sigma(X_0)_{\mathsf{State}}$ and φ a quantifier-free formula in $\mathsf{QF}_{\Sigma_0}(X_0)$.

Definition (Constrained Terms)

Let $\mathcal{R}=(\Sigma,E,R)$ be a rewrite theory modulo $\mathcal{E}_0$. A *constrained term* is a pair $\langle t\,;\varphi\rangle$ in $T_\Sigma(X_0)_{\mathsf{State}}\times\mathsf{QF}_{\Sigma_0}(X_0)$.
Its *denotation* $[\![t]\!]_\varphi$ is defined as
$[\![t]\!]_\varphi=\{t'\in T_{\Sigma,\mathsf{State}}\mid(\exists\sigma:X_0\longrightarrow T_{\Sigma_0})\;t'=t\sigma\wedge\mathcal{T}_{\mathcal{E}_0}\models\varphi\sigma\}$.

The Symbolic Rewrite Relation

Definition (Symbolic Rewrite Relation)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory modulo built-ins $\mathcal{E}_0$. The *symbolic rewrite relation* $\rightsquigarrow_{\mathcal{R}}$ induced by $\mathcal{R}$ on $T_\Sigma(X_0)_{State} \times QF_{\Sigma_0}(X_0)$ is defined for $t, u \in T_\Sigma(X_0)_{State}$ and $\varphi, \varphi' \in QF_{\Sigma_0}(X_0)$ by $\langle t\,;\varphi\rangle \rightsquigarrow_{\mathcal{R}} \langle u\,;\varphi'\rangle$ iff there is a rule $l \to r$ **if** ϕ in R and a substitution $\theta : X \longrightarrow T_\Sigma(X)$ such that

(a) $t =_E l\zeta\theta$ and $u = r\zeta\theta$,

(b) $\mathcal{E}_0 \vdash (\varphi' \Leftrightarrow \varphi \wedge \phi\zeta\theta)$, and

(c) φ' is $\mathcal{T}_{\mathcal{E}_0}$-satisfiable,

where $\zeta = \mathit{fresh\text{-}vars}(\mathit{vars}(t, \varphi))$ and $\mathit{fresh\text{-}vars}(Y)$ represents the choice of a variable renaming $\zeta : X \longrightarrow X$ satisfying $Y \cap ran(\zeta) = \varnothing$.

Soundness and Completeness

The symbolic $\rightsquigarrow_{\mathcal{R}^\circ}$ is sound and complete w.r.t. $\rightarrow_{\mathcal{R}}$.

Theorem (Soundness)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory modulo built-ins $\mathcal{E}_0$, $t, u \in T_\Sigma(X_0)_{State}$, and $\varphi, \varphi' \in QF_{\Sigma_0}(X_0)$. If $\langle t\,;\varphi\rangle \rightsquigarrow_{\mathcal{R}^\circ} \langle u\,;\varphi'\rangle$, then $t\rho \rightarrow_{\mathcal{R}^\circ} u\rho$ for all $\rho : X_0 \longrightarrow T_{\Sigma_0}$ satisfying $\mathcal{T}_{\mathcal{E}_0} \models \varphi'\rho$.

Theorem (Completeness)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory modulo built-ins $\mathcal{E}_0$, $t \in T_\Sigma(X_0)_{State}$, $u' \in T_{\Sigma,State}$, and $\varphi \in QF_{\Sigma_0}(X_0)$. For any $\rho : X_0 \longrightarrow T_{\Sigma_0}$ such that $t\rho \in [\![t]\!]_\varphi$ and $t\rho \rightarrow_{\mathcal{R}^\circ} u'$, there exist $u \in T_\Sigma(X_0)_{State}$ and $\varphi' \in QF_{\Sigma_0}(X_0)$ such that $\langle t\,;\varphi\rangle \rightsquigarrow_{\mathcal{R}^\circ} \langle u\,;\varphi'\rangle$ and $u' \in [\![u]\!]_{\varphi'}$.

Symbolic Reachability Analysis

Theorem (Symbolic Reachability Analysis)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory modulo built-ins $\mathcal{E}_0$. The model-theoretic satisfaction relation

$$\mathcal{T}_{\mathcal{R}} \models (\exists \overrightarrow{x} \uplus \overrightarrow{y})\ t(\overrightarrow{x}) \rightarrow^* u^\circ(\overrightarrow{y}) \wedge \varphi_1(\overrightarrow{x}) \wedge \varphi_2(\overrightarrow{x}, \overrightarrow{y})$$

has a solution iff there exist a term $v \in T_\Sigma(X)_{State}$, a constraint $\varphi' \in QF_{\Sigma_0}(X_0)$, and a substitution $\theta : X \longrightarrow T_\Sigma(X)$, with $dom(\theta) \subseteq \overrightarrow{y}$, such that

(a) $\langle t\,;\varphi_1\rangle \rightsquigarrow^*_{\mathcal{R}^\circ} \langle v\,;\varphi'\rangle$

(b) $v =_{B_1} u^\circ\theta$, *and*

(c) $\varphi' \wedge \varphi_2\theta$ *is $\mathcal{T}_{\mathcal{E}_0}$-satisfiable.*

Outline

Overview (0)

- A prototype offering support for symbolic rewriting modulo SMT in the Maude system has been implemented.
- The prototype relies on Maude's meta-level features, that implement rewriting logic's reflective capabilities, and on SMT solving for $\mathcal{E}_0^+$ integrated in Maude as CVC4's decision procedures.
- The extension of Maude with CVC4 is currently an alpha feature of Maude.

Overview (1)

By using some infrastructure (explained in the paper), the parametrized module *NEXT* implements the symbolic rewrite relation $\rightsquigarrow_{\mathcal{R}^\circ}$ as a *standard rewrite relation* by extending Maude's *META-LEVEL* module , by means of the following conditional rewrite rule:

$$\begin{array}{rl} \textbf{ceq} & \langle X{:}\mathit{State}\,;\varphi{:}\mathit{QFFormula}\rangle \to \langle Y{:}\mathit{State}\,;\varphi'{:}\mathit{QFFormula}\rangle \\ \textbf{if} & \langle \overline{Y}\,;\overline{\theta}\,;\overline{\phi}\rangle\, S := \mathit{next}(\overline{\mathcal{R}^\bullet},\overline{X},\overline{\varphi}) \wedge \mathit{sat}(\varphi\wedge\phi) = \top \wedge \varphi' := \varphi\wedge\phi \end{array}$$

where $\mathcal{R}^\bullet = ((S,\leqslant,F \uplus \mathit{var}(X_0)),B,R^\circ)$.
Note that a call to an external SMT solver is just an invocation of the function *sat* in order to achieve the above functionality more efficiently and in a built-in way.

Outline

Overview of the CASH Algorithm (0)

- CASH is a scheduling algorithm, which attempts to maximize system performance while guaranteeing that critical tasks are executed in a timely manner
  - CASH poses non-trivial modeling and analysis challenges because it contains an unbounded queue.
- It was specified and analyzed in Real-Time Maude by *explicit-state model checking* in an earlier paper by Ölveczky and Caccamo
  - they showed that, under certain variations on both the assumptions and the design of the protocol, CASH could miss deadlines

Overview of the CASH Algorithm (1)

- It is parametrized by: (i) the number N of servers in the system, and (ii) the values of a maximum budget b_i and period p_i, for each server $1 \leqslant i \leqslant N$. Even if N is fixed, *there are infinitely many initial states* for N servers, since the maximum budgets b_i and periods p_i range over the natural numbers.
  - explicit state model checking cannot perform a full analysis
  - if a counterexample for N servers exists, it may be found by explicit-state model checking for some chosen initial states, as done by Ölveczky and Caccamo
  - however, a counterexample could be missed if the wrong initial states are chosen

The Case Study

- The CASH algorithm was specified and symbolically analyzed in the prototype by applying model checking based on *rewriting modulo SMT* (joint work with Kyungmin Bae)
- The goal was to verify *symbolically* the existence of missed deadlines for the *infinite set of initial configurations* containing two server objects, with maximum budgets and periods as unspecified natural numbers.
- A counterexample was found at (modeling) time two, after exploring 233 symbolic states in less than 3 seconds.
- By using a satisfiability witness of the constraint computed by the search command, a concrete counterexample is found by exploring only 54 ground states.
- This result compares favorably, in both time and computational resources, with the ground counterexample found by explicit-state model checking, where more that 52,000 concrete states were explored before finding a counterexample.

The Plan Execution Interchange Language (PLEXIL)

- It is a **synchronous** plan execution language developed by NASA for spacecraft automation
  - tasks advance in parallel
- It is designed to be flexible, efficient, and reliable in space mission operations
  - also semantically clear and deterministic
- A rewriting logic semantics have been previously developed
  - being used at NASA as a semantic standard

Uses of PLEXIL in NASA

- Mars Drill
  - executive for the Drilling Automation for Mars Exploration drilling application
  - used at the Haughton Crater on Devon Island, perhaps, in the first fully automated drill rig
- International Space Station
  - demonstrate automation for ISS operations
- Habitat Demonstration Unit
  - automated control of several subsystems

PLEXIL Overview

- A PLEXIL plan, is a tree of nodes
  - hierarchical decomposition of tasks
  - nodes have an execution status
  - leaf nodes can be local memories
- Node conditions govern the execution of a plan
  - gate: start, skip, repeat, end
  - check: pre, post, invariant
  - 42 **atomic** transitions for tasks
- Execution in PLEXIL is driven by **external events**
  - trigger changes in gate conditions
  - external values are accessed by lookups

The Problem

- Although PLEXIL is a **deterministic** language, its interaction with a physical environment through sensors and actuators make it **non-deterministic**
- For using the executable formal semantics of PLEXIL, the non-determinism of the environment had to be dealt with by **assuming** specific environment inputs
- We want to **automatically detect reachability violations** on input plans in which values of the external environment can be left **unspecified**
  - for instance, a plan is considered **invalid** if there is an execution that leads to a race condition

An Example

A Parallel Assignment with a Potential Race Condition

```
AssignWithConflict: {
  Integer       x = 0;

  NodeList: {
    NonNeg: {
      Start:      Lookup(S) >= 0;
      Assignment: x := 1;
    }

    NonPos: {
      Start:      Lookup(S) <= 0;
      Assignment: x := 2;
    }
  }
}
```

- Four nodes
  - `AssignWithConflict`
  - `NodeList`
  - `NonNeg`
  - `NonPos`
- One local memory
  - `x`
- One external variable
  - `S`

Approach

- A **symbolic** rewriting logic semantics for PLEXIL in the form of a constrained rewrite theory $\mathcal{R}$
  - external values are specified by built-in variables
  - constraints are linear integer arithmetic formulae
- $\mathcal{R}$ is **executable** in Maude
  - constraints are handled by CVC4, i.e., $\mathcal{T}_{\mathcal{E}_0}$-satisfiability is handled by decision procedures
  - use of rewriting logic's reflective capabilities

An Example (2)

State Predicate `race-free`

- The symbolic rewriting logic semantics $\mathcal{R}$ of PLEXIL provides the state predicate `race-free` that checks for race conditions on local memories
  - it can be used by Maude's LTL model checker

An Example (3)

Automatic Detection of Race Conditions

$$\mathcal{T}_{\mathcal{R}}, \langle \texttt{init} \,;\, \texttt{true} \rangle \models \Box \texttt{race-free(x.AssignWithConflict)}$$

```
reduce in ASSIGNWITHCONFLICT :
  verify-lite({init, c(true)}, [] race-free(x . AssignWithConflict)) .
rewrites: 2590 in 525ms cpu (1629ms real) (4929 rewrites/second)
result Bool: false
```

An Example (3)

Automatic Detection of Race Conditions

$$\mathcal{T}_{\mathcal{R}}, \langle \texttt{init} \,;\, \texttt{true} \rangle \models \Box \texttt{race-free(x.AssignWithConflict)}$$

```
reduce in ASSIGNWITHCONFLICT :
  verify-lite({init, c(true)}, [] race-free(x . AssignWithConflict)) .
rewrites: 2590 in 525ms cpu (1629ms real) (4929 rewrites/second)
result Bool: false
```

$$\mathcal{T}_{\mathcal{R}}, \langle \texttt{init} \,;\, \texttt{S >= 1} \rangle \models \Box \texttt{race-free(x.AssignWithConflict)}$$

```
reduce in ASSIGNWITHCONFLICT :
  verify-lite( { init, i(0) >= c(1)}, [] race-free(x . AssignWithConflict)) .
rewrites: 2846 in 575ms cpu (614ms real) (4947 rewrites/second)
result Bool: true
```

Agenda

Summary of Main Ideas

A technique for symbolic reachability analysis of rewrite theories with constrained built-ins

- Symbolic execution achieved by rewriting
  - matching instead of unification
- Basis of PLEXIL's symbolic semantics
  - executable in Maude
  - symbolic model checking for detecting race conditions
- Complements the ground reachability analysis tools already available to PLEXIL via its rewriting logic semantics