Theory and Practice of Decision Procedures for Combinations of Theories

Part I: Theory

Clark Barrett* and Cesare Tinelli**

*New York University **The University of Iowa

Credits

- ***Slides inspired by previous presentations by:***

  Silvio Ghilardi, Sava Krstic, Albert Oliveras, Harald Ruess, Roberto Sebastiani, Natarajan Shankar, Ashish Tiwari, Calogero Zarba, and others.

- ***Special thanks to:***

  Albert Oliveras (for contributing some of the material) and the CAV PC (for the invitation).

Prologue: The T-Validity Problem

Let T be a first-order theory of signature Σ.

Let $\mathcal{L}$ be a class of Σ-formulas.

Prologue: The T-Validity Problem

Let T be a first-order theory of signature Σ.

Let $\mathcal{L}$ be a class of Σ-formulas.

Given φ in $\mathcal{L}$, is it the case that

$$T \models \varphi \,?$$

Prologue: The Combined Validity Problem

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas.

Prologue: The Combined Validity Problem

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas.

Let $T_1 \oplus T_2$ be a combination of T_1 and T_2.

Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Prologue: The Combined Validity Problem

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas.

Let $T_1 \oplus T_2$ be a combination of T_1 and T_2.

Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Given any φ in $\mathcal{L}_1 \oplus \mathcal{L}_2$, is it the case that

$$T_1 \oplus T_2 \models \varphi \,?$$

Prologue: The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas.

such that the T_i-validity problem for $\mathcal{L}_i$ is decidable.

Prologue: The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas.

such that the T_i-validity problem for $\mathcal{L}_i$ is decidable.

Let $T_1 \oplus T_2$ be a combination of T_1 and T_2.

Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Prologue: The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas.

such that the T_i-validity problem for $\mathcal{L}_i$ is decidable.

Let $T_1 \oplus T_2$ be a combination of T_1 and T_2.

Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Is the $(T_1 \oplus T_2)$-validity problem for $\mathcal{L}_1 \oplus \mathcal{L}_2$ decidable?

Prologue: The Combined Decidability Problem II

For $i = 1, 2$,

- let P_i be a decision procedure for the T_i-validity problem for $\mathcal{L}_i$,

Let $T_1 \oplus T_2$ be a combination of T_1 and T_2.

Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Prologue: The Combined Decidability Problem II

For $i = 1, 2$,

- let P_i be a decision procedure for the T_i-validity problem for $\mathcal{L}_i$,

Let $T_1 \oplus T_2$ be a combination of T_1 and T_2.

Let $\mathcal{L}_1 \oplus \mathcal{L}_2$ be a combination of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Can we combine P_1 and P_2 modularly into a decision procedure for the $(T_1 \oplus T_2)$-validity problem for $\mathcal{L}_1 \oplus \mathcal{L}_2$?

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

FOL with Equality: Lexicon

- We will assume the following pairwise disjoint sets:
  - a countably-infinite set $X = \{x, y, z, v, \ldots\}$ of variables
  - a countably infinite set $\mathcal{F} = \{c, d, f, g, \ldots\}$ of function symbols, each with an associated arity $n \geq 0$
  - a countably infinite set $\mathcal{P} = \{p, q, \ldots\}$ of predicate symbols, each with an associated arity $n \geq 0$
- A signature Σ is a subset of $\mathcal{F} \cup \mathcal{P}$.
- If C is a set of constant (i.e. 0-arity) symbols from $\mathcal{F}$, $\Sigma(C)$ denotes the signature $\Sigma \cup C$.

FOL with Equality: Language

Let Σ be a signature and $Y \subseteq X$ a set of variable.

- Σ-terms (over Y) are defined as usual.
- Σ-formulas are defined as usual over $\wedge, \vee, \neg, \forall, \exists, \approx$.
- Free (occurrences of) variables in a formula are those not bound by a quantifier.
- Literals are atomic formulas or their negation.
- Sentences are formulas with no free variables.
- Theories are sets of sentences.

FOL with Equality: Notation

Let Σ be a signature and $Y \subseteq X$ a set of variable.

$\approx$: the equality predicate symbol.

$\mathrm{T}(\Sigma, Y)$: the set of Σ-terms over Y.

$\varphi(\mathbf{x})$: a formula whose free variables occur in the tuple $\mathbf{x}$.

$\varphi[t]$: a formula with a subterm t.

FOL with Equality: Semantics

Let Σ be a signature.

A first-order Σ-structure $\mathcal{A}$ is defined as usual as consisting of:

- a set A of elements, the domain,
- a mapping of each n-ary function symbol $f \in \Sigma$ to a total function $f^{\mathcal{A}} : A^n \to A$,
- a mapping of each n-ary predicate symbol $p \in \Sigma$ to a relation $p^{\mathcal{A}} \subseteq A^n$.

Note: the equality symbol $\approx$ is always interpreted as the identity relation.

FOL with Equality: Semantics

Let $\mathcal{A}$ denote a structure, φ a formula, and T a theory, all of signature Σ.

The reduct $\mathcal{A}^{\Omega}$ of a $\mathcal{A}$ to $\Omega \subseteq \Sigma$ is an Ω-structure with same domain and interpretation of Ω's symbols as $\mathcal{A}$.

$(\mathcal{A}, \alpha) \models \varphi$: φ is true in $\mathcal{A}$ under the variable assignment $\alpha : X \to A$.

φ is satisfiable in (satisfied by) $\mathcal{A}$: $(\mathcal{A}, \alpha) \models \varphi$ for some α.

$\perp$: a formula satisfied by no structure.

φ is valid in $\mathcal{A}$ ($\mathcal{A} \models \varphi$): $(\mathcal{A}, \alpha) \models \varphi$ for every α.

Model of T: structure in which every sentence of T is valid.

FOL with Equality: Semantics

Let $\mathcal{A}$ denote structures,
α valuations of variables into $\mathcal{A}$,
φ formulas,
Φ sets of formulas,
T theories (sets of closed formulas),
all of signature Σ.

$\Phi \models \varphi$:	For all $(\mathcal{A}, \alpha)$ if $(\mathcal{A}, \alpha) \models \Phi$ then $(\mathcal{A}, \alpha) \models \varphi$
$\Phi_1, \Phi_2, \varphi \models \psi$:	$\Phi_1 \cup \Phi_2 \cup \{\varphi\} \models \psi$.
φ is T-satisfiable:	$T, \varphi \not\models \bot$.
φ is T-valid:	$T \models \varphi$.

FOL with Equality: Homomorphisms

Let $\mathcal{A}$, $\mathcal{B}$ be Σ-structures.

A homomorphism of $\mathcal{A}$ into $\mathcal{B}$ is a function $h : A \rightarrow B$ such that

- for all $a_1, \ldots, a_n \in A$ and n-ary $f \in \Sigma$,

$$h(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h(a_1), \ldots, h(a_n))$$

- for all $a_1, \ldots, a_n \in A$ and n-ary $p \in \Sigma$,

$$(h(a_1), \ldots, h(a_n)) \in p^{\mathcal{B}} \text{ whenever } (a_1, \ldots, a_n) \in p^{\mathcal{A}}.$$

FOL with Equality: Isomorphisms

Let $\mathcal{A}$, $\mathcal{B}$ be Σ-structures with the same cardinality.

An isomorphism of $\mathcal{A}$ into $\mathcal{B}$ is an invertible function $h : A \longrightarrow B$ s.t.

- h is a homomorphism of $\mathcal{A}$ into $\mathcal{B}$,
- h^{-1} is a homomorphism of $\mathcal{B}$ into $\mathcal{A}$.

$\mathcal{A}$ and $\mathcal{B}$ are isomorphic, written $\mathcal{A} \cong \mathcal{B}$, if there is an isomorphism of $\mathcal{A}$ into $\mathcal{B}$.

Fact 1: $\cong$ is an equivalence relation over structures.

Fact 2: Isomorphic Σ-structures satisfy exactly the same Σ-formulas.

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

The T-Validity Problem

Let T be a first-order theory of signature Σ.

Let $\mathcal{L}$ be a class of Σ-formulas.

Given φ in $\mathcal{L}$, is it the case that

$$T \models \varphi\ ?$$

The T-Validity Problem

Let T be a first-order theory of signature Σ.

Let $\mathcal{L}$ be a class of Σ-formulas.

Given φ in $\mathcal{L}$, is it the case that

$$T \models \varphi\ ?$$

This problem is decidable only for restricted $\mathcal{L}$ and T.

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\forall \mathbf{x} A(\mathbf{x}) \mid A$ atomic$\}$,
  the word problem.

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\forall \mathbf{x} A(\mathbf{x}) \mid A$ atomic$\}$,
  the word problem.
- $\{\forall \mathbf{x}(A_1 \wedge \cdots \wedge A_n \rightarrow B)(\mathbf{x}) \mid A_1, \ldots, A_n, B$ atomic$\}$,
  the conditional (or uniform) word problem.

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\forall \mathbf{x} A(\mathbf{x}) \mid A$ atomic$\}$,
  the word problem.
- $\{\forall \mathbf{x}(A_1 \wedge \cdots \wedge A_n \rightarrow B)(\mathbf{x}) \mid A_1, \ldots, A_n, B$ atomic$\}$,
  the conditional (or uniform) word problem.
- $\{\forall \mathbf{x} C(\mathbf{x}) \mid C$ disjunction of literals$\}$,
  the clausal validity problem.

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\forall \mathbf{x} A(\mathbf{x}) \mid A$ atomic$\}$,
  the word problem.
- $\{\forall \mathbf{x}(A_1 \wedge \cdots \wedge A_n \rightarrow B)(\mathbf{x}) \mid A_1, \ldots, A_n, B$ atomic$\}$,
  the conditional (or uniform) word problem.
- $\{\forall \mathbf{x} C(\mathbf{x}) \mid C$ disjunction of literals$\}$,
  the clausal validity problem.
- $\{\forall \mathbf{x} \varphi(\mathbf{x}) \mid \varphi$ quantifier-free$\}$,
  the universal validity problem.

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\exists \mathbf{x} \forall \mathbf{y} (A_1 \wedge \cdots \wedge A_n)(\mathbf{x}, \mathbf{y}) \mid A_1, \ldots, A_n \text{ atomic}\}$, the unification problem (with constants).

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\exists \mathbf{x} \forall \mathbf{y} (A_1 \wedge \cdots \wedge A_n)(\mathbf{x}, \mathbf{y}) \mid A_1, \ldots, A_n$ atomic$\}$, the unification problem (with constants).
- $\{\exists \mathbf{x} \forall \mathbf{y} (L_1 \wedge \cdots \wedge L_n)(\mathbf{x}, \mathbf{y}) \mid L_1, \ldots, L_n$ literals$\}$, the disunification problem (with constants).

Common Restrictions on $\mathcal{L}$

$\mathcal{L} =$

- $\{\exists \mathbf{x} \forall \mathbf{y} (A_1 \wedge \cdots \wedge A_n)(\mathbf{x}, \mathbf{y}) \mid A_1, \ldots, A_n$ atomic$\}$,
  the unification problem (with constants).

- $\{\exists \mathbf{x} \forall \mathbf{y} (L_1 \wedge \cdots \wedge L_n)(\mathbf{x}, \mathbf{y}) \mid L_1, \ldots, L_n$ literals$\}$,
  the disunification problem (with constants).

- $\{Q\varphi \mid Q \in \{\exists, \forall\}^*, \varphi \in \varphi$ quantifier-free and positive$\}$,
  the positive validity problem.

The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas

such that the T_i-validity problem for $\mathcal{L}_i$ is decidable.

Is the $(T_1 \oplus T_2)$-satisfiability for $\mathcal{L}_1 \oplus \mathcal{L}_2$ decidable?

The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas

such that the T_i-validity problem for $\mathcal{L}_i$ is decidable.

Is the $(T_1 \oplus T_2)$-satisfiability for $\mathcal{L}_1 \oplus \mathcal{L}_2$ decidable?

In general: No.

Main issue: how $T_1 \oplus T_2$ and $\mathcal{L}_1 \oplus \mathcal{L}_2$ are defined.

The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas

such that the T_i-validity problem for $\mathcal{L}_i$ is decidable.

Is the $(T_1 \oplus T_2)$-satisfiability for $\mathcal{L}_1 \oplus \mathcal{L}_2$ decidable?

In general: No.

Main issue: how $T_1 \oplus T_2$ and $\mathcal{L}_1 \oplus \mathcal{L}_2$ are defined.

Restrictions on T_1, T_2, $\mathcal{L}_1$, $\mathcal{L}_1$, $T_1 \oplus T_2$, and $\mathcal{L}_1 \oplus \mathcal{L}_2$ are needed to answer the questions affirmatively.

The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas

Usually,

$$\Sigma_1 \cap \Sigma_2 = \emptyset,$$
$$\mathcal{L}_i = \mathcal{L}^{\Sigma_i} = \{\varphi \in \mathcal{L} \mid \varphi \text{ has signature } \Sigma_i\} \text{ for some } \mathcal{L}.$$

The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas

Usually,

$$\Sigma_1 \cap \Sigma_2 = \emptyset,$$
$$\mathcal{L}_i = \mathcal{L}^{\Sigma_i} = \{\varphi \in \mathcal{L} \mid \varphi \text{ has signature } \Sigma_i\} \text{ for some } \mathcal{L}.$$

Then, one possibility is

$$\begin{aligned} \mathcal{L}_1 \oplus \mathcal{L}_2 &= \mathcal{L}^{\Sigma_1 \cup \Sigma_2} \\ T_1 \oplus T_2 &= T_1 \cup T_2 \end{aligned}$$

The Combined Decidability Problem I

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}_i$ be a class of Σ_i-formulas

Usually,

$$\begin{aligned}&\Sigma_1 \cap \Sigma_2 = \emptyset,\\ &\mathcal{L}_i = \mathcal{L}^{\Sigma_i} = \{\varphi \in \mathcal{L} \mid \varphi \text{ has signature } \Sigma_i\} \text{ for some } \mathcal{L}.\end{aligned}$$

Then, one possibility is

$$\begin{aligned}\mathcal{L}_1 \oplus \mathcal{L}_2 &= \mathcal{L}^{\Sigma_1 \cup \Sigma_2}\\ T_1 \oplus T_2 &= T_1 \cup T_2\end{aligned}$$

We will focus on this case here.

The Combined Decidability Problem II

Assume

- P_1, a procedure deciding the T_1-validity problem for $\mathcal{L}^{\Sigma_1}$,
- P_2, a procedure deciding the T_2-validity problem for $\mathcal{L}^{\Sigma_2}$.

Can we compose P_1 and P_2 modularly into a procedure that decides the $(T_1 \cup T_2)$-validity problem for $\mathcal{L}^{\Sigma_1 \cup \Sigma_2}$?

The Combined Decidability Problem II

Assume

- P_1, a procedure deciding the T_1-validity problem for $\mathcal{L}^{\Sigma_1}$,
- P_2, a procedure deciding the T_2-validity problem for $\mathcal{L}^{\Sigma_2}$.

Can we compose P_1 and P_2 modularly into a procedure that decides the $(T_1 \cup T_2)$-validity problem for $\mathcal{L}^{\Sigma_1 \cup \Sigma_2}$?

Almost invariably, additional functionalities are required of P_1 and P_2 (more on this in Part II).

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

The T-Satisfiability Problem

Every T-validity problem has a dual T-satisfiability problem.

Note: $T \models \varphi$ iff $T, \neg\varphi \models \bot$

The T-Satisfiability Problem

Every T-validity problem has a dual T-satisfiability problem.

Note: $T \models \varphi$ iff $T, \neg\varphi \models \bot$

Hence the T-validity problem for $\mathcal{L}$ is reducible to the T-satisfiability problem for $\mathcal{L}_{\mathrm{D}} = \{\neg\psi \mid \psi \in \mathcal{L}\}$:

Given $\psi \in \mathcal{L}_{\mathrm{D}}$, is ψ is T-satisfiable?

The T-Satisfiability Problem

Every T-validity problem has a dual T-satisfiability problem.

Note: $T \models \varphi$ iff $T, \neg\varphi \models \bot$

Hence the T-validity problem for $\mathcal{L}$ is reducible to the T-satisfiability problem for $\mathcal{L}_{\mathrm{D}} = \{\neg\psi \mid \psi \in \mathcal{L}\}$:

Given $\psi \in \mathcal{L}_{\mathrm{D}}$, is ψ is T-satisfiable?

For combination purposes, it is more convenient to work with satisfiability problems.

T-satisfiability vs. Constraint Solving

The field of Constraint Solving also deals with satisfiability problems.

But be careful:

- In Constraint Solving one is interested in whether a formula $\psi \in \mathcal{L}$ is satisfiable in a given, fixed model of a theory T.
- In constrast, in T-satisfiability one is interested in whether ψ is satisfiable in any model of T at all.

These are different problems!

T-satisfiability vs. Constraint Solving

Unfortunately, to confuse things, there are

(i) languages $\mathcal{L}$, (ii) theories T and (iii) structures $\mathcal{A}$

for which the two problems are equivalent:

for all $\psi \in \mathcal{L}$, ψ is T-satisfiable iff ψ is satisfiable in $\mathcal{A}$.

T-satisfiability vs. Constraint Solving

Unfortunately, to confuse things, there are

(i) languages $\mathcal{L}$, (ii) theories T and (iii) structures $\mathcal{A}$

for which the two problems are equivalent:

for all $\psi \in \mathcal{L}$, ψ is T-satisfiable iff ψ is satisfiable in $\mathcal{A}$.

Examples:

- (i) FOL formulas, (ii) the theory of real closed fields, (iii) the structure of the real numbers.
- (i) unification problems, (ii) any equational theory E, (iii) the initial model of E.

T-satisfiability vs. Constraint Solving

Unfortunately, to confuse things, there are

(i) languages $\mathcal{L}$, (ii) theories T and (iii) structures $\mathcal{A}$

for which the two problems are equivalent:

for all $\psi \in \mathcal{L}$, ψ is T-satisfiable iff ψ is satisfiable in $\mathcal{A}$.

Nevertheless, when theories are combined this equivalence may be lost.

Be warned.

The Combined Satisfiability Problem

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}^{\Sigma_i}$ be a class of Σ_i-formulas

such that the T_i-satisfiability problem for $\mathcal{L}^{\Sigma_i}$ is decidable.

The Combined Satisfiability Problem

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}^{\Sigma_i}$ be a class of Σ_i-formulas

such that the T_i-satisfiability problem for $\mathcal{L}^{\Sigma_i}$ is decidable.

Combination methods apply to languages $\mathcal{L}^{\Sigma_1 \cup \Sigma_2}$ that are effectively purifiable for T_1 and T_2,

The Combined Satisfiability Problem

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let $\mathcal{L}^{\Sigma_i}$ be a class of Σ_i-formulas

such that the T_i-satisfiability problem for $\mathcal{L}^{\Sigma_i}$ is decidable.

Combination methods apply to languages $\mathcal{L}^{\Sigma_1 \cup \Sigma_2}$ that are effectively purifiable for T_1 and T_2, i.e., such that

the $(T_1 \cup T_2)$-satisfiability of a formula $\varphi \in \mathcal{L}^{\Sigma_1 \cup \Sigma_2}$

is effectively reducible to

the $(T_1 \cup T_2)$-satisfiability of formulas of the form $\varphi_1 \wedge \varphi_2$ with $\varphi_i \in \mathcal{L}^{\Sigma_i}$ for $i = 1, 2$.

An Effectively Purifiable Language

The language of conjunctions of literals is effectively purifiable for any T_1 and T_2.

An Effectively Purifiable Language

The language of conjunctions of literals is effectively purifiable for any T_1 and T_2.

Let φ be a conjunction of $(\Sigma_1 \cup \Sigma_2)$-literals.

1. Apply to completion to φ (modulo AC of $\wedge$) the following term abstraction rule:

$$\frac{L[t] \wedge \psi}{L[x] \wedge x \approx t \wedge \psi} \quad \text{if} \quad \begin{array}{l} x \text{ is a fresh variable and} \\ t \text{ is an alien subterm of } L \end{array}$$

2. Group the Σ_1-literals in φ_1 and the rest in φ_2.

An Effectively Purifiable Language

The language of conjunctions of literals is effectively purifiable for any T_1 and T_2.

Let φ be a conjunction of $(\Sigma_1 \cup \Sigma_2)$-literals.

1. Apply to completion to φ (modulo AC of $\wedge$) the following term abstraction rule:

$$\frac{L[t] \wedge \psi}{L[x] \wedge x \approx t \wedge \psi} \quad \text{if} \quad \begin{array}{l} x \text{ is a fresh variable and} \\ t \text{ is an alien subterm of } L \end{array}$$

2. Group the Σ_1-literals in φ_1 and the rest in φ_2.

Proposition For every $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{A}$, φ is satisfiable in $\mathcal{A}$ iff $\varphi_1 \wedge \varphi_2$ is satisfiable in $\mathcal{A}$.

Alien Subterms

Let $\Sigma_0 = \Sigma_1 \cap \Sigma_2$ and $i \in \{1, 2\}$.

A term $t \in \mathrm{T}(\Sigma_1 \cup \Sigma_2, X)$ is an i-term
if $t \in X$ or $t = f(t_1, \ldots, t_n)$ with $f \in \Sigma_i$.

Let $t[s] \in \mathrm{T}(\Sigma_1 \cup \Sigma_2, X)$,

Case 1: the top symbol of t is in $\Sigma_i \setminus \Sigma_0$
s is an alien subterm of t
if every superterm of s in t is an i-term, but s is not.

Case 2: the top symbol of t is in Σ_0.
Consider it arbitrarily as a symbol of Σ_1 or of Σ_2 and
proceed as in Case 1. (See [BT02] for a better definition.)

Alien Subterms

Let $\Sigma_0 = \Sigma_1 \cap \Sigma_2$ and $i \in \{1, 2\}$.

Let $L = (\neg)A[s]$ be a $(\Sigma_1 \cup \Sigma_2)$-literal.

The term s is an alien subterm of L
if it is an alien subterm of $A[s]$
when A's top symbol is treated as a function symbol,
with $\approx$ treated as a symbol of Σ_0.

A Larger Effectively Purifiable Language

The language of quantifier free formulas is effectively purifiable for any T_1 and T_2.

A Larger Effectively Purifiable Language

The language of quantifier free formulas is effectively purifiable for any T_1 and T_2.

Let $\varphi \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$.

1. Let $\psi_1 \vee \cdots \vee \psi_n$ be φ's disjunctive normal form.
2. Purify each disjunct ψ_j into $\psi_{j,1} \wedge \psi_{j,2}$.

A Larger Effectively Purifiable Language

The language of quantifier free formulas is effectively purifiable for any T_1 and T_2.

Let $\varphi \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$.

1. Let $\psi_1 \vee \cdots \vee \psi_n$ be φ's disjunctive normal form.
2. Purify each disjunct ψ_j into $\psi_{j,1} \wedge \psi_{j,2}$.

For any $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{A}$,
φ is satisfiable in $\mathcal{A}$ iff
$\psi_{j,1} \wedge \psi_{j,2}$ is satisfiable in $\mathcal{A}$ for some $j \in \{1, \ldots, n\}$.

A Larger Effectively Purifiable Language

The language of quantifier free formulas is effectively purifiable for any T_1 and T_2.

Let $\varphi \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$.

1. Let $\psi_1 \vee \cdots \vee \psi_n$ be φ's disjunctive normal form.
2. Purify each disjunct ψ_j into $\psi_{j,1} \wedge \psi_{j,2}$.

For any $(\Sigma_1 \cup \Sigma_2)$-structure $\mathcal{A}$,
φ is satisfiable in $\mathcal{A}$ iff
$\psi_{j,1} \wedge \psi_{j,2}$ is satisfiable in $\mathcal{A}$ for some $j \in \{1, \ldots, n\}$.

Exercise.** Purify φ by first turning it into conjunctive normal form. Proof that satisfiability in any structure is preserved.
(Hint: every conjunct $C[s]$ is equisatisfiable with $x \not\approx s \vee C[x]$ for a fresh x.)

More Effectively Purifiable Languages

A few more complex languages are effective purifiable, for given theories T_1 and T_2, if one is allowed to introduce additional (free/uninterpreted) symbols.

For instance, the full language of $FOL^{\approx}$ is effectively purifiable for any T_1 and T_2. (How? **Exercise*****.)

Combined Satisfiability of Pure Literals

From now on, wlog we consider only

combined satisfiability problems of the form

$$\varphi_1 \wedge \varphi_2$$

where each φ_i is a Σ_i-formula.

Combined Satisfiability of Pure Literals

From now on, wlog we consider only

combined satisfiability problems of the form

$$\varphi_1 \wedge \varphi_2$$

where each φ_i is a Σ_i-formula.

Observation: Such problems are really just interpolation problems.

Combined Satisfiability as Interpolation

For $i = 1, 2$, let T_i-be a Σ_i-theory and $\varphi_i(\mathbf{x}_i)$ a Σ_i-formula.

$\varphi_1 \wedge \varphi_2$ is $(T_1 \cup T_2)$-unsatisfiable

Combined Satisfiability as Interpolation

For $i = 1, 2$, let T_i-be a Σ_i-theory and $\varphi_i(\mathbf{x}_i)$ a Σ_i-formula.

$\varphi_1 \wedge \varphi_2$ is $(T_1 \cup T_2)$-unsatisfiable

iff

$$(T_1, \varphi_1), (T_2, \varphi_2) \models \perp$$

Combined Satisfiability as Interpolation

For $i = 1, 2$, let T_i-be a Σ_i-theory and $\varphi_i(\mathbf{x}_i)$ a Σ_i-formula.

$\varphi_1 \wedge \varphi_2$ is $(T_1 \cup T_2)$-unsatisfiable

iff

$$(T_1, \varphi_1), (T_2, \varphi_2) \models \perp$$

iff (by Craig's interpolation lemma)

there is a $(\Sigma_1 \cap \Sigma_2)$-formula $\varphi(\mathbf{x})$ with $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2$ s.t.

$$T_1, \varphi_1 \models \varphi \quad \text{and} \quad T_2, \varphi_2, \varphi \models \perp$$

Combined Satisfiability as Interpolation

For $i = 1, 2$, let T_i-be a Σ_i-theory and $\varphi_i(\mathbf{x}_i)$ a Σ_i-formula.

$\varphi_1 \wedge \varphi_2$ is $(T_1 \cup T_2)$-unsatisfiable

iff

$$(T_1, \varphi_1), (T_2, \varphi_2) \models \bot$$

iff (by Craig's interpolation lemma)

there is a $(\Sigma_1 \cap \Sigma_2)$-formula $\varphi(\mathbf{x})$ with $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2$ s.t.

$T_1, \varphi_1 \models \varphi$ and $T_2, \varphi_2, \varphi \models \bot$

The problem then is "just" computing the interpolant φ.

Combined Satisfiability as Interpolation

For $i = 1, 2$, let T_i-be a Σ_i-theory and $\varphi_i(\mathbf{x}_i)$ a Σ_i-formula.

$\varphi_1 \wedge \varphi_2$ is $(T_1 \cup T_2)$-unsatisfiable

iff

$$(T_1, \varphi_1), (T_2, \varphi_2) \models \bot$$

iff (by Craig's interpolation lemma)

there is a $(\Sigma_1 \cap \Sigma_2)$-formula $\varphi(\mathbf{x})$ with $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2$ s.t.

$$T_1, \varphi_1 \models \varphi \quad \text{and} \quad T_2, \varphi_2, \varphi \models \bot$$

Unfortunately, Craig's lemma provides no information on

- what φ looks like or
- how to compute φ without an explicit proof that $T_1, T_2, \varphi_1, \varphi_2 \models \bot$.

Combined Satisfiability as Interpolation

For $i = 1, 2$, let T_i-be a Σ_i-theory and $\varphi_i(\mathbf{x}_i)$ a Σ_i-formula.

$\varphi_1 \wedge \varphi_2$ is $(T_1 \cup T_2)$-unsatisfiable

iff

$$(T_1, \varphi_1), (T_2, \varphi_2) \models \bot$$

iff (by Craig's interpolation lemma)

there is a $(\Sigma_1 \cap \Sigma_2)$-formula $\varphi(\mathbf{x})$ with $\mathbf{x} = \mathbf{x}_1 \cap \mathbf{x}_2$ s.t.

$$T_1, \varphi_1 \models \varphi \quad \text{and} \quad T_2, \varphi_2, \varphi \models \bot$$

All existing combination methods are in essence ways to compute φ, possibly incrementally, in finite time.

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

The Combination Problem for Universal Formulas

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let P_i be a procedure that decides the T_i-validity problem for universal Σ_i-formulas.

The Combination Problem for Universal Formulas

For $i = 1, 2$,

- let T_i a first-order theory of signature Σ_i and
- let P_i be a procedure that decides the T_i-validity problem for universal Σ_i-formulas.

How to decide the $(T_1 \cup T_2)$-validity problem for universal $(\Sigma_1 \cup \Sigma_2)$-formulas using P_1 and P_2 modularly?

The Combination Problem for Universal Formulas

- Problem most people mean when talking about combining decision procedures.
- Problem with the largest impact and most practical uses so far.
- Most common settings:
  - T_1 and T_2 are signature-disjoint.
  - presented as a satisfiability problem for qffs
    (as $T \models \forall \mathbf{x} \varphi(\mathbf{x})$ iff $\neg\varphi(\mathbf{x})$ is T-unsatisfiable).
- Basic combination method for the problem due to Greg Nelson and Derek Oppen [NO79].

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

The Nelson-Oppen Method

- For $i = 1, 2$, let T_i a first-order theory of signature Σ_i.
- Let $T = T_1 \cup T_2$.
- Let C be a set of free constants (i.e., not in $\Sigma_1 \cup \Sigma_2$).

The Nelson-Oppen Method

- For $i = 1, 2$, let T_i a first-order theory of signature Σ_i.
- Let $T = T_1 \cup T_2$.
- Let C be a set of free constants (i.e., not in $\Sigma_1 \cup \Sigma_2$).

We consider only input problems of the form

$$\Gamma_1 \cup \Gamma_2$$

where each Γ_i is a finite set of ground $\Sigma_i(C)$-literals.

The Nelson-Oppen Method

No loss of generality in considering ground $\Sigma_i(C)$-literals as:

The Nelson-Oppen Method

No loss of generality in considering ground $\Sigma_i(C)$-literals as:

1. for each $\varphi(\mathbf{x}) \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$,

   $\varphi(\mathbf{x})$ is T-sat iff $\varphi(\mathbf{c})$ is T-sat for some $\mathbf{c}$ in C

The Nelson-Oppen Method

No loss of generality in considering ground $\Sigma_i(C)$-literals as:

1. for each $\varphi(\mathbf{x}) \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$,
   $\varphi(\mathbf{x})$ is T-sat iff $\varphi(\mathbf{c})$ is T-sat for some $\mathbf{c}$ in C
2. for each ground $\varphi(\mathbf{c})$,
   $\varphi(\mathbf{c})$ is T-sat iff one disjunct ψ of $\varphi(\mathbf{c})$'s DNF is T-sat

The Nelson-Oppen Method

No loss of generality in considering ground $\Sigma_i(C)$-literals as:

1. for each $\varphi(\mathbf{x}) \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$,

   $\varphi(\mathbf{x})$ is T-sat iff $\varphi(\mathbf{c})$ is T-sat for some $\mathbf{c}$ in C

2. for each ground $\varphi(\mathbf{c})$,

   $\varphi(\mathbf{c})$ is T-sat iff one disjunct ψ of $\varphi(\mathbf{c})$'s DNF is T-sat

3. for each conjunction ψ of literals,

   ψ is T-sat iff its separate form $\psi_1 \wedge \psi_2$ is T-sat

The Nelson-Oppen Method

No loss of generality in considering ground $\Sigma_i(C)$-literals as:

1. for each $\varphi(\mathbf{x}) \in \mathrm{QF}(\Sigma_1 \cup \Sigma_2, X)$,

   $\varphi(\mathbf{x})$ is T-sat iff $\varphi(\mathbf{c})$ is T-sat for some $\mathbf{c}$ in C

2. for each ground $\varphi(\mathbf{c})$,

   $\varphi(\mathbf{c})$ is T-sat iff one disjunct ψ of $\varphi(\mathbf{c})$'s DNF is T-sat

3. for each conjunction ψ of literals,

   ψ is T-sat iff its separate form $\psi_1 \wedge \psi_2$ is T-sat

4. for each conjunction $\psi_1 \wedge \psi_2$ of literals,

   $\psi_1 \wedge \psi_2$ is T-sat iff $\Gamma_1 \cup \Gamma_2$ is T-sat
   where each Γ_i is the set of literals in ψ_i.

The Nelson-Oppen Method

Barebone, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

The Nelson-Oppen Method

Barebone, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

Input:	$\Gamma_1 \cup \Gamma_2$ with Γ_i a finite set of ground $\Sigma_i(C)$-literals.
Output:	**sat** or **unsat**.

The Nelson-Oppen Method

Barebone, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

Input: $\Gamma_1 \cup \Gamma_2$ with Γ_i a finite set of ground $\Sigma_i(C)$-literals.
Output: **sat** or **unsat**.

1. Guess an arrangement Δ, that is:
   - Choose any equivalence relation R on the constants from C shared by Γ_1 and Γ_2.
   - Let $\Delta = \{c \approx d \mid cRd\} \cup \{c \not\approx d \mid$ not $cRd\}$

The Nelson-Oppen Method

Barebone, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

Input: $\Gamma_1 \cup \Gamma_2$ with Γ_i a finite set of ground $\Sigma_i(C)$-literals.
Output: **sat** or **unsat**.

1. Guess an arrangement Δ, that is:
   - Choose any equivalence relation R on the constants from C shared by Γ_1 and Γ_2.
   - Let $\Delta = \{c \approx d \mid cRd\} \cup \{c \not\approx d \mid$ not $cRd\}$
2. If $\Gamma_i \cup \Delta$ is T_i-unsatisfiable for $i = 1$ or $i = 2$, return **unsat**

The Nelson-Oppen Method

Barebone, non-deterministic, non-incremental version [Opp80, Rin96, TH96]:

Input: $\Gamma_1 \cup \Gamma_2$ with Γ_i a finite set of ground $\Sigma_i(C)$-literals.
Output: **sat** or **unsat**.

1. Guess an arrangement Δ, that is:
   - Choose any equivalence relation R on the constants from C shared by Γ_1 and Γ_2.
   - Let $\Delta = \{c \approx d \mid cRd\} \cup \{c \not\approx d \mid$ not $cRd\}$
2. If $\Gamma_i \cup \Delta$ is T_i-unsatisfiable for $i = 1$ or $i = 2$, return **unsat**
3. Otherwise, return **sat**

Total Correctness of the NO Method

The method is always terminating because there is only a finite number of arrangements to guess.

Total Correctness of the NO Method

The method is always terminating because there is only a finite number of arrangements to guess.

When

- $\Sigma_1 \cap \Sigma_2 = \emptyset$ and
- T_1 and T_2 are stably infinite,

the method is sound and complete.

Soundness:

If the answer is **unsat** for every arrangement,
then the input is $(T_1 \cup T_2)$-unsatisfiable.

Completeness:

If the input is $(T_1 \cup T_2)$-is unsatisfiable,
then the answer is **unsat** for every arrangement.

Stably Infinite Theories

A Σ-theory T is stably infinite iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T.

Stably Infinite Theories

A Σ-theory T is stably infinite iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T.

Many *interesting* theories are stably infinite:

- Theories of an infinite structure.
- Complete theories with an infinite model.
- Convex theories with no trivial models (see later).

Stably Infinite Theories

A Σ-theory T is stably infinite iff every quantifier-free T-satisfiable formula is satisfiable in an infinite model of T.

Many *interesting* theories are stably infinite:

- Theories of an infinite structure.
- Complete theories with an infinite model.
- Convex theories with no trivial models (see later).

But others are not stably infinite:

- Theories of a finite structure.
- Theories with models of bounded cardinality.
- Some equational/Horn theories.

The NO Method: Soundness Proof

Recall:

If the answer is **unsat** for every arrangement, then the input $\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-unsatisfiable.

Equivalently (because of guaranteed termination):

If the input $\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-satisfiable, then the answer is **sat** for some arrangement.

Proof Sketch: Let C_0 be the free constants shared by Γ_1 and Γ_2. Let $\mathcal{A}$ be a $\Sigma_1(C) \cup \Sigma_2(C)$-model of $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2$. Let $\Delta = \{c \approx d \mid c, d \in C_0, c^{\mathcal{A}} = d^{\mathcal{A}}\} \cup \{c \not\approx d \mid c, d \in C_0, c^{\mathcal{A}} \neq d^{\mathcal{A}}\}$. The set Δ is a possible arrangement of C_0. Moreover, $\mathcal{A}^{\Sigma_i(C)} \models T_i \cup \Gamma_i \cup \Delta$ for $i = 1, 2$. So the procedure will return **sat** for Δ's choice.

The Combined Satisfiability Theorem

Theorem [TR03] For $i = 1, 2$, let Φ_i be a set of Ω_i-sentences. The following are equivalent:

1. $\Phi_1 \cup \Phi_2$ is satisfiable.
2. There are an Ω_1-structure $\mathcal{A}$ satisfying Φ_1 and a Ω_2-structure $\mathcal{B}$ satisying Φ_2 such that

$$\mathcal{A}^{\Omega_1 \cap \Omega_2} \cong \mathcal{B}^{\Omega_1 \cap \Omega_2}.$$

The Combined Satisfiability Theorem

Theorem [TR03] For $i = 1, 2$, let Φ_i be a set of Ω_i-sentences. The following are equivalent:

1. $\Phi_1 \cup \Phi_2$ is satisfiable.
2. There are an Ω_1-structure $\mathcal{A}$ satisfying Φ_1 and a Ω_2-structure $\mathcal{B}$ satisying Φ_2 such that
$$\mathcal{A}^{\Omega_1 \cap \Omega_2} \cong \mathcal{B}^{\Omega_1 \cap \Omega_2}.$$

Proof Sketch.

$1 \Rightarrow 2$. Assume some $(\Omega_1 \cap \Omega_2)$-structure $\mathcal{C}$ satisfies $\Phi_1 \cup \Phi_2$. Then, $\mathcal{A} = \mathcal{C}^{\Omega_1}$ and $\mathcal{B} = \mathcal{C}^{\Omega_2}$ will do.

The Combined Satisfiability Theorem

Theorem [TR03] For $i = 1, 2$, let Φ_i be a set of Ω_i-sentences. The following are equivalent:

1. $\Phi_1 \cup \Phi_2$ is satisfiable.
2. There are an Ω_1-structure $\mathcal{A}$ satisfying Φ_1 and a Ω_2-structure $\mathcal{B}$ satisying Φ_2 such that

$$\mathcal{A}^{\Omega_1 \cap \Omega_2} \cong \mathcal{B}^{\Omega_1 \cap \Omega_2}.$$

Proof Sketch.

$2 \Rightarrow 1$. If $\mathcal{A}^{\Sigma_1 \cap \Sigma_2} \cong \mathcal{B}^{\Sigma_1 \cap \Sigma_2}$, then $\mathcal{A}$ and $\mathcal{B}$ have the same cardinality and agree on the shared symbols. Then, they can be amalgamated into a $(\Omega_1 \cap \Omega_2)$-structure $\mathcal{C}$ such that $\mathcal{A} \cong \mathcal{C}^{\Omega_1}$ and $\mathcal{B} \cong \mathcal{C}^{\Omega_2}$. Clearly, $\mathcal{C}$ satisfies both Φ_1 and Φ_2.

The NO Method: Completeness Proof

Recall:

If the input $\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-is unsatisfiable, then the answer is **unsat** for every arrangement.

The NO Method: Completeness Proof

Equivalently:

If the answer is **sat** for some arrangement, then the input $\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-satisfiable.

Proof Sketch: Let C_i collect the free constants in Γ_i for $i = 1, 2$, and let $C_0 = C_1 \cap C_2$. Let Δ be an arrangement of C_0 and $\mathcal{A}_i$ a $\Sigma_i(C_i)$-model of $T_i \cup \Gamma_i \cup \Delta$ for $i = 1, 2$.

By the stable infiniteness of each T_i, we can assume that $\mathcal{A}_i$ is infinite. By the Löwenheim-Skolem theorems, we can assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same cardinality. Since they both satisfy Δ and their signatures share only C_0, one can show that $\mathcal{A}_1^{C_0} \cong \mathcal{A}_2^{C_0}$. By the Combined Satisfiability Theorem, $T_1 \cup \Gamma_1 \cup T_2 \cup \Gamma_2$ is satisfiable.

The NO Calculus

Declarative, non-deterministic, incremental version of the NO method

The NO Calculus

Declarative, non-deterministic, incremental version of the NO method

Let C_0 be the free constants shared by the initial Γ_1^0 and Γ_2^0.

The NO Calculus

Declarative, non-deterministic, incremental version of the NO method

Let C_0 be the free constants shared by the initial Γ_1^0 and Γ_2^0.

Apply these rules exhaustively, starting with the triple $\Gamma_1^0; \emptyset; \Gamma_2^0$:

The NO Calculus

Declarative, non-deterministic, incremental version of the NO method

Let C_0 be the free constants shared by the initial Γ_1^0 and Γ_2^0.

Apply these rules exhaustively, starting with the triple $\Gamma_1^0;\ \emptyset;\ \Gamma_2^0$:

$$\frac{\Gamma_1;\ \Delta;\ \Gamma_2}{\bot} \quad \text{if } \Gamma_i, \Delta \models_{T_i} \bot \text{ for } i = 1 \text{ or } i = 2$$

$$\frac{\Gamma_1;\ \Delta;\ \Gamma_2}{\Gamma_1;\ \Delta, c \approx d;\ \Gamma_2 \qquad \Gamma_1;\ \Delta, c \not\approx d;\ \Gamma_2} \quad \text{if } \begin{cases} c, d \in C_0, \\ c \approx d \notin \Delta, \\ c \not\approx d \notin \Delta \end{cases}$$

Correctness of the NO Calculus

Some terminology:

- A derivation tree in the NO calculus is a tree such that
  - every node is either a triple $\Gamma;\ \Delta;\ \Gamma$ or $\perp$
  - a node N is a child of a M only if it is a direct consequence of M.
- A derivation tree for $\Gamma_1;\ \Delta;\ \Gamma_2$ is a derivation tree with root $\Gamma_1;\ \Delta;\ \Gamma_2$.
- A refutation tree is a derivation tree all of whose leaves are $\perp$.

Correctness of the NO Calculus

The NO calculus is sound, complete and terminating whenever T_1 and T_2 are stably infinite and signature-disjoint.

Termination:

Every derivation tree in NO is finite.

Soundness and Completeness:

$\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-unsatisfiable
iff
$\Gamma_1;\ \emptyset;\ \Gamma_2$ has a refutation tree in NO.

Proof: Exercise*

The d-NO Calculus

Declarative, (more) deterministic, incremental version of the NO method (more faithful to the original [NO79])

The d-NO Calculus

Declarative, (more) deterministic, incremental version of the NO method (more faithful to the original [NO79])

Apply these rules exhaustively, starting with $\Gamma_1^0;\ \emptyset;\ \Gamma_2^0$:

$$\frac{\Gamma_1;\ \Delta;\ \Gamma_2}{\bot} \quad \text{if } \Gamma_i, \Delta \models_{T_i} \bot \text{ for } i = 1 \text{ or } i = 2$$

$$\frac{\Gamma_1;\ \Delta;\ \Gamma_2}{\Gamma_1;\ \Delta, c_1 \approx d_1;\ \Gamma_2 \qquad \cdots \qquad \Gamma_1;\ \Delta, c_n \approx d_n;\ \Gamma_2} \quad \text{if } (*)$$

$$(*) = \begin{cases} n \geq 1,\ c_1, \ldots, c_n, d_1, \ldots, d_n \in C_0, \\ i \in \{1, 2\},\ J = \{1, \ldots, n\}, \\ \Gamma_i, \Delta \models_{T_i} \bigvee_{j \in J} c_j \approx d_j \\ \Gamma_i, \Delta \not\models_{T_i} \bigvee_{j \in J'} c_j \approx d_j \text{ for any } J' \subsetneq J \end{cases}$$

Correctness of the d-NO Calculus

The d-NO calculus is sound, complete and terminating whenever T_1 and T_2 are stably infinite and signature-disjoint.

Termination:

Every derivation tree in d-NO is finite.

Soundness and Completeness:

$\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-unsatisfiable
iff
$\Gamma_1;\ \emptyset;\ \Gamma_2$ has a refutation tree in d-NO.

Proof: Exercise*

The d-NO Calculus and Convex Theories

The d-NO calculus becomes really deterministic when T_1 and T_2 are convex.

Then, every refutation tree consists of a single branch.

The d-NO Calculus and Convex Theories

The d-NO calculus becomes really deterministic when T_1 and T_2 are convex.

Then, every refutation tree consists of a single branch.

A Σ-theory T is convex iff

for all finite sets Γ of Σ-literals and
for all non-empty disjunctions $\bigvee_{i \in I} x_i \approx y_i$ of variables,

$\Gamma \models_T \bigvee_{i \in I} x_i \approx y_i$ iff $\Gamma \models_T x_i \approx y_i$ for some $i \in I$.

The d-NO Calculus and Convex Theories

The d-NO calculus becomes really deterministic when T_1 and T_2 are convex.

Then, every refutation tree consists of a single branch.

A Σ-theory T is convex iff

for all finite sets Γ of Σ-literals and
for all non-empty disjunctions $\bigvee_{i \in I} x_i \approx y_i$ of variables,

$$\Gamma \models_T \bigvee_{i \in I} x_i \approx y_i \text{ iff } \Gamma \models_T x_i \approx y_i \text{ for some } i \in I.$$

Useful fact: Every convex theory T with no trivial models (i.e., such that $T \models \exists x, y. x \not\approx y$) is stably infinite [BDS02b].

The d-NO Calculus and Convex Theories

Many interesting theories are convex (not immediate to show):

- All Horn theories—this includes all (conditional) equational theories.
- Some non-Horn theories, like linear rational arithmetic.

The d-NO Calculus and Convex Theories

Many interesting theories are convex (not immediate to show):

- All Horn theories—this includes all (conditional) equational theories.
- Some non-Horn theories, like linear rational arithmetic.

But many more are not convex:

- All theories of a finite structure. (Why?)
- Non-linear rational arithmetic. (Why?)
- Linear integer arithmetic. (Why?)
- The theory of arrays. (Why?)
- The theory of sets. (Why?)

Extending Nelson-Oppen

The main requirements of the method:

- The disjointness of Σ_1 and Σ_2 and
- the stable infiniteness of T_1 and T_2

are only sufficient conditions for its correctness.

Can they be relaxed?

Extending Nelson-Oppen

The main requirements of the method:

- The disjointness of Σ_1 and Σ_2 and
- the stable infiniteness of T_1 and T_2

are only sufficient conditions for its correctness.

Can they be relaxed?

Relaxing either of them turns out to be rather hard.

Only few results in this direction, all very recent, and perhaps mainly of academic interest for now.

Extending NO: Non-Stably Infinite Theories

The only existing results (we are aware of) are about

- combining arbitrary theories with the theory of equality (aka the empty theory, EUF, . . .) [Gan02],
- about combining arbitrary theories with shiny or polite theories [TZ05, RRZ05]
- combining universal theories [Zar04].

The results in [TZ05, RRZ05] subsume those in [Gan02] but are not comparable to those in [TZ05].

The results in [Zar04] also lift the disjointness restriction.

Extending Nelson-Oppen: Non-Disjoint Theories

Three main approaches, respectively described in: [TR03], [Ghi04], and [Zar04].

All of them need to extend the constraint sharing mechanism beyond (dis)equalities of shared constants.

None of them is more general than the others.

[TR03] and [Ghi04] are rather technical and beyond the scope of this tutorial.

[Zar04] is very general but yields weaker results both in theory (only semi-decidability) and in practice (too much to guess).

Extending Nelson-Oppen: Sorted Logics

Extending the method of many sorted logics (no subsorts) is intuitively simple [TZ04]:.

- Use a notion of stable infiniteness wrt. a set of sorts.
- Require component theories to be stably infinite only wrt. their shared sorts.
- Consider only well-sorted arrangements.

Extending Nelson-Oppen: Sorted Logics

Extending the method of many sorted logics (no subsorts) is intuitively simple [TZ04]:.

- Use a notion of stable infiniteness wrt. a set of sorts.
- Require component theories to be stably infinite only wrt. their shared sorts.
- Consider only well-sorted arrangements.

The advantages of sorts are:

- Combining sorted theories is more natural.
- It is easier for a sorted theory to be stably infinite wrt. just a subset of its sorts.
- No need to propagate equalities between variables of different sorts.

Extending Nelson-Oppen: Sorted Logics

Extending the method of many sorted logics (no subsorts) is intuitively simple [TZ04]:.

- Use a notion of stable infiniteness wrt. a set of sorts.
- Require component theories to be stably infinite only wrt. their shared sorts.
- Consider only well-sorted arrangements.

Extending the method of order-sorted logics (with subsorts) is non-trivial.

Currently, relatively strong restrictions on the component signatures are needed [TZ04].

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

Extending Nelson-Oppen: More than Literals

- The Nelson-Oppen method combines procedures for the satisfiability of sets of ground literals.
- However, actual problems involve, more generally, ground formulas.
- How to combine decision procedures for them?
- Before that, how to extend a decision procedure to ground formulas?

Satisfiability Modulo a Theory T (SMT)

- Observation: T-satisfiability is decidable for ground formulas whenever it is decidable for sets of literals.

  (By converting the formula in DNF.)

- Problem: In practice, dealing with Boolean combinations of literals is as hard as in the propositional case.

- Current solution: Exploit latest advances in propositional satisfiability technology.

  Specifically, use DPLL-based methods.

The Original DPLL Procedure [DLL62]

- Tries to build incrementally a satisfying truth assignment M for a CNF formula F.

The Original DPLL Procedure [DLL62]

- Tries to build incrementally a satisfying truth assignment M for a CNF formula F.
- M is grown by

The Original DPLL Procedure [DLL62]

- Tries to build incrementally a satisfying truth assignment M for a CNF formula F.
- M is grown by
    - deducing the truth value of a literal from M and F, or

The Original DPLL Procedure [DLL62]

- Tries to build incrementally a satisfying truth assignment M for a CNF formula F.
- M is grown by
  - deducing the truth value of a literal from M and F, or
  - guessing a truth value.

The Original DPLL Procedure [DLL62]

- Tries to build incrementally a satisfying truth assignment M for a CNF formula F.
- M is grown by
  - deducing the truth value of a literal from M and F, or
  - guessing a truth value.
- If a wrong guess leads to an inconsistency, the procedure backtracks and tries the opposite one.

The Original DPLL Procedure [DLL62]

- Tries to build incrementally a satisfying truth assignment M for a CNF formula F.
- M is grown by
  - deducing the truth value of a literal from M and F, or
  - guessing a truth value.
- If a wrong guess leads to an inconsistency, the procedure backtracks and tries the opposite one.
- Modern implementations add several sophisticated search techniques.

  (Backjumping, learning, restarts, watched literals, etc.)

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 4	$1,\ 2,\ 3,\ 4$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 4	$1,\ 2,\ 3,\ 4$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

Inconsistency!

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
deduce 4	$1,\ 2,\ 3,\ 4$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$
undo 3	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee \overline{3} \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
deduce 4	$1,\ 2,\ 3,\ 4$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
undo 3	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$

The Original DPLL Procedure – Example

Operation	Assign.	Formula
		$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
deduce 1	1	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
deduce 2	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
deduce 4	$1,\ 2,\ 3,\ 4$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
undo 3	$1,\ 2$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$
guess 3	$1,\ 2,\ 3$	$1 \vee 2,\ 2 \vee \overline{3} \vee 4,\ \overline{1} \vee \overline{2},\ \overline{1} \vee 3 \vee \overline{4},\ 1$

Model Found!

Lifting SAT to SMT

Eager approach [BLS02, SLB03, CKSY04, . . .]:

- translate into an equisatisfiable propositional formula,
- feed it to any SAT solver.

Lifting SAT to SMT

Eager approach [BLS02, SLB03, CKSY04, . . .]:

- translate into an equisatisfiable propositional formula,
- feed it to any SAT solver.

Lazy approach
[ACG00, ABC+02, BDS02a, dMR02, FJOS03, BCLZ04, . . .]:

- abstract the input formula into a propositional one,
- feed it to a DPLL-based SAT solver,
- use a theory decision procedure to refine the formula.

Lifting SAT to SMT

Eager approach [BLS02, SLB03, CKSY04, . . .]:

- translate into an equisatisfiable propositional formula,
- feed it to any SAT solver.

Lazy approach
[ACG00, ABC$^+$02, BDS02a, dMR02, FJOS03, BCLZ04, . . .]:

- abstract the input formula into a propositional one,
- feed it to a DPLL-based SAT solver,
- use a theory decision procedure to refine the formula.

DPLL(T) [Tin02, GHN$^+$04, NO05]:

- use the decision procedure to guide the search of a DPLL solver.

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

An Abstract Framework for DPLL

- The DPLL procedure can be described declaratively by simple sequent-style calculi [Tin02, BT03].
- Such calculi, however, cannot model meta-logical features such as backtracking, learning and restarts.
- One can better model DPLL and its enhancements as transition systems [NOT05].
- A transition system is a binary relation over states, induced by a set of conditional transition rules.

An Abstract Framework for DPLL [NOT05]

States:

$$fail \quad \text{or} \quad M \parallel F$$

where F is a CNF formula, a set of clauses, and
M is a sequence of annotated literals
denoting a partial truth assignment.

An Abstract Framework for DPLL [NOT05]

States:

$$fail \quad \text{or} \quad M \parallel F$$

Initial state:

- $\emptyset \parallel F$, where F is to be checked for satisfiability.

Expected final states:

- $fail$, if F is unsatisfiable
- $M \parallel G$, where M is a model of G and
  G is logically equivalent to F.

Transition Rules for Basic DPLL

Extending the assignment:

UnitProp

$$M \parallel F, C \vee l \quad \longrightarrow \quad M\, l \parallel F, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases}$$

Transition Rules for Basic DPLL

Extending the assignment:

UnitProp

$$M \parallel F, C \vee l \quad \longrightarrow \quad M\, l \parallel F, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases}$$

Decide

$$M \parallel F \quad \longrightarrow \quad M\, l^{\mathsf{d}} \parallel F \quad \textbf{if} \begin{cases} l \text{ or } \overline{l} \text{ occurs in } F, \\ l \text{ is undefined in } M \end{cases}$$

Notation: l^{d} annotates l as a decision literal.

Transition Rules for Basic DPLL

Repairing the assignment:

Fail

$$M \parallel F, C \quad \longrightarrow \quad \mathit{fail} \quad \textbf{if} \begin{cases} M \models \neg C, \\ M \text{ contains no decision literals} \end{cases}$$

Transition Rules for Basic DPLL

Repairing the assignment:

Backjump

$$M\, l^{\mathsf{d}}\, N \parallel F, C \quad \longrightarrow \quad M\, k \parallel F, C \quad \textbf{if} \begin{cases} 1.\ M\, l^{\mathsf{d}}\, N \models \neg C, \\ 2.\ \text{for some } D \vee k\text{:} \\ \quad F, C \models D \vee k, \\ \quad M \models \neg D, \\ \quad k \text{ is undefined in } M, \\ \quad k \text{ or } \overline{k} \text{ occurs in} \\ \quad M\, l^{\mathsf{d}}\, N \parallel F, C \end{cases}$$

Basic DPLL System

At the core, current DPLL-based SAT solvers are implementations of the transition system:

Basic DPLL

- UnitProp
- Decide
- Fail
- Backjump

Basic DPLL System – Example

$$\emptyset \quad \| \quad \overline{1} \vee 2,\;\; \overline{3} \vee 4,\;\; \overline{5} \vee \overline{6},\;\; 6 \vee \overline{5} \vee \overline{2}$$

Basic DPLL System – Example

$$\emptyset \parallel \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \implies (\text{Decide})$$
$$1 \parallel \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Basic DPLL System – Example

$$
\begin{array}{rcll}
\emptyset & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} &
\end{array}
$$

Basic DPLL System – Example

$$
\begin{array}{rcll}
\emptyset & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} &
\end{array}
$$

Basic DPLL System – Example

$$
\begin{array}{rcll}
\emptyset & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \quad (\text{Decide}) \\
1 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \quad (\text{UnitProp}) \\
1\ 2 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \quad (\text{Decide}) \\
1\ 2\ 3 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \quad (\text{UnitProp}) \\
1\ 2\ 3\ 4 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} &
\end{array}
$$

Basic DPLL System – Example

$$
\begin{array}{rcll}
\emptyset & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1 & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2 & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3 & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ 4 & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3\ 4\ 5 & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} &
\end{array}
$$

Basic DPLL System – Example

$$
\begin{array}{rcll}
\emptyset & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ 4 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3\ 4\ 5 & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ 4\ 5\ \overline{6} & \| & \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} &
\end{array}
$$

Basic DPLL System – Example

$$
\begin{array}{rcll}
\emptyset & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ 4 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Decide}) \\
1\ 2\ 3\ 4\ 5 & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{UnitProp}) \\
1\ 2\ 3\ 4\ 5\ \overline{6} & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Backjump}) \\
1\ 2\ \overline{5} & \| & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} &
\end{array}
$$

Backjump with clause $\overline{1} \vee \overline{5}$

Basic DPLL System – Example

$$\ldots$$

$$1\ 2\ 3\ 4\ 5\ \overline{6}\ \|\ \ \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2} \quad \Longrightarrow \quad (\text{Backjump})$$

$$1\ 2\ \overline{5}\ \|\ \ \overline{1} \lor 2,\ \ \overline{3} \lor 4,\ \ \overline{5} \lor \overline{6},\ \ 6 \lor \overline{5} \lor \overline{2}.$$

Backjump with clause $\overline{1} \lor 5$

$$\left\{\begin{array}{l} M\ l^{\mathsf{d}}\ N \models \neg C, \\ \text{for some clause } D \lor k\text{:} \\ \quad F \models D \lor k, \\ \quad M \models \neg D, \\ \quad k \text{ is undefined in } M \\ \quad k \text{ or } \overline{k} \text{ occurs in } F \end{array}\right.$$

$$\left\{\begin{array}{l} 1\ 2\ 3\ 4\ 5\ \overline{6} \models \neg(6 \lor \overline{5} \lor \overline{2}), \\ \text{for clause } \overline{1} \lor \overline{5}\text{:} \\ \quad F \models \overline{1} \lor \overline{5}, \\ \quad 1\ 2 \models 1, \\ \quad \overline{5} \text{ is undefined in } 1\ 2 \\ \quad \overline{5} \text{ occurs in } F \end{array}\right.$$

Basic DPLL System – Example

$$\ldots$$

$$1\ 2\ 3\ 4\ 5\ \overline{6} \ \| \quad \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} \quad \Longrightarrow \quad (\text{Backjump})$$

$$1\ 2\ \overline{5} \ \| \quad \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2}.$$

Indeed, $F \models \overline{1} \vee \overline{5}$. For instance, by resolution,

$$\dfrac{\dfrac{\overline{1} \vee 2 \qquad 6 \vee \overline{5} \vee \overline{2}}{\overline{1} \vee 6 \vee \overline{5}} \qquad \overline{5} \vee \overline{6}}{\overline{1} \vee \overline{5}}$$

Therefore, instead deciding 3, we could have deduced $\overline{5}$.

Basic DPLL System – Example

$$\begin{array}{rcll} & \ldots & & \\ 1\ 2\ 3\ 4\ 5\ \overline{6} & \| & \overline{1} \vee 2,\quad \overline{3} \vee 4,\quad \overline{5} \vee \overline{6},\quad 6 \vee \overline{5} \vee \overline{2} & \Longrightarrow \ (\text{Backjump}) \\ 1\ 2\ \overline{5} & \| & \overline{1} \vee 2,\quad \overline{3} \vee 4,\quad \overline{5} \vee \overline{6},\quad 6 \vee \overline{5} \vee \overline{2}. & \end{array}$$

Indeed, $F \models \overline{1} \vee \overline{5}$. For instance, by resolution,

$$\dfrac{\dfrac{\overline{1} \vee 2 \qquad 6 \vee \overline{5} \vee \overline{2}}{\overline{1} \vee 6 \vee \overline{5}} \qquad \overline{5} \vee \overline{6}}{\overline{1} \vee \overline{5}}$$

Therefore, instead deciding 3, we could have deduced $\overline{5}$.

Clauses like $\overline{1} \vee \overline{5}$ are computed by navigating conflict graphs.

The Basic DPLL System – Correctness

Some terminology

Irreducible state: state to which no transition rule applies.

Execution: sequence of transitions allowed by the rules and starting with states of the form $\emptyset \parallel F$.

Exhausted execution: execution ending in an irreducible state.

The Basic DPLL System – Correctness

Some terminology

Irreducible state: state to which no transition rule applies.

Execution: sequence of transitions allowed by the rules and starting with states of the form $\emptyset \parallel F$.

Exhausted execution: execution ending in an irreducible state.

Proposition (Strong Termination) Every execution in Basic DPLL is finite.

Note: This is not so immediate, because of Backjump.

The Basic DPLL System – Correctness

Some terminology

Irreducible state: state to which no transition rule applies.

Execution: sequence of transitions allowed by the rules and starting with states of the form $\emptyset \parallel F$.

Exhausted execution: execution ending in an irreducible state.

Proposition (Soundness) For every exhausted execution starting with $\emptyset \parallel F$ and ending in $M \parallel F$, $M \models F$.

Proposition (Completeness) If F is unsatisfiable, every exhausted execution starting with $\emptyset \parallel F$ ends with $fail$.

The Basic DPLL System – Correctness Proofs

The termination argument is based on the fact that each rule produces a smaller (i.e. more determined) state.

The Basic DPLL System – Correctness Proofs

The termination argument is based on the fact that each rule produces a smaller (i.e. more determined) state.

The soundness and completeness arguments are based on the following invariants.

Proposition If $M \parallel G$ is reachable from $\emptyset \parallel F$ then

1. All atoms in M and all atoms in G are in F.
2. M is a (partial) truth assignment.
3. G is logically equivalent to F
4. If $M = M_0\, l_1^{\mathsf{d}}\, M_1\, \cdots\, l_n^{\mathsf{d}}\, M_n$, then $F \cup \{l_1, \ldots, l_i\} \models M_i$ for $i = 0, \ldots, n$.

Enhancements to Basic DPLL

Learn

$$M \parallel F \quad \longrightarrow \quad M \parallel F, C \quad \textbf{if} \begin{cases} \text{all atoms of } C \text{ occur in } F, \\ F \models C \end{cases}$$

Forget

$$M \parallel F, C \quad \longrightarrow \quad M \parallel F \quad \textbf{if } F \models C$$

Restart

$$M \parallel F \quad \longrightarrow \quad \emptyset \parallel F \quad \textbf{if } \ldots \textit{you want to}$$

Enhancements to Basic DPLL

Learn

$$M \parallel F \quad \longrightarrow \quad M \parallel F, C \quad \textbf{if} \begin{cases} \text{all atoms of } C \text{ occur in } F, \\ F \models C \end{cases}$$

Forget

$$M \parallel F, C \quad \longrightarrow \quad M \parallel F \quad \textbf{if } F \models C$$

Restart

$$M \parallel F \quad \longrightarrow \quad \emptyset \parallel F \quad \textbf{if} \ldots \textit{you want to}$$

We will ignore these enhancements here for simplicity.

Roadmap

- Introduction to First-order Logic with Equality
- The Combined Validity Problem in FOL
- The Combined Satisfiability Problem
- The Combination Problem for Universal Formulas
- The Nelson-Oppen method
- From Literals to Clauses
- An Abstract DPLL Framework for SAT
- Extensions to Satisfiability Modulo Theories

From SAT to SMT — A (Very) Lazy Approach

$$g(a) = c \quad \wedge \quad f(g(a)) \neq f(c) \; \vee \; g(a) = d \quad \wedge \quad c \neq d$$

Theory: Equality

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds $\{1,\ \overline{2}\}$ E-unsatisfiable.

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds $\{1,\ \overline{2}\}$ E-unsatisfiable.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2\}$ to SAT solver.

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds $\{1,\ \overline{2}\}$ E-unsatisfiable.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2\}$ to SAT solver.
- SAT solver returns model $\{1,\ 2,\ 3,\ \overline{4}\}$.
  Theory solver finds $\{1,\ 3,\ \overline{4}\}$ E-unsatisfiable.

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds $\{1,\ \overline{2}\}$ E-unsatisfiable.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2\}$ to SAT solver.
- SAT solver returns model $\{1,\ 2,\ 3,\ \overline{4}\}$.
  Theory solver finds $\{1,\ 3,\ \overline{4}\}$ E-unsatisfiable.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4\}$ to SAT solver.

From SAT to SMT — A (Very) Lazy Approach

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

- Send $\{1,\ \overline{2} \vee 3,\ \overline{4}\}$ to SAT solver.
- SAT solver returns model $\{1,\ \overline{2},\ \overline{4}\}$.
  Theory solver finds $\{1,\ \overline{2}\}$ E-unsatisfiable.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2\}$ to SAT solver.
- SAT solver returns model $\{1,\ 2,\ 3,\ \overline{4}\}$.
  Theory solver finds $\{1,\ 3,\ \overline{4}\}$ E-unsatisfiable.
- Send $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4\}$ to SAT solver.
- SAT solver finds $\{1,\ \overline{2} \vee 3,\ \overline{4},\ \overline{1} \vee 2,\ \overline{1} \vee \overline{3} \vee 4\}$ unsatisfiable.

Modeling the Lazy Approach

Let T be the background theory.

The previous process can be modeled in Abstract DPLL using the following rules:

- UnitProp, Decide, Fail, Restart

  (as in the propositional case) and

- T-Backjump, Very Lazy Theory Learning

Note: The first component of a state $M \parallel F$ is still a truth assignment, but now for ground, first-order literals.

Modeling the Lazy Approach

T-Backjump

$$M\, l^{\mathsf{d}}\, N \parallel F, C \;\;\longrightarrow\;\; M\, k \parallel F, C \quad \textbf{if} \begin{cases} 1.\ M\, l^{\mathsf{d}}\, N \models \neg C, \\ 2.\ \text{for some } D \vee k\text{:} \\ \quad F, C \models_T D \vee k, \\ \quad M \models \neg D, \\ \quad k \text{ is undefined in } M, \\ \quad k \text{ or } \overline{k} \text{ occurs in} \\ \quad M\, l^{\mathsf{d}}\, N \parallel F, C \end{cases}$$

Only change: $\models_T$ instead of $\models$

Notation: $F \models_T G$ iff $T, F \models G$

Modeling the Lazy Approach

The interaction between theory solver and SAT solver in the previous example can be modeled with the rule

Very Lazy Theory Learning

$$M \parallel F \quad \longrightarrow \quad \emptyset \parallel F, \overline{l_1} \vee \ldots \vee \overline{l_n} \quad \textbf{if} \begin{cases} M \models F \\ \{l_1, \ldots, l_n\} \subseteq M \\ l_1 \wedge \cdots \wedge l_n \models_T \bot \end{cases}$$

Modeling the Lazy Approach

The interaction between theory solver and SAT solver in the previous example can be modeled with the rule

Very Lazy Theory Learning

$$M \parallel F \quad \longrightarrow \quad \emptyset \parallel F, \overline{l_1} \vee \ldots \vee \overline{l_n} \quad \textbf{if} \begin{cases} M \models F \\ \{l_1, \ldots, l_n\} \subseteq M \\ l_1 \wedge \cdots \wedge l_n \models_T \perp \end{cases}$$

A better approach is to detect partial assignments that are already T-unsatisfiable.

Modeling the Lazy Approach

Lazy Theory Learning

$$M \parallel F \;\; \longrightarrow \;\; M \parallel F, \overline{l_1} \vee \ldots \vee \overline{l_n} \quad \textbf{if} \; \begin{cases} \{l_1, \ldots, l_n\} \subseteq M \\ l_1 \wedge \cdots \wedge l_n \models_T \perp \\ \overline{l}_1 \vee \cdots \vee \overline{l}_n \notin F \end{cases}$$

Modeling the Lazy Approach

Lazy Theory Learning

$$M \parallel F \quad \longrightarrow \quad M \parallel F, \overline{l_1} \vee \ldots \vee \overline{l_n} \quad \textbf{if} \begin{cases} \{l_1, \ldots, l_n\} \subseteq M \\ l_1 \wedge \cdots \wedge l_n \models_T \bot \\ \bar{l}_1 \vee \cdots \vee \bar{l}_n \notin F \end{cases}$$

- The learned clause is false in M, hence either Backjump or Fail applies.
- If this is always done, the third condition of the rule is unnecessary
- In some solvers, the rule is applied as soon as possible, i.e., with $M = N\, l_n$.

Lazy Approach – Strategies

A common strategy is to apply the rules using the following priorities:

1. If a current clause is falsified by the current assignment, apply Fail/Backjump.
2. If the assignment is T-unsatisfiable, apply Lazy Theory Learning + Fail/Backjump.
3. Apply UnitProp.
4. Apply Decide.

DPLL(T) – Eager Theory Propagation

Use the theory information as soon as possible by eagerly applying

Theory Propagate

$$M \parallel F \quad \longrightarrow \quad M\, l \parallel F \quad \textbf{if} \begin{cases} M \models_T l \\ l \text{ or } \bar{l} \text{ occurs in } F \\ l \text{ is undefined in } M \end{cases}$$

Note: Test $M \models_T l$ provided by decision procedure (as $M \models_T l$ iff $M\,\bar{l} \models_T \bot$).

Eager Theory Propagation - Example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\emptyset \;\; \| \quad 1,\; \overline{2} \vee 3,\; \overline{4}$$

Eager Theory Propagation - Example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\begin{array}{lclcl} \emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\textsf{UnitProp}) \\ 1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & & \end{array}$$

Eager Theory Propagation - Example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\begin{array}{rcl}
\emptyset \;\|\; 1,\; \overline{2} \vee 3,\; \overline{4} & \Longrightarrow & (\text{UnitProp}) \\
1 \;\|\; 1,\; \overline{2} \vee 3,\; \overline{4} & \Longrightarrow & (\text{Theory Propagate}) \\
1\;2 \;\|\; 1,\; \overline{2} \vee 3,\; \overline{4} & &
\end{array}$$

Eager Theory Propagation - Example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\begin{array}{rcllcl}
\emptyset & \| & 1, & \overline{2} \vee 3, & \overline{4} & \Longrightarrow \quad (\text{UnitProp}) \\
1 & \| & 1, & \overline{2} \vee 3, & \overline{4} & \Longrightarrow \quad (\text{Theory Propagate}) \\
1\ 2 & \| & 1, & \overline{2} \vee 3, & \overline{4} & \Longrightarrow \quad (\text{UnitProp}) \\
1\ 2\ 3 & \| & 1, & \overline{2} \vee 3, & \overline{4} &
\end{array}$$

Eager Theory Propagation - Example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\begin{array}{rcllcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{Theory Propagate}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{UnitProp}) \\
1\ 2\ 3 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{Theory Propagate}) \\
1\ 2\ 3\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & &
\end{array}$$

Eager Theory Propagation - Example

$$\underbrace{g(a) = c}_{1} \quad \wedge \quad \underbrace{f(g(a)) \neq f(c)}_{\overline{2}} \vee \underbrace{g(a) = d}_{3} \quad \wedge \quad \underbrace{c \neq d}_{\overline{4}}$$

$$\begin{array}{rclcl}
\emptyset & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{UnitProp}) \\
1 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{Theory Propagate}) \\
1\ 2 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{UnitProp}) \\
1\ 2\ 3 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{Theory Propagate}) \\
1\ 2\ 3\ 4 & \| & 1,\ \overline{2} \vee 3,\ \overline{4} & \Longrightarrow & (\text{Fail}) \\
 & fail & & &
\end{array}$$

Eager Theory Propagation

- By eagerly applying Theory Propagate every assignment is T-satisfiable, since $M\,l$ is T-unsatisfiable iff $M \models_T \bar{l}$.
- As a consequence, Lazy Theory Learning never applies.
- For some logics, e.g., difference logic, his approach is extremely effective.
- For some others, e.g., the theory of equality, it is too expensive to detect all T-consequences.
- If Theory Propagate is not applied eagerly, Lazy Theory Learning is needed to repair T-unsatisfiable assignments.

Lazy Theory Propagation

- Assume a decision procedure P for the T-satisfiability of sets of ground literals.
- The 4 rules of the DPLL system + Lazy Theory Learning + Theory Propagate + P provide a decision procedure for the T-satisfiability of sets of ground clauses.
- Termination can be guaranteed by applying Fail/Backjump immediately after Lazy Theory Learning.
- Soundness and completeness are proved similarly to the propositional case.
- Arbitrary ground formulas can be dealt as usual by a preliminary CNF translation.

Abstract DPLL Modulo Multiple Theories

Let $T_1, \ldots, T_n$ be distinct theories with respective decision procedures $P_1, \ldots, P_n$.

How can we reason over all of them with Abstract DPLL?

Abstract DPLL Modulo Multiple Theories

Let $T_1, \ldots, T_n$ be distinct theories with respective decision procedures $P_1, \ldots, P_n$.

How can we reason over all of them with Abstract DPLL?

Quick Solution:

1. Combine $P_1, \ldots, P_n$ with Nelson-Oppen into a decision procedure for $T_1 \cup \cdots \cup T_n$.
2. Use Abstract DPLL with $T = T_1 \cup \cdots \cup T_n$.

Abstract DPLL Modulo Multiple Theories

Let $T_1, \ldots, T_n$ be distinct theories with respective decision procedures $P_1, \ldots, P_n$.

How can we reason over all of them with Abstract DPLL?

Better Solution [Bar02, Tin04, BBC$^+$05]:

1. Lift Nelson-Oppen to the DPLL level.
2. Use Abstract DPLL with multiple theories.

Abstract DPLL Modulo Multiple Theories

Preliminaries

- Let $n = 2$, for simplicity.
- Let T_i be of signature Σ_i for $i = 1, 2$, with $\Sigma_1 \cap \Sigma_2 = \emptyset$.
- Let C be a set of free constants.
- Assume wlog that each input literal has signature $\Sigma_1(C)$ or $\Sigma_2(C)$ (no mixed literals).
- Let $M^i = \{\Sigma_i(C)$-literals of $M\}$.
- Let $se(M) = \{c \approx d \mid c, d$ occur in C, M^1 and $M^2\}$ (*shared equalities*).

Abstract DPLL – Rules for Multiple Theories

UnitProp (unchanged)

Fail (unchanged)

T-Backjump (unchanged, with $T = T_1 \cup T_2$)

Decide

$$M \parallel F \quad \longrightarrow \quad M\, l^{\mathsf{d}} \parallel F \quad \textbf{if} \begin{cases} l \text{ or } \bar{l} \text{ occurs in } F \text{ or in } se(M), \\ l \text{ is undefined in } M \end{cases}$$

Only change: decide on (undefined) shared equalities as well.

Abstract DPLL – Rules for Multiple Theories

Lazy Theory Learning

$$M \parallel F \quad \to \quad M \parallel F, \overline{l_1} \vee \ldots \vee \overline{l_n} \quad \textbf{if} \begin{cases} i \in \{1, 2\} \\ \{l_1, \ldots, l_n\} \subseteq M^i \\ l_1 \wedge \cdots \wedge l_n \models_{T_i} \bot \\ \bar{l}_1 \vee \cdots \vee \bar{l}_n \notin F \end{cases}$$

Theory Propagate

$$M \parallel F \quad \to \quad M\, l \parallel F \quad \textbf{if} \begin{cases} i \in \{1, 2\} \\ M^i \models_{T_i} l \\ l \text{ or } \bar{l} \text{ occurs in } F \cup se(M) \\ l \text{ is undefined in } M \end{cases}$$

Changes: (i) reason locally in T_i, (ii) theory propagate shared equalities as well.

References

[ABC+02] Gilles Audemard, Piergiorgio Bertoli, Alessandro Cimatti, Artur Korniłowicz, and Roberto Sebastiani. A SAT-based approach for solving formulas over boolean and linear mathematical propositions. In Andrei Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*, pages 195–210. Springer, 2002

[ACG00] Alessandro Armando, Claudio Castellini, and Enrico Giunchiglia. SAT-based procedures for temporal reasoning. In S. Biundo and M. Fox, editors, *Proceedings of the 5th European Conference on Planning (Durham, UK)*, volume 1809 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2000

[Bar02] Clark W. Barrett. *Checking Validity of Quantifier-Free Formulas in Combinations of First-Order Theories*. PhD dissertation, Department of Computer Science, Stanford University, Stanford, CA, Sep 2002

[BBC+05] Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi Junttila, Silvio Ranise, Roberto Sebastiani, and Peter van Rossu. Efficient satisfiability modulo theories via delayed theory combination. In K.Etessami and S. Rajamani, editors, *Proceedings of the 17th International Conference on Computer Aided Verification*, Lecture Notes in Computer Science. Springer, 2005. (To appear)

References

[BCLZ04] Thomas Ball, Byron Cook, Shuvendu K. Lahiri, and Lintao Zhang. Zapato: Automatic theorem proving for predicate abstraction refinement. In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 457–461. Springer, 2004

[BDS02a] Clark W. Barrett, David L. Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In J. C. Godskesen, editor, *Proceedings of the International Conference on Computer-Aided Verification*, Lecture Notes in Computer Science, 2002

[BDS02b] Clark W. Barrett, David L. Dill, and Aaron Stump. A generalization of Shostak's method for combining decision procedures. In A. Armando, editor, *Proceedings of the 4th International Workshop on Frontiers of Combining Systems, FroCoS'2002 (Santa Margherita Ligure, Italy)*, volume 2309 of *Lecture Notes in Computer Science*, pages 132–147, apr 2002

[BLS02] Randal E. Bryant, Shuvendu K. Lahiri, and Sanjit A. Seshia. Deciding CLU logic formulas via boolean and pseudo-boolean encodings. In *Proc. Intl. Workshop on Constraints in Formal Verification*, 2002

References

[BT02] Franz Baader and Cesare Tinelli. Deciding the word problem in the union of equational theories. *Information and Computation*, 178(2):346–390, December 2002

[BT03] Peter Baumgartner and Cesare Tinelli. The model evolution calculus. In F. Baader, editor, *Proceedings of the 19th International Conference on Automated Deduction, CADE-19 (Miami, Florida, USA)*, number 2741 in Lecture Notes in Artificial Intelligence, pages 350–364. Springer, 2003

[CKSY04] Edmund Clarke, Daniel Kroening, Natasha Sharygina, and Karen Yorav. Predicate abstraction of ANSI–C programs using SAT. *Formal Methods in System Design (FMSD)*, 25:105–127, September–November 2004

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962

[dMR02] Leonardo de Moura and Harald Rueß. Lemmas on demand for satisfiability solvers. In *Proc. of the Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT'02)*, May 2002

References

[FJOS03] Cormac Flanagan, Rajeev Joshi, Xinming Ou, and James B. Saxe. Theorem proving using lazy proof explication. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2003

[Gan02] Harald Ganzinger. Shostak light. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Computer Science*, pages 332–346. Springer-Verlag, jul 2002

[Ghi04] Silvio Ghilardi. Model theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 3(3–4):221–249, 2004

[GHN+04] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. DPLL(T): Fast decision procedures. In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification, CAV'04 (Boston, Massachusetts)*, volume 3114 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2004

[NO79] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming Languages and Systems*, 1(2):245–257, October 1979

References

[NO05] Robert Nieuwenhuis and Albert Oliveras. DPLL(T) with exhaustive theory propagation and its application to difference logic. In K. Etessami and S. Rajamani, editors, *Proceedings of 17th International Conference on Computer Aided Verification*, Lecture Notes in Computer Science. Springer, 2005. (To appear)

[NOT05] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Abstract DPLL and abstract DPLL modulo theories. In F. Baader and A. Voronkov, editors, *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'04), Montevideo, Uruguay*, volume 3452 of *Lecture Notes in Artificial Intelligence*, pages 36–50. Springer, 2005

[Opp80] Derek C. Oppen. Complexity, convexity and combinations of theories. *Theoretical Computer Science*, 12:291–302, 1980

[Rin96] Christophe Ringeissen. Cooperation of decision procedures for the satisfiability problem. In F. Baader and K.U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop, Munich (Germany)*, Applied Logic, pages 121–140. Kluwer Academic Publishers, March 1996

References

[RRZ05] Silvio Ranise, Christophe Ringeissen, and Calogero G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In B. Gramlich, editor, *Proceedings of the Workshop on Frontiers of Combining Systems*, Lecture Notes in Computer Science. Springer, 2005. (To appear.)

[TH96] Cesare Tinelli and Mehdi T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K.U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop (Munich, Germany)*, Applied Logic, pages 103–120. Kluwer Academic Publishers, March 1996

[SLB03] Sanjit A. Seshia, Shuvendu K. Lahiri, and Randal E. Bryant. A hybrid SAT-based decision procedure for separation logic with uninterpreted functions. In *Proc. 40th Design Automation Conference*, pages 425–430. ACM Press, 2003

[Tin02] Cesare Tinelli. A DPLL-based calculus for ground satisfiability modulo theories. In Giovambattista Ianni and Sergio Flesca, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (Cosenza, Italy)*, volume 2424 of *Lecture Notes in Artificial Intelligence*. Springer, 2002

References

[TH96] Cesare Tinelli and Mehdi T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K.U. Schulz, editors, *Frontiers of Combining Systems: Proceedings of the 1st International Workshop (Munich, Germany)*, Applied Logic, pages 103–120. Kluwer Academic Publishers, March 1996

[Tin02] Cesare Tinelli. A DPLL-based calculus for ground satisfiability modulo theories. In Giovambattista Ianni and Sergio Flesca, editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (Cosenza, Italy)*, volume 2424 of *Lecture Notes in Artificial Intelligence*. Springer, 2002

[Tin04] Cesare Tinelli. The DPLL($T_1, \ldots, T_n$): modeling DPLL-based checkers for satisfiability modulo multiple theories. (Unpublished), 2004

[TR03] Cesare Tinelli and Christophe Ringeissen. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theoretical Computer Science*, 290(1):291–353, January 2003

[TZ04] Cesare Tinelli and Calogero Zarba. Combining decision procedures for sorted theories. In J. Alferes and J. Leite, editors, *Proceedings of the 9th European Conference on Logic in Artificial Intelligence (JELIA'04), Lisbon, Portugal*, volume 3229 of *Lecture Notes in Artificial Intelligence*, pages 641–653. Springer, 2004

References

[TZ05] Cesare Tinelli and Calogero Zarba. Combining non-stably infinite theories. *Journal of Automated Reasoning*, 2005. (To appear.)

[Zar04] Calogero G. Zarba. C-tableaux. Technical Report RR-5229, INRIA, 2004