Logic and Databases

Wolfgang Faber

University of Calabria, Italy

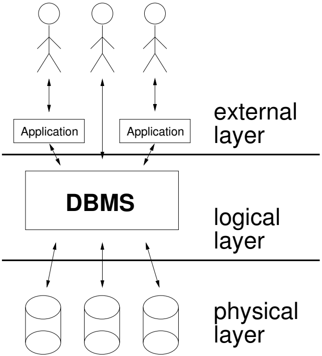
2007

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory
     - Proof Theory

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

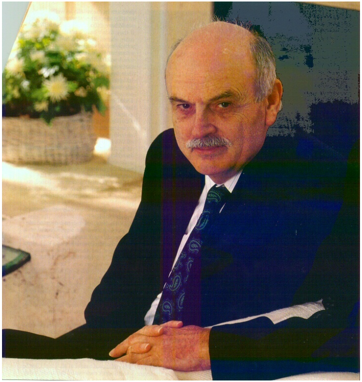
Three Layer Model



Three Layer Model

- External Layer
  How external users view the database.
- Logical/Conceptual Layer
  Logical, holistic view of the database.
- Physical/Internal Layer
  Organisation on the physical media.

Relational Model – Codd 1970



Edgar Frank Codd (1923–2003)

Relations

- Schema:
  - Domain (denumerable set)
  - Attributes (denumerable set)
  - Relations (subset of attributes)
- Instances:
  - Relation instances: Sets of tuples.
  - Each tuple is a function from the relation's attributes to domain elements.
  - Database instance: Collection of relation instances.

Relations: Example

$$A = \{X, Y\}, D = \{a, b, c, d\}$$
$$R = \{X, Y\}, S = \{Y\}$$

$$I(R) = \{t_1, t_2\}$$
$$t_1(X) = a, t_1(Y) = b, t_2(X) = c, t_2(Y) = d$$
$$I(S) = \{t_3\}, t_3(Y) = b$$

$$I(R) = \{\langle a, b\rangle, \langle c, d\rangle\}, I(S) = \{\langle b\rangle\}$$

Relations: Example

R	X	Y
	a	b
	c	d

S	Y
	d

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

Relational Algebra

Basic Operators:

- σ Selection
- π Projection
- $\times$ Cartesian Product
- $\cup$ Union
- $-$ Difference

Definable using Basic Operators:

- $\bowtie$ Join [$R \bowtie S = \sigma_F(R \times S)$]
- $\ltimes$ Semijoin [$R \ltimes S = \pi_{Schema(R)}(R \bowtie S)$]
- $\cap$ Intersection

Relational Algebra Example

$R \times S$	X_R	Y_R	Y_S
	a	b	d
	c	d	d

$\sigma_{2=3}(R \times S)$	X_R	Y_R	Y_S
	c	d	d

$\pi_{1,2}(\sigma_{2=3}(R \times S))$	X	Y
	c	d

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

Relations – Logical View

- Schema:
  - Domain – Constant symbols (denumerable set)
  - Relations – Predicate symbols (attributes are not explicitly named)
  - Attributes – implicit by predicate arity
- Instances:
  - Relation instances: Subset of ground instances for relation predicate.
  - Database instance: Subset of Herbrand Base.

Relations: Example

$D = \{a, b, c, d\}$
$R/2, S/1$

$I(R) = \{R(a, b), R(c, d)\}, I(S) = \{S(d)\}$

$I = \{R(a, b), R(c, d), S(d)\}$

Relational Calculus

- Based on First-Order Logic
- Atomic formulas $r(X_1, \ldots, X_n)$
- Comparison formulas $X = 2$ or $X = Y$ (pre-interpreted predicate)
- Composed formulas using $\neg$, $\wedge$, $\exists$
- $\rightarrow$, $\leftrightarrow$, $\vee$, $\forall$ added as "syntactic sugar"

Relational Calculus

- Relational Algebra expressions represent relation instances
- In Relational Calculus: $\{e_1, \ldots, e_n \mid \phi\}$
  - ϕ is a Relational Calculus formula
  - $e_1, \ldots, e_n$: terms containing exactly the free variables of ϕ
- Collect all substitutions for free variables such that ϕ is true in the interpretation formed by the database.
- The defined relation is obtained by applying all of these substitutions to $e_1, \ldots, e_n$.

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \land S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \land S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \land S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Algebra as Calculus

- σ_S r $\{X_1, \ldots, X_n \mid r(X_1, \ldots, X_n) \wedge S\}$
- π_i r $\{X_i \mid \exists X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n : r(X_1, \ldots, X_n)\}$
- r $\times$ s
  $\{X_1, \ldots, X_n, Y_1, \ldots, Y_m \mid r(X_1, \ldots, X_n) \wedge s(Y_1, \ldots, Y_m)\}$
- r $\cup$ s $\{X_1, \ldots, X_n \mid r(X_1, \ldots, X_n) \vee s(X_1, \ldots, X_n)\}$
- r $-$ s $\{X_1, \ldots, X_n \mid r(X_1, \ldots, X_n) \wedge \neg s(X_1, \ldots, X_n)\}$

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

Calculus: More then Algebra

Problematic expressions:

$$\{X \mid \neg R(a, X)\}$$
$$\{X, Y \mid R(a, X) \vee R(Y, b)\}$$
$$\{X \mid \forall Y : R(X, Y)\}$$

Calculus: More then Algebra

Using the domain of the database:

- $\{X \mid \neg R(a, X)\}$
  - all constants c of the domain such that (a, c) is no tuple in R
  - will be infinite if the domain is infinite
- $\{X, Y \mid R(a, X) \vee R(Y, b)\}$
  - if R contains some tuple (a, b), (b, c) for all constants c in the domain
  - will be infinite if the domain is infinite
- $\{X \mid \forall Y : R(X, Y)\}$
  - this will be always empty if the domain is infinite, because relations are finite

Calculus: More then Algebra

Using the active domain of the database (only constants appearing in the database and the query):

- $\{X \mid \neg R(a, X)\}$
  - all constants c in the database such that (a, c) is no tuple in R
  - will change if some unrelated constant is added
- $\{X, Y \mid R(a, X) \vee R(Y, b)\}$
  - if R contains some tuple (a, b), (b, c) for all constants c in the database
  - will change if some unrelated constant is added
- $\{X \mid \forall Y : R(X, Y)\}$
  - will unintuitively become empty if an unrelated constant is added

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

Domain Independent Queries

Idea: Consider only those queries for which Natural and Active Domain Semantics coincide.

Definition

A query in the relational calculus is domain independent, if it yields the same answer using the natural (full) domain and the active domain.

Domain Independent Queries

Idea: Consider only those queries for which Natural and Active Domain Semantics coincide.

Definition

A query in the relational calculus is domain independent, if it yields the same answer using the natural (full) domain and the active domain.

Domain Independent Queries

Theorem

Any query of the Relational Algebra can be written as a domain independent query of Relational Calculus, and vice versa.

Great, let's use only domain independent queries of Relational Calculus.

Domain Independent Queries

Theorem

Any query of the Relational Algebra can be written as a domain independent query of Relational Calculus, and vice versa.

Great, let's use only domain independent queries of Relational Calculus.

Domain Independent Queries

Theorem

Deciding whether a query of Relational Calculus is domain independent, is undecidable.

Outline

1 Relational Databases
- Relational Model
- Relational Algebra
- Relational Model – Logical View

2 Relational Calculus

3 Domain Independence
- Domain Dependent Queries
- Domain Independent Queries
- Safe Range Queries
- SQL

4 Datalog
- Motivation
- Syntax
- Semantics
  - Model Theory
  - Fixpoint Theory

Safe Range Queries

Define a syntactically restricted fragment of Relational Calculus queries, which is guaranteed to be domain independent.

1. Transform formula into a normal form (SRNF).
2. Determine range restricted variables of the SRNF formula.
3. Check whether the range restricted variables are exactly the free variables.

SRNF

- Normalize variables: Rename variables, so that each quantifier binds a distinct variable and free and bound variables are different.
- Remove $\forall$: $\forall X : \phi \Rightarrow \neg\exists X : \neg\phi$
- Remove $\rightarrow$: $\phi \rightarrow \psi \Rightarrow \neg\phi \vee \psi$
- Remove $\neg\neg$: $\neg\neg\phi \Rightarrow \phi$
- Push $\neg$: $\neg(\phi \wedge \psi) \Rightarrow (\neg\phi \vee \neg\psi)$
- Push $\neg$: $\neg(\phi \vee \psi) \Rightarrow (\neg\phi \wedge \neg\psi)$

Apply these rules as until none is applicable.

Range Restricted Variables

Intuition: Variables, for which the value is determined by the database.

- Variables in relational atoms are range restricted.
- Variables in equality comparisons with a constant are range restricted.
- Variables in conjunctions are range restricted if they are range restricted in the subformulas.
- Variables in disjunctions are only range restricted if they occur range restricted in both subformulas.
- Variables in negated formulas are never range restricted.
- Variables in existentially quantified subformulas (without the quantified variable) are range restricted if the quantified variable is range restricted in the subformula.

Range Restriction Algorithm

Funtion *rr*
Input: Formula ϕ in SRNF
Output: Subset of free variables of ϕ or $\perp$
case ϕ of

- $R(t_1, \ldots, t_n)$: $rr(\phi) =$ all variables in $t_1, \ldots, t_n$;
- $X = a$ or $a = X$: $rr(\phi) = \{X\}$;
- $\phi_1 \wedge \phi_2$: $rr(\phi) = rr(\phi_1) \cup rr(\phi_2)$;
- $\phi_1 \wedge X = Y : rr(\phi) =$
$$\begin{cases} rr(\phi_1) & \text{if } \{X, Y\} \cap rr(\phi_1) = \emptyset; \\ rr(\phi_1) \cup \{X, Y\} & \text{otherwise}; \end{cases}$$
- $\phi_1 \vee \phi_2$: $rr(\phi) = rr(\phi_1) \cap rr(\phi_2)$;
- $\neg\phi_1$: $rr(\phi) = \emptyset$;
- $\exists X : \psi$: if $X \in rr(\psi)$ then $rr(\phi) = rr(\psi) \setminus \{X\}$ else return $\perp$;

Assumption: Set operations with $\perp$ always result in $\perp$.

Safe Range Queries

Definition

A Relational Calculus query $\{e_1, \ldots, e_n \mid \phi\}$ is safe range, if $rr(SRNF(\phi))$ is equal to the free variables in ϕ.

Theorem

Each safe range query is domain independent.

Theorem

Any safe range query can be written as query of Relational Algebra, and vice versa.

Safe Range Queries

Definition

A Relational Calculus query $\{e_1, \ldots, e_n \mid \phi\}$ is safe range, if $rr(SRNF(\phi))$ is equal to the free variables in ϕ.

Theorem

Each safe range query is domain independent.

Theorem

Any safe range query can be written as query of Relational Algebra, and vice versa.

Safe Range Queries

Definition

A Relational Calculus query $\{e_1, \ldots, e_n \mid \phi\}$ is safe range, if $rr(SRNF(\phi))$ is equal to the free variables in ϕ.

Theorem

Each safe range query is domain independent.

Theorem

Any safe range query can be written as query of Relational Algebra, and vice versa.

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

SQL

- Exists since 1974 (developed by IBM).
- ISO/ANSI standardization 1986/87.
- First extension 1989.
- Second extension 1992 (SQL-92/SQL-2).
- Last (up to now) extension 1999/2000 (SQL-99/SQL-3)
- SQL combines query and manipulation languages.

SQL

SELECT P
FROM C
WHERE S

- P — Projections
- C — Cartesian Product
- S — Selections

SQL

Union:

SELECT . . .
UNION
SELECT . . .

Nesting:

SELECT *P*
FROM *C*
WHERE *A* [NOT] IN (SELECT . . .)

SQL

Theorem

The query portion of SQL-92 basically corresponds to Relational Algebra, and hence to safe range Relational Calculus.

Outline

1 Relational Databases
- Relational Model
- Relational Algebra
- Relational Model – Logical View

2 Relational Calculus

3 Domain Independence
- Domain Dependent Queries
- Domain Independent Queries
- Safe Range Queries
- SQL

4 Datalog
- Motivation
- Syntax
- Semantics
  - Model Theory
  - Fixpoint Theory

Recursion

- Some simple problems cannot be represented in relational calculus.
- Example: Reachability on deterministic graphs.
- Prototypical for LOGSPACE!
- Holds also for relational algebra, SQL-92 etc.

Transitive Closure

Key notion: Transitive Closure

Definition

Given graph $G = \langle V, E \rangle$, $E \subseteq V \times V$, and $a, b \in V$, the transitive closure $TC(G) \subseteq V \times V$ is:

$$
\begin{aligned}
TC(G) = \ & \{(x, y) \mid (x, y) \in E\} \\
& \cup \{(x, y) \mid (x, z) \in TC(G) \wedge (z, y) \in TC(G)\}
\end{aligned}
$$

Note: $TC(G)$ appears in its own definition.
In relational calculus we cannot refer to what we define.

Transitive Closure

Key notion: Transitive Closure

Definition

Given graph $G = \langle V, E \rangle$, $E \subseteq V \times V$, and $a, b \in V$, the transitive closure $TC(G) \subseteq V \times V$ is:

$$\begin{aligned} TC(G) = \; & \{(x, y) \mid (x, y) \in E\} \\ & \cup \{(x, y) \mid (x, z) \in TC(G) \wedge (z, y) \in TC(G)\} \end{aligned}$$

Note: $TC(G)$ appears in its own definition.

In relational calculus we cannot refer to what we define.

Transitive Closure

Key notion: Transitive Closure

Definition

Given graph $G = \langle V, E \rangle$, $E \subseteq V \times V$, and $a, b \in V$, the transitive closure $TC(G) \subseteq V \times V$ is:

$$\begin{aligned} TC(G) = \;& \{(x,y) \mid (x,y) \in E\} \\ & \cup \{(x,y) \mid (x,z) \in TC(G) \wedge (z,y) \in TC(G)\} \end{aligned}$$

Note: $TC(G)$ appears in its own definition.
In relational calculus we cannot refer to what we define.

- Idea: Use Horn clauses for named definitions.
- It is then possible to write definitions using the concept being defined.
- Positive Datalog

Outline

1 Relational Databases
- Relational Model
- Relational Algebra
- Relational Model – Logical View

2 Relational Calculus

3 Domain Independence
- Domain Dependent Queries
- Domain Independent Queries
- Safe Range Queries
- SQL

4 Datalog
- Motivation
- Syntax
- Semantics
  - Model Theory
  - Fixpoint Theory

Language Elements

- Set of extensional predicate symbols $\mathbf{PS_{EDB}}$
- Set of intensional predicate symbols $\mathbf{PS_{IDB}}$
- $\mathbf{PS_{EDB}} \cap \mathbf{PS_{IDB}} = \emptyset$
- Each predicate symbol has an associated arity $ar : \mathbf{PS_{EDB}} \cup \mathbf{PS_{IDB}} \to N_0$
- Set of constant symbols **CS**
- Set of variable symbols **VS**

Syntax

A Datalog rule is of the form:

$$r_1(t_{1_1}, \dots, t_{n_1}) \leftarrow r_2(t_{1_2}, \dots, t_{n_2}), \dots, r_m(t_{1_m}, \dots, t_{n_m}).$$

- $m \geq 1$
- $r_1 \in \mathbf{PS_{IDB}}$
- $r_2, \dots, r_m \in \mathbf{PS_{EDB}} \cup \mathbf{PS_{IDB}}$
- $t_{1_1}, \dots, t_{n_m} \in \mathbf{CS} \cup \mathbf{VS}$
- $\forall i\ 1 \leq i \leq m : ar(r_i) = n_i$
- $((t_{1_1} \cup \dots \cup t_{n_1}) \cap \mathbf{VS}) \subseteq ((t_{1_2} \cup \dots \cup t_{n_m}) \cap \mathbf{VS})$

Syntax

$$r_1(t_{1_1}, \ldots, t_{n_1}) \leftarrow r_2(t_{1_2}, \ldots, t_{n_2}), \ldots, r_m(t_{1_m}, \ldots, t_{n_m}).$$

- $H(r) = \{r_1(t_{1_1}, \ldots, t_{n_1})\}$
- $B(r) = \{r_2(t_{1_2}, \ldots, t_{n_2}), \ldots, r_m(t_{1_m}, \ldots, t_{n_m})\}$
- $V(r) = \{t_{1_1}, \ldots, t_{n_m}\} \cap \mathbf{VS}$
- $C(r) = \{t_{1_1}, \ldots, t_{n_m}\} \cap \mathbf{CS}$
- $H(r)$ is the head of r.
- $B(r)$ is the body of r.
- A Datalog program is a set of rules.

Outline

1. Relational Databases
   - Relational Model
   - Relational Algebra
   - Relational Model – Logical View
2. Relational Calculus
3. Domain Independence
   - Domain Dependent Queries
   - Domain Independent Queries
   - Safe Range Queries
   - SQL
4. Datalog
   - Motivation
   - Syntax
   - Semantics
     - Model Theory
     - Fixpoint Theory

Semantics

Intuitively: For each rule r, whenever $B(r)$ is true, $H(r)$ should also be true. $B(r) = \emptyset$ is considered to be true.

Different ways for defining the semantics:

- model theory
- fixpoint theory
- proof theory

Semantics

Intuitively: For each rule r, whenever $B(r)$ is true, $H(r)$ should also be true. $B(r) = \emptyset$ is considered to be true.
Different ways for defining the semantics:

- model theory
- fixpoint theory
- proof theory

Model Theory

Definition (Herbrand Universe)

$$\mathbf{HU}(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} C(r)$$

Definition (Herbrand Base)

$$\mathbf{HB}(\mathcal{P}) = \{r(t_1, \ldots, t_n) \mid r \in \mathbf{PS}_{\mathbf{EDB}} \cup \mathbf{PS}_{\mathbf{IDB}}, t_1, \ldots, t_n \in \mathbf{HU}(\mathcal{P}), ar(r) = n\}$$

- $\mathbf{HU}(\mathcal{P})$: Constants of the program (active domain!)
- $\mathbf{HB}(\mathcal{P})$: Ground atoms constructible from $\mathbf{HU}(\mathcal{P})$

Model Theory

Definition (Herbrand Universe)

$$\mathbf{HU}(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} C(r)$$

Definition (Herbrand Base)

$$\mathbf{HB}(\mathcal{P}) = \{r(t_1, \ldots, t_n) \mid r \in \mathbf{PS_{EDB}} \cup \mathbf{PS_{IDB}}, \\ t_1, \ldots, t_n \in \mathbf{HU}(\mathcal{P}), ar(r) = n\}$$

- $\mathbf{HU}(\mathcal{P})$: Constants of the program (active domain!)
- $\mathbf{HB}(\mathcal{P})$: Ground atoms constructible from $\mathbf{HU}(\mathcal{P})$

Model Theory

Definition (Herbrand Universe)

$$\mathbf{HU}(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} C(r)$$

Definition (Herbrand Base)

$$\mathbf{HB}(\mathcal{P}) = \{r(t_1, \ldots, t_n) \mid r \in \mathbf{PS}_{\mathbf{EDB}} \cup \mathbf{PS}_{\mathbf{IDB}}, \\ t_1, \ldots, t_n \in \mathbf{HU}(\mathcal{P}), ar(r) = n\}$$

- $\mathbf{HU}(\mathcal{P})$: Constants of the program (active domain!)
- $\mathbf{HB}(\mathcal{P})$: Ground atoms constructible from $\mathbf{HU}(\mathcal{P})$

Example: Herbrand Base

Example

$$\mathcal{P}_r = \{ \begin{array}{l} \texttt{arc}(\texttt{a},\texttt{b}). \\ \texttt{arc}(\texttt{b},\texttt{c}). \\ \texttt{reachable}(\texttt{a}). \\ \texttt{reachable}(\texttt{Y}) \leftarrow \texttt{arc}(\texttt{X},\texttt{Y}), \texttt{reachable}(\texttt{X}). \} \end{array}$$

$$\mathbf{HU}(\mathcal{P}_r) = \{\texttt{a},\texttt{b},\texttt{c}\}$$

$$\begin{array}{rl} \mathbf{HB}(\mathcal{P}_r) = & \{\texttt{arc}(\texttt{a},\texttt{a}), \texttt{arc}(\texttt{a},\texttt{b}), \texttt{arc}(\texttt{a},\texttt{c}), \\ & \texttt{arc}(\texttt{b},\texttt{a}), \texttt{arc}(\texttt{b},\texttt{b}), \texttt{arc}(\texttt{b},\texttt{c}), \\ & \texttt{arc}(\texttt{c},\texttt{a}), \texttt{arc}(\texttt{c},\texttt{b}), \texttt{arc}(\texttt{c},\texttt{c}), \\ & \texttt{reachable}(\texttt{a}), \texttt{reachable}(\texttt{b}), \texttt{reachable}(\texttt{c})\} \end{array}$$

Example: Herbrand Base

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc}(\texttt{a},\texttt{b}). \\ \texttt{arc}(\texttt{b},\texttt{c}). \\ \texttt{reachable}(\texttt{a}). \\ \texttt{reachable}(\texttt{Y}) \leftarrow \texttt{arc}(\texttt{X},\texttt{Y}), \texttt{reachable}(\texttt{X}). \}$$

$$\mathbf{HU}(\mathcal{P}_r) = \{\texttt{a},\texttt{b},\texttt{c}\}$$

$$\mathbf{HB}(\mathcal{P}_r) = \{\texttt{arc}(\texttt{a},\texttt{a}), \texttt{arc}(\texttt{a},\texttt{b}), \texttt{arc}(\texttt{a},\texttt{c}), \\ \texttt{arc}(\texttt{b},\texttt{a}), \texttt{arc}(\texttt{b},\texttt{b}), \texttt{arc}(\texttt{b},\texttt{c}), \\ \texttt{arc}(\texttt{c},\texttt{a}), \texttt{arc}(\texttt{c},\texttt{b}), \texttt{arc}(\texttt{c},\texttt{c}), \\ \texttt{reachable}(\texttt{a}), \texttt{reachable}(\texttt{b}), \texttt{reachable}(\texttt{c})\}$$

Instantiation

Definition

Valuation $v_{\mathcal{P}}(r)$ of a rule r: Set of all substitutions $V(r) \rightarrow \mathbf{HU}(\mathcal{P})$

Definition (Instantiation of a rule r)

$Ground_{\mathcal{P}}(r) = \bigcup_{v \in v_{\mathcal{P}}(r)} v(r)$

Definition (Instantiation of a program $\mathcal{P}$)

$Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground_{\mathcal{P}}(r)$

Instantiation

Definition

Valuation $v_{\mathcal{P}}(r)$ of a rule r: Set of all substitutions $V(r) \rightarrow \mathbf{HU}(\mathcal{P})$

Definition (Instantiation of a rule r)

$Ground_{\mathcal{P}}(r) = \bigcup_{v \in v_{\mathcal{P}}(r)} v(r)$

Definition (Instantiation of a program $\mathcal{P}$)

$Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground_{\mathcal{P}}(r)$

Instantiation

Definition

Valuation $v_{\mathcal{P}}(r)$ of a rule r: Set of all substitutions $V(r) \to \mathbf{HU}(\mathcal{P})$

Definition (Instantiation of a rule r)

$Ground_{\mathcal{P}}(r) = \bigcup_{v \in v_{\mathcal{P}}(r)} v(r)$

Definition (Instantiation of a program $\mathcal{P}$)

$Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground_{\mathcal{P}}(r)$

Example: Instantiation

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc}(\texttt{a},\texttt{b}).\ \texttt{arc}(\texttt{b},\texttt{c}).\ \texttt{reachable}(\texttt{a}).$$
$$\texttt{reachable}(\texttt{Y}) \leftarrow \texttt{arc}(\texttt{X},\texttt{Y}), \texttt{reachable}(\texttt{X}).\ \}$$

$$\mathit{Ground}(\mathcal{P}_r) = \{\texttt{arc}(\texttt{a},\texttt{b}).\ \texttt{arc}(\texttt{b},\texttt{c}).\ \texttt{reachable}(\texttt{a}).$$
$$\texttt{reachable}(\texttt{a}) \leftarrow \texttt{arc}(\texttt{a},\texttt{a}), \texttt{reachable}(\texttt{a}).$$
$$\texttt{reachable}(\texttt{b}) \leftarrow \texttt{arc}(\texttt{a},\texttt{b}), \texttt{reachable}(\texttt{a}).$$
$$\texttt{reachable}(\texttt{c}) \leftarrow \texttt{arc}(\texttt{a},\texttt{c}), \texttt{reachable}(\texttt{a}).$$
$$\texttt{reachable}(\texttt{a}) \leftarrow \texttt{arc}(\texttt{b},\texttt{a}), \texttt{reachable}(\texttt{b}).$$
$$\texttt{reachable}(\texttt{b}) \leftarrow \texttt{arc}(\texttt{b},\texttt{b}), \texttt{reachable}(\texttt{b}).$$
$$\texttt{reachable}(\texttt{c}) \leftarrow \texttt{arc}(\texttt{b},\texttt{c}), \texttt{reachable}(\texttt{b}).$$
$$\texttt{reachable}(\texttt{a}) \leftarrow \texttt{arc}(\texttt{c},\texttt{a}), \texttt{reachable}(\texttt{c}).$$
$$\texttt{reachable}(\texttt{b}) \leftarrow \texttt{arc}(\texttt{c},\texttt{b}), \texttt{reachable}(\texttt{c}).$$
$$\texttt{reachable}(\texttt{c}) \leftarrow \texttt{arc}(\texttt{c},\texttt{c}), \texttt{reachable}(\texttt{c}).\ \}$$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \not\subseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \not\subseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \not\subseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \not\subseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Example: Herbrand Models

Example

$\mathcal{P}_r = \{$ $\texttt{arc(a,b). arc(b,c). reachable(a).}$
$\texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).}$ $\}$

$M_1 = \{$ $\texttt{arc(a,b), arc(b,c),}$
$\texttt{reachable(a), reachable(b), reachable(c)}\}$

$M_2 = \mathbf{HB}(\mathcal{P}_r)$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \ \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).} \\ \mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

$$M_1 = \{ \ \mathtt{arc(a,b), arc(b,c),} \\ \mathtt{reachable(a), reachable(b), reachable(c)}\}$$

$M_2 = \mathbf{HB}(\mathcal{P}_r)$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b).}\ \texttt{arc(b,c).}\ \texttt{reachable(a).}$$
$$\texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y)}, \texttt{reachable(X).}\ \}$$

$$M_1 = \{ \quad \texttt{arc(a,b)}, \texttt{arc(b,c)},$$
$$\texttt{reachable(a)}, \texttt{reachable(b)}, \texttt{reachable(c)}\}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \; \texttt{arc(a,b).}\ \texttt{arc(b,c).}\ \texttt{reachable(a).}$$
$$\texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y)}, \texttt{reachable(X)}.\ \}$$

$$M_1 = \{ \; \texttt{arc(a,b)}, \texttt{arc(b,c)},$$
$$\texttt{reachable(a)}, \texttt{reachable(b)}, \texttt{reachable(c)} \}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc}(\texttt{a},\texttt{b}).\ \texttt{arc}(\texttt{b},\texttt{c}).\ \texttt{reachable}(\texttt{a}).$$
$$\texttt{reachable}(\texttt{Y}) \leftarrow \texttt{arc}(\texttt{X},\texttt{Y}), \texttt{reachable}(\texttt{X}).\ \}$$

$$M_1 = \{ \quad \texttt{arc}(\texttt{a},\texttt{b}), \texttt{arc}(\texttt{b},\texttt{c}),$$
$$\texttt{reachable}(\texttt{a}), \texttt{reachable}(\texttt{b}), \texttt{reachable}(\texttt{c})\}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Minimal Models

Theorem

$\mathbf{HB}(\mathcal{P})$ *is always a model for any Datalog program* $\mathcal{P}$.

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* $MM(\mathcal{P})$.

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by $MM(\mathcal{P})$

Note: Each element of $MM(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Minimal Models

Theorem

HB($\mathcal{P}$) *is always a model for any Datalog program* $\mathcal{P}$.

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* $MM(\mathcal{P})$.

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by $MM(\mathcal{P})$

Note: Each element of $MM(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Minimal Models

Theorem

$\mathbf{HB}(\mathcal{P})$ *is always a model for any Datalog program* $\mathcal{P}$*.*

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* $MM(\mathcal{P})$*.*

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by $MM(\mathcal{P})$

Note: Each element of $MM(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Minimal Models

Theorem

$\mathbf{HB}(\mathcal{P})$ *is always a model for any Datalog program* $\mathcal{P}$*.*

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* $MM(\mathcal{P})$*.*

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by $MM(\mathcal{P})$

Note: Each element of $MM(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Concept: Operator

"If we assume that all atoms in I are true, which other atoms must be true in order to satisfy the program?"

- Start with $I = \emptyset$ (nothing is true).
- Define operator $\mathbf{T}_{\mathcal{P}}$.
- Apply $\mathbf{T}_{\mathcal{P}}$, until there are no further additions.
- The obtained result (fixpoint) defines the semantics.

Immediate Consequences

Definition (Operator $\mathbf{T}_{\mathcal{P}}$ for Datalog program $\mathcal{P}$)

Given an interpretation I,

$$\mathbf{T}_{\mathcal{P}}(I) = \{h \mid r \in Ground(\mathcal{P}), B(r) \subseteq I, h \in H(r)\}$$

- $\mathbf{T}_{\mathcal{P}}(I)$ extends I, such that unsatisfied rules (w.r.t. I) become satisfied.
- Other rules may become unsatisfied w.r.t. $\mathbf{T}_{\mathcal{P}}(I)$.
- $\Rightarrow$ Iterative application.

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc}(\texttt{a},\texttt{b}).\ \texttt{arc}(\texttt{b},\texttt{c}).\ \texttt{reachable}(\texttt{a}). \\ \texttt{reachable}(\texttt{Y}) \leftarrow \texttt{arc}(\texttt{X},\texttt{Y}), \texttt{reachable}(\texttt{X}).\ \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\texttt{arc}(\texttt{a},\texttt{b}),\ \texttt{arc}(\texttt{b},\texttt{c}),\ \texttt{reachable}(\texttt{a})\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\texttt{reachable}(\texttt{b})\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\texttt{reachable}(\texttt{c})\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\texttt{arc}(\texttt{a},\texttt{b}),\ \texttt{arc}(\texttt{b},\texttt{c}),$ $\texttt{reachable}(\texttt{a}),\ \texttt{reachable}(\texttt{b}),\ \texttt{reachable}(\texttt{c})\}$

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b). arc(b,c). reachable(a).} \\ \texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).} \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\texttt{arc(a,b), arc(b,c), reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\texttt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\texttt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\texttt{arc(a,b), arc(b,c),}$ $\texttt{reachable(a), reachable(b), reachable(c)}\}$

Example: Immediate Consequences

Example

$\mathcal{P}_r = \{$ arc(a,b). arc(b,c). reachable(a).
reachable(Y) $\leftarrow$ arc(X,Y), reachable(X). $\}$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{$arc(a,b), arc(b,c), reachable(a)$\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{$reachable(b)$\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{$reachable(c)$\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{$arc(a,b), arc(b,c), reachable(a), reachable(b), reachable(c)$\}$

Example: Immediate Consequences

Example

$\mathcal{P}_r = \{$ `arc(a,b). arc(b,c). reachable(a).`
`reachable(Y)` $\leftarrow$ `arc(X,Y), reachable(X).` $\}$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\texttt{arc(a,b)},\ \texttt{arc(b,c)},\ \texttt{reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\texttt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\texttt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\texttt{arc(a,b)},\ \texttt{arc(b,c)},$ $\texttt{reachable(a)},\ \texttt{reachable(b)},\ \texttt{reachable(c)}\}$

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b). arc(b,c). reachable(a).} \\ \texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).} \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\texttt{arc(a,b), arc(b,c), reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\texttt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\texttt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\texttt{arc(a,b), arc(b,c),}$ $\texttt{reachable(a), reachable(b), reachable(c)}\}$

Properties of $\mathbf{T}_\mathcal{P}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_\mathcal{P}(X) \subseteq \mathbf{T}_\mathcal{P}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_\mathcal{P}(sup(X)) = sup(\mathbf{T}_\mathcal{P}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

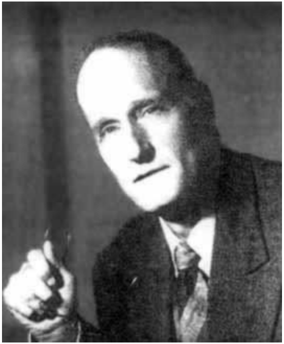
Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Tarski, Kleene

Bronisław Knaster
(1893–1990)

Alfred Tarski
(1902–1983)



Stephen Kleene
(1909–1994)

Existence of Fixpoints

Theorem

$\mathbf{T}_{\mathcal{P}}$ is monotone und continuous on the lattice of interpretations and subset relations.

Theorem (Knaster-Tarski)

For monotone operators on lattices a least fixpoint exists, and it is $inf(\{X \mid \mathbf{T}_{\mathcal{P}}(X) \subseteq X\})$

Construction of Fixpoints

Theorem (Kleene)

For continuous operators on lattices the least fixpoint can be computed by iteration starting from the infimum.

$\mathbf{T}_{\mathcal{P}}^{\omega} = sup(\{\mathbf{T}_{\mathcal{P}}^{i} \mid i \geq 0\})$,
$\mathbf{T}_{\mathcal{P}}^{0} = inf(V)$, $\mathbf{T}_{\mathcal{P}}^{i} = \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}^{i-1})$

Corollary

Our lattice is finite, therefore the least fixpoint of $\mathbf{T}_{\mathcal{P}}$ *can be computed by a finite number of itrations starting from* $\emptyset$.

$\mathbf{T}_{\mathcal{P}}^{\omega}$ – Minimal Model

Theorem

For all Datalog programs $\mathcal{P}$, we can show $\mathbf{T}_{\mathcal{P}}^{\omega} = MM(\mathcal{P})$.

Note: All consequences of a program can be computed by iteration over the immediate consequences.

$\mathbf{T}_{\mathcal{P}}^{\omega}$ – Minimal Model

Theorem

For all Datalog programs $\mathcal{P}$, we can show $\mathbf{T}_{\mathcal{P}}^{\omega} = MM(\mathcal{P})$.

Note: All consequences of a program can be computed by iteration over the immediate consequences.

Reminder: Horn and Goal Clauses, SLD Resolution

- A Horn clause is a clause containing at most one positive literal.
- A Goal clause is a clause containing no positive literal.
- SLD Resolution: Linear resolution, where at each step only goal clauses and (instances of) input clauses are used.

Theorem

SLD resolution is refutation complete for Horn clauses.

SLD Resolution for Datalog

- We can view each rule as a Horn clause.
- So SLD Resolution can be applied.
- Unification is simpler for Datalog because of absence of function symbols.

Definition (SLD Resolution Semantics)

Let $SLD(\mathcal{P})$ denote the set of ground atoms, for which an SLD refutation w.r.t. $\mathcal{P}$ exists.

Equivalence

Theorem

For all Datalog programs $\mathcal{P}$, we can show
$SLD(\mathcal{P}) = \mathbf{T}_{\mathcal{P}}^{\omega} = MM(\mathcal{P})$.

SLD Tree

Top-down and **bottom-up** views:

1. Top-down: Start at the root.
2. Bottom-up: Start at leaves.

$\mathbf{T}_{\mathcal{P}}^{\omega}$: Is like SLD bottom-up (on finite branches).

SLD Resolution – Termination

$\texttt{child\_of}(\texttt{charles}, \texttt{francis}).$
$\texttt{child\_of}(\texttt{francis}, \texttt{frida}).$
$\texttt{successor\_of}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{child\_of}(\texttt{X}, \texttt{Y}).$
$\texttt{successor\_of}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{child\_of}(\texttt{X}, \texttt{Z}), \texttt{successor\_of}(\texttt{Z}, \texttt{Y}).$
$\leftarrow \texttt{successor\_of}(\texttt{charles}, \texttt{X}).$

SLD Resolution – Termination

$$
\begin{array}{l}
\texttt{child\_of}(\texttt{charles}, \texttt{francis}). \\
\texttt{child\_of}(\texttt{francis}, \texttt{frida}). \\
\texttt{successor\_of}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{child\_of}(\texttt{X}, \texttt{Y}). \\
\texttt{successor\_of}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{successor\_of}(\texttt{X}, \texttt{Z}), \texttt{child\_of}(\texttt{Z}, \texttt{Y}). \\
\leftarrow \texttt{successor\_of}(\texttt{charles}, \texttt{X}).
\end{array}
$$

SLD Resolution – Termination

$\texttt{child\_of}(\texttt{charles}, \texttt{francis}).$
$\texttt{child\_of}(\texttt{francis}, \texttt{frida}).$
$\texttt{successor\_of}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{child\_of}(\texttt{X}, \texttt{Y}).$
$\texttt{successor\_of}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{successor\_of}(\texttt{X}, \texttt{Z}), \texttt{successor\_of}(\texttt{Z}, \texttt{Y}).$
$\leftarrow \texttt{successor\_of}(\texttt{charles}, \texttt{X}).$

Outline

1 Relational Databases
- Relational Model
- Relational Algebra
- Relational Model – Logical View

2 Relational Calculus

3 Domain Independence
- Domain Dependent Queries
- Domain Independent Queries
- Safe Range Queries
- SQL

4 Datalog
- Motivation
- Syntax
- Semantics
  - Model Theory
  - Fixpoint Theory

Simple Algorithms

From the semantic definitions, we can produce simple algorithms:

- Model Theory:
  Enumerate all subsets of $\mathbf{HB}(\mathcal{P})$, test whether they are models and take the minimal one.
- Fixpoint Theory:
  Extend $\emptyset$ by applying $\mathbf{T}_{\mathcal{P}}$ until a fixpoint is reached.
- Proof Theory:
  Use SLD Resolution bottom-up.

Simple Algorithms 2

Also for query answering we can find simple algorithms:

- Straightforward: Compute model and test whether query is true.
- Better: Use SLD resolution top-down.

Termination?

Simple Algorithms 2

Also for query answering we can find simple algorithms:

- Straightforward: Compute model and test whether query is true.
- Better: Use SLD resolution top-down.

Termination?