Introduction to Datalog and Query Containment

CPS 296.1
Topics in Database Systems

Datalog

• A Prolog-like query language
• Declarative
  – A query specifies what the result should be (using logic) rather than how to compute it (using algebraic operators)
• Expressive
  – Supports recursion
  – Without recursion but with negation it is equivalent in power to relational algebra
• Has affected real practice (e.g., recursion in SQL3, magic transformation)

Conjunctive queries (CQ’s)

• Most common form of query; equivalent to select-project-join queries
• Datalog rule:
  \[ q(X) :- p_1(X_1), p_2(X_2), \ldots, p_n(X_n) \]
  – \( q(X) \) is called the head
  – \( p_i(X_i) \)'s are called the subgoals
  – Predicates \( p_i \)'s represent database relations
  – Tuples \( X, X_1, \ldots, X_n \) contain either variables or constants
  – The rule must be safe; that is, every variable that appears in the head must also appear in the body

CQ examples

• Database schema: parent(parent, child)
• CQ’s
  – parent-of-bart(\( X \)) :- parent(\( X \), “Bart”)
  – Equivalent to relational algebra query:
    \[ \pi_{\text{parent}} \sigma_{\text{child=“Bart”}} \text{parent} \]
  – grandparent(\( X, Y \)) :- parent(\( X \), \( Z \)), parent(\( Z \), \( Y \))
  – Equivalent to relational algebra query:
    \[ \pi_{p_1.\text{parent}, p_2.\text{child}} \rho_{p_1.\text{parent}=p_2.\text{parent}} \text{parent} \]
• Unsafe query
  – unsafe-query(\( X, Y \)) :- parent(\( X \), \( Z \))
  – Where would we get the value of \( Y \) in the query result?

Evaluating CQ’s

• Substitute constants for variables in the body of \( Q \) such that all subgoals becomes true
  – Result contains the head under the same substitution
• Example
  – grandparent(\( X, Y \)) :- parent(\( X, Z \), parent(\( Z, Y \))
  – Only substitutions that make both subgoals true
    • \( X \rightarrow “Abe” \); \( Z \rightarrow “Homer” \); \( Y \rightarrow “Bart” \)
    • \( X \rightarrow “Abe” \); \( Z \rightarrow “Homer” \); \( Y \rightarrow “Lisa” \)
  – These substitutions yield heads grandparent(“Abe”, “Bart”) and grandparent(“Abe”, “Lisa”), which are the result tuples

Example of CQ containment

• \( Q_1: p(X, Y) :- r(X, W_1), b(W, Z), r(Z, Y) \)
• \( Q_2: p(X, Y) :- r(X, W_1), b(W, W), r(W, Y) \)
• Claim: \( Q_1 \) contains \( Q_2 \)
• Proof
  – If \( p(x, y) \) is in \( Q_2 \), then there is some \( w \) such that \( r(x, w) \), \( b(w, w) \), and \( r(w, y) \) are true
  – For \( Q_1 \), make the substitution \( X \rightarrow x \); \( Y \rightarrow y \); \( W_1 \rightarrow w \); \( Z \rightarrow w \)
  – All subgoals of \( Q_1 \) are true, and the head of \( Q_1 \) becomes \( p(x, y) \)
  – Thus, \( p(x, y) \) is also in \( Q_1 \), proving that \( Q_1 \) contains \( Q_2 \)
Containment mappings

• A containment mapping is a mapping from variables of CQ $Q_1$ to variables for CQ $Q_2$, s.t.
  – Head of $Q_1$ becomes head of $Q_2$
  – Each subgoal of $Q_1$ becomes some subgoal of $Q_2$
    • It is not necessary that every subgoal of $Q_2$ is the target of some subgoal of $Q_1$

Containment mapping examples

• $Q_1$: $p(X, Y) \rightarrow r(X, W), b(W, Z), r(Z, Y)$
  • Containment mapping from $Q_1$ to $Q_2$:
    $X \rightarrow X; Y \rightarrow Y; W \rightarrow W; Z \rightarrow W$
  – No containment mapping from $Q_1$ to $Q_2$
    • $W$ cannot be mapped correctly

• $Q_2$: $p(X, Y) \rightarrow r(X, W), b(W, Z), r(Z, W)$
  – Containment mapping from $Q_1$ to $Q_2$:
    $X \rightarrow X; Y \rightarrow Y; Z \rightarrow Z; W \rightarrow W$
  – No containment mapping from $Q_1$ to $Q_2$
    • $X$ cannot be mapped correctly

Containment mapping theorem

• $Q_1$ contains $Q_2$ if and only if there exists a containment mapping from $Q_1$ to $Q_2$

• Some intuition
  – Given the containment mapping, and a substitution that proves $t \in Q_2$, we can construct a substitution to prove $t \in Q_1$
  – $Q_1$ may have more answers than $Q_2$ because $Q_2$ may have additional subgoals that further restrict its answers

Justification for “if”

• Let $\mu$: $Q_1 \rightarrow Q_2$ be a containment mapping
• Let $D$ be any database state
• Every tuple $t$ in $Q_2(D)$ is produced by some substitution $\sigma$ on the variables of $Q_2$ that makes $Q_1$’s subgoals all become facts in $D$
• Claim: $\sigma \circ \mu$ is a substitution for variables of $Q_1$ that produces $t$
  – $\sigma \circ \mu$ (a subgoal of $Q_1$) $\sigma$ (some subgoal of $Q_2$); therefore, it is supported by $D$
  – $\sigma \circ \mu$ (head of $Q_1$) $\sigma$ (head of $Q_2$) $t$
  ➢ So $t$ is in $Q_1(D)$ as well

Justification for “only if” (slide 1)

• Key idea: “frozen” CQ
  – Create a unique constant for each variable in $Q$
  – Frozen $Q$ is a database consisting of just the subgoals of $Q$, with the chosen constants substituted for variables
• Example: $Q_1$: $p(X, Y) : r(X, Y), r(Y, Z), r(Z, W)$
  – $X \rightarrow x; Y \rightarrow y; Z \rightarrow z; W \rightarrow w$
  – Frozen $Q_1$ contains three facts $r(x, y), r(y, z), r(z, w)$

Justification for “only if” (slide 2)

• Suppose $Q_1$ contains $Q_2$
• Let database $D$ be the frozen $Q_2$
• $Q_2(D)$ contains $t$, the frozen head of $Q_2$
• So $Q_2(D)$ must also contain $t$
• Let $\sigma$ be the substitution of constants from $D$ for the variables of $Q_1$ that makes each subgoal of $Q_1$ a fact in $D$ and yields $t$ as the head
• Let $t$ be the mapping that maps constants of $D$ to their unique, corresponding variable of $Q_2$ (the inverse mapping is used in constructing frozen $Q_2$)
• Claim: $t \circ \sigma$ is a containment mapping from $Q_1$ to $Q_2$
Justification for “only if” (slide 3)

• \( \tau \circ \sigma \) is a containment mapping from \( Q_1 \) to \( Q_2 \)

  - The head of \( Q_1 \) is mapped by \( \sigma \) to \( t \), and \( t \) is the frozen head of \( Q_2 \), so \( \tau \circ \sigma \) maps the head of \( Q_1 \) to the “unfrozen” \( t \), that is, the head of \( Q_1 \)
  - Each subgoal \( g_i \) of \( Q_1 \) is mapped by \( \sigma \) to some fact in \( D \), which is a frozen version of some subgoal \( g_j \) of \( Q_2 \); therefore, \( \tau \circ \sigma \) maps \( g_i \) to the “unfrozen” fact, that is, to \( g_j \) itself

Dual view of containment mappings

• A containment mapping, defined as a mapping on variables, induces a mapping on subgoals
  - We can alternatively define a containment mapping as a function on subgoals, thus inducing a mapping on variables
  - New containment mapping condition
    - The subgoal mapping does not cause a variable to be mapped to two different variables or constants, nor cause a constant to be mapped to a variable or a constant other than itself

Example of subgoal mapping

• Same example
  - \( Q_1 \): \( p(X, Y) :- r(X, W), b(W, Z), r(Z, Y) \)
  - \( Q_2 \): \( p(X, Y) :- r(X, W), b(W, W), r(W, Y) \)
  - Containment mapping on variables from \( Q_1 \) to \( Q_2 \):
    \( X \rightarrow X; Y \rightarrow Y; W \rightarrow W; Z \rightarrow W \)
  - Containment mapping on subgoals from \( Q_1 \) to \( Q_2 \):
    \( 1 \rightarrow 1 (r(X, W) \rightarrow r(X, W)) ;
    2 \rightarrow 2 (b(W, Z) \rightarrow b(W, W)) ;
    3 \rightarrow 3 (r(Z, Y) \rightarrow r(W, Y)) \)

Canonical databases

• Instead of looking for a containment mapping to test if \( Q_1 \) contains \( Q_2 \), apply the following test
  - Create a canonical database \( D \) that is the frozen body of \( Q_2 \)
  - Compute \( Q_1(D) \)
  - If \( Q_1(D) \) contains the frozen head of \( Q_2 \), then \( Q_1 \) contains \( Q_2 \); else not

Example of using canonical database

• \( Q_1 \): \( p(X) :- r(X, Y), r(Y, Z), r(Z, W) \)
• \( Q_2 \): \( p(X) :- r(X, Y), r(Y, X) \)

• Here is the test for whether \( Q_1 \) contains \( Q_2 \)
  - Choose constants \( X \rightarrow 0; Y \rightarrow 1 \)
  - Canonical database from \( Q_2 \) is \( D = \{ p(0), 1 \} \)
  - \( Q_1(D) = \{ p(0) \} \)
  - Since the frozen head of \( Q_2 \) is in \( Q_1(D) \), \( Q_1 \) contains \( Q_2 \)
  - Note that the instantiation of \( Q_1 \) that shows \( p(0) \) in \( Q_1(D) \) is \( X \rightarrow 0; Y \rightarrow 1; Z \rightarrow 0; W \rightarrow 1 \)
  - If we map 0 and 1 back to \( X \) and \( Y \) we get a containment mapping!

Built-in predicates

• \( Q_1 \): \( p(X, Y) :- r(X, Y), s(U, V), U <= V \)
• \( Q_2 \): \( p(X, Y) :- r(X, Y), s(U, V), s(V, U) \)
• \( Q_1 \) contains \( Q_2 \), but obviously there is no containment mapping (“<=” does not map to any subgoal in \( Q_2 \))
  - Instead, we need to consider a set of canonical databases, each of which has a complete ordering on the constants in the database
Results on query containment

- CQ’s: containment mapping or canonical databases
  - NP-complete, but not “hard” in practical situations (short queries, few pairs of subgoals with same predicate)
- Unions of CQ’s: same
  - Interesting result: A CQ is contained in a union of CQ’s iff this CQ is contained in some CQ in the union
- Built-in predicates: canonical databases
- Equivalence of Datalog queries: undecidable
  ➢ Many, many results in between…

Recursion in Datalog

- A predicate $p$ in a Datalog program is said to depend on a predicate $q$ if $q$ appears in a rule whose head is $p$
- A Datalog program is recursive if there is a cycle of dependency
  ➢ Example
    - ancestor($X, Y$) :- parent($X, Y$)
    - ancestor($X, Z$) :- ancestor($X, Y$), parent($Y, Z$)
    - “ancestor” depends on parent and itself; recursive

Meaning of recursive queries

- Start with the known facts in the database
- Apply the rules in the program in arbitrary order
- An application of a rule may derive new facts
- Repeat until no more facts can be derived
  ➢ Things get much hairier when we mix recursion with negation

Further reading