Satisfiability Modulo Theories and DPLL(T)

Andrew Reynolds

March 18, 2015

Overview

- SAT : Satisfiability for Propositional Logic

  $(A \vee B) \wedge (C \vee D) \wedge \neg B$

  - Does there exist truth values for A, B, C, D that make this formula true?
- SMT : Satisfiability Modulo Theories

  $(x+1>0 \vee x+y>0) \wedge (x<0 \vee x+y>4) \wedge \neg x+y>0$

  - Does there exist integer values for x, y that make this formula true?
- Theories:
  - Linear Integer Arithmetic (LIA)
  - Inductive Datatypes (DT)
- Additional Topics:
  - Combination of Theories (DT+LIA)
  - Quantified Formulas

SAT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B$$

- DPLL algorithm
  - Input : clauses in Clausal Normal Form (CNF)
  - Alternates between:
    - Propagations : assign values to atoms whose value is forced
    - Decisions : choose an arbitrary value for an unassigned atom
  - Answers SAT when all clauses have one literal $\rightarrow$ true
  - Answer UNSAT when made no decisions, one clause has all of its literals $\rightarrow$ false

SAT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B$$

- DPLL algorithm

SAT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B$$

- DPLL algorithm
  - Propagate : B $\rightarrow$ false

SAT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B$$

- DPLL algorithm
  - Propagate : B $\rightarrow$ false
  - Propagate : A $\rightarrow$ true

SAT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B$$

- DPLL algorithm
  - Propagate : B $\rightarrow$ false
  - Propagate : A $\rightarrow$ true
  - Decide : C $\rightarrow$ true

SAT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B$$

- DPLL algorithm
  - Propagate : B $\rightarrow$ false
  - Propagate : A $\rightarrow$ true
  - Decide : C $\rightarrow$ true

  $\Rightarrow$ Input is satisfiable

SAT

- Optimizations:
  - Two watched literals
  - Conflict-driven clause learning (CDCL)
  - Inprocessing
- Using an encoding of problems into propositional SAT:
  - Pro : Very efficient methods available
  - Con : Not expressive

    $\Rightarrow$ Motivation for Satisfiability *Modulo Theories*

Satisfiability Modulo Theories (SMT)

- Extend SAT problems with reasoning about *theories*
  - E.g. linear integer arithmetic (LIA) : $(x+1>0 \vee x+y>0) \wedge (x<0 \vee x+y>4)$

- Formally, a theory T is a pair (Σ_T, I_T), where:
  - Σ_T is set of function symbols, the *signature* of T
    - E.g. $\Sigma_{LIA} = \{ +, -, <, \leq, >, \geq, 0, 1, 2, 3, \dots \}$
  - I_T is a set of *interpretations* for T
    - E.g. each interpretation in I_{LIA} interprets functions in Σ_{LIA} in standard way:
      - 1+1 = 2, 1+2 = 3, …
      - 1 > 0 = true, 0 > 1 = false, …
      - …
  - A formula F is *T-satisfiable* if there is an interpretation in I_T that satisfies F

SMT

$$(x+1>0 \lor x+y>0) \land (x<0 \lor x+y>4) \land \neg x+y>0$$

- DPLL(**T**) algorithm
  - Extends DPLL algorithm to incorporate reasoning about a theory **T**
  - Idea:
    - Use DPLL algorithm to find assignments for propositional abstraction of formula
      - Use off-the-shelf SAT solver
    - Check the T-satisfiability of assignments found by SAT solver
      - Use *Theory Solver for T*

SMT

$$(x+1>0 \lor x+y>0) \land (x<0 \lor x+y>4) \land \neg x+y>0$$

- DPLL(LIA) algorithm

  Invoke DPLL(T) for theory T = LIA (linear integer arithmetic)

SMT

$$(x+1>0 \vee x+y>0) \wedge (x<0 \vee x+y>4) \wedge \neg x+y>0$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B$$

- DPLL(LIA) algorithm
  - Map : { $A \Leftrightarrow x+1>0$, $B \Leftrightarrow x+y>0$, $C \Leftrightarrow x<0$, $D \Leftrightarrow x+y>4$ }
  - Invoke SAT solver:

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B$$

- DPLL(LIA) algorithm
  - Map : { $A \Leftrightarrow x+1>0$, $B \Leftrightarrow x+y>0$, $C \Leftrightarrow x<0$, $D \Leftrightarrow x+y>4$ }
  - Invoke SAT solver:
    - Propagate : $B \rightarrow$ false, Propagate : $A \rightarrow$ true, Decide : $C \rightarrow$ true

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B$$

- DPLL(LIA) algorithm
  - Map : { $A \Leftrightarrow x+1>0$, $B \Leftrightarrow x+y>0$, $C \Leftrightarrow x<0$, $D \Leftrightarrow x+y>4$ }
  - Invoke SAT solver:
    - Propagate : $B \rightarrow$ false, Propagate : $A \rightarrow$ true, Decide : $C \rightarrow$ true

  $\Rightarrow$ Are LIA literals corresponding to {A, $\neg$B, C } LIA-satisfiable?

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - x+1>0 $\wedge$ x<0 is *LIA-unsatisfiable!*

SMT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B \wedge (\neg A \vee \neg C)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses

      $\Rightarrow$ Since one of x+1>0, x<0 must be false

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B \quad \wedge$$
$$(\quad \neg A \quad \vee \neg C \quad)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Backtrack decision on C

SMT

$$(\quad A \quad \vee \quad B \quad) \wedge (\quad C \quad \vee \quad D \quad) \wedge \neg B \quad \wedge$$
$$(\quad \neg A \quad \vee \neg C \quad)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Propagate : C $\rightarrow$ false

SMT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B \wedge (\neg A \vee \neg C)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Propagate : C $\rightarrow$ false, Propagate : D $\rightarrow$ true

SMT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B \wedge (\neg A \vee \neg C)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Propagate : C $\rightarrow$ false, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0, x+y>4 }

SMT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B \wedge (\neg A \vee \neg C)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Propagate : C $\rightarrow$ false, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0, x+y>4 }
    - $\neg$x+y>0 $\wedge$ x+y>4 is *LIA-unsatisfiable!*

SMT

$$(A \vee B) \wedge (C \vee D) \wedge \neg B \wedge$$
$$(\neg A \vee \neg C) \wedge (B \vee \neg D)$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Propagate : C $\rightarrow$ false, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0, x+y>4 }
    - (B $\vee$ $\neg$D) added to list of clauses

SMT

$$(\ A\ \vee\ B\ )\wedge(\ C\ \vee\ D\ )\wedge\neg B\ \wedge$$
$$(\ \neg A\ \vee\neg C\ )\wedge(\ B\ \vee\ \neg D\ )$$

- DPLL(LIA) algorithm
  - Map : { A $\Leftrightarrow$ x+1>0, B $\Leftrightarrow$ x+y>0, C $\Leftrightarrow$ x<0, D $\Leftrightarrow$ x+y>4 }
  - Invoke SAT solver:
    - Propagate : B $\rightarrow$ false, Propagate : A $\rightarrow$ true, Decide : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0 }
    - ($\neg$A $\vee$ $\neg$C) added to list of clauses
  - Invoke SAT solver:
    - Propagate : C $\rightarrow$ false, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { x+1>0, $\neg$x+y>0, x<0, x+y>4 }
    - (B $\vee$ $\neg$D) added to list of clauses
  - No decisions to backtrack $\Rightarrow$ *input is LIA-unsatisfiable*

SMT : Exercise

$$(x>y \vee x>z) \wedge (x+1<y \vee \neg x>y) \wedge (x>y \vee z>y)$$

- Determine if above formula is LIA-satisfiable

SMT : Exercise

$$\Phi \{ (\ A \vee B\ ) \wedge (\ C \vee \neg A\ ) \wedge (\ A \vee D\ )$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }

SMT : Exercise

$$\Phi \{ \; (\; A \; \vee \; B \;) \wedge (\quad C \quad \vee \quad \neg A \;) \wedge (\quad A \; \vee \quad D \;)$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - If SAT, give values for { x, y, z }
  - If UNSAT, give a set of clauses C_1, ..., C_n, where:
    - Φ, C_1, ..., C_n, is UNSAT
    - Each C_i is of the form ($l_1 \vee \ldots \vee l_m$), where:
      - Each l_i is one of ($\neg$)A, ($\neg$)B, ($\neg$)C, ($\neg$)D
      - Negation of formulas mapped to by $l_1 \ldots l_m$ are LIA-unsatisfiable

SMT : Exercise

$$\Phi \{ (\ A \vee B\ ) \wedge (\ C \vee \neg A\ ) \wedge (\ A \vee D\ )$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }

SMT : Exercise

Φ { (A $\vee$ B) $\wedge$ (C $\vee$ $\neg$A) $\wedge$ (A $\vee$ D)

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : C $\rightarrow$ true

SMT : Exercise

Φ { (A $\vee$ B) $\wedge$ (C $\vee$ $\neg$A) $\wedge$ (A $\vee$ D)

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }

SMT : Exercise

$$\Phi \{ (\ A \vee B\ ) \wedge (\ C \vee \neg A\ ) \wedge (\ A \vee D\ ) \wedge (\ \neg A \vee \neg C\ )$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }
    - x>y $\wedge$ x+1<y is LIA-unsatisfiable, add ($\neg$A $\vee$ $\neg$C)

SMT : Exercise

$$\Phi \{ (A \vee B) \wedge (C \vee \neg A) \wedge (A \vee D) \wedge (\neg A \vee \neg C)$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }
    - x>y $\wedge$ x+1<y is LIA-unsatisfiable, add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver:
    - Backtrack decision on A

SMT : Exercise

$$\Phi \{ (\ A \vee B\ ) \wedge (\ C \vee \neg A\ ) \wedge (\ A \vee D\ )$$
$$(\ \neg A \vee \neg C\ )$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : D $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }
    - x>y $\wedge$ x+1<y is LIA-unsatisfiable, add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver:
    - Propagate : A $\rightarrow$ false, Propagate : B $\rightarrow$ true, Propagate : D $\rightarrow$ true

SMT : Exercise

$$\Phi \{ (A \vee B) \wedge (C \vee \neg A) \wedge (A \vee D) \wedge (\neg A \vee \neg C)$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : D $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }
    - x>y $\wedge$ x+1<y is LIA-unsatisfiable, add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver:
    - Propagate : A $\rightarrow$ false, Propagate : B $\rightarrow$ true, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { $\neg$x>y, x>z, z>y }

SMT : Exercise

$$\Phi \{ (A \vee B) \wedge (C \vee \neg A) \wedge (A \vee D)$$
$$(\neg A \vee \neg C) \wedge (A \vee \neg B \vee \neg D)$$

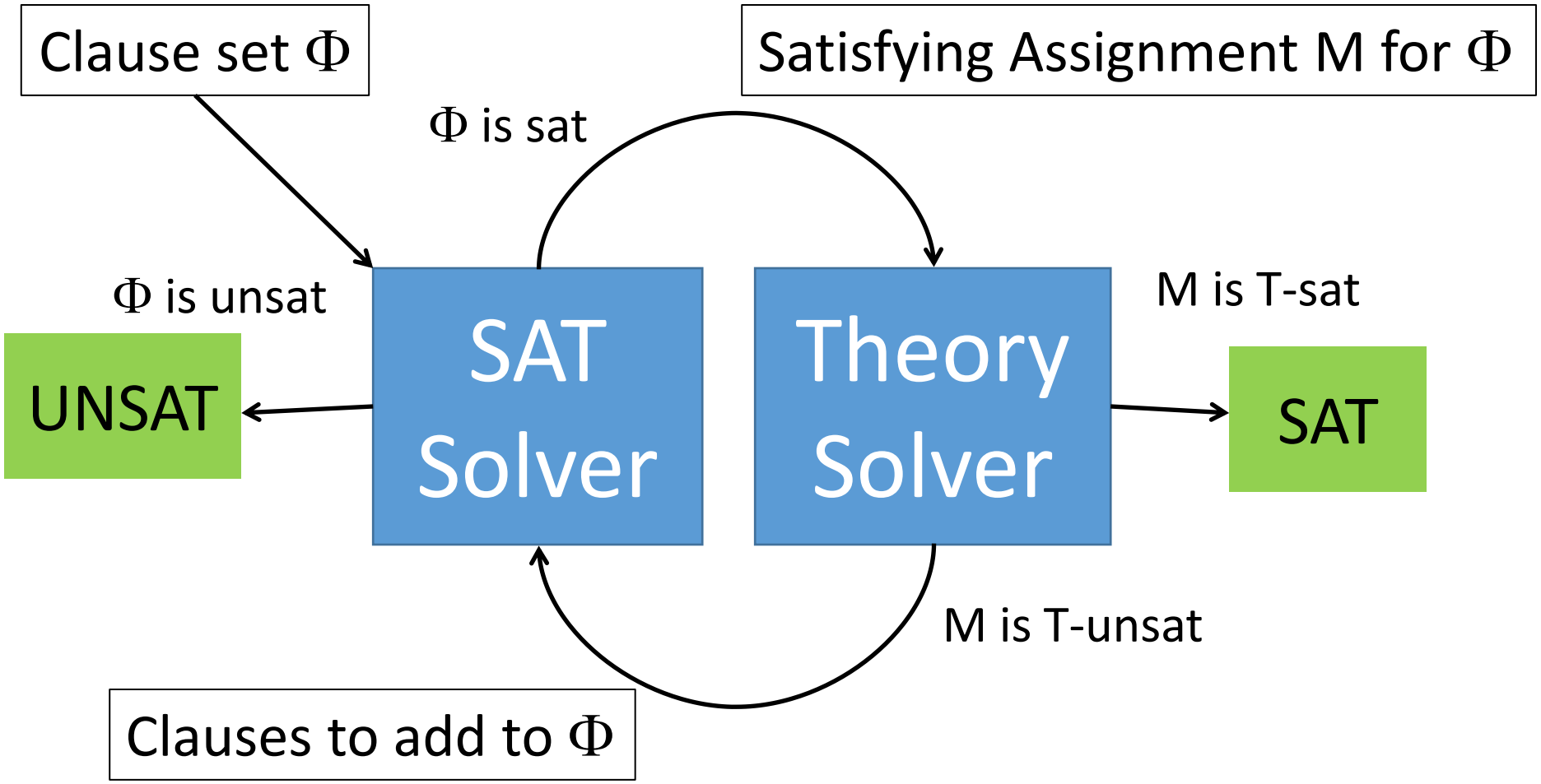
- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : D $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }
    - x>y $\wedge$ x+1<y is LIA-unsatisfiable, add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver:
    - Propagate : A $\rightarrow$ false, Propagate : B $\rightarrow$ true, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { $\neg$x>y, x>z, z>y }
    - $\neg$x>y $\wedge$ x>z $\wedge$ z>y is LIA-unsatisfiable, add (A $\vee$ $\neg$B $\vee$ $\neg$D)

SMT : Exercise

$$\Phi \{ (A \vee B) \wedge (C \vee \neg A) \wedge (A \vee D)$$
$$(\neg A \vee \neg C) \wedge (A \vee \neg B \vee \neg D)$$

- Determine if Φ is LIA-satisfiable
  - Map : { A $\Leftrightarrow$ x>y, B $\Leftrightarrow$ x>z, C $\Leftrightarrow$ x+1<y, D $\Leftrightarrow$ z>y }
  - Invoke SAT solver:
    - Decide : A $\rightarrow$ true, Propagate : D $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for LIA on: { x>y, x+1<y }
    - x>y $\wedge$ x+1<y is LIA-unsatisfiable, add **($\neg$A $\vee$ $\neg$C)**
  - Invoke SAT solver:
    - Propagate : A $\rightarrow$ false, Propagate : B $\rightarrow$ true, Propagate : D $\rightarrow$ true
  - Invoke theory solver for LIA on: { $\neg$x>y, x>z, z>y }
    - $\neg$x>y $\wedge$ x>z $\wedge$ z>y is LIA-unsatisfiable, add **(A $\vee$ $\neg$B $\vee$ $\neg$D)**
  - No decisions to backtrack $\Rightarrow$ *input is* ***LIA-unsatisfiable***

DPLL(T)


Clause set Φ
Satisfying Assignment M for Φ
Φ is sat
Φ is unsat
M is T-sat
UNSAT
SAT Solver
Theory Solver
SAT
M is T-unsat
Clauses to add to Φ

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
  2. $\{ l_1, \ldots, l_n \} \subseteq$ M is T-unsatisfiable
  3. Don’t know: return lemma

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
     - Return *model*, e.g. { x $\rightarrow$ 2, y $\rightarrow$ 3, z $\rightarrow$ -3, … }
  2. { l_1, …, l_n } $\subseteq$ M is T-unsatisfiable
  3. Don’t know: return lemma

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
     - Return *model*, e.g. { x $\rightarrow$ 2, y $\rightarrow$ 3, z $\rightarrow$ -3, … }
  2. { l_1, …, l_n } $\subseteq$ M is T-unsatisfiable
     - Add T-*conflict* clause ($\neg l_1 \vee \ldots \vee \neg l_n$) to Φ
  3. Don’t know: return lemma

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
     - Return *model*, e.g. { x $\rightarrow$ 2, y $\rightarrow$ 3, z $\rightarrow$ -3, … }
  2. { l_1, …, l_n } $\subseteq$ M is T-unsatisfiable
     - Add T-*conflict* clause ($\neg l_1 \vee \ldots \vee \neg l_n$) to Φ
  3. Don't know: return lemma
     - Add clause to Φ, e.g. splitting on demand (x = y $\vee \neg$x=y)

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
     - Return *model*, e.g. { x $\rightarrow$ 2, y $\rightarrow$ 3, z $\rightarrow$ -3, … }

     $\Rightarrow$ Should be *solution-sound*
       - Answers “M is T-satisfiable” only if M is T-satisfiable
  2. { l_1, …, l_n } $\subseteq$ M is T-unsatisfiable
     - Add T-*conflict* clause ($\neg l_1 \vee \ldots \vee \neg l_n$) to Φ
  3. Don’t know: return lemma
     - Add clause to Φ, e.g. splitting on demand (x = y $\vee \neg$x=y)

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
     - Return *model*, e.g. { x $\rightarrow$ 2, y $\rightarrow$ 3, z $\rightarrow$ -3, … }

     $\Rightarrow$ Should be *solution-sound*
       - Answers “M is T-satisfiable” only if M is T-satisfiable
  2. { l_1, …, l_n } $\subseteq$ M is T-unsatisfiable
     - Add T-*conflict* clause ($\neg l_1 \vee \ldots \vee \neg l_n$) to Φ

     $\Rightarrow$ Should be *refutation-sound*
       - Anwers “{ l_1, …, l_n } is T-unsatisfiable” only if { l_1, …, l_n } is T-unsatisfiable
  3. Don’t know: return lemma
     - Add clause to Φ, e.g. splitting on demand (x = y $\vee \neg$x=y)

DPLL(T) Theory Solvers

- Input : A set of T-literals M
- Output : either
  1. M is T-satisfiable
     - Return *model*, e.g. { x $\rightarrow$ 2, y $\rightarrow$ 3, z $\rightarrow$ -3, … }

     $\Rightarrow$ Should be *solution-sound*
     - Answers "M is T-satisfiable" only if M is T-satisfiable
  2. { l_1, …, l_n } $\subseteq$ M is T-unsatisfiable
     - Add T-*conflict* clause ($\neg l_1 \vee \ldots \vee \neg l_n$) to Φ

     $\Rightarrow$ Should be *refutation-sound*
     - Anwers "{ l_1, …, l_n } is T-unsatisfiable" only if { l_1, …, l_n } is T-unsatisfiable
  3. Don't know: return lemma
     - Add clause to Φ, e.g. splitting on demand (x = y $\vee \neg$x=y)

  $\Rightarrow$If solver is solution-sound, refutation-sound, and *terminating*,
     - Then it is a *decision procedure* for T

Design of DPLL(T) Theory Solvers

- A DPLL(T) theory solver:
  - Should be solution-sound, refutation-sound, terminating
  - Should produce models when M is T-satisfiable
  - Should produce T-conflicts of minimal size when M is T-unsatisfiable
  - Should be designed to work *incrementally*
    - M is constantly being appended to/backtracked upon
  - Can be designed to check T-satisfiability either:
    - Eagerly: Check if M is T-satisfiable immediately when any literal is added to M
    - Lazily: Check if M is T-satisfiable only when M is complete
  - Should cooperate with other theory solvers when combining theories
    - (see later)

DPLL(T) Theory Solvers : Examples

- SMT solvers incorporate:
  - Theory solvers that are *decision procedures* for e.g.:
    - Theory of Equality and Uninterpreted Functions (EUF)
      - Congruence closure algorithm
    - Theory of Linear Integer/Real Arithmetic
      - Simplex algorithm
    - Theory of Arrays
    - Theory of Bit Vectors
    - Theory of Inductive Datatypes
    - …
  - Theory solvers that are *incomplete procedures* for e.g.:
    - Theory of Non-Linear Integer Arithmetic
    - Theory of Strings + Length constraints
    - Quantified formulas

DPLL(T) Theory Solvers : Examples

- SMT solvers incorporate:
  - Theory solvers that are *decision procedures* for e.g.:
    - Theory of Equality and Uninterpreted Functions (EUF)
      - Congruence closure algorithm
    - Theory of Linear Integer/Real Arithmetic
      - Simplex algorithm
    - Theory of Arrays
    - Theory of Bit Vectors
    - *Theory of Inductive Datatypes*
    - …

    } Focus of the next part
  - Theory solvers that are *incomplete procedures* for e.g.:
    - Theory of Non-Linear Integer Arithmetic
    - Theory of Strings + Length constraints
    - Quantified formulas

Theory of Inductive Datatypes

- Family of theories specified by a set of *types* with *constructors*:

$$D_1 := C_1(s_1 : T_1, \ldots, s_i : T_i) \mid \ldots \mid C_j(\ldots)$$
$$\ldots$$
$$D_m := C_{m1}(\ldots) \mid \ldots \mid C_{mk}(\ldots)$$

  - $D_1, \ldots, D_m$ are *datatypes*
  - $C_1, \ldots, C_j$ are the *constructors* of datatype type D_1
  - C_1 has subfields of type $T_1 \ldots T_i$
    - $s_1, \ldots, s_i$ are the *selectors* for constructor C_1

Theory of Inductive Datatypes : Example

```
ClrList := cons( head : Clr, tail : ClrList ) | nil
Clr := red | green | blue
```

Theory of Inductive Datatypes : Example

ClrList := cons(head : Clr, tail : ClrList) | nil
Clr := red | green | blue

- Theory of Inductive Datatypes (DT) for ClrList and Clr
  - Σ_{DT} : { cons, head, tail, nil, red, green, blue }
  - Interpretations I_{DT} are such that:
    - Terms with different constructors are distinct
      - red ≠ green
    - Constructors are injective
      - If cons(c_1, l_1) = cons(c_2, l_2), then $c_1 = c_2$ and $l_1 = l_2$
    - Terms of a datatype must have one of its constructors as its topmost symbol
      - Each c is such that c = red or c = green or c = blue
    - Selectors access subfields
      - head(cons(c, l)) = c
    - Terms do not contain themselves as subterms
      - l ≠ cons(c, l)

Theory of Inductive Datatypes : Example

$$cons(x,nil)=cons(y,z) \wedge (x=red \vee x = blue) \wedge y = green$$

- DPLL(DT) algorithm

Theory of Inductive Datatypes : Example

$$A \quad \wedge (\quad B \quad \vee \quad C \quad) \wedge \; D$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }

Theory of Inductive Datatypes : Example

$$A \quad \wedge (\quad B \quad \vee \quad C \quad) \wedge \quad D$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }

Theory of Inductive Datatypes : Example

$$A \quad \wedge (\quad B \quad \vee \quad C \quad) \wedge \quad D$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - cons(x, nil) = cons(y, z) $\wedge$ x = red $\wedge$ y = green is *DT-unsatisfiable!*

      $\Rightarrow$ Since cons(x, nil) = cons(y, nil), we have x = y, but x = red and y = green and red $\neq$ green

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses
  - Invoke SAT solver
    - Backtrack decision on B

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses
  - Invoke SAT solver
    - Propagate : B $\rightarrow$ false, Propagate : C $\rightarrow$ true

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses
  - Invoke SAT solver
    - Propagate : B $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), $\neg$x = red, x=blue, y = green }

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses
  - Invoke SAT solver
    - Propagate : B $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), $\neg$x = red, x=blue, y = green }
    - cons(x, nil) = cons(y, z) $\wedge$ x = blue $\wedge$ y = green is *DT-unsatisfiable!*

      $\Rightarrow$ Since cons(x, nil) = cons(y, nil), we have x = y, but x = red and y = green and red $\neq$ green

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D) \wedge (\neg A \vee \neg C \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses
  - Invoke SAT solver
    - Propagate : B $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), $\neg$x = red, x=blue, y = green }
    - ($\neg$A $\vee$ $\neg$C $\vee$ $\neg$D) added to list of clauses

Theory of Inductive Datatypes : Example

$$A \wedge (B \vee C) \wedge D$$

$$(\neg A \vee \neg B \vee \neg D) \wedge (\neg A \vee \neg C \vee \neg D)$$

- DPLL(DT) algorithm
  - Map : { A $\Leftrightarrow$ cons(x, nil)=cons(y, z), B $\Leftrightarrow$ x=red, C $\Leftrightarrow$ x=blue, D $\Leftrightarrow$ y=green }
  - Invoke SAT solver
    - Propagate : A $\rightarrow$ true, Propagate : D $\rightarrow$ true, Decide : B $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), x = red, y = green }
    - ($\neg$A $\vee$ $\neg$B $\vee$ $\neg$D) added to list of clauses
  - Invoke SAT solver
    - Propagate : B $\rightarrow$ false, Propagate : C $\rightarrow$ true
  - Invoke theory solver for DT on: { cons(x, nil) = cons(y, z), $\neg$x = red, x=blue, y = green }
    - ($\neg$A $\vee$ $\neg$C $\vee$ $\neg$D) added to list of clauses
  - No decisions to backtrack $\Rightarrow$ *input is DT-unsatisfiable*

Combination of Theories

- What if we have:

IntList := cons(head : Int, tail : IntList) | nil

- Example input:

(head(x)+3 = y $\vee$ x = cons(y+1, nil)) $\wedge$ head(x) > y+1

$\Rightarrow$Requires reasoning about datatypes *and* integers

Combination of Theories

- What if we have:

IntList := cons(head : Int, tail : IntList) | nil

- Example input:

$$(\text{head}(x)+3 = y \vee x = \text{cons}(y+1, \text{nil})) \wedge \text{head}(x) > y+1$$

- Idea:
  - *Purify* the literals in the input
  - Use DPLL(LIA+DT): find satisfying assignments $M = M_{LIA} \cup M_{DT}$
    - Use existing solver for LIA to check if M_{LIA} is LIA-satisfiable
    - Use existing solver for DT to check if M_{DT} is DT-satisfiable
  - If either of $\{ M_{LIA}, M_{DT} \}$ is T-unsatisfiable, then M is T-unsatisfiable
  - If both $\{ M_{LIA}, M_{DT} \}$ are T-satisfiable, then solvers must combine models
    - Communicate equalities between *shared terms*

Combination of Theories : Example

$$(\text{head}(x)+3 = y \vee x = \text{cons}(y+1, \text{nil})) \wedge \text{head}(x) > y+1$$

- DPLL(LIA+DT) algorithm

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ head(x)+3 =y, B $\Leftrightarrow$ x=cons(y+1, nil), C $\Leftrightarrow$ head(x) > y+1 }

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ head(x)+3 =y, B $\Leftrightarrow$ x=cons(y+1, nil), C $\Leftrightarrow$ head(x) > y+1 }
  - Purify A, B, C, e.g. introduce fresh “shared” variables such that:
    - Each literal contains function symbols only belonging to one theory (DT or LIA)

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }

Combination of Theories : Example

(A ∨ B) ∧ C

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true

Combination of Theories : Example

(A ∨ B) ∧ C

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers on : { u1 + 3 = y, u1 > y+1 } ∪ { u_1 = head(x), u_2 = y+1 }

Combination of Theories : Example

(A ∨ B) ∧ C

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) }
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }

Combination of Theories : Example

(A ∨ B) ∧ C

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… $u_1 + 3 = y \wedge u_1 > y+1$ *is LIA-unsatisfiable!* Add ($\neg A \vee \neg C$)

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y $\wedge$ u_1 > y+1 *is LIA-unsatisfiable!* Add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver : Backtrack decision on A

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y $\wedge$ u_1 > y+1 *is LIA-unsatisfiable!* Add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y $\wedge$ u_1 > y+1 *is LIA-unsatisfiable!* Add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers on : { $\neg u_1$+3=y, x=cons(u_2, nil), u_1 > y+1 } $\cup$ { u_1 = head(x), u_2 = y+1 }

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y $\wedge$ u_1 > y+1 *is LIA-unsatisfiable!* Add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) }
    - Solver for LIA on : { $\neg u_1$+3=y, u_1 > y+1, u_2 = y+1 }

Combination of Theories : Example

(A ∨ B) ∧ C

(¬A ∨ ¬C)

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y ∧ u_1 > y+1 *is LIA-unsatisfiable!* Add (¬A ∨ ¬C)
  - Invoke SAT solver : Propagate A → false, Propagate : B → true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { ¬u_1+3=y, u_1 > y+1, u_2 = y+1 }

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y $\wedge$ u_1 > y+1 *is LIA-unsatisfiable!* Add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { $\neg u_1$+3=y, u_1 > y+1, u_2 = y+1 } … LIA-satisfiable

Combination of Theories : Example

(A ∨ B) ∧ C

(¬A ∨ ¬C)

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y ∧ u_1 > y+1 *is LIA-unsatisfiable!* Add (¬A ∨ ¬C)
  - Invoke SAT solver : Propagate A → false, Propagate : B → true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { ¬u_1+3=y, u_1 > y+1, u_2 = y+1 } … LIA-satisfiable

  ⇒ *To answer "satisfiable", theory solvers must agree on equalities between shared variables* u_1, u_2

Combination of Theories : Example

(A ∨ B) ∧ C

(¬A ∨ ¬C)

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y ∧ u_1 > y+1 *is LIA-unsatisfiable!* Add (¬A ∨ ¬C)
  - Invoke SAT solver : Propagate A → false, Propagate : B → true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) } … DT-satisfiable. Is u_1 = u_2?
    - Solver for LIA on : { ¬u_1+3=y, u_1 > y+1, u_2 = y+1 } … LIA-satisfiable. Is u_1 = u_2?

Combination of Theories : Example

(A ∨ B) ∧ C

(¬A ∨ ¬C)

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y ∧ u_1 > y+1 *is LIA-unsatisfiable!* Add (¬A ∨ ¬C)
  - Invoke SAT solver : Propagate A → false, Propagate : B → true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) } … DT-satisfiable. Is u_1 = u_2? YES: u_1 = head(x) = u_2
    - Solver for LIA on : { ¬u_1+3=y, u_1 > y+1, u_2 = y+1 } … LIA-satisfiable. Is u_1 = u_2? NO: u_2 = y+1 < u_1

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y $\wedge$ u_1 > y+1 *is LIA-unsatisfiable!* Add ($\neg$A $\vee$ $\neg$C)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) }
    - Solver for LIA on : { $\neg u_1$+3=y, u_1 > y+1, u_2 = y+1, u_1 = u_2 }

DT-solver tells LIA-solver u_1=u_2

…since x = cons(u_2, nil) $\wedge$ u_1 = head(x)

Combination of Theories : Example

(A ∨ B) ∧ C

(¬A ∨ ¬C)

- DPLL(LIA+DT) algorithm
  - Map : { A ⇔ u_1 +3 =y, B ⇔ x=cons(u_2 , nil), C ⇔ u_1 > y+1 }
  - Map shared variables : { u_1 ⇔ head(x), u_2 ⇔ y+1 }
  - Invoke SAT solver: Propagate : C → true, Decide : A → true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… u_1 + 3 = y ∧ u_1 > y+1 *is LIA-unsatisfiable!* Add (¬A ∨ ¬C)
  - Invoke SAT solver : Propagate A → false, Propagate : B → true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) }
    - Solver for LIA on : { ¬u_1+3=y, u_1 > y+1, u_2 = y+1, u_1 = u_2 }
      - u_1 > y+1 ∧ u_2 = y+1 ∧ u_1 = u_2 is *LIA-unsatisfiable!*

u_1=u_2

…since x = cons(u_2, nil) ∧ u_1 = head(x)

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… $u_1 + 3 = y \wedge u_1 > y+1$ *is LIA-unsatisfiable!* Add ($\neg A \vee \neg C$)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) }
    - Solver for LIA on : { $\neg u_1$+3=y, u_1 > y+1, u_2 = y+1, u_1 = u_2 }
      - Construct conflict based on *explanation* of u_1 > y+1, u_2 = y+1, u_1 = u_2

$$u_1 = u_2$$

…since x = cons(u_2, nil) $\wedge$ u_1 = head(x)

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… $u_1 + 3 = y \wedge u_1 > y+1$ *is LIA-unsatisfiable!* Add ($\neg A \vee \neg C$)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) }
    - Solver for LIA on : { $\neg u_1$+3=y, u_1 > y+1, u_2 = y+1, u_1 = u_2 }
      - Add ($\neg B \vee \neg C$)

$u_1 = u_2$

…since x = cons(u_2, nil) $\wedge$ u_1 = head(x)

Combination of Theories : Example

$$(\quad A \quad \vee \quad B \quad) \wedge \quad C$$

$$(\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$$

- DPLL(LIA+DT) algorithm
  - Map : { A $\Leftrightarrow$ u_1 +3 =y, B $\Leftrightarrow$ x=cons(u_2 , nil), C $\Leftrightarrow$ u_1 > y+1 }
  - Map shared variables : { $u_1 \Leftrightarrow$ head(x), $u_2 \Leftrightarrow$ y+1 }
  - Invoke SAT solver: Propagate : C $\rightarrow$ true, Decide : A $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { u_1 = head(x) } … DT-satisfiable
    - Solver for LIA on : { u_1 + 3 = y, u_1 > y+1, u_2 = y+1 }… $u_1 + 3 = y \wedge u_1 > y+1$ *is LIA-unsatisfiable!* Add ($\neg A \vee \neg C$)
  - Invoke SAT solver : Propagate A $\rightarrow$ false, Propagate : B $\rightarrow$ true
  - Invoke theory solvers
    - Solver for DT on : { x = cons(u_2, nil), u_1 = head(x) }
    - Solver for LIA on : { $\neg u_1$+3=y, u_1 > y+1, u_2 = y+1, u_1 = u_2 }
      - Add ($\neg B \vee \neg C$)

$\Rightarrow$ Input is *DT+LIA-unsatisfiable!*

$u_1 = u_2$

…since x = cons(u_2, nil) $\wedge$ u_1 = head(x)

SMT : Theory Combination

- Nelson-Oppen Theory Combination
  - SMT solvers use preexisting theory solvers for combined theories $T_1 + \ldots + T_n$
  - Basic idea given purified set of literals M:
    - Partition and distribute M to T1-solver, …, Tn-solver
      - If any T_i-solver says “unsat”, then M is unsatisfiable
      - If each T_i-solver says “sat”, then solvers must agree on equalities between shared variables
  - Requires theory solvers to:
    - Have disjoint signatures
      - E.g. arithmetic has functions { +, <, 0, 1, … }, datatypes has functions { cons, head, tail, … }
    - Know equalities/disequalities between shared variables
      - E.g. are $u_1 = u_2$ equal?
    - Theories agree on cardinalities for shared types
      - E.g. LIA and DT may agree that Int has infinite cardinality

Quantified Formulas

- What if input has *quantifiers*:

$$(\forall x.\ P(x) \vee \forall x.\ \neg P(x)) \wedge P(a) \wedge \exists\ x.\ \neg P(x)$$

for all Int x

there exists an Int x

Quantified Formulas

- What if input has *quantifiers*:

$$(\forall x.\ P(x) \vee \forall x.\ \neg P(x)) \wedge P(a) \wedge \exists\ x.\ \neg P(x)$$

- Problem is generally *undecidable*
  - E.g. no procedure for checking T-satisfiability of {$\forall$x. P(x), P(a), … }

Quantified Formulas

- What if input has *quantifiers*:

$$(\forall x.\ P(x) \vee \forall x.\ \neg P(x)) \wedge P(a) \wedge \quad \neg P(k)$$

- Problem is generally *undecidable*
  - E.g. no procedure for checking T-satisfiability of {$\forall x.\ P(x)$, $P(a)$, … }
- *Witness* existential quantification
  - Introduce a fresh constant that witnesses the formula, so $\exists\ x.\ \neg P(x)$ becomes $\neg P(k)$

Quantified Formulas

- What if input has *quantifiers*:

$$(\forall x.\ P(x) \vee \forall x.\ \neg P(x)) \wedge P(a) \wedge \quad \neg P(k)$$

$$(\neg \forall x.\ P(x) \vee P(a)) \wedge (\neg \forall x.\ P(x) \vee P(k)), \ldots.$$

- Problem is generally *undecidable*
  - E.g. no procedure for checking T-satisfiability of $\{\forall x.\ P(x), P(a), \ldots\}$
- Witness existential quantification
  - Introduce a fresh constant that witnesses the formula, so $\exists\ x.\ \neg P(x)$ becomes $\neg P(k)$
- *Instantiate* universal quantification
  - Add clauses of the form $(\neg \forall x.\ P(x) \vee P(a))$
    - Either P does not hold for all x, or it holds for a

Quantified Formulas

- What if input has *quantifiers*:

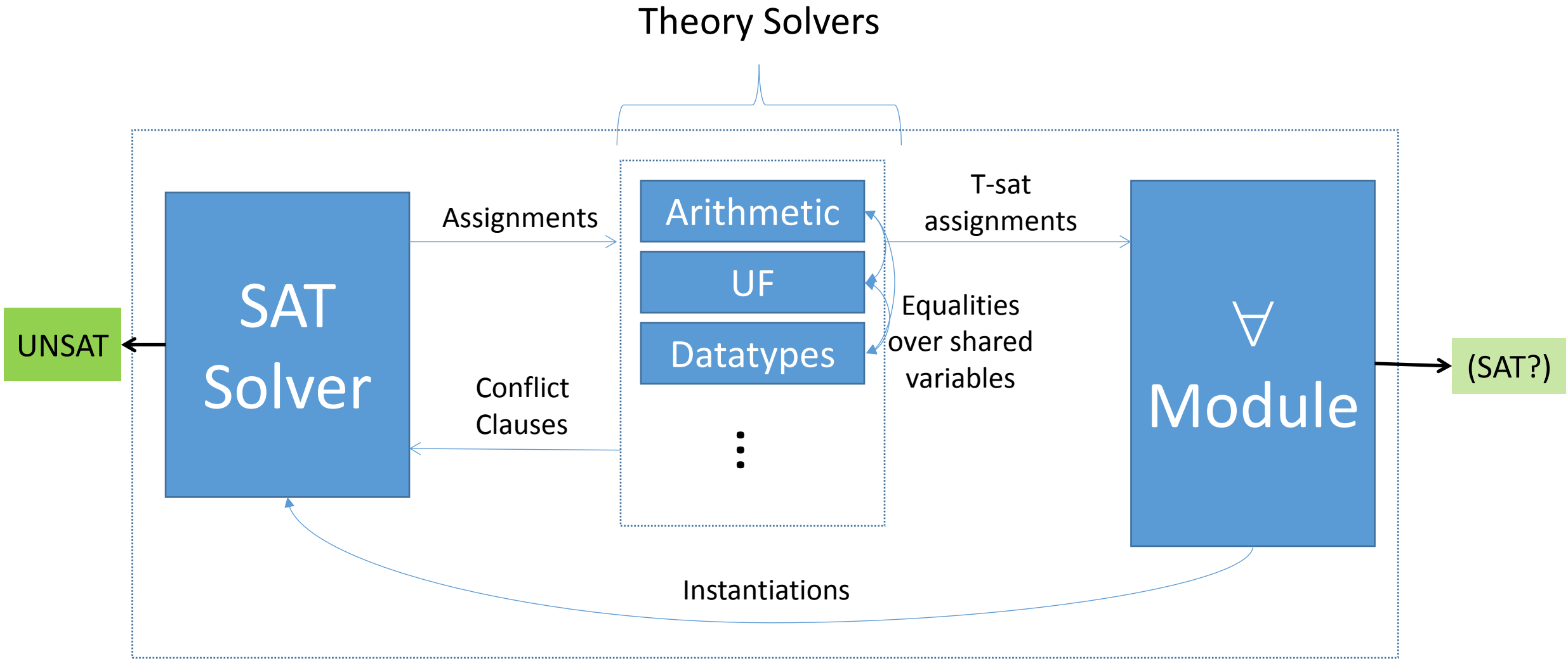
$$(\forall x.\ P(x) \vee \forall x.\ \neg P(x)) \wedge P(a) \wedge \quad \neg P(k)$$

$$(\neg \forall x.\ P(x) \vee P(a)) \wedge (\neg \forall x.\ P(x) \vee P(k)),\ \ldots.$$

- Problem is generally *undecidable*
  - E.g. no procedure for checking T-satisfiability of $\{\forall x.\ P(x), P(a), \ldots\}$
- Witness existential quantification
  - Introduce a fresh constant that witnesses the formula, so $\exists\ x.\ \neg P(x)$ becomes $\neg P(k)$
- *Instantiate* universal quantification
  - Add clauses of the form $(\neg \forall x.\ P(x) \vee P(a))$

$\Rightarrow$ *Sound but incomplete*, thus may be unable to answer “sat”

SMT Solver for Quantifiers : Summary



SMT Summary

- SMT solvers use
  - DPLL(T) algorithm for theory T, which uses:
    - Off-the-shelf SAT solver
    - Theory solver for T
  - Nelson-Oppen theory combination for combined theories $T_1 + T_2$, which uses:
    - Existing theory solvers for T_1 and T_2
  - Incomplete methods for quantified formulas
    - Primarily instantiation-based procedures for (universal) quantification