SGGS: conflict-driven first-order reasoning[1]

Maria Paola Bonacina

Dipartimento di Informatica, Università degli Studi di Verona,
Verona, Italy, EU

Invited talk at the 1st European Workshop on Higher-Order Automated Reasoning (Matryoshka),
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, EU, June 2018

(And 2nd half of a one-day tutorial on "Conflict-driven reasoning," LORIA Nancy, France, EU, February 2019)

[1] Joint work with David A. Plaisted

Logical methods for machine intelligence

- Theorem provers for higher-order (HO) reasoning
- Theorem provers for first-order (FO) reasoning
- Solvers for satisfiability modulo theories (SMT)
- Solvers for satisfiability in propositional logic (SAT)
- ....

- Traditionally: HO provers supported by solvers
- Matryoshka: HO provers supported by FO provers

Motivation

- Objective: automated reasoning in first-order logic (FOL)
- Observation: Conflict-Driven Clause Learning (CDCL) played a key role in bringing SAT-solving from theoretical hardness to practical success
  [Marques-Silva, Sakallah: ICCAD 1996, IEEE Trans. on Computers 1999], [Moskewicz, Madigan, Zhao, Zhang, Malik: DAC 2001]
  [Marques-Silva, Lynce, Malik: SAT Handbook 2009]
- Question: Can we lift CDCL to FOL?
- Answer: Semantically-Guided Goal-Sensitive (SGGS) reasoning

The big picture: conflict-driven reasoning

- For SAT: Conflict-Driven Clause Learning (CDCL)
- For several fragments of arithmetic: conflict-driven $\mathcal{T}$-satisfiability procedures
- For SMT: Model Constructing Satisfiability (MCSAT) [Jovanović, de Moura: VMCAI 2013], [Jovanović, Barrett, de Moura: FMCAD 2013]
- For SMT with combination of theories and SMA: Conflict-Driven Satisfiability (CDSAT) [Bonacina, Graham-Lengrand, Shankar: CADE 2017, CPP 2018]
- For FOL: Semantically-Guided Goal-Sensitive (SGGS) reasoning

Model representation in FOL

- Clauses have universally quantified variables: $\neg P(x) \vee R(x, g(x, y))$
- $P(x)$ has infinitely many ground instances: $P(a)$, $P(f(a))$, $P(f(f(a)))$ ...
- Infinitely many interpretations where each ground instance is either true or false
- What do we guess?! How do we get started?!
- Answer: Semantic guidance

Semantic guidance

- Take $\mathcal{I}$ with all positive ground literals true
- $\mathcal{I} \models S$: done! $\mathcal{I} \not\models S$: modify $\mathcal{I}$ to satisfy S
- How? Flipping literals from positive to negative
- Flipping $P(f(x))$ flips $P(f(a))$, $P(f(f(a)))$ ... at once, but not $P(a)$
- SGGS discovers which negative literals are needed
- Initial interpretation $\mathcal{I}$: starting point in the search for a model and default interpretation

Uniform falsity

- Propositional logic: if P is true (e.g., it is in the trail), $\neg P$ is false; if P is false, $\neg P$ is true
- First-order logic: if $P(x)$ is true, $\neg P(x)$ is false, but if $P(x)$ is false, we only know that there is a ground instance $P(t)$ such that $P(t)$ is false and $\neg P(t)$ is true
- Uniform falsity: Literal L is uniformly false in an interpretation $\mathcal{J}$ if all ground instances of L are false in $\mathcal{J}$
- If $P(x)$ is true in $\mathcal{J}$, $\neg P(x)$ is uniformly false in $\mathcal{J}$
  If $P(x)$ is uniformly false in $\mathcal{J}$, $\neg P(x)$ is true in $\mathcal{J}$

Truth and uniform falsity in the initial interpretation

- $\mathcal{I}$-true: true in $\mathcal{I}$
- $\mathcal{I}$-false: uniformly false in $\mathcal{I}$
- If L is $\mathcal{I}$-true, $\neg L$ is $\mathcal{I}$-false
  if L is $\mathcal{I}$-false, $\neg L$ is $\mathcal{I}$-true
- $\mathcal{I}$ all negative: negative literals are $\mathcal{I}$-true, positive literals are $\mathcal{I}$-false
- $\mathcal{I}$ all positive: positive literals are $\mathcal{I}$-true, negative literals are $\mathcal{I}$-false

SGGS clause sequence

- Γ: sequence of clauses
  Every literal in Γ is either $\mathcal{I}$-true or $\mathcal{I}$-false (invariant)
- SGGS-derivation: $\Gamma_0 \vdash \Gamma_1 \vdash \ldots \Gamma_i \vdash \Gamma_{i+1} \vdash \ldots$
- In every clause in Γ a literal is selected:
  $C = L_1 \vee L_2 \vee \ldots \vee L \vee \ldots \vee L_n$ denoted $C[L]$
- $\mathcal{I}$-false literals are preferred for selection (to change $\mathcal{I}$)
- An $\mathcal{I}$-true literal is selected only in a clause whose literals are all $\mathcal{I}$-true: $\mathcal{I}$-all-true clause

Examples

- $\mathcal{I}$: all negative
- A sequence of unit clauses:
  $[P(a,x)], [P(b,y)], [\neg P(z,z)], [P(u,v)]$
- A sequence of non-unit clauses:
  $[P(x)],\ \neg P(f(y)) \vee [Q(y)],\ \neg P(f(z)) \vee \neg Q(g(z)) \vee [R(f(z),g(z))]$
- A sequence of constrained clauses:
  $[P(x)],\ top(y) \neq g \rhd [Q(y)],\ z \not\equiv c \rhd [Q(g(z))]$

Candidate partial model represented by Γ

- Get a partial model $\mathcal{I}^p(\Gamma)$ by consulting Γ from left to right
- Have each clause $C_k[L_k]$ contribute the ground instances of L_k that satisfy ground instances of C_k not satisfied thus far
- Such ground instances are called proper
- Literal selection in SGGS corresponds to decision in CDCL

Candidate partial model represented by Γ

- If Γ is empty, $\mathcal{I}^p(\Gamma)$ is empty
- $\Gamma|_{k-1}$: prefix of length $k-1$
- If $\Gamma = C_1[L_1], \ldots, C_k[L_k]$, and $\mathcal{I}^p(\Gamma|_{k-1})$ is the partial model represented by $C_1[L_1], \ldots, C_{k-1}[L_{k-1}]$, then $\mathcal{I}^p(\Gamma)$ is $\mathcal{I}^p(\Gamma|_{k-1})$ plus the ground instances $L_k\sigma$ such that
  - $C_k\sigma$ is ground
  - $\mathcal{I}^p(\Gamma|_{k-1}) \not\models C_k\sigma$
  - $\neg L_k\sigma \notin \mathcal{I}^p(\Gamma|_{k-1})$

  $L_k\sigma$ is a proper ground instance

Example

- Sequence Γ: $[P(a,x)], [P(b,y)], [\neg P(z,z)], [P(u,v)]$
- Partial model $\mathcal{I}^p(\Gamma)$:
  $\mathcal{I}^p(\Gamma) \models P(a,t)$ for all ground terms t
  $\mathcal{I}^p(\Gamma) \models P(b,t)$ for all ground terms t
  $\mathcal{I}^p(\Gamma) \models \neg P(t,t)$ for t other than a and b
  $\mathcal{I}^p(\Gamma) \models P(s,t)$ for all distinct ground terms s and t

Candidate model represented by Γ

Consult first $\mathcal{I}^p(\Gamma)$ then $\mathcal{I}$:

- Ground literal L
- Determine whether $\mathcal{I}[\Gamma] \models L$:
  - If $\mathcal{I}^p(\Gamma)$ determines the truth value of L: $\mathcal{I}[\Gamma] \models L$ iff $\mathcal{I}^p(\Gamma) \models L$
  - Otherwise: $\mathcal{I}[\Gamma] \models L$ iff $\mathcal{I} \models L$
- $\mathcal{I}[\Gamma]$ is $\mathcal{I}$ modified to satisfy the clauses in Γ by satisfying the proper ground instances of their selected literals
- $\mathcal{I}$-false selected literals makes the difference

Example (continued)

- $\mathcal{I}$: all negative
- Sequence Γ: $[P(a,x)], [P(b,y)], [\neg P(z,z)], [P(u,v)]$
- Represented model $\mathcal{I}[\Gamma]$:
  $\mathcal{I}[\Gamma] \models P(a,t)$ for all ground terms t
  $\mathcal{I}[\Gamma] \models P(b,t)$ for all ground terms t
  $\mathcal{I}[\Gamma] \models \neg P(t,t)$ for t other than a and b
  $\mathcal{I}[\Gamma] \models P(s,t)$ for all distinct ground terms s and t
  $\mathcal{I}[\Gamma] \not\models L$ for all other positive literals L

Disjoint prefix

The disjoint prefix $dp(\Gamma)$ of Γ is

- The longest prefix of Γ where every selected literal contributes to $\mathcal{I}[\Gamma]$ all its ground instances
- That is, where all ground instances are proper
- No two selected literals in the disjoint prefix intersect
- Intuitively, a polished portion of Γ

Examples

$[P(a,x)], [P(b,y)], [\neg P(z,z)], [P(u,v)]$:
the disjoint prefix is $[P(a,x)], [P(b,y)]$

$[P(x)],\ \neg P(f(y)) \vee [Q(y)],\ \neg P(f(z)) \vee \neg Q(g(z)) \vee [R(f(z), g(z))]$:
the disjoint prefix is the whole sequence

$[P(x)],\ top(y) \neq g \rhd [Q(y)],\ z \not\equiv c \rhd [Q(g(z))]$:
the disjoint prefix is the whole sequence

First-order clausal propagation

- Consider literal M selected in clause C_j in Γ, and literal L in C_i, $i > j$:
  $\ldots, \ldots \vee \ldots [M] \ldots \vee \ldots, \ldots, \ldots \vee \ldots L \ldots \vee \ldots, \ldots$
  If all ground instances of L appear negated among the proper ground instances of M, L is uniformly false in $\mathcal{I}[\Gamma]$
- L depends on M, like $\neg L$ depends on L in propositional clausal propagation when L is in the trail
- Since every literal in Γ is either $\mathcal{I}$-true or $\mathcal{I}$-false, M will be one and L the other

Example

- $\mathcal{I}$: all negative
- Sequence Γ:
  $[P(x)]$, $\neg P(f(y)) \vee [Q(y)]$, $\neg P(f(z)) \vee \neg Q(g(z)) \vee [R(f(z), g(z))]$
- $\neg P(f(y))$ depends on $[P(x)]$
- $\neg P(f(z))$ depends on $[P(x)]$
- $\neg Q(g(z))$ depends on $[Q(y)]$

First-order clausal propagation

- Conflict clause:
  $L_1 \vee L_2 \vee \ldots \vee L_n$
  all literals are uniformly false in $\mathcal{I}[\Gamma]$
- Unit clause:
  $C = L_1 \vee L_2 \vee \ldots \vee L_j \vee \ldots \vee L_n$
  all literals but one (L_j) are uniformly false in $\mathcal{I}[\Gamma]$
- Implied literal: L_j with $C[L_j]$ as justification

Semantically-guided first-order clausal propagation

- SGGS employs assignments to keep track of the dependences of $\mathcal{I}$-true literals on selected $\mathcal{I}$-false literals
- An assigned literal is true in $\mathcal{I}$ and uniformly false in $\mathcal{I}[\Gamma]$
- Non-selected $\mathcal{I}$-true literals are assigned (invariant)
- Selected $\mathcal{I}$-true literals are assigned if possible
- $\mathcal{I}$-all-true clauses in Γ are either conflict clauses or justifications with their selected literal as implied literal
- All justifications are in the disjoint prefix

How does SGGS build clause sequences?

- Inference rule: SGGS-extension
- $\mathcal{I}[\Gamma] \not\models C$ for some clause $C \in S$
- $\mathcal{I}[\Gamma] \not\models C'$ for some ground instance C' of C
- Then SGGS-extension uses Γ and C to generate a (possibly constrained) clause $A \triangleright E$ such that
  - E is an instance of C
  - C' is a ground instance of $A \triangleright E$

  and adds it to Γ to get Γ'

How can a ground literal be false

$\mathcal{I}[\Gamma] \not\models C'$ (C' ground instance of $C \in S$)
Each literal L of C' is false in $\mathcal{I}[\Gamma]$:

- Either L is $\mathcal{I}$-true and it depends on an $\mathcal{I}$-false selected literal in Γ
- Or L is $\mathcal{I}$-false and it depends on an $\mathcal{I}$-true selected literal in Γ
- Or L is $\mathcal{I}$-false and not interpreted by $\mathcal{I}^p(\Gamma)$

SGGS-extension

- Clause $C \in S$: main premise
- Unify literals $L_1, \ldots, L_n$ $(n \geq 1)$ of C with $\mathcal{I}$-false selected literals $M_1, \ldots, M_n$ of opposite sign in $dp(\Gamma)$: most general unifier α
- Clauses where the $M_1, \ldots, M_n$ are selected: side premises
- Generate instance $C\alpha$ called extension clause

SGGS-extension

- $L_1\alpha, \ldots, L_n\alpha$ are $\mathcal{I}$-true and all other literals of $C\alpha$ are $\mathcal{I}$-false
- $M_1, \ldots, M_n$ are the selected literals that make the $\mathcal{I}$-true literals of C' false in $\mathcal{I}[\Gamma]$
- Assign the $\mathcal{I}$-true literals of $C\alpha$ to the side premises
- $M_1, \ldots, M_n$ are $\mathcal{I}$-false but true in $\mathcal{I}[\Gamma]$: instance generation is guided by the current model $\mathcal{I}[\Gamma]$

Example

- S contains $\{P(a), \neg P(x) \vee Q(f(y)), \neg P(x) \vee \neg Q(z)\}$
- $\mathcal{I}$: all negative
- Γ_0 is empty
  $\mathcal{I}[\Gamma_0] = \mathcal{I} \not\models P(a)$
- $\Gamma_1 = [P(a)]$ with α empty
- $\mathcal{I}[\Gamma_1] \not\models \neg P(x) \vee Q(f(y))$
- $\Gamma_2 = [P(a)],\ \neg P(a) \vee [Q(f(y))]$
  with $\alpha = \{x \leftarrow a\}$

How can a ground clause be false

$\mathcal{I}[\Gamma] \not\models C'$:

- Either C' is $\mathcal{I}$-all-true: all its literals depend on selected $\mathcal{I}$-false literals in Γ;
  C' is instance of an $\mathcal{I}$-all-true conflict clause
- Or C' has $\mathcal{I}$-false literals and all of them depend on selected $\mathcal{I}$-true literals in Γ;
  C' is instance of a non-$\mathcal{I}$-all-true conflict clause
- Or C' has $\mathcal{I}$-false literals and at least one of them is not interpreted by $\mathcal{I}^p(\Gamma)$: C' is a proper ground instance of C

Three kinds of SGGS-extension

The extension clause is

- Either an $\mathcal{I}$-all-true conflict clause: need to solve the conflict
- Or a non-$\mathcal{I}$-all-true conflict clause: need to explain and solve the conflict
- Or a clause that is not in conflict and extends $\mathcal{I}[\Gamma]$ into $\mathcal{I}[\Gamma']$ by adding the proper ground instances of its selected literal

Example (continued)

- S contains $\{P(a), \neg P(x) \vee Q(f(y)), \neg P(x) \vee \neg Q(z)\}$
- $\mathcal{I}$: all negative
- After two non-conflicting SGGS-extensions:
  $\Gamma_2 = [P(a)],\ \neg P(a) \vee [Q(f(y))]$
- $\mathcal{I}[\Gamma_2] \not\models \neg P(x) \vee \neg Q(z)$
- $\Gamma_3 = [P(a)],\ \neg P(a) \vee [Q(f(y))],\ \neg P(a) \vee [\neg Q(f(w))]$ with $\alpha = \{x \leftarrow a, z \leftarrow f(y)\}$ plus renaming
- Conflict! with $\mathcal{I}$-all-true conflict clause

First-order conflict explanation: SGGS-resolution

- It resolves a non-$\mathcal{I}$-all-true conflict clause E with a justification $D[M]$
- The literals resolved upon are an $\mathcal{I}$-false literal L of E and the $\mathcal{I}$-true selected literal M that L depends on

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': $[P(x)],\ [Q(y)],\ x \not\equiv c \rhd \neg P(f(x)) \vee \neg Q(g(x)) \vee [R(x)],\ [\neg R(c)],\ \neg P(f(c)) \vee [\neg Q(g(c))]$

First-order conflict explanation: SGGS-resolution

- Each resolvent is still a conflict clause and it replaces the previous conflict clause in Γ
- SGGS-resolution corresponds to resolution in CDCL
- It continues until all $\mathcal{I}$-false literals in the conflict clause have been resolved away and it gets either $\square$ or an $\mathcal{I}$-all-true conflict clause
- If $\square$ arises, S is unsatisfiable

First-order conflict-solving: SGGS-move

- It moves the $\mathcal{I}$-all-true conflict clause $E[L]$ to the left of the clause $D[M]$ such that L depends on M
- It flips at once from false to true the truth value in $\mathcal{I}[\Gamma]$ of all ground instances of L
- The conflict is solved, L is implied, $E[L]$ is satisfied, it becomes the justification of L and it enters the disjoint prefix
- SGGS-move corresponds to learn and backjump in CDCL

Example (continued)

- S contains $\{P(a), \neg P(x) \vee Q(f(y)), \neg P(x) \vee \neg Q(z)\}$
- $\mathcal{I}$: all negative
- $\Gamma_3 = [P(a)],\ \neg P(a) \vee [Q(f(y))],\ \neg P(a) \vee [\neg Q(f(w))]$
- $\Gamma_4 = [P(a)],\ \neg P(a) \vee [\neg Q(f(w))],\ \neg P(a) \vee [Q(f(y))]$
- $\Gamma_5 = [P(a)],\ \neg P(a) \vee [\neg Q(f(w))],\ [\neg P(a)]$
- $\Gamma_6 = [\neg P(a)],\ [P(a)],\ \neg P(a) \vee [\neg Q(f(w))]$
- $\Gamma_7 = [\neg P(a)],\ \square,\ \neg P(a) \vee [\neg Q(f(w))]$
- Refutation!

Further elements

- There's more to SGGS: first-order literals may intersect having ground instances with the same atom
- SGGS uses partitioning inference rules to partition clauses and isolate intersections that can then be removed by SGGS-resolution (different sign) or SGGS-deletion (same sign)
- Partitioning introduces constraints that are a kind of Herbrand constraints (e.g., $x \not\equiv y \rhd P(x, y)$, $top(y) \neq g \rhd Q(y)$)
- SGGS-deletion removes $C_k[L_k]$ satisfied by $\mathcal{I}^p(\Gamma|_{k-1})$: model-based redundancy

SGGS makes progress: fairness

- If $\mathcal{I}[\Gamma] \not\models C$ for some clause $C \in S$ and $\Gamma = dp(\Gamma)$, SGGS-extension applies to Γ
- If $\Gamma \neq dp(\Gamma)$, an SGGS inference rule other than SGGS-extension applies to Γ
- Every conflicting SGGS-extension is bundled with explanation by SGGS-resolution and conflict solving by SGGS-move
- Fairness also ensures that the procedure does not ignore inferences on shorter prefixes to work on longer ones

SGGS: Semantically-Guided Goal-Sensitive reasoning

- SGGS lifts CDCL to first-order logic (FOL)
- S: input set of clauses
- Refutationally complete: if S is unsatisfiable, SGGS generates a refutation
- Model-complete: if S is satisfiable, the limit of the derivation (which may be infinite) is a model

Initial interpretation $\mathcal{I}$

- All negative (as in positive hyperresolution)
- All positive (as in negative hyperresolution)
- Goal-sensitive interpretation:
  - $S = T \uplus SOS$ where SOS contains the clauses in the clausal form of the negation of the conjecture
  - $S = T \uplus SOS$ where T is the largest consistent subset

  If $\mathcal{I} \not\models SOS$ and $\mathcal{I} \models T$ then SGGS is goal-sensitive: all generated clauses deduced from SOS
- $\mathcal{I}$ satisfies the axioms of a theory $\mathcal{T}$

Current and future work

- Implementation of SGGS: algorithms and strategies
- Heuristic choices: literal selection, assignments
- Simpler SGGS? More contraction?
- Extension to equality
- Initial interpretations not based on sign
- SGGS for decision procedures for decidable fragments
- SGGS for FOL model building

References for SGGS

- Semantically-guided goal-sensitive reasoning: inference system and completeness. *Journal of Automated Reasoning*, 59(2):165–218, August 2017.
- Semantically-guided goal-sensitive reasoning: model representation. *Journal of Automated Reasoning* 56(2):113–141, February 2016.
- SGGS theorem proving: an exposition. 4th Workshop on Practical Aspects in Automated Reasoning (PAAR), Vienna, July 2014. EPiC 31:25-38, July 2015.
- Constraint manipulation in SGGS. 28th Workshop on Unification (UNIF), Vienna, July 2014. TR 14-06, RISC, 47–54, 2014.

Thanks

Thank you!