Lecture 29

November 11, 2005

1 Satisfiability as a Database Query

We would like to know how we build a query on a database that solves satisfiability. Recall that the solution to the 3-SAT problem is a map $vars \to bools$ or $AP(\psi) \to \{0,1\}$. We proceed with an example to help us think about satisfiability as a database query. Consider 3-SAT $\varphi = \bigwedge C$ where $C : l_{c_1}, l_{c_2}, l_{c_3}$. Each l can be positive or negative, so there are 8 choices for each C. Note that if we disregard the order, there are 4 different combinations (or types) indicating how many predicates are negated: (+,+,+), (-,+,+), (-,-,+), (-,-,-).

Let's say some clause is $C_i = \neg p \vee \neg q \vee r$. Consider the truth table; there are 7 satisfying assignments and one counterexample (1,1,0). We can define the set of satisfying assignments for any C:

$R_0 = \{0,1\}^3 - (0,0,0)$
$R_1 = \{0,1\}^3 - (0,0,1)$ (means last predicate is negated)
$R_2 = \{0,1\}^3 - (1,0,0)$
$R_3 = \{0,1\}^3 - (1,1,0)$
$R_4 = \{0,1\}^3 - (1,1,1)$
...

For the k-SAT problem, each R will have $2^k - 1$ elements.

We can rewrite $\bigwedge C$ as M_φ ("matrix of φ") in the following way:

Take the 4 types of $C = l_{c_1}, l_{c_2}, l_{c_3}$ above and call them types 0, 1, 2, and 3. Let $A_i : R_{j_i}(x_{i_1}, x_{i_2}, x_{i_3})$ where j_i is the type of c_i. Then $M_\varphi = \bigwedge_i A_i$. Its structure is $B = (\{0,1\}, R_0, R_1, R_2, R_3)$. This can be viewed as an "extension" of NPC. We can think of this as a describing a database; we relate query M to database B.

2 Complexity

The complexity of the 3-SAT problem is NPC.

Algorithmic complexity can be better than it first appears if some parameter is known to be small. Example: the clique problem tries to find a k-clique for a graph $G = (V, E)$. It's NP-complete in general, but it's cubic if $k = 3$. We can apply this idea to database queries. Typically, database queries are small; databases are large.

The time complexity of a database query is $|B|^{|\varphi|}$, where $B = \{A, P^B, \ldots\}$ and $|\varphi|$ is the size of the query. The time complexity says that given a fixed query, it takes $|B|^{|\varphi|}$ time to compute a result. It's exponential, but it's much better than if it were $|\varphi|^{|B|}$.

The space complexity of a database query is $|\varphi|.log|B|$ space. It turns out to be *PSPACE*-complete.

The data complexity (where $|\varphi|$ is constant) is polynomial.

Query complexity ($|B|$ is constant) is exponential.

[Note: how do we deal with the fact that our answer can be arbitrarily large? Complexity might "really" be polynomial, but we're forced to give an answer that's exponential size. Answer: instead of $\varphi(B)$, make it a decision problem $\alpha \in^? \varphi(B)$.]

This is significantly more expensive than propositional logic. A more expressive language always requires more complexity to deal with. Moral of the story: expressibility is never free.

3 Conjunctive Queries

Databases are manufactured with an optimization for $|B|$ large and φ small, so there are not yet adapted to queries as the one in the first section.

We want to select something to satisfy a set of criteria. So queries can be represented naturally as conjunctive queries.

Are there things we cannot express in first-order logic?

3.1 Genericity

Recall the Isomorphism Theorem: let A, B be structures on some vocabulary, φ a first-order sentence. If $A \simeq B$, then $A \models \varphi$ iff $B \models \varphi$.

Considering formulas, not sentences: if $A \simeq^h B$ then $A, \alpha \models \varphi$ iff $B, h \circ \alpha \models \varphi$. Rewriting, this means $\alpha \in \varphi(A)$ iff $(h \circ \alpha) \in \varphi(B)$.

Define $A = (D, P_1^A, ..., f_1^A, ...)$

$h : D \to D'$ where $\to$ is 1-1 and onto.

$R \subseteq D^k$

$h(R) = \{< h(a_1), ..., h(a_k) >:< a_1, ..., a_k >\in R\}$

$f : D^k \to D$

$h(f)(b_1, ..., b_k) = h(f(h^{-1}(b_1), ..., h^{-1}(b_k)))$ where $b_i \in D'$.

New isomorphism theorem with these definitions (same theorem, different notation): Let A be a structure on some vocabulary, φ a first-order formula, and $h : D \to D'$ where $\to$ is 1-1 and onto. Then $h(\varphi(A)) = \varphi(h(A))$.

Intuition: imagine renaming Nathan to Ajay and vice versa in the Registrar's database. Switching the names in a query to the database has the same effect as switching the names in the database.

Example: Consider the Database with the two relations Student-Course $[SC(s_n, c_n)]$ and Course-Professor $[CP(c_n, p_n)]$. Consider the query "List all students that have the same name as a professor from when they take a class." We can express this as follows: $(\exists c)(SC(x,c) \wedge CP(c,x))$.

What if the query said "same *last* name" instead of "same name"? If the last name is not defined in the database (structure), there is no way we can express this query. Even if we make an isomorphic transformation of the database (structure), that is, changing the last names of of professors and students, there will be no query able to identify that. We can't rename, say, John Smith and Richard Smith to the same name without losing the fact that they are different people.

This is called *genericity*. It means that the answer to a query depends only on the information in the database. If a system doesn't have genericity, we're in big trouble.

3.2 Reachability and Expressibility

Now imagine a dedicated traveler wants to ask the Travelocity database if it's possible to travel from Houston to Timbuktu. This is a reachability query on a graph. We can create the following inductive definition:

$G = (V, E)$

$Reach_0(x,y) = E(x,y)$

$Reach_{i+1} = (\exists z)(\exists(x,z) \wedge Reach_i(z,y))$

That's fine for finite graphs, but what about infinite graphs? We need to express reachability in a way that works for both infinite and finite graphs. As it turns out, this is impossible.

Theorem 1 *Reachability is not first-order definable.*

Proof (by contradiction): Suppose reachability is first-order definable. Then $Connected = (\forall x)(\forall y)Reach(x,y)$ expresses connectivity. We will show that connectivity is not $EC_\triangle$.

By contradiction: suppose we have Φ such that $models(\Phi)$ is the set of all connected graphs. Consider the set $\Psi = \Phi \cup \{\neg Reach_1(x,y), \neg Reach_2(x,y), ...\}$. Ψ is connected (because of Φ) but at the same time disconnected (because of the $\neg Reach_i(x,y)$ parts which say there exists a pair that cannot be bridged in i steps). This is a contradiction, so Ψ is unsatisfiable.

The compactness theorem means that there is a finite subset with $Reach_a$ and $Reach_b$. Since it's finite, we can put an upper bound k such that $k > a, b$. Consider a graph that is a huge ring, big enough that the diameter $> k$. The graph is connected, but our finite subset will not agree. Thus such queries are not expressible in first-order logic.