Planning for Information Gathering

Craig Knoblock
University of Southern California

These slides are based in part on slides from José Luis Ambite and Rao Kambhampati, which are in turn based in part on slides from Alon Halevy.

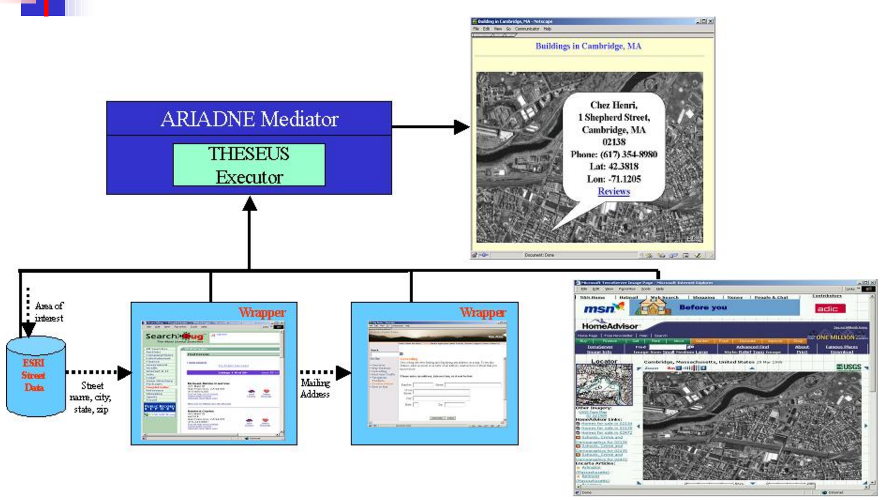
Planning on the Web

- Part I: Planning for Information Gathering
- Part II: Plan Execution for Information Gathering

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Information Gathering Example



Wrappers for Accessing Online Information Sources

- Wrappers provide uniform querying and data extraction

NAME	Casablanca Restaurant
STREET	220 Lincoln Boulevard
CITY	Venice
PHONE	(310) 392-5751

- State of the art in wrapper induction
  - Data extraction is based on Web page layout
    (Muslea *et al.* 1999, Kushmerick *et al.* 1997)
  - User labels examples of data on pages
  - Induction algorithm learns extraction rules for data

Planning for Information Gathering

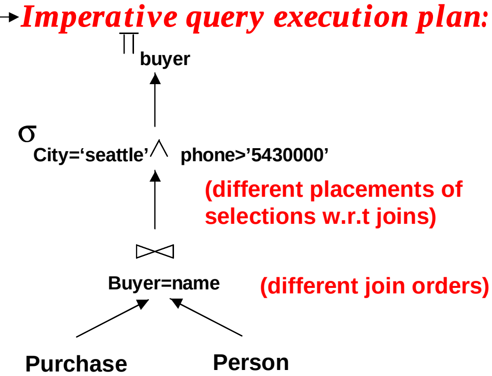
- Database query access planning
  - Specialized planner optimized for task
  - Sources are fixed
  - Mappings predefined in global schema
  - Complete plan is generated and then executed
  - Assumes closed-world and complete information

- Distributed, heterogeneous environments:
  - Sources and mappings are not fixed
  - Sources are autonomous
  - Overlapping and redundant sources
  - Sources may be incomplete
  - Sources may be unavailable
  - Additional information may be required to access a source
  - Access to sources may be costly

Database Query Access Plans

Declarative SQL query → ***Imperative query execution plan:***

SELECT S.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
Q.city='seattle' AND
Q.phone > '5430000'



Inputs:

- the query
- statistics about the data (indexes, cardinalities, selectivity factors)
- available memory

(different ways of scanning tables)

"shortest execution time"

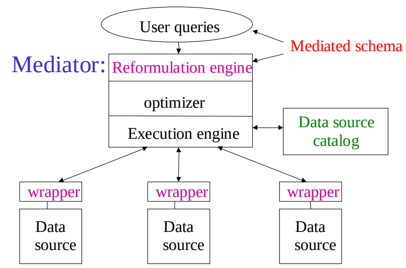
Ideally: Want to find best plan. Practically: Avoid worst plans!

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Virtual Integration Architecture

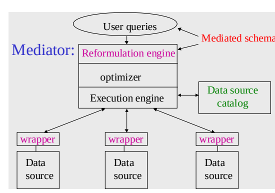
- Leave the data in the sources
- When a query comes in:
  - Determine the relevant sources to the query
  - Break down the query into sub-queries for the sources
  - Get the answers from the sources, and combine them appropriately
- Data is fresh. Approach scalable
- Issues:
  - Relating Sources & Mediator
  - Reformulating the query
  - Efficient planning & execution



Garlic [IBM], Hermes[UMD];Tsimmis, InfoMaster[Stanford]; DISCO[INRIA]; Information Manifold [AT&T]; SIMS/Ariadne[USC];Emerac/Havasu[ASU]

Desiderata for Relating Source-Mediator Schemas

- Expressive power: distinguish between sources with closely related data. Hence, be able to prune access to irrelevant sources.
- Easy addition: make it easy to add new data sources.
- Reformulation: be able to reformulate a user query into a query on the sources efficiently and effectively.
- Nonlossy: be able to handle all queries that can be answered by directly accessing the sources



Reformulation

- Given:
  - A query Q posed over the mediated schema
  - Descriptions of the data sources
- Find:
  - A query Q' over the data source relations, such that:
    - Q' provides only *correct answers* to Q, and
    - Q' provides *all* possible answers to Q given the sources.

Source Descriptions

Elements of source descriptions:

- Contents: source contains movies, directors, cast.
- Constraints: only movies produced after 1965.
- Completeness: contains *all* American movies.
- Capabilities:
  - Negative: source requires movie title or director as input
  - Positive: source can perform selections, joins, …

Approaches to Specification of Source Descriptions

- Global-as-View (GAV):

  Mediator relation defined as a view over source relations

  Ex: TSIMMIS (Stanford), HERMES (Maryland)
- Local-as-View (LAV):

  Source relation defined as view over mediator relations

  Ex: Information Manifold (AT&T), Tukwila(UW), InfoMaster (Stanford), Ariadne (USC)

View ~ named query ~ logical formula

Views and Conjunctive Queries

```
CREATE VIEW  Big-LA-buyers  AS
    SELECT  buyer, seller, price
    FROM     Person, Purchase
    WHERE  Person.city = "Los Angeles"  AND
                Person.name = Purchase.buyer AND
                Purchase.price > 10000
```

```
big-LA-buyers(Buyer,Seller, Price) :-
              person(Buyer, "Los Angeles"),
              purchase(Buyer, Seller, Product, Price),
              Price > 10000.
```

Datalog rule ~ view definition
Rule body ~ select-from-where construct of SQL

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Query Reformulation

Problem: rewrite the user query expressed in the mediated schema into a query expressed in the source schemas

Given a query Q in terms of the mediated-schema relations, and descriptions of the information sources,

Find a query Q' that uses only the source relations, such that

- Q' |= Q (i.e., answers are correct; i.e., Q' ⊆ Q) and
- Q' provides all possible answers to Q given the sources

Global-as-View (GAV)

Each mediator relation is defined as a view over source relations.

MovieActor(title,actor) ←
DB1(id,title,actor,year)

MovieActor(title,actor) ←
DB2(title,director,actor,year)

MovieReview(title, review) ←
DB1(id,title,actor,year) ^ DB3(id,review)

Query Reformulation in GAV

Query reformulation = rule unfolding+simplification

Query: *Find reviews for 'DeNiro' movies*

q(title,review) :- MovieActor(title,'DeNiro'),
MovieReview(title,review)

1. q'(title,review) :- DB1(id,title,'DeNiro',year), ~~DB1(id,title,actor,year'),~~ DB3(id,review)

Redundant

~~2. q'(title,review) :-~~
~~DB2(title,director,'DeNiro',year),~~
~~DB1(id,title,actor, year'), DB3(id,review)~~

Redundant wrt 1

Local-as-View (LAV)

- Each source relation is defined as a view over mediator relations

V1(title, year, director) $\xrightarrow{\subseteq}$ Movie(title,year,director,genre) ^ American(director) ^ year ≥1960 ^ genre = 'Comedy'

V2 (title, review) $\xrightarrow[\subseteq]{}$ Movie(title,year,director,genre) ^ year≥1990 ^ MovieReview(title, review)

Query Reformulation in LAV

Query: *Reviews for comedies produced after 1950*

q(title,review) :- Movie(title,year,director,'Comedy'), year ≥1950, MovieReview(title,review)

Reformulated query:
q'(title,review) :- V1(title,year,director),
V2(title,review)

q' ⊆ q

V1(title, year, director) → Movie(title,year,director,genre) ^ American(director) ^ year ≥1960 ^ genre = 'Comedy'

V2 (title, review) → Movie(title,year,director,genre) ^ year≥1990 ^ MovieReview(title, review)

Inverse-Rules Algorithm

[Duschka+1997]

Idea: Construct an equivalent logic program which evaluation yields the answer to the query

- The antecedent of the query and views is in term of mediator predicates
- Would like to have source predicates in antecedent so that program can be evaluated

⇒ Invert the rules

(simply by using standard logical manipulations)

The Inverse-Rules Algorithm: Example

V1(dept,course) $\rightarrow$ Enrolled(student,dept) ^ Registered(student,course)

$a \rightarrow b \equiv \neg a \vee b$

$\forall D,C\ [v1(D,C) \rightarrow \exists S\ [\ e(S,D) \wedge r(S,C)]]$

$\equiv \neg v1(D,C) \vee [e(f(D,C),D) \wedge r(f(D,C),C)]$

$\equiv [\neg v1(D,C) \vee e(f(D,C),D)] \wedge [\neg v1(D,C) \vee r(f(D,C),C)]$

$\equiv [v1(D,C) \rightarrow e(f(D,C),D)] \wedge [v1(D,C) \rightarrow r(f(D,C),C)]$

$\equiv$

$e(f(D,C),D) \leftarrow v1(D,C)$

$r(f(D,C),C) \leftarrow v1(D,C)$

The Inverse-Rules Algorithm: Example

q(D) $\leftarrow$ Enrolled(S,D) ^ Registered(S,"DB")

v1(D,C) $\rightarrow$ Enrolled(S,D) ^ Registered(S,C)

q(D) $\leftarrow$ Enrolled(S,D) ^ Registered(S,"DB")

Enrolled(f(D,C),D) $\leftarrow$ v1(D,C)

Registered(f(D,C),C) $\leftarrow$ v1(D,C)

q(D) $\leftarrow$ v1(D,"DB")

Ext(v1) = {("CS", "DB"), ("EE", "DB"), ("CS", "AI")}

Ext(q) = {("CS"), ("EE")}

GAV vs. LAV

- Not modular
  - Addition of new sources changes the mediated schema
- Can be awkward to write mediated schema without loss of information
- Query reformulation easy
  - reduces to view unfolding (polynomial)
  - Can build hierarchies of mediated schemas
- Best when
  - Few, stable, data sources
  - well-known to the mediator (e.g. corporate integration)
    - Garlic, TSIMMIS, HERMES

- Modular--adding new sources is easy
- Very flexible--power of the entire query language available to describe sources
- Reformulation is hard
  - Involves answering queries only using views (can be intractable)
- Best when
  - Many, relatively unknown data sources
  - possibility of addition/deletion of sources
    - Information Manifold, InfoMaster, Emerac

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Modeling Source Capabilities

Negative capabilities:

- A web-site may require certain inputs (in an HTML form) to answer a query
- Need to consider only valid query execution plans

Positive capabilities:

- A source may be database (understands SQL)
- Need to decide the placement of operations according to capabilities

Problem: how to describe and exploit source capabilities

Negative Capabilities: Binding Patterns

Sources:

$AAAIdb^{f}(X) \rightarrow AAAIPapers(X)$

$CitationDB^{bf}(X,Y) \rightarrow Cites(X,Y)$

$AwardDB^{b}(X) \rightarrow AwardPaper(X)$

Query: find all the award winning papers:

$q(X) \leftarrow AwardPaper(X)$

Recursive Rewritings

q(X) ← AwardPaper(X)

- Problem: *Unbounded* union of conjunctive queries

q1(X) ← AAAIdb(X), AwardDB(X)

q1(X) ← AAAIdb(X1), CitationDB(X1,X), AwardDB(X)

…

q1(X) ← AAAIdb(X1), CitationDB(X1,X2), …, CitationDB(Xn,X), AwardDB(X)

- Solution: Recursive Rewriting

papers(X) ← AAAIdb(X)

papers(X) ← papers(Y), CitationDB(Y,X)

q'(X) ← papers(X), AwardDB(X)

AAAIdb^{f} (X) → AAAIPapers(X)
CitationDB^{bf}(X,Y) → Cites(X,Y)
AwardDB^{b}(X) → AwardPaper(X)

Inverse-Rules Algorithm Binding Patterns

Sources:

AAAIdb^{f} (X) → AAAIPapers(X)

CitationDB^{bf}(X,Y) → Cites(X,Y)

AwardDB^{b}(X) → AwardPaper(X)

Query: find all the award winning papers:

q(X) ← AwardPaper(X)

Inverse-Rules Algorithm
Inverse + Domain Rules (1)

Inverted Rules:

AAAIPapers(X) ← AAAIdb(X)
Cites(X,Y) ← dom(X) ^ CitationDB(X,Y)
AwardPaper(X) ← dom(X) ^ AwardDB(X)

Domain Rules:

dom(Y) ← dom(X) ^ CitationDB(X,Y)
dom(X) ← AAAIdb(X)

Query:

q(X) ← AwardPaper(X)

Inverse-Rules Algorithm
Inverse + Domain Rules (2)

Simplyfing the program:

q(X) ← paper(X) ^ AwardDB(X)
paper(Y) ← paper(X) ^ CitationDB(X,Y)
paper(X) ← AAAIdb(X)

Outline

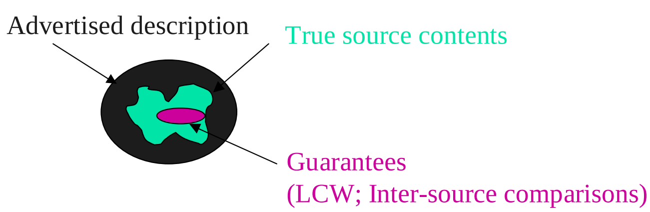
- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Managing Source Overlap

- Often, sources on the Internet have overlapping contents
  - The overlap is *not* centrally managed (unlike DDBMS—data replication etc.)
- Reasoning about overlap is important for plan optimality
  - We cannot possibly call all potentially relevant sources!
- Qns: How do we characterize and exploit source overlap?

Local Completeness Information

- If sources are incomplete, we may need to look at all of them
- Often, sources are *locally complete*
- Movie(title, director, year) complete for years after 1960, or for American directors
- Question: given a set of local completeness statements, is a query Q' a complete answer to Q?



Using LCW rules to minimize plans

Basic Idea:

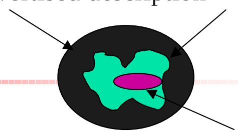
- If reformulation of Q leads to a union of conjunctive plans
  - $P_1 \vee P_2 \vee \ldots P_k$
- Then, if P_1 is "complete" for Q (under the given LCW information), then we can minimize the reformulation by pruning $P_2 \ldots P_k$
  - $[P_1 \wedge \text{LCW}]$ contains $P_1 \vee P_2 \vee \ldots P_k$

[Duschka, AAAI-97]

- For Recursive Plans (obtained when the sources have access restrictions)
  - We are allowed to remove a rule *r* from a plan *P*, if the "complete" version of *r* is already contained in *P-r*

Emerac [Lambrecht & Kambhampati, 99]

Example



- S1: Movie(title, director, year) (complete after 1960)
  S1(T,D,Y) → M(T,D,Y)
- S2: Show(title, theater, city, hour)(complete for Seattle)
  S2(T,Th,C,H) → Sh(T,Th,C,H)
  LCW: S2(T,Th,C,H) ← Sh(T,Th,C,H) & C = Seattle
- S3: Show(title, theater, city, hour)
  S3(T,Th,C,H) → Sh(T,Th,C,H)

- Query: Find movies and directors playing in Seattle
  Q(T,D) ← M(T,D,Y) & Sh(T,Th,C,H) & C = “Seattle”
- Plan: Combine S1 with S2 or S3
  Q(T,D) ← S1(T,D,Y) & S2(T,Th,C,H) & C = “Seattle”
  Q(T,D) ← S1(T,D,Y) & S3(T,Th,C,H) & C = “Seattle”
- Optimized Plan: Use LCW to prune S3
  Q(T,D) ← S1(T,D,Y) & S2(T,Th,C,H) & C = “Seattle”

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

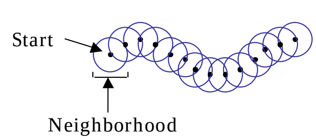
Planning by Rewriting
[Ambite & Knoblock, 1998]

- Efficiently generate an initial solution plan (possibly of low quality)
- Iteratively rewrite the current plan
  - using a set of declarative plan rewriting rules
  - improving plan quality
  - until an acceptable solution or resource limit reached

Efficient High-Quality Planning

Planning by Rewriting as Local Search

- PbR: efficient high-quality planning using local search
- Main issues:
  - <u>Selection of initial feasible point</u>: Initial plan generation
  - <u>Generation of a local neighborhood</u>: Set of plans obtained from application of the plan rewriting rules
  - <u>Cost function to minimize</u>: Measure of plan quality
  - <u>Selection of next point</u>: Next plan to consider -- determines how the global space is explored

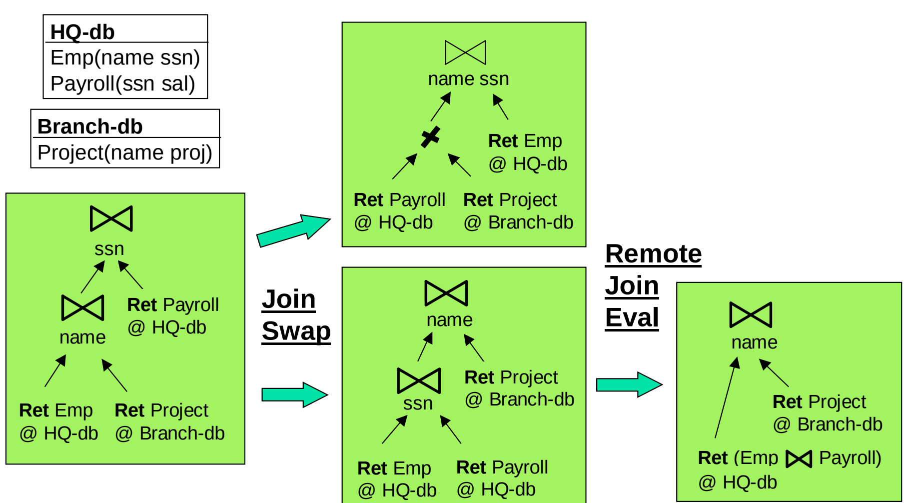


Planning by Rewriting for Query Planning in Mediators

- Initial plan generation: random parse of the query
- Plan rewriting rules: based on properties of:
  - relational algebra,
  - distributed environment,
  - integration axioms
- Plan quality: query execution time (size estimation)
- Search Strategies: gradient descent+restart, simulated annealing, variable-depth rewriting, ...

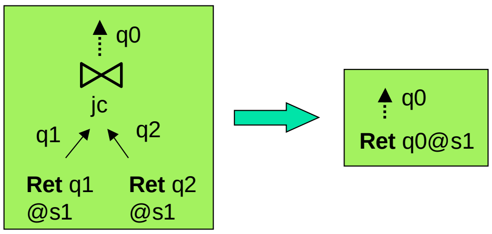
Query Planning in PbR

a(name sal proj) :- Emp(name ssn) ^ Payroll(ssn sal) ^ Projects(name proj)



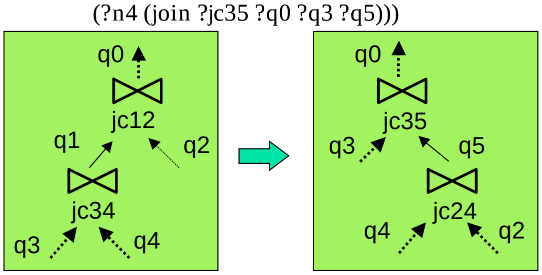
Rewriting Rules: Distributed Environment
remote-join-eval

(define-rule **remote-join-eval**
:if (:operators ((?n1 (retrieve ?source ?query1))
(?n2 (retrieve ?source ?query2)
(?n3 (join ?join-conds ?query0 ?query1 ?query2)))
:constraints (capability ?source join))
:replace (:operators (?n1 ?n2 ?n3))
:with (:operators ((?n4 (retrieve ?source ?query0)))))



Rewriting Rules: Relational Algebra
join-associativity

(define-rule :name **join-associativity**
:if (:operators ((?n1 (join ?jc34 ?q1 ?q3 ?q4)
(?n2 (join ?jc12 ?q0 ?q1 ?q2)))
:constraints (join-swappable ?jc34 ?q1 ?q3 ?q4 ?jc12 ?q0 ?q2 ;;in
?jc24 ?jc35 ?q5)) ;; out
:replace (:operators (?n1 ?n2))
:with (:operators ((?n3 (join ?jc24 ?q5 ?q4 ?q2))
(?n4 (join ?jc35 ?q0 ?q3 ?q5)))



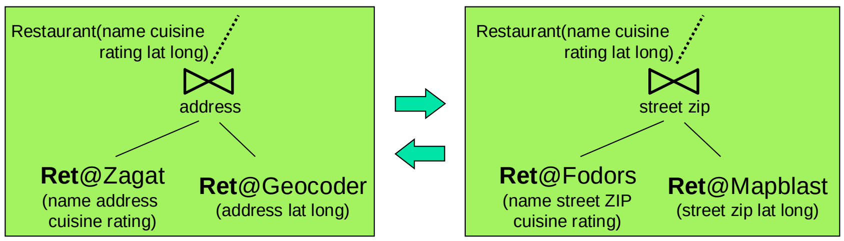
Rewriting Rules: Integration Axioms

- Rules computed from integration axioms relevant to query:

Restaurant(name cuisine rating lat long) =

a) Zagat(name address cuisine rating) ^ Geocoder(address lat long)

b) Fodors(name street zip cuisine rating) ^ Mapblast(street zip lat long)



PbR in Query Planning: Summary

- Operators: output, retrieve, assign, select, join, union
- Plan rewriting rules:
  - Relational algebra: join-swap, selection-push-in, selection-push-out, assignment-push-in, assignment-push-out, selection-assignment-swap, push-join-thru-union, and push-union-thru-join.
  - Distributed environment: source-swap, remote-join-eval, remote-selection-eval, and remote-assignment-eval.
  - Integration axioms: computed automatically from the relevant integration axioms for classes in the query
- Search: gradient descent + random restart
  - first-improvement
  - steepest descent

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Planning for the Internet Softbot
[Golden et al., 1996, Golden, 1998]

- XII and Puccini planners for the Internet Softbot
- Plans both gathering and manipulation actions
  - e.g., ls -a, chmod +r *
- Used to model Internet resources such as netfind
- Each resource modeled as a operator

```
Name: (netfind ?person)
Preconds:
    (current.shell csh)
    (isa netfind.server ?server)
    (firstname ?person ?firstname)
    (lastname ?person ?lastname)
    (or
      (person.city ?person ?keyword)
       (person.institution ?person ?keyword))
Postconds:
    (userid ?person !userid)
    (person.machine ?person !machine)
```

Netfind Operator from XII

Observational Effects and Knowledge Preconditions

- Observational Effects
  - Effect that changes the state of the world
    chmod + r foo.tex -- cause(readable(foo.tex))
  - Effect that changes the agent's model of the world
    wc -- observe(word.count (file, !word))
- Knowledge Preconditions
  - Information goal -- find-out(length (paper.tex, l))
  - Goals of achievement -- satisfy(readable (f) False)
- Verification Links
  - Alternative to knowledge preconditions
  - Assume secondary condition is true and then use an observational effect to determine whether it is true after execution

Sensing for Locally Complete Information

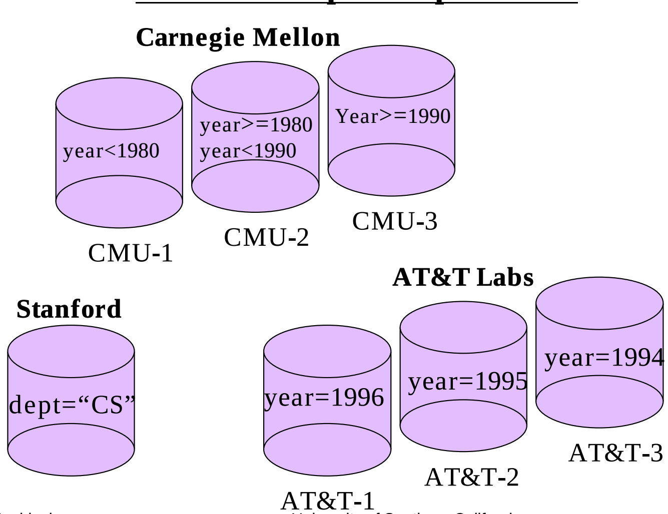
- Reasons about incomplete information
  - Uses LCW to reason about what it knows and what it doesn't know
  - e.g., ls -a * gives it locally complete information about the current directory
- Interleaves sensing actions to gather LCW information
  - LCW statements are a way of satisfying universally quantified goals
- Provides fine-grained reasoning
  - e.g., can request all recent techreports by X not already stored locally

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Sensing to Determine Relevant Sources [Ashish, Knoblock, & Levy, 1997]

Technical Report Repositories



Building a Discrimination Matrix

- Discrimination matrix specifies the relevant sources for each region of each attribute
- Approach:
  - Analyze source descriptions to build a discrimination matrix
  - Matrix partitions sources along some attribute
  - Discrimination matrix used to estimate the cost of querying with and without sensing
- Useful discriminations provided when:
  - Sources can be partitioned by some attribute
  - Exists another source that provides that attribute
- Example: Information about the year of a tech report reduces the relevant sources from 7 to 3

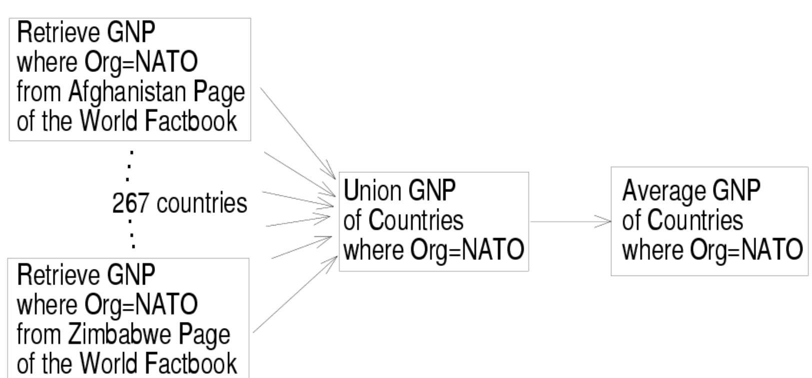
Discrimination Matrix

Region	Relevant Sources
< 1980	CMU-1, Stanford
[1980,1990)	CMU-2, Stanford
[1990,1994)	CMU-3, Stanford
[1994,1994]	CMU-3, Stanford, AT&T-1
[1995,1995]	CMU-3, Stanford, AT&T-2
[1996,1996]	CMU-3, Stanford, AT&T-3
> 1996	CMU-3, Stanford

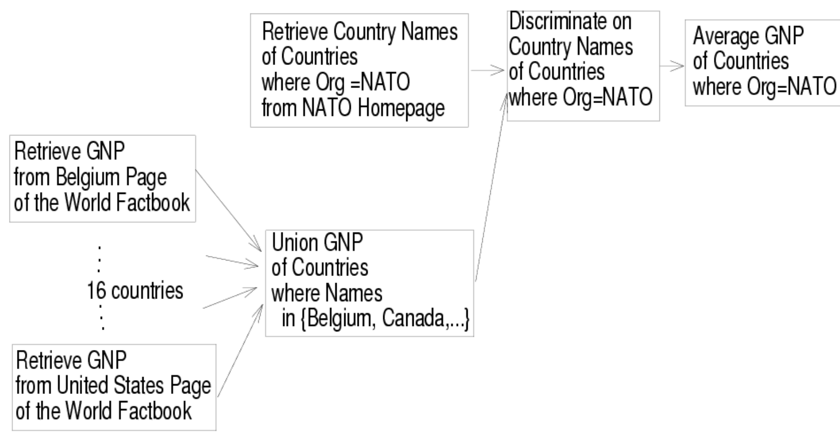
Planning with Discriminating Queries

- Consider inserting a discriminating query for any subquery that:
  - Requires accessing multiple sources
  - There exists a discriminating attribute in the matrix
- Compare the cost of no discrimination to the combined cost of discriminating and querying
- Since we cannot know the results of the discrimination, use the average estimated cost
- Potentially relevant sources: S = S1,...,S6
- Discriminating queries: R1, R2
- Possible plans: S, R1 S', R2 S'', R1 R2 S'''
  - R1: {{S1, S2}, {S3, S4, S5}, {S6}}
  - R2: {{S1}, {S2, S3}, {S4, S5, S6}}
  - R1 R2: {{S1}, {S2}, {S3}, {S4, S5}, {S6}}

Plan without Sensing



Plan with Sensing



Outline

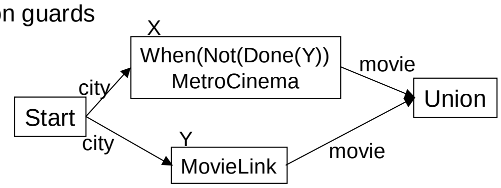
- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Contingent Planning for Information Gathering [Friedman & Weld '97]

- Use subsumption relationships to make a plan more resource conscious
  - Determined based on LCW statements
- Execution policies:
  - Brute force – ignore subsumption and execute everything greedily
  - Aggressive – execute multiple alternatives and cancel others once a subsumed source is successful
  - Frugal – execute the most general source first and only execute others if it fails

Augmenting the Plans

- Contingent plans
  - Operator can fire when its guard is true
  - Status variable for each operator
    - Sleeping, running, failed, and done
  - Approach:
    - Nodes initialized to running
    - Running nodes fired when input is available
    - Update status based on guards
  - Guards
    - Aggressive policy:
    - Frugal policy:
      - Failed(Y)



Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Composing Web Services

- Information sources only have inputs and outputs
  - Possibly with some additional constraints on those
- Services have:
  - Inputs and outputs
  - Preconditions and effects
- Could be cast as a traditional planning problem with preconditions and effects
- Example:
  - To purchase a book on Amazon has a precondition of having the money and has the effects of having the book and less money
- Services can be composed into compound services [McIlraith & Fadel, 2002]
  - Stored and reused similar to Macrops [Fikes, 1972]

Outline

- Information Gathering
- Planning for Information Gathering
  - View Integration
  - Query Reformulation
  - Source Capabilities
- Optimizing Information Gathering Plans
  - Removing Redundant Sources
  - Optimizing Sources and Queries
- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
  - Sensing to Optimize Plans
- Contingent Planning for Information Gathering
- Planning to Compose Web Sources
- Discussion

Discussion

- Is this planning?
  - Not in the sense of composing sequences of actions with interacting effects
  - Certainly in the broader sense of formulating a scheme or program for the accomplishment or attainment of some goal
- Good ideas can be shared across fields
  - Planning by rewriting based on traditional approaches to query planning
- Lots of interesting problems with real world applications
  - Optimizing the plans (e.g., interleaving sensing actions)
  - Interleaving source selection and plan optimization
  - Efficient execution of the plans (next class)

Bibliography

- Planning for Information Gathering
  - View Integration
    - Levy, Alon Y. (2000). Logic-based Techniques in Data Integration. *Logic Based Artificial Intelligence*, Edited by Jack Minker, Kluwer Publishers.
    - Halevy, Alon Y. (2001). Answering Queries Using Views: A Survey. VLDB Journal.
    - Duschka, Oliver M. (1997). Query Planning and Optimization in Information Integration. Ph.D. Thesis, Stanford University, Computer Science Technical Report STAN-CS-TR-97-1598.
    - Duschka, Oliver M. and Alon Y. Levy (1997). Recursive Plans for Information Gathering. *Proceedings of IJCAI-97l*

Bibliography

- Planning for Information Gathering
  - Traditional Planning Approaches
    - Lambrecht, Eric and Subbarao Kambhampati (1997). Planning for Information Gathering: A Tutorial Survey. ASU CSE Technical Report 96-017.
    - Knoblock, Craig A. (1995). Planning, Executing, Sensing, and Replanning for Information Gathering. In *Proceedings of IJCAI-95.*
    - Knoblock, Craig A. (1996) Building a planner for information gathering: A report from the trenches. In *Proceedings of AIPS-96.*
    - Kwok, Chung T. and Daniel S. Weld (1996). Planning to Gather Information. In *Proceedings of AAAI-96*.

Bibliography

- Optimizing Information Gathering Plans
  - Removing Redundant Sources
    - Duschka, Oliver M. (1997). Query Optimization Using Local Completeness. In *Proceedings of AAAI-97*.
    - Lambrecht, Eric and Subbarao Kambhampati, and Senthil Gnanaprakasam. (1999) Optimizing Recursive Information Gathering Plans. In *Proceedings of IJCAI-99*.
  - Optimizing Sources and Queries
    - Ambite, Jose Luis and Craig A. Knoblock (2000). Flexible and Scalable Cost-based Query Planning in Mediators: A Transformational Approach. *Artificial Intelligence Journal* , 118(1-2):115—161.
    - Jose Luis Ambite and Craig A. Knoblock (1997). Planning by Rewriting: Efficient Generating High-Quality Plans, In *Proceedings of AAAI-1997*.

Bibliography

- Interleaving Planning and Sensing
  - Sensing to Handle Incomplete Information
    - Golden, Keith, Oren Etzioni, and Daniel S. Weld (1996). Planning with Execution and Incomplete Information. University of Washington, Department of Computer Science, Technical Report UW-CSE-96-01-09.
    - Golden, Keith (1998) Leap Before you Look: Information Gathering in the PUCCINI Planner. In *Proceedings of AIPS-98*.
  - Sensing to Optimize Plans
    - Ashish, Naveen, Craig A. Knoblock, and Alon Y. Levy (1997). Information Gathering Plans with Sensing Actions, *Recent Advances in AI Planning: 4th European Conference on Planning, ECP'97* . Springer-Verlag, New York, 1997.

Bibliography

- Contingent Planning for Information Gathering
  - Friedman, Marc and Daniel S. Weld. (1997). Efficiently Executing Information-Gathering Plans. In *Proceedings of IJCAI-97*, Nagoya, Japan, August 1997.
- Planning to Compose Web Sources
  - McIlraith, Shiela and Ronald Fadel (2002). Planning with Complex Actions. In *Proceedings of AIPS-2002 Workshop: Is There Life Beyond Operator Sequencing? - Exploring Real-World Planning.*