

— Dagstuhl 2007 —
Seminar on Deduction and Decision Procedures

**Architecting Solvers for SAT Modulo Theories:
Nelson-Oppen with DPLL**

Sava Krstić, Amit Goel, Jim Grundy



Strategic CAD Labs

Overview

Part 1 Our FroCoS'07 paper on SMT solvers' architecture

Part 2 Implementation

Part 1

Motivation

- * our need at Intel for a new SMT solver, and work on it
- * the gap between the theory and practice in this area

Contribution

Definition of the **top-level architecture of an SMT solver as a transition system**

- * precisely formulates the main algorithmic features
- * can be conveniently modified, extended, refined
- * makes the whole solver a mathematical object that can be successfully reasoned about
- * leads to high-quality implementations, but ignores inessential implementation details

Related Work

Influences:

- 1 Detlefs, Nelson, Saxe *Simplify: a theorem prover for program checking* JACM (2005)
- 2 Nieuwenhuis, Oliveras, Tinelli *Solving SAT and SAT Modulo Theories: From an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T)* JACM (2006)
- 3 Conchon, Krstić *Strategies for combining decision procedures* TCS (2006)
- 4 Papers about *CVC**, *PVS*, *UCLID*, *MathSAT*, *Yices*

Formal models for “SAT plus more than one theory”:

- 1 Barrett *Checking Validity of Quantifier-Free Formulas in Combinations of First-Order Theories* PhD thesis (2002)
- 2 Barrett, Nieuwenhuis, Oliveras, Tinelli *Splitting on Demand in SAT Modulo Theories* LPAR 2006
- 3 Bozzano, Brutomesso, Cimatti, Juntilla, van Rossum, Ranise, Sebastiani *Efficient Theory Combination via Boolean Search* Information and Computation (2007)

Background: Components of an SMT Solver

- 1 Propositional SAT Solver
- 2 Nelson-Oppen Combination Algorithm
- 3 Congruence Closure
- 4 Linear Arithmetic
- 5 Quantifier Reasoning
- 6 Reduction Procedures

Background: Components of an SMT Solver

1 Propositional SAT Solver

Searches for satisfying assignments for the propositional skeleton of the input formula:

$$\neg(x \leq y \wedge u \leq v \Rightarrow x \leq v \vee u \leq y)$$
$$\neg(p \wedge q \Rightarrow r \vee s)$$

Reduces the job of theory solvers to checking satisfiability of sets of literals:

$$x \leq y \quad u \leq v \quad \neg(x \leq v) \quad \neg(u \leq y)$$

Background: Components of an SMT Solver

2 Nelson-Oppen Combination Algorithm

- 1 Transform (“purify”) the **mixed** input formula Φ into an equisatisfiable set of **pure** formulas Φ_1, \dots, Φ_n
- 2 If $\Phi_i \models x = y$, update all Φ_j with $\Phi_j \cup \{x = y\}$
- 3 Repeat 2 until $\Phi_i \vdash \perp$ for some i (return **unsat**), or 2 no longer applies (return **sat**)

Background: Components of an SMT Solver

3 Congruence Closure

Satisfiability of sets

$$\{u_1 = v_1, u_2 = v_2, \dots\} \cup \{w_1 \neq z_1, w_2 \neq z_2, \dots\}$$

where terms u_i, v_j, w_k, z_l are built using variables and free (unconstrained) function symbols.

EUF = “equality with uninterpreted functions”

- 1 Build a graph whose nodes are all occurring subterms and iteratively identify the nodes that must be equated because of the **congruence property**

$$x \approx y \wedge u \approx v \Rightarrow g(x, y) \approx g(u, v)$$

- 2 Check if $w \approx z$ for some asserted disequality $w \neq z$

Background: Components of an SMT Solver

4 Linear Arithmetic

Satisfiability of sets of linear equalities, disequalities, inequalities

- * Real coefficients: Fourier-Motzkin elimination, Simplex
- * Integer coefficients: Omega Test, Branch-and-Cut, . . .

- * **Difference Logic** fragment (literals like $x - y \leq 12$) is more efficiently tractable

Background: Components of an SMT Solver

5 Quantifier Reasoning

No decision procedure here, but there are good **heuristics for instantiating quantified assumptions**

$$\left[\begin{array}{l} \forall xyz. (x \cdot y) \cdot z = x \cdot (y \cdot z) \\ a \cdot (b \cdot (c \cdot (d \cdot e))) \neq (((a \cdot b) \cdot c) \cdot d) \cdot e \end{array} \right]$$

- * Some theories (e.g. arrays) are defined by simple axioms

Background: Components of an SMT Solver

6 Reduction Approach

[Kapur,Zarba (2005)]

Transform a formula of a theory T into an equisatisfiable formula of a simpler theory T'

- * Theory of arrays with signature $\{read, write\}$ can be reduced to EUF
- * *UCLID* is the ultimate reducer: everything to SAT

**Modelling the SAT Solver:
The System DPLL**

States of DPLL

... are triples $\langle \Psi, M, C \rangle$, where

- * Ψ is a set of propositional clauses
 - representation of the problem to be solved
- * M is a sequence of literals interspersed with the checkpoint symbols \square
 - partial model with info for backtracking
- * C is a set of literals or a special symbol `no_cflct`
 - a clause entailed by Ψ but falsified by M

To check satisfiability of Φ , run DPLL with the initial state $\langle \Phi, _ , \text{no_cflct} \rangle$.

Rules of DPLL

(Improved **Abstract DPLL** of [Nieuw,Oliv,Tin])

Decide
$$\frac{l \in L \quad l, \bar{l} \notin M}{M := M + \square + l}$$

UnitPropag
$$\frac{l \vee l_1 \vee \dots \vee l_k \in \Psi \quad \bar{l}_1, \dots, \bar{l}_k \in M \quad l, \bar{l} \notin M}{M := M + l}$$

Conflict
$$\frac{C = \text{no_cflct} \quad \bar{l}_1 \vee \dots \vee \bar{l}_k \in \Psi \quad l_1, \dots, l_k \in M}{C := \{l_1, \dots, l_k\}}$$

Explain
$$\frac{l \in C \quad l \vee \bar{l}_1 \vee \dots \vee \bar{l}_k \in \Psi \quad l_1, \dots, l_k \prec l}{C := C \cup \{l_1, \dots, l_k\} \setminus \{l\}}$$

Learn
$$\frac{C = \{l_1, \dots, l_k\} \quad \bar{l}_1 \vee \dots \vee \bar{l}_k \notin \Psi}{\Psi := \Psi \cup \{\bar{l}_1 \vee \dots \vee \bar{l}_k\}}$$

BackJump
$$\frac{C = \{l, l_1, \dots, l_k\} \quad \bar{l} \vee \bar{l}_1 \vee \dots \vee \bar{l}_k \in \Psi \quad \text{level } l > m \geq \text{level } l_i \quad (i = 1, \dots, k)}{C := \text{no_cflct} \quad M := M^{[m]} + \bar{l}}$$

Correctness of DPLL

Theorem All runs of DPLL are finite.

Theorem [Verified in *Isabelle* by Filip Marić] Suppose

$$\langle \Psi, _ \text{no_cflct} \rangle \rightarrow \dots \rightarrow \langle \Psi', M, C \rangle$$

is a maximal run of DPLL. Then

$$C = \text{no_cflct} \quad \text{iff} \quad \Psi \text{ is satisfiable.}$$

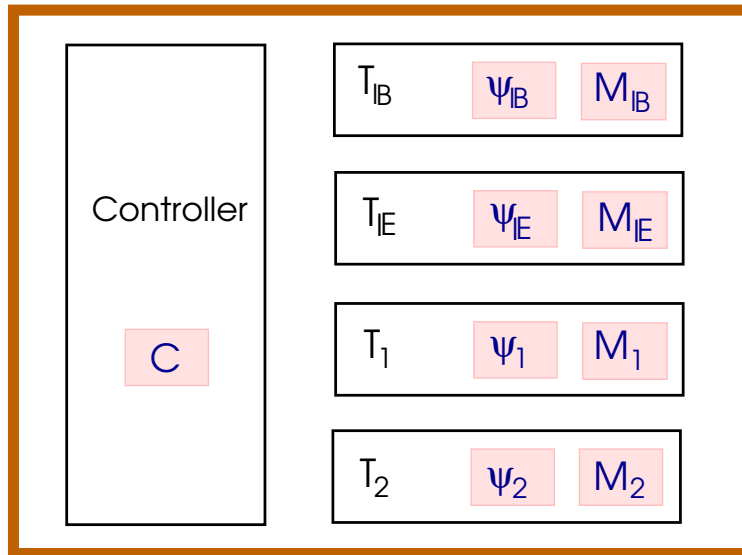
.

Chaff “is” a strategy

$((((\text{Conflict} ; \text{Explain_ui}^* ; [\text{Learn} ; \text{BackJump}]) \parallel \text{UnitPropag})^* ; [\text{Decide}])^*$

The System NODPLL

State of NODPLL

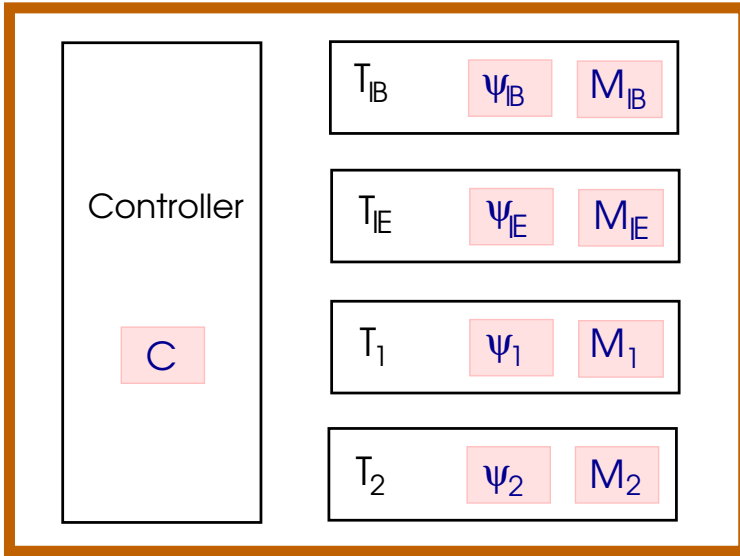


Theories T_i , $i \in \{\mathbb{B}, \mathbb{E}, 1, \dots, n\}$

Shared variable sets V_i

- Local constraints Ψ_i
- Local stacks M_i of *labeled* literals
- Conflict C —a set of lits, or *no_cflct*

State of NODPLL



Theories T_i , $i \in \{\mathbb{B}, \mathbb{E}, 1, \dots, n\}$

Shared variable sets V_i

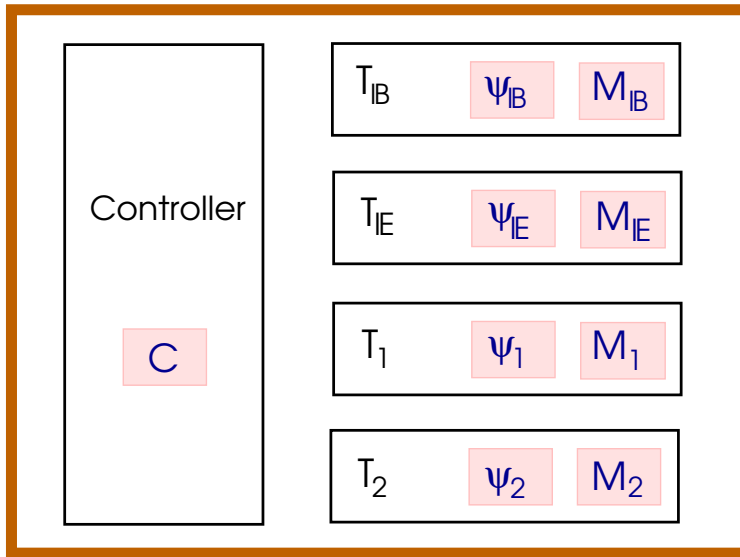
- Local constraints Ψ_i
- Local stacks M_i of *labeled* literals
- Conflict C —a set of lits, or *no_cflct*

Initialization: (1) Purify the quantifier-free input into an equisatisfiable set of simple pure formulas:

A	B	C	D
$p_1^{\pm 1} \vee \dots \vee p_k^{\pm 1}$	$p \Leftrightarrow (x = y)$	$x = t$	$p \Leftrightarrow \phi$

(2) Initialize by collecting A into $\Psi_{\mathbb{B}}$, B into $\Psi_{\mathbb{E}}$, and C, D into the Ψ_i

State of NODPLL



Theories T_i , $i \in \{\mathbb{B}, \mathbb{E}, 1, \dots, n\}$

Shared variable sets V_i

- Local constraints Ψ_i
- Local stacks M_i of *labeled* literals
- Conflict C —a set of lits, or *no_cflct*

Example.

$$A : p_1^{\pm 1} \vee \dots \vee p_k^{\pm 1} \quad B : p \Leftrightarrow (x = y) \quad C : x = t \quad D : p \Leftrightarrow \phi$$

$$\Psi_{\mathbb{B}} = \{p \vee q \vee \bar{r}, s \vee r, \dots\} \quad \Psi_{\mathbb{E}} = \{p = (x = y), r = (y = z), \dots\}$$

$$\Psi_{\text{UF}} = \{u = f @ y @ x, v = g @ x \dots\} \quad \Psi_{\text{Int}} = \{w = 2x + 3y, r \Leftrightarrow (x + y \leq 7)\}$$

Rules of NODPLL (Search)

Decide	$\frac{l \in L_{\mathbb{B}} \quad l, \bar{l} \notin M_{\mathbb{B}}}{M_{\mathbb{B}} := M_{\mathbb{B}} + \square + \langle l, \mathbb{B} \rangle \quad M_i := M_i + \square \quad (\text{all } i \neq \mathbb{B})}$
Infer _i	$\frac{l \in L_i \quad l, \bar{l} \notin M_i \quad \Psi_i, M_i \models_i l}{M_i := M_i + \langle l, i \rangle}$
Conflict _i	$\frac{C = \text{no_cflct} \quad \langle l_1, i_1 \rangle, \dots, \langle l_k, i_k \rangle \in M_i \quad \Psi_i, l_1, \dots, l_k \models_i \text{false}}{C := \{ \langle l_1, i_1 \rangle, \dots, \langle l_k, i_k \rangle \}}$
Explain _i	$\frac{\langle l, i \rangle \in C \quad \langle l_1, i_1 \rangle, \dots, \langle l_k, i_k \rangle \prec_{M_i} \langle l, i \rangle \quad \Psi_i, l_1, \dots, l_k \models_i l}{C := C \cup \{ \langle l_1, i_1 \rangle, \dots, \langle l_k, i_k \rangle \} \setminus \{ \langle l, i \rangle \}}$
Learn	$\frac{C = \{ l_1, \dots, l_k \} \quad C \subseteq L_{\mathbb{B}} \quad \bar{l}_1 \vee \dots \vee \bar{l}_k \notin \Psi_{\mathbb{B}}}{\Psi_{\mathbb{B}} := \Psi_{\mathbb{B}} \cup \{ \bar{l}_1 \vee \dots \vee \bar{l}_k \}}$
BackJump	$\frac{C = \{ l, l_1, \dots, l_k \} \quad \bar{l} \vee \bar{l}_1 \vee \dots \vee \bar{l}_k \in \Psi_{\mathbb{B}} \quad \text{level } l > m \geq \text{level } l_i \quad (i = 1, \dots, k)}{C := \text{no_cflct} \quad M_{\mathbb{B}} := M_{\mathbb{B}}^{[m]} + \langle \bar{l}, \mathbb{B} \rangle \quad M_i := M_i^{[m]} \quad (\text{all } i \neq \mathbb{B})}$

Rules of NODPLL (Exchange)

$$\text{LitDispatch}_i \quad \frac{l \in L_i \quad \langle l, j \rangle \in M_{\mathbb{B}} \quad l \notin M_i}{M_i := M_i + \langle l, j \rangle}$$

$(i \neq \mathbb{B})$

$$\text{EqDispatch}_i \quad \frac{x, y \in V_i \quad \langle x = y, j \rangle \in M_{\mathbb{E}} \quad x = y \notin M_i}{M_i := M_i + \langle x = y, j \rangle}$$

$(i \neq \mathbb{E}, \mathbb{B})$

$$\text{LitPropag}_i \quad \frac{l \in L_{\mathbb{B}} \quad \langle l, i \rangle \in M_i \quad l, \bar{l} \notin M_{\mathbb{B}}}{M_{\mathbb{B}} := M_{\mathbb{B}} + \langle l, i \rangle}$$

$(i \neq \mathbb{B})$

$$\text{EqPropag}_i \quad \frac{x, y \in V_{\mathbb{E}} \quad \langle x = y, i \rangle \in M_i \quad x = y \notin M_{\mathbb{E}}}{M_{\mathbb{E}} := M_{\mathbb{E}} + \langle x = y, i \rangle}$$

$(i \neq \mathbb{E}, \mathbb{B})$

Why Are Literals Labelled?

—To ensure accurate explanations:

$$M_{\mathbb{B}} : [] \xrightarrow{(2)} [l] \xrightarrow{(3)} [l, l']$$

$$M_1 : [] \xrightarrow{(1)} [l]$$

$$M_2 : [] \xrightarrow{(4)} [l'] \xrightarrow{(5)} [l', l]$$

Circularity!

$$M_{\mathbb{B}} : [] \xrightarrow{(2)} [\langle l, 1 \rangle] \xrightarrow{(3)} [\langle l, 1 \rangle, \langle l', \mathbb{B} \rangle]$$

$$M_1 : [] \xrightarrow{(1)} [\langle l, 1 \rangle]$$

$$M_2 : [] \xrightarrow{(4)} [\langle l', \mathbb{B} \rangle] \xrightarrow{(5)} [\langle l', \mathbb{B} \rangle, \langle l, 2 \rangle]$$

No circularity

(1) Infer₁

(2) LitPropag₁

(3) Infer_ℬ

(4) LitDispatch₂

(5) Infer₂

Correctness of NODPLL

Termination Every properly initialized run of NODPLL is finite and ends in a state where $C = \text{no_cflct}$ or $C = \emptyset$.

Soundness If a final state with $C = \emptyset$ is reachable, then the input query is unsatisfiable.

Completeness In favorable circumstances*, if a final state with $C = \text{no_cflct}$ is reachable, then the input query is satisfiable.

* either (i) all theories convex and datatypes infinite
or (ii) $\Psi_{\mathbb{E}}$ contains $p \Leftrightarrow (x = y)$ for every pair of shared variables x, y

Extensions

- ✱ For efficient SAT

$$\begin{array}{l} \text{Forget} \quad \frac{C = \text{no_cflct} \quad \phi \in \Psi_{\mathbb{B}} \quad \Psi_{\mathbb{B}} \setminus \{\phi\} \models \phi}{\Psi_{\mathbb{B}} := \Psi_{\mathbb{B}} - \{\phi\}} \\ \\ \text{Restart} \quad \frac{C = \text{no_cflct}}{M_i := M_i^{[0]} \quad (\text{all } i)} \end{array}$$

- ✱ For completeness when T_i is not convex

$$\text{ThLearn}_i \quad \frac{l_1, \dots, l_k \in L_{\mathbb{B}} \quad \Psi_i \models_i l_1 \vee \dots \vee l_k}{\Psi_{\mathbb{B}} = \Psi_{\mathbb{B}} \cup \{l_1 \vee \dots \vee l_k\}}$$

- ✱ **Delayed Theory Combination** [Bozzano et al \rightsquigarrow *MathSAT, Yices, Z3*]

- Extend initialization by introducing **proxy boolean variables** e_{xy} for all relevant equality literals $x = y$
- Dispatch/propagation of equalities can be simulated by dispatch/propagation of proxies e_{xy}
- Rules **EqDispatch** and **EqPropag** can be safely removed from the system

- ☹ Old proofs need not hold for modified system!

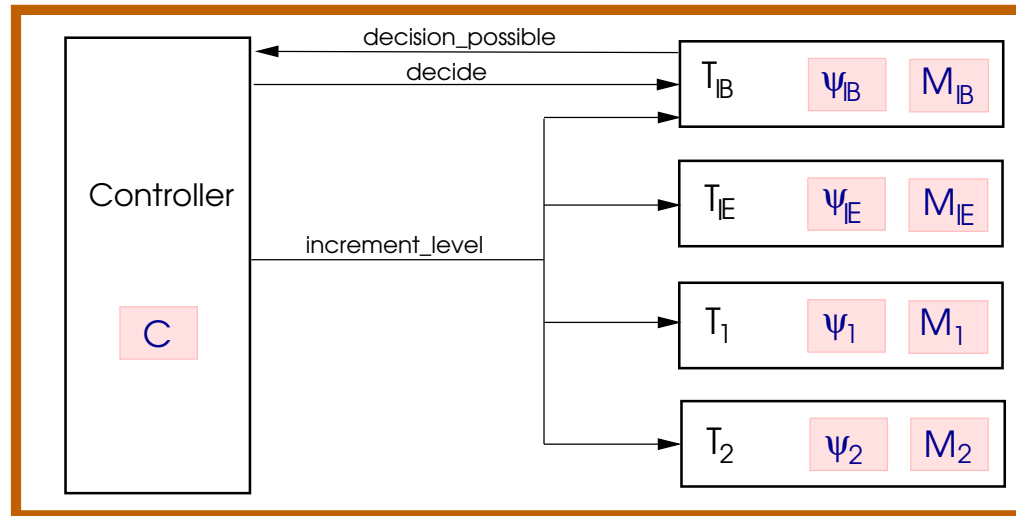
Conclusion

Raising the level of formal analysis of the architecture of SMT solvers:

- ✱ language to capture the major aspect of solver's operation
- ✱ correctness proofs
- ✱ tolerable gap between theoretical basis and implementation

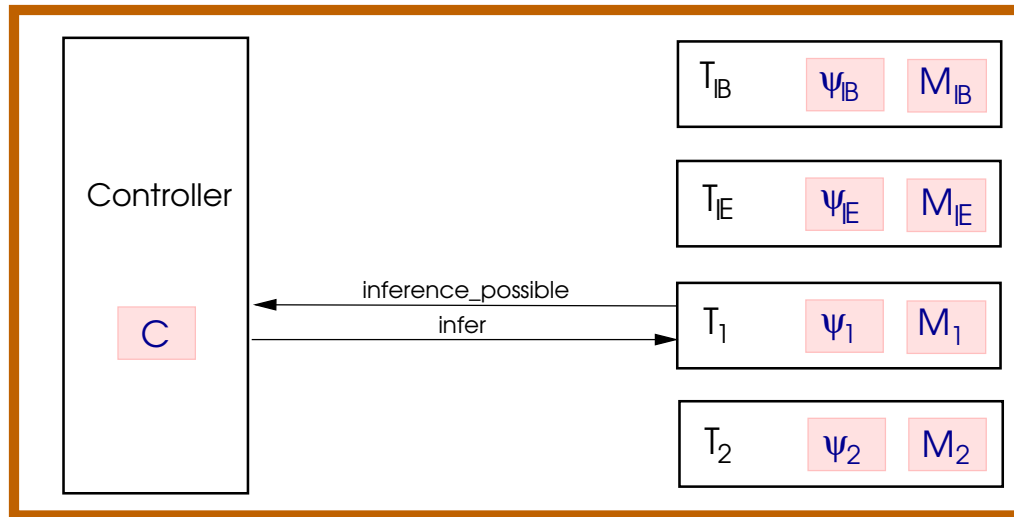
Part 2: Implementation

Rule Decide



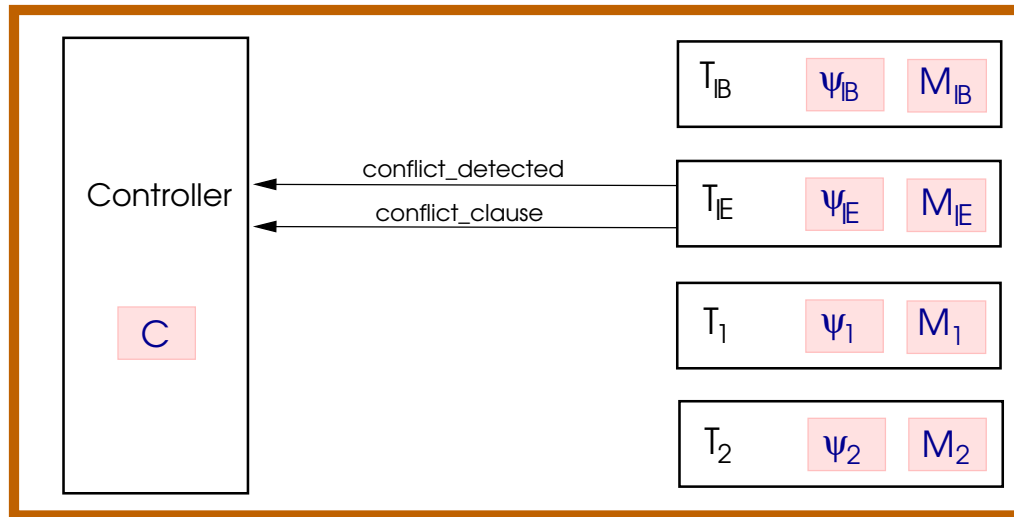
Decide	$l \in L_{\mathbb{B}} \quad l, \bar{l} \notin M_{\mathbb{B}}$
	$M_{\mathbb{B}} := M_{\mathbb{B}} + \square + \langle l, \mathbb{B} \rangle \quad M_i := M_i + \square \quad (\text{all } i \neq \mathbb{B})$

Rule Infer_i



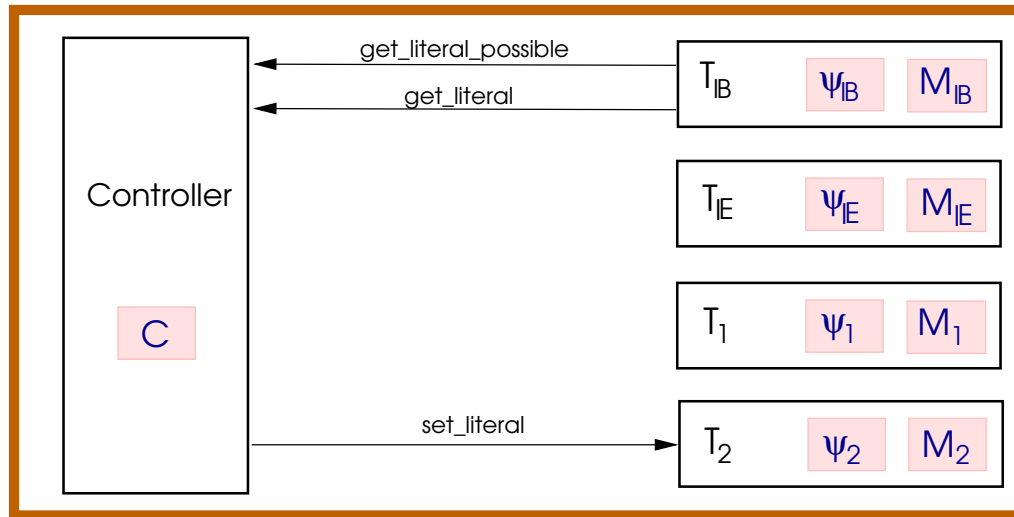
$$\text{Infer}_i \frac{l \in L_i \quad l, \bar{l} \notin M_i \quad \Psi_i, M_i \models_i l}{M_i := M_i + \langle l, i \rangle}$$

Rule Conflict_i



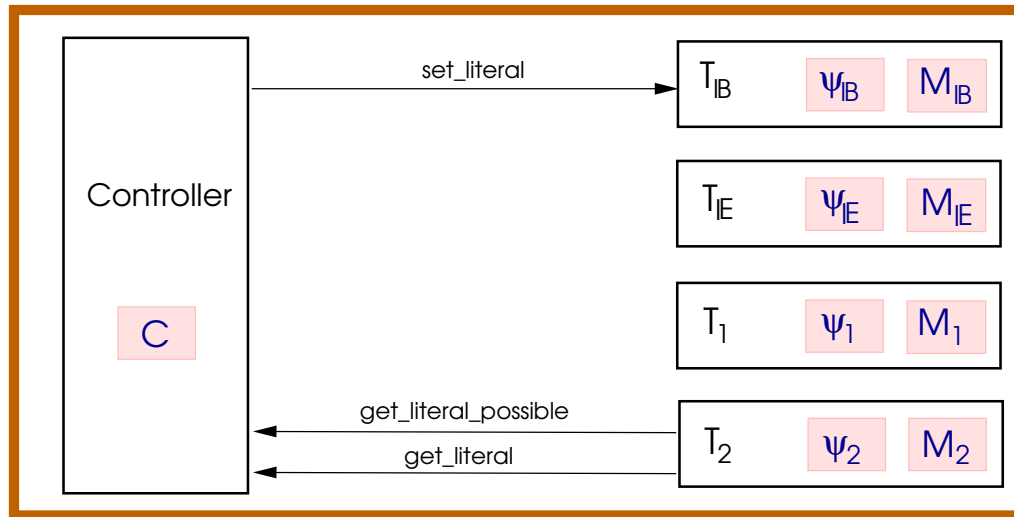
$$\text{Conflict}_i \quad \frac{C = \text{no_cflct} \quad \langle l_1, i_1 \rangle, \dots, \langle l_k, i_k \rangle \in M_i \quad \Psi_i, l_1, \dots, l_k \models_i \text{false}}{C := \{\langle l_1, i_1 \rangle, \dots, \langle l_k, i_k \rangle\}}$$

Rule LitDispatch_i



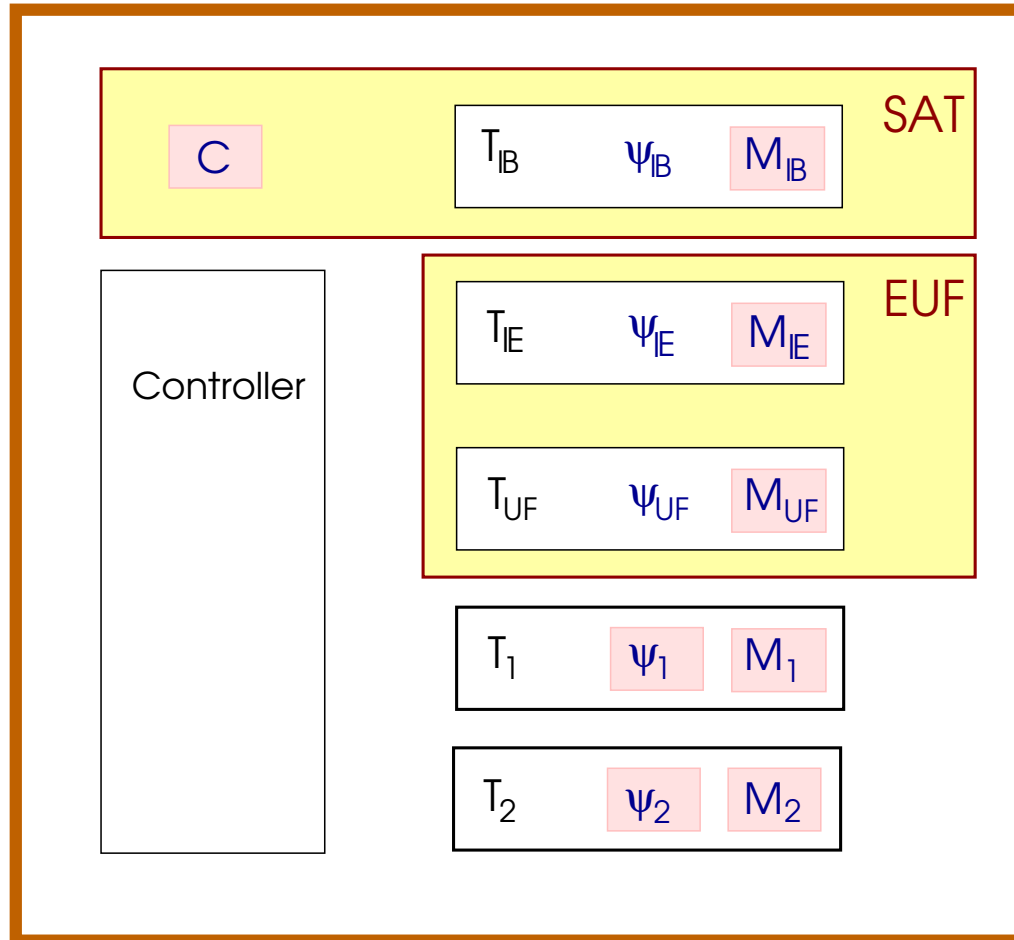
$$\text{LitDispatch}_i \quad \frac{l \in L_i \quad \langle l, j \rangle \in M_B \quad l \notin M_i}{M_i := M_i + \langle l, j \rangle}$$

Rule LitPropag_i



$$\text{LitPropag}_i \quad \frac{l \in L_{\mathbb{B}} \quad \langle l, i \rangle \in M_i \quad l, \bar{l} \notin M_{\mathbb{B}}}{M_{\mathbb{B}} := M_{\mathbb{B}} + \langle l, i \rangle}$$

A More Realistic Picture



DPT: Decision Procedure Toolkit

SMT solver developed at Intel Strategic CAD Labs

- ✱ Written in OCaml
- ✱ Open source*: **Controller + DPLL + EUF** just released
- ✱ Coming soon: **Linear Arithmetic, Arrays, DTC Controller**
- ✱ Competitive performance[†]
- ✱ Primary developers: **Amit Goel, Jim Grundy, Sava Krstić**
- ✱ Contributions by interns **Nathan Linger** and **Alexander Fuchs**
- ✱ External collaboration: **Sanjit Seshia**—bitvectors
- ✱ We seek further collaboration

*SourceForge; Apache 2 license

[†]download and check!

DPT: Any Takers?

- 1 Links to theorem provers such as HOL/*Isabelle*
- 2 Additional or better decision procedures (e.g. difference logic)
- 3 Efficient equality communication
- 4 A CamLP4-based quotation expander for DPT terms
- 5 Other contributions welcome