Decision Procedures for Verification

First-Order Logic (3)

2.12.2014

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Until now:

Syntax (one-sorted signatures vs. many-sorted signatures)

Semantics

Structures (also many-sorted)

Models, Validity, and Satisfiability

Entailment and Equivalence

Theories (Syntactic vs. Semantics view)

2.4 Algorithmic Problems

Validity(F): $\models F$?

Satisfiability(F): F satisfiable?

Entailment(F,G): does F entail G?

Model($\mathcal{A}$,F): $\mathcal{A} \models F$?

Solve($\mathcal{A}$,F): find an assignment β such that $\mathcal{A}, \beta \models F$

Solve(F): find a substitution σ such that $\models F\sigma$

Abduce(F): find G with "certain properties" such that G entails F

Decidability/Undecidability

In 1931, Gödel published his incompleteness theorems in

> "Über formal unentscheidbare Sätze der
> Principia Mathematica und verwandter Systeme"

(in English "On Formally Undecidable Propositions of Principia Mathematica and Related Systems").

He proved for any computable axiomatic system that is powerful enough to describe the arithmetic of the natural numbers (e.g. the Peano axioms or Zermelo-Fraenkel set theory with the axiom of choice), that:

- If the system is consistent, it cannot be complete.
- The consistency of the axioms cannot be proven within the system.

Decidability/Undecidability

These theorems ended a half-century of attempts, beginning with the work of Frege and culminating in Principia Mathematica and Hilbert's formalism, to find a set of axioms sufficient for all mathematics.

The incompleteness theorems also imply that not all mathematical questions are computable.

Consequences of Gödel's Famous Theorems

1. For most signatures Σ, validity is undecidable for Σ-formulas. (One can easily encode Turing machines in most signatures.)
2. For each signature Σ, the set of valid Σ-formulas is recursively enumerable.
   (We will prove this by giving complete deduction systems.)
3. For $\Sigma = \Sigma_{PA}$ and $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *)$, the theory $Th(\mathbb{N}_*)$ is not recursively enumerable.

These undecidability results motivate the study of subclasses of formulas (fragments) of first-order logic

Q: Can you think of any fragments of first-order logic for which validity is decidable?

Some Decidable Fragments/Problems

Validity/Satisfiability/Entailment: Some decidable fragments:

- Variable-free formulas without equality:
  satisfiability is NP-complete. (why?)
- Variable-free Horn clauses (clauses with at most one positive atom): entailment is decidable in linear time.
- Monadic class: no function symbols, all predicates unary;
  validity is NEXPTIME-complete.
- Q: Other decidable fragments of FOL (with variables)?
  Which methods for proving decidability?

Goals

Identify:

- decidable fragments of first-order logic
- fragments of FOL for which satisfiability checking is easy

Methods:

- Theoretical methods (automata theory, finite model property)
- Adjust automated reasoning techniques
  (e.g. to obtaining efficient decision procedures)

  Extend methods for automated reasoning in propositional logic?

    Instantiation/reduction to propositional logic

    Extend the resolution calculus for first-order logic

Goals

Extend methods for automated reasoning in propositional logic?

Instantiation/reduction to propositional logic

Extend the resolution calculus for first-order logic

Ingredients:

- Give a method for translating formulae to clause form
- Regard formulae with variables as a set of all their instances (where variables are instantiated with ground terms)
  - Show that only certain instances are needed
    $\mapsto$ reduction to propositional logic
  - Finite encoding of infinitely many inferences
    $\mapsto$ resolution for first-order logic

2.5 Normal Forms and Skolemization

Study of normal forms motivated by

- reduction of logical concepts,
- efficient data structures for theorem proving.

The main problem in first-order logic is the treatment of quantifiers. The subsequent normal form transformations are intended to eliminate many of them.

Prenex Normal Form

Prenex formulas have the form

$$Q_1 x_1 \ldots Q_n x_n \ F,$$

where F is quantifier-free and $Q_i \in \{\forall, \exists\}$;
we call $Q_1 x_1 \ldots Q_n x_n$ the quantifier prefix and F the matrix of the formula.

Prenex Normal Form

Computing prenex normal form by the rewrite relation $\Rightarrow_P$:

$$
\begin{array}{rcll}
(F \leftrightarrow G) & \Rightarrow_P & (F \rightarrow G) \wedge (G \rightarrow F) & \\
\neg QxF & \Rightarrow_P & \overline{Q}x\neg F & (\neg Q) \\
(QxF \ \rho \ G) & \Rightarrow_P & Qy(F[y/x] \ \rho \ G), \ \ y \text{ fresh}, \ \ \rho \in \{\wedge, \vee\} & \\
(QxF \rightarrow G) & \Rightarrow_P & \overline{Q}y(F[y/x] \rightarrow G), \ \ y \text{ fresh} & \\
(F \ \rho \ QxG) & \Rightarrow_P & Qy(F \ \rho \ G[y/x]), \ \ y \text{ fresh}, \ \ \rho \in \{\wedge, \vee, \rightarrow\} &
\end{array}
$$

Here $\overline{Q}$ denotes the quantifier dual to Q, i.e., $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$.

Example

$$F := (\forall x((p(x) \vee q(x, y)) \wedge \exists z\, r(x, y, z))) \rightarrow ((p(z) \wedge q(x, z)) \wedge \forall z\, r(z, x, y))$$

Example

$$F := (\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \quad \exists x' \, ((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z)) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

Example

$$F := (\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \quad \exists x'((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z)) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \quad \exists x'(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y$$

Example

$$F := (\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \ \exists x'((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \ \exists x'(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y$$

$$\Rightarrow_P \ \exists x' \forall z' (((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y$$

Example

$$F := (\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \;\exists x'((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \;\exists x'(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y$$

$$\Rightarrow_P \;\exists x'\forall z'\,((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \;\exists x'\forall z'\,((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow \forall z''((p(z) \wedge q(x,z)) \wedge r(z'',x,$$

Example

$$F := (\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \ \exists x'((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \ \exists x'(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y$$

$$\Rightarrow_P \ \exists x' \forall z'\, ((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

$$\Rightarrow_P \ \exists x' \forall z'\, ((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow \forall z''((p(z) \wedge q(x,z)) \wedge r(z'',x,$$

$$\Rightarrow_P \ \exists x' \forall z' \forall z''(((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow ((p(z) \wedge q(x,z)) \wedge r(z'',x,$$

Skolemization

Intuition: replacement of $\exists y$ by a concrete choice function computing y from all the arguments y depends on.

Transformation $\Rightarrow_S$ (to be applied outermost, *not* in subformulas):

$$\forall x_1, \ldots, x_n \exists y F \quad \Rightarrow_S \quad \forall x_1, \ldots, x_n F[f(x_1, \ldots, x_n)/y]$$

where f/n is a new function symbol (Skolem function).

Skolemization

Together: $F \overset{*}{\Rightarrow}_P \underbrace{G}_{\text{prenex}} \overset{*}{\Rightarrow}_S \underbrace{H}_{\text{prenex, no } \exists}$

Theorem 2.9:

Let F, G, and H as defined above and closed. Then

(i) F and G are equivalent.

(ii) $H \models G$ but the converse is not true in general.

(iii) G satisfiable (wrt. Σ-alg) $\Leftrightarrow$ H satisfiable (wrt. Σ'-Alg) where $\Sigma' = (\Omega \cup SKF, \Pi)$, if $\Sigma = (\Omega, \Pi)$.

Clausal Normal Form (Conjunctive Normal Form)

$$
\begin{array}{rcl}
(F \leftrightarrow G) & \Rightarrow_K & (F \rightarrow G) \land (G \rightarrow F) \\
(F \rightarrow G) & \Rightarrow_K & (\neg F \lor G) \\
\neg(F \lor G) & \Rightarrow_K & (\neg F \land \neg G) \\
\neg(F \land G) & \Rightarrow_K & (\neg F \lor \neg G) \\
\neg\neg F & \Rightarrow_K & F \\
(F \land G) \lor H & \Rightarrow_K & (F \lor H) \land (G \lor H) \\
(F \land \top) & \Rightarrow_K & F \\
(F \land \bot) & \Rightarrow_K & \bot \\
(F \lor \top) & \Rightarrow_K & \top \\
(F \lor \bot) & \Rightarrow_K & F
\end{array}
$$

These rules are to be applied modulo associativity and commutativity of $\land$ and $\lor$. The first five rules, plus the rule $(\neg Q)$, compute the negation normal form (NNF) of a formula.

The Complete Picture

$$
\begin{array}{llll}
F & \Rightarrow^*_P & Q_1 y_1 \dots Q_n y_n \; G & (G \text{ quantifier-free}) \\
 & \Rightarrow^*_S & \forall x_1, \dots, x_m \; H & (m \leq n,\ H \text{ quantifier-free}) \\
 & \Rightarrow^*_K & \underbrace{\underbrace{\forall x_1, \dots, x_m}_{\text{leave out}} \; \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clauses } C_i}}_{F'} &
\end{array}
$$

$N = \{C_1, \dots, C_k\}$ is called the clausal (normal) form (CNF) of F.

Note: the variables in the clauses are implicitly universally quantified.

Theorem 2.10:

Let F be closed. Then $F' \models F$. (The converse is not true in general.)

Theorem 2.11:

Let F be closed. Then F is satisfiable iff F' is satisfiable iff N is satisfiable

Example

Given: $\exists u \forall w (\exists x (p(w, x, u) \vee \forall y (q(w, x, y) \wedge \exists z\, r(y, z))))$

Example

Given: $\exists u \forall w (\exists x (p(w,x,u) \lor \forall y\, (q(w,x,y) \land \exists z\, r(y,z))))$

Prenex Normal Form:

$$\stackrel{*}{\Rightarrow}_P \; \exists u \forall w \exists x \forall y \exists z ((p(w,x,u) \lor (q(w,x,y) \land r(y,z))))$$

Example

Given: $\exists u \forall w (\exists x (p(w, x, u) \vee \forall y \, (q(w, x, y) \wedge \exists z \, r(y, z))))$

Prenex Normal Form:

$$\Rightarrow_P^* \; \exists u \forall w \exists x \forall y \exists z ((p(w, x, u) \vee (q(w, x, y) \wedge r(y, z))))$$

Skolemisation:

$$\Rightarrow_S^* \; \forall w \forall y ((p(w, sk_x(w), sk_u) \vee (q(w, sk_x(w), y) \wedge r(y, g(w, y)))))$$

Example

Given: $\exists u \forall w (\exists x (p(w, x, u) \vee \forall y\, (q(w, x, y) \wedge \exists z\, r(y, z))))$

Prenex Normal Form:

$\Rightarrow_P^* \ \exists u \forall w \exists x \forall y \exists z ((p(w, x, u) \vee (q(w, x, y) \wedge r(y, z))))$

Skolemisation:

$\Rightarrow_S^* \ \forall w \forall y ((p(w, sk_x(w), sk_u) \vee (q(w, sk_x(w), y) \wedge r(y, g(w, y)))))$

Clause normal form:

$\Rightarrow_K^* \ \forall w \forall y [(p(w, sk_x(w), sk_u) \vee q(w, sk_x(w), y)) \wedge (p(w, sk_x(w), sk_u) \vee r(y, g(w, y$

Set of clauses:

$\{p(w, sk_x(w), sk_u) \vee q(w, sk_x(w), y), \quad p(w, sk_x(w), sk_u) \vee r(y, g(w, y))\}$

Optimization

Here is lots of room for optimization since we only can preserve satisfiability anyway:

- size of the CNF exponential when done naively;
- want to preserve the original formula structure;
- want small arity of Skolem functions.

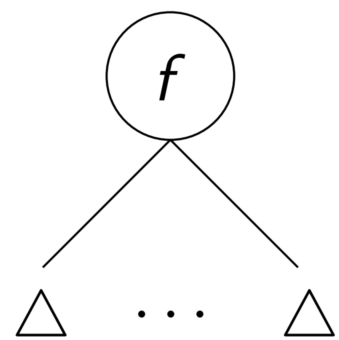
2.6 Herbrand Interpretations

From now an we shall consider PL without equality. Ω shall contains at least one constant symbol.

A Herbrand interpretation (over Σ) is a Σ-algebra $\mathcal{A}$ such that

- $U_{\mathcal{A}} = \mathsf{T}_{\Sigma}$ (= the set of ground terms over Σ)
- $f_{\mathcal{A}} : (s_1, \ldots, s_n) \mapsto f(s_1, \ldots, s_n),\ f/n \in \Omega$

$$f_{\mathcal{A}}(\triangle, \ldots, \triangle) =$$



Herbrand Interpretations

In other words, *values are fixed* to be ground terms and *functions are fixed* to be the term constructors. Only predicate symbols $p/m \in \Pi$ may be freely interpreted as relations $p_{\mathcal{A}} \subseteq \mathsf{T}_{\Sigma}^{m}$.

Proposition 2.12

Every set of ground atoms I uniquely determines a Herbrand interpretation $\mathcal{A}$ via

$$(s_1, \ldots, s_n) \in p_{\mathcal{A}} \quad :\Leftrightarrow \quad p(s_1, \ldots, s_n) \in I$$

Thus we shall identify Herbrand interpretations (over Σ) with sets of Σ-ground atoms.

Herbrand Interpretations

Example: $\Sigma_{Pres} = (\{0/0, s/1, +/2\},\ \{</2, \leq/2\})$

$\mathbb{N}$ as Herbrand interpretation over Σ_{Pres}:

$$
\begin{aligned}
I = \{ \quad & 0 \leq 0,\ 0 \leq s(0),\ 0 \leq s(s(0)),\ \ldots, \\
& 0 + 0 \leq 0,\ 0 + 0 \leq s(0),\ \ldots, \\
& \ldots,\ (s(0) + 0) + s(0) \leq s(0) + (s(0) + s(0)) \\
& \ldots \\
& s(0) + 0 < s(0) + 0 + 0 + s(0) \\
& \ldots\}
\end{aligned}
$$

Existence of Herbrand Models

A Herbrand interpretation I is called a Herbrand model of F, if $I \models F$.

Theorem 2.13

Let N be a set of Σ-clauses.

$$
\begin{aligned}
N \text{ satisfiable} \quad &\Leftrightarrow \quad N \text{ has a Herbrand model (over } \Sigma) \\
&\Leftrightarrow \quad G_\Sigma(N) \text{ has a Herbrand model (over } \Sigma)
\end{aligned}
$$

where $G_\Sigma(N) = \{C\sigma \text{ ground clause} \mid C \in N,\ \sigma : X \to \mathsf{T}_\Sigma\}$ is the set of ground instances of N.

(Proof – completeness proof of resolution for first-order logic.)

Example of a G_Σ

For Σ_{Pres} one obtains for

$$C = (x < y) \lor (y \leq s(x))$$

the following ground instances:

$(0 < 0) \lor (0 \leq s(0))$
$(s(0) < 0) \lor (0 \leq s(s(0)))$
...
$(s(0) + s(0) < s(0) + 0) \lor (s(0) + 0 \leq s(s(0) + s(0)))$
...

Consequences of Herbrans's theorem

Decidability results.

- Formulae without function symbols and without equality

  The Bernays-Schönfinkel Class $\exists^*\forall^*$

The Bernays-Schönfinkel Class

$\Sigma = (\Omega, \Pi)$, Ω is a finite set of constants

The Bernays-Schönfinkel class consists only of sentences of the form

$$\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m F(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

The Bernays-Schönfinkel Class

$\Sigma = (\Omega, \Pi)$, Ω is a finite set of constants

The Bernays-Schönfinkel class consists only of sentences of the form

$$\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m F(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

Theorem 2.14 Checking satisfiability of conjunctions of formulae in the Bernays-Schönfinkel class is decidable.

Idea: CNF translation:

$$\begin{aligned}
&\exists \overline{x}_1 \forall \overline{y}_1 F_1 \wedge \ldots \exists \overline{x}_n \forall \overline{y}_n F_n \\
&\qquad \Rightarrow_P \exists \overline{x}_1 \ldots \exists \overline{x}_n \forall \overline{y}_1 \ldots \forall \overline{y}_n F(\overline{x}_1, \ldots, \overline{x}_n, \overline{y}_1, \ldots, \overline{y}_n) \\
&\qquad \Rightarrow_S \forall \overline{y}_1 \ldots \forall \overline{y}_m F(\overline{c}_1, \ldots, \overline{c}_n, \overline{y}_1, \ldots, \overline{y}_n) \\
&\qquad \Rightarrow_K \forall \overline{y}_1 \ldots \forall \overline{y}_m \bigwedge \bigvee L_i((\overline{c}_1, \ldots, \overline{c}_n, \overline{y}_1, \ldots, \overline{y}_n)
\end{aligned}$$

$\overline{c}_1, \ldots, \overline{c}_n$ are tuples of Skolem constants

The Bernays-Schönfinkel Class

$\Sigma = (\Omega, \Pi)$, Ω is a finite set of constants

The Bernays-Schönfinkel class consists only of sentences of the form

$$\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m F(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

Theorem 2.14 Checking satisfiability of conjunctions of formulae in the Bernays-Schönfinkel class is decidable.

Idea: CNF translation:

$$\exists \overline{x}_1 \forall \overline{y}_1 F_1 \wedge \ldots \exists \overline{x}_n \forall \overline{y}_n F_n$$
$$\Rightarrow^*_K \forall \overline{y}_1 \ldots \forall \overline{y}_m \bigwedge \bigvee L_i((\overline{c}_1, \ldots, \overline{c}_n, \overline{y}_1, \ldots, \overline{y}_n)$$

$\overline{c}_1, \ldots, \overline{c}_n$ are tuples of Skolem constants

The Herbrand Universe is finite $\mapsto$ decidability

Tractable fragments of FOL

We showed that satisfiability of any finite set of ground Horn clauses can be checked in PTIME (linear time)

Variable-free Horn clauses

Data structures

Atoms $P_1, \dots, P_n \mapsto \{1, \dots, n\}$

neg-occ-list(A): list of all clauses in which A occurs negatively
pos-occ-list(A): list of all clauses in which A occurs positively

Clause:	P_1	P_2	$\dots$	P_n	counter
	neg	neg		pos	$\uparrow$
		$\uparrow$			number of literals

first-active-literal (fal): first literal not marked as deleted.

atom status:	pos	(deduced as positive unit clause)
	neg	(deduced as negative unit clause)
	nounit	(otherwise)

Variable-free Horn clauses

Input: Set N of Horn formulae

Step 1. Collect unit clauses; check if complementary pairs exist

forall $C \in N$ **do**

if is-unit(C) **then begin** const. time

L := first-active-literal(C) const. time

if state(atom(L)) = nounit **then** state(atom(L)) = sign(L) const. time

push(atom(L), stack)

else if state(atom(L)) $\neq$ sign(L) **then return false**

Variable-free Horn clauses

2. Process the unit clauses in the stack

while stack $\neq \emptyset$ **do**

begin A := top(stack); pop(stack)

if state(A) = pos **then** delete-literal-list := neg-oc-list(A) — O(# neg-oc-list)

else delete-literal-list := pos-oc-list(A) — O(# pos-oc-list)

endif

for all C in delete-literal-list **do**

if state(A) = pos **then** delete-literal(A,C) — const. time + nfal - ofal

if state(A) = neg **then** delete-literal($\neg$ A,C) — const. time + nfal - ofal

if unit(C) **then** L1 := first-active-literal(C) — const. time

if state(atom(L1)) = nounit **then** state(atom(L1)) = sign(L1),
L1 $\rightarrow$ stack

elseif state(atom(L1)) $\neq$ sign(L1) **then** return false

endif

end

Tractable fragments of FOL

We showed that satisfiability of any finite set of ground Horn clauses can be checked in PTIME (linear time)

- Similar fragment of the Bernays-Schönfinkel class?

Motivation: Deductive Databases

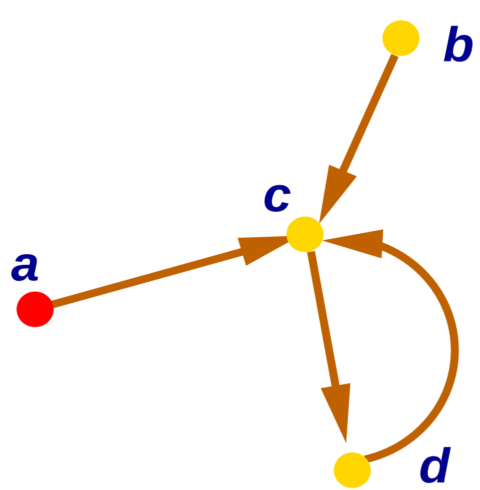
Deductive database

Inference rules:
Facts:
Query:

Motivation: Deductive Databases

Deductive database Example: reachability in graphs

Inference rules:	$\dfrac{S(x)}{R(x)} \qquad \dfrac{R(x) \quad E(x,y)}{R(y)}$
Facts:	$S(a), E(a,c), E(c,d), E(d,c), E(b,c)$
Query:	$R(d)$



$S(a), E(a,c), E(c,d), E(d,c), E(b,c)$

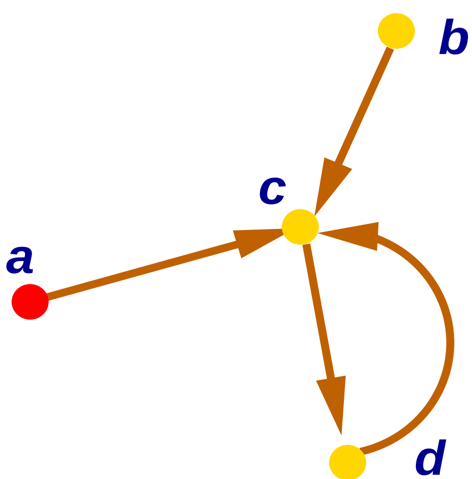
Note: S, E stored relations (Extensional DB)

R defined relation (Intensional DB)

Motivation: Deductive Databases

Deductive database Example: reachability in graphs

Inference rules:	$\dfrac{S(x)}{R(x)} \qquad \dfrac{R(x) \quad E(x,y)}{R(y)}$
Facts:	$S(a), E(a,c), E(c,d), E(d,c), E(b,c)$
Query:	$R(d)$



$S(a), E(a,c), E(a,d), E(c,d), E(b,c)$,
$R(a)$

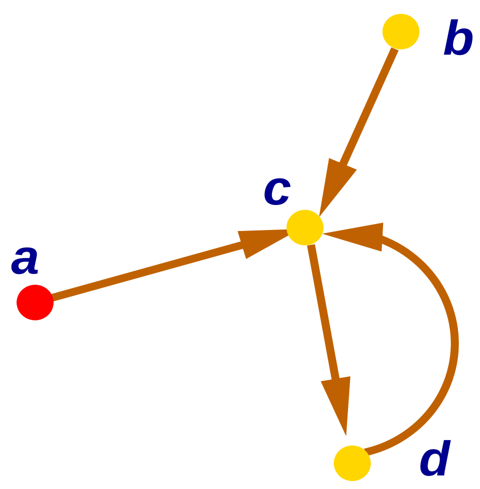
Note: S, E stored relations (Extensional DB)

R defined relation (Intensional DB)

Motivation: Deductive Databases

Deductive database Example: reachability in graphs

Inference rules:	$\dfrac{S(x)}{R(x)}$ $\qquad$ $\dfrac{R(x) \quad E(x,y)}{R(y)}$
Facts:	$S(a), E(a,c), E(c,d), E(d,c), E(b,c)$
Query:	$R(d)$



$S(a), E(a,c), E(a,d), E(c,d), E(b,c),$
$R(a), R(c)$

Note: S, E stored relations (Extensional DB)

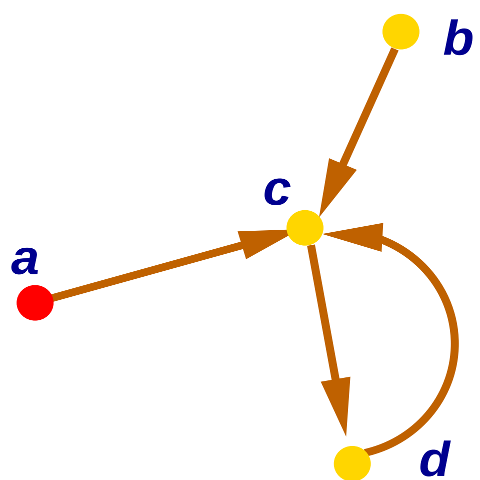
R defined relation (Intensional DB)

Motivation: Deductive Databases

Deductive database Example: reachability in graphs

Inference rules:	$\dfrac{S(x)}{R(x)} \qquad \dfrac{R(x) \quad E(x,y)}{R(y)}$
Facts:	$S(a), E(a,c), E(c,d), E(d,c), E(b,c)$

Query:	$R(d)$



$S(a), E(a,c), E(a,d), E(c,d), E(b,c),$
$R(a), R(c), R(d)$

Note: S, E stored relations (Extensional DB)

R defined relation (Intensional DB)

Motivation: Deductive Databases

Deductive database $\mapsto$ **Datalog** (Horn clauses, no function symbols)

Inference rules:	$\underbrace{S(x) \rightarrow R(x) \quad R(x) \wedge E(x,y) \rightarrow R(y)}_{\text{set } \mathcal{K} \text{ of Horn clauses}}$
Facts:	$\underbrace{S(a), E(a,c), E(c,d), E(d,c), E(b,c)}_{\text{set } \mathcal{F} \text{ of ground atoms}}$
Query:	$\underbrace{R(d)}_{\text{ground atom } G}$

$$\mathcal{F} \models_{\mathcal{K}} G \quad \text{iff} \quad \mathcal{K} \cup \mathcal{F} \models G \quad \text{iff} \quad \mathcal{K} \cup \mathcal{F} \cup \neg G \models \bot$$

Note: S, E stored relations (Extensional DB)

R defined relation (Intensional DB)

Motivation: Deductive Databases

Deductive database $\mapsto$ **Datalog** (Horn clauses, no function symbols)

Inference rules:	$\underbrace{S(x) \rightarrow R(x) \quad R(x) \wedge E(x,y) \rightarrow R(y)}_{\text{set } \mathcal{K} \text{ of Horn clauses}}$
Facts:	$\underbrace{S(a), E(a,c), E(c,d), E(d,c), E(b,c)}_{\text{set } \mathcal{F} \text{ of ground atoms}}$
Query:	$\underbrace{R(d)}_{\text{ground atom } G}$

Ex:

$$\cfrac{\cfrac{\cfrac{S(a) \quad S(x) \rightarrow R(x)}{R(a)} \quad E(a,c) \quad R(x) \wedge E(x,y) \rightarrow R(y)}{R(c)} \quad E(c,d) \quad R(x) \wedge E(x,y) \rightarrow R(}{R(d)}$$

Ground entailment for function-free Horn clauses

Assumption:

The signature does not contain function symbols of arity ≥ 1.

Given:

- Set H of (function-free) Horn clauses
- Ground Horn clause $G = \bigwedge A_i \rightarrow A$.

Theorem 2.15 The following are equivalent:

(1) $H \models \bigwedge A_i \rightarrow A$

(2) $H \wedge \bigwedge A_i \models A$

(3) $H \wedge \bigwedge A_i \wedge \neg A \models \perp$

Decidable in PTIME in the size of G for a fixed H.

Generalization: Local theories

[McAllester,Givan'92], [Basin,Ganzinger'96,01], [Ganzinger'01]

Assumption: the signature is allowed to contain function symbols

Definition. H set of Horn clauses is called local iff for every ground clause C the following are equivalent:

(1) $H \models C$

(2) $H[C] \models C$,
where $H[C]$ is the family of all instances of H in which the variables are replaced by ground subterms occurring in H or C.

Theorem 2.16 For a fixed local theory H, testing ground entailment w.r.t. H is in PTIME.

Will be discussed in more detail in the exercises

2.7 General Resolution

Propositional resolution:

refutationally complete,

clearly inferior to the DPLL procedure
(even with various improvements).

But: in contrast to the DPLL procedure, resolution can be easily extended to non-ground clauses.

Propositional resolution: reminder

Resolution inference rule:

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D}$$

Terminology: $C \vee D$: resolvent; A: resolved atom

(Positive) factorisation inference rule:

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

Resolution for ground clauses

- Exactly the same as for propositional clauses

  Ground atoms $\mapsto$ propositional variables

Theorem

Res is sound and refutationally complete (for all sets of ground clauses)

Sample Refutation

1.	$\neg P(f(a)) \vee \neg P(f(a)) \vee Q(b)$	(given)
2.	$P(f(a)) \vee Q(b)$	(given)
3.	$\neg P(g(b,a)) \vee \neg Q(b)$	(given)
4.	$P(g(b,a))$	(given)
5.	$\neg P(f(a)) \vee Q(b) \vee Q(b)$	(Res. 2. into 1.)
6.	$\neg P(f(a)) \vee Q(b)$	(Fact. 5.)
7.	$Q(b) \vee Q(b)$	(Res. 2. into 6.)
8.	$Q(b)$	(Fact. 7.)
9.	$\neg P(g(b,a))$	(Res. 8. into 3.)
10.	$\bot$	(Res. 4. into 9.)

Resolution for ground clauses

- Refinements with orderings and selection functions:

  Need: - well-founded ordering on ground atomic formulae/literals

  - selection function (for negative literals)

  $$S : C \quad \mapsto \quad \text{set of occurrences of } \textit{negative} \text{ literals in } C$$

  Example of selection with selected literals indicated as $\boxed{X}$:

  $\boxed{\neg A} \vee \neg A \vee B$

  $\boxed{\neg B_0} \vee \boxed{\neg B_1} \vee A$

Resolution Calculus $Res_S^{\succ}$

Ordered resolution with selection

$$\frac{C \vee A \qquad D \vee \neg A}{C \vee D}$$

if

1. $A \succ C$;
2. nothing is selected in C by S;
3. $\neg A$ is selected in $D \vee \neg A$,

   or else nothing is selected in $D \vee \neg A$ and $\neg A \succeq \max(D)$.

Note: For positive literals, $A \succ C$ is the same as $A \succ \max(C)$.

Ordered factoring

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

if A is maximal in C and nothing is selected in C.

Resolution for ground clauses

Let $\succ$ be a total and well-founded ordering on ground atoms, and S a selection function.

Theorem. $\text{Res}_S^{\succ}$ is sound and refutationally complete for all sets of ground clauses.

Soundness: sufficient to show that
(1) $C \vee A, D \vee \neg A \models C \vee D$
(2) $C \vee A \vee A \models C \vee A$

Completeness: Let $\succ$ be a clause ordering, let N be saturated wrt. $Res_S^{\succ}$, and suppose that $\bot \notin N$. Then $I_N^{\succ} \models N$, where $I_N^{\succ}$ is incrementally constructed as follows:

Construction of Candidate Models Formally

Let $N, \succ$ be given.

- Order N increasing w.r.t. the extension of $\succ$ to clauses.
- Define sets I_C and Δ_C for all ground clauses C over the given signature inductively over $\succ$:

$$I_C := \bigcup_{C \succ D} \Delta_D$$

$$\Delta_C := \begin{cases} \{A\}, & \text{if } C \in N,\ C = C' \vee A,\ A \succ C',\ I_C \not\models C \\ & \text{and nothing is selected in } C \\ \emptyset, & \text{otherwise} \end{cases}$$

We say that C produces A, if $\Delta_C = \{A\}$.

The candidate model for N (wrt. $\succ$) is given as $I_N^{\succ} := \bigcup_C \Delta_C$.

(We write I_N for $I_N^{\succ}$ if $\succ$ is irrelevant or known from the context.)

Completeness (Reminder)

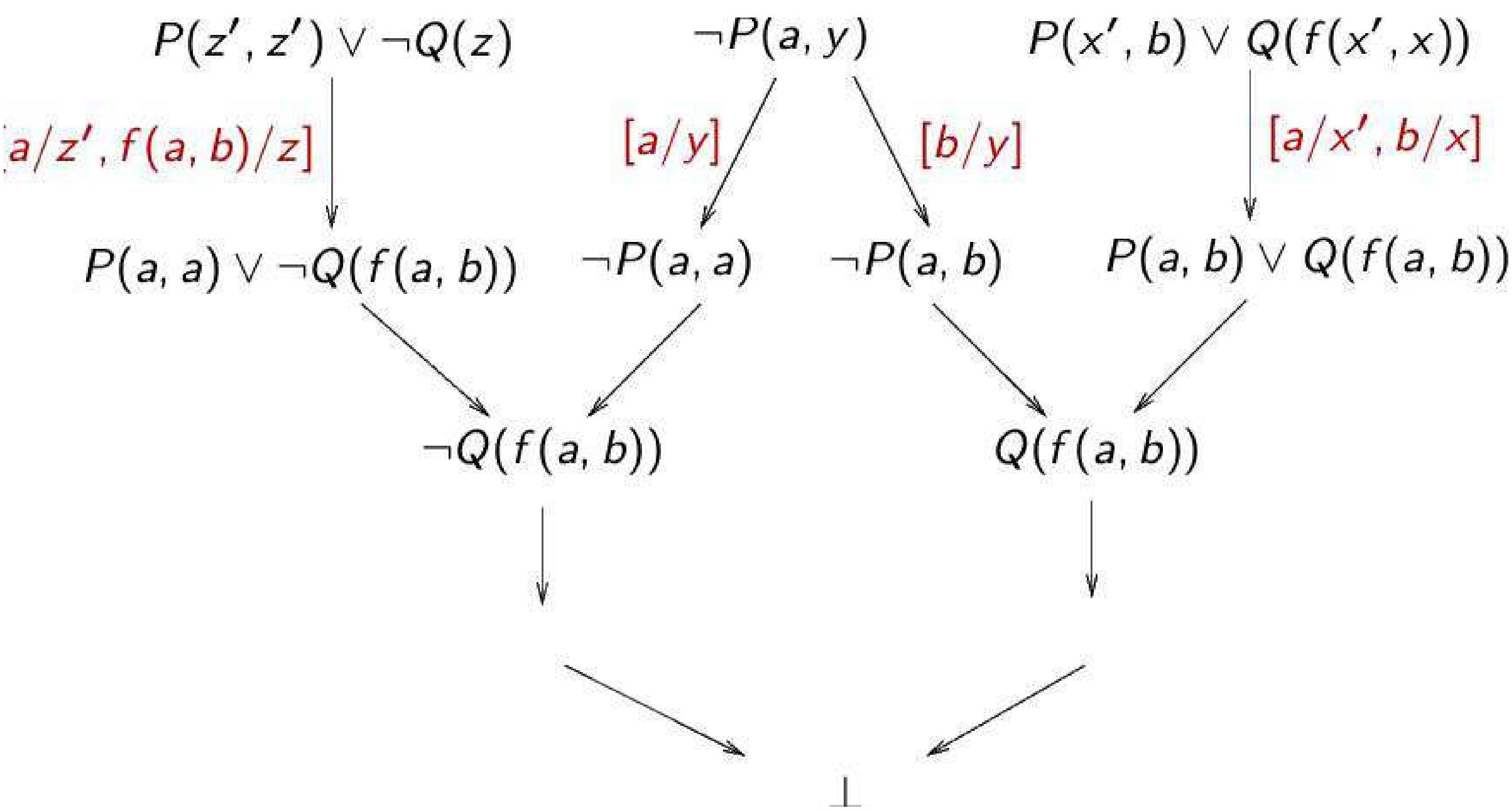
Theorem. Let $\succ$ be a clause ordering, let N be saturated wrt. $Res_S^{\succ}$, and suppose that $\bot \notin N$. Then $I_N^{\succ} \models N$.

Proof: Suppose $\bot \notin N$, but $I_N^{\succ} \not\models N$. Let $C \in N$ minimal (in $\succ$) such that $I_N^{\succ} \not\models C$. Since C is false in I_N, C is not productive. As $C \neq \bot$ there exists a maximal atom A in C.

1. $C = \neg A \vee C'$ (maximal atom occurs negatively) $\Rightarrow I_N \models A$, $I_N \not\models C'$
   Then some $D = D' \vee A \in N$ produces A. As $\frac{D' \vee A \qquad \neg A \vee C'}{D' \vee C'}$, we infer that $D' \vee C' \in N$, and $C \succ D' \vee C'$ and $I_N \not\models D' \vee C'$
   $\Rightarrow$ contradicts minimality of C.
2. $C = \boxed{\neg A} \vee C'$ ($\neg A$ is selected) $\Rightarrow I_N \models A$, $I_N \not\models C'$
   The argument in 1. applies also in this case.
3. $C = C' \vee A \vee A$. Then $\frac{C' \vee A \vee A}{C' \vee A}$ yields a smaller counterexample $C' \vee A \in N$. $\Rightarrow$ contradicts minimality of C.

General Resolution through Instantiation

Idea: instantiate clauses appropriately:



General Resolution through Instantiation

Problems:

More than one instance of a clause can participate in a proof.

Even worse: There are infinitely many possible instances.

Observation:

Instantiation must produce complementary literals
(so that inferences become possible).

Idea:

Do not instantiate more than necessary to get complementary literals.

Resolution Principle

Problem: Make saturation of infinite sets of clauses as they arise from taking the (ground) instances of finitely many general clauses (with variables) effective and efficient.

Idea (Robinson 65):

- Resolution for general clauses:
- *Equality* of ground atoms is generalized to *unifiability* of general atoms;
- Only compute *most general* (minimal) unifiers.

Resolution for General Clauses

General binary resolution ***Res*****:**

$$\frac{C \vee A \qquad D \vee \neg B}{(C \vee D)\sigma} \quad \text{if } \sigma = \text{mgu}(A, B) \qquad \text{[resolution]}$$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{if } \sigma = \text{mgu}(A, B) \quad \text{[factorization]}$$

For inferences with more than one premise, we assume that the variables in the premises are (bijectively) renamed such that they become different to any variable in the other premises.
We do not formalize this. Which names one uses for variables is otherwise irrelevant.

Unification

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ (s_i, t_i terms or atoms) a multi-set of equality problems. A substitution σ is called a unifier of E if $s_i\sigma = t_i\sigma$ for all $1 \leq i \leq n$.

If a unifier of E exists, then E is called unifiable.

Unification after Martelli/Montanari

$$
\begin{array}{llcl}
(1) & t \doteq t, E & \Rightarrow_{MM} & E \\
(2) & f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n), E & \Rightarrow_{MM} & s_1 \doteq t_1, \ldots, s_n \doteq t_n, E \\
(3) & f(\ldots) \doteq g(\ldots), E & \Rightarrow_{MM} & \bot \\
(4) & x \doteq t, E & \Rightarrow_{MM} & x \doteq t, E[t/x] \\
 & & & \quad \text{if } x \in var(E), x \notin var(t) \\
(5) & x \doteq t, E & \Rightarrow_{MM} & \bot \\
 & & & \quad \text{if } x \neq t, x \in var(t) \\
(6) & t \doteq x, E & \Rightarrow_{MM} & x \doteq t, E \\
 & & & \quad \text{if } t \notin X
\end{array}
$$

Examples

Example 1:

$$\{x \doteq f(a), g(x,x) \doteq g(x,y)\} \quad \Rightarrow_4$$
$$\{x \doteq f(a), g(f(a), f(a)) \doteq g(f(a), y)\} \quad \Rightarrow_2$$
$$\{x \doteq f(a), f(a) \doteq f(a), f(a) \doteq y\} \quad \Rightarrow_1$$
$$\{x \doteq f(a), f(a) \doteq y\} \quad \Rightarrow_6$$
$$\{x \doteq f(a), y \doteq f(a)\}$$

Example 2:

$$\{x \doteq f(a), g(x,x) \doteq h(x,y)\} \quad \Rightarrow_3 \bot$$

Example 3:

$$\{f(x,x) \doteq f(y, g(y))\} \quad \Rightarrow_2$$
$$\{x \doteq y, x \doteq g(y)\} \quad \Rightarrow_4$$
$$\{x \doteq y, y \doteq g(y)\} \quad \Rightarrow_5 \bot$$

MM: Main Properties

If $E = x_1 \doteq u_1, \ldots, x_k \doteq u_k$, with x_i pairwise distinct, $x_i \notin var(u_j)$, then E is called an (equational problem in) solved form representing the solution $\sigma_E = [u_1/x_1, \ldots, u_k/x_k]$.

Proposition 2.18:
If E is a solved form then σ_E is am mgu of E.

Theorem 2.19:

1. If $E \Rightarrow_{MM} E'$ then σ is a unifier of E iff σ is a unifier of E'
2. If $E \Rightarrow^*_{MM} \bot$ then E is not unifiable.
3. If $E \Rightarrow^*_{MM} E'$ with E' in solved form, then $\sigma_{E'}$ is an mgu of E.

MM: Main Properties

Theorem 2.19:

1. If $E \Rightarrow_{MM} E'$ then σ is a unifier of E iff σ is a unifier of E'
2. If $E \Rightarrow^*_{MM} \bot$ then E is not unifiable.
3. If $E \Rightarrow^*_{MM} E'$ with E' in solved form, then $\sigma_{E'}$ is an mgu of E.

Proof:
(1) We have to show this for each of the rules. Let's treat the case for the 4th rule here. Suppose σ is a unifier of $x \doteq t$, that is, $x\sigma = t\sigma$. Thus, $\sigma \circ [t/x] = \sigma[x \mapsto t\sigma] = \sigma[x \mapsto x\sigma] = \sigma$. Therefore, for any equation $u \doteq v$ in E: $u\sigma = v\sigma$, iff $u[t/x]\sigma = v[t/x]\sigma$. (2) and (3) follow by induction from (1) using Proposition 2.18.

Main Unification Theorem

Theorem 2.20:
E is unifiable if and only if there is a most general unifier σ of E, such that σ is idempotent and $dom(\sigma) \cup codom(\sigma) \subseteq var(E)$.

Proof: See e.g. Baader & Nipkow: Term rewriting and all that.

Problem: *exponential growth* of terms possible

Example:

$E = \{x_1 \approx f(x_0, x_0), x_2 \approx f(x_1, x_1), \ldots, x_n \approx f(x_{n-1}, x_{n-1})\}$

m.g.u. $[x_1 \mapsto f(x_0, x_0), x_2 \mapsto f(f(x_0, x_0), f(x_0, x_0)), ...]$

$x_i \mapsto$ complete binart tree of heigth i

Solution: Use acyclic term graphs; union/find algorithms

Lifting Lemma

Lemma 2.21

Let C and D be variable-disjoint clauses. If

$$\begin{array}{ccc} C & & D \\ \downarrow \sigma & & \downarrow \rho \\ C\sigma & & D\rho \\ \hline & C' & \end{array} \qquad \text{[propositional resolution]}$$

then there exists a substitution τ such that

$$\begin{array}{c} \dfrac{C \qquad\qquad D}{C''} \\ \downarrow \rho \\ C' = C''\tau \end{array} \qquad \text{[general resolution]}$$

Lifting Lemma

An analogous lifting lemma holds for factorization.

Saturation of Sets of General Clauses

Corollary 2.22:

Let N be a set of general clauses saturated under Res, i.e., $\mathit{Res}(N) \subseteq N$. Then also $G_\Sigma(N)$ is saturated, that is,

$$\mathit{Res}(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Saturation of Sets of General Clauses

Proof:

W.l.o.g. we may assume that clauses in N are pairwise variable-disjoint. (Otherwise make them disjoint, and this renaming process changes neither $Res(N)$ nor $G_\Sigma(N)$.)

Let $C' \in Res(G_\Sigma(N))$, meaning (i) there exist resolvable ground instances $C\sigma$ and $D\rho$ of N with resolvent C', or else (ii) C' is a factor of a ground instance $C\sigma$ of C.

Case (i): By the Lifting Lemma, C and D are resolvable with a resolvent C'' with $C''\tau = C'$, for a suitable substitution τ. As $C'' \in N$ by assumption, we obtain that $C' \in G_\Sigma(N)$.

Case (ii): Similar.

Herbrand's Theorem

Lemma 2.23:

Let N be a set of Σ-clauses, let $\mathcal{A}$ be an interpretation.

Then $\mathcal{A} \models N$ implies $\mathcal{A} \models G_\Sigma(N)$.

Lemma 2.24:

Let N be a set of Σ-clauses, let $\mathcal{A}$ be a *Herbrand* interpretation.

Then $\mathcal{A} \models G_\Sigma(N)$ implies $\mathcal{A} \models N$.

Herbrand's Theorem

Theorem 2.25 (Herbrand):
A set N of Σ-clauses is satisfiable if and only if it has a Herbrand model over Σ.

Proof:
The "$\Leftarrow$" part is trivial. For the "$\Rightarrow$" part let $N \not\models \bot$.

$$
\begin{aligned}
N \not\models \bot &\Rightarrow \bot \notin Res^*(N) && \text{(resolution is sound)}\\
&\Rightarrow \bot \notin G_\Sigma(Res^*(N))\\
&\Rightarrow I_{G_\Sigma(Res^*(N))} \models G_\Sigma(Res^*(N)) && \text{(Thm. 2.23; Cor. 2.32)}\\
&\Rightarrow I_{G_\Sigma(Res^*(N))} \models Res^*(N) && \text{(Lemma 2.34)}\\
&\Rightarrow I_{G_\Sigma(Res^*(N))} \models N && (N \subseteq Res^*(N))
\end{aligned}
$$

Refutational Completeness of General Resolution

Theorem 2.26:

Let N be a set of general clauses where $Res(N) \subseteq N$. Then

$$N \models \bot \Leftrightarrow \bot \in N.$$

Proof:

Let $Res(N) \subseteq N$. By Corollary 2.22: $Res(G_\Sigma(N)) \subseteq G_\Sigma(N)$

$N \models \bot \Leftrightarrow G_\Sigma(N) \models \bot$ (Lemma 2.23/2.24; Theorem 2.25)

$\Leftrightarrow \bot \in G_\Sigma(N)$ (propositional resolution sound and complete)

$\Leftrightarrow \bot \in N$