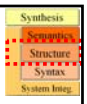
Projects Overview

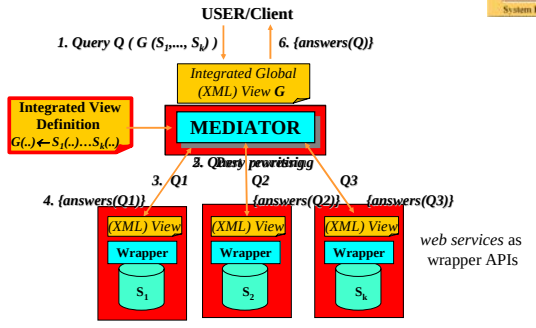
- 1. (T+P) Data Integration/Mediation
  - Implement the Global-as-View (GAV)/ view unfolding approach (intro to it: TODAY)
  - Examples, documentation, presentation (needed for all projects)
- 2. (T: deep) Data Integration/Mediation
  - Read/learn about other DI algorithms: Local-as-View (LAV), mixed approaches
  - Summarize, report, present
- 3. (T: broad) Intro/Overview on Schema Matching approaches
- 4. (T) Knowledge Representation & Ontologies of Biological Information
  - Gene Ontology, BioCyc, Pathway databases, …

Projects Overview

- 5. (P) Scientific Workflows/ Kepler
  - (5A) Analysis-intensive workflows (focus on data analysis, some data transformations, some visualization)
  - (5B) Data-intensive workflows / SDSC Storage Resource Broker (collection management on the "Data Grid")
  - (5C) Compute-intensive workflows / Condor, NIMROD, APST, … (job scheduling on cluster computers, simulations, "Compute Grid")
  - (5D) Database-intensive workflows (TBD)
  - (5E) Real-time data access workflows / ROADNet (accessing real-time data streams, some analysis and visualization)

Data Integration (Mediator System)





Global-as-View (GAV) & View Unfolding

- In GAV data integration, rules have the form
  - R_G(…) ← … R_L(…) …
- The **user query** is against the global (=integrated) views (relations) R_G:
  - ?- R_G(…)
- The system must come up with a **query plan** that eliminates references to R_G relations, and only keeps R_L relations (and maybe new aux-relations)

➔ GAV query rewriting

- Essentially "view unfolding"

Global-as-View (GAV) & View Unfolding

- Simplistic example:
- User query: ?- ans(X)
- View definitions
  - ans(X) ← p(X,X), u(X,Y).
  - p(X,Y) ← q(X,Z), r(Z,Y)
- Algorithm:
  - Given a goal (rhs) G, unify it with a matching head (lhs) H of a rule H(ead) ← B(ody)
  - Replace G by B, taking into account the substitution σ needed for unifying G and H
  - Repeat until a all relations are given (=EDB) relations

Querying vs Reasoning

*Q*1: answer1(S,C) ←
  student(S, N), takes(S, C), course(C, X), inCS(C), course(C, "DB")

Q2: answer2(S,C) ←
  student(S, N), takes(S, C), course(C, X)

- We can "run" both queries *Q* on a given database instance *D* ➔ **Query evaluation** (answer = eval(*Q*,*D*))
- Note: The answers to *Q1* are always *contained* in the answers to *Q2* (why?)
- Determining whether a query *Q* is contained (for all database instances D!) in another query Q' is a **reasoning** problem.