Semantically-Guided Goal-Sensitive Theorem Proving[1]

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy

July 2014

[1] Joint work with David Plaisted

Motivation

A first-order theorem-proving method simultaneously

- DPLL-style model based
- Proof confluent
- Semantically guided
- Goal sensitive

DPLL-style model based

- Derivation state includes candidate (partial) model
- Inference and search (for model) guide each other (e.g., CDCL in DPLL)
- Inference as model transformation

Proof confluent

- Confluence: diamond property: $\swarrow \searrow \Rightarrow \searrow \swarrow$
- Proof confluence:
  Committing to an inference never prevents proof
- No backtracking

Semantically guided

- Semantic guidance by a given initial interpretation $\mathcal{I}$
- In theory and manual examples: e.g., based on sign
- In practice: problems and knowledge bases may come with it
- SGGS: semantic guidance and model-based style connected; $\mathcal{I}$ as starting point and default

Goal sensitive

- Notion of goal:
  - $H \models^{?} \varphi$
  - $H \cup \{\neg\varphi\} \vdash^{?} \bot$
  - $H \cup \{\neg\varphi\} \rightsquigarrow S$ set of clauses
  - $S = T \uplus iSOS$ where $H \rightsquigarrow T$, $\{\neg\varphi\} \rightsquigarrow iSOS$
  - $S = T \uplus iSOS$, $iSOS$ input set of support
- Alternatively: $S = T \uplus iSOS$ with T consistent, $iSOS = S \setminus T$
- Generate only clauses connected with $iSOS$

Motivation summary

- A first-order reasoning method with all these properties?!
- Yes!!!

SGGS
Semantically Guided Goal Sensitive
reasoning

Model Representation

Model representation from PL to FOL:

- DPLL: Trail of literals
  $L_1, \ldots, L_n$
- SGGS:
  - Initial interpretation $\mathcal{I}$
  - Sequence of constrained clauses with selected literals
    $\Gamma = A_1 \rhd C_1[L_1], \ldots, A_n \rhd C_n[L_n]$
  - That modify $\mathcal{I}$

What does a constrained clause represent?

Its constrained ground instances (cgi's)
or ground instances satisfying the constraints

Example:

- $x \not\equiv y \rhd P(x, y)$
- $P(a, b) \in Gr(x \not\equiv y \rhd P(x, y))$
- $P(b, b) \notin Gr(x \not\equiv y \rhd P(x, y))$

Literal selection

- Every literal in sequence is either $\mathcal{I}$-true or $\mathcal{I}$-false
- $\mathcal{I}$-true: all cgi's true in $\mathcal{I}$
- $\mathcal{I}$-false: all cgi's false in $\mathcal{I}$
- Literal tells truth value of all its cgi's
- Prefer $\mathcal{I}$-false literals for selection:
  If clause has $\mathcal{I}$-false literals, one is selected

Interpretation $\mathcal{I}[\Gamma]$ represented by clause sequence Γ

- Partial interpretation $\mathcal{I}^p(\Gamma|_j)$ for prefix $\Gamma|_j$
- For each clause $A_j \rhd C_j[L_j]$ take its proper constrained ground instances (pcgi):
  - Not satisfied by $\mathcal{I}^p(\Gamma|_{j-1})$
  - Satisfiable by adding the pcgi of L_j
- $\mathcal{I}[\Gamma]$: complete $\mathcal{I}^p(\Gamma)$ by consulting $\mathcal{I}$ whenever $\mathcal{I}^p(\Gamma)$ does not determine truth value
- $\mathcal{I}[\Gamma]$ is $\mathcal{I}$ modified to satisfy the pcgi's of the selected literals

Example

- $\mathcal{I}$: all negative
- Sequence Γ: $[P(x)],\ top(y) \not\equiv g \triangleright [Q(y)],\ z \not\equiv c \triangleright [Q(g(z))]$
- Interpretation $\mathcal{I}[\Gamma]$:
  $\mathcal{I}[\Gamma] \models P(x)$
  $\mathcal{I}[\Gamma] \models Q(t)$ for all ground terms t whose top symbol is not g
  $\mathcal{I}[\Gamma] \models Q(g(t))$ for all ground terms t other than c
  $\mathcal{I}[\Gamma] \not\models L$ for all other positive literals L

SGGS-Derivation

- Input set of clauses S
- Initial interpretation $\mathcal{I}$
- Derivation $\Gamma_0 \vdash \Gamma_1 \vdash \ldots \Gamma_j \vdash \ldots$
- Γ_0 is empty, $\mathcal{I}[\Gamma_0]$ is $\mathcal{I}$
- Γ_j generated from Γ_{j-1}, S, and $\mathcal{I}$ by an SGGS inference rule
- Termination: either Γ_k contains empty clause (refutation) or no rule applies

Assignment function

- Every sequence Γ in derivation equipped with (a set of) assignment functions Φ (one per clause)
- Maps $\mathcal{I}$-true literal L not selected in $A_i \rhd C_i[L_i]$ to preceding clause $A_j \rhd C_j[L_j]$ ($j < i$) with $\mathcal{I}$-false selected literal
- All cgi's of $A_i \rhd L$ appear negated among pcgi's of $A_j \rhd L_j$
- $A_i \rhd C_i[L_i]$ depends on $A_j \rhd C_j[L_j]$
- Purposes: refutation, goal sensitivity

Main inference mechanisms

1. Instance generation: extend current candidate model
2. Resolution: amend candidate model removing inconsistencies or generate $\bot$ if impossible
3. Splitting inferences: amend candidate model pulling out duplications
   - Introduce constraints to capture different sets of ground instances
4. Deletion of disposable clauses (model-based redundancy)

SGGS-Extension

$\Gamma \vdash \Gamma'$

- Take input clause C and find instance E not satisfied by $\mathcal{I}[\Gamma]$ and such that all its literals are either $\mathcal{I}$-true or $\mathcal{I}$-false
- Find a place in Γ where E can be inserted so that the $\mathcal{I}$-true literals can be assigned properly
- E satisfied by $\mathcal{I}[\Gamma']$
- Lifting Theorem:
  For all ground instance $C\mu$ not satisfied by $\mathcal{I}[\Gamma]$, there is SGGS-extension of Γ into Γ' so that $C\mu$ satisfied by $\mathcal{I}[\Gamma']$

SGGS-Resolution

- Model-based: resolution in the current candidate model
- Resolves clauses $B \rhd D[M]$ and $A \rhd C[L]$ in the sequence, not in the input set
- Only selected literals are resolved upon
- One $\mathcal{I}$-true and one $\mathcal{I}$-false
- $B \rhd D[M]$ is $\mathcal{I}$-all-true and precedes $A \rhd C[L]$
- SGGS-resolution uses matching: $L = \neg M\vartheta$ and $A \supset B\vartheta$
- Resolvent replaces $A \rhd C[L]$

Inside SGGS-Resolution

Theorem:
Under the hypotheses of SGGS-resolution:

- $A \rhd L$ has no pcgi's
- The atoms of the cgi's of $A \rhd L$ that $A \rhd C[L]$ would capture are covered by $B \rhd D[M]$
- $A \rhd C[L]$ replaced by resolvent which captures the cgi's of $C \setminus \{L\}$

Example of SGGS-Resolution

- $\mathcal{I}$: all negative
- $\Gamma \vdash \Gamma'$
- Γ: $[P(x)],\ [Q(y)],\ x \not\equiv c \rhd \neg P(f(x)) \vee \neg Q(g(x)) \vee [R(x)],\ [\neg R(c)],\ \neg P(f(c)) \vee \neg Q(g(c)) \vee [R(c)]$
- Γ': $[P(x)],\ [Q(y)],\ x \not\equiv c \rhd \neg P(f(x)) \vee \neg Q(g(x)) \vee [R(x)],\ [\neg R(c)],\ \neg P(f(c)) \vee [\neg Q(g(c))]$

Splitting inferences

- Replace a clause by its partition
- Partition of a clause: a set of clauses that capture the same cgi's, and have disjoint selected literals (no cgi's with the same atoms)

- Clause: $true \rhd P(x, y)$ (or simply $P(x, y)$)
- Partition: $true \rhd P(f(z), y)$, $top(x) \neq f \rhd P(x, y)$

Example of splitting inference

- $\Gamma \vdash \Gamma'$
- Γ: $[P(x)],\ [Q(y)],\ x \not\equiv c \rhd \neg P(f(x)) \vee \neg Q(g(x)) \vee [R(x)],\ [\neg R(c)],\ \neg P(f(c)) \vee [\neg Q(g(c))]$
- Γ': $[P(x)],\ top(y) \not\equiv g \rhd [Q(y)],\ z \not\equiv c \rhd [Q(g(z))],\ [Q(g(c))],\ x \not\equiv c \rhd \neg P(f(x)) \vee \neg Q(g(x)) \vee [R(x)],\ [\neg R(c)],\ \neg P(f(c)) \vee [\neg Q(g(c))]$

Deletion of disposable clauses

- pcgi's: cgi's of selected literal that can be added to current candidate model
- ccgi's: cgi's of selected literal that contradict current candidate model:
  - cgi of clause not satisfied by induced partial interpretation
  - cgi of selected literal appears negated in induced partial interpretation
- A clause with neither is useless for model search in SGGS
- Disposable: (non-empty) clause with neither pcgi's nor ccgi's
- When deleted, all clauses depending on it also deleted

Inference control

- Bundled derivations: all inferences are bundled
- Bundled inferences: macro-inferences, e.g.: an SGGS-extension followed by a series of SGGS-resolutions until an $\mathcal{I}$-all-true resolvent is generated
  - An $\mathcal{I}$-all-true clause with selected literal not assigned encodes a lemma
  - An $\mathcal{I}$-all-true clause with selected literal assigned contradicts current candidate model, moves to the left, and amends it by SGGS-resolution (implicit backtracking)

Refutational completeness

- S: input set of clauses
- S unsatisfiable: any fair SGGS-derivation terminates with refutation
- S satisfiable: derivation may be infinite; its limiting sequence represents model

Proof of refutational completeness: building blocks

- A convergence ordering $>^c$ on clause sequences: ensures that there is no infinite descending chain of sequences of bounded length
- A notion of fairness for SGGS-derivations: ensures that the procedure does not get stuck working on longer prefixes when shorter ones can be reduced
- A notion of limiting sequence for SGGS-derivations: every prefix stabilizes eventually

Convergence and decreasingness theorems

- Convergence theorem:
  A derivation that is a non-ascending chain admits limiting sequence
- Decreasingness theorem:
  A bundled derivation forms a non-ascending chain

Completeness theorem

Theorem:
For all initial interpretations $\mathcal{I}$ and sets S of first-order clauses, if S is unsatisfiable, any fair bundled SGGS-derivation is a refutation

Idea of proof:
If not, infinitely many irredundant SGGS-extensions apply; infinite derivation with infinite limiting sequence, that gets reduced in a finite prefix that had already converged: contradiction

Goal sensitivity I

- $\mathcal{I} \models T$ and $\mathcal{I} \not\models iSOS$
- Two ground clauses connected: complementary literals
- Goal-relevant clauses: closure of the set of ground instances of clauses in *iSOS* wrt connection and resolution
- Γ is goal-relevant if all ground instances of all its clauses are

Goal sensitivity II

Theorem: SGGS only generates goal-relevant clause sequences

Idea of proof:
use assignments of $\mathcal{I}$-true literals to $\mathcal{I}$-false ones to connect literals

Summary

SGGS is simultaneously

- First order
- DPLL-style model based
- Proof confluent
- Semantically guided
- Refutationally complete
- Goal sensitive

Future work

- SGGS as an abstract transition system
- Practical inference control (e.g., partitioning inferences)
- Implementation
- Non-trivial initial interpretations
- SGGS for model building and decision procedures
- Extension to equality and theory reasoning

Towards a semantically-oriented style of theorem proving which may pay off for hard problems or new domains

References

- Constraint manipulation in SGGS. 28th Workshop on Unification (UNIF), Vienna, July 2014.
- SGGS theorem proving: an exposition. 4th Workshop on Practical Aspects in Automated Reasoning (PAAR), Vienna, July 2014.
- Model representation by SGGS clause sequences. Submitted, 1–24.
- Semantically-guided goal-sensitive theorem proving. Technical Report 92/2014, Dipartimento di Informatica, Università degli Studi di Verona, Jan. 2014, revised July 2014, 1–58.