Deductive Databases and Logic Programming

(Winter 2003/2004)

Chapter 2: Basic Notions of Predicate Logic

- Signature, Formula
- Interpretation, Model
- Implication, Consistency, Some Equivalences
- Clausal Form, Herbrand Interpretations

Objectives

After completing this chapter, you should be able to:

- explain the basic notions: signature, interpretation, variable assignment, term, formula, model, consistent, implication.
- use some common equivalences to transform logical formulas.
- write formulas for given specifications.
- check whether a formula is true in an interpretation,
- find models of a given formula (if consistent).

Overview

1. Signature, Interpretation
2. Formulas, Models
3. Implication, Equivalence
4. Clausal Form
5. Herbrand Interpretations

Alphabet (1)

Definition:

- Let $ALPH$ be some infinite, but enumerable set, the elements of which are called symbols.

  Formulas will be words over $ALPH$, i.e. sequences of symbols.

- $ALPH$ must contain at least the logical symbols, i.e. $LOG \subseteq ALPH$, where

$$LOG = \{(,\ ),\ ,\ ,\ \wedge,\ \vee,\ \leftarrow,\ \rightarrow,\ \leftrightarrow,\ \forall,\ \exists\}.$$

- In addition, $ALPH$ must contain an infinite subset $VARS \subseteq ALPH$, the set of variables. This must be disjoint to LOG (i.e. $VARS \cap LOG = \emptyset$).

  Some authors consider variables as logical symbols.

Alphabet (2)

- E.g., the alphabet might consist of
  - the special logical symbols LOG,
  - variables starting with an uppercase letter and consisting otherwise of letters, digits, and "_",
  - identifiers starting with a lowercase letter and consisting otherwise of letters, digits, and "_".
- Note that words like "`father`" are considered as symbols (elements of the alphabet).

  Compare with: lexical scanner vs. context-free parser in a compiler.
- In theory, the exact symbols are not important.

Signatures (1)

Definition:

- A signature $\Sigma = (\mathcal{S}, \mathcal{P}, \mathcal{F}, \alpha, \rho)$ consists of:
  - ◇ A non-empty and finite set $\mathcal{S}$, the elements of which are called sorts (data type names).
  - ◇ $\mathcal{P} \subseteq \mathit{ALPH} - (\mathit{LOG} \cup \mathit{VARS})$, the elements are called predicate symbols.
  - ◇ $\mathcal{F} \subseteq \mathit{ALPH} - (\mathit{LOG} \cup \mathit{VARS} \cup \mathcal{P})$, the elements are called function symbols.
  - ◇ $\alpha\colon (\mathcal{P} \cup \mathcal{F}) \to \mathcal{S}^*$, which defines the argument sorts of predicates and functions.
  - ◇ $\rho\colon \mathcal{F} \to \mathcal{S}$, this defines the result sort of functions.

Signatures (2)

- If $\alpha(c) = \epsilon$ for some $c \in \mathcal{F}$ (i.e. c has no arguments), then c is called a constant.
- A predicate symbol $p \in \mathcal{P}$ with $\alpha(p) = \epsilon$ is called a propositional symbol.
- The length of the argument string (number of arguments) is called the arity of the function/predicate.
- The above definition is for a multi-sorted (typed) logic. One can also use an unsorted logic.

  Unsorted means really one-sorted. Then $\mathcal{S}$ and ρ are not needed, and α defines the arity, i.e. $\alpha\colon (\mathcal{P} \cup \mathcal{F}) \to \mathbb{N}_0$. E.g. Prolog uses an unsorted logic. This is also common in textbooks about mathematical logic.

Signatures (3)

Example:

- $\mathcal{S} = \{\texttt{person}, \texttt{string}\}$.
- $\mathcal{F}$ consists of
  - ◇ constants of sort `person`, e.g. `arno`, `birgit`, `chris`.
  - ◇ infinitely many constants of sort `string`, e.g. `''`, `'a'`, `'b'`, ..., `'Arno'`, ...
  - ◇ function symbols `first_name`: `person` $\longrightarrow$ `string` and `last_name`: `person` $\longrightarrow$ `string`.
- $\mathcal{P}$ consists of
  - ◇ a predicate `married_with`: `person` $\times$ `person`.

Interpretations (1)

Definition:

- Let a signature $\Sigma = (\mathcal{S}, \mathcal{P}, \mathcal{F}, \alpha, \rho)$ be given.
- A Σ-interpretation $\mathcal{I}$ defines:
  - ◇ a non-empty set $\mathcal{I}(s)$ for every $s \in \mathcal{S}$ (domain),
  - ◇ a relation $\mathcal{I}(p) \subseteq \mathcal{I}(s_1) \times \cdots \times \mathcal{I}(s_n)$ for every $p \in \mathcal{P}$, where $s_1, \ldots, s_n := \alpha(p)$,
  - ◇ a function $\mathcal{I}(f)\colon \mathcal{I}(s_1) \times \cdots \times \mathcal{I}(s_n) \to \mathcal{I}(s)$ for every $f \in \mathcal{F}$, where $s_1, \ldots, s_n := \alpha(f)$ and $s := \rho(f)$.
- In the following, we write $\mathcal{I}[\![\ldots]\!]$ instead of $\mathcal{I}(\ldots)$.

Interpretations (2)

Example:

- $\mathcal{I}[\![\texttt{person}]\!]$ is the set of Arno, Birgit, and Chris.
- $\mathcal{I}[\![\texttt{string}]\!]$ is the set of all strings, e.g. 'a'.
- $\mathcal{I}[\![\texttt{arno}]\!]$ is Arno.
- For the string constants, $\mathcal{I}$ is the identity mapping.

  If one has \-escapes and octal codes as in C, several constants are mapped to the same string. Or consider 0, -0, 00.

- $\mathcal{I}[\![\texttt{first\_name}]\!]$ maps e.g. Arno to 'Arno'.
- $\mathcal{I}[\![\texttt{last\_name}]\!]$ maps all three persons to 'Schmidt'.
- $\mathcal{I}[\![\texttt{married\_with}]\!] = \{(\text{Birgit}, \text{Chris}), (\text{Chris}, \text{Birgit})\}$.

Relation to Databases (1)

- A DBMS defines a set of data types, such as strings and numbers, together with constants, data type functions (e.g. $+$) and predicates (e.g. $<$).
- For these, the DBMS defines names (in the signature) and their meaning (in the interpretation).
- The DB schema in the relational model then adds further predicate symbols (relation symbols).
- The DB state interprets these by finite relations.

  Whereas the interpretation of the data types is fixed and built into the DBMS, the interpretation of the additional predicate symbols can be modified by insertions, deletions, and updates. But it must be finite.

Relation to Databases (2)

- In the Entity-Relationship-Model, the DB schema can introduce
  - ◇ new sorts (entity types),
  - ◇ new functions of arity 1 from entity types to data types (the attributes),
  - ◇ new predicates between entity types, possibly restricted to arity 2 (the relationships).
- The interpretation of the entity types (in the DB state) must always be finite.

  Thus, also attributes and relationships are finite.

Overview

1. Signature, Interpretation
2. Formulas, Models
3. Implication, Equivalence
4. Clausal Form
5. Herbrand Interpretations

Variable Declaration (1)

Definition:

- Let a signature $\Sigma = (\mathcal{S}, \mathcal{P}, \mathcal{F}, \alpha, \rho)$ be given.
- A variable declaration for Σ is a partial mapping $\nu: \mathit{VARS} \rightarrow \mathcal{S}$ (defined only for a finite subset of VARS).

Remark:

- The variable declaration is not part of the signature because it is locally modified by quantifiers (see below).
- The signature is fixed for the entire application, the variable declaration changes even within a formula.

Variable Declaration (2)

Definition:

- Let ν be a variable declaration, $X \in VARS$, and $s \in \mathcal{S}$.
- Then we write $\nu\langle X/s\rangle$ for the modified variable declaration ν' with

$$\nu'(V) := \begin{cases} s & \text{if } V = X \\ \nu(V) & \text{otherwise.} \end{cases}$$

Remark:

- Both is possible: ν might have been defined before for X or it might be undefined.

Terms (1)

Definition:

- Let a signature $\Sigma = (\mathcal{S}, \mathcal{P}, \mathcal{F}, \alpha, \rho)$ and a variable declaration ν for Σ be given.
- The set $TE_{\Sigma,\nu}(s)$ of terms of sort s is recursively defined as follows:
  - ◇ Every variable $V \in VARS$ with $\nu(V) = s$ is a term of sort s (this of course requires that ν is defined for V).
  - ◇ If t_1 is a term of sort s_1, ..., t_n is a term of sort s_n, and $f \in \mathcal{F}$ with $\alpha(f) = s_1 \dots s_n$ and $\rho(f) = s$, then $f(t_1, \dots, t_n)$ is a term of sort s.
  - ◇ Nothing else is a term.

Terms (2)

- In particular every constant c of sort s is a term of sort s. Formally, one would have to write "$c()$" for the term, but one simplifies the notation to "c".
- Certain functions are also written as infix operators, e.g. $\mathtt{X+1}$ instead of the official notation $\mathtt{+}(\mathtt{X}, \mathtt{1})$.
- Such "syntactic sugar" is not important for the theory of logic.

  In programming languages, there is sometimes a distinction between "concrete syntax" and "abstract syntax" (the syntax tree).

- Let $TE_{\Sigma,\nu} := \bigcup_{s \in \mathcal{S}} TE_{\Sigma,\nu}(s)$ be the set of all terms.

Atomic Formulas

Definition:

- Let a signature $\Sigma = (\mathcal{S}, \mathcal{P}, \mathcal{F}, \alpha, \rho)$ and a variable declaration ν for Σ be given.
- An atomic formula is an expression of the form

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

  where $p \in \mathcal{P}$, $\alpha(p) = s_1 \ldots s_n$, and $t_i \in TE_{\Sigma,\nu}(s_i)$ for $i = 1, \ldots, n$.

  Again, one sometimes uses infix notation in concrete syntax, e.g. $X > 1$.

- Let $AT_{\Sigma,\nu}$ be the set of atomic formulas for Σ, ν.

Formulas (1)

Definition:

- Let a signature $\Sigma = (\mathcal{S}, \mathcal{P}, \mathcal{F}, \alpha, \rho)$ and a variable declaration ν for Σ be given.
- The sets $FO_{\Sigma,\nu}$ of (Σ, ν)-formulas are defined recursively as follows:
  - ◇ Every atomic formula $\varphi \in AT_{\Sigma,\nu}$ is a formula.
  - ◇ If φ and ψ are formulas, so are $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \leftarrow \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$.
  - ◇ $(\forall X\colon s\ \varphi)$ and $(\exists X\colon s\ \varphi)$ are in $FO_{\Sigma,\nu}$ if $s \in \mathcal{S}$, $X \in VARS$, and φ is a formula for Σ and $\nu\langle X/s\rangle$.
  - ◇ Nothing else is a formula.

Formulas (2)

- The intuitive meaning of the formulas is as follows:
  - $\neg\varphi$: "Not φ" (φ is false).
  - $\varphi \wedge \psi$: "φ and ψ" (φ and ψ are both true).
  - $\varphi \vee \psi$: "φ or ψ" (at least one of φ and ψ is true).
  - $\varphi \leftarrow \psi$: "φ if ψ" (if ψ is true, φ must be true).
  - $\varphi \rightarrow \psi$: "if φ, then ψ"
  - $\varphi \leftrightarrow \psi$: "$\varphi$ if and only if ψ".
  - $\forall X\colon s\, \varphi$: "for all X (of sort s), φ is true".
  - $\exists X\colon s\, \varphi$: "there is an X (of sort s) such that φ".

Formulas (3)

- A Σ-formula is a (Σ, ν)-formula for any variable declaration ν.

  The variable declaration is local to the formula. If one considers a set of Σ-formulas, each formula can use a different variable declaration.

- Variants of predicate logic:
  - ◇ One can add atomic formulas "$true$" and "$false$" that are interpreted as true and false, resp.
  - ◇ One can add atomic formulas of the form $t_1 = t_2$ and ensure that is is really interpreted as equality.

Formulas (4)

- Above, many parentheses are used in order to ensure that formulas have a unique syntactic structure.

  For the formal definition, this is a simple solution, but for writing formulas in practical applications, the syntax becomes clumsy.

- One uses the following rules to save parentheses:
  - ◇ The outermost parentheses are never needed.
  - ◇ $\neg$ binds strongest, then $\wedge$, then $\vee$, then $\leftarrow$, $\rightarrow$, $\leftrightarrow$ (same binding strength), and last $\forall$, $\exists$.
  - ◇ Since $\wedge$ and $\vee$ are associative, no parentheses are required for e.g. $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$.

    Note that $\rightarrow$ and $\leftarrow$ are not associative.

Formulas (5)

Formal Treatment of Binding Strengths:

- A level 0 formula is an atomic formula or a level 5 formula enclosed in parentheses.

  The level of a formula corresponds to the binding strength of its outermost operator (smallest number means highest binding strength). However, one can use a level i-formula like a level j-formula with $j > i$. In the opposite direction, parentheses are required.

- A level 1 formula is a level 0 formula or a formula of the form $\neg\varphi$ with a level 1 formula φ.
- A level 2 formula is a level 1 formula or a formula of the form $\varphi_1 \wedge \varphi_2$ with a level 2 formula φ_1 and a level 1 formula φ_2.

Formulas (6)

Formal Treatment of Binding Strengths, Continued:

- A level 3 formula is a level 2 formula or a formula of the form $\varphi_1 \vee \varphi_2$ with a level 3 formula φ_1 and a level 2 formula φ_2.
- A level 4 formula is a level 3 formula or a formula of the form $\varphi_1 \leftarrow \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$ with level 3 formulas φ_1 and φ_2.
- A level 5 formula is a level 4 formula or a formula of the form $\forall X\colon s\, \varphi$ or $\exists X\colon s\, \varphi$ with a level 5 formula φ.
- A formula is a level 5 formula.

Variables in a Term

Definition:

- The function $vars$ computes the set of variables that occur in a given term t.
  - If t is a variable V:
  $$vars(t) := \{V\}.$$
  - If t has the form $f(t_1, \ldots, t_n)$:
  $$vars(t) := \bigcup_{i=1}^{n} vars(t_i).$$

Free Variables in a Formula

Definition:

- The function $free$ computes the set of free variables (not bound by a quantifier) in a formula φ:
  - If φ is an atomic formula $p(t_1, \ldots, t_n)$:
    $$free(\varphi) := \bigcup_{i=1}^{n} vars(t_i).$$
  - If φ has the form $(\neg\psi)$: $free(\varphi) := free(\psi)$.
  - If φ has the form $(\psi_1 \wedge \psi_2)$, $(\psi_1 \vee \psi_2)$, etc.: $free(\varphi) := free(\psi_1) \cup free(\psi_2)$.
  - If φ has the form $(\forall X\colon s\ \psi)$ or $(\exists X\colon s\ \psi)$: $free(\varphi) := free(\psi) - \{X\}$.

Closed and Ground Formulas

Definition:

- A formula φ is closed iff $free(\varphi) = \emptyset$.

  A closed formula may contain variables, but they are all bound by quantifiers.

- A formula or a term is ground iff it does not contain any variables (not even quantified ones).

  Every ground formula is closed, but the opposite is not in general true.

- A formula is propositional iff it contains only predicates without arguments and no quantifiers.

  Predicates without arguments are called propositional symbols. A propositional formula is always ground.

Uniqueness of Variable Sorts

- Let φ be a (Σ, ν)-formula. If $X \in free(\varphi)$, the variable X can have only a unique sort, namely $s = \nu(X)$.

  I.e. if ν_1 and ν_2 are two variable declarations such that φ is a (Σ, ν_i)-formula, and if $X \in free(\varphi)$, then $\nu_1(X) = \nu_2(X)$. The reason is that function and predicate symbols cannot be overloaded, thus, when the variable appears in an argument, it must have a certian sort. Note that in practice there might be exceptions, e.g. the predicate "$=$".

- For every (Σ, ν)-formula φ, there is a unique variable declaration $\nu' \subseteq \nu$ that is defined exactly on $free(\varphi)$ and such that φ is a (Σ, ν')-formula.
- Let $vdecl(\varphi)$ be this minimal variable declaration.

Abbreviations of Quantifiers

- If $X \in free(\varphi)$, one can leave out the sort in the quantfiers, because it is uniquely determined by φ. I.e. one can write $\forall X\, \varphi$ instead of $\forall X\colon s\; \varphi$.

  And the same for $\exists$. If $X \notin free(\varphi)$, the quantifier is anyway a bit strange. But in this case, if $\mathcal{I}[\![s]\!]$ can be empty, it would make a difference which sort is chosen, thus this abbreviation cannot be used.

- Instead of a sequence of quantifiers of the same type, e.g. $\forall X_1 \ldots \forall X_n\, \varphi$, one can write $\forall X_1, \ldots, X_n\, \varphi$.

- The universal closure of a formula φ, written $\forall(\varphi)$, is $\forall X_1 \ldots \forall X_n\, \varphi$, where $\{X_1, \ldots, X_n\} := free(\varphi)$.

Variable Assignment

Definition:

- A variable assignment $\mathcal{A}$ for $\mathcal{I}$ and ν is a partial mapping from $VARS$ to $\bigcup_{s \in \mathcal{S}} \mathcal{I}[\![s]\!]$.
- It maps every variable V, for which ν is defined, to a value from $\mathcal{I}[\![s]\!]$, where $s := \nu(V)$.

  I.e. a variable assignment for $\mathcal{I}$ and ν defines values from $\mathcal{I}$ for the variables that are declared in ν.

Definition:

- $\mathcal{A}\langle X/d \rangle$ denotes a variable assignment $\mathcal{A}'$ that agrees with $\mathcal{A}$ except that $\mathcal{A}'(X) = d$.

Value of a Term

Definition:

- Let a signature Σ, a variable declaration ν for Σ, a Σ-interpretation $\mathcal{I}$, and a variable assignment $\mathcal{A}$ for $(\mathcal{I}, \nu)$ be given.
- The value $(\mathcal{I}, \mathcal{A})[\![t]\!]$ of a term $t \in TE_{\Sigma,\nu}$ is defined as follows (recursion over the structure of the term):
  - ◇ If t is a variable V, then $(\mathcal{I}, \mathcal{A})[\![t]\!] := \mathcal{A}(V)$.
  - ◇ If t has the form $f(t_1, \ldots, t_n)$, then

    $$(\mathcal{I}, \mathcal{A})[\![t]\!] := \mathcal{I}[\![f]\!]\big((\mathcal{I}, \mathcal{A})[\![t_1]\!], \ldots, (\mathcal{I}, \mathcal{A})[\![t_n]\!]\big).$$

Truth of a Formula (1)

Definition:

- The truth value $(\mathcal{I}, \mathcal{A})[\![\varphi]\!] \in \{0, 1\}$ of a formula φ in $(\mathcal{I}, \mathcal{A})$ is defined as follows (0 means false, 1 true):
  - ◇ If φ is an atomic formula $p(t_1, \ldots, t_n)$:

$$(\mathcal{I}, \mathcal{A})[\![\varphi]\!] := \begin{cases} 1 & \text{if } \big((\mathcal{I}, \mathcal{A})[\![t_1]\!], \ldots, (\mathcal{I}, \mathcal{A})[\![t_n]\!]\big) \\ & \qquad \in \mathcal{I}[\![p]\!] \\ 0 & \text{else.} \end{cases}$$

  - ◇ If φ is of the from $(\neg\psi)$:

$$(\mathcal{I}, \mathcal{A})[\![\varphi]\!] := \begin{cases} 1 & \text{if } (\mathcal{I}, \mathcal{A})[\![\psi]\!] = 0 \\ 0 & \text{else.} \end{cases}$$

Truth of a Formula (2)

Definition, continued:

- Truth value of a formula, continued:
  - ◇ If φ is of the from $(\psi_1 \wedge \psi_2)$, $(\psi_1 \vee \psi_2)$, etc.:

ψ_1	ψ_2	$\wedge$	$\vee$	$\leftarrow$	$\rightarrow$	$\leftrightarrow$
0	0	0	0	1	1	1
0	1	0	1	0	1	0
1	0	0	1	1	0	0
1	1	1	1	1	1	1

E.g. if $(\mathcal{I}, \mathcal{A})[\![\psi_1]\!] = 1$ and $(\mathcal{I}, \mathcal{A})[\![\psi_2]\!] = 0$ then $(\mathcal{I}, \mathcal{A})[\![(\psi_1 \wedge \psi_2)]\!] = 0$.

Truth of a Formula (3)

Definition, continued:

- Truth value of a formula, continued:
  - If φ has the form $(\forall X\colon s\ \psi)$:

$$(\mathcal{I}, \mathcal{A})[\![\varphi]\!] := \begin{cases} 1 & \text{if } (\mathcal{I}, \mathcal{A}\langle X/d\rangle)[\![\psi]\!] = 1 \\ & \quad \text{for all } d \in \mathcal{I}[\![s]\!] \\ 0 & \text{else.} \end{cases}$$

  - If φ has the form $(\exists X\colon s\ \psi)$:

$$(\mathcal{I}, \mathcal{A})[\![\varphi]\!] := \begin{cases} 1 & \text{if } (\mathcal{I}, \mathcal{A}\langle X/d\rangle)[\![\psi]\!] = 1 \\ & \quad \text{for at least one } d \in \mathcal{I}[\![s]\!] \\ 0 & \text{else.} \end{cases}$$

Model (1)

Notation:

- If $(\mathcal{I}, \mathcal{A})[\![\varphi]\!] = 1$, one also writes $(\mathcal{I}, \mathcal{A}) \models \varphi$.

Definition:

- Let φ be a (Σ, ν)-formula. A Σ-interpretation $\mathcal{I}$ is a model of the formula φ (written $\mathcal{I} \models \varphi$) iff $(\mathcal{I}, \mathcal{A})[\![\varphi]\!] = 1$ for all variable declarations $\mathcal{A}$.

  I.e. free variables are treated as $\forall$-quantified. Of course, if φ is a closed formula, the variable declaration is not important.

- A Σ-interpretation $\mathcal{I}$ is a model of a set Φ of Σ-formulas, written $\mathcal{I} \models \Phi$, iff $\mathcal{I} \models \varphi$ for all $\varphi \in \Phi$.

Model (2)

Definition:

- A formula φ or set of formulas Φ is called consistent iff it has a model. Otherwise it is called inconsistent.
- A formula φ is called satisfiable iff there is an interpretation $\mathcal{I}$ and a variable declaration $\mathcal{A}$ such that $(\mathcal{I}, \mathcal{A}) \models \varphi$.
- A (Σ, ν)-formula φ is called a tautology iff for all Σ-interpretations $\mathcal{I}$ and (Σ, ν)-variable assignments $\mathcal{A}$: $(\mathcal{I}, \mathcal{A}) \models \varphi$.

Formulas in Databases (1)

- Consider a database with relations
  - $\mathtt{EMP}(\underline{\mathtt{EMPNO}}, \mathtt{ENAME}, \mathtt{SAL}, \mathtt{DEPTNO} \rightarrow \mathtt{DEPT})$
  - $\mathtt{DEPT}(\underline{\mathtt{DEPTNO}}, \mathtt{DNAME}, \mathtt{LOC})$
- Formulas can be used as queries: They ask for values for the free variables that make the formula true in the current database state (interpretation).
- E.g. print name and salary of all employees in the research department (X and Y are the free variables):

$$\exists\, \mathtt{E}, \mathtt{D}, \mathtt{L}\;\; \mathtt{emp}(\mathtt{E}, \mathtt{X}, \mathtt{Y}, \mathtt{D}) \wedge \mathtt{dept}(\mathtt{D}, \mathtt{'RESEARCH'}, \mathtt{L})$$

Formulas in Databases (2)

- In order to make the free variables, for which values are sought, better visible, domain calculus queries are usually written in the form:

$$\{\mathtt{X}, \mathtt{Y} \mid \exists\, \mathtt{E}, \mathtt{D}, \mathtt{L}\;\; \mathtt{emp}(\mathtt{E}, \mathtt{X}, \mathtt{Y}, \mathtt{D}) \land \mathtt{dept}(\mathtt{D}, \mathtt{'RESEARCH'}, \mathtt{L})\}$$

- One cannot use arbitrary formulas as queries. E.g. some formulas would generate an infinite answer:

$$\{\mathtt{D}, \mathtt{N} \mid \neg\mathtt{dept}(\mathtt{D}, \mathtt{N}, \mathtt{'NEW\ YORK'}\}$$

- Other formulas would require that infinitely many values are tried for quantified variables:

$$\exists\, \mathtt{X}\, \exists\, \mathtt{Y}\, \exists\, \mathtt{Z}\;\; \mathtt{X} * \mathtt{X} + \mathtt{Y} * \mathtt{Y} = \mathtt{Z} * \mathtt{Z}$$

Formulas in Databases (3)

- A formula is domain independent iff for all possible DB states (interpretations), it suffices to replace variables by values that appear in any argument of the DB relations or as ground term in the query.

  For a given interpretation $\mathcal{I}$ and formula φ, the "active domain" is the set of values that appear in database relations in $\mathcal{I}$ or as ground term (e.g. constant) in φ. Domain independence means that (1) φ must be false if a value outside this set is inserted for a free variable. (2) For all subformulas $\exists X\ \psi$, the formula ψ must be false if X has a value outside the active domain. (3) For all subformulas $\forall X\ \psi$, the formula ψ must be true if X has a value outside the active domain.

- Since all database relations are finite, queries can be evaluated in finite time.

Formulas in Databases (4)

- "Range restriction" is a syntactic (decidable) constraint on formulas that implies the domain independence (i.e. it is stricter than domain independence).

  For every formula, one defines the set of restricted variables in positive context and in negative context.

  E.g. if φ is an atomic formula $p(t_1, \ldots, t_n)$ with database relation p, then $posres(\varphi) := free(\varphi)$ and $negres(\varphi) := \emptyset$.
  For other atomic formulas, both sets are empty, except that when φ has the form $X = t$ with a ground term t, $posres(\varphi) := \{X\}$.
  If φ is $\neg\psi$, then $posres(\varphi) := negres(\psi)$ and $negres(\varphi) := posres(\psi)$.
  If φ has the form $\psi_1 \wedge \psi_2$, then $posres(\varphi) := posres(\psi_1) \cup posres(\psi_2)$ and $negres(\varphi) := negres(\psi_1) \cap negres(\psi_2)$. Etc.

  A formula φ is range restricted if $free(\varphi) = posres(\varphi)$ and for every subformula $\forall X\, \psi$, it holds that $X \in negres(\psi)$, and for every subformula $\exists X\, \psi$, it holds that $X \in posres(\psi)$.

Formulas in Databases (5)

- Exercise: Write these queries in domin calculus:
  - ◇ Which employees work in New York or Dallas and earn more than 3000 $ per month?
  - ◇ Which department has no employees?
  - ◇ Print for every department the employee(s) with the greatest salary in that department.
- Relations:
  - ◇ EMP(<u>EMPNO</u>, ENAME, SAL, DEPTNO → DEPT)
  - ◇ DEPT(<u>DEPTNO</u>, DNAME, LOC)

Formulas in Databases (6)

- Closed formulas can appear as boolean queries, but more often they are used as constraints, e.g.
  - ◇ DEPTNO is a key of DEPT(DEPTNO, DNAME, LOC):

$$\forall\, D, N_1, L_1, D_2, N_2, L_3$$
$$\mathtt{dept}(D, N_1, L_1) \land \mathtt{dept}(D, N_2, L_2) \rightarrow$$
$$N_1 = N_2 \land L_1 = L_2$$

  - ◇ DEPTNO in EMP is a foreign key that references DEPT:

$$\forall\, X, Y, Z, D\ \Big(\mathtt{emp}(X, Y, Z, D) \rightarrow \exists\, N, L\ \mathtt{dept}(D, N, L)\Big).$$

  - ◇ Exercise: Write a constraint that DEPTNO in DEPT must be greater than 0.

Overview

1. Signature, Interpretation
2. Formulas, Models
3. Implication, Equivalence
4. Clausal Form
5. Herbrand Interpretations

Implication

Definition/Notation:

- A formula or set of formulas Φ (logically) implies a formula or set of formulas ψ iff every model of Φ is also a model of ψ. In this case we write $\Phi \vdash \psi$.
- Many authors write $\Phi \models \psi$.

  The difference is important if one talks also about axioms and deduction rules. Then $\Phi \vdash \psi$ is used for syntactic deduction, and $\Phi \models \psi$ for the implication defined above via models. Correctness and completeness of the deduction system then mean that both relations agree.

Lemma:

- $\Phi \vdash \psi$ if and only if $\Phi \cup \{\neg\forall(\psi)\}$ is inconsistent.

Equivalence (1)

Definition/Lemma:

- Two Σ-formulas or sets of Σ-formulas φ_1 and φ_2 are equivalent iff they have the same models, i.e. for every Σ-interpretation $\mathcal{I}$:

$$\mathcal{I} \models \varphi_1 \iff \mathcal{I} \models \varphi_2.$$

- φ_1 and φ_2 are equivalent iff $\varphi_1 \vdash \varphi_2$ and $\varphi_2 \vdash \varphi_1$.

Lemma:

- "Equivalence" of formulas is an equivalence relation, i.e. it is reflexive, symmetric, and transitive.

  This also holds for strong equivalence defined on the next page.

Equivalence (2)

Definition/Lemma:

- Two (Σ, ν)-formulas φ_1 and φ_2 are strongly equivalent iff for every Σ-interpretation $\mathcal{I}$ and every $(\mathcal{I}, \nu)$-variable declaration $\mathcal{A}$:

$$(\mathcal{I}, \mathcal{A}) \models \varphi_1 \iff (\mathcal{I}, \mathcal{A}) \models \varphi_2.$$

- Strong equivalence of φ_1 and φ_2 is written: $\varphi_1 \equiv \varphi_2$.
- Suppose that ψ_1 results from ψ_2 by replacing a subformula φ_1 by φ_2 and let φ_1 and φ_2 be strongly equivalent. Then ψ_1 and ψ_2 are strongly equivalent.

Some Equivalences (1)

- Commutativity (for and, or, iff):
  - $\varphi \wedge \psi \;\equiv\; \psi \wedge \varphi$
  - $\varphi \vee \psi \;\equiv\; \psi \vee \varphi$
  - $\varphi \leftrightarrow \psi \;\equiv\; \psi \leftrightarrow \varphi$
- Associativity (for and, or, iff):
  - $\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \;\equiv\; (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$
  - $\varphi_1 \vee (\varphi_2 \vee \varphi_3) \;\equiv\; (\varphi_1 \vee \varphi_2) \vee \varphi_3$
  - $\varphi_1 \leftrightarrow (\varphi_2 \leftrightarrow \varphi_3) \;\equiv\; (\varphi_1 \leftrightarrow \varphi_2) \leftrightarrow \varphi_3$

Some Equivalences (2)

- Distribution Law:
  - $\varphi \wedge (\psi_1 \vee \psi_2) \equiv (\varphi \wedge \psi_1) \vee (\varphi \wedge \psi_2)$
  - $\varphi \vee (\psi_1 \wedge \psi_2) \equiv (\varphi \vee \psi_1) \wedge (\varphi \vee \psi_2)$
- Double Negation:
  - $\neg(\neg\varphi) \equiv \varphi$
- De Morgan's Law:
  - $\neg(\varphi \wedge \psi) \equiv (\neg\varphi) \vee (\neg\psi)$.
  - $\neg(\varphi \vee \psi) \equiv (\neg\varphi) \wedge (\neg\psi)$.

Some Equivalences (3)

- Replacements of Implication Operators:
  - ◇ $\varphi \leftrightarrow \psi \;\equiv\; (\varphi \rightarrow \psi) \wedge (\varphi \leftarrow \psi)$
  - ◇ $\varphi \leftarrow \psi \;\equiv\; \psi \rightarrow \varphi$
  - ◇ $\varphi \rightarrow \psi \;\equiv\; \neg\varphi \vee \psi$
  - ◇ $\varphi \leftarrow \psi \;\equiv\; \varphi \vee \neg\psi$
- Together with De Morgan's Law this means that e.g. $\{\neg, \vee\}$ are sufficient, all other logical junctors $\{\wedge, \leftarrow, \rightarrow, \leftrightarrow\}$ can be expressed with them.

  As we will see, also only one of the quantifiers is needed.

Some Equivalences (4)

- Replacements for Quantifiers:
  - $\forall X{:}\, s\; \varphi \;\equiv\; \neg(\exists X{:}\, s\; (\neg\varphi))$
  - $\exists X{:}\, s\; \varphi \;\equiv\; \neg(\forall X{:}\, s\; (\neg\varphi))$
- Moving logical junctors over quantifiers:
  - $\neg(\forall X{:}\, s\; \varphi) \;\equiv\; \exists X{:}\, s\; (\neg\varphi)$
  - $\neg(\exists X{:}\, s\; \varphi) \;\equiv\; \forall X{:}\, s\; (\neg\varphi)$
  - $\forall X{:}\, s\; (\varphi \wedge \psi) \;\equiv\; (\forall X{:}\, s\; \varphi) \wedge (\forall X{:}\, s\; \psi)$
  - $\exists X{:}\, s\; (\varphi \vee \psi) \;\equiv\; (\exists X{:}\, s\; \varphi) \vee (\exists X{:}\, s\; \psi)$

Some Equivalences (5)

- Moving quantifiers: If $X \notin free(\varphi)$:
  - $\forall X\colon s\ (\varphi \vee \psi) \ \equiv\ \varphi \vee (\forall X\colon s\ \psi)$
  - $\exists X\colon s\ (\varphi \wedge \psi) \ \equiv\ \varphi \wedge (\exists X\colon s\ \psi)$

  If in addition $\mathcal{I}[\![s]\!]$ cannot be empty:
  - $\forall X\colon s\ (\varphi \wedge \psi) \ \equiv\ \varphi \wedge (\forall X\colon s\ \psi)$
  - $\exists X\colon s\ (\varphi \vee \psi) \ \equiv\ \varphi \vee (\exists X\colon s\ \psi)$
- Removing unnecessary quantifiers:

  If $X \notin free(\varphi)$ and $\mathcal{I}[\![s]\!]$ cannot be empty:
  - $\forall X\colon s\ \varphi \ \equiv\ \varphi$
  - $\exists X\colon s\ \varphi \ \equiv\ \varphi$

Some Equivalences (6)

- Exchanging quantifiers: If $X \neq Y$:
  - $\forall X\colon s_1\,(\forall Y\colon s_2\,\varphi) \;\equiv\; \forall Y\colon s_2\,(\forall X\colon s_1\,\varphi)$
  - $\exists X\colon s_1\,(\exists Y\colon s_2\,\varphi) \;\equiv\; \exists Y\colon s_2\,(\exists X\colon s_1\,\varphi)$

    Note that quantifiers of different type ($\forall$ and $\exists$) cannot be exchanged.

- Renaming bound variables: If $Y \notin \mathit{free}(\varphi)$ and φ' results from φ by replacing every free occurrence of X in φ by Y:
  - $\forall X\colon s\,\varphi \;\equiv\; \forall Y\colon s\,\varphi'$
  - $\exists X\colon s\,\varphi \;\equiv\; \exists Y\colon s\,\varphi'$

Substitutions (1)

Definition:

- A (Σ, ν)-substitution θ is a mapping from variables to terms that repects the sorts, i.e. if $\nu(X) = s$, then $\theta(X) \in TE_{\Sigma,\nu}(s)$.

  If one uses a logic without sorts, θ is defined for the infinite set $VARS$, but one usually requires that $\theta(X) \neq X$ only for finitely many X.

- A substitution is usually written as set of variable-term-pairs, e.g. $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$.

  It is the identity mapping for all not explicitly mentioned variables.

Substitutions (2)

Definition:

- The result of applying a substitution θ to a term t, written $\theta(t)$ or $t\theta$, is defined as follows:
  - ◇ If t is a variable X, then
  $$\theta(t) := \theta(X).$$
  - ◇ If t has the form $f(t_1, \ldots, t_n)$, then
  $$\theta(t) := f(t'_1, \ldots, t'_n),$$
  where $t'_i := \theta(t_i)$.

Substitutions (3)

Definition:

- The result of applying a substitution θ to a formula φ, written $\theta(\varphi)$ or $\varphi\theta$, is defined as follows:
  - ◇ If φ is an atomic formula $p(t_1, \ldots, t_n)$, then $\theta(\varphi) := p(t'_1, \ldots, t'_n)$, where $t'_i := \theta(t_i)$.
  - ◇ If φ is $(\neg\psi)$, then $\theta(\varphi) := (\neg\psi')$ with $\psi' := \theta(\psi)$.
  - ◇ If φ has the form $(\psi_1 \wedge \psi_2)$, then $\theta(\varphi) := (\psi'_1 \wedge \psi'_2)$, where $\psi'_i := \theta(\psi_i)$. The same for $\vee$, $\leftarrow$, $\rightarrow$, $\leftrightarrow$.
  - ◇ If φ has the form $\forall X\colon s\ \psi$, then $\theta(\varphi) := \forall X\colon s\ \psi'$, where $\psi' := \theta'(\psi)$ and θ' agrees with θ except that $\theta'(X) = X$. The same for $\exists$.

Substitutions (4)

- I.e. when a substitution is applied to a formula, one replaces the variables as specified by the substitution and leaves the rest of the formula as it is.
- Only free variables are replaced by a substitution.
- A ground substitution for a quantifier-free formula φ is a substitution that replaces all variables in $free(\varphi)$ by a ground term.
- I.e. the result of applying a ground substitution to a formula is a ground formula.

Normal Forms (1)

Definition:

- A formula φ is in Prenex Normal Form iff it is closed and has the form

$$\Theta_1 X_1\!: s_1 \;\ldots\; \Theta_n X_n\!: s_n \;\; \psi$$

where $\{\Theta_1, \ldots, \Theta_n\} \subseteq \{\forall, \exists\}$ and ψ is quantifier-free.

- A formula φ is in Disjunctive Normal Form iff it is in Prenex Normal Form, and ψ has the form

$$(\psi_{1,1} \wedge \cdots \wedge \psi_{1,k_1}) \;\vee \cdots \vee\; (\psi_{n,1} \wedge \cdots \wedge \psi_{n,k_n}),$$

where each $\psi_{i,j}$ is an atomic formula or a negated atomic formula.

Normal Forms (2)

Remark:

- Conjunctive Normal Form is like disjunctive normal form, but ψ must have the form

$$(\psi_{1,1} \vee \cdots \vee \psi_{1,k_1}) \wedge \cdots \wedge (\psi_{n,1} \vee \cdots \vee \psi_{n,k_n}).$$

Theorem:

- Under the assumption of non-empty domains, every formula can be equivalently translated into prenex normal form, disjunctive normal form, and conjunctive normal form.

Overview

1. Signature, Interpretation

2. Formulas, Models

3. Implication, Equivalence

4. Clausal Form

5. Herbrand Interpretations

Literals, Clauses (1)

Definition:

- A literal is an atomic formula ("positive literal") or a negated atomic formula ("negative literal").
- A clause is a disjunction of zero or more literals. The empty clause is treated as false. All variables in a clause are treated as $\forall$-quantified.

  A clause is often seen as set of literals.
- A Horn clause is a clause with at most one positive literal. A definite Horn clause is a clause with exactly one positive literal.

Literals, Clauses (2)

- Clauses can be written as implications with the positive literals in the head, and negative literals (unnegated) in the body:

$$\underbrace{A_1 \vee \cdots \vee A_n}_{\text{Head}} \leftarrow \underbrace{B_1 \wedge \cdots \wedge B_m}_{\text{Body}}$$

- This corresponds to the disjunction

$$A_1 \vee \cdots \vee A_n \vee \neg B_1 \vee \cdots \vee \neg B_m.$$

- A definite Horn clause is a standard Pure Prolog rule with only one literal in the head:

$$A \leftarrow B_1 \wedge \cdots \wedge B_m.$$

Literals, Clauses (3)

- A clause with only negative literals $\neg B_1 \vee \cdots \vee \neg B_m$ is written as: $\leftarrow B_1 \wedge \cdots \wedge B_m$.

  Here the head is the empty disjunction, which is understood as false: A disjunction is satisfied if at least one of its elements is true. If there are no elements, it can never be true.

- $\Phi \vdash \psi$ iff $\Phi \cup \{\neg\forall(\psi)\}$ is inconsistent. (See above.)
- Refutation theorem provers, such as resolution, try to derive the empty clause "false" from $\Phi \cup \{\neg\forall(\psi)\}$.
- Therefore, a clause with only negative literals is often understood as proof goal $\exists(B_1 \wedge \cdots \wedge B_m)$.

Literals, Clauses (4)

- "Ground" always means "without variables":
  - A ground term consists only of constants and function symbols.
  - A ground literal is a predicate (or its negation) applied to ground terms.
  - Ground clause: disjunction of ground literals.
  - A ground substitution removes all variables from a quantifier-free formula (e.g. a clause).
  - φ is a ground instance of a clause ψ iff there is a ground substitution θ for ψ such that $\varphi = \theta(\psi)$.

Skolemnization (1)

- In clauses, all variables are $\forall$-quantified.
- Skolemnization is a technique for removing existential quantifiers.
- The idea of Skolemnization is to introduce names (constants or function symbols) for the values that are required to exist.
- E.g. $\exists X{:}\, s\; p(X, a)$ is replaced by $p(c, a)$ with a new constant c of sort s.

Skolemnization (2)

- The new formula is not equivalent (it is a formula over a different signature), but it is consistent whenever the old formula is consistent.

  A model of $\exists X{:}\, s\ p(X, a)$ can be extended to a model of $p(c, a)$ by interpreting c as the value for X that makes $p(X, a)$ true.

  Conversely, if one has a model of $p(c, a)$, one can simply forget the interpretation of c (but keep the value in the domain), to get a model of $\exists X{:}\, s\ p(X, a)$ for the original signature.

- For refutation theorem provers, only the consistency is important, thus this is no restriction.

Skolemnization (3)

- Suppose that the existential quantifier $\exists X : s$ is in the scope of universal quantifiers $\forall Y_1 : s_1 \ldots \forall Y_n : s_n$.
- Then the value for X may depend on the values for $Y_1, \ldots, Y_n$.
- Thus, Skolemnization replaces each occurrence of the variable X by $f(Y_1, \ldots, Y_n)$ with a new function symbol $f : s_1 \times \cdots \times s_n \rightarrow s$.
- With Skolemnization, any formula can be translated into a set of clauses.

  But one needs non-empty domains to get first prenex normal form.

Skolemnization (4)

Exercise:

- Consider the foreign key constraint:

$$\forall\, \mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{D}\ \Big(\, \mathtt{emp}(\mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{D}) \longrightarrow \exists\, \mathtt{N}, \mathtt{L}\ \mathtt{dept}(\mathtt{D}, \mathtt{N}, \mathtt{L})\,\Big)\,.$$

- Use the above transformations to show that it is equivalent to

$$\forall\, \mathtt{D}\ \exists\, \mathtt{N}, \mathtt{L}\ \forall\, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\ \Big(\, \mathtt{emp}(\mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{D}) \longrightarrow \mathtt{dept}(\mathtt{D}, \mathtt{N}, \mathtt{L})\,\Big)\,.$$

- What Skolem functions would be introduced for this formula?

Overview

1. Signature, Interpretation

2. Formulas, Models

3. Implication, Equivalence

4. Clausal Form

5. Herbrand Interpretations

Herbrand Interpretations (1)

Definition:

- A Σ-interpretation $\mathcal{I}$ is a Herbrand interpretation iff
  - ◇ for every sort s, the domain $\mathcal{I}[\![s]\!]$ is the set of all ground terms of sort s, i.e. $\mathcal{I}[\![s]\!] = TE_{s,\emptyset}(s)$.

    This assumes that for every sort there is at least one ground term. Otherwise one must extend the signature.
  - ◇ all function symbols are interpreted as the corresponding term constructors, i.e.

$$\mathcal{I}[\![f]\!](t_1, \ldots, t_n) = f(t_1, \ldots, t_n).$$

Herbrand Interpretations (2)

Remark:

- Thus, a Herbrand interpretation is given by the interpretation of the predicates.
- If the set of ground terms is finite, there are only finitely many Herbrand interpretations.
- Often a Herbrand interpretation $\mathcal{I}$ is specified by writing down all ground literals that are true in $\mathcal{I}$.

  All not explicitly mentioned ground literals are assumed to be false. Since the domains do not contain unnamed elements (only ground terms), this completely specifies the interpretation.

Herbrand Interpretations (3)

Theorem:

- A set Φ of universal formulas (formulas in prenex normal form with only universal quantifiers) without "$=$" is consistent iff it has a Herbrand model.

  Assuming that only interpretations are considered with non-empty domains.

Remark:

- The consistency of a set of formulas does not depend on the signature. Thus, it suffices to consider Herbrand interpretations with respect to the signature that contains only the symbols appearing in Φ.

Herbrand Interpretations (4)

- In general (non-Herbrand) interpretations, it is possible that
  - there are anonymous domain elements (objects not named by a constant or ground term).

    For universal formulas, such domain elements are not important. "For all" statements are even simpler to satisfy if the quantifiers range only over a subset.
  - different ground terms can denote the same object, e.g. $1 + 1$ and 2, or `murderer` and `butler`.

    As long as the logic contains no real equality, this is no problem: One simply defines all predicates such that they treat e.g. $1 + 1$ and 2 in the same way. One can have a user-defined "$=$".

Herbrand Interpretations (5)

Example/Exercise:

- Suppose that we want to prove that $\Phi := \{p(a)\}$ implies $\psi := \exists X\, p(X)$.
- $\Phi \vdash \psi$ iff $\Phi \cup \{\neg\forall(\psi)\}$ is inconsistent. Thus, we must check $\Phi' := \{p(a),\ \forall X(\neg p(X))\}$ for consistency.
- It is consistent iff it has a Herbrand model.
- The only ground term is a. Thus, there are only two Herbrand interpretations, $\mathcal{I}_1 := \{p(a)\}$ and $\mathcal{I}_2 := \emptyset$. None of the two satisfies both formulas.
- Exercise: What happens if $\psi := \forall X p(X)$?