Predicate Calculus

- Formal language
  - True/False statements
  - Supports reasoning
- Usage
  - Integrity constraints
  - Non-procedural query languages
    - “what” rather than “how”
    - SQL (helps with formulation of some harder queries)
    - Prolog
  - Model theory (basis for design theory, techniques & tools)

Syntax – Symbols

- Truth Values: T F
- Constants
- Variables
- Functions: return a value
- Predicates: return T or F
- Logical Connectors
- Quantifiers: ∃ ∀
  - ∃xP(x) = P(x₁) ∨ P(x₂) ∨ ...
  - ∀xP(x) = P(x₁) ∧ P(x₂) ∧ ...
- Parentheses
Syntax – Terms & Atoms

- Terms: yield a value
  - Each variable and constant is a term.
  - Each function is a term (e.g., \( f(t_1, t_2) \) where \( t_1 \) and \( t_2 \) are terms).

- Atoms: yield T or F
  - \( T \)
  - \( F \)
  - Each predicate is an atom (e.g., \( p(t_1, t_2) \) where \( t_1 \) and \( t_2 \) are terms).

Syntax – Formulas

- Formulas
  - An atom is a formula.
  - If \( P \) & \( Q \) are formulas, so are \( \neg P \), \( (P \land Q) \), \( (P \lor Q) \), \( (P \rightarrow Q) \), \( (P \leftarrow Q) \)
  - If \( P \) is a formula and \( x \) is a variable, \( \exists x(P) \) and \( \forall x(P) \) are formulas

- Example: \( \exists x(x+y = 10 \lor \forall z \exists x(x > z)) \)
  - bound and free variables
  - scope of a variable
  - closed and open formulas
Semantics

- Interpretation
  - Specify domain D
  - Assign values in D to:
    - Constants (unique name assumption: literals denote themselves & constant symbols like $n$ have only one value)
    - $n$-ary functions: $f: D^n \rightarrow D$ (e.g., $+, -, \ldots$)
    - $n$-ary predicates: $p: D^n \rightarrow \{T, F\}$ (e.g., $<, =, \ldots$)

- Evaluation
  - Closed formulas: evaluation yields T or F
  - Open formulas: evaluation yields set of domain elements

- Examples
  - $D = \{0, \ldots, 99\}$
  - $\exists x (2x \mod 100 > 95)$ evaluates to T
  - $\forall x (2x \mod 100 > 95)$ evaluates to F
  - $2x \mod 100 > 95$ yields $\{48, 49, 98, 99\}$

Counting Quantifiers

- Notational shorthand
  - $\exists^1 x (P(x))$ T if there is exactly 1 value for which $P$ is true
  - $\exists^2 x (P(x))$ ... exactly 2 ...
  - $\exists^5 x (P(x))$ ... 5 or more ...
  - $\exists^3 x (P(x))$ ... less than 3 ...

- Equivalent expressions
  - $\exists^0 x (P(x)) \equiv \neg \exists x (P(x)) \equiv \forall x (\neg P(x))$
  - $\exists^1 x (P(x)) \equiv \exists x (P(x)) \land \forall x \forall y (P(x) \land P(y) \rightarrow x = y)$
Model Theory & Relational DBs

- Let relation names be predicates with a place for each attribute.
- Write integrity constraints as closed formulas.
- Example:

  \[ r(A, B, C) \]

  Domain = \{1, 2, 3\}

  \[ r(1, 2, 3) = T \]

  \[ r(2, 2, 2) = T \]

  \[ r(x, y, z) = F \text{ (for all others)} \]

  A is a Key.

Interpretation:

\[ \forall x \exists y, z (r(x, y, z)) \]

Model Theory, OSM, & Rel. DBs

- Relations:

<table>
<thead>
<tr>
<th>Room(x) has Cost(y)</th>
<th>Cost(x) is equivalent to Amount(y) in Currency(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>90</td>
</tr>
<tr>
<td>R2</td>
<td>80</td>
</tr>
<tr>
<td>R3</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
</tbody>
</table>

- Integrity Constraints:
  - Key Constraints
    - \( \forall x \exists y, z (\text{Room}(x) \text{ has Cost}(y)) \)
    - \( \forall x \forall y \exists z (\text{Cost}(x) \text{ is equivalent to Amount}(z) \text{ in Currency}(y)) \)
    - \( \forall x \forall y \exists z (\text{Cost}(z) \text{ is equivalent to Amount}(x) \text{ in Currency}(y)) \)
  - Referential-Integrity Constraints
    - \( \forall x \forall y \forall z (\text{Cost}(x) \text{ is equivalent to Amount}(y) \text{ in Currency}(z) \Rightarrow \exists w (\text{Room}(w) \text{ has Cost}(x))) \)
  - Attribute-Domain Constraints
    - \( \forall x (\exists y \forall z (\text{Cost}(y) \text{ is equivalent to Amount}(z) \text{ in Currency}(x) \Rightarrow x \in \{\text{Mark, Drachma, ...}\}) \)
    - ...
Valid Interpretations

- Interpretation
  - Domain
    - all values in the current DB
    - domain-closure assumption: these are the only elements that can be substituted for variables
  - Value Assignments
    - assign constants to themselves
    - use the closed-world assumption (tuple substitutions for predicates yield T, and all other substitutions yield F)
- Valid Interpretation
  - All closed formulas evaluate to T
  - DB integrity: the constraints of the DB are satisfied
  - Examples: the previous two examples show valid interpretations

Relational Calculus

- Queries are open formulas,
- General form: \{ <x_1, \ldots, x_n> | F(x_1, \ldots, x_n) \}
- Examples:
  - List $80 rooms.
    \{ <x> | Room(x) has Cost(80) \}
    \begin{array}{c|c|c}
    Room & R2 & R3 \\
    \hline
    \end{array}
  - List room costs in Marks.
    \{ <x, y> | \exists z (Room(x) has Cost(z) \land Cost(z) is equivalent to Amount(y) in Currency(Mark)) \}
    \begin{array}{c|c|c}
    Room & Amount \\
    \hline
    R1 & 150 \\
    R2 & 133 \\
    R3 & 133 \\
    \end{array}
Relational Calculus
Basic Project-Select-Join Examples

Get room information.
\{ <x, y, z, w> \mid r(x, y, z, w) \}

Get room number and room name of rooms that cost less than $85 and have 2 beds.
\{ <x, y> \mid \exists z(r(x, y, 2, z) \land z < 85) \}

Get name and address of guests arriving on 10 May.
\{ <x, y, z> \mid \exists u \exists v \exists w(g(u, x, y, z) \land s(u, v, 10 \text{ May}, w)) \}

Relational Calculus
Join, Renaming & Union Examples

Get name and address of guests who have a reservation for a room whose name is the same as the guest’s name.
\{ <x, y, z> \mid \exists a \exists b \exists c \exists d \exists e \exists f(g(a, x, y, z) \land r(b, x, c, d) \\
\land s(a, b, e, f)) \}

Get name and address of guests who have reservations for more than two days or reservations for two-bed rooms.
\{ <x, y: \text{StreetAddr}, z: \text{Location}> \mid \exists a \exists b \exists c \exists d \exists e \exists f( \\
g(a, x, y, z) \land s(a, b, c, d) \land (d > 2 \lor r(b, e, 2, f))) \}
Relational Calculus
Negation

Get guest number and name of guests not from Boston.
\[ \{ <x, y> | \exists z \exists w (g(x, y, z, w) \land w \neq \text{Boston}) \} \]

Get guest number of guests who do not have a reservation for room 1.
The following is not correct.
\[ \{ <x> | \exists y \exists z \exists w (s(x, y, z, w) \land y \neq 1) \} \]

Relational Calculus
Negation and Universal Quantification

Get guest number of guests who do not have a reservation for room 1. (continuation of example)

Find those who do and negate.
\[ \{ <x> | \neg \exists y \exists z \exists w (s(x, y, z, w) \land y = 1) \} \]
\[ = \{ <x> | \forall y \forall z \forall w \neg(s(x, y, z, w) \land y = 1) \} \]
\[ = \{ <x> | \forall y \forall z \forall w (\neg s(x, y, z, w) \lor y \neq 1) \} \]

Almost correct, but yields universal complement. Restrict by using relative complement.
\[ \{ <x> | \exists t \exists u \exists v (g(x, t, u, v) \land \neg \exists y \exists z \exists w (s(x, y, z, w) \land y = 1)) \} \]
Relational Calculus
Universal Quantification

Get name and address of guests who have reservations for all presidential suites (rooms with two beds).

\[
\{ <x, y, z> \mid \exists w(g(w, x, y, z) \land \\
\forall a \forall b \forall c(r(a, b, 2, c) \Rightarrow \exists d \exists e(s(w, a, d, e)))) \}
\]

Universal Quantification
and SQL Queries

\[
\{ <x, y, z> \mid \exists w(g(w, x, y, z) \land \\
\forall a \forall b \forall c(r(a, b, 2, c) \Rightarrow \exists d \exists e(s(w, a, d, e)))) \}
\]

= \{ <x, y, z> \mid \exists w(g(w, x, y, z) \land \\
\neg \neg a \forall b \forall c(\neg r(a, b, 2, c) \lor \exists d \exists e(s(w, a, d, e)))) \}

= \{ <x, y, z> \mid \exists w(g(w, x, y, z) \land \\
\neg a \exists b \exists c(r(a, b, 2, c) \land \\
\neg d \exists e(s(w, a, d, e)))) \}

= \text{select Name, StreetNr, City from Guest g where}
\text{not exists (select * from Room r where NrBeds = 2 and}
\text{not exists (select * from Reservation s where}
\text{g.GuestNr = s.GuestNr and r.RoomNr = s.RoomNr))}