Two Alternative Views of Data Semantics
+
Application to Data Mapping/Exchange

Alex Borgida
Rutgers University

Joint work with ***John Mylopoulos***
and others at University of Toronto

Data Semantics

- **Data semantics = establishing and maintaining some relationship between an *information source* (viewed as a model), and its *intended subject matter.***
- **Data semantics essential in database design, data integration and data exchange**
- **We'll advance a couple of proposals for data semantics that are more complex than the standard ones that are based on conceptual modeling (in ER, UML, OWL) of the domain**
- **Look at some consequence for data mapping & exchange**

I. Modeling & the Mapping Continuum
[BoMy SemDb04]

- ***Two intriguing suggestions for a more careful look at the notion of modeling***
  - Ladkin's view of modeling *for a purpose*
  - B.C.Smith's notion of chains of models

What is a "Model" in general?

"***M is a model of subject S for purpose P***" [Ladkin97]

- Often, the purpose is *"answering certain kinds of questions* (about the subject)"
- Why do we build models? Because it is easier to answer those questions about the model than about the subject.

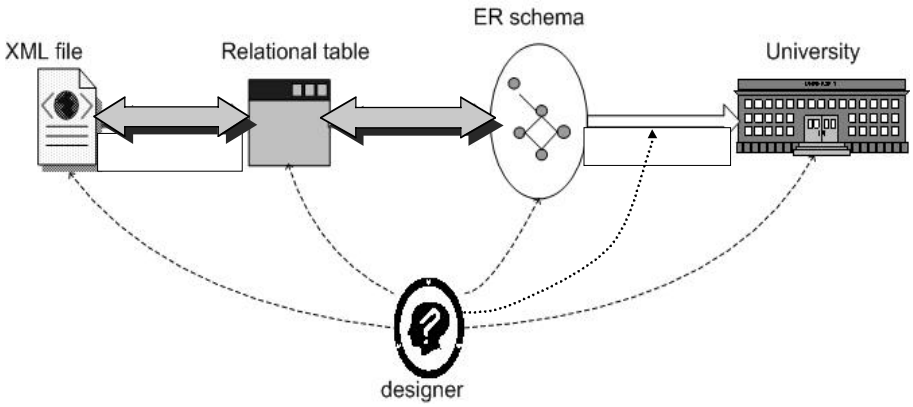
What is a Model? (cont'd)

- **So for every modeling situation, we need**
  - methods for building/changing the model
  - asking and answering questions in the model
  - a way to translate applicable questions about the subject matter into questions about the model ("$mapping_1$")
  - conversely, a way to translate results of the query on the model to answers about the subject ("$mapping_2$")
- **Application to Information Systems:**
  - Example: a classroom & scheduling database as a model of a part of the (actual) university

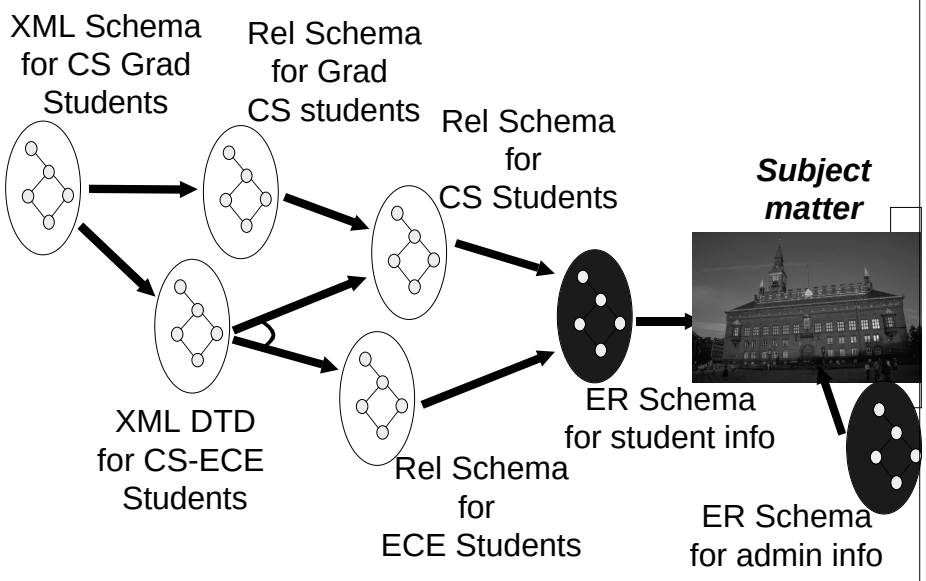
The Correspondence Continuum

- **Meaning is rarely a simple mapping from symbol to 'object'; instead, it often involves a** *continuum of (semantic) correspondences* **from symbol to (symbol to)* object [Brian Cantwell Smith'87]**

Application to Data Semantics: Lineal Semantic Mappings



More generally...



Data Semantics - take 1

Every (non-root) model comes with an explicit semantic mapping to some other model(s)

- **The complete meaning of data in a model includes the** composition **of the semantic mappings relating it to roots.**
- **Of course, keeping around the semantic mapping will be expensive; but the alternative is the mess of legacy data!**

Lot's of Related Work

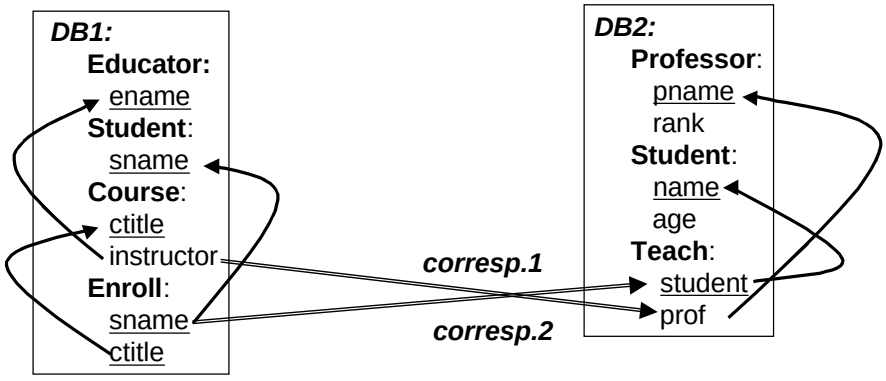
- **Data Integration**
- **Ontology Integration**
- **Model and mapping management**
- **Peer data management**
- **Data provenance**

Where do we get Mappings?

- **The *mapping* could/should be saved during design.**
- **Other mappings must be *discovered* and specified.**
- **Tools may be needed to do the later:**
  - **Some use heuristics based on structure and naming**
  - **Others, including Clio [Miller+00] [Popa+02] discover complex mappings between two schemas, given a set of simple "correspondences" between their elements.**

Clio: Discovering Schema Mappings

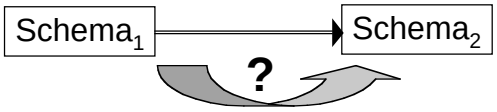
INPUT:



OUTPUT:

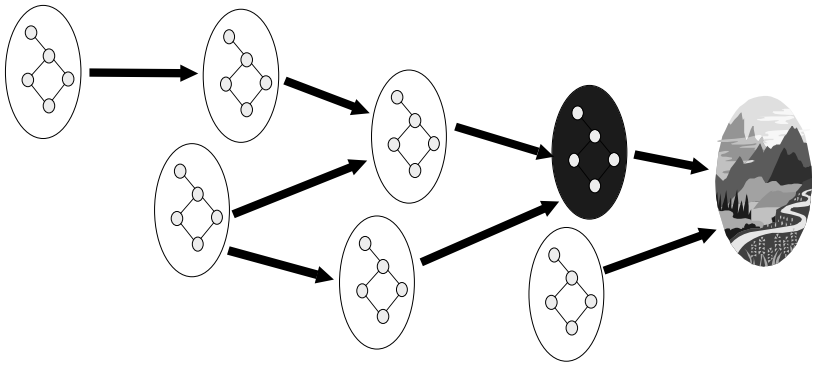
```
DB2: teach(S,P) :-
       DB1: course(C, P),
       DB1: enroll(C, S)
```

The Clio Approach



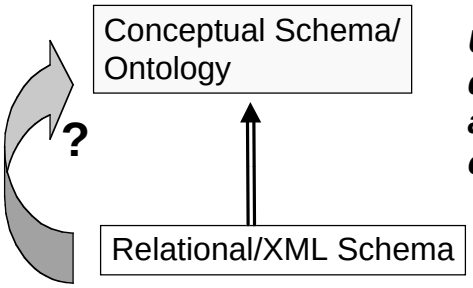
- ***Clio's thesis:*** **it is easier for users to specify correspondences, and have tools support discovery of the actual, more complex, mapping.**
- **Clio relies on co-occurrence of attributes and foreign key constraints to discover *heuristically* "reasonable" mappings. ("*grow islands using lossless joins*".**
- **Extended to deal with mapping XML schemas**

Applying the idea to the data semantic continuum



- **Currently, Clio can discover mappings between relational and/or XML models**
- **What if we add conceptual models/ontologies?** (*Joint work with Yuan An and Renée Miller)*

Situation 1

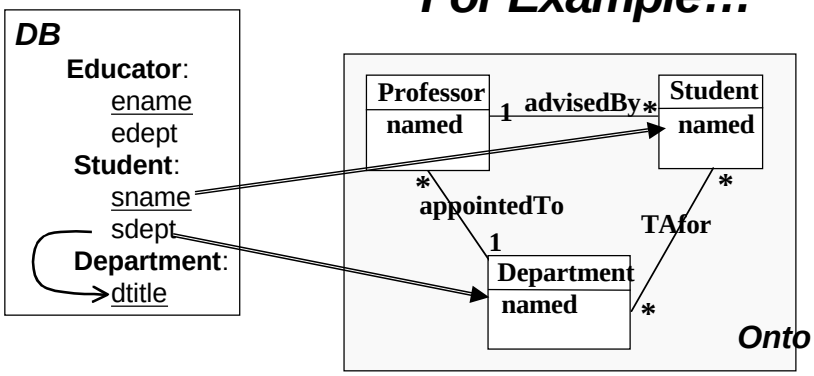


Useful for data integration and service composition

- **Our prototype tool exploits richer semantics in the conceptual schema ("object", association cardinality, IsA hierarchy, disjointness,...)**
- **Also, uses theory of relational schema design from ER diagrams as basis of heuristics**

For Example…

INPUT:

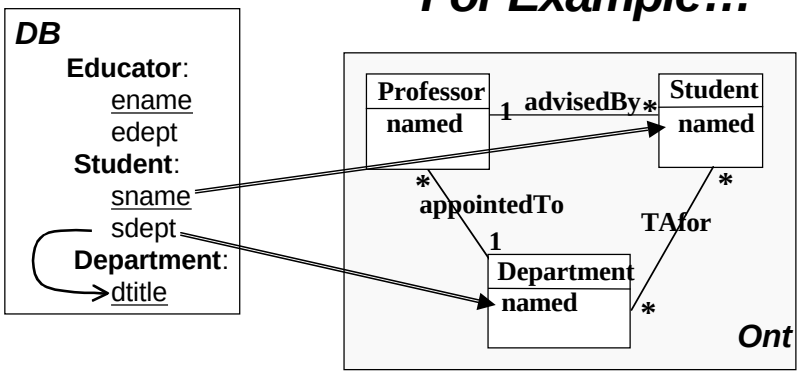


OUTPUT:

```
DB:Student(sname,sdept) →
      O:Student(x), O:Professor(y), O:Department(z),
      named(x,sname), named(y,ename), named(z,dtitle),
      advisedBy(x,y), appointedTo(y,z).
```

For Example…

INPUT:



OUTPUT:

```
DB:Student(sname,sdept) →
      O:Student(x), O:Professor(y), O:Department(z),
      named(x,sname), named(y,ename), named(z,dtitle),
      advisedBy(x,y), appointedTo(y,z).
```

Not output: shortest path!

```
DB:Student(sname,sdept) → Student(x), Department(z),
named(x,sname),named(z,dtitle),TAfor(x,z).
```


General Strategy

1. **Mark concept nodes having attribute(s) corresponding to table T's column(s).**
2. **Create a "skeleton tree" of marked nodes which have correspondences to the *key columns* of table T, by finding cheapest spanning tree of the "appropriate kind"**
3. **Connect other marked nodes to the skeleton by shortest paths that are "function-like" (cardinality upper bound 1) - (**Because key uniquely determines these values in the table**)**
4. **Translate resulting tree into Datalog formula by recursive descent. (Can also be translated to OWL!)**

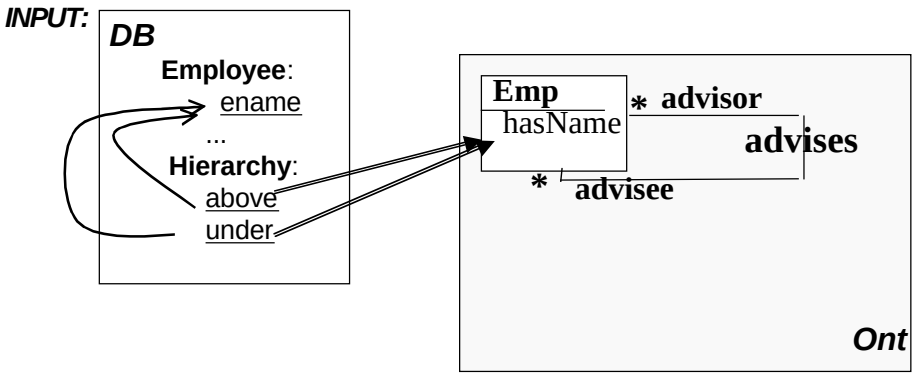
(Partial) Analysis of Relational Table Types (for UML model recovery)

Notation:

- *underscored italic* columns form *table key*;
- red columns are foreign keys

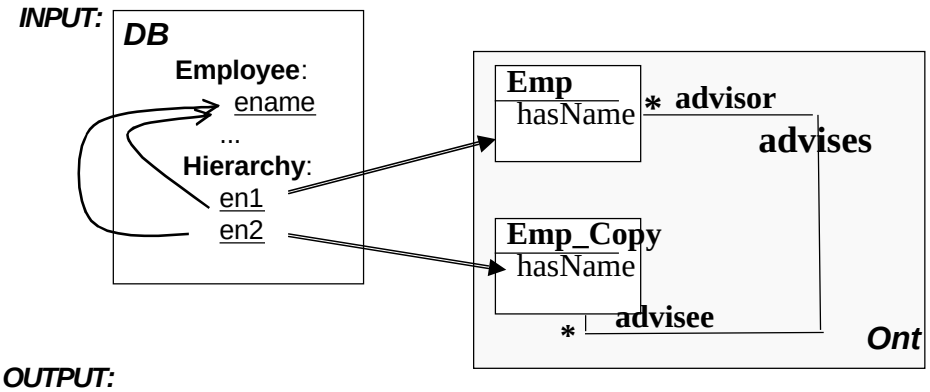
- **T(*b*,h...) -- an entity's table; 'skeleton' is b's node**
- **T(*b*,h,...) -- an entity table, merged with a functional (N-1, 1-1) relationship; 'skeleton' is b's node**
- **T(*b*,h,...) -- subclass; 'skeleton' is b's node**
- **T(*b,c*,h,...) -- 'weak entity' table; skeleton is c's node**
- **T(*b,c*,h,...) -- N-M relationship table; skeleton is a shortest *non-functional path* between b's & c's node**

Interesting Complication



Reflexive relationship!

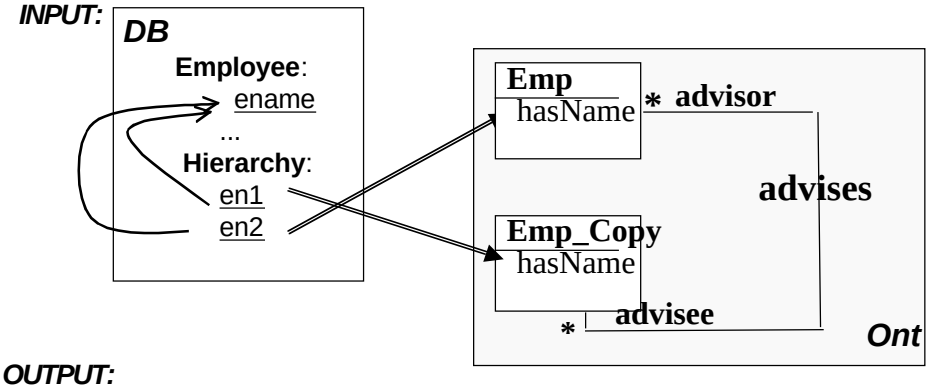
Solution: duplicate node



OUTPUT:

```
DB:Hierarchy(en1,en2) →
      O:Emp(x), hasName(x,en1),
      O:Emp(y), hasName(y,en2),
      advises(x,y)
```


Yet another solution!



OUTPUT:

```
DB:Hierarchy(en1,en2) →
      O:Emp(x), hasName(x,en2),
      O:Emp(y), hasName(y,en1),
      advises(x,y).
```

Somewhat reassuring

If we treat EER diagrams as notation for logic (entity classes = unary predicates; relationships and attributes = binary predicates) then the standard relational schema design rules found in textbooks assign a specific formula to each table.

- ***Theorem* 1 ("completeness") For any specific table T1 derived from E1, one of the formulas returned by our algorithm is the one assigned by the above design rules.**
- ***Theorem 2 ("soundness")* If the algorithm returns a semantic formula, there was a way of deriving that table from E1 with that semantics.**

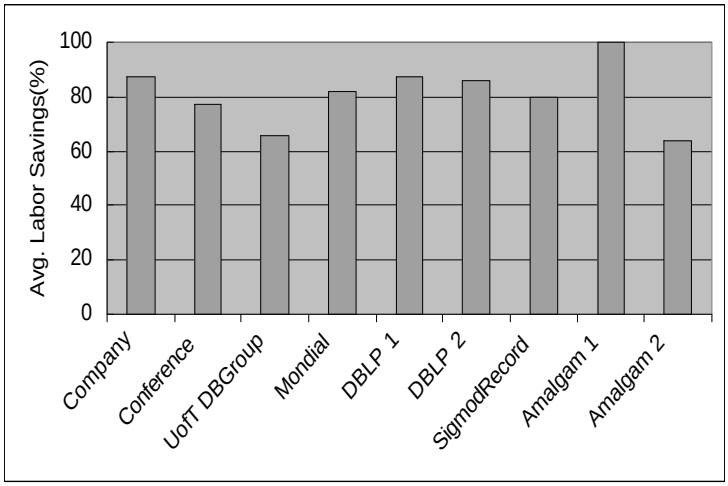
The algorithm works in other cases too, but of course not in all 'denormalized' schemas.

Experience

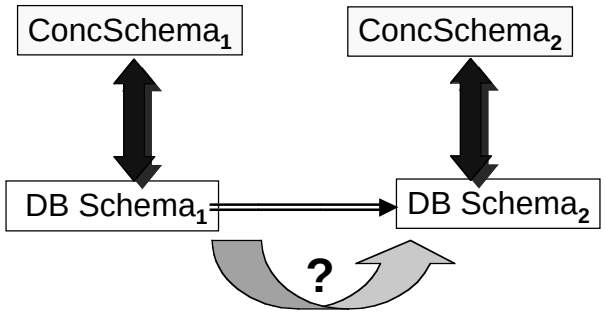
- **Implemented tool: *MapOnto***
- **To measure the usefulness of the tool, we want to know how much can a user benefit from its use.**
- **Suppose that *MapOnto* returns formula *f* instead of the desired correct formula *g*. If there are**
  - **n** atoms in the returned formula f;
  - **m** atoms in the correct formula g;
  - **c** atoms in common to f and g
- **Then tool user needs to delete n-c atoms from *f* and add m-c atoms to *g*.**

Labor savings=1- ((n-c)+(m-c))/m

Experimental Results: labor savings for XML Schema semantics



Situation 2



- The original problem for Clio only used db schemas and correspondences as inputs to seek mappings.
  - Suppose we also were given semantic mappings for DB schemas.
  - Can we help Clio produce better results?

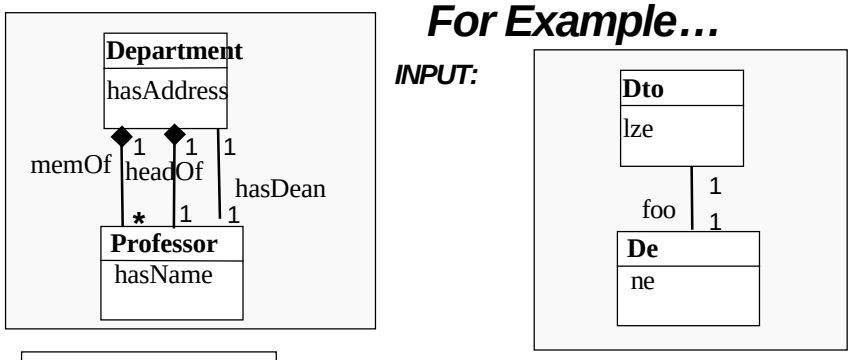
For Example…

INPUT:


```
{∃ d#,p# . d2_T(d#,Addr),p2_T(p#,Name),foo(d#,p#) }
       :- d1_T(did,Addr),p1_T(pid,Name), ???(did,pid).
```

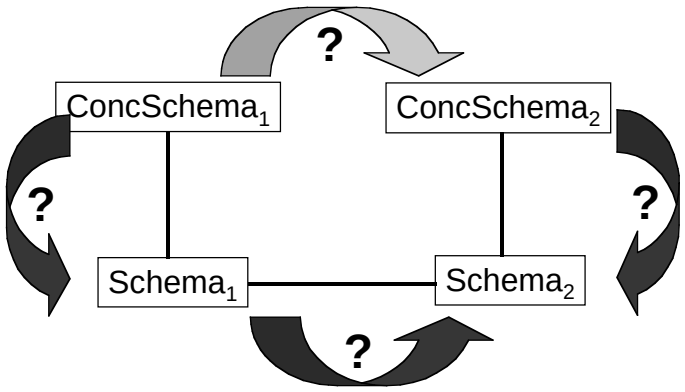

For Example…

INPUT:



- ***foo*** **is semantically more like** ***hasDean*** **than** ***headOf*** **or** ***memOf*** **(see** ***cardinality***, ***partOf***)

Extensions: ontology mapping?



II. An Intentional Dimension for Data Semantics

- **Traditionally, data semantics deals with "*what/when*" questions: *objects, inter-relationships, groupings* and *constraints* on them.**
- **But to achieve real understanding, humans also rely on "*how*" and especially "*why*" questions: *How is the data used? Why was the data gathered?***
- **Recent progress in *Requirements Engineering* ("GORE") has shown how *goals* and (*organizational) actors* fit in.**

Example...

- Consider the relational schema

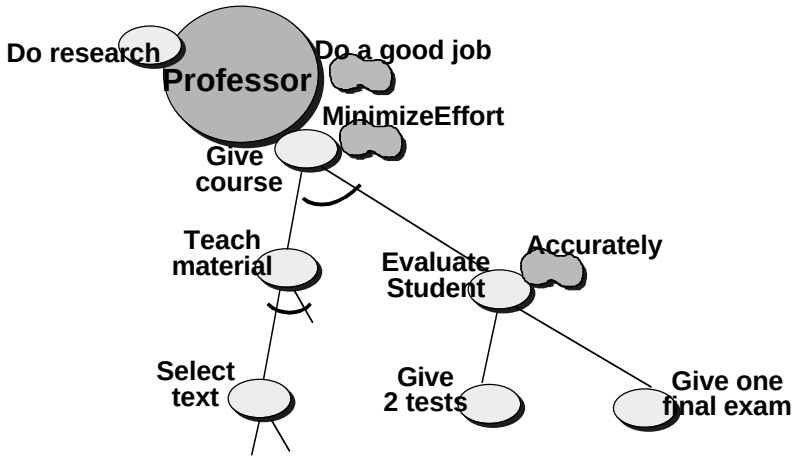
  Grades(<u>crs#, st#,</u>test1,test2,finalGrade)

- Part of the data semantics ought to be what are grade scales *(%? letter?),* how is final grade computed *(weighted sum and scale?).*
- Richer semantics would also describe the workflow of TAs and professors which results in these data values; and the *motivation* for choosing this (“*multiple tests provide better evaluation”)*

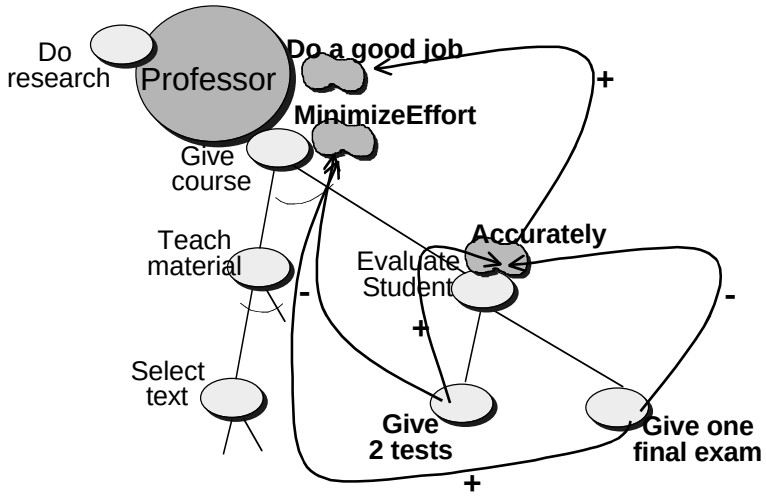
This has been made precise: i, Tropos* [Mylopoulos, Yu,...]

- (Institutional) actors
- Goals, and their analysis (decomposition, means/ends)
- Softgoals (e.g., non-functional requirements such as *accuracy, security, cost*) and how they motivate choices between alternatives

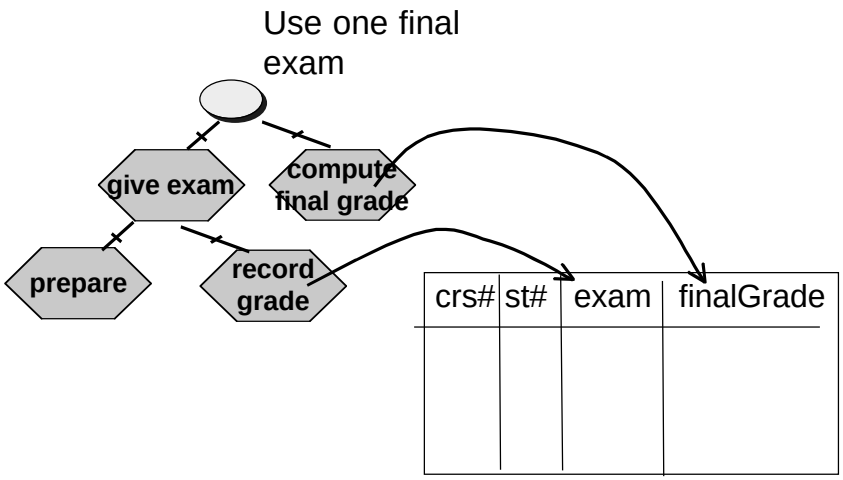
Actors, Goals and Softgoals



Analysis of alternatives



Task Analysis and Database Design



GORE *Database Design Methodology*

(joint work with Lei Jiang and T.Topaloglou)

1. Find stakeholders, their goals & softgoals
2. Analyze goals and softgoals; find (alternative) ways of fulfilling each goal; evaluate these with respect to softgoals; for some alternatives, define tasks to achieve them
3. Identify (concerned) objects for goals and tasks; develop a **domain terminology** describing understanding (using ontologies and goals)
4. Identify query, persistence,... requirements
5. Design **database conceptual schema** from 3 and 4
6. Alternatives are chosen and justified by reference to goals or softgoals.

Data semantics - take 2

- Keep this information
- Also keep history of decisions (for evolution)

These are part of data semantics! (Just like data provenance.)

- There are precise, formal notations for recording and reasoning with these (formal Tropos, "satisficing")

Potential application to data exchange

- Suppose there are 2 databases for departments which decided to use different grading schemes
  - DB1: grades1(crs#, st#, test1, test2, finalGrade)
  - DB2: grades2(crs#, st#, exam, finalGrade)
- If loading data from DB1 to DB2

```
grades2(f(crs#),g(st#),_,_,FG)
          :- grades1(crs#,st#,_,FG)
```

  we can deduce that *accuracy in* finalGrade *is likely gained*, while loading data in the opposite direction is not! Might help decide where to load data.

- (Scientific database evolution & loading)

Related Ideas

- **Hippocratic Databases [Agrawal02]**
- **Why questions in data provenance [Buneman]**
- **Data semantics in systems involving workflows and processes.**

Discussion

- **Data semantics is and will remain a core problem for databases**
- **Ultimately, the meaning of data needs to be tied down to the intentions of its designers and users.**
- **This (should) affect data mapping and exchange**

References

- [Ladkin97] Ladkin, P.: "*Abstraction and Modeling*", Research Report RVS-Occ-97-04, University of Bielefeld, 1997
- [BoMy94] Borgida, A. and J. Mylopoulos: "*Data Semantics Revisited*", SWDB 2004
- [Miller00] Miller, R., Haas, L., Hernadez, M.: "*Schema Mapping as Query Discovery*", VLDB 2000
- [Popa02] Popa, L., Velegrakis, Y., Miller, R., Hernandez, M., and R. Fagin: "*Translating Web Data*.", VLDB 2002
- An, Y., Borgida, A. and J. Mylopoulos, "*Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences*", ODBASE 2005 /JoDS 06/ ICDE07)
- Jiang, L., Borgida, A., Topaloglou, T., and J. Mylopopulos "*Goal-driven database design",* ReqEng 2006, ReqEng 07,