Extensions of the Relational Model

More structure:

* ∗ complex object model (RC^{cv}, ALG^{cv})
* ∗ OO models

More expressive power:

* ∗ iteration and recursion
  * ⇒ Datalog
  * ⇒ Fixed-point Logics
* ∗ second-order quantification

More interpretation:

* ∗ constraint model
  * ⇒ in the queries
  * ⇒ as representation of databases

Why?

- Modeling is hard
  * ∗ everything has to be a *flat* relation (1NF)
- Can't express simple (graph) properties:
  * ∗ connectivity (and therefore transitive closure)
  * ∗ parity
- from the DB point of view, everything is a constant

Complex values

- data types (sorts):

$$\tau = \iota \mid \langle a_1 : \tau_1, \ldots, a_k : \tau_k \rangle \mid \{\tau\}$$

- interpretation:

$$\begin{aligned} [\![\iota]\!] &= \mathbf{dom} \\ [\![\langle a_1 : \tau_1, \ldots, a_k : \tau_k \rangle]\!] &= \{\langle a_1 : v_1, \ldots, a_k : v_k \rangle : v_i \in [\![\tau_i]\!]\} \\ [\![\{\tau\}]\!] &= \{V : V \subseteq [\![\tau]\!], V \text{ finite}\} \end{aligned}$$

Complex values (cont.)

- Queries:
  1. Atomic formulas: $R(x)$, $x = y$, $x.a_i = y$, $x \in y$.
  2. Connectives: $\wedge, \vee, \neg, \forall, \exists$.
  3. Constructed terms: $\{y : \varphi(y)\}$ (similarly for tuples)
  4. Query: $\{x : \varphi(x)\}$.
- "all people who liked all movies they have seen":

$$\{x : \{y : liked(x, y)\} \subseteq \{y : saw(x, y)\}\}$$

- Similarly to RC, we can define an algebra for this language.

Complex values (cont.)

- extends the expressive power (substantially):

  can express any query with hyper-exponential data complexity.
- transitive closure? (homework).

Expressive power

- iteration and recursion
  $\Rightarrow$ Datalog
  $\Rightarrow$ Fixed-point Logics
- second-order quantification

Datalog

- the language of function-free Horn clauses:

  $h(x_1, \ldots, x_k) \leftarrow g_1(y_1, \ldots, y_l), \ldots, g(z_1, \ldots, z_m)$

  $\Rightarrow$ $h(x_1, \ldots, x_k)$ head of the clause

  $\Rightarrow$ $g_i(\ldots)$ (sub-)goals
- an atomic clause: has only a head (a fact) $\Rightarrow$ *the database*
- a query ?- $g(\ldots)$

Datalog: An Example

- a *transitive closure* query over a graph:

$$tc(x, y) \leftarrow e(x, y)$$
$$tc(x, y) \leftarrow e(x, z), tc(z, y)$$

- two ways to compute the answer:

  $\Rightarrow$ resolution

  $\Rightarrow$ bottom up fixed point computation

Datalog: what happened to negation?

- positive Datalog: always a minimal model
- stratified Datalog

  $\Rightarrow$ no recursion through negation (still ok)

  $\Rightarrow$ well-founded semantics:

$$whitewins(x) \leftarrow move(y, x), \neg blackwins(y)$$

Fixpoints and Second-order Logic

- Various kinds of fixed-point logics:

  $\Rightarrow$ add the least fixed point operator $\mu X.\varphi(X)$
- Second order logic too strong:

  [Fagin] SOE captures all NP-complete problems (on graphs)

  [Immerman] descriptive complexity

Constraint Extensions

- extension of the

  ⇒ query language

  ⇒ way the data is stored

- takes advantage of natural structure of many domains

  ⇒ time (order constraints)

  ⇒ space (linear constraints)

Constraint Database

- atomic formulas and their meaning
- a *defineble set*

  ⇒ using quantifier-free formulas

- a constraint tuple:

  conjunction of atomic constraints (arity?)

- a constraint relation:

  finite set of constraint tuples

  ⇒ a disjunction of conjunctions of atomic constraints

Notation: Ω signature of the constraint theory

σ database schema

Databases as Formulas

- Relational Databases:

  ⇒ a (constraint) tuple: a conjunction of $x_i = v_i$

  ⇒ a (constraint) relation: a disjunction of tuples

- Constraint Database:

  ⇒ constraints from richer theory:

  * order constraints over Z
  * linear inequalities over R
  * polynomial inequalities over R
  * linear constraints over Z (?)

- function symbols vs. relations

Constraint Queries

- abstract (unrestricted) query
- concrete (constraint) query

- consistency

$$\|D_1\| = \|D_2\| \supset \|\varphi(D_1)\| = \|\varphi(D_2)\|$$

- closure

$$\varphi(D) \text{ is definable}$$

Constraint Calculus

- first order logic over $\Omega \cup \sigma$

  FO+POLY: $\Omega = (+, *, 0, 1, <)$,
  FO+LIN: $\Omega = (+, -, 0, 1, <)$)

- every query is consistent and closed
- what about other FO+X's??

Example

- for $R(x, y)$ a set of points; do al lthe points lie on a line?

$$\exists a, b. \forall x, y. R(x, y) \supset y = a * x + b$$

  $\Rightarrow$ is this "safe"? How can we find out a and b?

- do relations $R(x, y)$ and $S(x, y)$ intersect in the plane?

$$\exists x, y. R(x, y) \wedge S(x, y)$$

  $\Rightarrow$ imagine R was storing "objects" rather than points...

Query evaluation

- a (first-order) formula in $\Omega \cup \sigma$
- results ought to be quantifier-free again!

  QUANTIFIER ELIMINATION (well-studied concept in logic)

Definition. A theory $\mathcal{T}$ admits *quantifier elimination* if for every formula $\psi \in L_{\mathcal{T}}$ there is a *quantifier free* formula $\varphi \in L_{\mathcal{T}}$ such that $\mathcal{T} \models \psi \iff \varphi$.

Questions: is it effective? efficient? examples?

Benefits?

- does it help?

  ⇒ more databases? (yes, in general)

  [Grumbach and Su: Finitely Representable Databases. JCSS 55(2): 273-298 (1997)]

  ⇒ more compact encoding (yesss!)

  ⇒ more queries? (often not clear)

  ...and very hard to answer (intuition often wrong)

  [Benedikt et al.: Languages for Relational Databases over Interpreted Structures, PODS'97]

Problems

- classes closed under FO operations

  ⇒ what is allowed in the encoding vs.

  ⇒ what is allowed in the language
- complexity of query evaluation
- complexity (computability) of subsumption
- combinations with

  ⇒ Aggregation? (closure)

  ⇒ Datalog? (termination?)

  ⇒ Complex values?

Aggregation

- more interesting aggregates

  ⇒ length, area, diameter, convex hull
- problems with closure:

  $R = \{(x, y, z) : x \leq y \wedge x \geq 0 \wedge x \leq z\}$

  $[\text{area}(x, y) \text{ group by } z](R) = \{(z, w) : w = z^2/2\}$

  ⇒ even polynomial constraints don't suffice:

  $R = \{(x, y, z) : x^2 + y^2 \leq 1 \wedge x \leq z \wedge x \geq 0 \wedge y \geq 0\}$

  $[\text{area}(x, y) \text{ group by } z](R) =$

  $\{(z, w) : 0 \leq z \leq 1 \wedge w = \int_0^z \sqrt{1 - x^2}dx\}$
- variable independence
- safe aggregates

Constraint Datalog

- A Datalog program P: a set Horn clauses of the form

$$A \leftarrow C, B_1, \ldots, B_k$$

- an interpretation $\Rightarrow$ "what is true"

$$\{A(a_1, \ldots, a_k) : A \in \sigma, a_i \in D\}$$

- a (immediate) consequence operator:

$$T_P(I) = \{A\theta : \quad A \leftarrow C, B_1, \ldots, B_k \in P, \\ \theta \models C, \text{ and } B_i\theta \in I \quad \}$$

- A (minimal model) M_P of P: $M_P = T_P^\omega(\emptyset)$
  $\Rightarrow$ always exists
  $\Rightarrow$ unique and minimal

Constraint Datalog (cont.)

- interpretations: infinite in general
  $\Rightarrow$ can't be used in query evaluation
- constraint interpretation:
  representation using constraints
- $T_P^{\mathcal{C}}$: a constraint version of T_P
  [Toman: Memoing Evaluation for Constraint Extensions of Datalog, Constraints 2, 1997]
- termination (how many iterations of $T_P^{\mathcal{C}}$ do we need?)
  $\Rightarrow$ constraint compactness

Summary of Constraint classes

Domain	Constraints	Relational Calculus	Datalog	Datalog$^\neg$
D	$=$	AC_0 [1] LOGSPACE [1]	PTIME [1,4]	PTIME [1,4]
Q	$\neq, \leq, <$	AC_0 LOGSPACE	PTIME [4]	PTIME [2,4]
Z	$\neq, \leq, <, <_k$	NC	in DEXPTIME DEXPTIME-hard	non-elem. [3] undec.
N,Z	$\equiv_k$	PTIME ($\exists$ only) DEXPTIME	PTIME [4,5] DEXPTIME	PTIME [4,5] DEXPTIME
B_m	$=_{B_m}$	PTIME ($\exists$ only)	Π_2^p-hard	undec.
$P(D)$ $P(Z)$	$\subseteq, c \in, c \notin$	PTIME ($\exists$ only)	DEXPTIME-comp	undec.
N,Z	$1S$	NC	PSPACE-comp	PSPACE-comp
R	$<, +$	NC	undec.	undec.
N,Z	$<, +$	in $O(2^{n^{cn}})$	undec.	undec.
N,Z	lrp	in PTIME	undec.	undec.
R	$+, *$	NC	undec.	undec.
Z	$+, *$	undec.	undec.	undec.

[1] safe queries: the answers have to be finite relations; [2] inflationary semantics.
[3] syntactically restricted stratified Datalog$^\neg$; [4] data complexity.
[5] [illegible]