

Ordering-based strategies for theorem proving

Maria Paola Bonacina

Visiting: Computer Science Laboratory, SRI International, Menlo Park, CA, USA

Affiliation: Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy, EU

May 26, 2016

Outline

Automated reasoning
Some building blocks for reasoning
The theorem-proving problem
Ordering-based inference mechanisms
Theorem-proving strategies

Automated reasoning

Some building blocks for reasoning

The theorem-proving problem

Ordering-based inference mechanisms

Theorem-proving strategies

Automated reasoning

Automated reasoning is

- ▶ Symbolic computation
- ▶ Artificial intelligence
- ▶ Computational logic
- ▶ ...

- ▶ Knowledge **described precisely**: **symbols**
- ▶ **Symbolic** reasoning: Logico-deductive, Probabilistic ...

The gist of this lecture

- ▶ Logico-deductive reasoning
- ▶ Focus: first-order logic (FOL)
- ▶ Take-home message:
 - ▶ FOL as **machine** language
 - ▶ Reasoning is about ignoring what's **redundant** as much as it is getting what's relevant
 - ▶ Orderings and ordering-based strategies
 - ▶ **Expansion** and **Contraction**
 - ▶ **Inference** and **Search**
 - ▶ Algorithmic building blocks

Signature

- ▶ A finite set of constant symbols: e.g., $a, b, c \dots$
- ▶ A finite set of function symbols: e.g., $f, g, h \dots$
- ▶ A finite set of predicate symbols: $P, Q, \simeq \dots$
- ▶ Arities
- ▶ Sorts (important but key concepts can be understood without)

An infinite supply of variable symbols: $x, y, z, w \dots$

Defined symbols and free symbols

- ▶ A symbol is **defined** if it comes with axioms, e.g., \simeq
- ▶ It is **free** otherwise, e.g., P
- ▶ Aka: **interpreted/uninterpreted**
- ▶ Equality (\simeq) comes with the **congruence axioms**

Terms and atoms

- ▶ Terms: $a, x, f(a, b), g(y)$
- ▶ **Herbrand universe** \mathcal{U} : all ground terms
(add a constant if there is none in the given signature)
- ▶ Atoms: $P(a), f(x, x) \simeq x$
- ▶ Literals: $P(a), f(x, x) \simeq x, \neg P(a), f(x, x) \not\simeq x$
- ▶ **Herbrand base** \mathcal{B} : all ground atoms
- ▶ If there is at least one function symbol, \mathcal{U} and \mathcal{B} are **infinite**
- ▶ This is key if the reasoner builds **new** terms and atoms

Substitution

- ▶ A **substitution** is a function from variables to terms that is not identity on a finite set of variables
- ▶ $\sigma = \{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\}$
- ▶ $\sigma = \{x \leftarrow a, y \leftarrow f(z), z \leftarrow w\}$
- ▶ Application: $h(x, y, z)\sigma = h(a, f(z), w)$

Idempotent substitution

- ▶ A substitution σ is **idempotent** if $t\sigma\sigma = \sigma$ for all t
- ▶ $\sigma = \{x \leftarrow a, y \leftarrow f(z), z \leftarrow w\}$ is not idempotent:
 - ▶ $h(x, y, z)\sigma = h(a, f(z), w)$
 - ▶ $h(x, y, z)\sigma\sigma = h(a, f(w), w)$
- ▶ $\sigma = \{x \leftarrow a, y \leftarrow f(w), z \leftarrow w\}$ is idempotent
- ▶ We are interested only in idempotent substitutions

Matching

- ▶ Given terms or atoms s and t
- ▶ $f(x, g(y))$ and $f(g(b), g(a))$
- ▶ Find **matching substitution**: σ s.t. $s\sigma = t$
 $\sigma = \{x \leftarrow g(b), y \leftarrow a\}$
- ▶ $s\sigma = t$: t is **instance** of s , s is **more general** than t

Unification

- ▶ Given terms or atoms s and t
- ▶ $f(g(z), g(y))$ and $f(x, g(a))$
- ▶ Find substitution σ s.t. $s\sigma = t\sigma$:
 $\sigma = \{x \leftarrow g(z), y \leftarrow a\}$
- ▶ **Unification problem:** $E = \{s_i =? t_i\}_{i=1}^n$
- ▶ **Most general unifier:** e.g., not
 $\sigma' = \{x \leftarrow g(b), y \leftarrow a, z \leftarrow b\}$

Tree-solved form

- ▶ $E = \{s_i =^? t_i\}_{i=1}^n$ is in **tree-solved form** if
 - ▶ All the s_i 's are variables
 - ▶ For all i, j , $1 \leq i \neq j \leq n$, $s_i \neq s_j$
 - ▶ For all i, j , $1 \leq i, j \leq n$, s_i does not occur in t_j
- ▶ $\sigma = \{s_i \leftarrow t_i\}_{i=1}^n$ is an **idempotent substitution**

Robinson's unification algorithm

- ▶ Transform $E = \{s_i =? t_i\}_{i=1}^n$ in tree-solved form
- ▶ Exponential in the worst case
- ▶ It works well in practice
- ▶ Used by most reasoners

(Early version already in Herbrand's thesis)

Orderings

- ▶ View \mathcal{U} and \mathcal{B} as **ordered sets**
- ▶ With variables: **partial** order
- ▶ Extend to literals (add sign) and clauses
- ▶ Extend to proofs (e.g., equational chains)
- ▶ Why? To detect and delete or replace **redundant** data
- ▶ E.g., replace something by something **smaller** in a well-founded ordering

Precedence

- ▶ A **partial** order $>$ on the signature
- ▶ Example: the Ackermann function
 - ▶ $ack(0, y) \simeq succ(y)$
 - ▶ $ack(succ(x), 0) \simeq ack(x, succ(0))$
 - ▶ $ack(succ(x), succ(y)) \simeq ack(x, ack(succ(x), y))$
- ▶ Precedence $ack > succ > 0$

Stability

- ▶ \succ ordering
- ▶ $s \succ t$
- ▶ $f(f(x, y), z) \succ f(x, f(y, z))$
- ▶ **Stability:** $s\sigma \succ t\sigma$ for all substitutions σ
- ▶ $f(f(g(a), b), z) \succ f(g(a), f(b, z))$
 $\sigma = \{x \leftarrow g(a), y \leftarrow b\}$

Monotonicity

- ▶ \succ ordering
- ▶ $s \succ t$
- ▶ Example: $f(x, i(x)) \succ e$
- ▶ **Monotonicity**: $r[s] \succ r[t]$ for all contexts r
(A context is an expression, here a term or atom, with a hole)
- ▶ $f(f(x, i(x)), y) \succ f(e, y)$

Subterm property

- ▶ \succ ordering
- ▶ $s[t] \succ t$
- ▶ Example: $f(x, i(x)) \succ i(x)$

Simplification ordering

- ▶ Stable, monotonic, and with the subterm property:
simplification ordering
- ▶ A simplification ordering is **well-founded**
- ▶ No infinite descending chain $s_0 \succ s_1 \succ \dots s_i \succ s_{i+1} \succ \dots$

Multiset extension

- ▶ **Multisets**, e.g., $\{a, a, b\}$, $\{5, 4, 4, 4, 3, 1, 1\}$
- ▶ From \succ to \succ_{mul} :
 - ▶ $M \succ_{mul} \emptyset$
 - ▶ $M \cup \{a\} \succ_{mul} N \cup \{a\}$ if $M \succ_{mul} N$
 - ▶ $M \cup \{a\} \succ_{mul} N \cup \{b\}$ if $a \succ b$ and $M \cup \{a\} \succ_{mul} N$
- ▶ $\{5\} \succ_{mul} \{4, 4, 4, 3, 1, 1\}$
- ▶ If \succ is well-founded then \succ_{mul} is well-founded

Recursive path ordering (RPO)

$s = f(s_1, \dots, s_n) \succ g(t_1, \dots, t_m) = t$ if

- ▶ Either $f > g$ and $\forall k, 1 \leq k \leq m, s \succ t_k$
- ▶ Or $f = g$ and $\{s_1, \dots, s_n\} \succ_{mul} \{t_1, \dots, t_n\}$
- ▶ Or $\exists k$ such that $s_k \succeq t$

Distributivity by RPO

- ▶ Precedence: $*$ $>$ $+$
- ▶ $x * (y + z) \succ x * y + x * z$ because
 - ▶ $*$ $>$ $+$ and
 - ▶ $x * (y + z) \succ x * y$ since $\{x, y + z\} \succ_{mul} \{x, y\}$
 - ▶ $x * (y + z) \succ x * z$ since $\{x, y + z\} \succ_{mul} \{x, z\}$

Lexicographic extension

- ▶ **Tuples, vectors, words**, e.g., (a, a, b) , $(5, 4, 4, 4, 3, 1, 1)$
- ▶ From \succ to \succ_{lex} :
 $(a_1, \dots, a_n) \succ_{lex} (b_1, \dots, b_m)$ if $\exists i$ s.t. $\forall j, 1 \leq j < i, a_j = b_j$,
and $a_i \succ b_i$
- ▶ $(5) \succ_{lex} (4, 4, 4, 3, 1, 1)$
- ▶ $(1, 2, 3, 5, 1) \succ_{lex} (1, 2, 3, 3, 4)$
- ▶ If \succ is well-founded then \succ_{lex} is well-founded

Lexicographic path ordering (LPO)

$s = f(s_1, \dots, s_n) \succ g(t_1, \dots, t_m) = t$ if

- ▶ Either $f > g$ and $\forall k, 1 \leq k \leq m, s \succ t_k$
- ▶ Or $f = g, (s_1, \dots, s_n) \succ_{lex} (t_1, \dots, t_n)$,
and $\forall k, i < k \leq n, s \succ t_k$
- ▶ Or $\exists k$ such that $s_k \succeq t$

Multiset and lexicographic extension can be mixed: give each function symbol either multiset or lexicographic status

Ackermann function by LPO

- ▶ Precedence $ack > succ > 0$
- ▶ $ack(0, y) \succ succ(y)$
because $ack > succ$ and $ack(0, y) \succ y$
- ▶ $ack(succ(x), 0) \succ ack(x, succ(0))$
because $(succ(x), 0) \succ_{lex} (x, succ(0))$,
as $succ(x) \succ x$, and $ack(succ(x), 0) \succ succ(0)$,
since $ack > succ$ and $ack(succ(x), 0) \succ 0$
- ▶ $ack(succ(x), succ(y)) \succ ack(x, ack(succ(x), y))$
because $(succ(x), succ(y)) \succ_{lex} (x, ack(succ(x), y))$,
since $succ(x) \succ x$ and $ack(succ(x), succ(y)) \succ ack(succ(x), y)$,
because $(succ(x), succ(y)) \succ_{lex} (succ(x), y)$,
as $succ(x) = succ(x)$ and $succ(y) \succ y$

Ordering atoms and literals

- ▶ Atoms are treated like terms
- ▶ Also predicate symbols in the precedence $>$
- ▶ \simeq is typically the smallest predicate symbol in $>$
- ▶ \simeq has multiset status: $s \simeq t$ as $\{s, t\}$
- ▶ Literals: make the positive version smaller than the negative
- ▶ Add \top and \perp both $>$ -smaller than any other symbol and with $\perp > \top$
- ▶ For literal L take multiset $\{atom(L), \perp\}$ if L negative, $\{atom(L), \top\}$, otherwise

Variables cause partiality

- ▶ Let s and t be two distinct non-ground terms or atoms
- ▶ If $\exists x \in \text{Var}(s) \setminus \text{Var}(t)$ then $t \not\prec s$
- ▶ $g(x) \not\prec f(x, y)$
- ▶ If $\exists y \in \text{Var}(t) \setminus \text{Var}(s)$ then $s \not\prec t$
- ▶ Both: $t \# s$ (uncomparable)
- ▶ $f(x) \# g(y)$, $f(x) \# f(y)$, $g(x, z) \# f(x, y)$

Transfiniteness

If there is more than one function symbol, these orderings are not order-isomorphic to ω since, e.g., $\{f^n(a)\}_{n \geq 0}$ alone is

Complete simplification ordering (CSO)

- ▶ LPO and RPO are simplification orderings
- ▶ Simplification ordering **total** on **ground** terms and atoms:
complete simplification ordering (CSO)
- ▶ LPO and RPO with a total precedence are CSO
- ▶ LPO and RPO do not correlate with size
e.g., $f(a) \succ g^5(a)$ if $f > g$
- ▶ **Knuth-Bendix ordering** (KBO): based on precedence and a **weight** function

Summary of the first part

- ▶ Language: signature, terms, atoms, literals
- ▶ Substitutions instantiate variables
- ▶ Matching and unification
- ▶ A partially ordered world of terms, atoms, literals
- ▶ More building blocks: **indexing** to detect matching and unification fast

At the dawn of computer science

- ▶ Kurt Gödel: **completeness of first-order logic**
Later: Leon Henkin (consistency implies satisfiability)
- ▶ Alan Turing: **Entscheidungsproblem**; computer; Turing machine; universal computer; halting problem; undecidability; **undecidability of first-order logic**
- ▶ Herbrand theorem: **semi-decidability of first-order logic**

Herbrand theorem: Jacques Herbrand + Thoralf Skolem + Kurt Gödel

The theorem-proving problem

- ▶ A set H of formulas viewed as **assumptions** or **hypotheses**
- ▶ A formula φ viewed as **conjecture**
- ▶ Theorem-proving problem: $H \models? \varphi$
- ▶ Equivalently: is $H \cup \{\neg\varphi\}$ unsatisfiable?
- ▶ If $H \models \varphi$, then φ is a **theorem** of H , or $H \supset \varphi$ is a **theorem**
- ▶ $Th(H) = \{\varphi : H \models \varphi\}$
- ▶ Infinitely many interpretations on infinitely many domains:
how do we start?

Two simplifications

- ▶ Restrict formulas to **clauses**: less expressive, but suitable as **machine language**
- ▶ Restrict interpretations to **Herbrand interpretations**: a semantics built out of syntax
- ▶ All we have in machine's memory are symbols, that is, syntax

Clausal form

- ▶ **Clause**: disjunction of literals where all variables are implicitly universally quantified
- ▶ $\neg P(f(z)) \vee \neg Q(g(z)) \vee R(f(z), g(z))$
- ▶ Ordering \succ on literals extended to clauses by multiset extension
- ▶ No loss of generality: every formula can be transformed into a set of clauses
- ▶ Every clause has its own variables

Transformation into clausal form

- ▶ Eliminate \equiv and \supset : $F \equiv G$ becomes $(F \supset G) \wedge (G \supset F)$ and $F \supset G$ becomes $\neg F \vee G$
- ▶ Reduce the scope of all occurrences of \neg to an atom:
 $(\neg(F \vee G))$ becomes $\neg F \wedge \neg G$, $\neg(F \wedge G)$ becomes $\neg F \vee \neg G$, $\neg\neg F$ becomes F , $\neg\exists F$ becomes $\forall\neg F$, and $\neg\forall F$ becomes $\exists\neg F$
- ▶ Standardize variables apart
(each quantifier occurrence binds a distinct variable symbol)
- ▶ Skolemize \exists and then drop \forall
- ▶ Distributivity and associativity: $F \vee (G \wedge H)$ becomes $(F \vee G) \wedge (F \vee H)$ and $F \vee (G \vee H)$ becomes $F \vee G \vee H$
- ▶ Replace \wedge by comma and get a **set of clauses**

Skolemization

- ▶ Outermost \exists :
 - ▶ $\exists x F[x]$ becomes $F[a]$ (all occurrences of x replaced by a)
 a is a **new Skolem constant**
 - ▶ There exists an element such that F : let this element be named a
- ▶ \exists in the scope of \forall :
 - ▶ $\forall y \exists x F[x, y]$ becomes $\forall y F[g(y), y]$
(all occurrences of x replaced by $g(y)$)
 g is a **new Skolem function**
 - ▶ For all y there is an x such that F : x depends on y ;
let g be the map of this dependence

A simple example

- ▶ $\neg\{[\forall x P(x)] \supset [\exists y \forall z Q(y, z)]\}$
- ▶ $\neg\{\neg[\forall x P(x)] \vee [\exists y \forall z Q(y, z)]\}$
- ▶ $[\forall x P(x)] \wedge \neg[\exists y \forall z Q(y, z)]$
- ▶ $[\forall x P(x)] \wedge [\forall y \exists z \neg Q(y, z)]$
- ▶ $[\forall x P(x)] \wedge [\forall y \neg Q(y, f(y))]$ where f is a Skolem function
- ▶ $\{P(x), \neg Q(y, f(y))\}$: a set of two unit clauses

Clausal form and Skolemization

- ▶ All steps in the transformation into clauses except Skolemization preserve **logical equivalence**
(for every interpretation, F is true iff F' is true)
- ▶ Skolemization only preserves **equisatisfiability**
(F is (un)satisfiable iff F' is (un)satisfiable)
- ▶ Why Skolem symbols must be **new**?
So that we can interpret them as in the model of F when building a model of F'

Herbrand interpretations

- ▶ First-order interpretation $\mathcal{I} = \langle \mathcal{D}, \Phi \rangle$
- ▶ Let \mathcal{D} be \mathcal{U}
- ▶ Let Φ interpret constant and function symbols as themselves:
 - ▶ $\Phi(a) = a$
 - ▶ $\Phi(f)(t_1, \dots, t_n) = f(t_1, \dots, t_n)$
- ▶ Predicate symbols? All possibilities
- ▶ The powerset $\mathcal{P}(\mathcal{B})$ gives all possible Herbrand interpretations
- ▶ Herbrand model: a satisfying Herbrand interpretation

Clausal form and Herbrand interpretations

- ▶ Theorem-proving problem: is $H \cup \{\neg\varphi\}$ unsatisfiable?
- ▶ Transform $H \cup \{\neg\varphi\}$ into set S of clauses
- ▶ $H \cup \{\neg\varphi\}$ and S are equisatisfiable
- ▶ Theorem-proving problem: is S unsatisfiable?
- ▶ S is unsatisfiable iff S has no Herbrand model
- ▶ From now on: only Herbrand interpretations

Not for formulas

- ▶ $\exists x P(x) \wedge \neg P(a)$
- ▶ Is it satisfiable? Yes
- ▶ Herbrand model? No!
- ▶ \emptyset and $\{P(a)\}$ or $\{\neg P(a)\}$ and $\{P(a)\}$
- ▶ Clausal form: $\{P(b), \neg P(a)\}$
- ▶ Herbrand model: $\{P(b)\}$ or $\{P(b), \neg P(a)\}$

Satisfaction

- ▶ \mathcal{I} : Herbrand interpretation
- ▶ $\mathcal{I} \models S$ if $\mathcal{I} \models C$ for all $C \in S$
- ▶ $\mathcal{I} \models C$ if $\mathcal{I} \models C\sigma$ for all ground instances $C\sigma$ of C
- ▶ $\mathcal{I} \models C\sigma$ if $\mathcal{I} \models L\sigma$ for some ground literal $L\sigma$ in $C\sigma$

Herbrand theorem

- ▶ S : set of clauses
- ▶ S is unsatisfiable iff there exists a **finite** set S' of **ground** instances of clauses in S such that S' is unsatisfiable
- ▶ Finite sets of ground instances can be enumerated and tested for propositional satisfiability which is decidable: the first-order theorem-proving problem is **semi-decidable**

Equality

- ▶ Congruence axioms in clausal form:
 - ▶ $x \simeq x$
 - ▶ $x \not\simeq y \vee y \simeq x$
 - ▶ $x \not\simeq y \vee y \not\simeq z \vee x \simeq z$
 - ▶ $x \not\simeq y \vee f(\dots, x, \dots) \simeq f(\dots, y, \dots)$
 - ▶ $x \not\simeq y \vee \neg P(\dots, x, \dots) \vee P(\dots, y, \dots)$
- ▶ E -satisfiability, E -interpretations, Herbrand E -interpretations

Herbrand theorem

- ▶ S : set of clauses
- ▶ S is E -unsatisfiable iff there exists a **finite** set S' of **ground** instances of clauses in S such that S' is E -unsatisfiable

Summary of the second part

- ▶ First-order theorem-proving problem
- ▶ Clauses
- ▶ Herbrand interpretations
- ▶ Herbrand theorem
- ▶ Theorem proving in first-order logic is **semi-decidable**
- ▶ Design **theorem-proving strategies** that are **semi-decision procedures** and implement Herbrand theorem

Expansion and contraction

Like many search procedures, most reasoning methods combine various forms of **growing** and **shrinking**:

- ▶ Recall CDCL in SAT/SMT: decisions and propagations grow the model while backjumps shrink it
- ▶ **Ordering-based strategies**: **expansion** and **contraction** of a set of clauses
- ▶ Ordering \succ on clauses extended to sets of clauses by multiset extension

Expansion

An inference

$$\frac{A}{B}$$

where A and B are sets of clauses is an **expansion** inference if

- ▶ $A \subset B$: something is added
- ▶ Hence $A \prec B$ and
- ▶ $B \setminus A \subseteq Th(A)$ hence $B \subseteq Th(A)$ hence $Th(B) \subseteq Th(A)$
(**soundness**)

Contraction

An inference

$$\frac{A}{B}$$

where A and B are sets of clauses is a **contraction** inference if

- ▶ $A \not\subseteq B$: something is deleted or replaced, and
- ▶ $B \prec A$: if replaced, replaced by something smaller, and
- ▶ $A \setminus B \subseteq Th(B)$ hence $A \subseteq Th(B)$ hence $Th(A) \subseteq Th(B)$
(**monotonicity** or **adequacy** or **soundness of contraction**)

Propositional resolution

$$\frac{P \vee \neg Q \vee \neg R, \neg P \vee O}{O \vee \neg Q \vee \neg R}$$

where O , P , Q , and R are propositional atoms
(aka propositional variables, aka 0-ary predicates)

Propositional resolution

is an expansion inference rule:

$$\frac{S \cup \{P \vee \neg Q \vee \neg R, \neg P \vee O\}}{S \cup \{P \vee \neg Q \vee \neg R, \neg P \vee O, O \vee \neg Q \vee \neg R\}}$$

where S is a set of clauses

Propositional resolution

$$\frac{S \cup \{L \vee C, \neg L \vee D\}}{S \cup \{L \vee C, \neg L \vee D, C \vee D\}}$$

- ▶ L is an atom
- ▶ C and D are disjunctions of literals
- ▶ L and $\neg L$ are the **literals resolved upon**
- ▶ $C \vee D$ is called **resolvent**

First-order resolution

$$\frac{SU\{L_1 \vee C, \neg L_2 \vee D\}}{SU\{L_1 \vee C, \neg L_2 \vee D, (C \vee D)\sigma\}}$$

where $L_1\sigma = L_2\sigma$ for σ most general unifier

First-order resolution

$$\frac{P(g(z), g(y)) \vee \neg R(z, y), \neg P(x, g(a)) \vee Q(x, g(x))}{\neg R(z, a) \vee Q(g(z), g(g(z)))}$$

where $\sigma = \{x \leftarrow g(z), y \leftarrow a\}$

Ordered resolution

$$\frac{S \cup \{L_1 \vee C, \neg L_2 \vee D\}}{S \cup \{L_1 \vee C, \neg L_2 \vee D, (C \vee D)\sigma\}}$$

where

- ▶ $L_1\sigma = L_2\sigma$ for σ most general unifier
- ▶ $L_1\sigma \not\leq M\sigma$ for all $M \in C$
- ▶ $\neg L_2\sigma \not\leq M\sigma$ for all $M \in D$

Ordered resolution

$$\frac{P(g(z), g(y)) \vee \neg R(z, y), \neg P(x, g(a)) \vee Q(x, g(x))}{\neg R(z, a) \vee Q(g(z), g(g(z)))}$$

- ▶ $\sigma = \{x \leftarrow g(z), y \leftarrow a\}$
- ▶ $P(g(z), g(a)) \not\leq \neg R(z, a)$
- ▶ $\neg P(g(z), g(a)) \not\leq Q(g(z), g(g(z)))$
- ▶ Allowed, e.g., with $P > R > Q > g$
- ▶ Not allowed, e.g., with $Q > R > P > g > a$

Subsumption

$$\frac{S \cup \{P(x, y) \vee Q(z), Q(a) \vee P(b, b) \vee R(u)\}}{S \cup \{P(x, y) \vee Q(z)\}}$$

because $C = P(x, y) \vee Q(z)$ **subsumes** $D = Q(a) \vee P(b, b) \vee R(u)$,
as there is a substitution $\sigma = \{z \leftarrow a, x \leftarrow b, y \leftarrow b\}$ such that
 $C\sigma \subset D$ which means $\{C\} \models \{D\}$ (**adequacy**)

Subsumption ordering

- ▶ **Subsumption ordering**: $C \leq D$ if $\exists \sigma \ C \sigma \subseteq D$ (as multisets)
- ▶ **Strict** subsumption ordering: $C \prec D$ if $C \leq D$ and $C \not\leq D$
- ▶ The strict subsumption ordering \prec is **well-founded**
- ▶ Equality up to variable renaming: $C \doteq D$ if $C \leq D$ and $C \leq D$
(C and D are **variants**)

Subsumption

$$\frac{S \cup \{C, D\}}{S \cup \{C\}}$$

- ▶ Either $C \prec D$ (**strict subsumption**)
- ▶ Or $C \doteq D$ and $C \prec D$ where \prec is the lexicographic combination of \prec and another well-founded ordering (e.g., C was generated before D) (**subsumption of variants**)
- ▶ Clause D is **redundant**
- ▶ Subsumption uses matching, resolution uses unification

And equality?

Replacing equals by equals as in **ground rewriting**:

$$\frac{S \cup \{f(a, a) \simeq a, P(f(a, a)) \vee Q(a)\}}{S \cup \{f(a, a) \simeq a, P(a) \vee Q(a)\}}$$

It can be done as $f(a, a) \succ a$

Simplification

is a contraction inference rule:

$$\frac{S \cup \{f(x, x) \simeq x, P(f(a, a)) \vee Q(a)\}}{S \cup \{f(x, x) \simeq x, P(a) \vee Q(a)\}}$$

- ▶ $f(x, x)$ matches $f(a, a)$ with $\sigma = \{x \leftarrow a\}$
- ▶ $f(a, a) \succ a$

Simplification

$$\frac{S \cup \{s \simeq t, L[r] \vee C\}}{S \cup \{s \simeq t, L[t\sigma] \vee C\}}$$

- ▶ L is a literal with r as subterm (L could be another equation)
- ▶ C is a disjunction of literals
- ▶ $\exists \sigma$ such that $s\sigma = r$ and $s\sigma \succ t\sigma$
- ▶ Clause $L[r] \vee C$ is entailed by the resulting set (**adequacy**)
- ▶ Clause $L[r] \vee C$ is **redundant**

Expansion for equality reasoning

- ▶ Simplification is a powerful rule that often does most of the work in presence of equality
- ▶ But it is not enough
- ▶ Equality reasoning requires to generate **new** equations
- ▶ We need an expansion rule that **builds equality into resolution** and uses unification not only matching

Superposition/Paramodulation

$$\frac{f(z, e) \simeq z, f(l(x, y), y) \simeq x}{l(x, e) \simeq x}$$

- ▶ $f(z, e)\sigma = f(l(x, y), y)\sigma$
- ▶ $\sigma = \{z \leftarrow l(x, e), y \leftarrow e\}$ most general unifier
- ▶ $f(l(x, e), e) \succ l(x, e)$
- ▶ $f(l(x, e), e) \succ x$
- ▶ Superposing two equations yields a **peak**:
 $l(x, e) \leftarrow f(l(x, e), e) \rightarrow x$

Completion

- ▶ New equations closing such peaks are called **critical pairs**, as they complete the set of equations into a **confluent** one
- ▶ Confluence ensures uniqueness of **normal forms**
- ▶ This procedure is known as **Knuth-Bendix completion**
- ▶ Unfailing or Ordered Knuth-Bendix completion ensures ground confluence (unique normal form of ground terms) which suffices for theorem proving in equational theories as the Skolemized form of $\neg(\forall \bar{x} s \simeq t)$ is ground

Superposition/Paramodulation

$$\frac{S \cup \{l \simeq r, p[s] \simeq q\}}{S \cup \{l \simeq r, p[s] \simeq q, (p[r] \simeq q)\sigma\}}$$

- ▶ s is not a variable
- ▶ $l\sigma = s\sigma$ most general unifier
- ▶ $l\sigma \not\leq r\sigma$
- ▶ $p\sigma \not\leq q\sigma$

Superposition/Paramodulation

$$\frac{S \cup \{l \simeq r, p[s] \bowtie q\}}{S \cup \{l \simeq r, p[s] \bowtie q, (p[r] \bowtie q)\sigma\}}$$

- ▶ \bowtie is either \simeq or \neq
- ▶ s is not a variable
- ▶ $l\sigma = s\sigma$ most general unifier
- ▶ $l\sigma \not\approx r\sigma$ and $p\sigma \not\approx q\sigma$

Superposition/Paramodulation

$$\frac{S \cup \{l \simeq r \vee C, p[s] \bowtie q \vee D\}}{S \cup \{l \simeq r \vee C, p[s] \bowtie q \vee D, (p[r] \bowtie q \vee C \vee D)\sigma\}}$$

- ▶ C and D are disjunctions of literals
- ▶ \bowtie is either \simeq or $\not\simeq$
- ▶ s is not a variable
- ▶ $l\sigma = s\sigma$ most general unifier
- ▶ $l\sigma \not\simeq r\sigma$ and $p\sigma \not\simeq q\sigma$
- ▶ $(l \simeq r)\sigma \not\simeq M\sigma$ for all $M \in C$
- ▶ $(p[s] \bowtie q)\sigma \not\simeq M\sigma$ for all $M \in D$

Superposition/Paramodulation

$$\frac{S \cup \{I \simeq r \vee C, L[s] \vee D\}}{S \cup \{I \simeq r \vee C, L[s] \vee D, (L[r] \vee C \vee D)\sigma\}}$$

- ▶ C and D are disjunctions of literals
- ▶ L is any literal, either equational or not, called **literal paramodulated into**
- ▶ s is not a variable
- ▶ $l\sigma = s\sigma$ most general unifier
- ▶ $l\sigma \not\approx r\sigma$
- ▶ $(I \simeq r)\sigma \not\approx M\sigma$ for all $M \in C$
- ▶ $L\sigma \not\approx M\sigma$ for all $M \in D$

What's in a name

- ▶ Paramodulation was used first in resolution-based theorem proving where simplification was called demodulation
- ▶ Superposition and simplification, or rewriting, were used first in Knuth-Bendix completion
- ▶ Some authors use superposition between unit equations and paramodulation otherwise
- ▶ Other authors use superposition when the literal paramodulated into is an equational literal and paramodulation otherwise

Derivation

- ▶ Input set S
- ▶ **Inference system** \mathcal{IS} : a set of inference rules
- ▶ **\mathcal{IS} -derivation** from S :

$$S_0 \underset{\mathcal{IS}}{\vdash} S_1 \underset{\mathcal{IS}}{\vdash} \dots S_i \underset{\mathcal{IS}}{\vdash} S_{i+1} \underset{\mathcal{IS}}{\vdash} \dots$$

where $S_0 = S$ and for all i , S_{i+1} is derived from S_i by an inference rule in \mathcal{IS}

- ▶ **Refutation**: a derivation such that $\square \in S_k$ for some k

Refutational completeness

An inference system \mathcal{IS} is **refutationally complete** if for all sets S of clauses, if S is unsatisfiable, there exists an \mathcal{IS} -derivation from S that is a refutation.

Refutational completeness

An inference system with

- ▶ Expansion rules: resolution, factoring, superposition/paramodulation, equational factoring, reflection (resolution with $x \simeq x$)
- ▶ Contraction rules: subsumption, simplification, tautology deletion, clausal simplification (unit resolution + subsumption)

is **refutationally complete**

Summary of the third part

- ▶ Expansion and contraction
- ▶ Resolution and subsumption
- ▶ Paramodulation/superposition and simplification
- ▶ Contraction uses matching, expansion uses unification
- ▶ Inference system
- ▶ Derivation
- ▶ Refutational completeness

Search

- ▶ Given S and IS , many IS -derivations from S are possible
- ▶ An inference system is **non-deterministic**
- ▶ Which one to build? **Search problem**
- ▶ Search space
- ▶ Rules and moves: inference rules and inference steps

Strategy

- ▶ Theorem-proving strategy: $\mathcal{C} = \langle \mathcal{IS}, \Sigma \rangle$
- ▶ \mathcal{IS} : inference system
- ▶ Σ : search plan
- ▶ The search plan picks at every stage of the derivation which inference to do next
- ▶ A deterministic proof procedure

Completeness

- ▶ Inference system: **refutational completeness**
there exist refutations
- ▶ Search plan: **fairness**
ensure that the generated derivation is a refutation
- ▶ Refutationally complete inference system + fair search plan =
complete theorem-proving strategy

Fairness

- ▶ **Fairness**: consider **eventually** all **needed** steps: What is needed?
- ▶ Dually: what is **not needed**, or: what is **redundant**?
- ▶ Fairness and redundancy are related

Redundancy

- ▶ Based on ordering \succ on clauses:
a clause is **redundant** if all its ground instances are;
a ground clause is **redundant** if there are ground instances of other clauses that entail it and are smaller
- ▶ Based on ordering \succ on proofs:
a clause is **redundant** if adding it does not decrease any minimal proofs (dually, removing it does not increase proofs)
- ▶ Agree if proofs are measured by maximal premises
- ▶ Redundant inference: uses/generates redundant clause

Fairness

- ▶ A derivation is **fair** if whenever a minimal proof of the target theorem is reducible by inferences, it is reduced eventually
- ▶ A derivation is **uniformly fair** if all non-redundant inferences are done eventually
- ▶ A search plan is **(uniformly) fair** if all its derivations are

Contraction first

Eager-contraction search plan: schedule contraction **before** expansion

The given-clause algorithm

- ▶ Two lists: *ToBeSelected* and *AlreadySelected*
(Other names: *SOS* and *Usable*; *Active* and *Passive*)
- ▶ Initialization: $ToBeSelected = S_0$ and $AlreadySelected = \emptyset$
- ▶ Alternative: $ToBeSelected = clauses(\neg\varphi)$ and
 $AlreadySelected = clauses(H)$ (set of support strategy)

The given-clause algorithm: expansion

- ▶ Loop until either proof found or $ToBeSelected = \emptyset$, the latter meaning satisfiable
- ▶ At every iteration: pick a **given-clause** from $ToBeSelected$
- ▶ How? **Best-first search**: the best according to an **evaluation function** (e.g., weight, FIFO, pick-given ratio)
- ▶ Perform all expansion steps with the given-clause and clauses in $AlreadySelected$ as premises
- ▶ Move the given-clause from $ToBeSelected$ to $AlreadySelected$
- ▶ Insert all newly generated clauses in $ToBeSelected$

Forward contraction

- ▶ **Forward contraction**: contract newly generated clauses by pre-existing ones
- ▶ Forward contract each new clause prior to insertion in *ToBeSelected*
- ▶ A very high number of clauses gets deleted typically by forward contraction

Backward contraction

- ▶ **Backward contraction**: contract pre-existing clauses by new ones
- ▶ For fairness backward contraction must be applied **after** forward contraction (e.g., subsumption)
- ▶ Detect which clauses can be backward-contracted and treat them as new
- ▶ Every backward-contracted clause may backward-contract others
- ▶ How much to do? How often?

A choice of two invariants

- ▶ Keep $ToBeSelected \cup AlreadySelected$ **contracted**
(Otter version of the given-clause algorithm)
- ▶ Keep only $AlreadySelected$ **contracted**
(Discount version of the given-clause algorithm)
 - ▶ Backward-contract $\{given-clause\} \cup AlreadySelected$ right after picking the given-clause
 - ▶ Deletion of “orphans” in $ToBeSelected$

More issues

- ▶ Options (binary flags) and parameters (numeric values)
- ▶ Proof reconstruction: **ancestor-graph** of \square
- ▶ Proof presentation

Interactivity

- ▶ Proof assistant \sim interpreter
- ▶ Theorem prover \sim compiler
 - ▶ Iterative experimentation with settings
 - ▶ Incomplete strategies
 - ▶ Auto mode

Some theorem provers

- ▶ [Otter](#), [EQP](#), and [Prover9](#) by the late Bill McCune
- ▶ [SNARK](#) by the late Mark E. Stickel
- ▶ [SPASS](#) by Christoph Weidenbach et al.
- ▶ [E](#) by Stephan Schulz
- ▶ [Vampire](#) by Andrei Voronkov et al.
- ▶ [Metis](#) by Joe Leslie-Hurd
- ▶ [MetiTarski](#) by Larry Paulson et al.

Some applications

- ▶ Analysis, verification, synthesis of systems, e.g.:
 - ▶ cryptographic protocols
 - ▶ message-passing systems
 - ▶ software specifications
 - ▶ theorem proving support to model checking
- ▶ Mathematics: proving non-trivial theorems in, e.g.,
 - ▶ Boolean algebras (e.g., the Robbins conjecture)
 - ▶ theories of rings (e.g., the Moufang identities), groups and quasigroups
 - ▶ many-valued logics (e.g., Lukasiewicz logic)

Some textbooks

- ▶ Chin-Liang Chang, Richard Char-Tung Lee. Symbolic Logic and Mechanical Theorem Proving. Computer Science Classics, Academic Press, 1973
- ▶ Alexander Leitsch. The Resolution Calculus. Texts in Theoretical Computer Science, An EATCS Series, Springer, 1997
- ▶ Rolf Socher-Ambrosius, Patricia Johann. Deduction Systems. Graduate Texts in Computer Science, Springer, 1997
- ▶ John Harrison. Handbook of Practical Logic and Automated Reasoning. Cambridge University Press, 2009

More textbooks

- ▶ Raymond M. Smullyan. First-order logic. Dover Publications 1995 (republication of the original published by Springer Verlag in 1968)
- ▶ Allan Ramsay. Formal Methods in Artificial Intelligence. Cambridge Tracts in Theoretical Computer Science 6, Cambridge University Press, 1989
- ▶ Ricardo Caferra, Alexander Leitsch, Nicolas Peltier. Automated Model Building. Applied Logic Series 31, Kluwer Academic Publishers, 2004
- ▶ Martin Davis. The Universal Computer. The Road from Leibniz to Turing. Turing Centenary Edition. Mathematics/Logic/Computing Series. CRC Press, Taylor and Francis Group, 2012

Some surveys

- ▶ Maria Paola Bonacina. A taxonomy of theorem-proving strategies. In Michael J. Wooldridge, Manuela Veloso (Eds.) Artificial Intelligence Today – Recent Trends and Developments, LNAI 1600:43–84, Springer, 1999 [providing 150 references]
- ▶ Maria Paola Bonacina. A taxonomy of parallel strategies for deduction. Annals of Mathematics and Artificial Intelligence 29(1/4):223–257, 2000 [providing 104 references]

More surveys

- ▶ Maria Paola Bonacina. On theorem proving for program checking – Historical perspective and recent developments. In Maribel Fernández (Ed.) Proceedings of the 12th International Symposium on Principles and Practice of Declarative Programming (PPDP), 1–11, ACM Press, 2010 [providing 119 references]
- ▶ Maria Paola Bonacina, Ulrich Furbach, Viorica Sofronie-Stokkermans. On first-order model-based reasoning. In Narciso Martí-Oliet, Peter Olveczky, Carolyn Talcott (Eds.) Logic, Rewriting, and Concurrency, LNCS 9200:181–204, Springer, 2015 [providing 88 references]

Some topics for further reading

- ▶ Strategies seeking proof/counter-model in one search:
model-based first-order reasoning
- ▶ Adding built-in theories
- ▶ Integration of theorem-proving strategies with SAT/SMT solvers
- ▶ Theorem-proving strategies as decision procedures
- ▶ Parallel/distributed theorem proving
- ▶ Goal-sensitive or target-oriented strategies
- ▶ Machine-independent evaluation of strategies: strategy analysis, search complexity

Selected papers

- ▶ Maria Paola Bonacina, David A. Plaisted. Semantically-guided goal-sensitive reasoning: model representation. *Journal of Automated Reasoning* 56(2):113–141, 2016 [providing 96 references]
- ▶ Maria Paola Bonacina, Christopher A. Lynch, Leonardo de Moura. On deciding satisfiability by theorem proving with speculative inferences. *Journal of Automated Reasoning* 47(2):161–189, 2011 [providing 65 references]
- ▶ Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, Stephan Schulz. New results on rewrite-based satisfiability procedures. *ACM Transactions on Computational Logic* 10(1):129–179, 2009 [providing 90 references]

Selected papers

- ▶ Maria Paola Bonacina, Nachum Dershowitz. Abstract canonical inference. *ACM Transactions on Computational Logic* 8(1):180–208, 2007 [providing 47 references]
- ▶ Maria Paola Bonacina and Jieh Hsiang. On the modelling of search in theorem proving – Towards a theory of strategy analysis. *Information and Computation*, 147:171–208, 1998 [providing 44 references]
- ▶ Maria Paola Bonacina and Jieh Hsiang. Towards a foundation of completion procedures as semidecision procedures. *Theoretical Computer Science*, 146:199–242, 1995 [providing 62 references]

Thanks

Thank you!