Introduction to Automated Theorem Proving

IA085: Satisfiability and Automated Reasoning

Martin Jonáš

FI MUNI, Spring 2024

Contents

Propositional satisfiability (SAT)

- $(A \vee \neg B) \wedge (\neg A \vee C)$
- is it satisfiable?

Satisfiability modulo theories (SMT)

- $x = 1 \;\wedge\; x = y + y \;\wedge\; y > 0$
- is it satisfiable over reals?
- is it satisfiable over integers?

Automated theorem proving (ATP)

- axioms: $\forall x\,(x + x = 0)$, $\forall x \forall y\,(x + y = y + x)$
- do they imply $\forall x \forall y\,((x + y) + (y + x) = 0)$?

Big change

Today we are dealing with quantifiers!

Big change

Today we are dealing with quantifiers!

Today we are not dealing with theories!

First-order theorem proving

Problem specification

Input

- a set of hypotheses $\{H_1, H_2, \ldots, H_k\}$ that are arbitrary closed formulas
- a goal G that is arbitrary closed formula

Problem

- decide whether $H_1 \wedge H_2 \wedge \ldots \wedge H_k \models G$

Notes

- not considering any background theory, only interpreted symbol is equality (theory of UF)

Example

Claim If all elements of a group have order 2, the group is commutative.

Example

Claim If all elements of a group have order 2, the group is commutative.

Formalization in signature $\Sigma = \{=, \cdot, 1\}$

Hypotheses

- $H_1 = \forall x\,(1 \cdot x = x \ \wedge\ x \cdot 1 = x)$
- $H_2 = \forall x \exists y\,(x \cdot y = 1)$
- $H_3 = \forall x \forall y \forall z\,((x \cdot y) \cdot z = x \cdot (y \cdot z))$
- $H_4 = \forall x\,(x \cdot x = 1)$

Goal

- $G = \forall x \forall y (x \cdot y = y \cdot x)$

Task

$$H_1 \wedge H_2 \wedge H_3 \wedge H_4 \models G$$

Proof by refutation

Goal

- prove $H_1 \wedge H_2 \wedge \ldots \wedge H_k \models G$

Proof by refutation

- prove $H_1 \wedge H_2 \wedge \ldots \wedge H_k \wedge \neg G$ is unsatisfiable

Proving unsatisfiability

System $\mathbb{I}$ of inference rules

$$\frac{C_1 \quad C_2 \quad \dots \quad C_k}{A}$$

Proof of unsatisfiability of set of formulas Φ is a tree with

- leaves from Φ
- inner nodes corresponding to inference rules
- root $\bot$

Demo

Sound inference rule

- if

$$\frac{C_1 \quad C_2 \quad \dots \quad C_k}{A}$$

- then

$$C_1 \wedge C_2 \wedge \dots \wedge C_k \models A$$

Proving unsatisfiability

Important distinction

- Φ is unsatisfiable ($\Phi \models \bot$)
- Φ can be proven unsatisfiable using the proof system $\mathbb{I}$ ($\Phi \vdash \bot$)

Soundness

- $\Phi \vdash \bot$ implies $\Phi \models \bot$
- can be proven by proving soundness of each inference rule separately

Refutation completeness

- $\Phi \models \bot$ implies $\Phi \vdash \bot$
- proofs usually much harder

Proving unsatisfiability

Proving unsatisfiability of sets of first-order formulas

- in general undecidable

Proving unsatisfiability

Proving unsatisfiability of sets of first-order formulas

- in general undecidable

Challenge for the rest of the lecture

- Is the problem semi-decidable (recursively enumerable)?
- Is its complement semi-decidable (recursively enumerable)?

Today

Two proof systems

- **resolution** – sound and refutation complete for formulas without equalities
- **superposition** – sound and refutation complete for arbitrary formulas (with or without equalities)

Notation

- variables $x, y, z, \ldots$
- set of variables $\overrightarrow{X}$
- constants c, d (in the Σ-signature, fixed elements of the Σ-structure)
- $t[s]$ = a term t that can contain a subterm s
- given $t[s]$, we denote as $t[s_2]$ the result of replacing s in t by s_2

Normal forms

Goal

We want to convert the input formula to a conjunctive normal form.

- **atomic formula** = predicate symbol applied to terms $(P(x, f(y), g(c)))$
- **literal** = atomic formula or its negation
- **clause** = disjunction of literals with all variables quantified universally $(\forall x \forall y. \, (P(x, f(y), g(c)) \vee Q(y)))$
- **formula in CNF** = conjunction of clauses

Negation normal form

Rationale

- we want to remove existential quantifiers
- some universal quantifiers under negations are in fact existential

Negation Normal Form (NNF)

- negations are applied only to atomic formulas
- the formula does not contain implication ($\rightarrow$) and equivalence ($\leftrightarrow$)

Conversion to NNF

1. rewrite all $\varphi \leftrightarrow \psi$ to $(\varphi \rightarrow \psi) \wedge (\varphi \leftarrow \psi)$
2. rewrite all $\varphi \rightarrow \psi$ to $\neg\varphi \vee \psi$
3. apply double negation elimination, De Morgan rules, and quantifier negations until fixed point
   - rewrite $\neg\neg\varphi$ to φ
   - rewrite $\neg(\varphi \wedge \psi)$ to $(\neg\varphi) \vee (\neg\psi)$
   - rewrite $\neg(\varphi \vee \psi)$ to $(\neg\varphi) \wedge (\neg\psi)$
   - rewrite $\neg(\exists x\, \varphi)$ to $\forall x\, \neg\varphi$
   - rewrite $\neg(\forall x\, \varphi)$ to $\exists x\, \neg\varphi$

If the formulas are represented by DAGs, the conversion is linear.

Prenex normal form

Rationale

- we want to move the quantifiers to the top level (to create clauses)

Prenex Normal Form (PNF)

- formula is of form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \, \varphi$ where
- $Q_i \in \{\exists, \forall\}$
- φ is quantifier free

Conversion to PNF

1. convert to NNF
2. rename bound variables to unique names
3. apply prenexing rules until fixed point
   - rewrite $\varphi \wedge (\forall x\, \psi)$ to $\forall x\, (\varphi \wedge \psi)$
   - rewrite $\varphi \wedge (\exists x\, \psi)$ to $\forall x\, (\exists \wedge \psi)$
   - + symmetric variants

Skolem normal form

Skolem Normal Form (SNF)

- formula is of form $\forall x_1 \forall x_2 \ldots \forall x_n \, \varphi$ where
- φ is quantifier free

Conversion to SNF

Conversion to SNF

1. convert to PNF
2. while the formula is of form

$$\forall x_1 \forall x_2 \ldots \forall x_m \exists y . \varphi,$$

   where φ can contain quantifiers, replace y by $f_y(x_1, x_2, \ldots, x_m)$, where f_y is a new function symbol

Conversion to SNF

The formula $\mathit{skolemize}(\varphi)$ is in general not equivalent to φ.

- $\varphi = \forall x \exists y\, (x + y = 0)$
- $\mathit{skolemize}(\varphi) = \forall x\, (x + f(x) = 0)$

Conversion to SNF

The formula $skolemize(\varphi)$ is in general not equivalent to φ.

- $\varphi = \forall x \exists y \, (x + y = 0)$
- $skolemize(\varphi) = \forall x \, (x + f(x) = 0)$

Theorem
The formulas φ and $skolemize(\varphi)$ are equisatisfiable.

Conversion to CNF

1. convert to an equisatisfiable formula in SNF
2. obtain formula $\forall \overrightarrow{X}\, \varphi$
3. convert φ to CNF (using distributivity or Tseitin)
4. obtain formula $\forall \overrightarrow{X}\, (C_1 \wedge C_2 \wedge \ldots \wedge C_k)$
5. obtain a set of formulas $(\forall \overrightarrow{X_1}\, C_1) \wedge (\forall \overrightarrow{X_2}\, C_2) \wedge \ldots (\forall \overrightarrow{X_k}\, C_k)$

Conversion to CNF

1. convert to an equisatisfiable formula in SNF
2. obtain formula $\forall \overrightarrow{X}\, \varphi$
3. convert φ to CNF (using distributivity or Tseitin)
4. obtain formula $\forall \overrightarrow{X}\, (C_1 \wedge C_2 \wedge \ldots \wedge C_k)$
5. obtain a set of formulas $(\forall \overrightarrow{X_1}\, C_1) \wedge (\forall \overrightarrow{X_2}\, C_2) \wedge \ldots (\forall \overrightarrow{X_k}\, C_k)$
   (why is this correct?)

Conversion to CNF: improvements

In practice, conversion to full PNF is not needed.

Improvements

- convert to conjunctions of formulas in PNF, or
- convert to NNF, perform skolemization, then convert to conjunctions of formulas in PNF

Why is this better?

CNF conventions

- clauses = sets of literals
- formulas = sets of clauses
- all variables are universally quantified $\rightarrow$ do not write the quantifiers

Example

- $(\forall P(x)) \ \wedge \ (\forall y(Q(c_1, y) \vee Q(c_2, y)))$
- $\{\{P(x)\}, \{Q(c_1, y), Q(c_2, y)\}$

Example

Recall the example

- $H_1 = \forall x\,(1 \cdot x = x \ \wedge\ x \cdot 1 = x)$
- $H_2 = \forall x \exists y\,(x \cdot y = 1)$
- $H_3 = \forall x \forall y \forall z\,((x \cdot y) \cdot z = x \cdot (y \cdot z))$
- $H_4 = \forall x\,(x \cdot x = 1)$
- $C = \forall x \forall y (x \cdot y = y \cdot x)$

What is the CNF of the formula that we are trying to refute?

Saturation based theorem-proving

Saturation algorithm

1. start with the set of formulas Φ (not containing $\perp$)
2. while there is an inference rule $i \in \mathbb{I}$ with premises from Φ and conclusion $A \notin \Phi$
   - if $A = \perp$, return unsatisfiable
   - otherwise set Φ to $\Phi \cup \{A\}$ and continue
3. otherwise, there are no new inferences to add (the set Φ is saturated)
4. return satisfiable

Theorem
Let $\mathbb{I}$ be a sound proof system. If the set of formulas Φ is satisfiable, the saturation algorithm either returns satisfiable or does not terminate.

Not a theorem
Let $\mathbb{I}$ be a complete proof system. If the set of formulas Φ is unsatisfiable, the saturation algorithm returns unsatisfiable.

Theorem
Let $\mathbb{I}$ be a complete proof system. If the set of formulas Φ is unsatisfiable, the saturation algorithm returns unsatisfiable, given that rule selection is fair.

First-order resolution

Reminder: Propositional resolution

$$\frac{A \vee C_1 \quad \neg A \vee C_2}{C_1 \vee C_2}$$

In first-order logic, the above rule is still sound, but is not complete.

- $\{\{P(x)\}, \{\neg P(y+z)\}\} \models \bot$
- $\{\{P(x)\}, \{\neg P(y+z)\}\} \not\vdash \bot$

Reminder: Propositional resolution

$$\frac{A \vee C_1 \quad \neg A \vee C_2}{C_1 \vee C_2}$$

In first-order logic, the above rule is still sound, but is not complete.

- $\{\{P(x)\}, \{\neg P(y+z)\}\} \models \bot$
- $\{\{P(x)\}, \{\neg P(y+z)\}\} \not\vdash \bot$

Solution

- allow not only exactly matching complementary literals A and $\neg A$,
- but also literals that can be made complementary by using a suitable instantiation of universal quantifiers

Substitution

Substitution

- a function from variables to terms
- $\theta = \{x \mapsto y + 3, y \mapsto f(y)\}$
- not mentioned variables are not changed ($\theta(z) = z$)

Application

- formula φ
- result $\varphi\theta$ is result of simultaneous replacement of each x in φ by $\theta(x)$
- $(x - y)\theta = (y + 3) - f(y)$
- analogous $t\theta$ for terms

Substitution

Theorem
If C is a clause and θ is a substitution, then $C \models C\theta$.

Message

If we have a set of clauses Φ, it is safe to apply substitution θ to $C \in \Phi$ and assume that $C\theta$ holds.

Unification

Unifier of terms t and s

- substitution θ such that $t\theta = s\theta$
- analogous definition for atomic formulas

Examples

- $f(g(x), y)$ and $f(z, h(z))$ are

Unification

Unifier of terms t and s

- substitution θ such that $t\theta = s\theta$
- analogous definition for atomic formulas

Examples

- $f(g(x), y)$ and $f(z, h(z))$ are unifiable
- $f(g(x), y)$ and $f(h(z), h(z))$ are

Unification

Unifier of terms t and s

- substitution θ such that $t\theta = s\theta$
- analogous definition for atomic formulas

Examples

- $f(g(x), y)$ and $f(z, h(z))$ are unifiable
- $f(g(x), y)$ and $f(h(z), h(z))$ are not unifiable
- $f(f(x))$ and $f(c)$ are

Unification

Unifier of terms t and s

- substitution θ such that $t\theta = s\theta$
- analogous definition for atomic formulas

Examples

- $f(g(x), y)$ and $f(z, h(z))$ are unifiable
- $f(g(x), y)$ and $f(h(z), h(z))$ are not unifiable
- $f(f(x))$ and $f(c)$ are not unifiable

Most-general unifier

Problem

- there are many different unifiers for a given set of pairs
- unifier of x and $f(y)$ can be $\theta = \{x \mapsto f(10), y \mapsto 10\}$, which is too specific

Most general unifier of terms t and s

- unifier θ such that every unifier ρ can be obtained as $\rho = f \circ \theta$ for suitable substitution f
- unique up to isomorphism (variable renaming)
- denoted $\mathrm{mgu}(t, s)$

Resolution rule

$$\frac{A_1 \vee C_1 \quad \neg A_2 \vee C_2}{C_1\theta \vee C_2\theta} \text{ if } \theta = \mathrm{mgu}(A_1, A_2)$$

Resolution rule

$$\frac{A_1 \vee C_1 \quad \neg A_2 \vee C_2}{C_1\theta \vee C_2\theta} \text{ if } \theta = \mathrm{mgu}(A_1, A_2)$$

Examples

- resolvent of $P(x) \vee Q(f(x), y)$ and $\neg P(f(z)) \vee R(g(z, v))$

Resolution rule

$$\frac{A_1 \vee C_1 \quad \neg A_2 \vee C_2}{C_1\theta \vee C_2\theta} \text{ if } \theta = \mathrm{mgu}(A_1, A_2)$$

Examples

- resolvent of $P(x) \vee Q(f(x), y)$ and $\neg P(f(z)) \vee R(g(z, v))$

Theorem
The resolution inference rule is sound.

Resolution rule

Note

- need to allow renaming of bound variables of a clause before resolution
- example: $P(x) \rightsquigarrow P(y)$ (sound because $\forall x\, P(x) \equiv \forall y\, P(y)$)
- why is this needed?

Resolution proof system

- sound for all first-order formulas
- refutation complete for first-order formulas without equality

Superposition calculus

Motivation

Goal

- extend the resolution proof system with rules that reason with equality
- make it refutation complete for all first-order formulas (with or without equality)

Superposition rule

Naive version (not used)

$$\frac{(l = r) \lor C_1 \quad L[l] \lor C_2}{L[r] \lor C_1 \lor C_2}$$

Superposition rule

Naive version (not used)

$$\frac{(l = r) \vee C_1 \quad L[l] \vee C_2}{L[r] \vee C_1 \vee C_2}$$

General version

$$\frac{(l = r) \vee C_1 \quad L[s] \vee C_2}{(L[r] \vee C_1 \vee C_2)\theta} \text{ if } \theta = \mathrm{mgu}(l, s)$$

Purpose

- use the equality for substitution

Factoring rule

Naive version (not used)

$$\frac{A \vee A \vee C}{A \vee C}$$

Factoring rule

Naive version (not used)

$$\frac{A \lor A \lor C}{A \lor C}$$

General version

$$\frac{A_1 \lor A_2 \lor C}{(A \lor C)\theta} \text{ if } \theta = \mathrm{mgu}(A_1, A_2)$$

Purpose

- remove duplicate literals

Equality resolution rule

Naive version (not used)

$$\frac{(x \neq x) \vee C}{C}$$

General version

$$\frac{(s \neq t) \vee C}{C\theta} \text{ if } \theta = \mathrm{mgu}(s, t)$$

Purpose

- remove disequalities

Equality factoring rule

Naive version (not used)

$$\frac{(s = t) \lor (s = t') \lor C}{(s = t) \lor (t \neq t') \lor C}$$

General version

$$\frac{(s = t) \lor (s' = t') \lor C}{((s = t) \lor (t \neq t') \lor C)\theta} \text{ if } \theta = \mathrm{mgu}(s, s')$$

Purpose

- case split on $s = t$ and $s \neq t$

Superposition calculus

Superpositon calculus

- inference system with rules: resolution, superposition, factoring, equality resolution, and equality factoring rules
- sound for arbitrary first-order formulas
- refutationally complete for arbitrary first-order formulas (with or without equalities)

Next time

- SMT with quantifiers
- quantifier instantiation