RMT: a tool for rewriting modulo theories

Ştefan Ciobâcă

Faculty of Computer Science
Alexandru Ioan Cuza University, Iaşi, 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

\[
\text{Ax} \quad \frac{\varphi \in E}{\vdash \varphi \downarrow^\infty E} \\
\text{CsQ} \quad \frac{\vdash \varphi \rightarrow \exists \bar{x}. \varphi'}{\vdash \varphi' \downarrow^\infty E}
\]

\[
\text{CA} \quad \frac{\vdash \varphi \downarrow^\infty E \quad \vdash \varphi' \downarrow^\infty E}{\vdash \varphi \lor \varphi' \downarrow^\infty E}
\]

\[
\text{STEP} \quad \frac{\vdash \varphi_1 \Rightarrow_1^* \varphi'_1 \quad \vdash \varphi_2 \Rightarrow_2^* \varphi'_2}{\vdash \langle \varphi'_1, \varphi'_2 \rangle \downarrow^\infty E}
\]

\[
\text{CIRC} \quad \frac{\vdash \varphi_1 \Rightarrow_1^+ \varphi'_1 \quad \vdash \varphi_2 \Rightarrow_2^+ \varphi'_2}{\vdash \langle \varphi'_1, \varphi'_2 \rangle \downarrow^\infty E \cup \{ \langle \varphi_1, \varphi_2 \rangle \}}
\]
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

\mu f. \lambda n. \lambda a.
    \begin{align*}
        &\text{if } n \neq 1 \\
        &\text{then}
        &\text{if } n \mod 2 \neq 0 \\
        &\text{then } f(3 \times n + 1)(a + 1) \\
        &\text{else } f(n/2)(a + 1) \\
        &\text{else } a
    \end{align*}
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* $\Sigma^b \triangleq (S^b, F^b)$ is any many-sorted signature that includes the following distinguished objects:

- a sort $\text{Bool}$ together with two constants $\top$ and $\bot$ of sort $\text{Bool}$,
- the propositional operations $\neg : \text{Bool} \rightarrow \text{Bool}$, $\land, \lor, \rightarrow : \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$, and
- an equality predicate $=: ? : s \times s \rightarrow \text{Bool}$ for each sort $s \in S^b$. 

Definition (Builtin Model)
A *builtin model* $M^b$ is a model of a builtin signature $\Sigma^b$, where the interpretation of the distinguished objects of the builtin signature is fixed as follows:

- $M^b_{\text{Bool}} = \{\top, \bot\}$,
- $M^b_{\top} = \top$,
- $M^b_{\bot} = \bot$,
- $M^b_{=} (a, b) = \top$ iff $a = b$,
- $M^b_{\neg}(\top) = \bot$,
- $M^b_{\neg}(\bot) = \top$,
- $M^b_{\land}(\top, b) = M^b_{\land}(b, \top) = b$,
- and so on.
Builtins

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

- a sort $\text{Bool}$ together with two constants $\top$ and $\bot$ of sort $\text{Bool}$,
- the propositional operations $\neg : \text{Bool} \to \text{Bool}$, $\land, \lor, \to : \text{Bool} \times \text{Bool} \to \text{Bool}$, and
- an equality predicate $\equiv^? : s \times s \to \text{Bool}$ for each sort $s \in S^b$.

Definition (Builtin Model)
A *builtin model* $M^b$ is a model of a builtin signature $\Sigma^b$, where the interpretation of the distinguished objects of the builtin signature is fixed as follows: $M^b_{\text{Bool}} = \{\top, \bot\}$, $M^b_\top = \top$, $M^b_\bot = \bot$, $M^b_{\equiv^?}(a, b) = \top$ iff $a = b$, $M^b_\land(\top) = \bot$, $M^b_\land(\bot) = \top$, $M^b_\land(\top, b) = M^b_\land(b, \top) = b$, $M^b_\land(\bot, b) = M^b_\land(b, \bot) = \bot$, 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 $\bot$, respectively.
Definition (Signature Modulo a Builtin Model)

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

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

We further assume that the only builtin constants in $\Sigma$ are the elements of the builtin model.
IMP Syntax as a Signature Modulo a Builtin Model

Example

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

\[
\begin{align*}
\text{Exp} & ::= \text{Int} \mid \text{Bool} \\
& \quad \mid \text{Exp} + \text{Exp} & [\text{cons}(\text{plus})] \\
& \quad \mid \text{Exp} \leq \text{Exp} & [\text{cons}(\text{le})] \\
\text{Stmt} & ::= \text{Id} := \text{Exp} & [\text{cons}(\text{asgn})] \\
& \quad \mid \text{if (Exp) Stmt else Stmt} & [\text{cons}(\text{if})] \\
& \quad \mid \text{while (Exp) Stmt} & [\text{cons}(\text{wh})] \\
& \quad \mid \text{Stmt Stmt} & [\text{cons}(\text{seq})] \\
\text{Cfg} & ::= < \text{Stmt} , \text{Map} > & [\text{cons}(\text{cfg})]
\end{align*}
\]

$S$ includes the sorts $\text{Exp}$, $\text{Stmt}$ and $\text{Cfg}$. We have $\text{Int} \leq \text{Exp}$ and $\text{Bool} \leq \text{Exp}$. $F$ consists of the labels used to construct the abstract syntax trees (ASTs): $F_{\text{Exp},\text{Exp},\text{Exp}} = \{\text{plus}, \text{le}\}, \ldots$
Example program configuration:
\[
< \text{s := 0; while (i <= n) s = s + i, i \mapsto 1 n \mapsto 15} >
\]

Representation as a term:
\[
\text{cfg(seq(asgn(s, 0),
  \text{wh(}i, n, \text{asgn(s, plus(s, i))))
, i \mapsto 1 n \mapsto 15)}
\]
Constraint Formulae

Definition (Constraint Formulae)

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

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

where $b$ ranges over $T_{\Sigma, \text{Bool}}(X)$, $t_i$ 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 $\text{CF}(\Sigma, X)$ of constraint formulae over variables $X$ is inductively defined as follows:

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

where $b$ ranges over $T_{\Sigma, \text{Bool}}(X)$, $t_i$ 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)

- $M^\Sigma, \alpha \models b$ iff $\alpha(b) = \text{true}$, where $b \in T_{\Sigma, \text{Bool}}(X)$;
- $M^\Sigma, \alpha \models t_1 = ? t_2$ iff $\alpha(t_1) = \alpha(t_2)$;
- $M^\Sigma, \alpha \models (\exists x)\phi$ iff there exists $a \in M_s$ (where $x \in X_s$) such that $M^\sigma, \alpha[x \mapsto a] \models \phi$;
- $M^\Sigma, \alpha \models \neg \phi$ iff $M^\Sigma, \alpha \not\models \phi$;
- $M^\Sigma, \alpha \models \phi_1 \land \phi_2$ iff $M^\Sigma, \alpha \models \phi_1$ and $M^\Sigma, \alpha \models \phi_2$. 
Constrained Terms

Definition (Constrained Terms)

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

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

Definition (Valuation Semantics of Constrained Terms)
The *valuation semantics* of a constrained term $(t \mid \phi)$ is the set of valuations $\llbracket (t \mid \phi) \rrbracket \triangleq \{ \alpha : X \rightarrow M^\Sigma \mid M^\Sigma, \alpha \models \phi \}$. 
Constrained Terms

Definition (Constrained Terms)
A constrained term $\varphi$ of sort $s \in S$ is a pair $(t \mid \phi)$ with $t \in T_{\Sigma,s}(X)$ and $\phi \in \text{CF}(\Sigma, X)$.

Definition (Valuation Semantics of Constrained Terms)
The valuation semantics of a constrained term $(t \mid \phi)$ is the set of valuations $\llbracket (t \mid \phi) \rrbracket \triangleq \{ \alpha : X \rightarrow M^\Sigma \mid M^\Sigma, \alpha \models \phi \}$.

Definition (State Predicate Semantics of Constrained Terms)
The state predicate semantics of a constrained term $(t \mid \phi)$ is defined by

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

\[
\begin{align*}
(\text{cfg} \left( \begin{array}{l}
\text{while}(i, n), \text{asgn}(s, \text{plus}(s, i)), \\
n \mapsto n, i \mapsto i, s \mapsto s
\end{array} \right), \\
&\quad n \geq 0 \land i = 2 \land i \leq n \land s = 1)
\end{align*}
\]
Constrained Rule Systems

Definition (Constrained Rule Systems)

A constrained rule is a tuple \((l, r, \phi)\), often written as \(l \rightarrow r \text{ if } \phi\), where \(l, r\) are terms in \(T_\Sigma(X)\) having sorts in the same connected component, and \(\phi \in CF(\Sigma, X)\).
Definition (Constrained Rule Systems)

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

\[
\mathcal{R} \models^\forall \varphi \Rightarrow \varphi'
\]

iff \( (M^\Sigma, \rightsquigarrow) \models^\forall [\sigma(\varphi)] \Rightarrow [\sigma(\varphi')] \) for each substitution \( \sigma : \text{var}(\varphi) \cap \text{var}(\varphi') \rightarrow M^\Sigma \).
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 \( l \rightarrow r \) if \( \phi_{lr} \) is

\[
\Delta_{l,r,\phi_{lr}}(\varphi) \triangleq \{(c[r] \mid \phi') \mid M^\Sigma \models \phi' \leftrightarrow (\phi \land t =? c[l] \land \phi_{lr}), c[\cdot] \text{ an appropriate context}, \\
\phi' \text{ is satisfiable}\}
\]

(1)

where we assume that \( l \rightarrow r \) if \( \phi \) and \( \varphi \) 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 \( \varphi \) is \( \mathcal{R} \)-derivable if \( \Delta_{\mathcal{R}}(\varphi) \neq \emptyset \).
Proof System for Symbolic Execution

**Figure:** The DSTEP(\(R\)) Proof System

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

\[
\text{[subs]} \quad (t'' \mid \phi'' \land \neg \phi''') \Rightarrow (t' \mid \phi') \\
(t \mid \phi) \Rightarrow (t' \mid \phi')
\]

\[
(t'' \mid \phi'') \Rightarrow \phi' \equiv (t \mid \phi) \Rightarrow \phi', \text{ and} \\
M^\Sigma \models \phi''' \iff (\exists X)(t'' =? t' \land \phi'), \text{ and} \\
X \triangleq \text{var}(t', \phi') \setminus \text{var}(t'', \phi'')
\]

\[
\text{[der\AA]} \quad \left\{ (t^j \mid \phi^j) \Rightarrow \phi' \mid (t^j \mid \phi^j) \in \Delta_R((t'' \mid \phi'')) \right\} \\
(t \mid \phi) \Rightarrow \phi'
\]

\[
(t \mid \phi) \text{ \(R\)-derivable, and} \\
(t'' \mid \phi'') \Rightarrow \phi' \equiv (t \mid \phi) \Rightarrow \phi', \text{ and} \\
\phi'' \land \bigwedge \left\{ \neg (\exists Y)\phi^j \mid (t^j \mid \phi^j) \in \Delta_R((t'' \mid \phi'')), \right. \\
Y \triangleq \text{var}(t^j, \phi^j) \setminus \text{var}(t'', \phi'') \right\} \text{ not satisfiable}
\]
Let $\mathcal{R}$ be a constrained rules system. Then $\mathcal{R} \models \forall \, \nu \text{DSTEP}(\mathcal{R})$. 
Circularity

Definition (Demonic circular coinduction)

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

$$
\begin{align*}
\forall t &\in \Sigma \models \phi'' \iff (\exists \text{var}(t_c, \phi_c))(t = ? t_c \land \phi'') \\
(t_c \mid \phi_c) &\Rightarrow (t_c' \mid \phi'_c) \in G
\end{align*}
$$

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

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

\[
\begin{align*}
\text{cfg} & \left( \text{wh}(n, \text{seq}(\text{asgn}(s, \text{plus}(s, n)), \text{asgn}(n, \text{minus}(n, 1))))), \\
& \quad n \mapsto n \ s \mapsto 0
\end{align*}
\]

\[
\begin{align*}
\text{cfg} & \left( \text{skip}, \\
& \quad n \mapsto 0 \ s \mapsto n \ast (n - 1)/2 \right) \mid \top
\end{align*}
\]

\[
\begin{align*}
& \text{can be proven by using the following circularity:}
\end{align*}
\]
Example of Circularity

\[
\begin{align*}
\text{can be proven by using the following circularity:} \\
\text{(cfg \( \left( \begin{array}{c}
\text{wh}(n, \text{seq}(\text{asgn}(s, \text{plus}(s, n)), \text{asgn}(n, \text{minus}(n, 1))))), \\
n \mapsto n \\ s \mapsto 0
\end{array} \right), \n \geq 0) \Rightarrow \\
\text{(cfg \( \left( \begin{array}{c}
\text{skip}, \\
n \mapsto 0 \\ s \mapsto n \ast (n - 1)/2
\end{array} \right), \top) \\
\text{(cfg \( \left( \begin{array}{c}
\text{wh}(n, \text{seq}(\text{asgn}(s, \text{plus}(s, n)), \text{asgn}(n, \text{minus}(n, 1))))), \\
n \mapsto n \\ s \mapsto s' \\
\text{skip}, \\
n \mapsto 0 \\ s \mapsto s' + n \ast (n - 1)/2
\end{array} \right), \top).}
\end{align*}
\]
Program Equivalence

- similar, language-parametric, proof system.
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).