Lecture 10: Query Complexity

Thursday, February 1, 2001

Safe-FO = Relational Algebra

- Recall the 5 operators in the relational algebra: U, -, x, σ, Π

Theorem. A query is expressible in safe-FO iff it is expressible in the relational algebra

Proof

RA query E ➔ safe FO query ϕ

$R \Rightarrow R(x_1,...,x_n)$

$E \cup E' \Rightarrow \varphi(x_1,...,x_n) \vee \varphi'(x_1,...,x_n)$

$E - E' \Rightarrow \varphi(x_1,...,x_n) \wedge \neg\varphi'(x_1,...,x_n)$

$E \times E' \Rightarrow \varphi(x_1,...,x_n) \wedge \varphi'(y_1,...,y_m)$

$\sigma_{x=a}(E) \Rightarrow \varphi(...) \wedge (x = a)$

$\Pi_{x_1,...,x_n}(E) \Rightarrow \exists x_{n+1},...,\exists x_m.\varphi(x_1,...,x_n,x_{n+1},...,x_m)$

Proof

Define: Active domain formula:

$D_a = (\Pi_1(R) \cup \Pi_2(R) \cup ...) \cup (\Pi_1(S) \cup \Pi_2(S) \cup ...) \cup ...$

safe FO query ϕ ➔ RA query E

$R(x_1,...,x_n) \Rightarrow R$

$\varphi(x_1,...,x_n,y_1,...,y_m) \wedge \varphi'(x_1,...,x_n,z_1,...,z_p) \Rightarrow E \bowtie E'$

$\varphi(x_1,...,x_n,y_1,...,y_m) \vee \varphi'(x_1,...,x_n,z_1,...,z_p)$

$\Rightarrow E \times (D_a)^p \cup \Pi_{1,...,n,n+m+1,...,n+m+p,n+1,...,n+m}(E' \times (D_a)^m)$

$\neg\varphi(x_1,...,x_n) \Rightarrow (D_a)^n - E$

$\exists x_1.\varphi(x_1,...,x_n) \Rightarrow \Pi_{x_2,x_3,...,x_n}(E)$

No need for ∀ (why ?)

Examples

- Vocabulary: D(x), L(x,y), B(y)
- Find drinkers who like Bud:

$D(x) \wedge \exists y.(L(x,y) \wedge y = Bud) \Rightarrow D \cap (\Pi_x(\sigma_{y=Bud}(L)))$

Examples

- Find drinkers who like only Bud
  - SQL:
    select D.x
    from D
    where "Bud" = ALL (select L.y from L where D.x=L.x)
  - First Order Logic to Relational Algebra:

$$D(x) \wedge \forall y.(L(x,y) \Rightarrow y = Bud) \quad \Rightarrow \quad D - (\Pi_x(\sigma_{y \neq Bud}(L)))$$

  - Why ? Because:

$$D(x) \wedge \neg\exists y.(L(x,y) \wedge y \neq Bud) \quad \Rightarrow \quad D - (\Pi_x(\sigma_{y \neq Bud}(L)))$$

Discussion

- (safe)-FO and RA:
  - (safe)-FO: for *declarative query.*
  - RA: for *query plan.*
  - Theorem says: translate (safe)-FO to RA
  - In practice: need to consider "best" RA
- Query languages
  - (safe)-FO is just one instance; will discuss smaller and larger languages
  - All will express only computable, generic, and domain independent queries

Classical Logic v.s. Logic on Finite Models

- Recall:
  - given a model $\mathbf{D}=(D,R_1,...,R_k)$
  - and given a closed FO formula ϕ
  - we have defined what $\mathbf{D} \models \phi$ means
- A formula is *valid* if, for every $\mathbf{D}$, $\mathbf{D} \models \phi$
  - It is *finitely valid* if for every finite $\mathbf{D}$, $\mathbf{D} \models \phi$
- A formula is *satisfiable* if there exists $\mathbf{D}$ s.t. $\mathbf{D} \models \phi$
  - It is *finitely satisfiable* if there exists a finite $\mathbf{D}$ s.t. $\mathbf{D} \models \phi$
- Obviously: ϕ is valid iff not(ϕ) is not satisfiable

Classical Logic

- Notation: $\models \phi$ means ϕ is valid
- Notation: $\vdash \phi$ means ϕ is "provable"

Godel's Completeness Theorem: $\models \phi$ iff $\vdash \phi$

Corollary. The set of valid formulas is r.e.
  - Idea: enumerate all proofs

Church's Theorem: if $ar(R_i) > 1$ for some i, then the set of valid formulas is not decidable.

Corollary. The set of satisfiable formulas is not r.e.

Logic on Finite Models

Simple Fact: the set of finitely satisfiable formulas is r.e.
  - Idea: enumerate all finite models $\mathbf{D}$, and all formulas ϕ s.t. $\mathbf{D} \models \phi$

Trakhtenbrot's Theorem: if $ar(R_i) > 1$ for some i, then the set of finitely satisfiable formulas is not decidable

Corollary: the set of finitely valid formulas is not r.e.

An Example Where Finite/Infinite Differ

A formula ϕ that is satisfiable but not finitely satisfiable
  - "$<$ is a total order and has no maximal element"

$$\varphi = \forall x.\neg(x < x) \wedge$$
$$\forall x.\forall y.(x < y \vee y < x) \wedge \neg(x < y \wedge y < x) \wedge$$
$$\forall x.\forall y.\forall z.(x < y \wedge y < z) \rightarrow (x < z) \wedge$$
$$\forall x.\exists y.x < y$$

- It has an infinite model, but no finite one

Applications of Trakhtenbrot's Theorem

- Given a FO query ϕ, it is undecidable if ϕ is safe
  - Proof: the query $\neg R(x) \wedge \varphi$ is unsafe iff ϕ is finitely satisfiable
- Given two FO queries ϕ, ϕ', it is undecidable if they are equivalent, i.e. $\phi \Leftrightarrow \phi$'
  - Proof the queries $R(x) \wedge \varphi$ and $\neg R(x) \wedge \varphi$ are equivalent iff ϕ is not finitely satisfiable
- Trakhtenbrot's theorem for FO queries = like Rice's theorem for programs

13

More of This Stuff

- **Definition**. A query q is monotone if, for any two finite models
  D = (D, R_1, ..., R_k) and **D'** = (D', R_1', ..., R_k')
  s.t. $D \subseteq D'$, $R_1 \subseteq R_1'$, ..., $R_k \subseteq R_k'$
  we have q(**D**) $\subseteq$ q(**D'**).
- **Proposition**. It is undecidable if a query q in FO is monotone.
- **Proof**: why ?

14

Complexity of Query Languages

- All queries in a query language L are computable
- But usually L does not express *all* computable queries
  - Limited expressive power.
- Why do we care about such languages ?
  - Typically queries always terminate (e.g. FO)
  - Typically queries have a *low complexity* (next)

15

Complexity of Query Languages

For a query language L, define:

- **Data complexity**: fix a query q, how complex is it to evaluate q(**D**), for finite models **D**.
- **Expression complexity**: fix a finite model **D**, how complex is it to evaluate q(**D**), for queries q in L
- **Combined complexity**: how complex is it to evaluate q(**D**), for finite models **D** and queries q in L

16

Complexity of Query Languages

Formally:

- **Data complexity** of L is the complexity of deciding the set:

$$S_q = \{(D, \bar{a}) \mid D = \text{finite model, and } \bar{a} \in q(D)\}$$

  for some q in L
- **Combined complexity** of L is the complexity of deciding the set:

$$S = \{(D, \bar{a}) \mid D = \text{finite model, and } \bar{a} \in q(D),\ q \in L\}$$

17

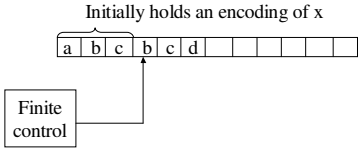
Who Cares About What

- Users: care about data complexity:
  - the query q is fixed; the database **D** is variable
- Database Systems: care about combined complexity:
  - both the query q and the database **D** are variable
- Database Theoreticians:
  - care about expression complexity, when they need to publish more papers ☺

18

Crash Course in Complexity Classes

- Fix a problem, i.e. a set S. Given a value x, how difficult is it for a Turing Machine to decide whether x ∈ S



19

Four Important Complexity Classes

- Let n = |x|
- **Definition**. S is in PTIME if there exists a Turing machine that on every input x takes $n^{O(1)}$ steps (i.e. $O(n^k)$, for some k > 0).
- Example: S = {G | G is connected}
  n = |G|, then one can check if G is connected in $O(n^3)$ steps (Warshall's algorithm)

20

Four Important Complexity Classes

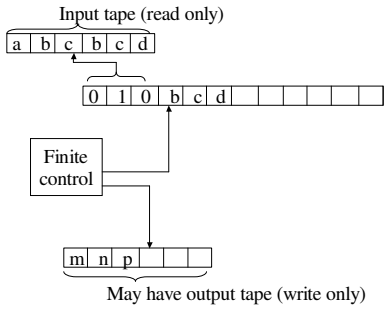
- **Definition**. S is in PSPACE if there exists a Turing machine for S that on every input x takes $n^{O(1)}$ space.
- Example. S = {G | G has a Hamiltonean path}
  space: O(n)
- Can run for a very long time: $c^{O(n)}$

21

Four Important Complexity Classes

- **Definition**. S is LOGSPACE if there exists a Turing machine for S that on every input takes O(log n) space.
- OOPS ! We need O(n) space to encode the input. How can we use *less* space ?
- Use two separate tapes:
  - Read only for the input: length = n
  - Read/write for work area: length = O(log n)
  - Use work tape as index into the input tape

22



23

Four Important Complexity Classes

- **Definition**. S is NLOGSPACE if there exists a nondeterministic Turing machine for S that on every input takes O(log n) space.

24

Example

- S = {(G, x, y) | there exists a path from x to y in G}
- u = x;
  for i = 1,n do
    if u = y then accept;
    u = (choose one of u's successors);
  endfor;
  reject;
- Need space for i: only takes O(log n)
- In English: transitive closure is in NLOGSPACE

25

Remarks

- How long can it run ? At most $2^{O(\log n)}=n^{O(1)}$.
- Hence:
  - LOGSPACE$\subseteq$NLOGSPACE $\subseteq$PTIME
- Suppose T_1, T_2 are Turing machines using O(log n) space. Can we construct a Turing machine computing $T_2 \circ T_1$? YES

26

FO Data Complexity

- Theorem. The data complexity for safe-FO is LOGSPACE.
- Proof. Compute bottom up. Example:
  $D(x) \wedge \forall y.(L(x,y) \Rightarrow \exists z.(L(z,y) \wedge \forall u.(L(z,u) \Rightarrow u = Bud)))$
  - T1 computes $L(z, u) \Rightarrow u = Bud$ needs 2log n space
  - T2 computes $\forall u.(L(z,u) \Rightarrow u = Bud)$ needs 2log n space
  - T3 computes $L(z, y)$ needs 2log n space
  - T4 computes $L(z,y) \wedge \forall u.(L(z,u) \Rightarrow u = Bud))$ needs 2log n space
  - …. Compose all these machines: one machine, O(log n)

27

Management of Variables in FO

- How much *time* did we need ?
  - Answer: $n^{O(\text{number of variables})}$
- FO^k = FO restricted to the variables x_1, …, x_k
- Find nodes (x,y) connected by a path of length 4:
  $\exists z_1 \exists z_2 \exists z_3.(R(x,z_1) \wedge R(z_1,z_2) \wedge R(z_2,z_3) \wedge R(z_3,y)$
  - FO^5, running time $O(n^5)$
  $\exists z.(R(x,z) \wedge \exists x.(R(z,x) \wedge \exists z.(R(x,z) \wedge R(z,y))))$
  - FO^3, running time $O(n^3)$

28

FO Combined Complexity

- Theorem. The combined (data+query) complexity in FO is in PSPACE.
- Theorem. The combined (data+expression) complexity of FO^k for fixed k is PTIME
- Proof: assignment.

29