CS228 : Logic for Computer Science

Module 3: Predicate Logic

Lecture 18: DPLL(T), SMT, and Moving on to Hoare Logic

Instructor: S. Akshay

IIT Bombay, India

Topics Covered till date

Module 1: Propositional Logic

1. Syntax
2. Semantics
3. Formal proof systems
4. Normal Forms

Module 2: The Problem of Satisfiability

1. Propositional Resolution
2. DPLL
3. CDCL
4. SAT/SMT tool demo

Topics Covered till date

Module 1: Propositional Logic

1. Syntax
2. Semantics
3. Formal proof systems
4. Normal Forms

Module 2: The Problem of Satisfiability

1. Propositional Resolution
2. DPLL
3. CDCL
4. SAT/SMT tool demo

Module 3: Predicate Logic

1. Syntax of predicate logic
2. Semantics of predicate logic
3. Natural deduction for FOL
4. Structures, expressiveness, and limits of FOL
5. Normal forms for FOL
6. Today: DPLL(T) and SMT solving

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

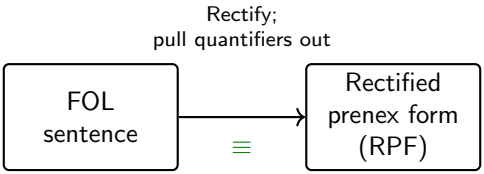
Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



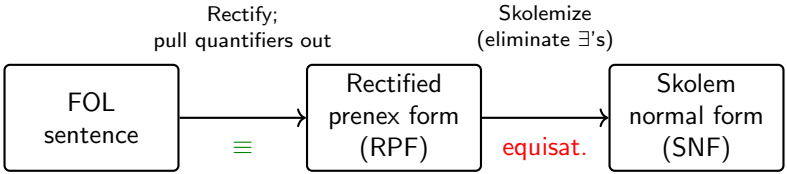
Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



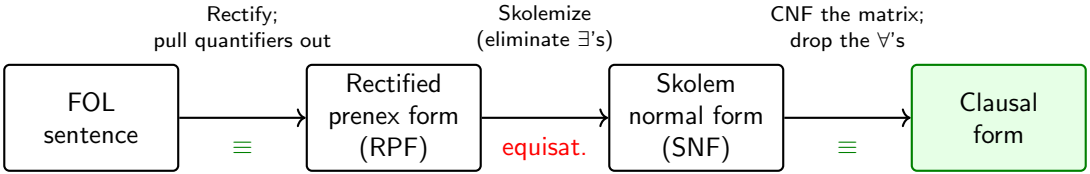
Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



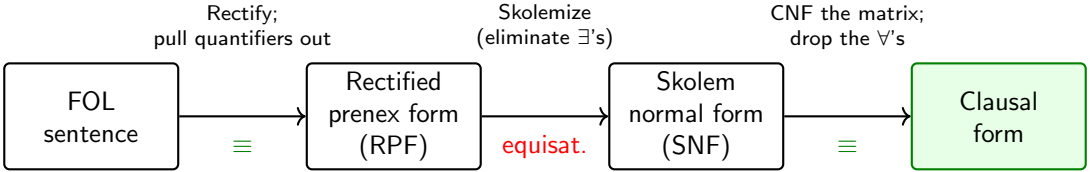
Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



Quick Recap: Normal Forms So Far

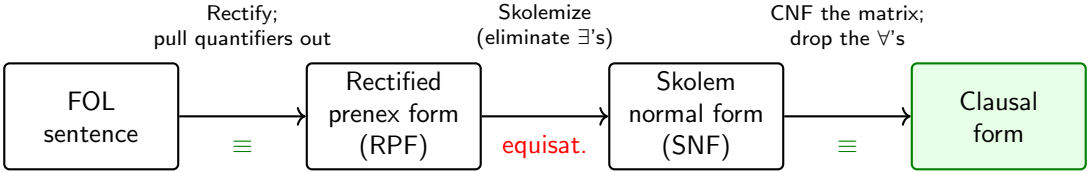
- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



Consider: every non-negative number has a non-negative half.

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

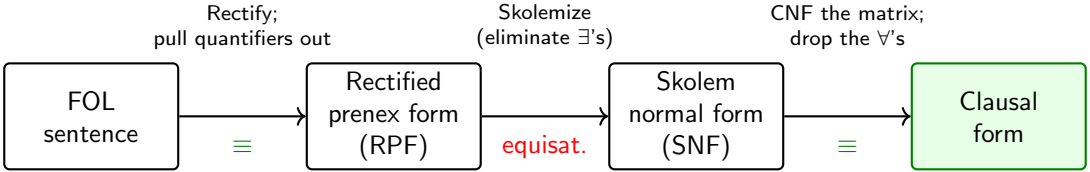


Consider: every non-negative number has a non-negative half.

1. Write this in FOL

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



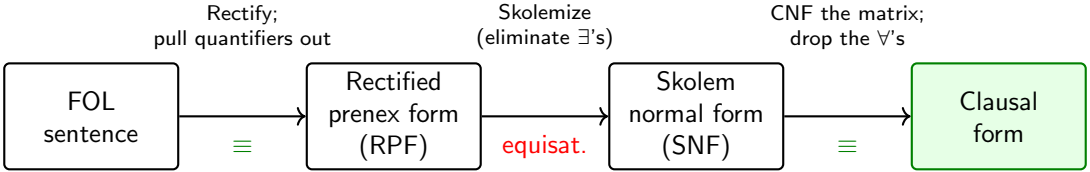
Consider: every non-negative number has a non-negative half.

1. Write this in FOL

$$\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



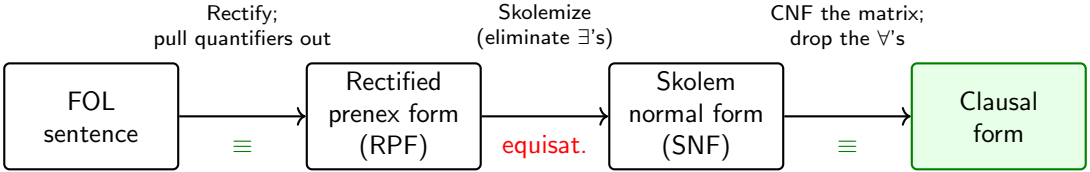
Consider: every non-negative number has a non-negative half.

1. Write this in FOL
2. Write the RPF (one alternation)

$$\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

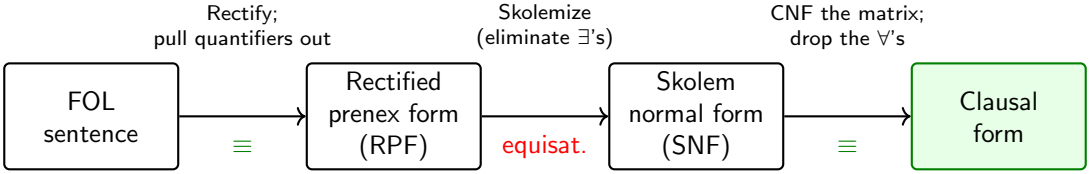


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\quad \varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\quad \forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



Consider: every non-negative number has a non-negative half.

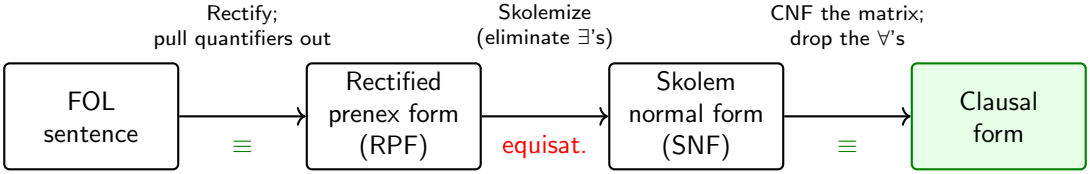
1. Write this in FOL
2. Write the RPF (one alternation)
3. Write the SNF (f is *new*)

$$\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$$

$$\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

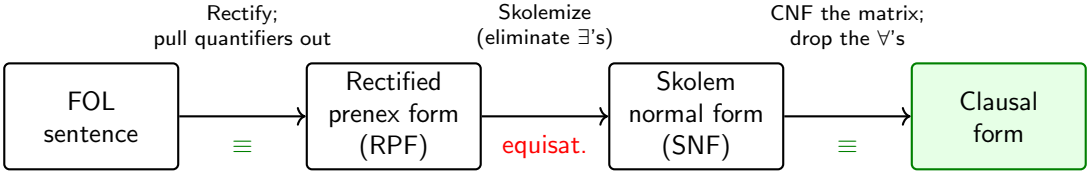


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

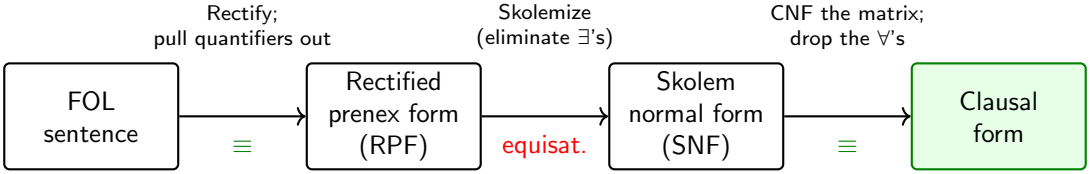


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

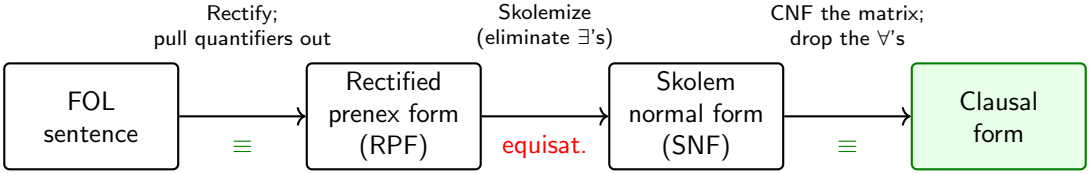


Consider: every non-negative number has a non-negative half.

1. Write this in FOL — $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) — $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) — $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form — $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

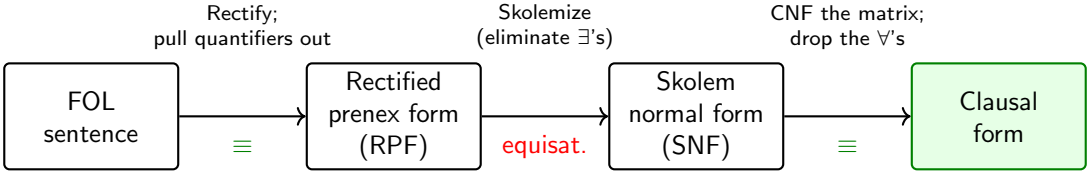


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$.

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

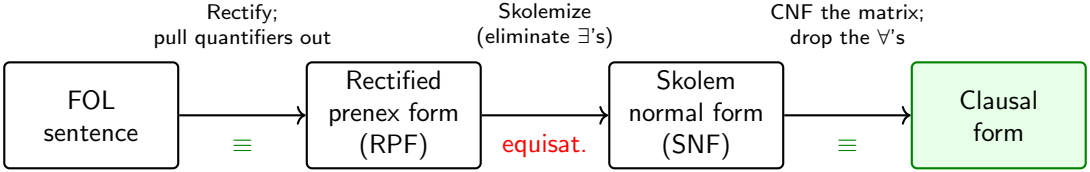


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

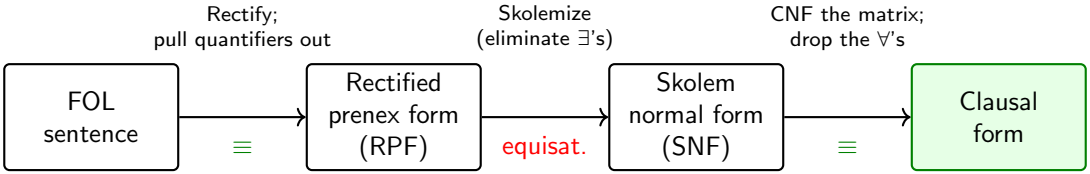


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \to \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$
   If $x < 0$: $\neg(x \geq 0)$ holds, so both do. If $x \geq 0$: we need $f(x) \geq 0$ and $f(x) + f(x) = x$ – both true.

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

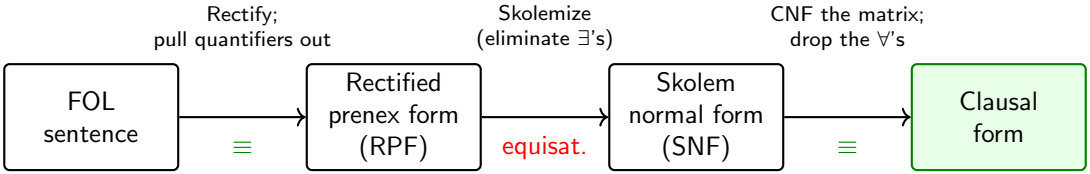


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$
   What if the structure is $(\mathbb{N}, 0, +, \geq, =)$?

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

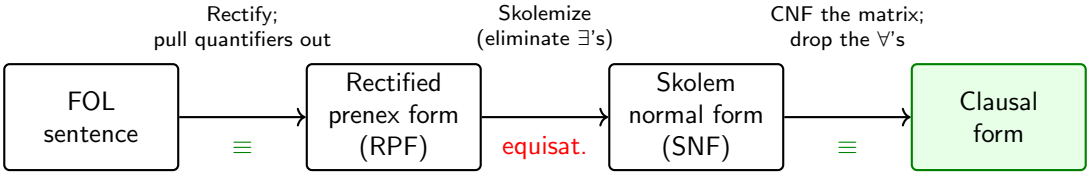


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$
   What if the structure is $(\mathbb{N}, 0, +, \geq, =)$? $f(x) := x/2$ doesn't work: not well-defined for odd x.

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

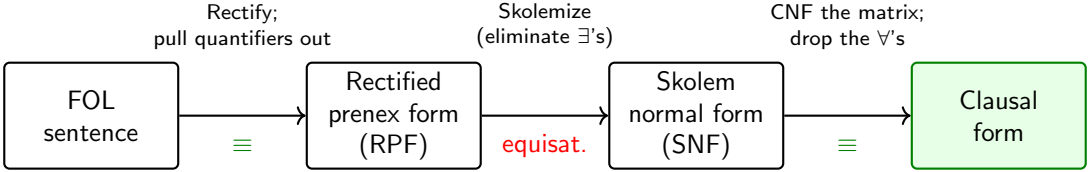


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$
   What if the structure is $(\mathbb{N}, 0, +, \geq, =)$? Any other f? For $x = 1$ we must have $f(1) + f(1) = 1$,

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.

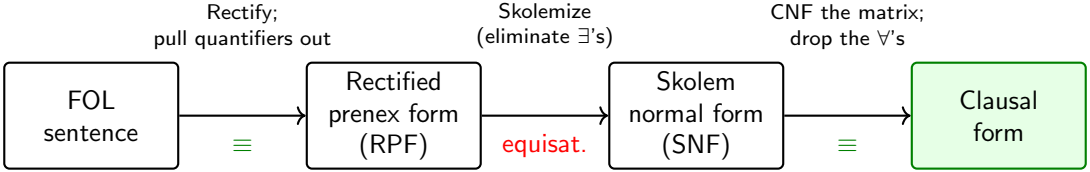


Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \to \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$
   What if the structure is $(\mathbb{N}, 0, +, \geq, =)$? Any other f? For $x = 1$ we must have $f(1) + f(1) = 1$, but $2f(1)$ is even, so *no* f works.

Quick Recap: Normal Forms So Far

- Convert FOL formulas to equisatisfiable clausal form, ready for automated reasoning.



Consider: every non-negative number has a non-negative half.

1. Write this in FOL $\varphi := \forall x \left(x \geq 0 \rightarrow \exists y \left(y \geq 0 \wedge y + y = x\right)\right)$
2. Write the RPF (one alternation) $\forall x \, \exists y \left(\neg(x \geq 0) \vee \left(y \geq 0 \wedge y + y = x\right)\right)$
3. Write the SNF (f is *new*) $\sigma := \forall x \left(\neg(x \geq 0) \vee \left(f(x) \geq 0 \wedge f(x) + f(x) = x\right)\right)$
4. Write the clausal form $\{\, \neg(x \geq 0) \vee f(x) \geq 0, \quad \neg(x \geq 0) \vee f(x) + f(x) = x \,\}$
5. What should f be in $(\mathbb{R}, 0, +, \geq, =)$? Both clauses must hold for *every* $x \in \mathbb{R}$. $f(x) := x/2$

   What if the structure is $(\mathbb{N}, 0, +, \geq, =)$? Any other f? For $x = 1$ we must have $f(1) + f(1) = 1$, but $2f(1)$ is even, so *no* f works. In fact, φ is false in $\mathbb{N}$. This does not change equisatisfiability!

Topic 18.1

The Problem of Satisfiability for FOL

The Problem of Satisfiability... for FOL

- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.

The Problem of Satisfiability... for FOL

- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.
- Now the same question, but for FOL: given a FOL formula, is it satisfiable?

The Problem of Satisfiability... for FOL

- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.
- Now the same question, but for FOL: given a FOL formula, is it satisfiable?

Why is this harder?

The Problem of Satisfiability... for FOL

- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.
- Now the same question, but for FOL: given a FOL formula, is it satisfiable?

Why is this harder?

- FOL validity and satisfiability are undecidable – no algorithm always halts with yes/no.

The Problem of Satisfiability... for FOL

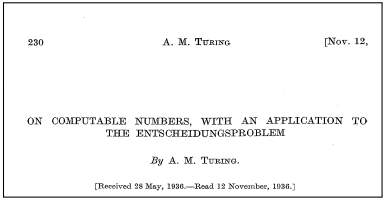
- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.
- Now the same question, but for FOL: given a FOL formula, is it satisfiable?

Why is this harder?

- FOL validity and satisfiability are undecidable – no algorithm always halts with yes/no.
- Quantifiers range over infinite domains ($\mathbb{N}, \mathbb{R}, \ldots$), and FOL can encode computation – no finite truth table to brute-force.

The Problem of Satisfiability... for FOL

- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.
- Now the same question, but for FOL: given a FOL formula, is it satisfiable?



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

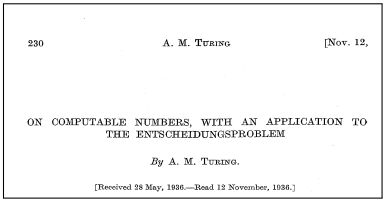
[Received 28 May, 1936.—Read 12 November, 1936.]

Why is this harder?

- FOL validity and satisfiability are undecidable – no algorithm always halts with yes/no.
- Quantifiers range over infinite domains ($\mathbb{N}, \mathbb{R}, \ldots$), and FOL can encode computation – no finite truth table to brute-force.
- In fact, Turing introduced Turing machines while proving this for "Functional calculus K" (the old name for FOL)!

The Problem of Satisfiability... for FOL

- Recall: given a propositional formula, is it satisfiable?
  - Resolution gives UNSAT proofs; DPLL/CDCL always halts with SAT/UNSAT.
- Now the same question, but for FOL: given a FOL formula, is it satisfiable?



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

Why is this harder?

- FOL validity and satisfiability are undecidable – no algorithm always halts with yes/no.
- Quantifiers range over infinite domains ($\mathbb{N}, \mathbb{R}, \ldots$), and FOL can encode computation – no finite truth table to brute-force.
- In fact, Turing introduced Turing machines while proving this for "Functional calculus K" (the old name for FOL)!

So do we give up? **No!**

- restrict to a fixed vocabulary and structure (a "theory") where satisfiability *is* decidable!

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Example 2: BV (fixed-width Bit-Vectors)

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Example 2: BV (fixed-width Bit-Vectors)

- A bit-vector of width n is simply a tuple $(b_{n-1}, \ldots, b_0) \in \{0, 1\}^n$.

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Example 2: BV (fixed-width Bit-Vectors)

- A bit-vector of width n is simply a tuple $(b_{n-1}, \ldots, b_0) \in \{0,1\}^n$.
- Universe $\{0,1\}^n$, $\mathcal{V}_T = \{\&, |, \oplus, +, \leq_u, \ldots\}$ (bitwise ops applied per-bit, plus arithmetic mod 2^n reading the tuple as a number)

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Example 2: BV (fixed-width Bit-Vectors)

- A bit-vector of width n is simply a tuple $(b_{n-1}, \ldots, b_0) \in \{0,1\}^n$.
- Universe $\{0,1\}^n$, $\mathcal{V}_T = \{\&, |, \oplus, +, \leq_u, \ldots\}$ (bitwise ops applied per-bit, plus arithmetic mod 2^n reading the tuple as a number)
- e.g., For a 4-bit register w, doubling it *can overflow* – and become (unsigned) smaller than w itself (e.g. $w = 1100$, i.e. 12, doubled is 11000, truncated to 4 bits it's 1000, i.e. 8).

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Example 2: BV (fixed-width Bit-Vectors)

- A bit-vector of width n is simply a tuple $(b_{n-1}, \ldots, b_0) \in \{0,1\}^n$.
- Universe $\{0,1\}^n$, $\mathcal{V}_T = \{\&, |, \oplus, +, \leq_u, \ldots\}$ (bitwise ops applied per-bit, plus arithmetic mod 2^n reading the tuple as a number)
- e.g., For a 4-bit register w, doubling it *can overflow* – and become (unsigned) smaller than w itself (e.g. $w = 1100$, i.e. 12, doubled is 11000, truncated to 4 bits it's 1000, i.e. 8).
  $w + w <_u w$

Theories: Two Examples

A theory T fixes a vocabulary $\mathcal{V}_T$ and a single intended $\mathcal{V}_T$-structure.

Example 1: LRA (Linear Real Arithmetic)

- Universe $\mathbb{R}$, $\mathcal{V}_T = \{0, 1, +, \leq, =\}$ (only linear comb, no $x \times y$), interpreted as usual.
- e.g, There are real numbers x, y such that $x \geq 5$ and $2x - y = 3$

Example 2: BV (fixed-width Bit-Vectors)

- A bit-vector of width n is simply a tuple $(b_{n-1}, \ldots, b_0) \in \{0,1\}^n$.
- Universe $\{0,1\}^n$, $\mathcal{V}_T = \{\&, |, \oplus, +, \leq_u, \ldots\}$ (bitwise ops applied per-bit, plus arithmetic mod 2^n reading the tuple as a number)
- e.g., For a 4-bit register w, doubling it *can overflow* – and become (unsigned) smaller than w itself (e.g. $w = 1100$, i.e. 12, doubled is 11000, truncated to 4 bits it's 1000, i.e. 8).

  $w + w <_u w$

  Used in hardware/crypto/systems verification.

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution
    - How would you check if you only had equality constraints?

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution
    - How would you check if you only had equality constraints? Gaussian elimination!

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution
    - How would you check if you only had equality constraints? Gaussian elimination!
    - Use "Simplex Algorithm" to check feasibility of corresponding Linear Program.

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution
    - How would you check if you only had equality constraints? Gaussian elimination!
    - Use "Simplex Algorithm" to check feasibility of corresponding Linear Program.
  - BV: each variable has a finite domain (2^n values) –

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution
    - How would you check if you only had equality constraints? Gaussian elimination!
    - Use "Simplex Algorithm" to check feasibility of corresponding Linear Program.
  - BV: each variable has a finite domain (2^n values) – bit-blast into an equisatisfiable propositional formula, then run usual DPLL/CDCL !

Satisfiability Modulo Theories

The T-satisfiability problem a.k.a. Satisfiability Modulo Theories (SMT)

Given a quantifier-free $\mathcal{V}_T$-formula, is there an assignment of *values in T's universe* that makes it true?

- For both theories we considered, T-satisfiability is decidable!
  - That is, there is an algorithm that always halts with yes/no.
  - (Not every theory qualifies – e.g. general nonlinear arithmetic over the integers doesn't.)
  - LRA: It suffices to check if a conjunction of linear (in)equalities has a solution
    - How would you check if you only had equality constraints? Gaussian elimination!
    - Use "Simplex Algorithm" to check feasibility of corresponding Linear Program.
  - BV: each variable has a finite domain (2^n values) – bit-blast into an equisatisfiable propositional formula, then run usual DPLL/CDCL !

replace each n-bit variable by n Bool variables, and each op by a circuit (AND/OR/XOR gates) over those bits.

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories?

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

- Atomic $\mathcal{V}_{\mathcal{T}}$-formulas – e.g. $x \leq 5$ or $x + y = 3$ – are called theory atoms.

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

- Atomic $\mathcal{V}_{\mathcal{T}}$-formulas – e.g. $x \leq 5$ or $x + y = 3$ – are called theory atoms.
- Take the quantifier-free $\mathcal{V}_{\mathcal{T}}$-formula in CNF and replace each distinct theory atom by a fresh Boolean variable – the Boolean skeleton.

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

- Atomic $\mathcal{V}_{\mathcal{T}}$-formulas – e.g. $x \leq 5$ or $x + y = 3$ – are called theory atoms.
- Take the quantifier-free $\mathcal{V}_{\mathcal{T}}$-formula in CNF and replace each distinct theory atom by a fresh Boolean variable – the Boolean skeleton. $(x \leq 5 \vee x + y = 3) \wedge \neg(x \leq 5) \rightsquigarrow (p \vee q) \wedge \neg p$

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

- Atomic $\mathcal{V}_\mathcal{T}$-formulas – e.g. $x \leq 5$ or $x + y = 3$ – are called theory atoms.
- Take the quantifier-free $\mathcal{V}_\mathcal{T}$-formula in CNF and replace each distinct theory atom by a fresh Boolean variable – the Boolean skeleton. $(x \leq 5 \vee x + y = 3) \wedge \neg(x \leq 5) \rightsquigarrow (p \vee q) \wedge \neg p$
- Run DPLL/CDCL on the skeleton to find a candidate assignment.

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

- Atomic $\mathcal{V}_{\mathcal{T}}$-formulas – e.g. $x \leq 5$ or $x + y = 3$ – are called theory atoms.
- Take the quantifier-free $\mathcal{V}_{\mathcal{T}}$-formula in CNF and replace each distinct theory atom by a fresh Boolean variable – the Boolean skeleton. $(x \leq 5 \vee x + y = 3) \wedge \neg(x \leq 5) \rightsquigarrow (p \vee q) \wedge \neg p$
- Run DPLL/CDCL on the skeleton to find a candidate assignment.
- Ask an underlying T-solver: is this *conjunction of theory literals* satisfiable in T's structure?

Extending DPLL to DPLL(T)

So, can we extend DPLL/CDCL to handle theories? Yes!

- Atomic $\mathcal{V}_{\mathcal{T}}$-formulas – e.g. $x \leq 5$ or $x + y = 3$ – are called theory atoms.
- Take the quantifier-free $\mathcal{V}_{\mathcal{T}}$-formula in CNF and replace each distinct theory atom by a fresh Boolean variable – the Boolean skeleton. $(x \leq 5 \vee x + y = 3) \wedge \neg(x \leq 5) \rightsquigarrow (p \vee q) \wedge \neg p$
- Run DPLL/CDCL on the skeleton to find a candidate assignment.
- Ask an underlying T-solver: is this *conjunction of theory literals* satisfiable in T's structure?
- If yes: SAT, done. If no: (learn a conflict clause and) keep searching.

DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.

DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.

DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1 := (p_1 \vee p_2)$, $c_2 := p_3$, $c_3 := p_4$.

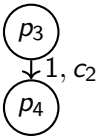
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1 := (p_1 \vee p_2)$, $c_2 := p_3$, $c_3 := p_4$.



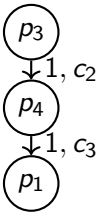
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1 := (p_1 \vee p_2)$, $c_2 := p_3$, $c_3 := p_4$.



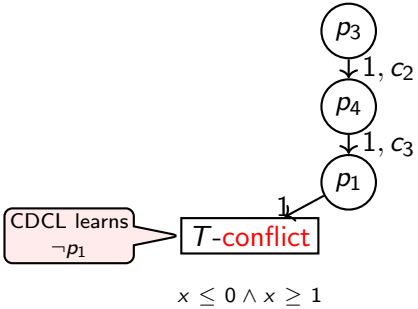
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1 := (p_1 \vee p_2)$, $c_2 := p_3$, $c_3 := p_4$.



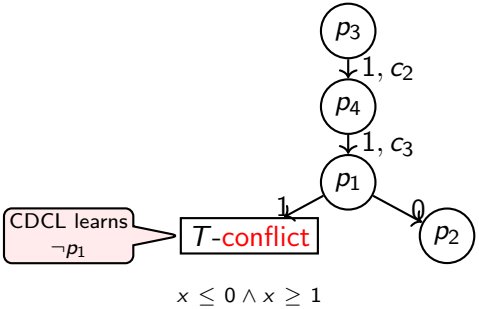
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1 := (p_1 \vee p_2)$, $c_2 := p_3$, $c_3 := p_4$.



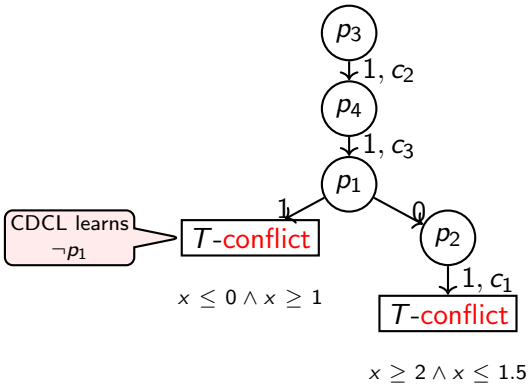
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1{:=}(x \leq 0)$, $p_2{:=}(x \geq 2)$, $p_3{:=}(x \geq 1)$, $p_4{:=}(x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1{:=}(p_1 \vee p_2)$, $c_2{:=}p_3$, $c_3{:=}p_4$.



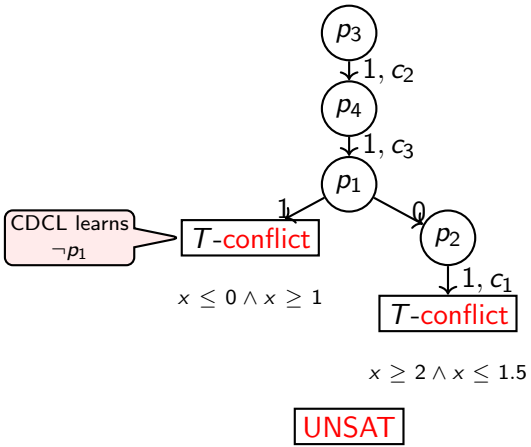
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1{:=}(x \leq 0)$, $p_2{:=}(x \geq 2)$, $p_3{:=}(x \geq 1)$, $p_4{:=}(x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1{:=}(p_1 \vee p_2)$, $c_2{:=}p_3$, $c_3{:=}p_4$.



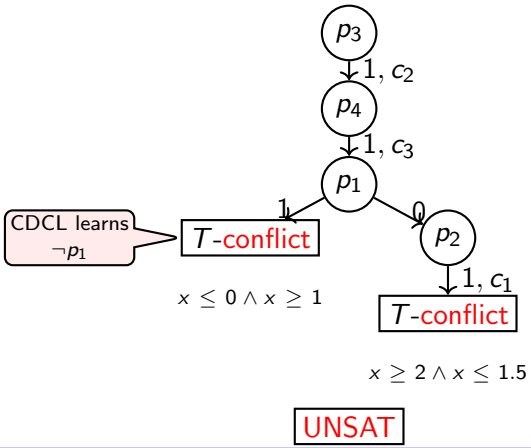
DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1 := (x \leq 0)$, $p_2 := (x \geq 2)$, $p_3 := (x \geq 1)$, $p_4 := (x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1 := (p_1 \vee p_2)$, $c_2 := p_3$, $c_3 := p_4$.



DPLL(T): A Worked Example (LRA)

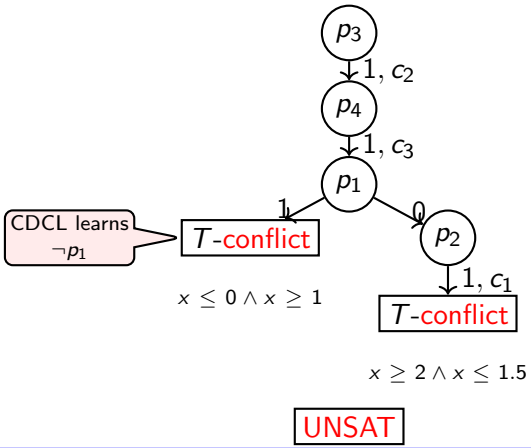
- Input: A quantifier-free LRA-formula in CNF $\varphi := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 1.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1{:=}(x \leq 0)$, $p_2{:=}(x \geq 2)$, $p_3{:=}(x \geq 1)$, $p_4{:=}(x \leq 1.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1{:=}(p_1 \vee p_2)$, $c_2{:=}p_3$, $c_3{:=}p_4$.



- p_1 was the only real decision – both values lead to a theory conflict, so φ is UNSAT.

DPLL(T): A Worked Example (LRA)

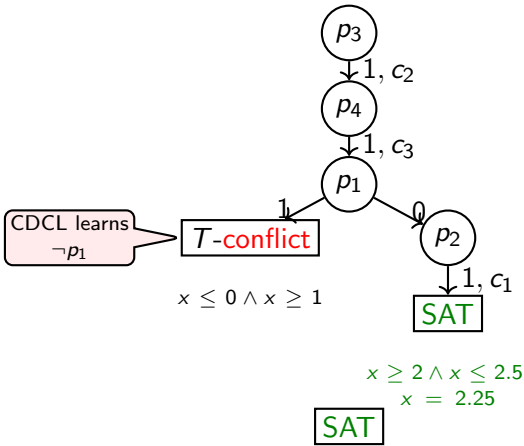
- Input: A quantifier-free LRA-formula in CNF $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1{:=}(x \leq 0)$, $p_2{:=}(x \geq 2)$, $p_3{:=}(x \geq 1)$, $p_4{:=}(x \leq 2.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1{:=}(p_1 \vee p_2)$, $c_2{:=}p_3$, $c_3{:=}p_4$.



- p_1 was the only real decision – both values lead to a theory conflict, so φ is UNSAT.
- What if $c_3 := (x \leq 2.5)$ instead? Only the p_2 branch changes

DPLL(T): A Worked Example (LRA)

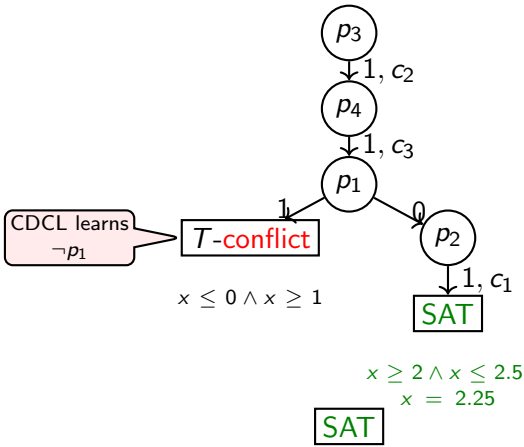
- Input: A quantifier-free LRA-formula in CNF $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1{:=}(x \leq 0)$, $p_2{:=}(x \geq 2)$, $p_3{:=}(x \geq 1)$, $p_4{:=}(x \leq 2.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1{:=}(p_1 \vee p_2)$, $c_2{:=}p_3$, $c_3{:=}p_4$.



- p_1 was the only real decision – both values lead to a theory conflict, so φ is UNSAT.
- What if $c_3 := (x \leq 2.5)$ instead? Only the p_2 branch changes
- T-solver check:
  $\{x \geq 1, x \leq 2.5, x > 0, x \geq 2\}$ – SAT!
  witness $x = 2.25$.

DPLL(T): A Worked Example (LRA)

- Input: A quantifier-free LRA-formula in CNF $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- Theory atoms $\rightarrow$ Bool variables: $p_1{:=}(x \leq 0)$, $p_2{:=}(x \geq 2)$, $p_3{:=}(x \geq 1)$, $p_4{:=}(x \leq 2.5)$.
- Boolean skeleton: $c_1 \wedge c_2 \wedge c_3$, where $c_1{:=}(p_1 \vee p_2)$, $c_2{:=}p_3$, $c_3{:=}p_4$.



- p_1 was the only real decision – both values lead to a theory conflict, so φ is UNSAT.
- What if $c_3 := (x \leq 2.5)$ instead? Only the p_2 branch changes
- T-solver check: $\{x \geq 1, x \leq 2.5, x > 0, x \geq 2\}$ – SAT! witness $x = 2.25$.
- So φ' is satisfiable – the $p_1{=}0, p_2{=}1$ branch succeeded.

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- $\exists x\, \varphi'(x)$: true – the witness $x = 2.25$ comes for free: it is the model the T-solver already found!

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- $\exists x\, \varphi'(x)$: true – the witness $x = 2.25$ comes for free: it is the model the T-solver already found!
- $\forall x\, \varphi'(x)$ is true iff $\neg\varphi'(x)$ is UNSAT – again just DPLL(T)!

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- $\exists x\, \varphi'(x)$: true – the witness $x = 2.25$ comes for free: it is the model the T-solver already found!
- $\forall x\, \varphi'(x)$ is true iff $\neg\varphi'(x)$ is UNSAT – again just DPLL(T)!
  - $\neg\varphi'(x) \equiv (x > 0 \wedge x < 2) \vee x < 1 \vee x > 2.5$

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- $\exists x\, \varphi'(x)$: true – the witness $x = 2.25$ comes for free: it is the model the T-solver already found!
- $\forall x\, \varphi'(x)$ is true iff $\neg\varphi'(x)$ is UNSAT – again just DPLL(T)!
  - $\neg\varphi'(x) \equiv (x > 0 \wedge x < 2) \vee x < 1 \vee x > 2.5$
  - It is SAT, e.g. x=0 $(0 < 1)$.

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- $\exists x\, \varphi'(x)$: true – the witness $x = 2.25$ comes for free: it is the model the T-solver already found!
- $\forall x\, \varphi'(x)$ is true iff $\neg\varphi'(x)$ is UNSAT – again just DPLL(T)!
  - $\neg\varphi'(x) \equiv (x > 0 \wedge x < 2) \vee x < 1 \vee x > 2.5$
  - It is SAT, e.g. x=0 $(0 < 1)$.
  - So $\forall x\, \varphi'(x)$ is false.

Adding Quantifiers

- In the modified, satisfiable version: $\varphi' := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.
- $\exists x\, \varphi'(x)$: true – the witness $x = 2.25$ comes for free: it is the model the T-solver already found!
- $\forall x\, \varphi'(x)$ is true iff $\neg\varphi'(x)$ is UNSAT – again just DPLL(T)!
  - $\neg\varphi'(x) \equiv (x > 0 \wedge x < 2) \vee x < 1 \vee x > 2.5$
  - It is SAT, e.g. x=0 $(0 < 1)$.
  - So $\forall x\, \varphi'(x)$ is false.
- Bottomline: one block of $\exists$ or of $\forall$ over a quantifier-free formula is just SAT/UNSAT. The trouble starts when quantifiers alternate!

Quantifier Alternation is Hard

$\varphi'(x) := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.

Pure existential

- $\exists x\, \varphi'(x)$: true, witness $x = 2.25$ – just the SAT problem DPLL(T) already solved!

Quantifier Alternation is Hard

$\varphi'(x) := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.

Pure existential

- $\exists x\, \varphi'(x)$: true, witness $x = 2.25$ – just the SAT problem DPLL(T) already solved!

Now alternate: $\forall y\, \exists x$

- Let $\psi(x, y) := (x \leq y \vee x \geq y{+}2) \wedge (x \geq y{+}1) \wedge (x \leq y{+}2.5)$ -same shape as φ', shifted by y.

Quantifier Alternation is Hard

$\varphi'(x) := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.

Pure existential

- $\exists x\, \varphi'(x)$: true, witness $x = 2.25$ – just the SAT problem DPLL(T) already solved!

Now alternate: $\forall y\, \exists x$

- Let $\psi(x, y) := (x \leq y \vee x \geq y{+}2) \wedge (x \geq y{+}1) \wedge (x \leq y{+}2.5)$ -same shape as φ', shifted by y.
- $\forall y\, \exists x\, \psi(x, y)$: for *every* y, need a witness x – but now it can (must!) depend on y. A single fixed x won't work for all y at once.

Quantifier Alternation is Hard

$\varphi'(x) := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.

Pure existential

- $\exists x\, \varphi'(x)$: true, witness $x = 2.25$ – just the SAT problem DPLL(T) already solved!

Now alternate: $\forall y\, \exists x$

- Let $\psi(x, y) := (x \leq y \vee x \geq y{+}2) \wedge (x \geq y{+}1) \wedge (x \leq y{+}2.5)$ -same shape as φ', shifted by y.
- $\forall y\, \exists x\, \psi(x, y)$: for *every* y, need a witness x – but now it can (must!) depend on y. A single fixed x won't work for all y at once.
- Here $x := y + 2.25$ works for *every* y (the old witness, shifted) – that dependency is exactly a Skolem function!

Quantifier Alternation is Hard

$\varphi'(x) := (x \leq 0 \vee x \geq 2) \wedge (x \geq 1) \wedge (x \leq 2.5)$.

Pure existential

- $\exists x\, \varphi'(x)$: true, witness $x = 2.25$ – just the SAT problem DPLL(T) already solved!

Now alternate: $\forall y\, \exists x$

- Let $\psi(x, y) := (x \leq y \vee x \geq y{+}2) \wedge (x \geq y{+}1) \wedge (x \leq y{+}2.5)$ -same shape as φ', shifted by y.
- $\forall y\, \exists x\, \psi(x, y)$: for *every* y, need a witness x – but now it can (must!) depend on y. A single fixed x won't work for all y at once.
- Here $x := y + 2.25$ works for *every* y (the old witness, shifted) – that dependency is exactly a Skolem function!
- In general, *finding* that function (or enough instantiations to pin it down) is the hard part – that's where the real search/work in quantifier instantiation lives.

Summary

- Satisfiability for FOL is undecidable – so fix a theory T (LRA, BV, ...) where it is decidable: SMT.

Summary

- Satisfiability for FOL is undecidable – so fix a theory T (LRA, BV, ...) where it is decidable: SMT.
- DPLL(T) = CDCL on the Boolean skeleton + a T-solver for conjunctions of theory literals.

Summary

- Satisfiability for FOL is undecidable – so fix a theory T (LRA, BV, ...) where it is decidable: SMT.
- DPLL(T) = CDCL on the Boolean skeleton + a T-solver for conjunctions of theory literals.
- *Quantifier-free* is somewhat easy, depending on the theory; so is even one block of quantifiers (just SAT/UNSAT in T) – but alternating quantifiers are hard!

Summary

- Satisfiability for FOL is undecidable – so fix a theory T (LRA, BV, ...) where it is decidable: SMT.
- DPLL(T) = CDCL on the Boolean skeleton + a T-solver for conjunctions of theory literals.
- *Quantifier-free* is somewhat easy, depending on the theory; so is even one block of quantifiers (just SAT/UNSAT in T) – but alternating quantifiers are hard!

That wraps up our module on first-order logic

- Payoff: SMT solvers such as Z3 put much of the power of FOL to work – with decision procedures inside the theories (and heuristics beyond).
- Optional Z3 demo soon!

Topic 18.2

Introduction to Program Verification

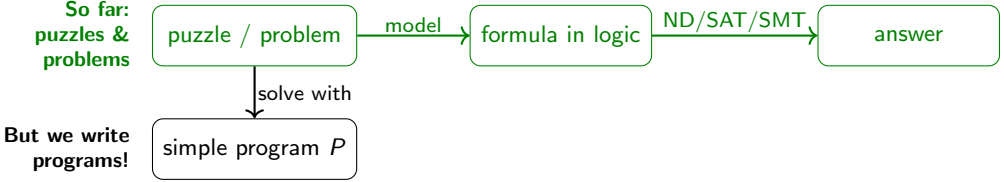
Next topic alert!

An introduction to Program Verification



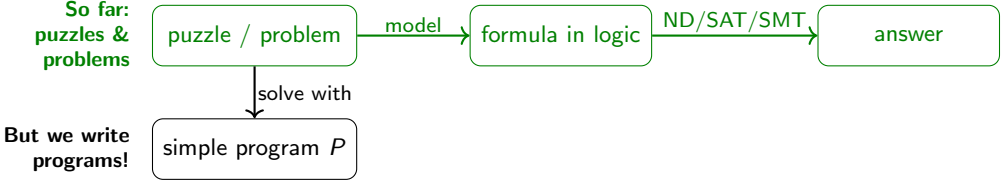
Next topic alert!

An introduction to Program Verification



Next topic alert!

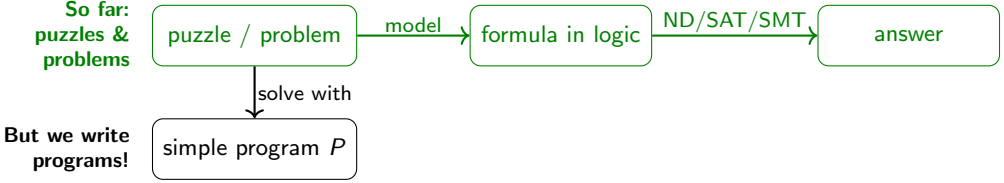
An introduction to Program Verification



- Can we use logic to reason about programs, too?

Next topic alert!

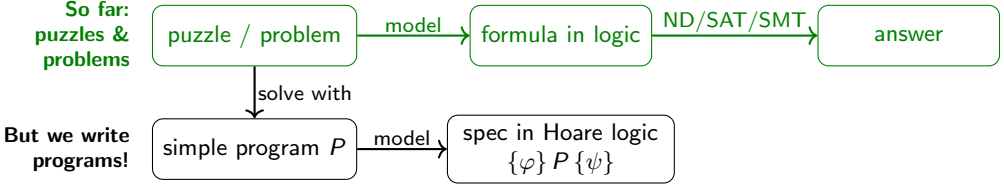
An introduction to Program Verification



- Can we use logic to reason about programs, too?
- How?

Next topic alert!

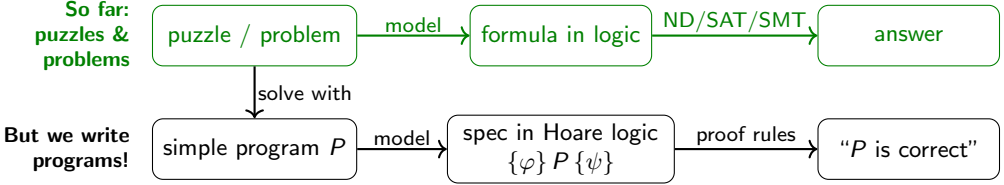
An introduction to Program Verification



- Can we use logic to reason about programs, too?
- How? A logic of programs, Hoare logic, built on top of FOL!

Next topic alert!

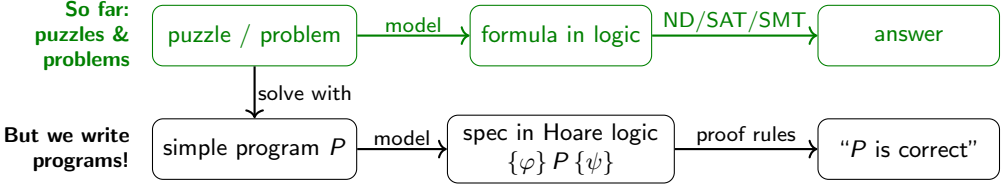
An introduction to Program Verification



- Can we use logic to reason about programs, too?
- How? A logic of programs, Hoare logic, built on top of FOL!
- In this course: for *simple* imperative programs, we'll see how to *state* properties in logic (pre/postconditions) and *prove* them using proof rules.

Next topic alert!

An introduction to Program Verification



- Can we use logic to reason about programs, too?
- How? A logic of programs, Hoare logic, built on top of FOL!
- In this course: for *simple* imperative programs, we'll see how to *state* properties in logic (pre/postconditions) and *prove* them using proof rules.
- Real program verifiers (e.g., Dafny) largely automate the proof step using SMT solvers!

Remember Ariane 5? (Lecture 0)



June 4, 1996 – Ariane 5, Flight 501

- Ariane 4 code, reused in Ariane 5, overflowed – the rocket was lost within a minute of lift-off.

Remember Ariane 5? (Lecture 0)



June 4, 1996 – Ariane 5, Flight 501

- Ariane 4 code, reused in Ariane 5, overflowed – the rocket was lost within a minute of lift-off.
- On Ariane 4 that value never got so large: the code was correct under Ariane 4's assumptions, but not under Ariane 5's.

Remember Ariane 5? (Lecture 0)



June 4, 1996 – Ariane 5, Flight 501

- Ariane 4 code, reused in Ariane 5, overflowed – the rocket was lost within a minute of lift-off.
- On Ariane 4 that value never got so large: the code was correct under Ariane 4's assumptions, but not under Ariane 5's.
- A violated precondition!

Remember Ariane 5? (Lecture 0)



June 4, 1996 – Ariane 5, Flight 501

- Ariane 4 code, reused in Ariane 5, overflowed – the rocket was lost within a minute of lift-off.
- On Ariane 4 that value never got so large: the code was correct under Ariane 4's assumptions, but not under Ariane 5's.
- A violated precondition!

What does "correct" mean?

"Correct" only makes sense relative to an assumed precondition and a promised postcondition

Remember Ariane 5? (Lecture 0)



June 4, 1996 – Ariane 5, Flight 501

- Ariane 4 code, reused in Ariane 5, overflowed – the rocket was lost within a minute of lift-off.
- On Ariane 4 that value never got so large: the code was correct under Ariane 4's assumptions, but not under Ariane 5's.
- A violated precondition!

What does "correct" mean?

"Correct" only makes sense relative to an assumed precondition and a promised postcondition

How do we formalize this in logic?