Clause Learning for Modular Systems

David Mitchell and Eugenia Ternovska

LPNMR 2015

Outline

Motivation

Modern Problem Solving:

- Many solvers, KBs.
- Accessed via API
- Solution is a structure all agree with

Goal:

- CDCL-like algorithms for this setting
- Problems specified with Algebra of Modular Systems
  - Module: class of structures
  - Operations like Relational Algebra lifted to classes of structures
  - (See Ternovska, GTTV '15).

Assumptions for Today

- Conjunctions of Modules
- Response times highly variable
  $\Rightarrow$ need asynchronous solving
- Module M_i:
  - set of assignments for propositional vocabulary σ_i
  - queried with a partial assignment
  - responds (in finite time) with one of:
    - $\langle Accept, A \rangle$
    - $\langle Reject, A \rangle$

  where A is a set of clauses with $M_i \models A$.

CDCL (a bit abstractly)

Input: Clause set Φ
Output: SAT or UNSAT
$\Gamma \leftarrow \Phi$ // Clause set, initialized to the input clauses. 1
$\delta \leftarrow \langle\rangle$ // Decision sequence, initialized to empty. 2
repeat 3
  $\Gamma, \delta \leftarrow$ **Extend-and-Learn**(Γ, δ) 4
  if $\square \in \Gamma$ **then** 5
    return *UNSAT* 6
  if $\delta \models \Gamma$ **then** 7
    return *SAT* 8
end 9

CDCL → CDCL-AMS

Input: Modular System $\mathcal{M}$ with vocabulary σ
Output: SAT or UNSAT

$\Gamma \leftarrow \emptyset$ // Clause set, initialized to empty. 1
$\delta \neq \langle\rangle$ // Initial decision sequence nonempty. 2
repeat 3
  $\Gamma, \delta \Leftarrow$ *Modified by Module Response* 4
  $\Gamma, \delta \leftarrow$ **Extend-and-Learn**(Γ, δ) 5
  if $\square \in \Gamma$ **then** 6
    return *UNSAT* 7
  if $\delta \models \Gamma$ **then** 8
    $\delta \Rightarrow$ *Sent Modules* 9
end 10

CDCL-AMS Data

Γ:

- current set of clauses ($\mathcal{M} \models \Gamma$)

Query:

- A CDCL Assignment stack,
- For each M_i, label on prefix accepted by M_i

QUERIES:

- Set of queries waiting to be sent to a module
- Each satisfied Γ when added

HOLD:

- Clauses corresponding to members of QUERIES
- Used in UP, but not in conflict clause deriviation

CONTINUE:

- module responses waiting to be handled

CDCL-AMS

Input: Modular System $\mathcal{M}$ with vocabulary σ
Output: SAT or UNSAT

$\Gamma \leftarrow \emptyset$ // Clause set, initialized to empty. 1
$\delta \neq \langle\rangle$ // Initial decision sequence nonempty. 2
repeat 3
  if CONTINUE *not empty* **then** 4
    Remove a response from CONTINUE 5
    Update Γ,δ and HOLD 6
    if $\delta \models \mathcal{M}$ **then return** *SAT* 7
  $\Gamma, \delta \leftarrow$ **Extend-and-Learn**(Γ, δ) 8
  if $\square \in \Gamma$ **then return** *UNSAT* 9
  if $\delta \models \Gamma$ **then** 10
    Add δ to QUEUE, Decisions(δ) to HOLD 11
    $\delta \leftarrow$ a proper prefix of δ 12
end 13

Correctness and Complexity

Partial Correctness:

- CDCL-AMS returns SAT
  $\Rightarrow \delta$ total for and accepted by every module
- CDCL-AMS returns UNSAT
  $\Rightarrow \mathcal{M} \models \Gamma$ and $\Gamma \models \square$, so $\mathcal{M}$ has no solution.

Termination: Progress is

1. δ is extended, or
2. A solver accepted a larger prefix of δ
3. δ was "killed" by a new clause

Complexity: $O(T(n)2^n)$

- $n = |\sigma|$
- $T(n)$ is max response time for a module.

Work in Progress

- Refinements, Heuristics, Implementability issues
- Versions for:
  1. Modules which are Expanders
  2. Modules which return a limited set of alternatives
  3. Systems over full Modular System Algebra
  4. Modules which are Dynamic
  5. Exploiting Problem Structure
  6. Exploiting Specific Modules