Foundations of Databases

Recursion in Relational Algebra and Calculus

(Slides by Thomas Eiter and Wolfgang Faber)

Adding Recursion to Relational Algebra and Calculus

- Datalog can been seen as an extension of conjunctive queries with disjunction and recursion
- Logically, datalog thus offers $\wedge$, $\vee$, $\exists$, and recursion (but no $\neg$)
- Issue: Extend Relational Algebra resp. Relational Calculus with recursion
- Relational Algebra: variable assignments and looping construct
- Relational Calculus: recursion by fixpoint operators

Recursion in Relational Algebra

- **Problem**: Relational Algebra has only unnamed results (expressions).
- **Solution**: Introduce relation variables R, which may be assigned (the value of) expressions $Expr$, which have the same sort (resp. arity):

$$R := Expr$$

- The variable R may occur in $Expr$ itself:

$$T := R \cup (\pi_{1,4}\,(\sigma_{2=3}\,(R \times T)))$$

- Add imperative control structures (sequence, loop)

The While language

The While language extends relational algebra

- A While program is a finite sequence of assignments and while statements.
- A While statement has the form

  while change do
  begin
  <loop body>
  end

  where *<loop body>* is recursively a While program, and nesting of loops is finite

Semantics of While

A While program P is evaluated on a database instance $\mathbf{I}$ from $inst(\mathbf{R})$ as follows:

- Each relation $R \in \mathbf{R}$ is initialized to $\mathbf{I}(R)$.
- Each relation $S \notin \mathbf{R}$ is initialized to $\emptyset$.
- Process the statements in sequential order.
- For an assignment $R := Expr$, the result of evaluating $Expr$ on the current relation values is assigned to R
- The body of a While statement is executed as long as some relation value changes
- The result of the computation, $P(I)$, is the final result assigned to a designated output (query) relation, if the computation terminated (otherwise, undefined)

Example

A While program for the transitive closure of a graph `G: From, To`:

> *T := G;*
> *while change do*
> *begin*
> *T :=* $T \cup \pi_{From,To}(\rho_{A \leftarrow To}(T) \bowtie \rho_{A \leftarrow From}(G))$
> *end*

- The program terminates for each (finite) input $\mathbf{I}$
- T contains the transitive closure of graph encoded by $\mathbf{I}$

- **Problem:** Program P might not terminate
- Example (`G: From, To`):

  D := $\rho_{A \leftarrow From}(\pi_{From}(G)] \cup \rho_{A \leftarrow To}(\pi_{To}(G))$;
  while change do
  begin
  G := ($\rho_{From \leftarrow A}(D) \times \rho_{From \leftarrow A}(D)) \setminus G$;
  end

- **Theorem.** Whether a given While program P terminates on every $\mathbf{I}$ is undecidable
- Note: Whether P terminates on a given $\mathbf{I}$ is decidable (exact complexity later)

While$^+$ Programs

- Avoid termination problem by change in the semantics: Assignments are "inflationary"

  $$R+ = Expr$$

  add the value of $Expr$ to R
- The resulting language is called While$^+$
- **Proposition.** For each input database $\mathbf{I}$, $P(I)$ is well-defined
- Variants of While, While$^+$: instead of *"while change do"*:
  - *"while Expr* $\neq \emptyset$ *do"* in While
  - *"while Expr*$_1$ $\neq$ *Expr*$_2$ *do"* in While$^+$

  do permit the same expressiveness.

Recursion in Relational Calculus

- First Way: Assignments and loops as in Relational Algebra
- Proviso here: Active domain semantics for relational calculus
- More logic-oriented construct: Fixpoint-Operator
- Example: Transitive closure of graph G

$$\varphi(T) = G(x,y) \vee T(x,y) \vee \exists z(T(x,z) \wedge G(z,y))$$

Free variables: x, y; T is a relational variable

Define the value of T, given a valuation of G, as the limit of the sequence $\{J_i\}_{i \geq 0}$

$$\begin{aligned} J_0 &:= \emptyset, \\ J_i &:= \varphi(J_{i-1}), \quad i > 0. \end{aligned}$$

- For each input G, the limit exists and equals J_k, for some $k \geq 0$
- J_k is a *fixpoint* of the operator defined by $\varphi(\cdot)$ on the valuations of T on the active domain (wrt. G)
- This fixpoint is denoted by $\mu_T(\varphi(T))$
- The variable T and the variables x, y are bound to μ_T
- In general, $\mu_T(\varphi)$ may not be defined:

$$\varphi(T) = (x = 0 \wedge \neg T(0) \wedge \neg T(1)) \vee (x = 0 \wedge T(1)) \vee (x = 1 \wedge T(0))$$

Partial Fixpoint Operator

- Let $\mathbf{R}$ be a database schema, let T be a fresh n-ary relation, and let $\mathbf{S}$ be the schema $\mathbf{R} \cup \{T\}$.
- Let $\varphi(T)$ be a formula using T and relations in $\mathbf{R}$, with n free variables.
- Given $\mathbf{I} \in inst(\mathbf{R})$, $\mu_T(\varphi(T))$ denotes the limit of the sequence $\{J_i\}_{i \geq 0}$, if it exists,

$$\begin{aligned} J_0 &:= \emptyset, \\ J_i &:= \varphi(J_{i-1}), \quad i > 0. \end{aligned}$$

  where $\varphi(J_{i-1})$ denotes the result of evaluating φ on the database instance $\mathbf{J}_{i-1} \in inst(\mathbf{S})$ such that
  - $\mathbf{J}_{i-1}(R) = \mathbf{I}(R)$ for each $R \in \mathbf{R}$, and
  - $\mathbf{J}_{i-1}(T) = J_{i-1}$.

Partial Fixpoint Logic

- $\mu_T(\varphi)$ denotes a new n-ary relation (if defined), which can be used in more complex formulas.
- Examples: Let $\varphi(T) = G(x,y) \vee T(x,y) \vee \exists z(T(x,z) \wedge G(z,y))$

  $\mu_T(\varphi(T))(a,x), \qquad \neg\mu_T(\varphi(T))(x,y)$
- Partial fixpoint logic, CALC+μ, is the extension of Relational Calculus with μ
- Formulas are built from atoms by applying the RC operators ($\wedge$, $\vee$, $\exists$, $\neg$) and the μ operator.
- If $\varphi(T)$ has n free variables, T has arity n, and $e_1, \ldots, e_n$ are variables or constants, then $\mu_T(\varphi(T))(e_1, \ldots, e_n)$ is a formula
- Note: Nestings of μ_T are possible.

Partial Fixpoint Queries

- CALC+μ queries (aka *partial fixpoint queries*) are expressions Q of the form

$$\{e_1, \ldots, e_n \mid \varphi\}$$

  where the free variables $x_1, \ldots, x_m$ of φ are the variables occurring in the list of constants and variables $e_1, \ldots, e_n$.

- The query result of Q in input $\mathbf{I}$, denoted $Q(\mathbf{I})$, is undefined, whenever the evaluation of μ in a subformula of φ is undefined; otherwise, it is the set of all valuations ν for $e_1, \ldots, e_n$ such that $\varphi(\nu(x_1), \ldots, \nu(x_m))$ is defined and true (wrt. $\mathbf{I}$).

Examples

$$\varphi(T) = G(x, y) \vee T(x, y) \vee \exists z(T(x, z) \wedge G(z, y))$$

- all nodes reachable from a:

$$\{x : \mu_T(\varphi(T))(a, x)\}$$

- Complement of transitive closure

$$\{x, y : \neg\mu_T(\varphi(T))(x, y)\}$$

- Nodes that do not lie on a directed cycle:

$$\{x : \exists y(G(x, y) \vee G(y, x)) \wedge \neg\mu_T(\varphi(T))(x, x)\}$$

Inflationary Fixpoint Queries

- Problem similar as with While queries: Undefineness
- Similar remedy: compute fixpoints in inflationary manner

  Replace in definition of $\mu_T(\varphi(T))$

  $$J_i := \varphi(J_{i-1}), \quad i > 0.$$

  by

  $$J_i := J_{i-1} \cup \varphi(J_{i-1}), \quad i > 0.$$

  Equivalently, replace $\varphi(T)$ by $T(\vec{x}) \vee \varphi(T)$, where $\vec{x}$ are the free variables of $\varphi(T)$.

- The resulting operator is denoted $\mu^+_T(\varphi(T))$.
- The emerging set of queries are the CALC+μ^+ queries or (*inflationary fixpoint queries*, aka *fixpoint queries*)

Fixpoint logic: Examples

- Transitive closure query:

  $$\{x, y \mid \mu^+_T(G(x,y) \vee \exists z(T(x,z) \wedge G(z,y)))(x,y)\}$$

  Note: "$T(x,y)$" is implicitly added by the semantics.

- Same-Generation query ($\mathbf{R} = \{Par, Person\}$):

$$\{x, y \mid \mu^+_T((Person(x) \wedge x = y) \vee \\ \exists u, v(Par(x,u) \wedge T(u,v) \wedge Par(y,v)))(x,y)\}$$

While$^{(+)}$ vs CALC+$\mu^{(+)}$

Theorem. Suppose that in Relational Algebra expressions special constant relations $R_a := \{\langle a \rangle\}$, for each $a \in \mathbf{dom}$, may be used. Then,

1. While^{+} = CALC+μ^{+}
2. While = CALC+μ

- This can be shown by structural simulations: encode While$^{(+)}$ programs in CALC+$\mu^{(+)}$ (using active domain semantics)
- Vice versa, evaluate CALC+$\mu^{(+)}$ expressions with While$^{(+)}$ programs
- Relation constants R_a are needed to produce constant query output
  Example: CALC+μ^{+} query $\{x \mid x = a\}$.

Normal Forms

- Nested recursion in CALC+$\mu^{(+)}$ resp. in While$^{(+)}$ does not add expressivity
- Each CALC+$\mu^{(+)}$ query is equivalent to a query of the form

$$\{\vec{x} \mid \mu_T^{(+)}(\varphi(T))(\vec{t})\}$$

  where $\varphi(T)$ contains no $\mu^{(+)}$
- In fact, $\varphi(T)$ can be an *existential* formula
- Analogous normal forms hold for While$^{(+)}$ programs
- Proof: via equivalence to extensions of datalog with negation
- **Difficult open question:** CALC+μ = CALC+μ^{+} ?

Recursion in SQL

- **Problem**: Same as in Relational Algebra.
- **Solution**: Name the resulting relation and allow to use it in its definition!

  Construct: **WITH**

  ```
  WITH RECURSIVE T(X,Y) AS (
     SELECT R.X, R.Y
     FROM R
   UNION
     SELECT R.X, T.Y
     FROM R, T
     WHERE R.Y = T.X
  ) Query
  ```

- **Semantics:** Also here a fixpoint.

Indirect Recursion in SQL-3

```
WITH RECURSIVE
  EVEN (N) AS
   (VALUES (0) UNION SELECT M+1 FROM ODD),
  ODD (M) AS
   (SELECT N+1 FROM EVEN)
SELECT * FROM EVEN WHERE N < 10
```

Non-linear Recursion in SQL

```
WITH RECURSIVE
  DESCENDANT (N, V) AS (
    SELECT K, E FROM CHILD
    UNION
    SELECT N1.N, N2.V
      FROM DESCENDANT AS N1, DESCENDANT AS N2
      WHERE N1.V = N2.N)
SELECT N FROM DESCENDANT WHERE V = 'Adam'
```

Non-linear recursion is not allowed in SQL-3; will perhaps be allowed in future standards.

Final comment: Current commercial relational implementations (e.g., Oracle, IBM/DB2) support the SQL-3 WITH clause (but check out the respective manuals for details).

Readings

- S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

  Chapter 14.