4.0 Why?

• Today the lecturer looks different…
  – Silke Eckstein
    Lecturer of ‘Relational Databases 2’
  – By the way… very important and interesting lecture!

• Unfortunately Tilo Balke & Christoph Lofi are at a very important conference in Fès, Morocco…

4.0 Summary Last Lecture

• A well-formed term may consist of constant symbols, function symbols, and variables
  – E.g., \( f(a, f(a,b)) \) with \( \Gamma = \{a, b\}, \Omega = \{f\} \)
  – Terms can be used in other terms or atomic formulae

• A well-formed atomic formula includes a single predicate symbol
  – E.g., \( p(a, f(a,b)) \) with \( \Gamma = \{a, b\}, \Omega = \{f\}, \Pi = \{p\} \)
  – Atomic formulae cannot be used in other terms or atomic formulae
  – Logical junctors and quantifiers can be used to built non-atomic formulae

4.0 Summary Last Lecture

• Basic distinction between terms and formulae
  – A term represents some object on which propositions can be made
    • A term itself is neither true nor false
    • E.g., with interpretation \( a=1, b=2 \) and \( f='+' \) the term \( f(a, f(a,b)) \) represents the number ‘4’
  – A formula represents such a proposition
    • A formula can be either true or false
    • A predicate is a kind of ‘truth function’
    • E.g., with interpretation \( a=1, b=2 \) and \( p='<' \) the formula \( p(a, f(a,b)) \) represents a true proposition
Given is a set of formulae $\mathcal{W}$
- A model of $\mathcal{W}$ is an interpretation $I$ such that all formulas in $\mathcal{W}$ evaluate to true with respect to $I$.
- If $\mathcal{W}$ has a model, it is called satisfiable.
- If $\mathcal{W}$ has no model, it is called unsatisfiable or inconsistent.
- If two formulas always evaluate to the same truth value given any interpretation $I$, they are called semantically equivalent.

If every possible interpretation is a model of $\mathcal{W}$, the formulas in $\mathcal{W}$ are called tautologies.
- Sometimes also called valid.
- Denoted by $\models W$.
- Tautologies can be used to provide transformation rules for generating semantically equivalent formulas.

Clauses are special formulas containing only disjunctions of positive or negative literals.
- Horn clauses contain at most one positive literal.
- Lemma: Given a set of clauses $\mathcal{W}$
  - $\mathcal{W}$ has a model, if and only if $\mathcal{W}$ has a Herbrand model.
  - $\mathcal{W}$ is unsatisfiable, if and only if $\mathcal{W}$ has no Herbrand model.
- Open Question: How can Herbrand interpretations help evaluating queries in a deductive DB?

Using the Hilbert-style proof system show that:
- $\vdash A \rightarrow A$
  - Easy trick: use deduction theorem: $\{A\} \vdash A$
  - $W_1 \equiv A$
    (Hypothesis)
  - $W_2 \equiv A$
    (Assertion)
- $\vdash B \rightarrow ((B \rightarrow A) \rightarrow A)$
  - Deduction theorem: $\{B, B \rightarrow A\} \vdash A$
  - $W_1 \equiv B$
    (Hypothesis)
  - $W_2 \equiv B \rightarrow A$
    (Hypothesis)
  - $W_3 \equiv A$
    (MP $W_1 \& W_2$)
Exercise 2.1

- $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$
  - Deduction theorem: $\{A \rightarrow B, B \rightarrow C, A\} \vdash C$
  - $W_1 \equiv A \rightarrow B$ (Hypothesis)
  - $W_2 \equiv B \rightarrow C$ (Hypothesis)
  - $W_3 \equiv A$ (Hypothesis)
  - $W_4 \equiv B$ (MP $W_3 \& W_1$)
  - $W_5 \equiv C$ (MP $W_4 \& W_2$)

Exercise 2.2

- Transform the following statements to clauses
  - $A \rightarrow ((B \land C) \rightarrow D)$
  - $A \rightarrow (\neg(B \land C) \lor D)$
  - $A \rightarrow (\neg B \lor \neg C \lor D)$ (is also a Horn clause)
  - $(A \lor B \lor C) \rightarrow D$
  - $\neg(A \lor B \lor C) \lor D$
  - $(\neg A \land \neg B \land \neg C) \lor D$
  - $(\neg A \lor D) \land (\neg B \lor D) \land (\neg C \lor D)$ (cannot be a clause)

Exercise 2.3

- $\neg A \rightarrow \neg B$
  - $A \lor B$ (is also a Horn clause)
- $\neg A \rightarrow C$
  - $A \lor C$ (is not a Horn clause)
- $B \land (C \lor D)$
  - $(B \land C) \lor (A \land C)$ (cannot be a clause)

Exercise 3.1

- To check if a Herbrand Interpretation is a Herbrand model, check if all formulas in $\mathcal{W}$ are true if interpretation is applied
  a) Not a model as 2nd formula is not true
  b) Is a model
  c) Not a model as no formula is true

4.1 Relational Model

- With the logical tools a given above we can for example model a normal relational database
  - A relational database consists of
    - a relation schema describing the syntactical form of data together with the necessary integrity constraints
    - The actual data instance
- How can we model this with logic!!

4.1 Basic Model

- A relational database is a triple $\mathcal{DB} = (\mathcal{L}, \mathcal{C}, \mathcal{F})$
  - $\mathcal{L}$ is a language of first order predicate logic with an empty set of function symbols
  - $\mathcal{C}$ is a finite set of closed formulae over $\mathcal{L}$, called integrity constraints
  - $\mathcal{F}$ is a finite set of ground atoms of $\mathcal{L}$, called facts
- The relational schema $(\mathcal{L}, \mathcal{C})$ consists of a signature and integrity constraints
- $\mathcal{F}$ is the set of actual data
4.1 Basic Model

- Example database $DB_{uni} = (L, C, F)$
  - $L$ is given by $\Gamma = \{204, 207, 208, \text{Anne Huber}, \text{Peter Meier}, \text{Michael Schmidt}, \text{Braunschweig}, \text{Hannover}, \text{Computer Science}, \text{Math}\}$, $\Omega = \{\}$, $\Pi = \{\text{student, course}\}$, $X = \{x_1, x_2, x_3, x_4\}$
  - $C$ is given by
    \[
    \forall x_1 \forall x_2 \forall x_3 \in (\text{student}(x_1, x_2, x_3) \rightarrow \exists x_4 \in \text{course}(x_4, x_1))
    \]
  - $F$ is given by
    \[
    \text{student}(204, \text{Anne Huber}, \text{Braunschweig}).
    \text{student}(207, \text{Peter Meier}, \text{Hannover}).
    \text{course}(204, \text{Computer Science}).
    \text{course}(204, \text{Math}).
    \text{course}(207, \text{Math}).
    \text{course}(208, \text{Computer Science}).
    \]

4.1 Queries

- Of course the database can also be queried
  - For instance "Which students do not study math?"
  - Queries are translated into formulae that may contain free variables
    \[
    \exists x_1 \exists x_3 (\text{student}(x_1, x_2, x_3) \land \neg \text{course}(x_1, \text{Math}))
    \]
    - If there are no free variables the answer is generally true or false
    - If there are free variables the answer is given by all substitutions for these variables that make the statement true
    \[
    x_2 = \text{Michael Schmidt}
    \]

4.1 DB-Formulae

- For any relational database $DB = (L, C, F)$ we define a database formula as
  - Every atomic formula over $L$ is a database formula
  - If $G_1$ and $G_2$ are database formulae, so are $\neg G_1$, $(G_1 \land G_2)$ and $(G_1 \lor G_2)$
  - If $A$ is an atomic database formula with variables $\{x_1, ..., x_n\}$ and $G$ is a database formula, then also
    \[
    \forall x_1 \forall x_2 ... \forall x_n \in A \rightarrow G\text{ and } \exists x_1 \exists x_2 ... \exists x_n \in A \rightarrow G\text{ and } \exists x_1 \exists x_2 ... \exists x_n \in A \land G\text{ are database formulae}
    \]

4.1 DB-Formulae

- Every integrity constraint is simply a closed database formula
- Every query $Q$ either...
  - Is also a closed database formula (answered with true/false)
  - Or has free variables $\{x_1, ..., x_n\}$ such that the formula $\exists x_1 \exists x_2 ... \exists x_n (Q)$ is a closed database formula
    - If $Q$ deals with some predicate $p$ this compares to the SQL statement $\text{SELECT} \ x_1, ..., x_n \ FROM \ p$
    - With a closed formula $G$ the query $(Q \land G)$ compares to the SQL statement $\text{SELECT} \ x_1, ..., x_n \ FROM \ p \ WHERE \ G$

4.1 Example

- Example database $DB_{uni} = (L, C, F)$
  - The database schema features
    - A predicate student giving the matricel-number, name and address of each student
    - A predicate course giving a matricel-number and the respective course of studies
    - An integrity constraint stating that every student has to be assigned to some course of studies
  - The current set of facts does not violate the integrity constraint
  - Actually, the a-priori definition of all possible constants (e.g., names) is not practical for realistic relational databases, but only data types are defined
4.1 Closed World

• With our definition of database formulae we can respect the closed world assumption
  – Consider the query \( Q := \text{course}(208, \text{Math}) \)
  – We can deduce neither \( \mathcal{F} \models Q \) nor \( \mathcal{F} \models \neg Q \)
  – There exist models for \( \mathcal{F} \) where Michael Schmidt studies only computer science and other models where he studies both math and computer science
  – Deduction cannot make statements about what is not in the database

4.1 Integrity Constraints

• Following our definition of a database formula also integrity constraints are special cases of queries
  – Closed database formulae
  – A relational database is called consistent, if \( C \) can be derived from \( \mathcal{F} \) for all \( C \in \mathcal{C} \)

4.1 Integrity Constraints

• So let’s substitute the ground terms...

\[
\begin{align*}
\mathcal{F} \not\models (\text{student}(204, \text{Anne Huber}, \text{Braunschweig}) \\
\quad \land \neg \exists x_1 \text{ course}(204, x_1))
\end{align*}
\]

and

\[
\begin{align*}
\mathcal{F} \not\models (\text{student}(207, \text{Peter Meier}, \text{Hannover}) \\
\quad \land \neg \exists x_2 \text{ course}(207, x_1))
\end{align*}
\]

and

\[
\begin{align*}
\mathcal{F} \not\models (\text{student}(208, \text{Michael Schmidt}, \text{Braunschweig}) \\
\quad \land \neg \exists x_3 \text{ course}(208, x_1))
\end{align*}
\]

\[
\begin{align*}
\mathcal{F} \not\models \neg \exists x_4 \text{ course}(204, x_4)
\end{align*}
\]

and

\[
\begin{align*}
\mathcal{F} \not\models \neg \exists x_5 \text{ course}(207, x_4)
\end{align*}
\]

and

\[
\begin{align*}
\mathcal{F} \not\models \neg \exists x_6 \text{ course}(208, x_4)
\end{align*}
\]

4.1 Integrity Constraints

• But if we identify every query \( Q \) with a closed formula, where all free variables are existentially quantified and bound to database facts (IF=)...

- With the set of free variables \( \{x_1, \ldots, x_n\} \) in query \( Q \):
  \( \mathcal{F} \models Q \iff \mathcal{F} \models \exists x_1 \exists x_2 \ldots \exists x_n (Q) \) with suitable substitutions

- Since \( Q := \text{course}(208, \text{Math}) \) cannot be derived from \( \mathcal{F} \) with any substitution, the opposite has to be true (\( \neg Q \))
  - For everything that is not in the database, and cannot be deduced from the database, now the negation is true
  - That is usually intuitive, a student that is not in the database will very probably not exist as a student...

4.1 Integrity Constraints

• Let’s have a look on our example database \( DB_{\text{ini}} \)

\[
\begin{align*}
\mathcal{F} \models \forall x_1 \forall x_2 \forall x_3 (\text{student}(x_1, x_2, x_3) & \Rightarrow \exists x_4 \text{ course}(x_1, x_4)) \\
\mathcal{F} \models \forall x_1 \exists x_2 \exists x_3 (\text{student}(x_1, x_2, x_3) & \land \neg \exists x_4 \text{ course}(x_1, x_4)) \\
\mathcal{F} \models \exists x_1 \exists x_2 \exists x_3 (\text{student}(x_1, x_2, x_3) & \land \neg \exists x_4 \text{ course}(x_1, x_4)) \\
\mathcal{F} \models \exists x_1 \exists x_2 \exists x_3 (\text{student}(x_1, x_2, x_3) & \land \exists x_4 \text{ course}(x_1, x_4))
\end{align*}
\]

with ground terms \( c_1, c_2, c_3, c_4 \) from the database

- Note: the last statement can only be true, if student\( (c_1, c_2, c_3) \) is true
- And all such ground terms are explicitly given by \( \mathcal{F} \)
- Our definition of database formulas implies that ground terms for quantified variables can always be taken directly from some facts

4.1 Integrity Constraints

• And finally...

\[
\begin{align*}
\mathcal{F} \models \exists x_1 \text{ course}(204, x_1) \\
\mathcal{F} \models \exists x_2 \text{ course}(207, x_1) \\
\mathcal{F} \models \exists x_3 \text{ course}(208, x_1)
\end{align*}
\]

- The last set of statements again can directly be verified from \( \mathcal{F} \) and thus our database is consistent
4.1 Model

• By binding our ground terms to the database facts we have in fact given a (finite) **Herbrand base**
  – The **intended model** of any relational database $\mathcal{DB} = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ is a Herbrand interpretation $\mathcal{H}(\mathcal{F})$ represented by the ground atoms in $\mathcal{F}$
  – If $\mathcal{DB} = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ and $\mathcal{F}$ a closed database formula then $\mathcal{F} \models \mathcal{F}$, if $\mathcal{H}(\mathcal{F}) \models \mathcal{F}$
  – Hence instead of modeling facts as **ground atoms** $\mathcal{F}$, an alternative is modeling facts as $\mathcal{L}$-interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{C}$

4.1 Views

• The model of the database can even be specified by **other formulae** (together with the ground atoms)
  – This reflects the idea of **views** in relational databases
  – Example: for our $\mathcal{DB}_{uni}$ we could add another predicate math-student by adding the formula
    $\forall x_1 \forall x_3 (\exists x_2 (\text{student}(x_1, x_2, x_3) \land \text{course}(x_1, \text{Math}) \rightarrow \text{math-student}(x_2, x_3))$
    • This derives name and address of all students studying math
  – The new formula can be either **derived at query time**, or can be **calculated once** and stored as additional ground atoms (‘materialized’ view)

Outlook

• Finally: Herbrand’s theorem
• Evaluation of deductive database queries
• Datalog