Institut für Informationssysteme
Technische Universität Braunschweig

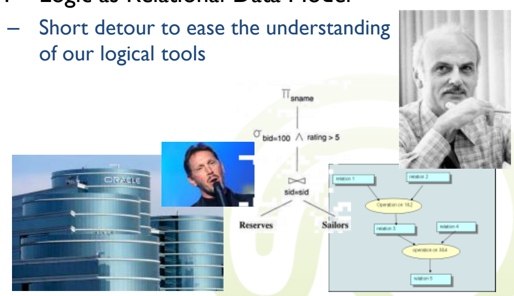
Knowledge-Based Systems and Deductive Databases

Wolf-Tilo Balke
Christoph Lofi
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de

4. Relational Database Model

4.1 Logic as Relational Data Model

– Short detour to ease the understanding of our logical tools

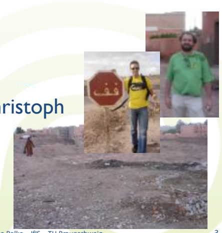


4.0 Why?

- 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 **first order logic language** can be defined as a quadruple $\mathcal{L} = (\Gamma, \Omega, \Pi, X)$
  - Γ is the non-empty and decidable set of **constant symbols**
  - Ω is the disjunctive union of the finite sets of n-ary **functional symbols**
  - Π is the disjunctive union of the finite sets of n-ary **predicate symbols**
  - X is the enumerable set of **variables**

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 $a=1$, $b=2$, f='+' and p='<' the formula $p(a, f(a,b))$ represents a **true** proposition

4.0 Summary Last Lecture

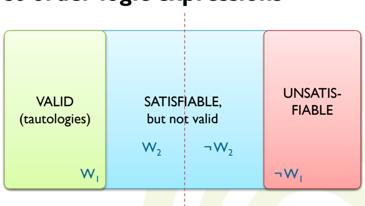
- Given is a set of formulae $\mathcal{W}$
  - A **model** of $\mathcal{W}$ is an **interpretation** I such that all formulas in $\mathcal{W}$ evaluate to true with respect to I
- If $\mathcal{W}$ has a model, it is called **satisfiable**
  - If $\mathcal{W}$ has no model, it is called **unsatisfiable** or **inconsistent**
  - If two formulas **always** evaluate to the same truth value given any interpretation I, they are called **semantically equivalent**

4.0 Summary Last Lecture

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

4.0 Summary Last Lecture

- **All first-order logic expressions**



  - You might think of the **negation** as mirror operation along the red-dotted line

4.0 Summary Last Lecture

- A formula W is a **semantic conclusion** of $\mathcal{W}$, iff every model of $\mathcal{W}$ is also a model of W
  - $\mathcal{W}\vDash$W (W semantically follows from $\mathcal{W}$)
  - Test for $\mathcal{W}\vDash$W: show that $\mathcal{W} \cup \{\neg W\}$ is **unsatisfiable**
  - Testing unsatisfiability is generally quite difficult due to the **unlimited number of possible interpretations**
- Idea: **Herbrand Interpretations**
  - Herbrand interpretations interpret each constant and each closed formula on mirror of itself
  - Purely **symbolic interpretations**, as such they represent some kind of a worst case scenario

4.0 Summary Last Lecture

- **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?

Exercise 2.1

Solutions

- 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 Solutions

- $\models (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$
  - Deduction theorem: $\{A \rightarrow B, B \rightarrow C, A\} \models C$
  - $W_1 \equiv A \rightarrow B$ (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 Solutions

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

Exercise 2.3 Solutions

- $\neg A \rightarrow \neg B$
  - $A \vee \neg B$ (is also a **Horn clause**)
- $\neg A \rightarrow C$
  - $A \vee C$ (is not a Horn clause)
- $B \wedge (C \vee D)$
  - $(B \wedge C) \vee (A \wedge C)$ (cannot be a clause)

Exercise 3.1 Solutions

- 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 Detour

- 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 Detour

- 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
Detour

- **Example database $\mathcal{DB}_{uni} = (\mathcal{L}, \mathcal{C}, \mathcal{F})$**
  - $\mathcal{L}$ is given by Γ={204, 207, 208, Anne Huber, Peter Meier, Michael Schmidt, Braunschweig, Hannover, Computer Science, Math}, Ω={}, Π ={student, course}, **X**={x_1, x_2, x_3, x_4}
  - $\mathcal{C}$ is given by
    $\forall x_1 \forall x_2 \forall x_3$ (student(x_1, x_2, x_3) $\rightarrow \exists x_4$ course(x_1,x_4))
  - $\mathcal{F}$ is given by
    student(204, Anne Huber, Braunschweig).
    student(207, Peter Meier, Hannover).
    student(208, Michael Schmidt, Braunschweig).
    course(204, Computer Science).
    course(204, Math).
    course(207, Math).
    course(208, Computer Science).

4.1 Example
Detour

- **Example database $\mathcal{DB}_{uni} = (\mathcal{L}, \mathcal{C}, \mathcal{F})$**
  - The database schema features
    - A **predicate student** giving the matrikel-number, name and address of each student
    - A **predicate course** giving a matrikel-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 Queries
Detour

- 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
    $\exists x_1 \exists x_3$ (student(x_1, x_2, x_3) $\wedge$ ¬course(x_1, Math))
    - If there are **no free variables** the answer is generally either **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 = Michael Schmidt

4.1 Queries
Detour

- But such queries can be **difficult** to answer
  - For instance 'Who is not a student?'
  - $\neg(\exists x_1 \exists x_3$ student(x_1, x_2, x_3))
  - Answer is the (possibly infinite) **complement** of our three students???
  - Remember: databases follow the **closed world assumption**

4.1 DB-Formulae
Detour

- For any relational database $\mathcal{DB}$=($\mathcal{L}, \mathcal{C}, \mathcal{F}$) we define a **database formula** as
  - Every atomic formula over $\mathcal{L}$ is a database formula
  - If G, G_1 and G_2 are database formulae, so are ¬G, ($G_1 \wedge G_2$) and ($G_1 \vee G_2$)
  - If A is an atomic database formula with variables {x_1, ..., x_n} and G is a database formula, then also $\forall x_1 \forall x_2 \ldots \forall x_n$ (A → G) and $\exists x_1 \exists x_2 \ldots \exists x_n$ (A → G) and $\exists x_1 \exists x_2 \ldots \exists x_n$ (A $\wedge$ G) are database formulae

4.1 DB-Formulae
Detour

- 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 $\exists x_1 \exists x_2 \ldots \exists x_n$ (Q) is a closed database formula
    - If Q deals with some predicate p this compares to the **SQL** statement SELECT x_1,...,x_n FROM p
    - With a closed formula G the query (Q $\wedge$ G) compares to the **SQL** statement SELECT x_1,...,x_n FROM p WHERE G

4.1 Closed World Detour

- 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 $\mathcal{F} \vDash Q$, nor $\mathcal{F} \vDash \neg Q$
  - 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 Closed World Detour

- But if we **identify** every query Q with a closed formula, where all free variables are existentially quantified and **bound to** database facts ($\Vdash$)…
  - With the set of free variables $\{x_1, \ldots, x_n\}$ in query Q: $\mathcal{F} \Vdash Q \;\Leftrightarrow\; \mathcal{F} \Vdash \exists x_1\, \exists x_2 \ldots \exists x_n\, (Q)$ 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 Detour

- 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 Detour

- Let's have a look on our example database $\mathcal{DB}_{uni}$

  $\mathcal{F} \Vdash \forall x_1\, \forall x_2\, \forall x_3\, (\text{student}(x_1, x_2, x_3) \rightarrow \exists x_4\, \text{course}(x_1, x_4))$
  $\Leftrightarrow \mathcal{F} \Vdash \neg\exists x_1\, \exists x_2\, \exists x_3\, (\text{student}(x_1, x_2, x_3) \wedge \neg\exists x_4\, \text{course}(x_1, x_4))$
  $\Leftrightarrow \mathcal{F} \nVdash \exists x_1\, \exists x_2\, \exists x_3\, (\text{student}(x_1, x_2, x_3) \wedge \neg\exists x_4\, \text{course}(x_1, x_4))$
  $\Leftrightarrow \mathcal{F} \nVdash \exists c_1\, \exists c_2\, \exists c_3\, (\text{student}(c_1, c_2, c_3) \wedge \neg\exists x_4\, \text{course}(c_1, x_4))$
  with ground terms c_1, c_2, c_3 from the database
  - Note: the **last statement** can only be true, if $\text{student}(c_1, c_2, c_3)$ 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 Detour

- So let's substitute the ground terms…

$\Leftrightarrow \;\mathcal{F} \nVdash (\text{student}(204, \text{Anne Huber}, \text{Braunschweig})$
$\quad \wedge \neg\exists x_4\, \text{course}(204, x_4))$
and $\mathcal{F} \nVdash (\text{student}(207, \text{Peter Meier}, \text{Hannover})$
$\quad \wedge \neg\exists x_4\, \text{course}(207, x_4))$
and $\mathcal{F} \nVdash (\text{student}(208, \text{Michael Schmidt}, \text{Braunschweig})$
$\quad \wedge \neg\exists x_4\, \text{course}(208, x_4))$
$\Leftrightarrow \;\mathcal{F} \nVdash \neg\exists x_4\, \text{course}(204, x_4))$
and $\mathcal{F} \nVdash \neg\exists x_4\, \text{course}(207, x_4))$
and $\mathcal{F} \nVdash \neg\exists x_4\, \text{course}(208, x_4))$

4.1 Integrity Constraints Detour

- And finally…

$\Leftrightarrow \;\mathcal{F} \Vdash \exists x_4\, \text{course}(204, x_4))$
and $\mathcal{F} \Vdash \exists x_4\, \text{course}(207, x_4))$
and $\mathcal{F} \Vdash \exists x_4\, \text{course}(208, x_4))$

  - The last set of statements again can directly be verified from $\mathcal{F}$ and thus our database is **consistent**

4.1 Model

Detour

- 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 $\mathcal{H}_{\mathcal{L}}(\mathcal{F})$ represented by the ground atoms in $\mathcal{F}$
  - If $\mathcal{DB} = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ and F a closed database formula then $\mathcal{F} \Vdash \mathrm{F}$, iff $\mathcal{H}_{\mathcal{L}}(\mathcal{F}) \vDash \mathrm{F}$
  - Hence instead of modeling facts as **ground atoms** $\mathcal{F}$, an alternative is modeling facts as **$\mathcal{L}$-interpretation** I with $\mathrm{I} \vDash \mathcal{C}$

4.1 Views

Detour

- 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 $\mathcal{DB}_{\mathrm{uni}}$ we could add another predicate math-student by adding the formula
    $\forall x_2\ \forall x_3\ (\exists x_1\ (\mathrm{student}(x_1, x_2, x_3) \wedge \mathrm{course}(x_1, \mathrm{Math}))$
    $\rightarrow \mathrm{math\text{-}student}(x_2, x_3))$
    - 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 queires
- Datalog