Foundations of Artificial Intelligence

11. Predicate Logic

Syntax and Semantics, Normal Forms, Herbrand Expansion, Resolution

Wolfram Burgard, Bernhard Nebel, and Martin Riedmiller

Albert-Ludwigs-Universität Freiburg

July 15, 2011

Contents

Motivation

We can already do a lot with propositional logic. It is, however, annoying that there is no structure in the atomic propositions.

Example:

"All blocks are red"
"There is a block A"
It should follow that "A is red"

But propositional logic cannot handle this.

Idea: We introduce individual variables, predicates, functions, ...

→ First-Order Predicate Logic (PL1)

The Alphabet of First-Order Predicate Logic

Symbols:

- Operators: $\neg, \vee, \wedge, \forall, \exists, =$
- Variables: $x, x_1, x_2, \ldots, x', x'', \ldots, y, \ldots, z, \ldots$
- Brackets: (), [], {}
- Function symbols (e.g., $weight()$, $color()$)
- Predicate symbols (e.g., $block()$, $red()$)
- Predicate and function symbols have an arity (number of arguments).
  - 0-ary predicate: propositional logic atoms
  - 0-ary function: constant
- We suppose a countable set of predicates and functions of any arity.
- "$=$" is usually not considered a predicate, but a logical symbol

The Grammar of First-Order Predicate Logic (1)

Terms (represent objects):

1. Every variable is a term.
2. If $t_1, t_2, \ldots, t_n$ are terms and f is an n-ary function, then

$$f(t_1, t_2, \ldots, t_n)$$

   is also a term.

   Terms without variables: ground terms.

Atomic Formulae (represent statements about objects)

1. If $t_1, t_2, \ldots, t_n$ are terms and P is an n-ary predicate, then $P(t_1, t_2, \ldots, t_n)$ is an atomic formula.
2. If t_1 and t_2 are terms, then $t_1 = t_2$ is an atomic formula.

   Atomic formulae without variables: ground atoms.

The Grammar of First-Order Predicate Logic (2)

Formulae:

1. Every atomic formula is a formula.
2. If φ and ψ are formulae and x is a variable, then

$$\neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,\ \varphi \Rightarrow \psi,\ \varphi \Leftrightarrow \psi,\ \exists x\varphi \text{ and } \forall x\varphi$$

   are also formulae.
   $\forall$, $\exists$ are as strongly binding as $\neg$.

Propositional logic is part of the PL1 language:

1. Atomic formulae: only 0-ary predicates
2. Neither variables nor quantifiers.

Alternative Notation

Here	Elsewhere
$\neg\varphi$	$\sim\varphi \quad \overline{\varphi}$
$\varphi \wedge \psi$	$\varphi \& \psi \quad \varphi \bullet \psi \quad \varphi, \psi$
$\varphi \vee \psi$	$\varphi \vert \psi \quad \varphi ; \psi \quad \varphi + \psi$
$\varphi \Rightarrow \psi$	$\varphi \rightarrow \psi \quad \varphi \supset \psi$
$\varphi \Leftrightarrow \psi$	$\varphi \leftrightarrow \psi \quad \varphi \equiv \psi$
$\forall x \varphi$	$(\forall x)\varphi \wedge x\varphi$
$\exists x \varphi$	$(\exists x)\varphi \vee x\varphi$

Meaning of PL1-Formulae

Our example: $\forall x[Block(x) \Rightarrow Red(x)]$, $Block(a)$

For all objects x: If x is a block, then x is red and a is a block.

Generally:

- Terms are interpreted as objects.
- Universally-quantified variables denote all objects in the universe.
- Existentially-quantified variables represent one of the objects in the universe (made true by the quantified expression).
- Predicates represent subsets of the universe.

Similar to propositional logic, we define interpretations, satisfiability, models, validity, ...

Semantics of PL1-Logic

Interpretation: $I = \langle D, \bullet^I \rangle$ where D is an arbitrary, non-empty set and $\bullet^I$ is a function that

- maps n-ary function symbols to functions over D:
  $f^I \in [D^n \mapsto D]$
- maps individual constants to elements of D:
  $a^I \in D$
- maps n-ary predicate symbols to relations over D:
  $P^I \subseteq D^n$

Interpretation of ground terms:

$$(f(t_1, \ldots, t_n))^I = f^I(t_1^I, \ldots, t_n^I)$$

Satisfaction of ground atoms $P(t_1, \ldots, t_n)$:

$$I \models P(t_1, \ldots, t_n) \text{ iff } \langle t_1^I, \ldots, t_n^I \rangle \in P^I$$

Example (1)

$$\begin{aligned} D &= \{d_1, \ldots, d_n \mid n > 1\} \\ a^I &= d_1 \\ b^I &= d_2 \\ c^I &= \ldots \\ Block^I &= \{d_1\} \\ Red^I &= D \\ I &\models Red(b) \\ I &\not\models Block(b) \end{aligned}$$

Example (2)

$$\begin{aligned} D &= \{1, 2, 3, \ldots\} \\ 1^I &= 1 \\ 2^I &= 2 \\ &\ldots \\ Even^I &= \{2, 4, 6, \ldots\} \\ succ^I &= \{(1 \mapsto 2), (2 \mapsto 3), \ldots\} \\ I &\models Even(2) \\ I &\not\models Even(succ(2)) \end{aligned}$$

Semantics of PL1: Variable Assignment

Set of all variables V. Function $\alpha : V \mapsto D$

Notation: $\alpha[x/d]$ is the same as α apart from point x.

For $x : \alpha[x/d](x) = d$.

Interpretation of terms under I, α:

$$\begin{aligned} x^{I,\alpha} &= \alpha(x) \\ a^{I,\alpha} &= a^I \\ (f(t_1, \ldots, t_n))^{I,\alpha} &= f^I(t_1^{I,\alpha}, \ldots, t_n^{I,\alpha}) \end{aligned}$$

Satisfaction of atomic formulae:

$$I, \alpha \models P(t_1, \ldots, t_n) \text{ iff } \langle t_1^{I,\alpha}, \ldots, t_n^{I,\alpha} \rangle \in P^I$$

Example

$$\alpha = \{(x \mapsto d_1), (y \mapsto d_2)\}$$
$$I, \alpha \models \mathit{Red}(x)$$
$$I, \alpha[y/d_1] \models \mathit{Block}(y)$$

Semantics of PL1: Satisfiability

A formula φ is satisfied by an interpretation I and a variable assignment α, i.e., $I, \alpha \models \varphi$:

$$I, \alpha \models \top$$
$$I, \alpha \not\models \bot$$
$$I, \alpha \models \neg\varphi \text{ iff } I, \alpha \not\models \varphi$$
$$\dots$$

and all other propositional rules as well as

$I, \alpha \models P(t_1, \dots, t_n)$	iff	$\langle t_1^{I,\alpha}, \dots, t_n^{I,\alpha} \rangle \in P^{I,\alpha}$
$I, \alpha \models \forall x \varphi$	iff	for all $d \in D$, $I, \alpha[x/d] \models \varphi$
$I, \alpha \models \exists x \varphi$	iff	there exists a $d \in D$ with $I, \alpha[x/d] \models \varphi$

Example

$$T = \{\mathit{Block}(a), \mathit{Block}(b), \forall x(\mathit{Block}(x) \Rightarrow \mathit{Red}(x))\}$$
$$D = \{d_1, \dots, d_n \mid n > 1\}$$
$$a^I = d_1$$
$$b^I = d_2$$
$$\mathit{Block}^I = \{d_1\}$$
$$\mathit{Red}^I = D$$
$$\alpha = \{(x \mapsto d_1), (y \mapsto d_2)\}$$

Questions:

1. $I, \alpha \models \mathit{Block}(b) \vee \neg \mathit{Block}(b)$?
2. $I, \alpha \models \mathit{Block}(x) \Rightarrow (\mathit{Block}(x) \vee \neg \mathit{Block}(y))$?
3. $I, \alpha \models \mathit{Block}(a) \wedge \mathit{Block}(b)$?
4. $I, \alpha \models \forall x(\mathit{Block}(x) \Rightarrow \mathit{Red}(x))$?
5. $I, \alpha \models \top$?

Free and Bound Variables

$$\forall x[R(\boxed{y}, \boxed{z}) \wedge \exists y\{\neg P(y, x) \vee R(y, \boxed{z})\}]$$

The boxed appearances of y and z are free. All other appearances of x,y,z are bound.

Formulae with no free variables are called closed formulae or sentences. We form theories from closed formulae.

Note: With closed formulae, the concepts *logical equivalence, satisfiability, and implication, etc.* are not dependent on the variable assignment α (i.e., we can always ignore all variable assignments).

With closed formulae, α can be left out on the left side of the model relationship symbol:

$$I \models \varphi$$

Terminology

An interpretation I is called a model of φ under α if

$$I, \alpha \models \varphi$$

A PL1 formula φ can, as in propositional logic, be satisfiable, unsatisfiable, falsifiable, or valid.

Analogously, two formulae are logically equivalent ($\varphi \equiv \psi$) if for all I, α:

$$I, \alpha \models \varphi \text{ iff } I, \alpha \models \psi$$

Note: $P(x) \not\equiv P(y)$!

Logical Implication is also analogous to propositional logic.

Question: How can we define derivation?

Prenex Normal Form

Because of the quantifiers, we cannot produce the CNF form of a formula directly.

First step: Produce the prenex normal form

$$\underbrace{Qx_1 Qx_2 Qx_3 \ldots Qx_n}_{\text{quantifier prefix}} + \underbrace{\varphi}_{\text{(quantifier-free) matrix}}$$

Equivalences for the Production of Prenex Normal Form

$$\begin{aligned}
(\forall x \varphi) \wedge \psi &\equiv \forall x(\varphi \wedge \psi) \text{ if } x \text{ not free in } \psi \\
(\forall x \varphi) \vee \psi &\equiv \forall x(\varphi \vee \psi) \text{ if } x \text{ not free in } \psi \\
(\exists x \varphi) \wedge \psi &\equiv \exists x(\varphi \wedge \psi) \text{ if } x \text{ not free in } \psi \\
(\exists x \varphi) \vee \psi &\equiv \exists x(\varphi \vee \psi) \text{ if } x \text{ not free in } \psi \\
\forall x \varphi \wedge \forall x \psi &\equiv \forall x(\varphi \wedge \psi) \\
\exists x \varphi \vee \exists x \psi &\equiv \exists x(\varphi \vee \psi)
\end{aligned}$$

$$\begin{aligned}
\neg \forall x \varphi &\equiv \exists x \neg \varphi \\
\neg \exists x \varphi &\equiv \forall x \neg \varphi
\end{aligned}$$

... and propositional logic equivalents

Production of Prenex Normal Form

1. Eliminate $\Rightarrow$ and $\Leftrightarrow$
2. Move $\neg$ inwards
3. Move quantifiers outwards

Example:

$$\begin{aligned}
&\neg \forall x[(\forall x P(x)) \Rightarrow Q(x)] \\
\rightarrow\ &\neg \forall x[\neg(\forall x P(x)) \vee Q(x)] \\
\rightarrow\ &\exists x[(\forall x P(x)) \wedge \neg Q(x)]
\end{aligned}$$

And now?

Renaming of Variables

$\varphi[\frac{x}{t}]$ is obtained from φ by replacing all free appearances of x in φ by t.

Lemma: Let y be a variable that does not appear in φ. Then it holds that

$$\forall x\varphi \equiv \forall y\varphi[\tfrac{x}{y}] \text{ and } \exists x\varphi \equiv \exists y\varphi[\tfrac{x}{y}]$$

Theorem: There exists an algorithm that calculates the prenex normal form of any formula.

Derivation in PL1

Why is prenex normal form useful?

Unfortunately, there is no simple law as in propositional logic that allows us to determine satisfiability or general validity (by transformation into DNF or CNF).

But: we can reduce the satisfiability problem in predicate logic to the satisfiability problem in propositional logic. In general, however, this produces a very large number of propositional formulae (perhaps infinitely many)

Then: apply resolution.

Skolemization

Idea: Elimination of existential quantifiers by applying a function that produces the "right" element.

Theorem (Skolem Normal Form): Let φ be a closed formula in prenex normal form such that all quantified variables are pair-wise distinct and the function symbols g_1, g_2, ... do not appear in φ. Let

$$\varphi = \forall x_1 \cdots \forall x_i \exists y \psi,$$

then φ is satisfiable iff

$$\varphi' = \forall x_1 \cdots \forall x_i \psi \left[\frac{y}{g_i(x_1,\ldots,x_i)}\right]$$

is satisfiable.

Example: $\forall x \exists y[P(x) \Rightarrow Q(y)] \rightarrow \forall x[P(x) \Rightarrow Q(g(x))]$

Skolem Normal Form

Skolem Normal Form: Prenex normal form without existential quantifiers. Notation: φ^* is the SNF of φ.

Theorem: It is possible to calculate the Skolem normal form of every closed formula φ.

Example: $\exists x((\forall x P(x)) \wedge \neg Q(x))$ develops as follows:

$$\exists y((\forall x P(x)) \wedge \neg Q(y))$$

$$\exists y(\forall x(P(x) \wedge \neg Q(x)))$$

$$\forall x(P(x) \wedge \neg Q(g_0))$$

Note: This transformation is not an equivalence transformation; it only preserves satisfiability!

Note: ... and is not unique.

Ground Terms & Herbrand Expansion

The set of ground terms (or Herbrand Universe) over a set of SNF formulae $\theta*$ is the (infinite) set of all ground terms formed from the symbols of $\theta*$ (in case there is no constant symbol, one is added). This set is denoted by $D(\theta*)$.

The Herbrand expansion $E(\theta*)$ is the instantiation of the Matrix ψ_i of all formulae in $\theta*$ through all terms $t \in D(\theta*)$:

$$E(\theta*) = \{\psi_i[\frac{x_1}{t_1}, \dots, \frac{x_n}{t_n}] \mid (\forall x_1, \dots, x_n \psi_i) \in \theta*, t_j \in D(\theta*)\}$$

Theorem (Herbrand): Let $\theta*$ be a set of formulae in SNF. Then $\theta*$ is satisfiable iff $E(\theta*)$ is satisfiable.

Note: If $D(\theta*)$ and $\theta*$ are finite, then the Herbrand expansion is finite $\rightarrow$ finite propositional logic theory.

Note: This is used heavily in AI and works well most of the time!

Infinite Propositional Logic Theories

Can a finite proof exist when the set is infinite?

Theorem (compactness of propositional logic): A (countable) set of formulae of propositional logic is satisfiable if and only if every finite subset is satisfiable.

Corollary: A (countable) set of formulae in propositional logic is unsatisfiable if and only if a finite subset is unsatisfiable.

Corollary: (compactness of PL1): A (countable) set of formulae in predicate logic is satisfiable if and only if every finite subset is satisfiable.

Recursive Enumeration and Decidability

We can construct a semi-decision procedure for validity, i.e., we can give a (rather inefficient) algorithm that *enumerates* all valid formulae step by step.

Theorem: The set of valid (and unsatisfiable) formulae in PL1 is recursively enumerable.

What about *satisfiable* formulae?

Theorem (undecidability of PL1): It is undecidable, whether a formula of PL1 is valid.

(Proof by reduction from PCP)

Corollary: The set of satisfiable formulae in PL1 is not recursively enumerable.

In other words: If a formula is valid, we can effectively confirm this fact. Otherwise, we can end up in an infinite loop.

Derivation in PL1

Clausal Form instead of Herbrand Expansion.

Clauses are universally quantified disjunctions of literals; all variables are universally quantified

$(\forall x_1, \dots, x_n)(l_1 \vee \dots \vee l_n)$ written as

$l_1 \vee \dots \vee l_n$ or

$\{l_1, \dots, l_n\}$

Production of Clausal Form from SNF

Skolem Normal Form
quantifier prefix + (quantifier-free) matrix
$\forall x_1 \forall x_2 \forall x_3 \cdots \forall x_n \ \varphi$

1. Put Matrix into CNF using distribution rule
2. Eliminate universal quantifiers
3. Eliminate conjunction symbol
4. Rename variables so that no variable appears in more than one clause.

Theorem: It is possible to calculate the clausal form of every closed formula φ.

Note: Same remarks as for SNF

Conversion to CNF (1)

Everyone who loves all animals is loved by someone:

$$\forall x[\forall y Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y Loves(y,x)]$$

1. Eliminate biconditionals and implications

   $\forall x \neg[\forall y \neg Animal(y) \vee Loves(x,y)] \vee [\exists y Loves(y,x)]$

2. Move $\neg$ inwards: $\neg \forall x p \equiv \exists x \neg p$, $\neg \exists x p \equiv \forall x \neg p$

   $\forall x[\exists y \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y Loves(y,x)]$

   $\forall x[\exists y \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$

   $\forall x[\exists y Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$

Conversion to CNF (2)

3. Standardize variables: each quantifier should use a different one

   $\forall x[\exists y Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z Loves(z,x)]$

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

   $\forall x[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

5. Drop universal quantifiers:

   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

6. Distribute $\wedge$ over $\vee$:

   $[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Clauses and Resolution

Assumption: All formulae in the KB are clauses.

Equivalently, we can assume that the KB is a *set of clauses*.

Due to commutativity, associativity, and idempotence of $\vee$, clauses can also be understood as sets of literals. The empty set of literals is denoted by $\square$.

Set of clauses: Δ

Set of literals: C, D

Literal: l

Negation of a literal: $\bar{l}$

Propositional Resolution

$$\frac{C_1\dot{\cup}\{l\}, C_2\dot{\cup}\{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ are called resolvents of the parent clauses $C_1\dot{\cup}\{l\}$ and $C_2\dot{\cup}\{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not equivalent to the parent clauses, but it follows from them!

Notation: $R(\Delta) = \Delta \cup \{C \mid C$ is a resolvent of two clauses from $\Delta\}$

What Changes?

Examples

$\{\{Nat(s(0)), \neg Nat(0)\}, \{Nat(0)\}\} \vdash \{Nat(s(0))\}$

$\{\{Nat(s(0)), \neg Nat(x)\}, \{Nat(0)\}\} \vdash \{Nat(s(0))\}$

$\{\{Nat(s(x)), \neg Nat(x)\}, \{Nat(0)\}\} \vdash \{Nat(s(0))\}$

We need unification, a way to make literals identical.

Based on the notion of substitution, e.g., $\{\frac{x}{0}\}$.

Substitutions

A substitution $s = \{\frac{v_1}{t_1}, \ldots, \frac{v_n}{t_n}\}$ substitutes variables v_i for terms t_i (t_i does NOT contain v_i).

Applying a substitution s to an expression φ yields the expression φs which is φ with all occurrences of v_i replaced by t_i for all i.

Substitution Examples

$$P(x, f(y), B)$$

$$P(z, f(w), B) \quad s = \{\frac{x}{z}, \frac{y}{w}\}$$

$$P(x, f(A), B) \quad s = \{\frac{y}{A}\}$$

$$P(g(z), f(A), B) \quad s = \left\{\frac{x}{g(z)}, \frac{y}{A}\right\}$$

$$P(C, f(A), A)$$

Composing Substitutions

Composing substitutions s_1 and s_2 gives $s_1 s_2$ which is that substitution obtained by first applying s_2 to the terms in s_1 and adding remaining term/variable pairs (not occurring in s_1) to s_1.

Example: $\left\{\frac{z}{g(x,y)}\right\}\left\{\frac{x}{A},\frac{y}{B},\frac{w}{C},\frac{z}{D}\right\} = \left\{\frac{z}{g(A,B)},\frac{x}{A},\frac{y}{B},\frac{w}{C}\right\}$

Application example: $P(x,y,z) \rightarrow P(A,B,g(A,B))$

Properties of substitutions

For a formula φ and substitutions s_1, s_2

$(\varphi s_1)s_2 = \varphi(s_1 s_2)$

$(s_1 s_2)s_3 = s_1(s_2 s_3)$ associativity

$s_1 s_2 \neq s_2 s_1$ no commutativity!

Unification

Unifying a set of expressions $\{w_i\}$

Find substitution s such that $w_i s = w_j s$ for all i, j

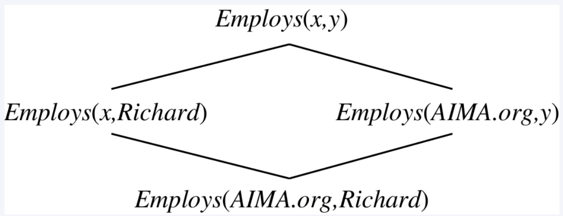
Example
$\{P(x, f(y), B), P(x, f(B), B)\}$
$s = \{\frac{y}{B}, \frac{z}{A}\}$ not the simplest unifier
$s = \{\frac{y}{B}\}$ most general unifier (mgu)

The most general unifier, the mgu, g of $\{w_i\}$ has the property that if s is any unifier of $\{w_i\}$ then there exists a substitution s' such that $\{w_i\}s = \{w_i\}gs'$

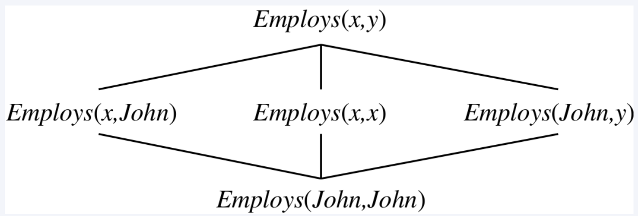
Property: The common instance produced is unique up to alphabetic variants (variable renaming)

Subsumption Lattice

a)



b)



Disagreement Set

The disagreement set of a set of expressions $\{w_i\}$ is the set of sub-terms $\{t_i\}$ of $\{w_i\}$ at the first position in $\{w_i\}$ for which the $\{w_i\}$ disagree

Examples

$\{P(x, A, f(y)), P(v, B, z)\}$ gives $\{x, v\}$

$\{P(x, A, f(y)), P(x, B, z)\}$ gives $\{A, B\}$

$\{P(x, y, f(y)), P(x, B, z)\}$ gives $\{y, B\}$

Unification Algorithm

UNIFY($Terms$):

Initialize $k \leftarrow 0$;
Initialize $T_k = Terms$;
Initialize $s_k = \{\}$;

*If T_k is a singleton, then output s_k. Otherwise continue.

Let D_k be the disagreement set of T_k.
If there exists a var v_k and a term t_k in D_k such that v_k does not occur in t_k, continue. Otherwise, exit with failure.

$s_{k+1} \leftarrow s_k\{\frac{v_k}{t_k}\}$;

$T_{k+1} \leftarrow T_k\{\frac{v_k}{t_k}\}$;

$k \leftarrow k + 1$;

Goto *.

Example

$\{P(x, f(y), y), P(z, f(B), B)\}$

Binary Resolution

$$\frac{C_1 \dot\cup \{l_1\}, C_2 \dot\cup \{\overline{l_2}\}}{[C_1 \cup C_2]s}$$

where $s = mgu(l_1, l_2)$, the most general unifier $[C_1 \cup C_2]s$ is the resolvent of the parent clauses $C_1 \dot\cup \{l_1\}$ and $C_2 \dot\cup \{\overline{l_2}\}$.

$C_1 \dot\cup \{l_1\}$ and $C_2 \dot\cup \{\overline{l_2}\}$ do not share variables l_1 and l_2 are the resolution literals.

Examples: $\{\{Nat(s(0)), \neg Nat(0)\}, \{Nat(0)\}\} \vdash \{Nat(s(0))\}$

$\{\{Nat(s(0)), \neg Nat(x)\}, \{Nat(0)\}\} \vdash \{Nat(s(0))\}$

$\{\{Nat(s(x)), \neg Nat(x)\}, \{Nat(0)\}\} \vdash \{Nat(s(0))\}$

Some Further Examples

Resolve $P(x) \vee Q(f(x))$ and $R(g(x)) \vee \neg Q(f(A))$

Standardizing the variables apart gives $P(x) \vee Q(f(x))$ and $R(g(y)) \vee \neg Q(f(A))$

Substitution $s = \{\frac{x}{A}\}$ Resolvent $P(A) \vee R(g(y))$

Resolve $P(x) \vee Q(x, y)$ and $\neg P(A) \vee \neg R(B, z)$

Standardizing the variables apart

Substitution $s = \{\frac{x}{A}\}$ and Resolvent $Q(A, y) \vee \neg R(B, z)$

Factoring

$$\frac{C_1 \dot\cup \{l_1\} \dot\cup \{l_2\}}{[C_1 \cup \{l_1\}]s}$$

where $s = mgu(l_1, l_2)$ is the most general unifier.

Needed because:

$$\{\{P(u), P(v)\}, \{\neg P(x), \neg P(y)\}\} \models \square$$

but $\square$ cannot be derived by binary resolution

Factoring yields:
$\{P(u)\}$ and $\{\neg P(x)\}$ whose resolvent is $\square$.

Derivations

Notation: $R(\Delta) = \Delta \cup \{C \mid C$ is a resolvent or a factor of two clauses from $\Delta\}$

We say D can be derived from Δ, i.e.,

$$\Delta \vdash D,$$

if there exist $C_1, C_2, C_3, \ldots, C_n = D$ such that $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\})$ for $1 \leq i \leq n$.

Example

From Russell and Norvig:

> The law says it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal.

Example

. . . it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono . . . has some missiles, i.e., $\exists x Owns(Nono, x) \wedge Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

. . . all of its missiles were sold to it by Colonel West.
$\forall x Missiles(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

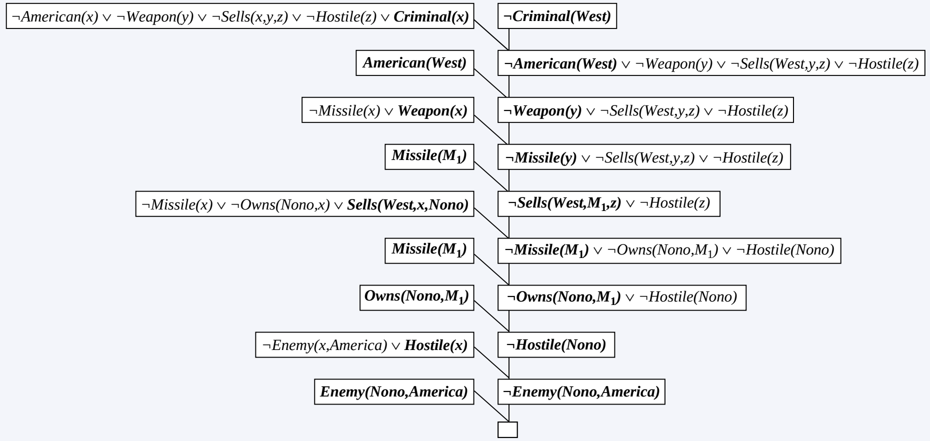
Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":
$Enemy(x, America) \Rightarrow Hostile(x)$

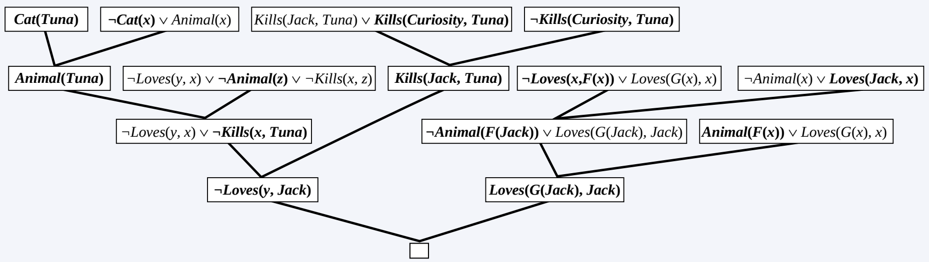
West, who is American . . .
$American(West)$

The country Nono, an enemy of America
$Enemy(Nono, America)$

An Example



Another Example



Properties of Resolution

Lemma: (soundness) If $\Delta \vdash D$, then $\Delta \models D$.

Lemma: resolution is refutation-complete:
Δ is unsatisfiable implies $\Delta \vdash \Box$.

Theorem: Δ is unsatisfiable iff $\Delta \vdash \Box$.

Technique: to prove that $\Delta \models C$
negate C and prove that $\Delta \cup \{\neg C\} \vdash \Box$.

The Lifting Lemma

Lemma: Let C_1 and C_2 be two clauses with no shared variables, and let C_1' and C_2' be ground instances of C_1 and C_2. If C' is a resolvent of C_1' and C_2', then there exists a clause such that

(1) C is a resolvent of C_1 and C_2

(2) C' is a ground instance of C

Can be easily generalized to derivations

The General Picture

Any set of sentences S is representable in clausal form

$\downarrow$

Assume S is unsatisfiable, and in clausal form

$\downarrow$ $\longleftarrow$ Herbrand's theorem

Some set S' of ground instances is unsatisfiable

$\downarrow$ $\longleftarrow$ Ground resolution theorem

Resolution can find a contradiction in S'

$\downarrow$ $\longleftarrow$ Lifting lemma

There is a resolution proof for the contradiction in S

Closing Remarks: Processing

- PL1-Resolution: forms the basis of
  - most state of the art theorem provers for PL1
  - the programming language Prolog
    - only Horn clauses
    - considerably more efficient methods.
  - not dealt with : search/resolution strategies
- Finite theories: In applications, we often have to deal with a fixed set of objects. Domain closure axiom:
  $$\forall x[x = c_1 \lor x = c_2 \lor \ldots \lor x = c_n]$$
  - Translation into finite propositional theory is possible.

Closing Remarks: Possible Extensions

- PL1 is definitely very expressive, but in some circumstances we would like more ...
- Second-Order Logic: Also over predicate quantifiers
  $$\forall x, y[(x = y) \Leftrightarrow \{\forall p[p(x) \Leftrightarrow p(y)]\}]$$
- Validity is no longer semi-decidable (we have lost compactness)
- Lambda Calculus: Definition of predicates, e.g., $\lambda x, y[\exists z P(x, z) \land Q(z, y)]$ defines a new predicate of arity 2
- Reducible to PL1 through Lambda-Reduction
- Uniqueness quantifier: $\exists! x \varphi(x)$ - there is exactly one x ...
- Reduction to PL1:
  $$\exists x[\varphi(x) \land \forall y\{\varphi(y) \Rightarrow x = y\}]$$

Summary

- PL1 makes it possible to structure statements, thereby giving us considerably more expressive power than propositional logic.
- Formulae consist of terms and atomic formulae, which, together with connectors and quantifiers, can be put together to produce formulae.
- Interpretations in PL1 consist of a universe and an interpretation function.
- The Herbrand Theory shows that satisfiability in PL1 can be reduced to satisfiability in propositional logic (although infinite sets of formulae can arise under certain circumstances).
- Resolution is refutation complete
- Validity in PL1 is not decidable (it is only semi-decidable)