DPLL($\Gamma + \mathcal{T}$): a new style of reasoning

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy

June, 2013
A new style of reasoning: $\text{DPLL}(\Gamma + \mathcal{T})$

Speculative inferences for decision procedures
Problem statement

- Determine *validity* (*unsatisfiability*) or *invalidity* (*satisfiability*) of first-order formulæ
- Modulo *background theories* (some arithmetic is a must)
- With *quantifiers* for expressivity: QFF do not suffice
- Emphasis on *automation*: prover called by other tools
Some key state-of-the-art reasoning methods

- Davis-Putnam-Logemann-Loveland (DPLL) procedure for SAT
- $T_i$-solvers: *Satisfiability procedures* for the $T_i$’s
- DPLL($T$)-based SMT-solver: *Decision procedure* for $T$ with combination by *equality sharing* of the $T_i$-sat procedures
- First-order engine $\Gamma$ to handle $\mathcal{R}$ (additional theory): *Resolution*+*Rewriting*+*Superposition*: *Superposition-based*
How to combine their strengths?

- **DPLL**: SAT-problems; large non-Horn clauses
- **Theory solvers**: e.g., ground equality, linear arithmetic
- **DPLL(\(T\))**-based SMT-solver: efficient, scalable, integrated theory reasoning
- **Superposition-based inference system \(\Gamma\)**:
  - FOL+= clauses with *universally quantified variables* 
    (*automated* instantiation)
  - Sat-procedure for several theories of data structures 
    (e.g., lists, arrays, records)
Shape of problem

- Background theory $\mathcal{T}$
  - $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$, e.g., linear arithmetic
- Set of formulæ: $\mathcal{R} \cup \mathcal{P}$
  - $\mathcal{R}$: set of non-ground clauses without $\mathcal{T}$-symbols
  - $\mathcal{P}$: large ground formula (set of ground clauses) typically with $\mathcal{T}$-symbols
Combination of theories

- If \( \Gamma \) terminates on \( R_i \)-sat problems, it terminates on \( R \)-sat problems for \( R = \bigcup_{i=1}^{n} R_i \), if \( R_i \)’s disjoint and variable-inactive

- Variable-inactivity: no maximal literal \( t \simeq x \) where \( x \notin \text{Var}(t) \) (no superposition from variables)

- Only inferences across theories: superpositions from shared constants

- Variable inactivity implies stable infiniteness:
  \( \Gamma \) reveals lack of stable infiniteness by generating cardinality constraint (e.g., \( y \simeq x \lor y \simeq z \)) not variable-inactive
Propositional logic, ground problems in built-in theories

- Build candidate model $M$
- Decision procedure:
  - model found: return $sat$;
  - failure: return $unsat$

- Backtracking
DPLL(\mathcal{T})

State of derivation: \( M \parallel F \)

- \( \mathcal{T}\text{-Propagate} \): add to \( M \) an \( L \) that is \( \mathcal{T}\)-consequence of \( M \)
- \( \mathcal{T}\text{-Conflict} \): detect that \( L_1, \ldots, L_n \) in \( M \) are \( \mathcal{T}\)-inconsistent

If \( T_i \)-solver builds \( T_i \)-model (model-based theory combination):

- \( \text{PropagateEq} \): add to \( M \) a ground \( s \simeq t \) true in \( T_i \)-model
DPLL(Γ+Γ): integrate Γ in DPLL(Γ)

- **Idea:** literals in $M$ can be premises of Γ-inferences
- Stored as *hypotheses* in inferred clause
- *Hypothetical clause:* $(L_1 \land \ldots \land L_n) \triangleright (L'_1 \lor \ldots \lor L'_m)$
  interpreted as $\neg L_1 \lor \ldots \lor \neg L_n \lor L'_1 \lor \ldots \lor L'_m$
- Inferred clauses inherit hypotheses from premises
State of derivation: $M \parallel F$

- **Expansion**: take as premises *non-ground* clauses from $F$ and $R$-literals (unit clauses) from $M$ and add result to $F$
- **Backjump**: remove hypothetical clauses depending on undone assignments
- **Contraction**: as above + *scope level* to prevent situation where clause is deleted, but clauses that make it redundant are gone because of backjumping
DPLL(Γ+T): expansion inferences

- **Deduce**: Γ-rule $\gamma$, e.g., superposition, using *non-ground* clauses $\{H_1 \triangleright C_1, \ldots, H_m \triangleright C_m\}$ in $F$ and ground $R$-literals $\{L_{m+1}, \ldots, L_n\}$ in $M$

  $$M \parallel F \implies M \parallel F, H \triangleright C$$

  where $H = H_1 \cup \ldots \cup H_m \cup \{L_{m+1}, \ldots, L_n\}$

  and $\gamma$ infers $C$ from $\{C_1, \ldots, C_m, L_{m+1}, \ldots, L_n\}$

- Only $R$-literals: Γ-inferences ignore $T$-literals

- Take ground unit $R$-clauses from $M$ as PropagateEq puts them there
DPLL($\Gamma^+T$): contraction inferences

- Single premise $H \triangleright C$: apply to $C$ (e.g., tautology deletion)
- Multiple premises (e.g., subsumption, simplification): prevent situation where clause is deleted, but clauses that make it redundant are gone because of backjumping
- **Scope level:**
  - $level(L)$ in $M L M'$: number of decided literals in $M L$
  - $level(H) = \max\{level(L) \mid L \in H\}$ and 0 for $\emptyset$
DPLL(Γ+T): contraction inferences

- Say we have $H \triangleright C$, $H_2 \triangleright C_2$, $\ldots$, $H_m \triangleright C_m$, and $L_{m+1}, \ldots, L_n$
- $C_2, \ldots, C_m, L_{m+1}, \ldots, L_n$ simplify $C$ to $C'$ or subsume it
- Let $H' = H_2 \cup \ldots \cup H_m \cup \{L_{m+1}, \ldots, L_n\}$
- Simplification: replace $H \triangleright C$ by $(H \cup H') \triangleright C'$
- Both simplification and subsumption:
  - if $level(H) \geq level(H')$: delete
  - if $level(H) < level(H')$: disable (re-enable when backjumping $level(H')$)
DPLL(Γ+T) as a transition system

- Search mode: State of derivation $M \models F$
  - $M$ sequence of assigned ground literals: partial model
  - $F$ set of hypothetical clauses
- Conflict resolution mode: State of derivation $M \models F \models C$
  - $C$ ground conflict clause

Initial state: $M$ empty, $F$ is $\{\emptyset \triangleright C \mid C \in \mathcal{R} \cup P\}$
Completeness of $\text{DPLL}(\Gamma+\mathcal{T})$

- **Refutational completeness** of the inference system:
  - from that of $\Gamma$, $\text{DPLL}(\mathcal{T})$ and equality sharing
  - made combinable by variable-inactivity
- **Fairness** of the search plan:
  - depth-first search fair only for ground SMT problems;
  - add *iterative deepening* on *inference depth*
DPLL(Γ+T): Summary

Use each engine for what is best at:

- DPLL(T) works on ground clauses
- Γ not involved with ground inferences and built-in theory
- Γ works on non-ground clauses and ground unit clauses taken from \( M \): inferences guided by current partial model
- Γ works on \( R \)-sat problem
Speculative inferences for decision procedures
How to get decision procedures?

- SW development: false conjectures due to mistakes in implementation or specification
- Need theorem prover that terminates on satisfiable inputs
- Not possible in general:
  - FOL is only semi-decidable
  - First-order formulæ of linear arithmetic with uninterpreted functions: not even semi-decidable

However we need less than a general solution.
Problematic axioms do occur in relevant inputs

Example:

1. $\neg (x \sqsupseteq y) \lor f(x) \sqsupseteq f(y)$ (Monotonicity)
2. $a \sqsupseteq b$ generates by resolution
3. $\{f^i(a) \sqsupseteq f^i(b)\}_{i \geq 0}$

E.g. $f(a) \sqsupseteq f(b)$ or $f^2(a) \sqsupseteq f^2(b)$ often suffice to show satisfiability
Idea: Allow speculative inferences

1. $\neg(x \sqsubseteq y) \lor f(x) \sqsubseteq f(y)$
2. $a \sqsubseteq b$
3. $a \sqsubseteq f(c)$
4. $\neg(a \sqsubseteq c)$
Idea: Allow speculative inferences

1. \( \neg(x \sqsubseteq y) \lor f(x) \sqsubseteq f(y) \)
2. \( a \sqsubseteq b \)
3. \( a \sqsubseteq f(c) \)
4. \( \neg(a \sqsubseteq c) \)

1. Add \( f(x) \simeq x \)
2. Rewrite \( a \sqsubseteq f(c) \) into \( a \sqsubseteq c \) and get \( \Box \): backtrack!
Idea: Allow speculative inferences

1. $\neg(x \sqsubseteq y) \lor f(x) \sqsubseteq f(y)$
2. $a \sqsubseteq b$
3. $a \sqsubseteq f(c)$
4. $\neg(a \sqsubseteq c)$

1. Add $f(x) \simeq x$
2. Rewrite $a \sqsubseteq f(c)$ into $a \sqsubseteq c$ and get $\Box$: backtrack!
3. Add $f(f(x)) \simeq x$
4. $a \sqsubseteq b$ yields only $f(a) \sqsubseteq f(b)$
5. $a \sqsubseteq f(c)$ yields only $f(a) \sqsubseteq c$
6. Terminate and detect satisfiability
Speculative inferences in \( \text{DPLL}(\Gamma+\mathcal{T}) \)

- Speculative inference: add *arbitrary* clause \( C \)
- To induce termination on sat input
- What if it makes problem unsat?!
- Detect conflict and backjump:
  - Keep track by adding \([C] \implies C\)
  - \([C]\): new propositional variable (a “name” for \( C \))
  - Speculative inferences are *reversible*
Speculative inferences in DPLL(Γ+Γ)

State of derivation: \( M \parallel F \)

Inference rule:

- \textit{SpeculativeIntro}: add \( \lceil C \rceil \triangleright C \) to \( F \) and \( \lceil C \rceil \) to \( M \)

- Rule \textit{SpeculativeIntro} also bounded by iterative deepening
Example as done by system

1. \( \neg(x \sqsubseteq y) \lor f(x) \sqsubseteq f(y) \)
2. \( a \sqsubseteq b \)
3. \( a \sqsubseteq f(c) \)
4. \( \neg(a \sqsubseteq c) \)
Example as done by system

1. \( \neg (x \sqsubseteq y) \lor f(x) \sqsubseteq f(y) \)
2. \( a \sqsubseteq b \)
3. \( a \sqsubseteq f(c) \)
4. \( \neg (a \sqsubseteq c) \)

1. Add \([f(x) \simeq x] \triangleright f(x) \simeq x\)
2. Rewrite \( a \sqsubseteq f(c) \) into \([f(x) \simeq x] \triangleright a \sqsubseteq c\)
Example as done by system

1. \( \neg (x \sqsubseteq y) \lor f(x) \sqsubseteq f(y) \)
2. \( a \sqsubseteq b \)
3. \( a \sqsubseteq f(c) \)
4. \( \neg (a \sqsubseteq c) \)

1. Add \([f(x) \simeq x] \triangleright f(x) \simeq x\)
2. Rewrite \( a \sqsubseteq f(c) \) into \([f(x) \simeq x] \triangleright a \sqsubseteq c\)
3. Generate \([f(x) \simeq x] \triangleright \Box\); Backtrack, learn \( \neg [f(x) \simeq x] \)
Example as done by system

1. \( \neg (x \sqsubseteq y) \lor f(x) \sqsubseteq f(y) \)
2. \( a \sqsubseteq b \)
3. \( a \sqsubseteq f(c) \)
4. \( \neg (a \sqsubseteq c) \)
   
   1. Add \( \lceil f(x) \simeq x \rceil \triangleright f(x) \simeq x \)
   2. Rewrite \( a \sqsubseteq f(c) \) into \( \lceil f(x) \simeq x \rceil \triangleright a \sqsubseteq c \)
   3. Generate \( \lceil f(x) \simeq x \rceil \triangleright \square \); Backtrack, learn \( \neg \lceil f(x) \simeq x \rceil \)
   4. Add \( \lceil f(f(x)) \simeq x \rceil \triangleright f(f(x)) \simeq x \)
   5. \( a \sqsubseteq b \) yields only \( f(a) \sqsubseteq f(b) \)
   6. \( a \sqsubseteq f(c) \) yields only \( f(a) \sqsubseteq f(f(c)) \)
      rewritten to \( \lceil f(f(x)) = x \rceil \triangleright f(a) \sqsubseteq c \)
   7. Terminate and detect satisfiability
To decide satisfiability modulo $\mathcal{T}$ of $\mathcal{R} \cup P$:

- Find sequence of “speculative axioms” $U$
- Show that there exists $k$ s.t. $k$-bounded $\text{DPLL}(\Gamma + \mathcal{T})$ is guaranteed to terminate
  - with $\text{Unsat}$ if $\mathcal{R} \cup P$ is $\mathcal{T}$-unsat
  - in a state which is not stuck at $k$ if $\mathcal{R} \cup P$ is $\mathcal{T}$-sat
Decision procedures

- $\mathcal{R}$ has single monadic function symbol $f$
- *Essentially finite*: if $\mathcal{R} \cup P$ is sat, has model where range of $f$ is finite
- Such a model satisfies $f^j(x) \simeq f^k(x)$ for some $j \neq k$
Decision procedures

- $\mathcal{R}$ has single monadic function symbol $f$
- *Essentially finite*: if $\mathcal{R} \cup P$ is sat, has model where range of $f$ is finite
- Such a model satisfies $f^j(x) \simeq f^k(x)$ for some $j \neq k$
- SpeculativeIntro adds “pseudo-axioms” $f^j(x) \simeq f^k(x)$, $j > k$
- Use $f^j(x) \simeq f^k(x)$ as rewrite rule to limit term depth
Decision procedures

- $\mathcal{R}$ has single monadic function symbol $f$
- *Essentially finite*: if $\mathcal{R} \cup P$ is sat, has model where range of $f$ is finite
- Such a model satisfies $f^j(x) \simeq f^k(x)$ for some $j \neq k$
- *SpeculativeIntro* adds “pseudo-axioms” $f^j(x) \simeq f^k(x)$, $j > k$
- Use $f^j(x) \simeq f^k(x)$ as rewrite rule to limit term depth
- Clause length limited by properties of $\Gamma$ and $\mathcal{R}$
- Only finitely many clauses generated: termination without getting stuck
Situations where clause length is limited

$\Gamma$: Superposition, Resolution + negative selection, Simplification

Negative selection: only positive literals in positive clauses are active

- $R$ is Horn
- $R$ is ground-preserving: variables in positive literals appear also in negative literals; the only positive clauses are ground
Axiomatizations of type systems

Reflexivity \( x \sqsubseteq x \) (1)

Transitivity \( \neg (x \sqsubseteq y) \lor \neg (y \sqsubseteq z) \lor x \sqsubseteq z \) (2)

Anti-Symmetry \( \neg (x \sqsubseteq y) \lor \neg (y \sqsubseteq x) \lor x \simeq y \) (3)

Monotonicity \( \neg (x \sqsubseteq y) \lor f(x) \sqsubseteq f(y) \) (4)

Tree-Property \( \neg (z \sqsubseteq x) \lor \neg (z \sqsubseteq y) \lor x \sqsubseteq y \lor y \sqsubseteq x \) (5)

Multiple inheritance: \( \text{MI} = \{(1), (2), (3), (4)\} \)

Single inheritance: \( \text{SI} = \text{MI} \cup \{(5)\} \)
DPLL(Γ+T) with SpeculativeIntro adding $f^j(x) \simeq f^k(x)$ for $j > k$ decides the satisfiability modulo $T$ of problems

- $\text{MI} \cup P$
- $\text{SI} \cup P$
- $\text{MI} \cup \text{TR} \cup P$ and $\text{SI} \cup \text{TR} \cup P$

where $\text{TR} = \{\neg (g(x) \simeq \text{null}), h(g(x)) \simeq x\}$