9. Predicate Logic

Syntax and Semantics, Normal Forms, Herbrand Expansion

Wolfram Burgard & Bernhard Nebel
Contents

• Motivation
• Syntax and Semantics
• Normal Forms
• Reduction to Propositional Logic: Herbrand Expansion
• Closing Remarks
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: \(\neg, \lor, \land, \forall, \exists, =\)
- 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( ), 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 $\varphi$ and $\psi$ are formulae and $x$ is a variable, then
   $\neg \varphi$, $\varphi \land \psi$, $\varphi \lor \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$, $\exists x \varphi$ 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

<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 \cdot \psi ) ( \varphi, \psi )</td>
</tr>
<tr>
<td>( \varphi \lor \psi )</td>
<td>( \varphi \mid \psi ) ( \varphi ; \psi ) ( \varphi + \psi )</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>
Meaning of PL1-Formulae

Our 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 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**: \( I = \langle D, \cdot^I \rangle \) where \( D \) is an arbitrary, non-empty set and \( \cdot^I \) is a function that

- maps any function symbols to functions over \( D \):
  \[ f^I \in [D^n \to D] \]

- maps individual constants to elements of \( D \):
  \[ a^I \in D \]

- maps any 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 \( P(t_1,\ldots,t_n) \):

\[ I \models P(t_1,\ldots,t_n) \iff \langle t_1^I,\ldots,t_n^I \rangle \in P^I \]
Example (1)

\[ D = \{d_1, \ldots, d_n \mid n > 1\} \]
\[ a' = d_1 \]
\[ b' = d_2 \]
\[ c' = \ldots \]
\[ Block' = \{d_1\} \]
\[ Red' = D \]
\[ I \models Red(b) \]
\[ I \not\models Block(b) \]
Example 2

\[ D = \{1, 2, 3, \ldots\} \]
\[ 1' = 1 \]
\[ 2' = 2 \]
\[ \vdots \]
\[ Even' = \{2, 4, 6, \ldots\} \]
\[ succ' = \{(1\rightarrow 2),(2\rightarrow 3), \ldots\} \]
\[ I \vDash Even(2) \]
\[ I \not\vDash Even(succ(2)) \]
Semantics of PL1: Variable Assignment

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

Notation: $\alpha[x/d]$ is the same as $\alpha$ up to point $x$. For $x: \alpha[x/d](x) = d$.

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

$$x^{I,\alpha} = \alpha(x)$$

$$a^{I,\alpha} = a'$$

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

**Satisfiability 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

\[ \alpha = \{(x \rightarrow d_1),(y \rightarrow d_2)\} \]

\[ l,\alpha \vdash \text{Red}(x) \]

\[ l,\alpha[y/d_1] \vdash \text{Block}(y) \]
Semantics of PL1: Satisfiability

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

\[
\begin{align*}
I, \alpha & \models T \\
I, \alpha & \not\models \bot \\
I, \alpha & \models \neg \varphi \text{ iff } I, \alpha \not\models \varphi \\
& \ldots
\end{align*}
\]

and all other propositional rules as well as

\[
\begin{align*}
I, \alpha & \models P(t_1, \ldots, t_n) \quad \text{iff} \quad \langle t_1, I, \alpha, \ldots, t_n, I, \alpha \rangle \in P \langle I, \alpha \rangle \\
I, \alpha & \models \forall x \varphi \quad \text{iff} \quad \text{for all } d \in D, I, \alpha[x/d] \models \varphi \\
I, \alpha & \models \exists x \varphi \quad \text{iff} \quad \text{there exists a } d \in D \text{ with } I, \alpha[x/d] \models \varphi
\end{align*}
\]
Example

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

Questions:

1. \( l, \alpha \models \text{Block}(b) \lor \neg \text{Block}(b) \) ?
2. \( l, \alpha \models \text{Block}(x) \Rightarrow (\text{Block}(x) \lor \neg \text{Block}(y)) \) ?
3. \( l, \alpha \models \text{Block}(a) \land \text{Block}(b) \) ?
4. \( l, \alpha \models \forall x \ (\text{Block}(x) \Rightarrow \text{Red}(x)) \) ?
5. \( l, \alpha \models \emptyset \) ?
Free and Bound Variables

∀x[R(\overline{y},\overline{z}) \land \exists y\neg P(y,x) \lor R(y,\overline{z})]

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 \(\alpha\) (i.e. we can always ignore all variable assignments).

With closed formulae, \(\alpha\) 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 $\phi$ under $\alpha$ if

$$I,\alpha \models \phi$$

A PL1 formula $\phi$ can, as in propositional logic, be **satisfiable**, **unsatisfiable**, **falsifiable**, or **valid**.

Analogously, two formulae are **logically equivalent** ($\phi \equiv \psi$), if for all $I,\alpha$:

$$I,\alpha \models \phi \iff I,\alpha \models \psi$$

**Note**: $P(x) \neq P(y)$!

**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

quantifier prefix + (quantifier-free) Matrix $\varphi$:

$$\forall x_1 \forall x_2 \forall x_3 \ldots \forall x_n \varphi$$
Equivalences for the Production of Prenex-Normal Form

\((\forall x \varphi) \land \psi \quad = \quad \forall x (\varphi \land \psi)\) if \(x\) not free in \(\psi\)

\((\forall x \varphi) \lor \psi \quad = \quad \forall x (\varphi \lor \psi)\) if \(x\) not free in \(\psi\)

\((\exists x \varphi) \land \psi \quad = \quad \exists x (\varphi \land \psi)\) if \(x\) not free in \(\psi\)

\((\exists x \varphi) \lor \psi \quad = \quad \exists x (\varphi \lor \psi)\) if \(x\) not free in \(\psi\)

\(\forall x \varphi \land \forall x \psi \quad = \quad \forall x (\varphi \land \psi)\)

\(\exists x \varphi \lor \exists x \psi \quad = \quad \exists x (\varphi \lor \psi)\)

\(\neg \forall x \varphi \quad = \quad \exists x \neg \varphi\)

\(\neg \exists x \varphi \quad = \quad \forall x \neg \varphi\)

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

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

**Example:**

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

$\to \neg \forall x[\neg(\forall x P(x)) \lor Q(x)]$

$\to \exists x [(\forall x P(x)) \land \neg Q(x)]$

and now? **Solution:** *Renaming of variables*

$\varphi[x/t]$ comes from $\varphi$, in which all free appearances of $x$ in $\varphi$ are replaced by the term $t$.

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

$\forall x \varphi \equiv \forall y \varphi[x/y]$ and $\exists x \varphi \equiv \exists y \varphi[x/y]$.

**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 $\varphi$ be a closed formula in prenex normal form such that all quantified variables are pair-wise distinct and the function symbols $g_1, g_2, \ldots$ do not appear in $\varphi$. Let

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

then $\varphi$ is satisfiable iff

$$\varphi' = \forall x_1 \ldots \forall x_i \psi[y/g_i(x_1, \ldots, x_i)]$$

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: $\varphi^*$ is the SNF of $\varphi$.

Theorem: It is possible to calculate the skolem normal form of every closed formula $\varphi$.

Example: $\exists x \ ((\forall x \ P(x)) \land \neg Q(x))$ develops as follows:
$\exists y \ ((\forall x \ P(x)) \land \neg Q(y))$
$\exists y \ (\forall x \ (P(x) \land \neg Q(y)))$
$\exists y \ (\forall x \ (P(x) \land \neg Q(y)))$
$\exists y \ (\forall x \ (P(x) \land \neg Q(y)))$
$\forall x \ (P(x) \land \neg Q(g_0))$

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

Note: … and is not unique.

Example: $\exists x \ (p(x)) \land \forall y \ (q(y))$
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 $\psi_i$ of all formulae in $\theta^*$ through all terms $t \in D(\theta^*)$:

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

**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.
Closing Remarks: Possible Extensions

PL1 is definitely very expressive, but in some circumstances we would like more…

• **Second-Order Logic**: Also over predicate quantifiers
  \[
  \forall x, y [(x=y) \iff \{\forall p[p(x) \iff p(y)]}\]
  \]
  → 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:
  \[
  \exists x [\varphi(x) \land \forall y \{\varphi(y) \Rightarrow x = y}\]
  \]
Closing Remarks: Processing

- **PL1-Resolution**: Execute resolution on clauses with variables instead of on the Herbrand expansion.
  - **Unification**
  - Resolution on classes of ground instances.
- **Restriction of the syntactical form**: Only Horn clauses
  - **PROLOG**
  - Considerably more efficient methods.
- **Finite theories**: In applications, we often have to deal with a fixed set of objects. **Domain closure axiom**:
  \[ \forall x \left[ x = c_1 \lor x = c_2 \lor \ldots \lor x = c_n \right] \]
  - Translation into finite propositional theory is possible.
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).
• Validity in PL1 is not decidable (it is only semi-decidable)