Plan of the course

1. Datalog
2. Negation
3. Schema mapping
4. Data integration and exchange
5. Schematic discrepancies
6. Query evaluation
7. Inconsistency and incompleteness
8. XML databases
9. XML query languages
10. XML data integration
11. Semantic Web
Datalog [AHV95]

A logical language:
- Datalog programs consist of logical facts and rules
- Datalog is a subset of Prolog (no data structures).

Basic concepts:
- term: constant, variable
- predicate (relation)
- atom
- clause, rule, fact
- substitution
- unification

Logic programs

Atom
- syntax: $P(T_1, \ldots, T_n)$
- semantics: predicate $P$ is true of terms $T_1$ and ... and $T_n$

Implication (clause)
- syntax: $A_0 : - A_1, \ldots , A_k$
- semantics: atom $A_0$ is true if atoms $A_1$ and ... and $A_k$ are true
- $k = 0$: fact (ground if no variables)
- $k > 0$: rule consisting of head $A_0$ and body $A_1, \ldots , A_k$
- all the variables universally quantified
- all the variables in the head occur also in the body

Logic program $P$
- $EDB(P)$: a set of ground facts encoding a database instance
- $IDB(P)$: a set of rules encoding a query, with a special predicate query to return the result
### Ancestry

#### Facts
- parent(witold, tom).
- parent(tom, jan).
- parent(tom, tony).
- parent(jan, dave).

#### Rules
- anc(X, Y) :- parent(X, Y).
- anc(X, Z) :- parent(X, Y), anc(Y, Z).

#### Query 1
- query(X) :- anc(X, dave).

#### Query 2
- query(X) :- anc(X, dave), anc(X, tony).

### Logical semantics

#### The least Herbrand model $M_P$ of a program $P$
- **Herbrand**: constants interpreted as themselves
- **Least**: contained in all the Herbrand models of $P$

#### Properties of $M_P$
- $M_P$: a subset of the Herbrand base (the set of all possible ground atoms) of $P$
- $M_P$ = set of ground facts implied by $P$

#### Query answer
A tuple $t$ is an answer to the query $Q$ encoded by $IDB(P)$ in the database encoded by $EDB(P)$ if $\text{query}(t) \in M_P$. 

#### Ancestry
$M_P$ consists of all parent facts and:

- **First-level ancestors**
  - anc(witold, tom).
  - anc(tom, jan).
  - anc(tom, tony).
  - anc(jan, dave).

- **Second-level ancestors**
  - anc(witold, jan).
  - anc(witold, tony).
  - anc(tom, dave).

- **Third-level ancestors**
  - anc(witold, dave).
Substitutions

Substitution
- mapping from variables to terms
- ground: all the terms in the range are constants

Substitutions can be applied to atoms and clauses:
- instance \( h(A) \) is obtained by replacing all the occurrences of variable \( x \) in \( A \) by the term \( h(x) \) (for every variable)

Substitutions can be composed:
\[
(h \circ g)(x) = g(h(x)).
\]

Ground program ground\((P)\)
All the ground instances of the clauses in \( P \), constructed using the constants in \( P \) (active domain).

Fixpoint semantics

Given:
- a logic program \( P \)
- a set of ground facts \( I \).

\( T_P \) operator
\[
T_P(I) = \{ A \mid \exists r \in \text{ground}(P).\ r = A : -A_1, \ldots, A_n \land A_1 \in I \land \cdots A_n \in I.\}
\]

Fixpoint
\( I \) is a fixpoint of \( T_P \) if \( T_P(I) = I \).

Properties of \( T_P \)
- \( T_P \) is monotonic, i.e., \( I \subseteq J \Rightarrow T_P(I) \subseteq T_P(J) \)
- \( T_P \) has a least fixpoint \( \text{lfp}(T_P) \) contained in all the fixpoints
- \( \text{lfp}(T_P) = M_P \)
- \( \text{lfp}(T_P) \) is obtained by iterating \( T_P \) finitely many times
Naive evaluation

- multiple iterations: compute $T_P(\emptyset), T_P(T_P(\emptyset)), \ldots$ until result does not change
- evaluation terminates: result is $M_P$
- retrieve query predicate facts from $M_P$

!!! facts rederived multiple times

Semi-naive evaluation

- the result is accumulated
- at least one new fact is used in each rule application
- rule $r$ has with only EDB atoms in the body: $r$ is applied only in the first iteration
- rule $r$ has IDB atoms $A_1, \ldots, A_n$ in the body: in every iteration $r$ is applied $n$ times, each time considering for $A_i$ only the facts added in the last iteration
- further optimizations are possible

Top-down evaluation

Ground case

Evaluation of a ground atom $A$:

- $A$ succeeds immediately if there is a fact $A$ in the program $P$
- $A$ succeeds if there is a clause $A : -A_1, \ldots, A_n \in \text{ground}(P)$ such that $A_1, \ldots, A_n$ succeed

General case

Evaluation of a non-ground atom $A$:

- unify $A$ with a fact or a rule head (after renaming apart) in $P$
- propagate the substitutions to the body of the rule
- evaluate the body
**Unification**

**Most general unifier**

A substitution $h$:
- **unifies** two atoms $A$ and $B$ if $h(A) = h(B)$.
- is a **most general unifier (mgu)** of $A$ and $B$ if for all other substitutions $g$ unifying $A$ and $B$, there is a substitution $g'$ such that $g = h \circ g'$.

**Unification algorithm**

Returns **failure** or an **mgu** $h$ of $A$ and $B$, given two atoms $A = P(x_1, \ldots, x_n)$ and $B = Q(y_1, \ldots, y_k)$.

If $P \neq Q$ or $n \neq k$: return **failure**. Otherwise $Pairs = (x_i, y_i) \mid i = 1, \ldots, n$ and $Pairs^*$ is the reflexive symmetric transitive closure of $Pairs$:

1. If any equivalence class of $Pairs^*$ contains two different constants: return **failure**.
2. Otherwise for every equivalence class of $Pairs^*$:
   - if the class contains a constant $c$, then for every variable $x$ in the class, $h(x) = c$,
   - otherwise, pick a variable $x_0$, which is the smallest in some fixed ordering of variables, in the class, and for every other variable $x$ in the class, $h(x) = x_0$.

**General logic programs**

**Syntax**

- terms can be built using **function symbols**: $f(0), f(f(X)), \ldots$
- terms encode data structures

**Semantics**

- active domain contains ground terms
- least Herbrand model $M_P = \text{lfp}(T_P)$
- $M_P$ can be **infinite**

**Query evaluation**

- bottom-up or top-down: may fail to terminate
- an **undecidable** problem
- unification: requires **occur check**