Databases 1

Logikai lekérdező nyelv: Datalog

Logika, mint lekérdező nyelv

- Abstract Query Languages:
  - Relational Algebra (procedural $\rightarrow$ optimization)
  - Logical QL: Datalog, Rel.Calculus (declarative)
- Datalog = 'Data'- Database, 'log'- logic,Prolog
- If-then logical rules have been used in many systems.
- Nonrecursive rules are equivalent to the core relational algebra.
- Recursive rules extend relational algebra and appear in SQL-99.

Intuitív bevezetés --- (1)

Tankönyv 10.2. fejezet példája (az ELJUT feladat)

- Jaratok(legitarsasag, honnan, hova, koltseg, indulas, erkezes) táblában repülőjáratok adatait tároljuk.

  Mely (x,y) párokra lehet eljutni x városból y városba?

- Datalogban felírva

  Eljut(x, y) <- Jaratok(l, x, y, k, i, e)
  Eljut(x, y) <- Eljut(x, z) AND Jaratok(l, z, y, k, i, e)

- Vagy másképp felírva Datalogban (mi a különbség?)

  Eljut(x, y) <- Jaratok(_, x, y, _, _, _)
  Eljut(x, y) <- Eljut(x, z) AND Eljut(z, y)

Intuitív bevezetés --- (2)

- Example 1: Ancestors

  ParentOf(parent,child)

  - Find all of Mary's ancestors

- Example 2: Company hierarchy

  Employee(ID,salary)

  Manager(mID,eID)

  Project(name,mgrID)

  - Find total salary cost of project 'X'

- Example 3: Airline flights

  Flight(orig,dest,airline,cost)

  - Find cheapest way to fly from 'A' to 'B'

Logika ismétlés

- Ítéletkalkulus
  - ítéletváltozók:

    - x, y, z, …

    - igaz/hamis értékek
  - kifejezések

    - konstans {I, H}

    - változó

    - ha e, e1, e2 kifejezés, akkor e1 AND e2, e1 OR e2, NOT e, és (e) is kifejezés
- Elsőrendű predikátumkalkulus

Datalog szabályok és lekérdezések

- Our first example of a rule uses the relations
  Frequents(drinker, bar),
  Likes(drinker, beer),
  Sells(bar, beer, price).
- The rule is a query asking for "happy" drinkers --- those that frequent a bar that serves a beer that they like.

Datalog szabályok felépítése

Happy(d) ← Frequents(d,bar) AND Likes(d,beer) AND Sells(bar,beer,p)

Head = consequent, a single subgoal

Body = antecedent = AND of *subgoals*.

Read this symbol “if”

Részcélok, predikátumok, atomok

- An *atom* is a *predicate*, or relation name with variables or constants as arguments.
- The head is an atom; the body is the AND of one or more atoms.
- Convention: Predicates begin with a capital, variables begin with lower-case.

Példa: Atom

(1) Sells(bar, beer, p)

The predicate = name of a relation

Arguments are variables (or constants).

(2) Proba(x, x, y, 5, ‘alma’)

Datalog szabályok jelentése

- A variable appearing in the head is distinguished (kitüntetett); otherwise it is nondistinguished (nem-kitüntetett változó).
- Rule meaning: The head is true for given values of the distinguished variables if there exist values of the nondistinguished variables that make all subgoals of the body true.

Example: Interpretation

Happy(d) ← Frequents(d,bar) AND

Likes(d,beer) AND Sells(bar,beer,p)

Distinguished variable

Nondistinguished variables

Interpretation: drinker *d* is happy if there exist a bar, a beer, and a price *p* such that *d* frequents the bar, likes the beer, and the bar sells the beer at price *p*.

Datalog szabályok kiértékelése --- (1)

- Approach 1: consider all combinations of values of the variables.
- If all subgoals are true, then evaluate the head.
- The resulting head is a tuple in the result.

Example: Rule Evaluation

Happy(d) $\leftarrow$ Frequents(d,bar) AND

Likes(d,beer) AND Sells(bar,beer,p)

FOR (each d, bar, beer, p)

IF (Frequents(d,bar), Likes(d,beer), and Sells(bar,beer,p) are all true)

add Happy(d) to the result

- Note: set semantics so add only once.
- Set semantics vice versa bag semantics

Datalog szabályok kiértékelése --- (2)

- Approach 2: For each subgoal, consider all tuples that make the subgoal true.
- If a selection of tuples define a single value for each variable, then add the head to the result.

Happy(d) ← Frequents(d,bar) AND
    Likes(d,beer) AND Sells(bar,beer,p)

FOR (each f in Frequents, i in Likes,
    and s in Sells)

IF (f[1]=i[1] and f[2]=s[1] and i[2]=s[2])
    add Happy(f[1]) to the result

Milyen problémák merülnek fel? (később jön)

- Relations are finite sets.
- We want rule evaluations to be finite and lead to finite results.
- “Unsafe” rules like $P(x) \leftarrow Q(y)$ have infinite results, even if *Q* is finite.
- Even $P(x) \leftarrow Q(x)$ requires examining an infinity of *x*-values.

Aritmetikai részcélok

- In addition to relations as predicates, a predicate for a subgoal of the body can be an arithmetic comparison.
- We write arithmetic subgoals in the usual way, e.g., $x < y$.

Példa: Aritmetikai részcélok

- A beer is “cheap” if there are at least two bars that sell it for under $2.

Cheap(beer) ← Sells(bar1,beer,p1) AND
Sells(bar2,beer,p2) AND p1 < 2.00
AND p2 < 2.00 AND bar1 <> bar2

Negált részcélok

- NOT in front of a subgoal negates its meaning.
- Example: Think of Arc(a,b) as arcs in a graph.
  - S(x,y) says the graph is not transitive from *x* to *y* ; i.e., there is a path of length 2 from *x* to *y*, but no arc from *x* to *y*.

$$S(x,y) \leftarrow Arc(x,z) \text{ AND } Arc(z,y) \text{ AND NOT } Arc(x,y)$$

Biztonságos szabályok

- A rule is *safe* if:
  1. Each distinguished variable,
  2. Each variable in an arithmetic subgoal, and
  3. Each variable in a negated subgoal,

  also appears in a nonnegated,

  relational subgoal, amivel az x korlátozott:
  - pred(x, y, …) argumentuma (értéke a táblából)
  - vagy x=c (konstans)
  - vagy x=y (ahol y korlátozott)
- Safe rules prevent infinite results.

Példa: Nem biztonságos szabályokra

- Each of the following is unsafe and not allowed:
  1. $S(x) \leftarrow R(y)$
  2. $S(x) \leftarrow R(y)$ AND $x < y$
  3. $S(x) \leftarrow R(y)$ AND NOT $R(x)$
- In each case, an infinity of *x* 's can satisfy the rule, even if *R* is a finite relation.

A biztonságos szabályok előnyei

- We can use “approach 2” to evaluation, where we select tuples from only the nonnegated, relational subgoals.
- The head, negated relational subgoals, and arithmetic subgoals thus have all their variables defined and can be evaluated.

Datalog programok

- Datalog program = collection of rules.
- In a program, predicates can be either
  1. EDB = Extensional Database = stored table.
  2. IDB = Intensional Database = relation defined by rules.
- Never both! No EDB in heads.

Datalog programok kiértékelése

- As long as there is no recursion, we can pick an order to evaluate the IDB predicates, so that all the predicates in the body of its rules have already been evaluated.
- If an IDB predicate has more than one rule, each rule contributes tuples to its relation.

Példa: Datalog program

- Using EDB Sells(bar, beer, price) and Beers(name, manf), find the manufacturers of beers Joe doesn't sell.

JoeSells(b) ← Sells('Joe''s Bar', b, p)

Answer(m) ← Beers(b,m)

AND NOT JoeSells(b)

Példa: Kiértékelése

- Step 1: Examine all Sells tuples with first component 'Joe''s Bar'.
  - Add the second component to JoeSells.
- Step 2: Examine all Beers tuples (b,m).
  - If *b* is not in JoeSells, add *m* to Answer.

Datalog kifejező ereje

- Without recursion, Datalog can express all and only the queries of core relational algebra.
  - The same as SQL select-from-where, without aggregation and grouping.
- But with recursion, Datalog can express more than these languages.

Következik

- Relációs algebrai kifejezésfák átírása nemrekurzív Datalogba.
- Mi a leggyakrabban előforduló típus, amiből építkezek? $\Pi_{Lista}(\sigma_{Felt}(R \bowtie S \bowtie \ldots))$

  Ezt a komponenst támogatja legerősebben az SQL is: SELECT lista

  FROM táblák összekapcsolása

  WHERE felt

  Ez felel meg egy Datalog szabálynak…