Congruence Closure with Free Variables (Work in Progress)

Haniel Barbosa, Pascal Fontaine

INRIA Nancy – VeriDis
Université de Lorraine
UFRN

2015–08–03

Outline

- SMT solving
- Congruence Closure with Free Variables
- Extensions and next tasks

SMT solving

First-order logic modulo theories:

$$\varphi = \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\}$$

SMT solving

First-order logic modulo theories:

$$\varphi = \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\}$$

Through SAT solving one may obtain that $\mathcal{L} \cup \mathcal{Q} \models \varphi$, for

$$\begin{array}{lcl} \mathcal{L} & = & \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\} \\ \mathcal{Q} & = & \{\forall x_1, x_2.\ (f(x_1) \not\approx a \vee f(x_2) \approx b)\} \end{array}$$

SMT solving

First-order logic modulo theories:

$$\varphi = \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\}$$

Through SAT solving one may obtain that $\mathcal{L} \cup \mathcal{Q} \models \varphi$, for

$$\begin{array}{rcl} \mathcal{L} & = & \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\} \\ \mathcal{Q} & = & \{\forall x_1, x_2.\ (f(x_1) \not\approx a \vee f(x_2) \approx b)\} \end{array}$$

Through ground reasoning, $\mathcal{L}$ is shown satisfiable.

What about $\mathcal{Q}$?

SMT solving

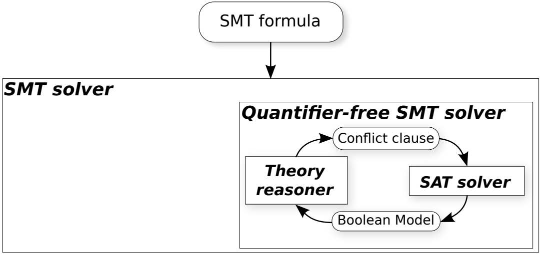
How to handle quantified formulas in the SMT context?

- FOL with equality is semi-decidable, but considering theories frequently leads to undecidability.
- Reasoning through incomplete techniques relying on decidable fragments — *instantiation*.

SMT solving

How to handle quantified formulas in the SMT context?

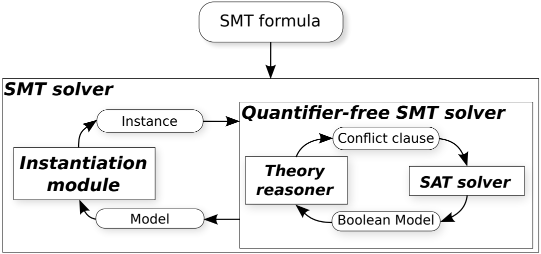
- FOL with equality is semi-decidable, but considering theories frequently leads to undecidability.
- Reasoning through incomplete techniques relying on decidable fragments — *instantiation*.



SMT solving

How to handle quantified formulas in the SMT context?

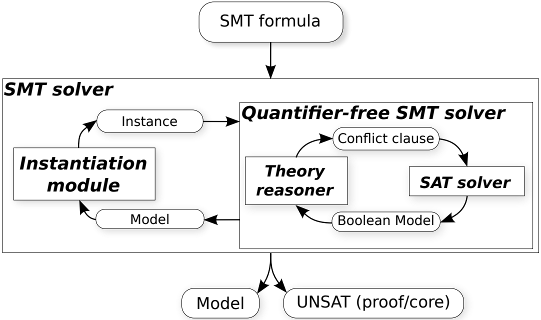
- FOL with equality is semi-decidable, but considering theories frequently leads to undecidability.
- Reasoning through incomplete techniques relying on decidable fragments — *instantiation*.



SMT solving

How to handle quantified formulas in the SMT context?

- FOL with equality is semi-decidable, but considering theories frequently leads to undecidability.
- Reasoning through incomplete techniques relying on decidable fragments — *instantiation*.



With too many instances available, their selection becomes crucial.

Ground conflicting instances generation

Context (ground model)

- Given a formula φ and a theory $\mathcal{T}$, SMT solver derives, if any, groundly $\mathcal{T}$-satisfiable sets of literals $\mathcal{L}$ and $\mathcal{Q}$ s.t. $\mathcal{L} \cup \mathcal{Q} \models \varphi$.
- $\mathcal{L}$ is a set of ground literals.
- $\mathcal{Q}$ is a set of quantified formulas.

Ground conflicting instances [Reynolds et al., 2014]

- Derive, for some $\forall \mathbf{x}.\psi \in \mathcal{Q}$, ground substitutions σ s.t. $\mathcal{L} \models \neg\psi\sigma$.
- As instances $\forall \mathbf{x}.\psi \rightarrow \psi\sigma$ refute $\mathcal{L} \cup \mathcal{Q}$, their addition to φ require the derivation of a new ground model, if any.

Congruence Closure with Free Variables

- Finding ground conflicting instances is equivalent to solving a non-simultaneous E-unification problem (NP-complete).

  [Tiwari et al., 2000]

- It has also been shown to be amenable to the use of congruence closure procedures.

- Algorithm CCFV: extends congruence closure decision procedure, being able to perform unification on free variables.

CCFV

Finding substitutions

It computes, **if any**, a sequence of substitutions $\sigma_0, \ldots, \sigma_k$ such that, for $\neg\psi = l_1 \wedge \cdots \wedge l_k$,

$$\sigma_0 = \varnothing;\ \sigma_{i-1} \subseteq \sigma_i \text{ and } \mathcal{L} \models l_i\sigma_i$$

which guarantees that $\mathcal{L} \models \neg\psi\sigma_k$.

Unification

Adapts the *recursive descent E-unification* algorithm in [Baader et al., 2001].

Example

$$\varphi = \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\}$$

$$\mathcal{L} = \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\}$$

Example

$$
\begin{aligned}
\varphi &= \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\} \\
\mathcal{L} &= \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\} \\
\neg\psi &= (f(x_1) \approx a \wedge f(x_2) \not\approx b)
\end{aligned}
$$

Example

$$\begin{array}{rcl} \varphi & = & \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\} \\ \mathcal{L} & = & \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\} \\ \neg\psi & = & (f(x_1) \approx a \wedge f(x_2) \not\approx b) \end{array}$$

1. Evaluates $f(x_1) \approx a$:
    - since $f(c) \in [a]$, unifies $\langle f(x_1), f(c) \rangle$.
    - leads to the substitution $\sigma_1 = \{x_1 \mapsto c\}$, such that $\mathcal{L} \models (f(x_1) \approx a)\sigma_1$.

Example

$$\begin{aligned}
\varphi &= \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\} \\
\mathcal{L} &= \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\} \\
\neg\psi &= (f(x_1) \approx a \wedge f(x_2) \not\approx b)
\end{aligned}$$

1. Evaluates $f(x_1) \approx a$:
   - since $f(c) \in [a]$, unifies $\langle f(x_1), f(c) \rangle$.
   - leads to the substitution $\sigma_1 = \{x_1 \mapsto c\}$, such that $\mathcal{L} \models (f(x_1) \approx a)\sigma_1$.
2. Evaluates $f(x_2) \not\approx b$:
   - since $f(a) \in [b]$, if the pair $\langle f(x_2), f(b) \rangle$ is unifiable then the resulting σ is conflicting.
   - leads to the substitution $\sigma_2 = \{x_1 \mapsto c, x_2 \mapsto b\}$ such that $\mathcal{L} \models (f(x_2) \not\approx b)\sigma_2$.

Example

$$
\begin{aligned}
\varphi &= \left\{ \begin{array}{l} f(c) \approx a \vee c \approx d, f(a) \approx b, f(b) \not\approx f(a), \\ \forall x_1, x_2.\ f(x_1) \not\approx a \vee f(x_2) \approx b \end{array} \right\} \\
\mathcal{L} &= \{f(c) \approx a,\ f(a) \approx b,\ f(b) \not\approx f(a)\} \\
\neg\psi &= (f(x_1) \approx a \wedge f(x_2) \not\approx b)
\end{aligned}
$$

1. Evaluates $f(x_1) \approx a$:
   - since $f(c) \in [a]$, unifies $\langle f(x_1), f(c) \rangle$.
   - leads to the substitution $\sigma_1 = \{x_1 \mapsto c\}$, such that $\mathcal{L} \models (f(x_1) \approx a)\sigma_1$.
2. Evaluates $f(x_2) \not\approx b$:
   - since $f(a) \in [b]$, if the pair $\langle f(x_2), f(b) \rangle$ is unifiable then the resulting σ is conflicting.
   - leads to the substitution $\sigma_2 = \{x_1 \mapsto c, x_2 \mapsto b\}$ such that $\mathcal{L} \models (f(x_2) \not\approx b)\sigma_2$.

CCFV returns $\sigma = \{x_1 \mapsto c, x_2 \mapsto b\}$, which is a ground conflicting substitution, since $\mathcal{L} \wedge \psi\sigma$ is groundly unsatisfiable.

Algorithm

proc $\mathrm{CCFV}(\mathcal{L}, \psi)$

- $\mathfrak{C} \leftarrow \{s \approx t \mid s \approx t \in \mathcal{L}\}$; $\mathfrak{D} \leftarrow \{s \not\approx t \mid s \not\approx t \in \mathcal{L}\}$; $\Delta_{\mathbf{x}} \leftarrow \varnothing$ `// Init`
- **foreach** $l \in \neg\psi$ **do**
  - **if** $\mathbf{not}(\textsc{Handle}(\mathfrak{C}, \mathfrak{D}, \Delta_{\mathbf{x}}, l))$ **then**
    - $\Delta_{\mathbf{x}} \leftarrow \Delta_{\mathbf{x}} \cup \{\{x \mapsto \mathrm{SEL}(x) \mid x \in \mathbf{x}\}\}$
    - **if** $\varnothing \in \Delta_{\mathbf{x}}$ **then return** $\varnothing$ `// No` σ `s.t.` $\mathcal{L} \models \neg\psi\sigma$
    - $\textsc{Reset}(\mathfrak{C}, \mathfrak{D}, \neg\psi)$ `// Backtracking`
- **return** $\{x \mapsto \mathrm{SEL}(x) \mid x \in \mathbf{x}\}$ `//` $\mathcal{L} \models \neg\psi\sigma$

proc $\textsc{Handle}(\mathfrak{C}, \mathfrak{D}, \Delta_{\mathbf{x}}, \ l)$

- **match** l **:**
  - $u \approx v$:
    - **if** $\mathfrak{C} \cup \mathfrak{D} \models u \not\approx v$ **then return** $\bot$ `// Checks consistency`
    - $\mathfrak{C} \leftarrow \mathfrak{C} \cup \{u \approx v\}$ `// Updates` $\mathfrak{C} \cup \mathfrak{D}$
  - $u \not\approx v$: ...
- $\Lambda \leftarrow (\textsc{Unify}\ \delta\ l) \setminus_{\mathfrak{C}} \Delta_{\mathbf{x}}$ `//` $\mathcal{L} \models l\sigma$`, for every` $\sigma \in \Lambda$
- **if** $\Lambda \neq \varnothing$ **then**
  - **let** $\sigma \in \Lambda$ **in**
    - $\mathfrak{C} \leftarrow \mathfrak{C} \cup \bigcup_{x \in dom(\sigma)} \{x \approx x\sigma\}$
  - **return** $\top$
- **return** $\bot$

Extensions

- CCFV only works in very restricted scenarios.

Extensions

- CCFV only works in very restricted scenarios.

- Basis for broader procedures.
  - E-matching
  - MBQI

Extensions

- CCFV only works in very restricted scenarios.
- Basis for broader procedures.
  - E-matching
  - MBQI
  - Simultaneous (Bounded) Rigid E-Unification [Backeman et al., 2015]

Extensions

- CCFV only works in very restricted scenarios.

- Basis for broader procedures.
  - E-matching
  - MBQI
  - Simultaneous (Bounded) Rigid E-Unification [Backeman et al., 2015]
  - Saturation based procedures (Inst-Gen-Eq, Hierarchic Superposition, ...)

Next tasks

- Continue implementation.
- Integrating extensions into general framework.
- Handling arithmetic reasoning together with conflict driven instantiation.

More details in the paper: http://www.loria.fr/∼hbarbosa/quantify2015.pdf