Sistemi di Calcolo per Logica del Primo Ordine Calculi for First-Order Logic

Wolfgang Faber

University of Calabria, Italy

2007

Outline

Methods

- Propositional Logic
  - Truth tables
  - DLL
  - Resolution
- Quantified Boolean Formulas
  - DLL Extensions
- First-Order Logic
  - Sequent Calculus
  - Resolution

Methods

- Propositional Logic
  - Truth tables
  - DLL
  - Resolution
- Quantified Boolean Formulas
  - DLL Extensions
- First-Order Logic
  - Sequent Calculus
  - Resolution

Methods

- Propositional Logic
  - Truth tables
  - DLL
  - Resolution
- Quantified Boolean Formulas
  - DLL Extensions
- First-Order Logic
  - Sequent Calculus
  - Resolution

Outline

Gerhard Gentzen

Gerhard Gentzen (1909–1945)

Sequent Calculus

- Idea: Define inference rules for sequents $\Gamma \vdash \Delta$.
- Γ and Δ are sequences of formulas
- Intuition: Read $\Gamma \vdash \Delta$ like $(\bigwedge \Gamma) \rightarrow (\bigvee \Gamma)$.
- Goal 1: $\Gamma \vdash \Delta$ holds if $\Gamma \models \Delta$ (completeness)
- Goal 2: If $\Gamma \vdash \Delta$ holds, then $\Gamma \models \Delta$ (soundness)
- Notation: $\dfrac{S_1 \quad \ldots \quad S_n}{S}$ means: From sequents $S_1 \ldots S_n$ we conclude sequent S.
- System considered here: LK

Sequent Calculus – Axioms

- Begin with axioms (true statements)
- $$\frac{}{f \vdash f}$$
- ...for any formula f

Sequent Calculus – Structural

- weakening left: $\dfrac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}$
- weakening right: $\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}$
- contraction left: $\dfrac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}$
- contraction right: $\dfrac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$
- permutation left: $\dfrac{\Gamma, A, B \vdash \Delta}{\Gamma, B, A \vdash \Delta}$
- permutation left: $\dfrac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2}$

Sequent Calculus – Conjunction

- $\wedge$ left 1: $\dfrac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}$
- $\wedge$ left 2: $\dfrac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}$
- $\wedge$ right: $\dfrac{\Gamma \vdash A, \Delta \qquad \Sigma \vdash B, \Pi}{\Gamma, \Sigma \vdash A \wedge B, \Delta, \Pi}$

Sequent Calculus – Disjunction

- $\vee$ right 1: $\dfrac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \vee B, \Delta}$
- $\vee$ right 2: $\dfrac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta}$
- $\vee$ left: $\dfrac{\Gamma, A \vdash \Delta \qquad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \vee B \vdash \Delta, \Pi}$

Sequent Calculus – Negation

- $\neg$ right: $\dfrac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$
- $\neg$ left: $\dfrac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}$

Sequent Calculus – Implication

- $\rightarrow$ right: $\dfrac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}$
- $\rightarrow$ left: $\dfrac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta}$

Sequent Calculus – Quantifiers

- $A[t]$ means that a term t occurs in A.
- $A[Y]$ means that a variable Y occurs in A, which is not free elsewhere (i.e. neither in Γ nor in Δ).
- $\forall$ right: $\dfrac{\Gamma \vdash A[Y], \Delta}{\Gamma \vdash \forall X\ A[X/Y], \Delta}$
- $\forall$ left: $\dfrac{\Gamma, A[t] \vdash \Delta}{\Gamma, \forall X\ A[X/t] \vdash \Delta}$
- $\exists$ right: $\dfrac{\Gamma \vdash A[t], \Delta}{\Gamma \vdash \exists X\ A[t/X], \Delta}$
- $\exists$ left: $\dfrac{\Gamma, A[Y] \vdash \Delta}{\Gamma, \exists X\ A[X/Y] \vdash \Delta}$

Sequent Calculus – Cut

- Cut – a special structural inference
- $$\frac{\Gamma \vdash A, \Delta \qquad \Sigma, A \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}$$

Sequent Calculus

- Use these inference rules consecutively.
- Example: $\dfrac{\overline{A \vdash A}}{\vdash \neg A, A}$
- If on top there are only axioms, then it is a derivation of the bottom sequent.

Sequent Calculus – Theorem

Theorem

Sequent Calculus is sound and complete. I.e. if we can derive $\Gamma \vdash \Delta$, then $\Gamma \models \Delta$, and if $\Gamma \models \Delta$ then there is a derivation for $\Gamma \vdash \Delta$.

Outline

Reminder — Propositional Resolution

- Input: Formulas in CNF $\rightarrow$ set of clauses
- Resolvents of two clauses
- Factorization of a clause (automatic for set representation)
- Derivations
- Refutations (derivations of empty clause $\square$)

Relationship

- Similar to DLL procedure!
- Similar to cut!
- Works on CNFs!

First Order Resolution

- Can we generalize propositional resolution to first-order formulas?
- Biggest obstacle: "Equality" of atoms to be resolved.
- $\forall X : (h(X) \rightarrow m(X)) \wedge h(socrates)$
- $\{\{\neg h(X) \vee m(X)\}, \{h(socrates)\}\}$
- $h(X) \neq h(socrates)$!
- For the special case $\{\{\neg h(socrates) \vee m(socrates)\}, \{h(socrates)\}\}$ it works.
- Formalize this idea!

First Order Resolution

- Can we generalize propositional resolution to first-order formulas?
- Biggest obstacle: "Equality" of atoms to be resolved.
- $\forall X : (h(X) \rightarrow m(X)) \wedge h(socrates)$
- $\{\{\neg h(X) \vee m(X)\}, \{h(socrates)\}\}$
- $h(X) \neq h(socrates)$!
- For the special case $\{\{\neg h(socrates) \vee m(socrates)\}, \{h(socrates)\}\}$ it works.
- Formalize this idea!

First Order Resolution

- Can we generalize propositional resolution to first-order formulas?
- Biggest obstacle: "Equality" of atoms to be resolved.
- $\forall X : (h(X) \rightarrow m(X)) \wedge h(socrates)$
- $\{\{\neg h(X) \vee m(X)\}, \{h(socrates)\}\}$
- $h(X) \neq h(socrates)$!
- For the special case $\{\{\neg h(socrates) \vee m(socrates)\}, \{h(socrates)\}\}$ it works.
- Formalize this idea!

First Order Resolution

- Can we generalize propositional resolution to first-order formulas?
- Biggest obstacle: "Equality" of atoms to be resolved.
- $\forall X : (h(X) \rightarrow m(X)) \land h(socrates)$
- $\{\{\neg h(X) \lor m(X)\}, \{h(socrates)\}\}$
- $h(X) \neq h(socrates)$!
- For the special case $\{\{\neg h(socrates) \lor m(socrates)\}, \{h(socrates)\}\}$ it works.
- Formalize this idea!

First Order Resolution

- Can we generalize propositional resolution to first-order formulas?
- Biggest obstacle: "Equality" of atoms to be resolved.
- $\forall X : (h(X) \rightarrow m(X)) \wedge h(socrates)$
- $\{\{\neg h(X) \vee m(X)\}, \{h(socrates)\}\}$
- $h(X) \neq h(socrates)$!
- For the special case $\{\{\neg h(socrates) \vee m(socrates)\}, \{h(socrates)\}\}$ it works.
- Formalize this idea!

Substitution

Definition

A substitution is a set of the form $\{t_1/X_1, \ldots, t_n/X_n\}$ where each X_i is a distinct (object) variable, and $X_i \neq t_i$ $(1 \leq i \leq n)$.

Usually denoted by lowercase greek letters $(\sigma, \vartheta, \rho)$.

Usually $\epsilon = \{\}$ is the empty substitution.

Substitution

Definition

A substitution is a set of the form $\{t_1/X_1, \ldots, t_n/X_n\}$ where each X_i is a distinct (object) variable, and $X_i \neq t_i$ $(1 \leq i \leq n)$.

Usually denoted by lowercase greek letters $(\sigma, \vartheta, \rho)$.

Usually $\epsilon = \{\}$ is the empty substitution.

Substitution

Definition

A substitution is a set of the form $\{t_1/X_1, \ldots, t_n/X_n\}$ where each X_i is a distinct (object) variable, and $X_i \neq t_i$ ($1 \leq i \leq n$).

Usually denoted by lowercase greek letters (σ, ϑ, ρ).
Usually $\epsilon = \{\}$ is the empty substitution.

Application of Substitutions

Definition

Let E be an atomic first-order formula (or other syntactic first-order structure) and $\sigma = \{t_1/X_1, \ldots, t_n/X_n\}$ be a substitution. Then $E\sigma$ is the application of σ on E, obtained by simultaneously replacing each variable X_i by t_i.

Composition of Substitutions

Definition

Let $\sigma = \{t_1/X_1, \ldots, t_n/X_n\}$ $\vartheta = \{u_1/Y_1, \ldots, u_m/Y_m\}$ be substitutions. The composition $\sigma \circ \vartheta$ (or simply $\sigma\vartheta$) is derived from $\{t_1\vartheta/X_1, \ldots, t_n\vartheta/X_n, u_1/Y_1, \ldots, u_m/Y_m\}$, where u_j/Y_j is omitted if $Y_j \in \{X_1, \ldots, X_n\}$, and $t_k\vartheta/X_k$ is omitted if $X_k = t_k\vartheta$.

Properties of Substitutions

- $\sigma \circ \epsilon = \epsilon \circ \sigma = \sigma$
- $(\sigma \circ \vartheta) \circ \rho = \sigma \circ (\vartheta \circ \rho)$
- $(E\sigma)\vartheta = E(\sigma \circ \vartheta) = E\sigma\vartheta$
- $\sigma \circ \vartheta \neq \vartheta \circ \sigma$

Unification

Definition

Let E_1,E_2 be atomic first-order formulas (or other syntactic first-order structures). A substitution σ is a unifier if $E_1\sigma = E_2\sigma$.

Most General Unifier

Definition

Let E_1,E_2 be atomic first-order formulas (or other syntactic first-order structures). A unifier σ is a most general unifier (mgu) if for any unifier ϑ of E_1, E_2 it holds that $\vartheta = \sigma \circ \rho$ for some substitution ρ.

Most General Unifier – Properties

- If E_1,E_2 are unifiable, an mgu exists.
- If E_1,E_2 are unifiable, the mgu is unique modulo variable renamings.

Algorithm: Disagreement Set

- D(E)
- Input: E set of formulas or terms
- Output: set of disagreeing terms
- return the set of terms (or formulas) at leftmost subexpressions on which expressions in E differ

Example: Disagreement Set

- $\mathrm{D}(\{p(X, f(a)), p(g(b), Y)\}) = \{X, g(b)\}$
- $\mathrm{D}(\{p(X, f(a)), q(g(b), Y)\}) = \{p(X, f(a)), q(g(b), Y)\}$
- $\mathrm{D}(\{p(g(b), f(a)), p(g(b), f(Y))\}) = \{a, Y\}$
- $\mathrm{D}(\{p(g(b), f(a, c)), p(g(b), f(Y, d))\}) = \{a, Y\}$

Algorithm: Unification

- unify(E)
- Input: E set of formulas or terms
- Output: MGU or $\perp$
  1. k := 0; σ_k := ϵ;
  2. if $|E\sigma_k| = 1$ then return σ_k; else D := D($E\sigma_k$);
  3. if a variable X and term t exist in D such that X does not occur in t
     - $\sigma_{k+1} = \sigma_k \circ \{t/X\}$; k++; goto 2;
  4. return $\perp$

Example: Unification

- unify($\{p(X, f(X)), p(Y, f(g(b)))\}$)
- $\sigma_0 = \epsilon$
- $E\sigma_0 = \{p(X, f(X)), p(Y, f(g(b)))\}$
- D($\{p(X, f(X)), p(Y, f(g(b)))\}$) = $\{X, Y\}$
- $\sigma_1 = \sigma_0 \circ \{Y/X\} = \{Y/X\}$
- $E\sigma_1 = \{p(Y, f(Y)), p(Y, f(g(b)))\}$
- D($\{p(Y, f(Y)), p(Y, f(g(b)))\}$) = $\{Y, g(b)\}$
- $\sigma_2 = \sigma_1 \circ \{g(b)/Y\} = \{Y\{g(b)/Y\}/X, g(b)/Y\} = \{g(b)/X, g(b)/Y\}$
- $E\sigma_2 = \{p(g(b), f(g(b)))\}$
- mgu is $\{g(b)/X, g(b)/Y\}$

First-Order Resolution: Resolvent

Definition

- Given two clauses C_1 and C_2, assume two variable renaming substitutions σ_1 and σ_2, such that $C_1\sigma_1$ and $C_2\sigma_2$ do not share variables.
- If $a \in C_1\sigma_1$ and $\neg b \in C_2\sigma_2$ such that a and b are unifiable with mgu ϑ, then $((C_1\sigma_1 \setminus \{a\}) \cup (C_2\sigma_2 \setminus \{\neg b\}))\vartheta$ is a resolvent of C_1 and C_2.

First-Order Resolution: Resolvent

Definition

- Given two clauses C_1 and C_2, assume two variable renaming substitutions σ_1 and σ_2, such that $C_1\sigma_1$ and $C_2\sigma_2$ do not share variables.
- If $a \in C_1\sigma_1$ and $\neg b \in C_2\sigma_2$ such that a and b are unifiable with mgu ϑ, then $((C_1\sigma_1 \setminus \{a\}) \cup (C_2\sigma_2 \setminus \{\neg b\}))\vartheta$ is a resolvent of C_1 and C_2.

First-Order Resolution: Factorization

Definition

Given a clause C, and two literals a, b of C, such that a and b are unifiable with mgu ϑ, then $C\vartheta$ is a factor of C.

Derivation

Definition

Given a set of clauses S, a derivation by resolution of a clause C from S is a sequence $C_1, \ldots, C_n$, such that $C_n = C$ and for each C_i $(0 \leq i \leq n)$ we have

1. $C_i \in S$ or
2. C_i is a resolvent of C_j and C_k, where $j < i$ and $k < i$ or
3. C_i is a factor of C_j, where $j < i$.

If a derivation by resolution of C from S exists, we write $S \vdash_R C$.

Derivation

Definition

Given a set of clauses S, a derivation by resolution of a clause C from S is a sequence $C_1, \ldots, C_n$, such that $C_n = C$ and for each C_i $(0 \leq i \leq n)$ we have

1. $C_i \in S$ or
2. C_i is a resolvent of C_j and C_k, where $j < i$ and $k < i$ or
3. C_i is a factor of C_j, where $j < i$.

If a derivation by resolution of C from S exists, we write $S \vdash_R C$.

Refutation

Definition

A derivation by resolution of $\square$ from S is called a refutation of S.

Resolution

Theorem

$S \vdash_R \Box$ if and only if S is unsatisfiable.

Proof.

Soundness by showing $S \models \Box$.

Completeness using Herbrand's theorem. □

Resolution

Theorem

$S \vdash_R \Box$ if and only if S is unsatisfiable.

Proof.

Soundness by showing $S \models \Box$.
Completeness using Herbrand's theorem. □

Linear Resolution

- Linear Resolution: Any intermediate derivation uses a clause obtained in the previous step.

Theorem

Linear resolution is refutation complete; i.e. if a formula is unsatisfiable, a refutation by linear resolution exists.

Horn and Goal Clauses, SLD Resolution

- A Horn clause is a clause containing at most one positive literal.
- A Goal clause is a clause containing no positive literal.
- SLD Resolution: Linear resolution, where at each step only goal clauses and (instances of) input clauses are used.

Theorem

SLD resolution is refutation complete for Horn clauses.

Prolog

- Prolog: Programmation en Logique
- Allow only Horn clauses and one goal clause.
- SLD resolution is the basis of Prolog.
- Additional procedural semantics.