Artificial Intelligence

An introduction

Maik Kschischo

Institute for Computer Science
University of Koblenz

Table of Contents

Chap. 1 Overview and history of AI

Chap. 2 Propositional logic

Chap. 3 First order logic

Limitations of Propositional Logic

Example

$a =$ "Robot 7 is situated at the xy position (35, 79)" can be used as a propositional logic variable. However, to specify the position of 100 different robots on a grid of 100×100 we would need $100^3 = 10^6$ different propositional variables.

First order predicate logic (PL1) can overcome this and other limitations.

Syntax I

Terms label the objects we are considering (Domain $\mathcal{D}$). Terms are constants, variables and functions.

- **Constants:** are concrete objects
  Example: A certain machine, an alarm, a drug,...
  Notation: Lower case letters
- **Variables:** represent any object of the domain
  Notation: upper case letters $X, Y, \ldots$
- **Functions:** $f(t_1, \ldots, t_n)$ describe relations between objects. The arguments $t_1, \ldots, t_n$ are terms and f assigns these terms a value which is an object in the domain.
  Example: $\mathit{target}(X)$ might assign an edge X in a directed graph to its target node.
  Notation: Lower case italic letters.

Syntax II

Definition (Terms)

Let $\mathcal{K}$ be a set of constants, $\mathcal{V}$ be a set of variables and $\mathcal{F}$ be a set of function symbols. These sets are pairwise disjoint. We define the set of **terms** recursively:

- All variables and constants are (atomic) terms.
- If $t_1, \ldots, t_n$ are terms and f an n-place function symbol, then $f(t_1, \ldots, t_n)$ is also a term.

Syntax III

Predicates: are propositions about objects in the domain and describe properties. A predicate

$$P(t_1, \ldots, t_n)$$

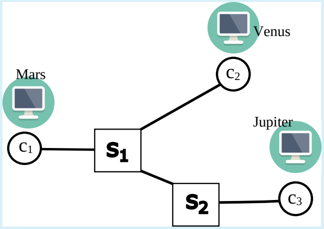
can be interpreted as the proposition: "The terms $t_1, \ldots, t_n$ have the property P."

- If all arguments $t_1, \ldots, t_n$ of P are constants, the predicate is either true T or false.
- If one argument t_i is a variable, the truth value of the predicate is not determined.

Syntax IV

Example

First order logic description of a computer network



Communication network consisting of three computers c_1, c_2, c_3 named Mars, Venus, Jupiter and two switches s_1, s_2.

- Constants: $\mathcal{K} = \{c_1, c_2, c_3, s_1, s_2, \text{Mars}, \text{Venus}, \text{Jupiter}\}$

Syntax V

Example

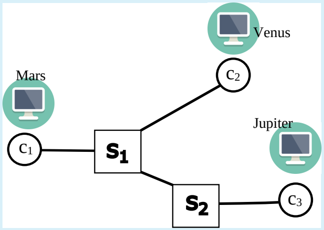
- The predicate $Computer(X)$ describes which elements are computers. Here

  $$\text{Computer(Mars)}$$
  $$\text{Computer(Venus)}$$
  $$\text{Computer(Jupiter}$$

  are three statements with truth value T.
- The predicate Edge(X, Y) describes a connection between the nodes corresponding to the variables X and Y. This is a predicate which has no defined truth value, since X and Y are variables.

Syntax VI

Example



- The statements

$$\begin{aligned}&\text{Edge}(c_1, s_1)\\&\text{Edge}(s_1, c_2)\\&\text{Edge}(s_1, s_2)\\&\text{Edge}(s_1, c_3)\end{aligned}$$

are all true, whereas Edge(c_1, s_2) is false.

Syntax VII

Example

- The function f can be used to assign network nodes to the names of the computers: $c_1 = f(\text{Mars}), c_2 = f(\text{Venus}), c_3 = f(\text{Jupiter})$.

Syntax VIII

Definition

Let **Predicate logic formulas** are built as follows:

- If $t_1, \dots, t_n$ are terms and P is a n-place predicate symbol, then $P(t_1, \dots, t_n)$ is an (atomic) formula.
- If a and b are formulas, then $\neg a, (a), a \vee b, a \wedge b, a \Rightarrow b, a \Leftrightarrow b$ are also formulas.
- If X is a variable and a a formula, the $\forall X\ a$ and $\exists X\ a$ are also formulas. Here, $\forall$ is the universal quantifier and $\exists$ is the existential quantifier.
- $P(t_1, \dots, t_n)$ and $\neg P(t_1, \dots, t_n)$ are called literals.

Quantifiers

- $\forall X$ reads "for all X"
- $\exists X$ reads "there exists X"

Syntax IX

Example

1. Let $N(X)$ denote the predicate "X is a natural number" and PositiveReal(X) be the predicate "X is a positive real number", then we have

   $\forall X\, N(X) \Rightarrow \text{PositiveReal}(X)$

   "For all X: If X is a natural number, then it is a positive real number"

2. For the existence quantor

   $\exists X\, \text{Edge}(c_1, X)$

   "There is a node having an incoming edge emanating from c_1."

Syntax X

Definition

Formulas in which every variable is in the scope of a quantifier are called **first-order sentences** or **closed formulas**. Variables which are not in the scope of a quantifier are called **free variables**.

Semantics I

In predicate logic, the meaning of formulas is recursively defined over the construction of the formula, in that we first assign constants, variables, and function symbols to objects in the real world.

Definition

An interpretation $\mathcal{I}$ is defined as

- A mapping from the set of constants and variables $\mathcal{K} \cup \mathcal{V}$ to a set $\mathcal{W}$ of names of objects in the world.
- A mapping from the set of function symbols to the set of functions in the world. Every $n-$place function symbol is assigned an $n-$place function.
- A mapping from the set of predicate symbols to the set of relations in the world. Every $n-$place predicate symbol is assigned an $n-$place relation.

Semantics II

Remark: Some authors use the term "arity of n" to indicate the number of argumentsin functions or predicates, instead of $n-$ place.

Semantics III

Example

Let k_1, k_2 and k_3 be constants. "plus" a two-place function symbol and gr a predicate symbol. The truth of the formula

$$p \equiv gr(plus(k_1, k_3), k_2)$$

depends on the interpretation $\mathcal{I}$. Let us choose one interpretation:

$$\mathcal{I}_1 : k_1 \mapsto 1,\ k_2 \mapsto 2,\ k_3 \mapsto 3,\ plus \mapsto +,\ gr \mapsto >$$

The formula above is mapped to

$$1 + 3 > 2 \qquad \text{or} \qquad 4 > 2$$

For all pairs $x > y$ with $x, y \in \{1, 2, 3, 4\}$ we can evaluate the th eformula $x > y$ and obtain the set $G = \{(4,3), (4,2), (4,1), (3,2), (3,1), (2,1)\}$. Since $(4,2) \in \mathbb{I}_1$ we can say that $p = T$ under the interpretation $\mathcal{I}_1$.

Example

For the interpretation

$$\mathcal{I}_2 : k_1 \mapsto 2,\ k_2 \mapsto 1,\ k_3 \mapsto 3,\ plus \mapsto -,\ gr \mapsto >$$

we find

$$2 - 3 - > 1 \qquad \text{or} \qquad -1 > 1$$

We see that $p = F$ under the interpretation $\mathcal{I}_2$.

Truth in PL1

Definition

- An atomic formula $P(t_1, \ldots, t_n)$ is *true* (or short T) under the interpretation $\mathcal{I}$ if, after interpretation and evaluation of all terms $t_1, \ldots, t_n$ and interpretation of the predicate P through the $n-$place relation r, it holds that
$$(\mathcal{I}(t_1), \ldots \mathcal{I}(t_n)) \in r$$
- The truth of quantifierless formulas follow from the truth of atomic formulas- as in propositional calculus—through the semantics of the logical operators "$\neg$", "$\wedge$", "$\vee$", "$\Rightarrow$" and "$\Leftrightarrow$".
- A formula $\forall X\ P$ is true under the interpretation $\mathcal{I}$ exactly when its is true given an arbitrary change of the interpretation for the variable X (and only for X).
- A formula $\exists X\ P$ is true under the interpretation $\mathcal{I}$ exactly when there is an interpretation for X which makes the formula true.

- Semantic equivalence, satisfiability, tautology, unsatisfiability, model and semantic entailment carry over from propositional calculus to first order logic.
- The deduction theorem and semantic entailment hold analogously.

Example: Family Tree I

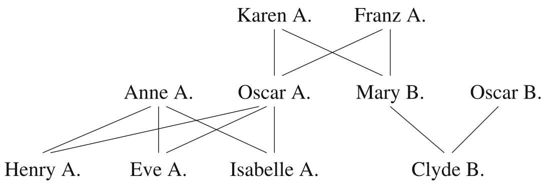


Fig. 3.1 A family tree. The edges going from Clyde B. upward to Mary B. and Oscar B. represent the element (Clyde B., Mary B., Oscar B.) as a child relationship

Figure: From Ertel, W., Introduction to AI, page 43

The tree represents family relationships. For example, the triple (Oscar A.,Karen A.,Franz A.) stands for the proposition "Oscar A. is a child of Karen A. and Franz A.".

Example: Family Tree II

All family relationships are expressed by the relation

$$\begin{aligned} \textit{child} = \{&(\text{Oscar A.,Karen A.,Franz A.}), (\text{Mary B.,Karen A.,Franz A.}),\\ &(\text{Henry A., Anne A., Oscar A.}), (\text{Eve A., Anne A., Oscar A.}),\\ &(\text{Isabelle A., Anne A., Oscar A.}), (\text{Clyde B., Mary B., Oscar B.})\}. \end{aligned}$$

The one-place relation

$$\textit{Female} = \{\text{Karen A., Anne A., Mary B., Eve A., Isabelle A.}\}$$

represents the female persons.

Example: Family Tree III

To establish formulas for family relationships we define a three-place predicate $\text{Child}(X, Y, Z)$ with the semantic

$$\mathcal{I}(\text{Child}(X, Y, Z)) = T \equiv (\mathcal{I}(X), \mathcal{I}(Y), \mathcal{I}(Z)) \in \textit{child}.$$

For the interpretation

$$\mathcal{I}(\textit{oscar}) = \text{Oscar A.}, \mathcal{I}(\textit{eve}) = \text{Eve A.}, \mathcal{I}(\textit{anne}) = \text{Anne A.}$$

the predicate $\text{Child}(\textit{eve}, \textit{oscar}, \textit{anne})$ is true.
The formula

$$\forall X \forall Y \forall Z\, \text{Child}(X, Y, Z) \Leftrightarrow \text{Child}(X, Z, Y)$$

expresses the symmetry of the predicate Child in the last two elements.
Therefore, $\text{Child}(\textit{eve}, \textit{anne}, \textit{oscar})$ is also true.

Example: Family Tree IV

The predicate Descendant can recursively be defined

$$\begin{aligned}&\forall X \forall Y \, \mathrm{Descendant}(X, Y) \Leftrightarrow \\ &\qquad \exists Z \, \mathrm{Child}(X, Y, Z) \vee (\exists U \exists V \, \mathrm{Child}(X, U, V) \wedge \mathrm{Descendant}(U, Y))\end{aligned}$$

Example: Family Tree V

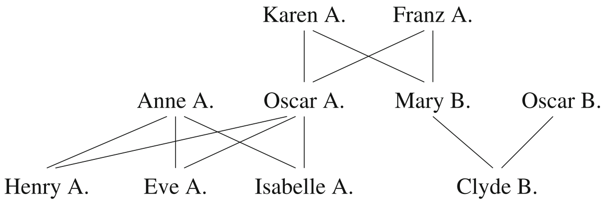


Fig. 3.1 A family tree. The edges going from Clyde B. upward to Mary B. and Oscar B. represent the element (Clyde B., Mary B., Oscar B.) as a child relationship

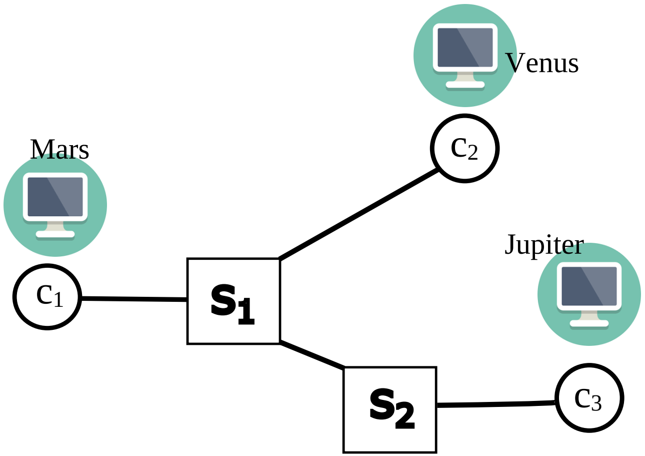
The knowledge about the familiy relationships in the above graph can be collected into the knowledge base KB.

Example: Family Tree VI

$$\begin{aligned}
KB \equiv\ & \text{Female}(\mathit{karen}) \wedge \text{Female}(\mathit{anne}) \wedge \text{Female}(\mathit{mary}) \wedge \text{Female}(\mathit{eve}) \\
& \wedge \text{Female}(\mathit{isabelle}) \\
& \wedge \text{Child}(\mathit{oscar}, \mathit{karen}, \mathit{franz}) \wedge \text{Child}(\mathit{mary}, \mathit{karen}, \mathit{franz}) \\
& \wedge \text{Child}(\mathit{eve}, \mathit{anne}, \mathit{oscar}) \wedge \text{Child}(\mathit{henry}, \mathit{anne}, \mathit{oscar}) \\
& \wedge \text{Child}(\mathit{isabelle}, \mathit{anne}, \mathit{oscar}) \wedge \text{Child}(\mathit{clyde}, \mathit{mary}, \mathit{oscarb}) \\
& \wedge (\forall X \forall Y \forall Z\, \text{Child}(X, Y, Z) \Leftrightarrow \text{Child}(X, Z, Y)) \\
& \wedge (\forall X \forall Y \text{Descendant}(X, Y) \Leftrightarrow \\
& \quad \exists Z \text{Child}(X, Y, Z) \vee (\exists U \exists V\, \text{Child}(X, U, V) \wedge \text{Descendant}(U, Y)))
\end{aligned}$$

We can now ask whether Child($\mathit{eve}, \mathit{oscar}, \mathit{anne}$) or Descendant($\mathit{eve}, \mathit{franz}$) can be derived from KB. This will be possible, when we have a calculus.

Example: Computer network (contd.) I



Example: Computer network (contd.) II

To connect two computers we need to have a path from the node of the first to the node of the second computer. We use the predicate Path:

$$\forall X \forall Y\, \text{Edge}(X, Y) \Rightarrow \text{Path}(X, Y)$$
$$\forall X \forall Y \forall Z\, \text{Path}(X, Y) \wedge \text{Edge}(Y, Z) \Rightarrow \text{Path}(X, Z)$$

The first formula states that two nodes X, Y are connected by a path, if there is an edge between them. The second formula state that two nodes X and Z are connected by a path, if there is a path between the nodes X, Y and an edge between Y and Z.
Two computers C_1 and C_2 are connected, if there is a path between their corresponding network nodes $f(C_1)$ and $f(R_2)$

$$\forall C_1 \forall C_2\ \text{Computer}(C_1) \wedge \text{Computer}(C_2) \wedge \text{Path}(C_1, C_2)$$
$$\Rightarrow \text{Connected}(C_1, C_2)$$

Example: Computer network (contd.) III

If we know that there is path beween two nodes we can also make the following statement

$$\forall X \forall Y (\text{Path}(X, Y) \Rightarrow \text{Edge}(X, Y) \lor \exists Z \text{Path}(X, Z) \land \text{Edge}(Z, Y))$$

expressing the fact there need to be certain edges in the graph, if there is is a path between X and Y.

Equality in PL1

We want compare terms and define equality via a predicate "=", which we could also write $eq(x, y)$. We define equality as

$$\forall X \; X = X \qquad \text{(reflexity)}$$
$$\forall X \forall Y \; X = Y \Rightarrow Y = X \qquad \text{(symmetry)}$$
$$\forall X \forall Y \forall Z \; X = Y \wedge Y = Z \Rightarrow X = Z \qquad \text{(transitivity)}$$

In addition we want functions to be unique. For a function f we have

$$\forall X \forall Y \; X = Y \Rightarrow f(x) = f(y) \qquad \text{substitution axiom for functions}$$

In a similar way we require for predicate symbols

$$\forall X \forall Y \; X = Y \Rightarrow P(x) = P(y) \qquad \text{substitution axiom for predicates}$$

Substitution of a variable by a term

Definition

We write $\Phi[X/t]$ for the formula that results when we replace every free occurrence of the variable X in Φ with the term t. Thereby we do not allow any variables in the term t that are quantified in Φ. In those cases variables must be renamed to ensure this.

Example

Consider the formula $\forall X\, X = Y$. Here, Y is a free variable. Replacing Y b the term $X + 1$ yields $\forall X\, X = X + 1$, which is incorrect. Instead, we correctly replace $\forall X\, X = Y + 1$ which has a different semantic.

Quantifiers and Normal Forms

The formula $\forall X\, P(X)$ is true, iff it is true for all interpretations of the variable X. One could write $P(c_1) \wedge P(c_2) \wedge \ldots \wedge P(c_n)$ for all constants $c_1, \ldots, c_n \in \mathcal{K}$. For $\exists X\, P(X)$ one could write $P(c_1) \vee P(c_2) \vee \ldots \vee P(c_n)$. From this we find

$$\forall X\, \Phi \equiv \neg \exists X\, \neg \phi$$

Accordingly, universal and exitential quantifiers are mutually replaceable.

Example

The propositions "Everbody wants to be loved." and "Nobody does not want to be loved" are equivalent.

Prenex normal form I

The quantifiers are not very easy to use for automatic reasoning, It is often useful to find equivalent formulas im a normal form.

Definition

A predicate logic formula Φ is in **prenex normal form** if for $Q_i \in \{\forall, \exists\},\ i = 1, \ldots, n$ we have

- $\Phi = Q_1 X_1 \cdots Q_n X_n\ \Psi$
- Ψ is a formular without any quantifiers.

Prenex normal form II

Examples

- Be careful with out of scope quantifiers: $\forall X\, A(X) \Rightarrow \exists X\, B(X)$. Instead, we should write
$$\forall X\, A(X) \Rightarrow \exists Y\, B(Y).$$
Here, we can bring the quantifier to the front
$$\forall X \exists Y\, A(X) \Rightarrow\ B(Y).$$

Prenex normal form III

Examples

- To bring the quantifiers in front of

$$(\forall X\, A(X)) \Rightarrow \exists Y\, B(Y)$$

we write the formula in equivalent form

$$\neg\,(\forall X\, A(X)) \vee \exists Y\, B(Y)$$

Now we replace the universal quantifier

$$(\exists X\, \neg A(X)) \vee \exists Y\, B(Y)$$

and pull the two quantifiers to the front

$$\exists X \exists Y\, \neg A(X) \vee\ B(Y).$$

Prenex normal form IV

Examples

This formula is equivalent to

$$\forall X \exists Y \, A(X) \Rightarrow B(Y).$$

Remarks:

- First eliminate implications so that there are no negations to the quantifiers.
- In general, one can only pull quantifiers out if negations only exist directly on atomic sub-formulas.

Prenex normal form V

Theorem

Every predicate logic formula can be transformed into an equivalent formula in prenex normal form.

Elimination of existential quantifiers I

Skolemization

Existential quantifiers can be eliminated by replacing the variables under the scope of $\exists$ by a function of the remaining variables. This is called **Skolemization**.

Example for illustration:

The formula

$$\forall U\,\forall V\,\exists X\,\forall W\,\exists Y \quad P(f(U),V,X) \lor Q(X,W,Y)$$

is in prenex normal form. The variable X is under the scope of the existential quantifier and depends on U and V, because these are still free at the level of this existential quantifier. We replace X by a Skolem function $g(U,V)$ (**new symbol!**) to eliminate the existential quantifier regarding correspondingt to X

$$\forall U\,\forall V\;\;\forall W\,\exists Y \quad P(f(U),V,g(U,V)) \lor Q(g(U,V),W,Y)$$

Elimination of existential quantifiers II

Skolemization

Analogously we replace Y by the skolem function $h(U, V, W)$

$$\forall U\, \forall V\; \forall W \quad P(f(U), V, g(U, V)) \lor Q(g(U, V), W, h(U, V, W))$$

Now there are only universal quantifiers left. Now, we leave them out in the notation and obtain

$$P(f(U), V, g(U, V)) \lor Q(g(U, V), W, h(U, V, W)).$$

When a formula in prenex normal form is skolemnized

Elimination of existential quantifiers III

Skolemization

- all existential quantifiers are eliminated from the outside inward, where a formula $\forall x_1 \dots \forall x_n \exists y\ \Phi$ is replaced by $\forall x_1 \dots \forall x_n\, \Phi\,(y/f(x_1, \dots x_n))$, where f is a Skolem function.
- If an existential quantifier is far outside, such as in $\exists y P(y)$, then Y must be replaced by constant (that is, by a zero-place function symbol).

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side
   - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side
   - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
     - Elimination of equivalences and implications.

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
    - Bring all the quantifiers to the left hand side
    - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
        - Elimination of equivalences and implications.
        - Repeated application of de Morgan's law and distributive law.

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side
   - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
     - Elimination of equivalences and implications.
     - Repeated application of de Morgan's law and distributive law.
   - Renaming of variables if necessary.

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side
   - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
     - Elimination of equivalences and implications.
     - Repeated application of de Morgan's law and distributive law.
   - Renaming of variables if necessary.
   - Factoring out universal quantifiers.

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side
   - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
     - Elimination of equivalences and implications.
     - Repeated application of de Morgan's law and distributive law.
   - Renaming of variables if necessary.
   - Factoring out universal quantifiers.
2. Skolemization

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
    - Bring all the quantifiers to the left hand side
    - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
        - Elimination of equivalences and implications.
        - Repeated application of de Morgan's law and distributive law.
    - Renaming of variables if necessary.
    - Factoring out universal quantifiers.
2. Skolemization
    - Replacement of existentially quantified variables by new Skolem functions.

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

1. Transformation into prenex normal form:
   - Bring all the quantifiers to the left hand side
   - Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
     - Elimination of equivalences and implications.
     - Repeated application of de Morgan's law and distributive law.
   - Renaming of variables if necessary.
   - Factoring out universal quantifiers.
2. Skolemization
   - Replacement of existentially quantified variables by new Skolem functions.
   - Deletion of resulting universal quantifiers.

Example I

Computer networks (contd.)

The formula

$$\forall X \forall Y (\mathrm{Path}(X, Y) \Rightarrow \mathrm{Edge}(X, Y) \lor \exists Z \mathrm{Path}(X, Z) \land \mathrm{Edge}(Z, Y))$$

expresses the fact there need to be certain edges in the graph, if there is is a path between X and Y.

1. Transformation into prenex normal form: Here we can shift all quantifiers to the left

$$\forall X \forall Y \exists Z \, (\mathrm{Path}(X, Y) \Rightarrow \mathrm{Edge}(X, Y) \lor \mathrm{Path}(X, Z) \land \mathrm{Edge}(Z, Y))$$

Example II

Computer networks (contd.)

Now we transform the expression without quantifiers into conjunctive normal form

$$\begin{aligned}
&\text{Path}(X, Y) \Rightarrow \text{Edge}(X, Y) \vee \text{Path}(X, Z) \wedge \text{Edge}(Z, Y)\\
&\equiv \text{Edge}(X, Y) \vee \text{Path}(X, Z) \wedge \text{Edge}(Z, Y) \vee \neg\text{Path}(X, Y)\\
&\equiv (\text{Edge}(X, Y) \vee \text{Path}(X, Z) \vee \neg\text{Path}(X, Y))\\
&\quad \wedge (\text{Edge}(X, Y) \vee \text{Edge}(Z, Y) \vee \neg\text{Path}(X, Y))
\end{aligned}$$

2. Skolemization To remove the existencial quantifier for the variable Z we replace $Z/g(X, Y)$

$$\begin{aligned}
&\text{Edge}(X, Y) \vee \text{Path}(X, g(X, Y)) \vee \neg\text{Path}(X, Y)\\
&\text{Edge}(X, Y) \vee \text{Edge}(g(X, Y), Y) \vee \neg\text{Path}(X, Y)
\end{aligned}$$

Example III

Computer networks (contd.)

Here, we do not explicitly write $\forall$, but the all quantifier is implicitly assumed as well as the conjunctive connection of both clauses.

Proof calculi

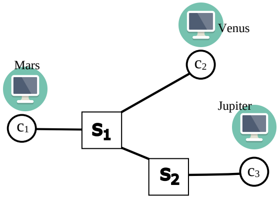
There are various proof calculi for PL1. We will focus on the resolution rule.

$$\frac{\begin{array}{l} A \vee B_1 \vee \ldots \vee B_n \\ \neg A \vee C_1 \vee \ldots \vee C_m \end{array}}{B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m}$$

which remains true for PL1. However, there we need to add a process called **unification**, as illustrated by the following example.

Example I

Computer networks (contd.)



- We want to show that there is a path from node c_1 to node s_1. We use the axioms (knowledge base)

$$\mathrm{Edge}(c_1, s_1)$$
$$\mathrm{Path}(X, Y) \vee \neg\mathrm{Edge}(X, Y)$$

where the last formula is equivalent to the definition $\forall X \forall Y\, \mathrm{Edge}(X, Y) \Rightarrow \mathrm{Path}(X, Y)$.

Example II

Computer networks (contd.)

- We can't apply the resolution rule, because $\text{Edge}(c_1, s_1)$ are $\neg\text{Edge}(X, Y)$ are not complementary.
- The last clause is only shorthand for $\forall X \forall Y\, \text{Path}(X, Y) \vee \neg\text{Edge}(X, Y)$ and is therefore valid for all interpretations.
- Therefore we can replace the variables by constants and we chose the replacements X/c_1 and Y/s_1 in the second clause.
- Now we can unify the two clauses to

$$\text{Edge}(c_1, s_1)$$
$$\text{Path}(c_1, s_1) \vee \neg\text{Edge}(c_1, s_1)$$

Example III

Computer networks (contd.)

- Resolution yields

$$\frac{\begin{array}{l}\text{Edge}(c_1, s_1)\\ \neg\text{Edge}(c_1, s_1) \vee \text{Path}(c_1, s_1)\end{array}}{\text{Path}(c_1, s_1)}$$

Unification I

Definition

Two literals are called **unifiable** if there is a substitution σ for all variables which makes the literals equal. Such a σ is called a **unifier**.

- Only variables can be substituted. Constants, function symbols or predicate name must not be replaced.
- A specific replacement of variables by a specific set of terms is called an **instantiation**.

Unification II

Example

The literals $P(X, f(Y))$ and $P(g(Z), f(a))$ can be unified in three different ways

$$\begin{aligned}
\sigma_1 :& \quad X/g(Z), && Y/a \\
\sigma_2 :& \quad X/g(b), && Y/a,\ Z/b \\
\sigma_3 :& \quad X/g(f(a)), && Y/a,\ Z/f(a)
\end{aligned}$$

and the unified formulas are

$$\begin{aligned}
P(X, f(Y))_{\sigma_1} &= P(g(Z), f(a)) = P(g(Z), f(a))_{\sigma_1} \\
P(X, f(Y))_{\sigma_2} &= P(g(b), f(a)) = P(g(Z), f(a))_{\sigma_2} \\
P(X, f(Y))_{\sigma_3} &= P(g(f(a)), f(a)) = P(g(Z), f(a))_{\sigma_3}
\end{aligned}$$

Unification III

Definition

A unifier is called the **most general unifier** if all other unifieres can be obtained from it by substitution of variables.

Example

For the literals $P(X, f(Y))$ and $P(g(Z), f(a))$ the unification $\sigma_1 : X/g(Z), Y/a$ is the most general unifier.

Unification IV

Unification algorithm:
Given: Two literals $A = P_A(s_1, \ldots, s_m)$ and $B = P_B(t_1, \ldots, t_m)$, which have different names for their variables.

1. Check the predicate names and their arity. If the names are different $P_A \neq P_B$ or the predicates have a different number of arguments (arity), then the literals can not be unified.
2. Unify the terms s_i and t_i for $i = 1, 2, \ldots, n$ in the following way:
   - If s_i and t_i are constants with the same name, then they are unified. If they have different names, the literals A and B can not be unified.
   - If s_i is a variable and t_i a term (or vice versa), then the substitution s_i/t_i unifies them.

Unification V

3. If $s_i = f(u_1, \dots, u_p)$ and $t_i = g(v_1, \dots, v_q)$ are functions, then they can only be unified if they have the same names ($f = g$) and the same arity ($p = q$). Then, the two functions are unified by unifyiung their arguments u_i and v_i for all $i = 1, \dots, p$ one ofter the other using the method of step (2).

Result: The most general unifier for the literals A and B

Resolution rule for PL1 I

Theorem

Let σ be the most general unification (MGU) of the literals A and $\neg D$, i.e. $A_\sigma = D_\sigma$. From the true clauses $A \vee B_1 \vee \ldots \vee B_n$ and $\neg D \vee C_1 \vee \ldots \vee C_n$ containing these literals we can derive $(B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m)_\sigma$:

$$\frac{\begin{array}{l} A \vee B_1 \vee \ldots \vee B_n \\ \neg D \vee C_1 \vee \ldots \vee C_n \qquad \textbf{with} \quad A_\sigma = D_\sigma \end{array}}{(B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m)_\sigma} \tag{1}$$

Remark: Both the parent clauses and the resolvent can contain variables.

Resolution rule for PL1 II

Theorem

The resolution rule is correct. That is, the resolvent is a semantic consequence of the two parent clauses.

For completeness, we need a small addition.

- It can happen, that some terms occur more than once in a clause or resolvent. For example, the resolvent could have the form $A \vee B \vee B \vee C$.
- Since $A \vee B \vee B \vee C \equiv A \vee B \vee C$ we can simplify this formula. This is called **factorization**.
- Factorization can also be applied to unified literals.

Resolution rule for PL1 III

Theorem (Factorization)

A clause can be factorized as

$$\frac{A \vee D \vee B_1 \ldots \vee B_n \qquad \textbf{with} \quad A_\sigma = D_\sigma}{(A \vee B_1 \ldots \vee B_n)_\sigma} \tag{2}$$

where σ is the MGU of A and D.

Resolution rule for PL1 IV

Example (Russell paradox)

"There is a barber who shaves everyone who does not shave himself."

- This statement ist unsatisfiable (Should the barber shave himsel?)
- Formalization in PL1

$$\forall X\ Shaves(barber, X) \Leftrightarrow \neg Shaves(X, X)$$

- Transform this formula into a clause

$$\begin{aligned}&(\neg Shaves(barber, X) \vee \neg Shaves(X, X))\\ &\quad \wedge (Shaves(barber, X) \vee Shaves(X, X))\end{aligned}$$

Resolution rule for PL1 V

Example (Russell paradox)

- From this, we can not derive a contradiction. First, we need to unify both clauses by $\sigma : X/barber$ and obtain

$$
\begin{aligned}
&(\neg Shaves(barber, X) \vee \neg Shaves(X, X))_{\sigma_1} \\
&\qquad \equiv (\neg Shaves(barber, barber) \vee \neg Shaves(barber, barber)) \\
&\qquad \equiv \neg Shaves(barber, barber) \\
&(Shaves(barber, X) \vee Shaves(X, X)) \\
&\qquad \equiv (Shaves(barber, barber) \vee Shaves(barber, barber)) \\
&\qquad \equiv Shaves(barber, barber)
\end{aligned}
$$

Resolution rule for PL1 VI

Example (Russell paradox)

- Now we can apply resolution

$$\frac{\begin{array}{l} \mathit{Shaves}(\mathit{barber}, \mathit{barber}) \\ \neg \mathit{Shaves}(\mathit{barber}, \mathit{barber}) \end{array}}{\emptyset}$$

Resolution rule for PL1 VII

Theorem

The resolution rule (1) *together with the factorization rule* (2) *is refutation complete. That is, by application of factorization and resolution steps, the empty clause can be derived from any unsatisfiable formula in conjunctive normal form.*

Decidability I

Remember the proof task:

Proof task

Given the axioms (knowledge base) and a proposition (query), proof the query (if possible).

The theorem shows, that the resolution calculus is **not decidable**:

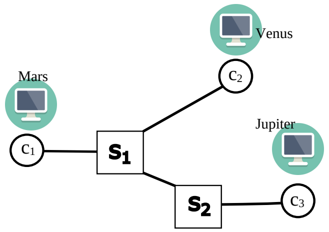
- Either, the empty clause can be generated (using proof by contradiction) after a finite number of steps and the query can be proofed, if it is a tautology (i.e. always true),
  or
- the query is false and the proof doesn't terminate. Then, it is not clear whether the query is false or whether the proof system needs further resolution steps.

Decidability II

- Even, if the negation of the query can be used to partially overcome the problem of non-terminating algorithms, it can not decide, whether the query is satisfiable, but not a tautology (i.e. not general).

Example: Communication between computers (contd.) I

Communication network consisting of three computers c_1, c_2, c_3 named Mars, Venus, Jupiter and two switches s_1, s_2.



Assertion: There is a connection between the computers Mars and Venus.

$Connected(Mars, Venus)$

Example: Communication between computers (contd.) II

Axioms:

$$Computer(Mars) \tag{3}$$
$$Computer(Venus) \tag{4}$$
$$Computer(Jupiter) \tag{5}$$
$$Edge(c_1, s_1) \tag{6}$$
$$Edge(s_1, c_2) \tag{7}$$
$$Edge(s_1, s_2) \tag{8}$$
$$Edge(s_2, c_3) \tag{9}$$
$$\neg Edge(X, Y) \vee Path(X, Y) \tag{10}$$
$$\neg Path(X, Y) \vee \neg Edge(Y, Z) \vee Path(X, Z) \tag{11}$$
$$\neg Computer(R_1) \vee \neg Computer(R_2) \vee \neg Path(f(R_1), f(R_2)) \vee Connected(R_1.R_2) \tag{12}$$

Example: Communication between computers (contd.) III

Here, f was the function which assigns node names to computer names (i.e. $c_1 = f(Mars)$).

1. We use the so called **set of support** strategy which adds the negated assertion to the axioms:

   $$\neg Connected(Mars, Venus)\,.$$

   We use the substitution

   $$\sigma_1 : R_1/Mars, R_2/Venus$$

   to unify

   $$Connected(Mars, Venus) \quad \text{and} \quad Connected(X, Y)$$

Example: Communication between computers (contd.) IV

The resulting complementary literals can be used in the first resolution step (see clause (12))

$$\frac{\begin{array}{l} Connected(R_1.R_2) \vee \neg Computer(R_1) \vee \neg Computer(R_2) \vee \neg Path(f(R_1), f(R_2)) \\ \neg Connected(Mars, Venus) \end{array}}{(\neg Computer(R_1) \vee \neg Computer(R_2) \vee \neg Path(f(R_1), f(R_2)))_{\sigma_1}}$$

which yields the resolvent (after unifiying substitution)

$$\neg Computer(Mars) \vee \neg Computer(Venus) \vee \neg Path(f(Mars), f(Venus)))$$

Example: Communication between computers (contd.) V

2. The last resolvent will be used in the next inference steps:

$$\frac{\begin{array}{l}\neg Computer(Mars) \vee \neg Computer(Venus) \vee \neg Path(f(Mars), f(Venus))) \\ Computer(Mars)\end{array}}{\neg Computer(Venus) \vee \neg Path(f(Mars), f(Venus)))}$$

and

$$\frac{\begin{array}{l}\neg Computer(Venus) \vee \neg Path(f(Mars), f(Venus)) \\ Computer(Venus)\end{array}}{\neg Path(f(Mars), f(Venus))}.$$

3. Using $c_1 = f(Mars)$ and $c_2 = f(Venus)$ we obtain $\neg Path(c_1, c_2)$. Now we use clause (10) and unify with the substitution

Example: Communication between computers (contd.) VI

$$\sigma_2 : X/c_1, \quad Y/c_2$$

to perform the next resolution step

$$\frac{\begin{array}{l}\neg Path(c_1, c_2) \\ Path(X, Y) \vee \neg Edge(X, Y)\end{array}}{(\neg Edge(X.Y))_{\sigma_2}} .$$

which yield after subsitution σ_2 the clause $Edge(c_1, c_2)$. There is, however, no axiom which can be used in a resolution step with this last clause.

Example: Communication between computers (contd.) VII

4. Therefore, we use clause (11) with a complementary literal. With the unification

$$\sigma_3 : X/c_1, \quad Z/c_2$$

we have

$$\frac{\begin{array}{l}\neg Path(c_1, c_2)\\ Path(X, Z) \vee \neg Path(X, Y) \vee \neg Edge(Y, Z)\end{array}}{(\neg Path(X, Y) \vee \neg Edge(Y, Z))_{\sigma_3}} .$$

which yields $\neg Path(c_1, Y) \vee \neg Edge(Y, c_2)$. Please note, that hier we have an example weher the resolvent and also the parent clauses contain a variable.

Example: Communication between computers (contd.) VIII

5. For the first literal in the last resolvent clause we find a complementary literal in clause (10). First, we need to rename the variables in (10)

$$\neg Edge(X_1, Y_1) \vee Path(X_1, Y_1)$$

and then we substitute

$$\sigma_4 : X_1/c_1, \quad Y_1/Y \, .$$

This yields a complementary literal for the last resolvent clause and we perform the resolution step

$$\frac{\begin{array}{c}\neg Path(c_1, Y) \vee \neg Edge(Y, c_2) \\ Path(c_1, Y) \vee \neg Edge(c_1, Y)\end{array}}{\neg Edge(Y, c_2) \vee Edge(c_1, Y)} \, .$$

Example: Communication between computers (contd.) IX

6. Now we substitute

$$\sigma_5 : Y/s_1$$

   and use clause (7)

$$\frac{\begin{array}{l}\neg Edge(s_1, c_2) \lor Edge(c_1, s_1)\\ Edge(s_1, c_2)\end{array}}{\neg Edge(c_1, s_1)}\,.$$

7. Using axiom (6) we can obtain the empty clause

$$\frac{\begin{array}{l}\neg Edge(c_1, s_1)\\ Edge(c_1, s_1)\end{array}}{\emptyset}$$

   and the assertion is proofed.

Chap. 4 Applications and limitations of logic in AI

Chap. 5 Probability theory and probabilistic logic

Chap. 6 Bayesian Networks

Chap. 7 Further Approaches