RMT: a tool for rewriting modulo theories

Ștefan Ciobâcă

Faculty of Computer Science
Alexandru Ioan Cuza University, Iași, Romania
stefan.ciobaca@info.uaic.ro

May 15, 2017

(joint work with Dorel Lucanu)

Outline

Introduction

Theory behind RMT
- Constrained Rewrite Rules
- Constrained Terms
- Proving Reachability Properties

Applications
- Proving Reachability and Partial Correctness
- Proving Equivalence

Discussion

Motivation: Program Equivalence

- FAOC 2016: language-parametric proof system for full equivalence;

Motivation: Program Equivalence

- FAOC 2016: language-parametric proof system for full equivalence;
- $L_1, L_2 \vdash P_1 \sim P_2$

Motivation: Program Equivalence

- FAOC 2016: language-parametric proof system for full equivalence;
- $L_1, L_2 \vdash P_1 \sim P_2$
- implementation?

Proof System for Program Equivalence

$$\textsc{Ax}\ \frac{\varphi \in E}{\vdash \varphi \Downarrow^{\infty} E} \qquad \textsc{Csq}\ \frac{\models \varphi \rightarrow \exists \tilde{x}.\varphi' \qquad \vdash \varphi' \Downarrow^{\infty} E}{\vdash \varphi \Downarrow^{\infty} E}$$

$$\textsc{CA}\ \frac{\vdash \varphi \Downarrow^{\infty} E \qquad \vdash \varphi' \Downarrow^{\infty} E}{\vdash \varphi \vee \varphi' \Downarrow^{\infty} E}$$

$$\textsc{Step}\ \frac{\models \varphi_1 \Rightarrow_1^* \varphi_1' \qquad \models \varphi_2 \Rightarrow_2^* \varphi_2' \qquad \vdash \langle \varphi_1', \varphi_2' \rangle \Downarrow^{\infty} E}{\vdash \langle \varphi_1, \varphi_2 \rangle \Downarrow^{\infty} E}$$

$$\textsc{Circ}\ \frac{\models \varphi_1 \Rightarrow_1^+ \varphi_1' \qquad \models \varphi_2 \Rightarrow_2^+ \varphi_2' \qquad \vdash \langle \varphi_1', \varphi_2' \rangle \Downarrow^{\infty} E \cup \{\langle \varphi_1, \varphi_2 \rangle\}}{\vdash \langle \varphi_1, \varphi_2 \rangle \Downarrow^{\infty} E}$$

Motivation: How to Prove Program Equivalence

```
c := n;
n := 1;
while (c != 1)
  n := n + 1;
  if (c % 2 != 0)
    then c := 3 * c + 1
    else c := c / 2
```

```
μf.λn.λa.
  if n != 1
   then
    if n % 2 != 0
      then f (3 * n + 1) (a + 1)
      else f (n / 2) (a + 1)
    else
      a
```

Outline

Introduction

Theory behind RMT
- Constrained Rewrite Rules
- Constrained Terms
- Proving Reachability Properties

Applications
- Proving Reachability and Partial Correctness
- Proving Equivalence

Discussion

Builtins

Definition (Builtin Signature)

A *builtin signature* $\Sigma^{\mathrm{b}} \triangleq (S^{\mathrm{b}}, F^{\mathrm{b}})$ is any many-sorted signature that includes the following distinguished objects:

- a sort *Bool* together with two constants $\top$ and $\bot$ of sort *Bool*,
- the propositional operations $\neg : \mathit{Bool} \to \mathit{Bool}$, $\wedge, \vee, \to\ : \mathit{Bool} \times \mathit{Bool} \to \mathit{Bool}$, and
- an equality predicate $=^? : s \times s \to \mathit{Bool}$ for each sort $s \in S^{\mathrm{b}}$.

Builtins

Definition (Builtin Signature)

A *builtin signature* $\Sigma^{\mathrm{b}} \triangleq (S^{\mathrm{b}}, F^{\mathrm{b}})$ is any many-sorted signature that includes the following distinguished objects:

- a sort *Bool* together with two constants $\top$ and $\bot$ of sort *Bool*,
- the propositional operations $\neg : \mathit{Bool} \to \mathit{Bool}$, $\wedge, \vee, \to \ : \mathit{Bool} \times \mathit{Bool} \to \mathit{Bool}$, and
- an equality predicate $=^? : s \times s \to \mathit{Bool}$ for each sort $s \in S^{\mathrm{b}}$.

Definition (Builtin Model)

A *builtin model* M^{b} is a model of a builtin signature Σ^{b}, where the interpretation of the distinguished objects of the builtin signature is fixed as follows: $M^{\mathrm{b}}_{\mathit{Bool}} = \{\top, \bot\}$, $M^{\mathrm{b}}_{\top} = \top$, $M^{\mathrm{b}}_{\bot} = \bot$, $M^{\mathrm{b}}_{=^?}(a, b) = \top$ iff $a = b$, $M^{\mathrm{b}}_{\neg}(\top) = \bot$, $M^{\mathrm{b}}_{\neg}(\bot) = \top$, $M^{\mathrm{b}}_{\wedge}(\top, b) = M^{\mathrm{b}}_{\wedge}(b, \top) = b$, $M^{\mathrm{b}}_{\wedge}(\bot, b) = M^{\mathrm{b}}_{\wedge}(b, \bot) = \bot$, and so on.

Example of a Builtin Model

Example

By *INT* we denote the usual builtin model of integers and booleans. The set of sorts, denoted by $S(INT)$, consists of the sorts *Int* and *Bool*, and the set of operation symbols, denoted by $F(INT)$, includes the usual operations over integers and booleans. Similar to the case of booleans, we shall use the infix notation for the binary integer operations.
For instance, $2 + 3$ and $3 + 5 \leq 7$ are $\Sigma(INT)$-terms and their interpretation in *INT* are 5 and $\perp$, respectively.

Signature Modulo a Builtin Model

Definition (Signature Modulo a Builtin Model)

A *signature modulo a builtin model* is a tuple $\Sigma \triangleq (S, \leq, F, M^{\mathrm{b}})$ consisting of

- an order-sorted signature $(S, \leq, F)$, and
- a builtin Σ^{b}-model M^{b}, where $\Sigma^{\mathrm{b}} \triangleq (S^{\mathrm{b}}, F^{\mathrm{b}})$ is a builtin subsignature of $(S, \leq, F)$.

We further assume that the only builtin constants in Σ are the elements of the builtin model.

IMP Syntax as a Signature Modulo a Builtin Model

Example

Let IMP^{b} the builtin model INT together with a sort Id and a sort Map. $\Sigma(\mathrm{IMP}) = (S, \leq, F, IMP^{\mathsf{b}})$.

$$
\begin{array}{rlr}
Exp ::= & Int \mid Bool & \\
 & \mid Exp\ \texttt{+}\ Exp & [\texttt{cons(plus)}] \\
 & \mid Exp\ \texttt{<=}\ Exp & [\texttt{cons(le)}] \\
Stmt ::= & Id\ \texttt{:=}\ Exp\ \texttt{;} & [\texttt{cons(asgn)}] \\
 & \texttt{if}\ \texttt{(}\ Exp\ \texttt{)}\ Stmt\ \texttt{else}\ Stmt & [\texttt{cons(if)}] \\
 & \texttt{while}\ \texttt{(}\ Exp\ \texttt{)}\ Stmt & [\texttt{cons(wh)}] \\
 & Stmt\ Stmt & [\texttt{cons(seq)}] \\
Cfg ::= & \texttt{<}\ Stmt\ \texttt{,}\ Map\ \texttt{>} & [\texttt{cons(cfg)}]
\end{array}
$$

S includes the sorts Exp, $Stmt$ and Cfg. We have $Int \leq Exp$ and $Bool \leq Exp$. F consists of the labels used to construct the abstract syntax trees (ASTs): $F_{Exp,Exp,Exp} = \{\texttt{plus}, \texttt{le}\}, \ldots$

Example Configuration

Example program configuration:

```
< s := 0; while (i <= n) s = s + i,
  i ↦ 1 n ↦ 15 >
```

Representation as a term:

$$\begin{aligned}
&\mathtt{cfg}(\mathtt{seq}(\mathtt{asgn}(\mathtt{s}, 0),\\
&\qquad\quad \mathtt{wh}(\mathtt{le}(\mathtt{i}, \mathtt{n}),\\
&\qquad\qquad \mathtt{asgn}(\mathtt{s}, \mathtt{plus}(\mathtt{s}, \mathtt{i}))))\\
&\quad , \mathtt{i} \mapsto 1\ \mathtt{n} \mapsto 15)
\end{aligned}$$

Constraint Formulae

Definition (Constraint Formulae)

The set $\mathrm{CF}(\Sigma, X)$ of *constraint formulae* over variables X is inductively defined as follows:

$$\phi ::= b \mid t_1 =^? t_2 \mid (\exists x)\phi' \mid \neg\phi' \mid \phi_1 \wedge \phi_2$$

where b ranges over $T_{\Sigma,Bool}(X)$, t_i over $T_{\Sigma,s_i}(X)$ such that s_1 and s_2 are in the same connected component, and x ranges over all variables.

Constraint Formulae

Definition (Constraint Formulae)

The set $\mathrm{CF}(\Sigma, X)$ of *constraint formulae* over variables X is inductively defined as follows:

$$\phi ::= b \mid t_1 =^? t_2 \mid (\exists x)\phi' \mid \neg\phi' \mid \phi_1 \wedge \phi_2$$

where b ranges over $T_{\Sigma,Bool}(X)$, t_i over $T_{\Sigma,s_i}(X)$ such that s_1 and s_2 are in the same connected component, and x ranges over all variables.

Definition (Semantics of Constraint Formulae)

- $M^\Sigma, \alpha \vDash b$ iff $\alpha(b) = \mathit{true}$, where $b \in T_{\Sigma,Bool}(X)$;
- $M^\Sigma, \alpha \vDash t_1 =^? t_2$ iff $\alpha(t_1) = \alpha(t_2)$;
- $M^\Sigma, \alpha \vDash (\exists x)\phi$ iff there exists $a \in M_s$ (where $x \in X_s$) such that $M^\sigma, \alpha[x \mapsto a] \vDash \phi$;
- $M^\Sigma, \alpha \vDash \neg\phi$ iff $M^\Sigma, \alpha \nvDash \phi$;
- $M^\Sigma, \alpha \vDash \phi_1 \wedge \phi_2$ iff $M^\Sigma, \alpha \vDash \phi_1$ and $M^\Sigma, \alpha \vDash \phi_2$.

Constrained Terms

Definition (Constrained Terms)

A *constrained term* φ of sort $s \in S$ is a pair $(t \mid \phi)$ with $t \in T_{\Sigma,s}(X)$ and $\phi \in \mathrm{CF}(\Sigma, X)$.

Constrained Terms

Definition (Constrained Terms)

A *constrained term* φ of sort $s \in S$ is a pair $(t \mid \phi)$ with $t \in T_{\Sigma,s}(X)$ and $\phi \in \mathrm{CF}(\Sigma, X)$.

Definition (Valuation Semantics of Constrained Terms)

The *valuation semantics* of a constrained term $(t \mid \phi)$ is the set of valuations $[\![(t \mid \phi)]\!] \triangleq \{\alpha : X \to M^{\Sigma} \mid M^{\Sigma}, \alpha \vDash \phi\}$.

Constrained Terms

Definition (Constrained Terms)

A *constrained term* φ of sort $s \in S$ is a pair $(t \mid \phi)$ with $t \in T_{\Sigma,s}(X)$ and $\phi \in \mathrm{CF}(\Sigma, X)$.

Definition (Valuation Semantics of Constrained Terms)

The *valuation semantics* of a constrained term $(t \mid \phi)$ is the set of valuations $\llbracket (t \mid \phi) \rrbracket \triangleq \{\alpha : X \to M^{\Sigma} \mid M^{\Sigma}, \alpha \vDash \phi\}$.

Definition (State Predicate Semantics of Constrained Terms)

The *state predicate semantics* of a constrained term is defined by

$$\llbracket (t \mid \phi) \rrbracket \triangleq \{\alpha(t) \mid \alpha \in \llbracket (t \mid \phi) \rrbracket\}.$$

Constrained Term Example

$$(\mathtt{cfg}\begin{pmatrix} \mathtt{wh(le(i,n),asgn(s,plus(s,i)))}, \\ \mathtt{n} \mapsto n\ \mathtt{i} \mapsto i\ \mathtt{s} \mapsto s \end{pmatrix} \mid$$
$$n \geq 0 \wedge i =^? 2 \wedge i \leq n \wedge s =^? 1)$$

Constrained Rule Systems

Definition (Constrained Rule Systems)

A *constrained rule* is a tuple (l, r, ϕ), often written as $l \twoheadrightarrow r$ `if` ϕ, where l, r are terms in $T_{\Sigma}(X)$ having sorts in the same connected component, and $\phi \in \mathrm{CF}(\Sigma, X)$.

Constrained Rule Systems

Definition (Constrained Rule Systems)

A *constrained rule* is a tuple (l, r, ϕ), often written as $l \twoheadrightarrow r \texttt{ if } \phi$, where l, r are terms in $T_\Sigma(X)$ having sorts in the same connected component, and $\phi \in \mathrm{CF}(\Sigma, X)$.
A *constrained rule system* $\mathcal{R}$ is a set of rules. $\mathcal{R}$ defines a *transition relation* $\rightsquigarrow_{\mathcal{R}}$ on M^Σ as follows: $t \rightsquigarrow_{\mathcal{R}} t'$ iff there exist a rule $l \twoheadrightarrow r \texttt{ if } \phi$ in $\mathcal{R}$, a context $c[\cdot]$, and a valuation $\alpha : X \to M^\Sigma$ such that $t = \alpha(c[l])$, $t' = \alpha(c[r])$ and $M^\Sigma, \alpha \vDash \phi$.

Reachability Properties

Definition (Reachability Properties of Constrained Rule Systems)

A *reachability formula* is a pair of constrained terms written as $\varphi \Rightarrow \varphi'$, which may share some variables. We say that a constrained rule system $\mathcal{R}$ *demonically satisfies* $\varphi \Rightarrow \varphi'$, and write

$$\mathcal{R} \vDash^{\forall} \varphi \Rightarrow \varphi'$$

iff $(M^{\Sigma}, \rightsquigarrow_{\mathcal{R}}) \vDash^{\forall} [\![\sigma(\varphi)]\!] \Rightarrow [\![\sigma(\varphi')]\!]$ for each substitution $\sigma : \mathit{var}(\varphi) \cap \mathit{var}(\varphi') \to M^{\Sigma}$.

Derivatives of Constrained Terms

Definition (Derivatives of Constrained Terms)

The *set of derivatives* of a constrained term $\varphi \triangleq (t \mid \phi)$ w.r.t. a rule $l \twoheadrightarrow r$ `if` ϕ_{lr} is

$$\begin{aligned}\Delta_{l,r,\phi_{lr}}(\varphi) \triangleq \{(c[r] \mid \phi') \mid\ & M^{\Sigma} \vDash \phi' \leftrightarrow (\phi \wedge t =^{?} c[l] \wedge \phi_{lr}),\\ & c[\cdot] \text{ an appropriate context,}\\ & \phi' \text{ is satisfiable}\}\end{aligned} \quad (1)$$

where we assume that $l \twoheadrightarrow r$ `if` ϕ and φ have disjoint variables. If $\mathcal{R}$ is a set of rules, then $\Delta_{\mathcal{R}}(\varphi) = \bigcup_{(l,r,\phi_{lr})\in\mathcal{R}} \Delta_{l,r,\phi_{lr}}(\varphi)$. A constrained term φ is *$\mathcal{R}$-derivable* if $\Delta_{\mathcal{R}}(\varphi) \neq \emptyset$.

Proof System for Symbolic Execution

Figure: The DSTEP($\mathcal{R}$) Proof System

$$[\text{axiom}]\ \frac{}{(t \mid \phi) \Rightarrow \varphi'}\ M^{\Sigma} \models \phi \leftrightarrow \bot$$

$$[\text{subs}]\ \frac{(t'' \mid \phi'' \wedge \neg\phi''') \Rightarrow (t' \mid \phi')}{(t \mid \phi) \Rightarrow (t' \mid \phi')}\quad \begin{array}{l} (t'' \mid \phi'') \Rightarrow \varphi' \equiv (t \mid \phi) \Rightarrow \varphi',\ \text{and} \\ M^{\Sigma} \models \phi''' \leftrightarrow (\exists X)(t'' =^{?} t' \wedge \phi'),\ \text{ar} \\ X \triangleq var(t', \phi') \setminus var(t'', \phi'') \end{array}$$

$$[\text{der}^{\forall}]\ \frac{\{(t^j \mid \phi^j) \Rightarrow \varphi' \mid (t^j \mid \phi^j) \in \Delta_{\mathcal{R}}((t'' \mid \phi''))\}}{(t \mid \phi) \Rightarrow \varphi'}$$

$$\begin{array}{l} (t \mid \phi)\ \mathcal{R}\text{−derivable, and} \\ (t'' \mid \phi'') \Rightarrow \varphi' \equiv (t \mid \phi) \Rightarrow \varphi',\ \text{and} \\ \phi'' \wedge \bigwedge \left\{ \neg(\exists Y)\phi^j \;\middle|\; \begin{array}{l} (t^j \mid \phi^j) \in \Delta_{\mathcal{R}}((t'' \mid \phi'')), \\ Y \triangleq var(t^j, \phi^j) \setminus var(t'', \phi'') \end{array} \right\}\ \text{not satisfial} \end{array}$$

Soundness

Let $\mathcal{R}$ be a constrained rules system. Then $\mathcal{R} \models^{\forall} \nu\, \widehat{\mathsf{DSTEP}}(\mathcal{R})$.

Circularity

Definition (Demonic circular coinduction)

Let G be a finite set reachability formulae. Then the set of rules DCC($\mathcal{R}$, G) consists of DSTEP($\mathcal{R}$) together with

$$[\text{circ}]\ \frac{\begin{array}{c}(t'_c \mid \phi'_c \wedge \phi \wedge \phi'')) \Rightarrow \varphi', \\ (t \mid \phi \wedge \neg\phi'') \Rightarrow \varphi'\end{array}}{(t \mid \phi) \Rightarrow \varphi'} \quad \begin{array}{l} M^{\Sigma} \models \phi'' \leftrightarrow (\exists var(t_c, \phi_c))(t =^{?} t_c \wedge \\ (t_c \mid \phi_c) \Rightarrow (t'_c \mid \phi'_c) \in G \end{array}$$

where the variables in $(t_c \mid \phi_c) \Rightarrow (t'_c \mid \phi'_c)$ are renamed such that $var(t_c, \phi_c) \cap var(t, \phi) = \emptyset$.

Soundness of Circularity

Definition

Let PT be a proof tree of $\varphi \Rightarrow \varphi'$ under DCC($\mathcal{R}, G$). A [circ] node in PT is *progressive* iff it has as ancestor a [der$^\forall$] node. PT is *progressive* iff all its [circ] nodes are progressive.

Soundness of Circularity

Definition

Let PT be a proof tree of $\varphi \Rightarrow \varphi'$ under $\mathrm{DCC}(\mathcal{R}, G)$. A [circ] node in PT is *progressive* iff it has as ancestor a [der$^\forall$] node. PT is *progressive* iff all its [circ] nodes are progressive.

[Circularity Principle]
Let $\mathcal{R}$ be a constrained rule system and G a set of goals. If $(\mathcal{R}, G) \vdash^\forall G$ then $\mathcal{R} \models^\forall G$.

Example of Circularity

$$(\mathtt{cfg}\begin{pmatrix} \mathtt{wh}(\mathtt{n}, \mathtt{seq}(\mathtt{asgn}(\mathtt{s}, \mathtt{plus}(\mathtt{s}, \mathtt{n})), \mathtt{asgn}(\mathtt{n}, \mathtt{minus}(\mathtt{n}, 1)))), \\ \mathtt{n} \mapsto n \ \mathtt{s} \mapsto 0 \end{pmatrix} \mid n \geq 0) \Rightarrow$$

$$(\mathtt{cfg}\begin{pmatrix} \mathtt{skip}, \\ \mathtt{n} \mapsto 0 \ \mathtt{s} \mapsto n * (n-1)/2 \end{pmatrix} \mid \top)$$

can be proven by using the following circularity:

Example of Circularity

$$(\mathtt{cfg}\begin{pmatrix} \mathtt{wh(n, seq(asgn(s, plus(s, n)), asgn(n, minus(n, 1))))}, \\ \mathtt{n} \mapsto n\ \mathtt{s} \mapsto 0 \end{pmatrix} \mid n \geq 0) \Rightarrow$$
$$(\mathtt{cfg}\begin{pmatrix} \mathtt{skip}, \\ \mathtt{n} \mapsto 0\ \mathtt{s} \mapsto n * (n-1)/2 \end{pmatrix} \mid \top)$$

can be proven by using the following circularity:

$$(\mathtt{cfg}\begin{pmatrix} \mathtt{wh(n, seq(asgn(s, plus(s, n)), asgn(n, minus(n, 1))))}, \\ \mathtt{n} \mapsto n\ \mathtt{s} \mapsto s' \end{pmatrix} \mid n \geq 0) \Rightarrow$$
$$(\mathtt{cfg}\begin{pmatrix} \mathtt{skip}, \\ \mathtt{n} \mapsto 0\ \mathtt{s} \mapsto s' + n * (n-1)/2 \end{pmatrix} \mid \top).$$

Program Equivalence

- similar, language-parametric, proof system.

DEMO

Discussion

http://github.com/ciobaca/rmt/

- work in progress tool for term rewriting + SMT solving;
- procedure for reachability;
- procedure for full-equivalence;

- polish and publish as library;
- applications in verification (compiler correctness, optimizations, etc).