Artificial Intelligence

5. First-Order Reasoning
On Computers that Think about How Objects Relate to Each Other

Jörg Hoffmann    Wolfgang Wahlster

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Summer Term 2012

Thanks to Profs. Burgard, Nebel, Riedmiller (Univ. Freiburg) for slide sources
Agenda

1. Motivation
2. Syntax and Semantics
3. Normal Forms
4. Reduction to Propositional Reasoning
5. PL1 Reasoning
6. PL1 Reasoning Examples
7. Closing Remarks
We can already do a lot with propositional logic. It does, however, have annoying limitations.

Example:

“All blocks are red.”
“A is a block.”

It should follow that

But propositional logic cannot handle this. Why?

Idea: Introduce variables ranging over objects, predicates, functions, . . .

→ First-Order Predicate Logic (PL1).
General Problem Solving using PL1

(some new problem)

model problem in first-order logic \(\mapsto\) use off-the-shelf reasoning tool

(its solution)

• “Any problem that can be formulated as drawing conclusions from a first-order formula”.

• Difference to previous chapter: “just” the language.

• In many applications, compile into propositional logic! (cf. later)
The Alphabet of First-Order Predicate Logic

Symbols:

- **Operators**: \(\neg, \lor, \land, \forall, \exists\)
- **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., \(block(.)\), \(above(.,.))\)

Predicate and function symbols have an **arity** (number of arguments).
- \(\rightarrow\) 0-ary predicate: propositional logic atoms.
- \(\rightarrow\) 0-ary function: **constant** (“object”).

- We suppose a countable set of predicates and functions of any arity.
The Grammar of First-Order Predicate Logic

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 Formulas (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.

Atomic formulas without variables: ground atoms.
The Grammar of First-Order Predicate Logic, ctd.

Formulas:

1. Every atomic formula, and $\top$ and $\bot$, is a formula.
2. If $\varphi$ and $\psi$ are formulas and $x$ is a variable, then
   
   \[ \neg \varphi, \varphi \land \psi, \varphi \lor \psi, \varphi \Rightarrow \psi, \varphi \Leftrightarrow \psi, \exists x \varphi \text{ and } \forall x \varphi \]

   are also formulas.

Is propositional logic part of the PL1 language?
## Alternative Notation

<table>
<thead>
<tr>
<th>Here</th>
<th>Elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\neg \varphi)</td>
<td>(\sim \varphi) (\bar{\varphi})</td>
</tr>
<tr>
<td>(\varphi \land \psi)</td>
<td>(\varphi &amp; \psi) (\varphi \bullet \psi) (\varphi, \psi)</td>
</tr>
<tr>
<td>(\varphi \lor \psi)</td>
<td>(\varphi</td>
</tr>
<tr>
<td>(\varphi \Rightarrow \psi)</td>
<td>(\varphi \rightarrow \psi) (\varphi \supset \psi)</td>
</tr>
<tr>
<td>(\varphi \Leftrightarrow \psi)</td>
<td>(\varphi \leftrightarrow \psi) (\varphi \equiv \psi)</td>
</tr>
<tr>
<td>(\forall x \varphi)</td>
<td>((\forall x) \varphi \land x\varphi)</td>
</tr>
<tr>
<td>(\exists x \varphi)</td>
<td>((\exists x) \varphi \lor x\varphi)</td>
</tr>
</tbody>
</table>

In line with previous notation: **Literal** \(l\) is possibly negated atomic formula; \(\bar{l}\) (rather than \(\neg l\)) denotes opposite literal.
Meaning of PL1-Formulas

Example: $\forall x [\text{Block}(x) \Rightarrow \text{Red}(x)], \text{Block}(a)$

For all objects $x$, if $x$ is a block, then $x$ is red. $a$ is a block.

More generally: (Intuition)

- Terms are interpreted as objects.
- Predicates represent subsets of the universe.
- Universally-quantified variables denote all objects in the universe.
- Existentially-quantified variables represent one of the objects in the universe.

→ Similar to propositional logic, we define interpretations, satisfiability, models, validity, ...
Semantics of PL1-Logic

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

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

**Interpretation** of ground terms:

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

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

$I \models P(t_1, \ldots, t_n)$ iff $\langle t_1^I, \ldots, t_n^I \rangle \in P^I$
Example “Blocks”

\[ D = \{ d_1, \ldots, d_n \mid n > 1 \} \]

\[ a^I = d_1 \]

\[ b^I = d_2 \]

\[ \text{Block}^I = \{ d_1 \} \]

\[ \text{Red}^I = D \]

\[ I \models \text{Red}(b)? \]

\[ I \models \text{Block}(b)? \]
Example “Integers”

\[ D = \{1, 2, 3, \ldots \} \]
\[ 1^I = 1 \]
\[ 2^I = 2 \]
\[ \ldots \]
\[ Even^I = \{2, 4, 6, \ldots \} \]
\[ succ^I = \{(1 \mapsto 2), (2 \mapsto 3), \ldots \} \]
\[ Equals^I = \{(1, 1), (2, 2), \ldots \} \]

\[ I \models Even(2) \]
\[ I \models Even(succ(2)) \]
\[ I \models Equals(x, succ(2)) \]
Semantics of PL1: Variable Assignment

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

**Notation:** $\alpha[x/d]$ is like $\alpha$ except at $x$, where $\alpha[x/d](x) =$

**Interpretation of terms** under $I, \alpha$:

\[
x^{I,\alpha} = \alpha(x) \\
\alpha^{I,\alpha} = \alpha^I \\
(f(t_1, \ldots, t_n))^{I,\alpha} = f^I(t_1^{I,\alpha}, \ldots, t_n^{I,\alpha})
\]

**Satisfaction of atomic formulas:**

\[
I, \alpha \models P(t_1, \ldots, t_n) \text{ iff } \langle t_1^{I,\alpha}, \ldots, t_n^{I,\alpha} \rangle \in P^I
\]
Example “Blocks”

\[ D = \{d_1, \ldots, d_n \mid n > 1\} \]
\[ a^I = d_1 \]
\[ b^I = d_2 \]
\[ c^I = \ldots \]
\[ Block^I = \{d_1\} \]
\[ Red^I = D \]
\[ \alpha = \{(x \mapsto d_1), (y \mapsto d_2)\} \]

\[ I, \alpha \models Red(x)? \]
\[ I, \alpha[y/d_1] \models Block(y)? \]
\[ I, \alpha[y/d_2] \models Block(y)? \]
An interpretation $I$ and a variable assignment $\alpha$ model formula $\varphi$ ($\varphi$ is satisfied by $I, \alpha$), written $I, \alpha \models \varphi$, under the following conditions:

- $I, \alpha \models \top$
- $I, \alpha \not\models \bot$
- $I, \alpha \models \neg \varphi$ iff $I, \alpha \not\models \varphi$

\[ \ldots \]

\[ \ldots \text{and all other propositional rules, as well as:} \]

- $I, \alpha \models P(t_1, \ldots, t_n)$ iff $\langle t_1^I, \alpha, \ldots, t_n^I, \alpha \rangle \in P^I, \alpha$
- $I, \alpha \models \forall x \varphi$ iff for all $d \in D$ we have $I, \alpha[x/d] \models \varphi$
- $I, \alpha \models \exists x \varphi$ iff there exists a $d \in D$ so that $I, \alpha[x/d] \models \varphi$
Example “Blocks”

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

Questions:

1. \( I, \alpha \models \text{Block}(b) \lor \neg \text{Block}(b) \)?
2. \( I, \alpha \models \text{Block}(x) \Rightarrow (\text{Block}(x) \lor \neg \text{Block}(y)) \)?
3. \( I, \alpha \models \text{Block}(a) \land \text{Block}(b) \)?
4. \( I, \alpha \models \forall x (\text{Block}(x) \Rightarrow \text{Red}(x)) \)?
5. \( I, \alpha \models \forall x (\text{Red}(x) \Rightarrow \text{Block}(x)) \)?
As in propositional logic: PL1 formula $\varphi$ is satisfiable, unsatisfiable, falsifiable, or valid.

Two formulas are logically equivalent ($\varphi \equiv \psi$) if:

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

Is $P(x) \equiv P(y)$?

→ Similar for logical implication.

How to derive new PL1 formulas?

→ Reasoning about satisfiability, on normal forms.
Questionnaire

Question!
Do you like being asked all these questions?
(A): Yes. (B): No.

Question!
Which of these are PL-1 formulas?
(A): \( \exists x[Even(x) \Rightarrow Even(succ(succ(x)))]. \)
(C): \( Even(1) \Rightarrow \forall x Equals(x, succ(x)). \)
(B): \( \exists x[Even(x) \Rightarrow succ(Even(succ(succ(x))))]. \)
(D): \( Even(1) \Rightarrow \forall 2 Equals(2, succ(2)). \)
Questionnaire, ctd.

**Question!**

Which of these are satisfied in Example “Integers”?

(A): \( \exists x [\text{Even}(x) \Rightarrow \text{Even}(\text{succ}(\text{succ}(x)))]. \)

(B): \( \exists x [\text{Even}(x) \Rightarrow \text{succ}(\text{Even}(\text{succ}(x)))] . \)

(C): \( \text{Even}(1) \Rightarrow \forall x \text{Equals}(x, \text{succ}(x)). \)

(D): \( \text{Even}(1) \Rightarrow \forall 2 \text{Equals}(2, \text{succ}(2)). \)
Free and Bound Variables

\[\forall x [R(y, z) \land \exists y (\neg P(y, x) \lor R(y, z))]\]

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

Formulas with no free variables are called closed formulas or sentences. Knowledge base (aka logical theory) = set of closed formulas.

Note: Closed formulas do not depend on the variable assignment \( \alpha \)!

Hence we omit \( \alpha \) on the left side of the model relationship symbol:

\[ I \models \varphi \]
How to produce a CNF form of a PL1 formula?

First step: Produce the prenex normal form.

\[ \text{quantifier prefix} + (\text{quantifier-free) matrix} \]

\[ Qx_1 Qx_2 Qx_3 \ldots Qx_n \varphi \]
Equivalences for the Production of Prenex Normal Form

\[(\forall x \varphi) \land \psi \equiv \forall x (\varphi \land \psi) \text{ if } x \text{ not free in } \psi\]

\[(\forall x \varphi) \lor \psi \equiv \forall x (\varphi \lor \psi) \text{ if } x \text{ not free in } \psi\]

\[(\exists x \varphi) \land \psi \equiv \exists x (\varphi \land \psi) \text{ if } x \text{ not free in } \psi\]

\[(\exists x \varphi) \lor \psi \equiv \exists x (\varphi \lor \psi) \text{ if } x \text{ not free in } \psi\]

\[\forall x \varphi \land \forall x \psi \equiv \forall x (\varphi \land \psi)\]

\[\exists x \varphi \lor \exists x \psi \equiv \exists x (\varphi \lor \psi)\]

\[\neg \forall x \varphi \equiv \exists x \neg \varphi\]

\[\neg \exists x \varphi \equiv \forall x \neg \varphi\]

... and propositional logic equivalences.
Production of Prenex Normal Form

1. Eliminate $\Rightarrow$ and $\Leftrightarrow$.
2. Move $\neg$ inwards.
3. Variable renaming.

**Example:** $\neg\forall x[(\forall x P(x)) \Rightarrow Q(x)]$

$\rightarrow$ Eliminate $\Rightarrow$ and $\Leftrightarrow$:

$\rightarrow$ Move $\neg$ inwards:

$\rightarrow$ Propositional equivalences:
Variable Renaming

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

**Lemma.** Let \( y \) be a variable that does not appear in \( \varphi \). Then it holds that

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

**Example:** \( \exists x[(\forall x P(x)) \land \neg Q(x)] \)

\( \rightarrow \) Rename \( \frac{x}{y} \) in \( [(\forall x P(x)) \land \neg Q(x)] \):

\( \rightarrow \) Move \( \forall x \) outwards:

**Theorem.** There exists an algorithm that, for any PL1 formula \( \varphi \), efficiently calculates an equivalent formula in prenex normal form. ("Efficiently": in polynomial time.)

**Proof.** We just outlined that algorithm.
Skolem Normal Form

**Idea:** Eliminate existential quantifiers by applying a function that *can* produce the “right” objects.

**Theorem (Skolem).** Let \( \varphi \) be a closed PL1 formula in prenex normal form such that all quantified variables are pair-wise distinct and the function symbol \( g \) does not appear in \( \varphi \). Let

\[
\varphi = \forall x_1 \cdots \forall x_i \exists y \psi.
\]

Then \( \varphi \) is satisfiable iff

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

is satisfiable.

→ To get rid of all existential quantifiers, iterate this step.

**Example.** \( \forall x \exists y [\text{dog}(x) \Rightarrow \text{chases}(x, y)] \) transformed to
Skolem Normal Form, ctd.

**Skolem Normal Form (SNF):** Prenex normal form without existential quantifiers. Notation: $\varphi^*$ is the SNF of $\varphi$.

**Example.** $\neg \forall x ((\forall x P(x)) \Rightarrow Q(x))$

→ Prenex normal form: $\exists y \forall x [P(x) \land \neg Q(y)]$.

→ SNF:

**Theorem.** *There exists an algorithm that, for any closed PL1 formula $\varphi$, efficiently calculates $\varphi^*$.*

**Proof.** We just outlined that algorithm.

Is $\varphi^*$ equivalent to $\varphi$?

Is $\varphi^*$ unique?
Questionnaire

**Question!**

Which are skolem normal forms of $\forall x \exists y \text{ Equals}(y, \text{succ}(\text{succ}(x)))$?

(A): $\forall x \exists y$

$$\text{Equals}(f(x), \text{succ}(\text{succ}(x)))$$.

(B): $\forall x$

$$\text{Equals}(f, \text{succ}(\text{succ}(x)))$$.

(C): $\forall x$

$$\text{Equals}(f(x), \text{succ}(\text{succ}(x)))$$.

(D): $\forall x$

$$\text{Equals}(g(x), \text{succ}(\text{succ}(x)))$$. 
Questionnaire, ctd.

Question!

Which are skolem normal forms of $\forall x \exists y \text{Equals}(x, \text{succ}(\text{succ}(y)))$?

(A): $\forall x \exists y$

$\text{Equals}(x, \text{succ}(\text{succ}(f(x))))$.

(B): $\forall x$

$\text{Equals}(x, \text{succ}(\text{succ}(f)))$.

(C): $\forall x$

$\text{Equals}(x, \text{succ}(\text{succ}(f(x))))$.

(D): $\forall x$

$\text{Equals}(x, \text{succ}(\text{succ}(g(x))))$. 
Why is the **Skolem normal form (SNF)** useful?

→ Together with **Herbrand expansion** (see next), reduce the satisfiability problem in predicate logic to the satisfiability problem in propositional logic!

→ Apply **resolution, DPLL, clause learning**, …

- May produce very large number of propositional formulas.
- Infinitely many, in general!
- Still, often works well.
model problem in first-order logic ⟷ reduce to SAT ⟷ use off-the-shelf SAT solver

“First-order logic as syntactic sugar for propositional logic.”

Remember all these propositions in the Wumpus world?

It's of course not that easy in general . . .
Ground Terms & Herbrand Expansion

The **Herbrand universe** $D(\theta^*)$ over a (finite) set of SNF formulas $\theta^*$ is the (possibly infinite) set of all ground terms that can be formed from the symbols present in $\theta^*$ (adding one constant symbol in case none is present).

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

$$E(\theta^*) = \{ \psi\left[\frac{x_1}{t_1}, \ldots, \frac{x_n}{t_n}\right] \mid (\forall x_1, \ldots, x_n \psi) \in \theta^*, t_j \in D(\theta^*) \}$$

**Theorem (Herbrand).** Let $\theta^*$ be a set of formulas in SNF. Then $\theta^*$ is satisfiable iff $E(\theta^*)$ is satisfiable. (Proof omitted.)

$\Rightarrow D(\theta^*)$ finite $\implies E(\theta^*)$ finite $\implies \text{propositional SAT!}$

$\Rightarrow$ But what if $D(\theta^*)$ (and thus $E(\theta^*)$) is infinite?
Infinite Propositional Logic Theories

Can a finite proof exist when the set of formulas is infinite?

**Theorem (Compactness of propositional logic).** A set of formulas of propositional logic is satisfiable if and only if every finite subset is satisfiable. (Proof omitted.)

**(Corollary (Compactness of PL1).** An infinite set of formulas in predicate logic is satisfiable if and only if every finite subset is satisfiable.)

**Proof.** “⇒”: Every finite θ₁ ⊆ θ satisfiable.
⇒ Every θ₁* satisfiable (T Skolem Normal Form “⇒”).
⇒ Every $E(θ₁*)$ satisfiable (T Herbrand “⇒”).
⇒ Every finite subset of $E(θ*)$ satisfiable ($E(θ*)$ is the union of the $E(θ₁*)$).
⇒ $E(θ*)$ satisfiable (T Compactness of propositional logic).
⇒ θ* satisfiable (T Herbrand “⇐”).
⇒ θ satisfiable (T Skolem Normal Form “⇐”).
So, can we use propositional reasoning to prove PL1 UNSAT?

**Corollary.** A set \( \theta \) of formulas in propositional logic is *unsatisfiable* if and only if at least one finite subset is unsatisfiable.

→ Given a PL1 formula set \( \theta \), enumerate all finite subsets \( \theta_1 \) of the Herbrand expansion \( E(\theta^*) \), and test propositional satisfiability of \( \theta_1 \). By Skolem, Herbrand, and compactness of propositional logic, if \( \theta \) is unsatisfiable then

Only \( \ldots \) which \( \theta_1 \) will do the job?

→ If the domain and Herbrand expansion are finite, all is fine with propositional reasoning. If not, to show unsatisfiability we must somehow choose a “relevant” subset of the Herbrand expansion.

→ More direct PL1 reasoning: “lifted”, with variables and terms.

→ And anyway, what about satisfiable cases?
Semi-Decidability of PL1

**Theorem.** *The set of unsatisfiable PL1 formulas is recursively enumerable.*

**Proof.** Enumerate all PL1 formulas \( \varphi \). Incrementally for all of these, enumerate all finite subsets \( \theta_1 \) of the Herbrand expansion \( E(\varphi^*) \), and test propositional satisfiability of \( \theta_1 \). By compactness of propositional logic, if \( E(\varphi^*) \) is unsatisfiable then one of the \( \theta_1 \) is.

**Corollary.** *The set of valid PL1 formulas is recursively enumerable.*

**Proof.** \( \varphi \) is valid iff

**Theorem (Undecidability of PL1).** *It is undecidable whether a formula of PL1 is valid.* (Proof omitted.)

**Corollary.** *The set of satisfiable PL1 formulas is*

→ If a PL1 formula is unsatisfiable, then we can confirm this. Otherwise, we might end up in an infinite loop.
Questionnaire

Question!

What is the Herbrand universe of \( \{ \text{Equals}(1, 2), \forall x \text{Equals}(f, \text{succ}(x)) \} \)?

(A): \( \{ f, 1, 2 \} \).

(C): \( \{ f, 1, 2, \text{succ}(1), \text{succ}(\text{succ}(1)), \ldots \} \).

(B): \( \{ f, 1, 2, \text{succ} \} \).

(D): \( \{ f, 1, 2, \text{succ}(f), \text{succ}(1), \text{succ}(2), \text{succ}(\text{succ}(f)), \ldots \} \).
Questionnaire, ctd.

**Question!**

What is the Herbrand expansion of \{\textit{Equals}(1, 2), \\
\forall x \textit{Equals}(f, \text{succ}(x))\}? Is this satisfiable?

**Question!**

Is \(\exists x [\textit{foo}(f(x)) \land \neg \textit{foo}(f(x))]\) satisfiable?
(some new problem)

model problem in first-order logic \(\rightarrow\) use PL1 resolution

(its solution)

- Resolution method native to first-order logic.
- Need to “match” (unify) the terms in the clauses …
PL1 Clauses

Clausal Form instead of Herbrand Expansion.

Clauses are universally quantified disjunctions of literals:

$$\forall x_1, \ldots, x_n (l_1 \lor \ldots \lor l_n)$$

→ Written as $l_1 \lor \ldots \lor l_n$ or $\{l_1, \ldots, l_n\}$. 
Production of Clausal Form from SNF

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

2. Put Matrix into CNF.

3. Drop universal quantifiers.

4. (Standardize variables apart.) Needed for resolution, see later.

**Theorem.** *There exists an algorithm that, for any closed PL1 formula $\varphi$, efficiently calculates its clausal form.*

Is the clausal form equivalent to $\varphi$?

Is the clausal form unique?
Conversion to CNF

\[ \forall x[\forall y (\text{Animal}(y) \Rightarrow \text{Loves}(x, y)) \Rightarrow \exists y \text{Loves}(y, x)] \]

Means what?

1. **Eliminate equivalences and implications:**

   \[ \forall x[\neg \forall y (\neg \text{Animal}(y) \lor \text{Loves}(x, y)) \lor \exists y \text{Loves}(y, x)] \]

2. **Move negation inwards:** (using \( \neg \forall p \equiv \exists x \neg p, \neg \exists p \equiv \forall x \neg p \))

   \[ \forall x[\exists y (\neg \text{Animal}(y) \lor \text{Loves}(x, y)) \lor \exists y \text{Loves}(y, x)] \]
   \[ \forall x[\exists y (\neg \text{Animal}(y) \land \neg \text{Loves}(x, y)) \lor \exists y \text{Loves}(y, x)] \]
   \[ \forall x[\exists y (\text{Animal}(y) \land \neg \text{Loves}(x, y)) \lor \exists y \text{Loves}(y, x)] \]
Conversion to CNF, ctd.

1. **Rename variables:** Each quantifier should use a different one.
   \[
   \forall x \left[ \exists y (Animal(y) \land \neg Loves(x, y)) \lor \exists z Loves(z, x) \right]
   \]

2. **Skolemize:** Replace existential variables by new function symbols.
   \[
   \forall x \left[ (Animal(f(x)) \land \neg Loves(x, f(x))) \lor Loves(g(x), x) \right]
   \]

3. **Distribute \( \land \) over \( \lor \):**
   \[
   \forall x \left[ (Animal(f(x)) \lor Loves(g(x), x)) \land (\neg Loves(x, f(x)) \lor Loves(g(x), x)) \right]
   \]

4. **Drop universal quantifiers (and write as formula set):**
   \[
   \{ Animal(f(x)) \lor Loves(g(x), x), \neg Loves(x, f(x)) \lor Loves(g(x), x) \}
   \]

5. **(Standardize variables apart:)** Needed for resolution, see later.
   \[
   \{ Animal(f(x)) \lor Loves(g(x), x), \neg Loves(y, f(y)) \lor Loves(g(y), y) \}
   \]
Reminder: Resolution Notation

**Assumption:** All formulas in the KB are in CNF (equivalently: the KB is one CNF formula).

→ KB is handled as a *set of clauses*. Each *clause* is handled as a *set of literals*.

**Notational conventions:**

- Set of clauses: \( \Delta \)
- Set of literals: \( C, D \)
- Empty set of literals: \( \Box \)
- Literal: \( l \); Inverse (negation) of a literal: \( \overline{l} \)

→ An interpretation \( I \) satisfies a clause \( C \) iff there exists \( l \in C \) such that \( I \models l \). \( I \) satisfies \( \Delta \) iff, for all \( C \in \Delta \), we have \( I \models C \).
Reminder: Propositional Resolution

With exclusive union \( \dot{\cup} \) meaning that the two sets are disjoint:

\[
\frac{C_1 \dot{\cup} \{l\}, C_2 \dot{\cup} \{\bar{l}\}}{C_1 \cup C_2}
\]

\( C_1 \cup C_2 \) is called the **resolvent** 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:** \( \{P, Q, \neg R\} \) resolves with \( \{P, S, R\} \) to \( \{P, Q, S\} \).

Is the resolvent equivalent to the parent clauses?

We say that \( C \) can be **derived** from \( \Delta \), \( \Delta \vdash C \), if \( C \) is the outcome of a sequence of applications of the resolution rule from \( \Delta \).
What Changes?

What about this:

\[
\{(\text{Nat}(s(0)), \neg\text{Nat}(0)), \{\text{Nat}(0)\}\} \models \{\text{Nat}(s(0))\}?
\]

\[
\{(\text{Nat}(s(0)), \neg\text{Nat}(0)), \{\text{Nat}(0)\}\} \vdash \{\text{Nat}(s(0))\}?
\]

And this?

\[
\{(\text{Nat}(s(x)), \neg\text{Nat}(x)), \{\text{Nat}(0)\}\} \models \{\text{Nat}(s(0))\}?
\]

\[
\rightarrow \text{But how to get}\ \{(\text{Nat}(s(x)), \neg\text{Nat}(x)), \{\text{Nat}(0)\}\} \vdash \{\text{Nat}(s(0))\}?
\]

We need **unification**, a way to make literals identical.

Based on the notion of **substitution**. Here: \(\{x \mapsto 0\}\).

\[
\rightarrow \text{Applying a substitution} \text{ specializes} \text{ the clause, which is valid because}\ 
\text{the variables are universally quantified.}
\]
Substitutions

**Definition.** A substitution \( s = \{ x_1 \mapsto t_1, \ldots, x_n \mapsto t_n \} \) is a function that substitutes variables \( x_i \) for terms \( t_i \), where \( x_i \neq t_i \) for all \( i \). Applying substitution \( s \) to an expression \( \varphi \) yields the expression \( \varphi s \), which is \( \varphi \) with all occurrences of \( x_i \) simultaneously replaced by \( t_i \).

\[ \rightarrow \] The variable renaming we used before is a special case of this.

**Example:** For \( s = \{ \frac{x}{y}, \frac{y}{h(a,b)} \} \), \( P(x, y)s = \)
Example: Can we apply a substitution to $P(x, f(y), b)$ so that it becomes:

1. $P(z, f(w), b)$?
2. $P(x, f(a), c)$?
3. $P(y, f(h(a, b, w)), b)$?
4. $Q(x, f(y), b)$?
5. $P(x, f(f(y)), b)$?
Composing Substitutions

**Definition.** Given substitutions $s_1$ and $s_2$, by $s_1 s_2$ we denote the composed substitution, a single substitution whose outcome is identical to $s_2 \circ s_1$.

**Example:** With $s_1 = \{ \frac{z}{g(x,y)}, \frac{v}{w} \}$ and $s_2 = \{ \frac{x}{a}, \frac{y}{b}, \frac{w}{v}, \frac{z}{d} \}$, we have $P(x,y,z)(s_2 \circ s_1) =$

We obtain $s_1 s_2$ by: (i) applying $s_2$ to the replacement terms $t_i$ in $s_1$; (ii) for any variable $x_i$ replaced by $s_2$ but not by $s_1$, applying the respective variable/term pair $\frac{x_i}{t_i}$ of $s_2$; and (iii) removing any pairs of variable $x$ and term $t$ where $x = t$.

**Example:** $\{ \frac{z}{g(x,y)}, \frac{v}{w} \} \{ \frac{x}{a}, \frac{y}{b}, \frac{w}{v}, \frac{z}{d} \} =$
Properties of Substitutions

For any formula $\varphi$ and substitutions $s_1, s_2, s_3$:

$$(\varphi s_1)s_2 = \varphi(s_1s_2)?$$

$$(s_1s_2)s_3 = s_1(s_2s_3)?$$

$$s_1s_2 = s_2s_1?$$

**Proposition.** A substitution $s = \{\frac{x_1}{t_1}, \ldots, \frac{x_n}{t_n}\}$ is idempotent, i.e., $\varphi ss = \varphi s$ for all $\varphi$, iff $t_i$ does not contain $x_j$ for $1 \leq i, j \leq n$.

**Proof.** “$\Leftarrow$”: The second application of $s$ does not do anything because all $x_i$ have been removed. “$\Rightarrow$”: if $t_i$ contains $x_j$ then the second application of $s$ replaces $x_j$ with $t_j \neq x_j$.

**Example:** For $s = \{\frac{x}{y}, \frac{y}{h(a,b)}\}$, $P(x, y)s = \neq P(x, y)ss = \ldots$
Unification

**Definition.** We say that a substitution $s$ is a unifier for a set of expressions $\{w_1, \ldots, w_k\}$ if $w_is = w_js$ for all $i, j$. (We’ll write $\{w_i\}$ for $\{w_1, \ldots, w_k\}$.)

**Example:** $\{P(x, f(y, z), b), P(x, f(b, w), b)\}$

$s = \{\frac{y}{b}, \frac{z}{w}, \frac{x}{h(a, b)}\}$?

$s = \{\frac{y}{b}, \frac{z}{w}\}$?

**Definition.** We say that $g$ is an mgu of $\{w_i\}$ if, for any unifier $s$ of $\{w_i\}$, there exists a substitution $s'$ such that $\{w_i\}s = \{w_i\}gs'$.

**Theorem.** If there exists a unifier $s$ of $\{w_i\}$, then there exists an idempotent mgu $g$ of $\{w_i\}$. (Proof omitted.)
When Unification is Impossible: Example (5)

**Example:** (5) on slide “Substitution Examples”. There exists a substitution $s$ so that $P(x, f(y), b)s = P(x, f(f(y)), b)$. But:

Can we unify $\{w_i\} = \{P(x, f(y), b), P(x, f(f(y)), b)\}$?

→ If the only way to unify $\{w_i\}$ is to unify a variable $x$ with a term $t$ that contains $x$, then $\{w_i\}$ cannot be unified.
Unification Algorithm: Disagreement Set

Our unification algorithm (next slide) makes use of this notion:

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:**

\[
\{P(x, c, f(y)), P(x, z, z)\}\?
\]

\[
\{P(x, a, f(y)), P(y, a, f(y))\}\?
\]

\[
\{P(v, f(z), g(w)), P(v, f(z), g(f(z)))\}\?
\]
Unification Algorithm

**Theorem.** If there exists a unifier of \( \{ w_i \} \), then the following algorithm returns an idempotent mgu of \( \{ w_i \} \). (Proof omitted.)

\[
k \leftarrow 0, \ T_k = \{ w_i \}, \ s_k = \{ \};
\]

**while** \( T_k \) is not a singleton **do**

\[
\text{Let } D_k \text{ be the disagreement set of } T_k;
\]

\[
/* \text{if } t_k \text{ contains } x_k \text{ then unification is impossible, cf. Example (5) */}
\]

\[
\text{Let } x_k, t_k \in D_k \text{ be a variable and term s.t. } t_k \text{ does not contain } x_k;
\]

\[
\text{if such } x_k, t_k \text{ do not exist then exit with output “failure”};
\]

\[
s_{k+1} \leftarrow s_k \{ \frac{x_k}{t_k} \}; \ /* t_k \text{ does not contain any of } x_1, \ldots, x_k */
\]

\[
T_{k+1} \leftarrow T_k \{ \frac{x_k}{t_k} \}; \ /* x_k \text{ does not occur in } T_{k+1 */}
\]

\[
k \leftarrow k + 1;
\]

**endwhile**

**exit** with output \( s_k \);
Unification Algorithm: An Example

\{P(x, f(y), y), P(z, f(b), b)\}

(Blackboard.)
With exclusive union $\dot{\cup}$ meaning that the two sets are disjoint:

$$\frac{C_1 \dot{\cup} \{A_1\}, C_2 \dot{\cup} \{\neg A_2\}}{[C_1 \cup C_2]g}$$

where the variables in $C_1 \dot{\cup} \{A_1\}$ and $C_2 \dot{\cup} \{\neg A_2\}$ are standardized apart, and $g$ is an mgu of $\{A_1, A_2\}$. $[C_1 \cup C_2]g$ is the resolvent of the parent clauses $C_i \dot{\cup} \{A_i\}$. $A_1$ and $\neg A_2$ are the resolution literals.

Is the resolvent equivalent to the parent clauses?

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

$$\{\{Nat(s(x)), \neg Nat(x)\}, \{Nat(0)\}\} \vdash$$
Standardizing Variables Apart: Example (5)

Example (5): **Cannot unify** \( \{ w_i \} = \{ P(x, f(y), b), P(x, f(f(y)), b) \} \).

**Modified Example:** Given the clauses \( C_1 = \{ P(x, f(y), b) \} \), \( C_2 = \{ \neg P(x, f(f(y)), b) \} \), say we first standardize the variables apart, getting \( C'_1 = \{ P(v, f(w), b) \} \) and \( C'_2 = C_2 \). The question then is:

**Can we unify** \( \{ w_i \} = \{ P(v, f(w), b), P(x, f(f(y)), b) \} \)?

But renaming variables is a substitution. Why does it work with standardizing variables apart, but not with unification?
Standardizing Variables Apart: “Who Knows Who”

Example: Clauses \{knows(John, x)\}, \{-knows(x, Elizabeth), king(x)\}. We should be able to conclude that?

Unification 1: \{w_i\} = \{knows(John, x), knows(x, Elizabeth)\}. Does there exist a unifier for \{w_i\}? 

Unification 2, after standardizing variables apart: \{w_i\} = \{knows(John, x), knows(y, Elizabeth)\}. Does there exist a unifier for \{w_i\}? 

(→ An alternative to standardize-variables-apart-then-apply-unification would be to substitute atoms separately to the same outcome; we don’t consider this here.)
Binary PL1 Resolution: Further Examples

\[ \text{Resolve } P(x) \lor Q(f(x)) \text{ and } R(g(x)) \lor \neg Q(f(a)) \]

Standardizing variables apart:

Substitution: \hspace{1cm} Resolvent:

\[ \text{Resolve } P(x) \lor Q(x, y) \text{ and } \neg P(a) \lor \neg R(b, z) \]

Standardizing variables apart:

Substitution: \hspace{1cm} Resolvent:
PL1 Resolution

With exclusive union $\dot{\cup}$ meaning that the two sets are disjoint:

$$\frac{C_1 \dot{\cup} \{A_1^1, \ldots, A_n^1\}, \ C_2 \dot{\cup} \{-A_1^2, \ldots, -A_m^2\}}{[C_1 \cup C_2]g}$$

where the variables in $C_1 \dot{\cup} \{A_1^1, \ldots, A_n^1\}$ and $C_2 \dot{\cup} \{-A_1^2, \ldots, -A_m^2\}$ are standardized apart, and $g$ is an mgu of $\{A_1^1, \ldots, A_n^1, A_1^2, \ldots, A_m^2\}$.

Why is this needed? **Example:** $\{\{P(u), P(v)\}, \{-P(x), -P(y)\}\}$.

Is this satisfiable?

Can we derive $\Box$ with binary PL1 resolution?

Can we derive $\Box$ with PL1 resolution?
With exclusive union $\hat{\cup}$ meaning that the two sets are disjoint:

$$
\frac{C_1 \hat{\cup} \{l_1\} \hat{\cup} \{l_2\}}{[C_1 \cup \{l_1\}]g}
$$

where $g$ is an mgu of $\{l_1, l_2\}$. $[C_1 \cup \{l_1\}]g$ is a factor of the parent clause $C_1 \hat{\cup} \{l_1\} \hat{\cup} \{l_2\}$.

**Example:** $\{\{P(u), P(v)\}, \{\neg P(x), \neg P(y)\}\}$.

How can we apply factoring?

Can we now derive $\square$ with binary PL1 resolution?
Properties of Resolution

Lemma **(Soundness)**. If $\Delta \vdash D$, then $\Delta \models D$.

Proof.

Lemma **(Refutation-Completeness)**. If $\Delta$ is unsatisfiable, then $\Delta \vdash \square$.

Proof?

Theorem. $\Delta$ is unsatisfiable iff $\Delta \vdash \square$.

Proof?

Use as before: To prove that $\Delta \models \varphi$, show that $\Delta \cup \{\neg \varphi\} \vdash \square$. 
The Lifting Lemma

**Lemma (Lifting 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 $C$ such that:

1. $C$ is a resolvent of $C_1$ and $C_2$; and
2. $C'$ is a ground instance of $C$.

**Proof Sketch.** If the ground instances $C'_1$ and $C'_2$ “match” for propositional resolution, then we can unify $C_1$ and $C_2$ to get the corresponding PL1 resolvent $C$.

Does this work with *binary* PL1 resolution?
Proof of Completeness

Any set of sentences $\theta$ is representable in clausal form.

\[\downarrow\]

Assume $\theta$ is unsatisfiable, and in clausal form.

\[\downarrow\]

←−−− Herbrand, prop. compactness

Some finite set $\theta'$ of ground instances is unsatisfiable.

\[\downarrow\]

←−−− Prop. resolution completeness

Propositional resolution can find a contradiction in $\theta'$.

\[\downarrow\]

←−−− Lifting lemma

PL1 resolution can find a contradiction in $\theta$. 
Questionnaire

Question!

PL1 resolution is sound and complete. Does it also terminate?

(A): Yes.  
(B): No.
Questionnaire, ctd.

Question!

Which of these expression pairs can be unified?

(A): $\text{dog}(x), \text{cat}(y)$.
(C): $\text{dog}(f(x)), \text{dog}(y)$.
(B): $\text{dog}(x), \text{dog}(y)$.
(D): $\text{dog}(g(x, f(a))),
\text{dog}(g(c, y))$.

Question!

In which cases is the second clause a factor of the first one?

(A): $\{\text{dog}(x), \text{dog}(c)\}, \{\text{dog}(c)\}$.
(C): $\{\text{dog}(x), \neg \text{dog}(x)\},
\{\text{dog}(x)\}$.
(B): $\{\text{dog}(x), \text{cat}(c)\}, \{\text{cat}(c)\}$.
(D): $\{\text{dog}(b), \text{dog}(c)\}, \{\text{dog}(c)\}$. 
Example “Integers”

\[ \forall x \exists y (\text{Equals}(x, \text{succ}(y))) \]\n
Is this satisfiable?

Axiomatizing \textit{succ}, \textit{Equals}:

d “1 is not the successor of anybody:” \[ \forall x \neg \text{Equals}(1, \text{succ}(x)) \]  

\textbf{Resolution refutation:} (Blackboard)
Example: Col. West

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.

(In what follows, for better readability we will sometimes write implications \( P \land Q \land R \Rightarrow S \) instead of clauses \( \neg P \lor \neg Q \lor \neg R \lor S \).)
Example: Col. West, ctd.

It is a crime for an American to sell weapons to hostile nations:
→ Clause: \( \text{American}(x) \land \text{weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \)

Nono has some missiles:
\( \exists x [\text{Owns}(\text{Nono}, x) \land \text{Missile}(x)] \) → SNF & Clauses: \( \text{Owns}(\text{Nono}, M_1), \text{Missile}(M_1) \)

All of Nono’s missiles were sold to it by Colonel West.
→ Clause: \( \text{Missiles}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \)

Missiles are weapons:
→ Clause: \( \text{Missile}(x) \Rightarrow \text{Weapon}(x) \)

An enemy of America counts as “hostile”:
→ Clause: \( \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \)

West is an American:
\( \text{American}(\text{West}) \)

The country Nono is an enemy of America:
\( \text{Enemy}(\text{Nono}, \text{America}) \)
Example: Col. West, ctd.

\[
\neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x) \\
\neg \text{Criminal}(\text{West}) \\
\text{American}(\text{West}) \\
\neg \text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \\
\text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \\
\text{Missile}(M_1) \\
\neg \text{Missile}(x) \lor \text{Weapon}(x) \\
\neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \\
\text{Missile}(M_1) \\
\neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono}) \\
\neg \text{Sells}(\text{West},M_1,z) \lor \neg \text{Hostile}(z) \\
\text{Missile}(M_1) \\
\neg \text{Missile}(y) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono}) \\
\text{Owns}(\text{Nono},M_1) \\
\neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono}) \\
\neg \text{Enemy}(x,\text{America}) \lor \text{Hostile}(x) \\
\neg \text{Hostile}(\text{Nono}) \\
\text{Enemy}(\text{Nono},\text{America}) \\
\neg \text{Enemy}(\text{Nono},\text{America})
\]
Curiosity killed the cat?

From Russell and Norvig:

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by noone.
Jack loves all animals.
Cats are animals.
Either Jack or curiosity killed the cat (whose name is “Tuna”).

Prove that curiosity killed the cat.

(In what follows, for better readability we will sometimes write implications \( P \wedge Q \wedge R \Rightarrow S \) instead of clauses \( \neg P \lor \neg Q \lor \neg R \lor S \).)
Curiosity killed the cat? Ctd.

Everyone who loves all animals is loved by someone:
\[\forall x [\forall y (\text{Animal}(y) \Rightarrow \text{Loves}(x, y)) \Rightarrow \exists z \text{Loves}(z, x)]\]
\[\rightarrow \text{SNF & Clauses:}\]
\[\text{Animal}(F(x)) \lor \text{Loves}(G(x), x), \neg\text{Loves}(x, F(x)) \lor \text{Loves}(G(x), x)\]

Anyone who kills an animal is loved by noone:
\[\forall x [\exists y (\text{Animal}(y) \land \text{Kills}(x, y)) \Rightarrow \forall z \neg\text{Loves}(z, x)]\]
\[\rightarrow \text{Clause: } \neg\text{Animal}(y) \lor \neg\text{Kills}(x, y) \lor \neg\text{Loves}(z, x)\]

Jack loves all animals:
\[\rightarrow \text{Clause: } \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)\]

Cats are animals:
\[\rightarrow \text{Clause: } \text{Cat}(x) \Rightarrow \text{Animal}(x)\]

Either Jack or curiosity killed the cat (whose name is “Tuna”):
\[\rightarrow \text{Clause: } \text{Kills}(\text{Jack}, \text{Tuna}) \lor \text{Kills}(\text{Curiosity}, \text{Tuna})\]
Curiosity killed the cat? Ctd.

\[
\begin{align*}
\text{Cat(Tuna)} & \quad \neg \text{Cat}(x) \lor \text{Animal}(x) \\
\text{Animal(Tuna)} & \quad \neg \text{Loves}(y, x) \lor \neg \text{Animal}(z) \lor \neg \text{Kills}(x, z) \\
\text{Kills(Jack, Tuna)} & \quad \text{Kills(Jack, Tuna)} \\
\text{Kills(Jack, Tuna)} & \quad \neg \text{Loves}(x, F(x)) \lor \text{Loves}(G(x), x) \\
\text{Animal(F)} & \quad \text{Animal}(x) \lor \text{Loves}(G(x), x) \\
\text{Loves}(G(Jack), Jack) & \quad \neg \text{Loves}(y, Jack) \\
\end{align*}
\]
Practical Aspects

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 here: search/resolution strategies.

Finite theories: In applications, we often have a fixed set of objects.
→ Translation into finite propositional theory is possible.
Further Extensions

PL1 is very expressive, but some people just can’t get enough . . .

- **Second-Order Logic:** Quantification over predicates.

\[
\forall x, y [\text{Equals}(x, y) \Leftrightarrow [\forall p (p(x) \Leftrightarrow p(y))]]
\]

Validity is no longer semi-decidable (we have lost compactness).

- **Lambda Calculus:** Definition of new predicates.

\[
\lambda x, y [\exists z P(x, z) \land Q(z, y)]
\]

Reducible to PL1 through Lambda-Reduction.

- **Temporal Logics:** Quantification over future behaviors.

\[
\text{AG}[\varphi \implies \text{EF} \psi]
\]

“For **A**ll futures, we **G**lobally have that, if \(s \models \varphi\), then there **E**xists a future from \(s\) on which **F**inally we have \(\psi\).”
Summary

- PL1 makes it possible to structure statements, thereby giving us considerably more expressive power than propositional logic.

- Formulas consist of terms and atomic formulas, which, together with connectors and quantifiers, form complex statements about relationships between objects.

- 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 (but in general requires infinite sets of formulas).

- PL1 Resolution is refutation-complete.

- Validity in PL1 is not decidable (it is only semi-decidable).