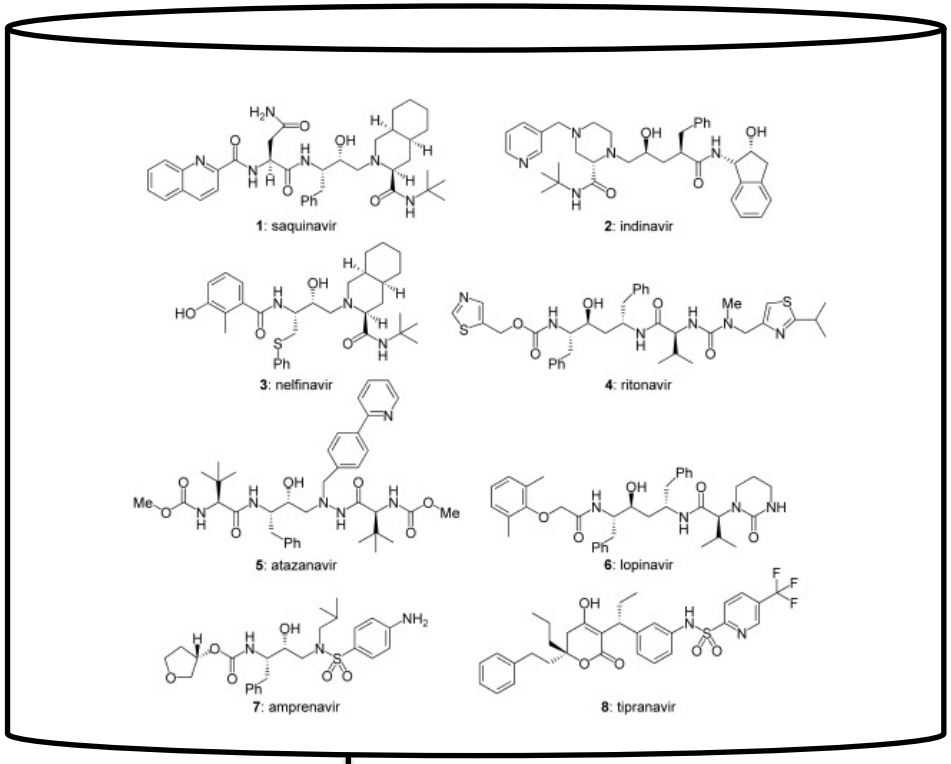
Towards Automatically Setting Language Bias in Relational Learning

Jose Picado, Arash Termehchy, Alan Fern, Sudhanshu Pathak

Information and Data Management and Analytics (IDEA) Lab

Oregon State University

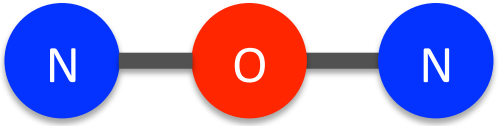
Design a drug to treat HIV



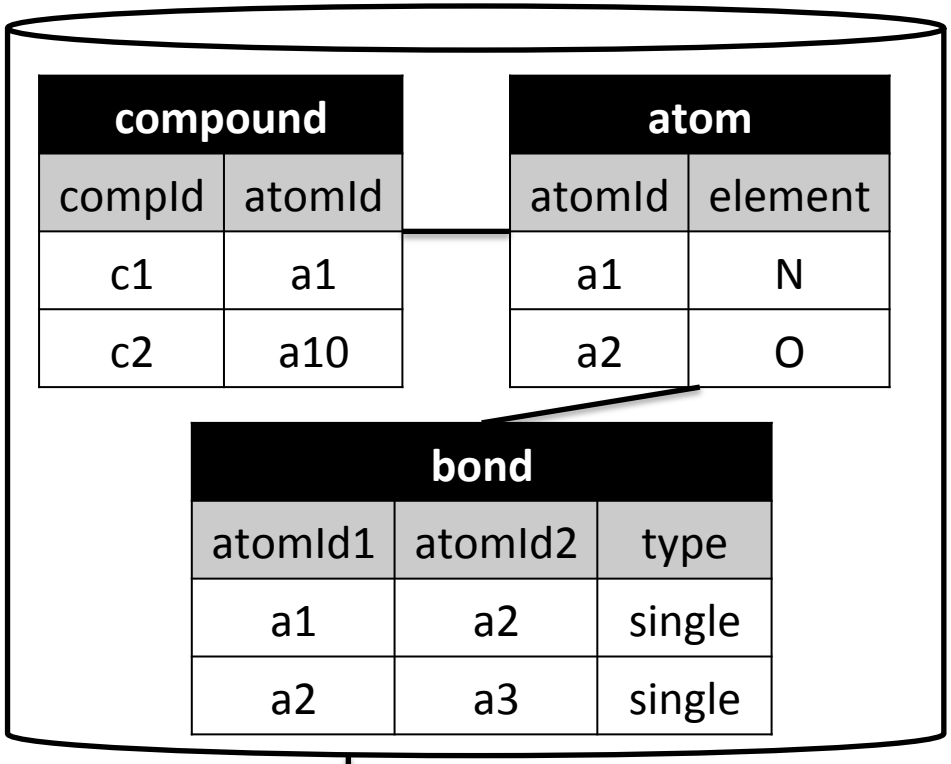
What is the structure of compounds that have **anti-HIV** activity?

Oracle

A compound has **anti-HIV** activity if it has the following substructure:



Relational learning can learn definition for anti-HIV



compound	
compId	atomId
c1	a1
c2	a10

atom	
atomId	element
a1	N
a2	O

bond		
atomId1	atomId2	type
a1	a2	single
a2	a3	single

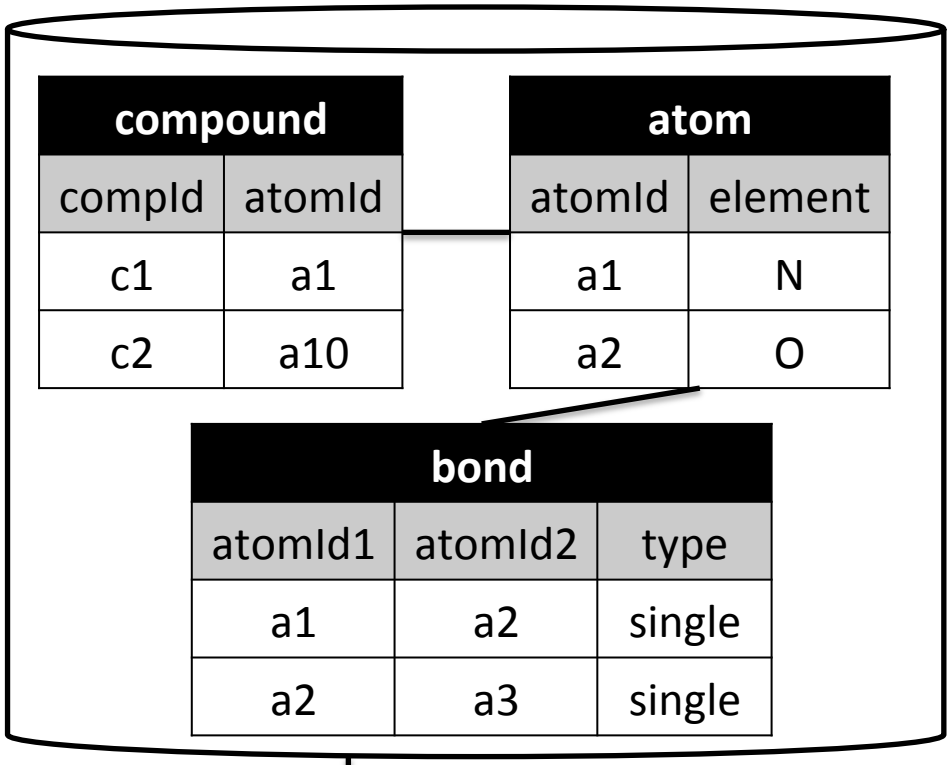
Training data:

anti-HIV	no-anti-HIV
compId	compId
c1	c2
c3	c4

Relational learning algorithm

anti-HIV(x) :- compound(x,u), atom(u,N),
compound(x,v), atom(v,O),
compound(x,w), atom(w,N),
bond(u,v,single), bond(v,w,single).

Benefits of relational learning



compound

compId	atomId
c1	a1
c2	a10

atom

atomId	element
a1	N
a2	O

bond

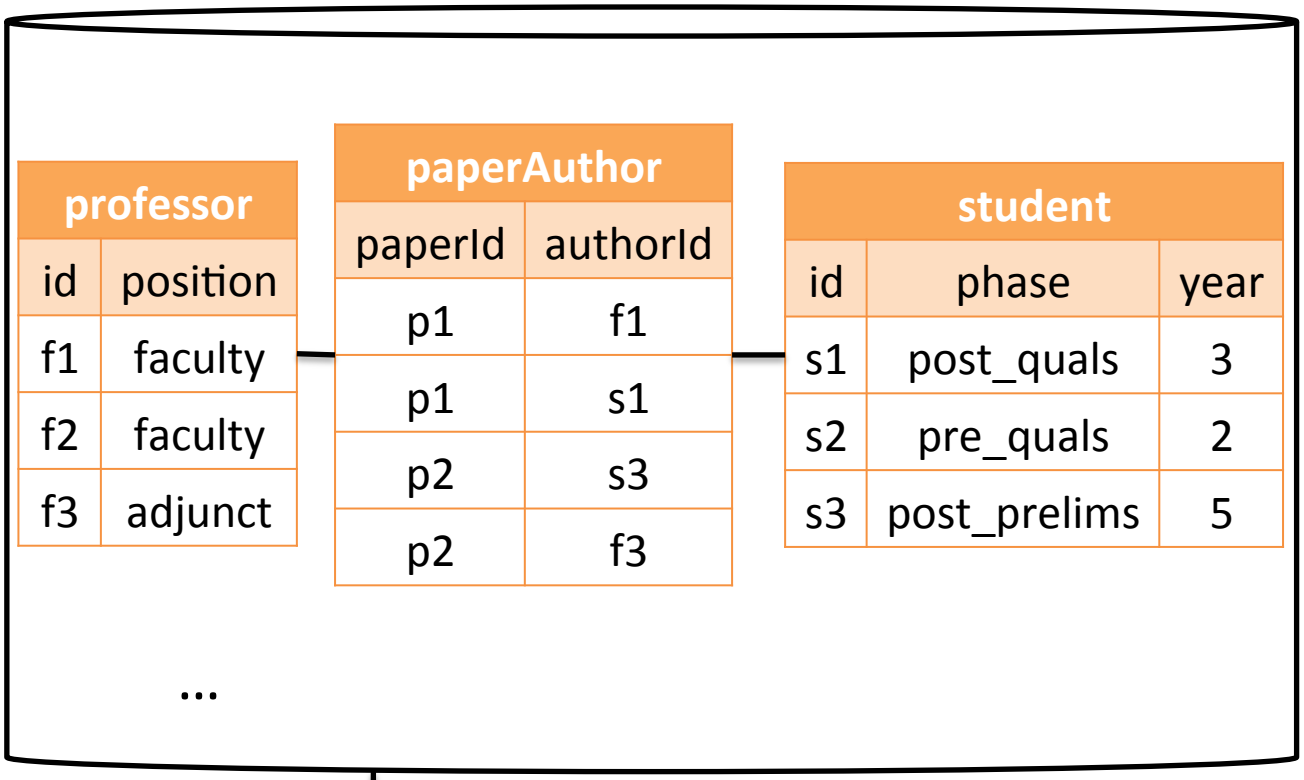
atomId1	atomId2	type
a1	a2	single
a2	a3	single

- ✓ Leverage the structure of data and learn over complex schemas with multiple tables
- ✓ Automatic feature extraction and selection
- ✓ Results are interpretable (Datalog)

Relational learning algorithm

```
anti-HIV(x) :-  compound(x,u), atom(u,N),
                compound(x,v), atom(v,O),
                compound(x,w), atom(w,N),
                bond(u,v,single), bond(v,w,single).
```

How relational learning works



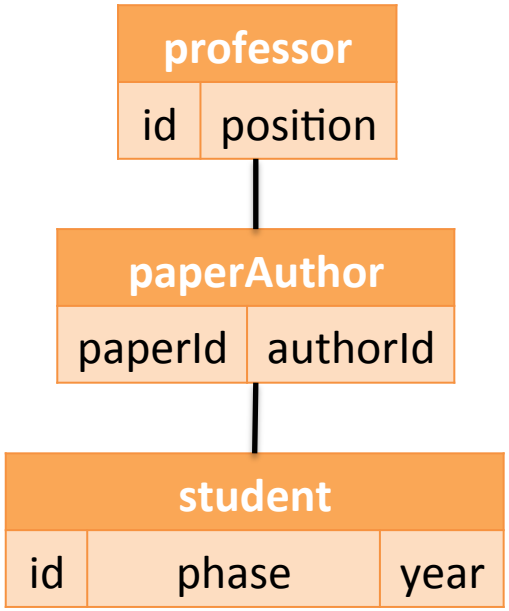
What is the definition of the **advisedBy** relation?

advisedBy	
studId	profId
s1	f1
s3	f3

not-advisedBy	
studId	profId
s2	f3
s1	f3

Relational learning algorithm → ?

Generic relational learning algorithm

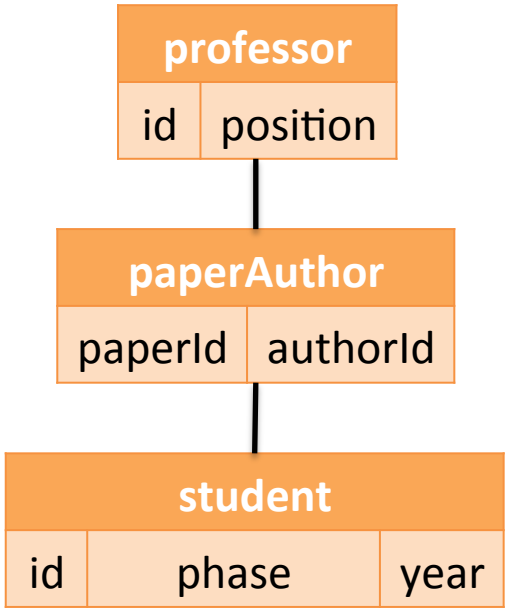


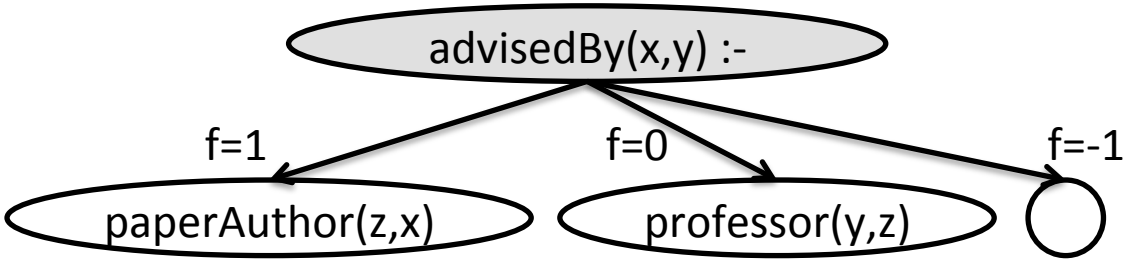
advisedBy(x,y) :-

Scoring function **f**: P - N
P: positive examples covered
N: negative examples covered

```
advisedBy(x,y) :-
        true.
```

Generic relational learning algorithm

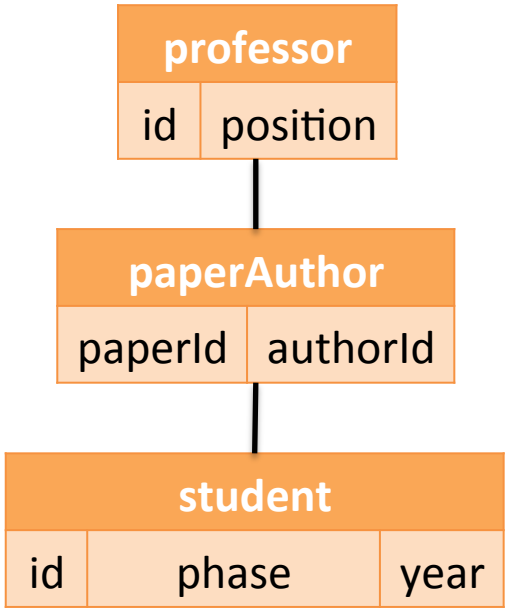


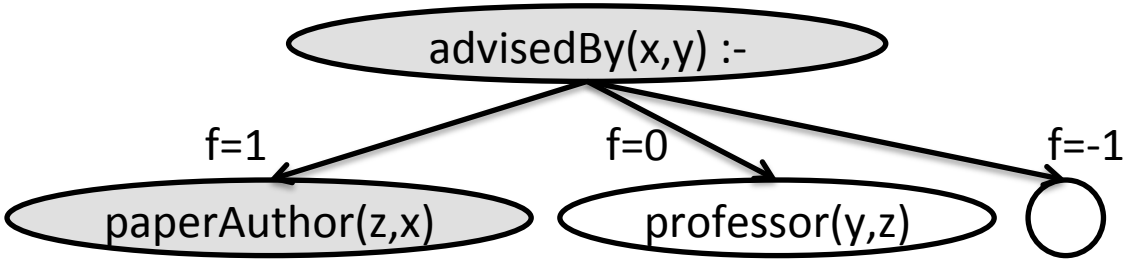


Scoring function **f**: P - N
P: positive examples covered
N: negative examples covered

```
advisedBy(x,y) :-
        true.
```

Generic relational learning algorithm

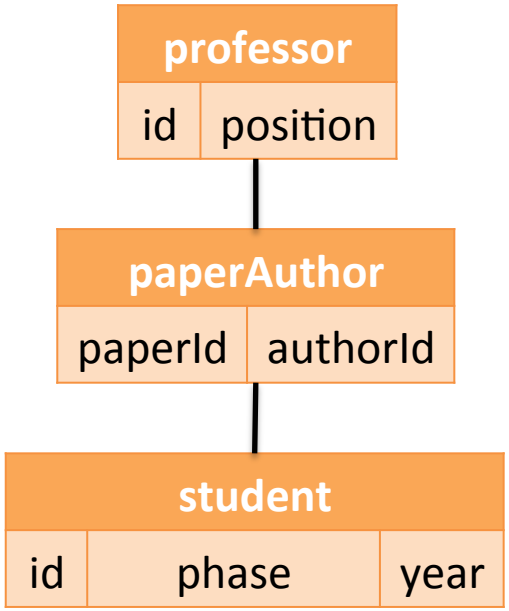


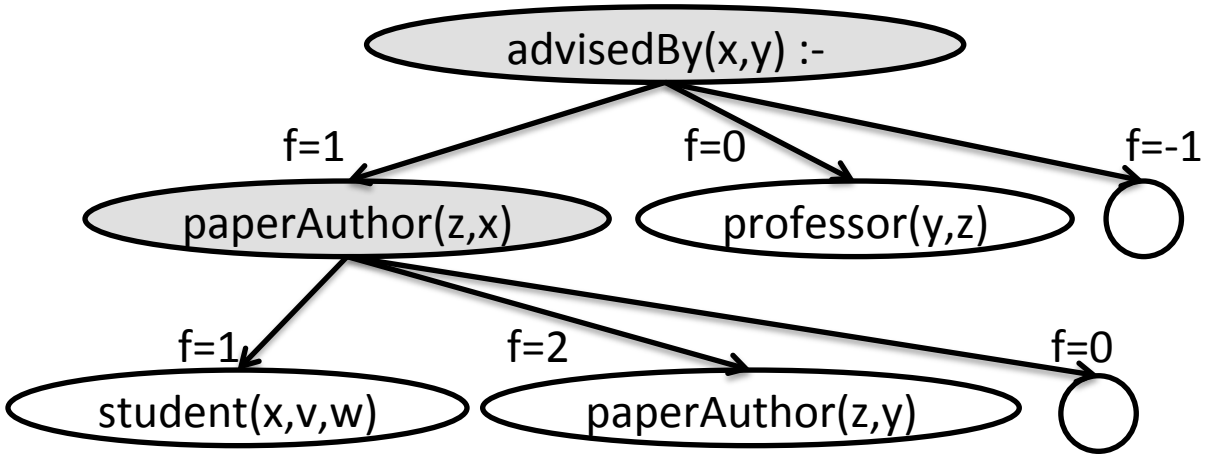


Scoring function **f**: P - N
P: positive examples covered
N: negative examples covered

```
advisedBy(x,y) :-
        paperAuthor(z,x).
```

Generic relational learning algorithm

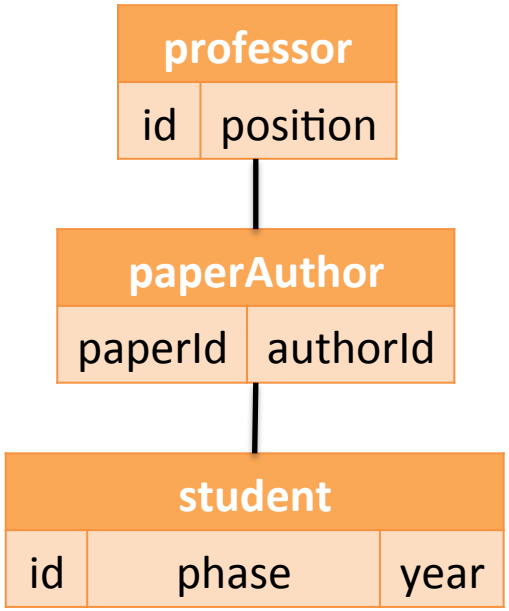


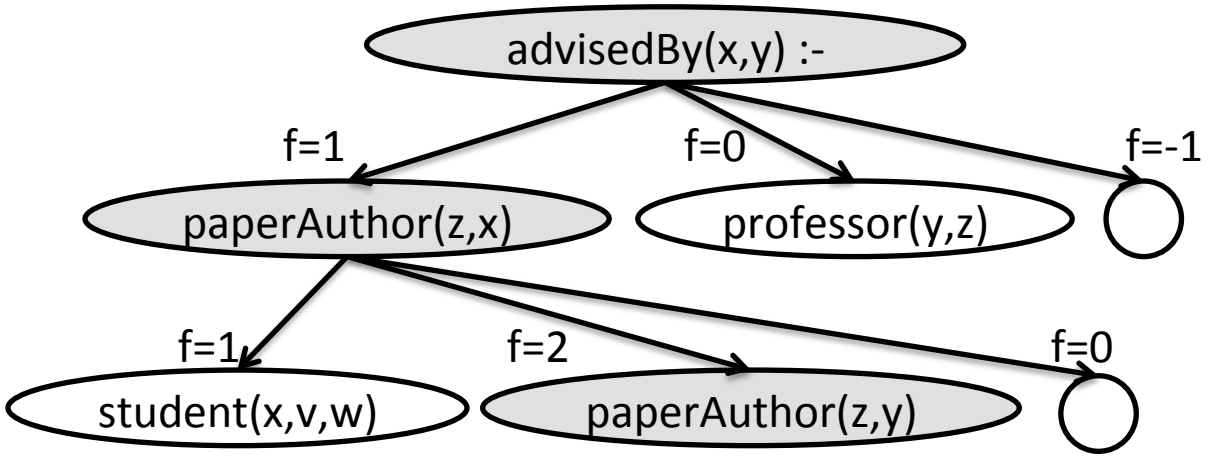


Scoring function **f**: P - N
P: positive examples covered
N: negative examples covered

```
advisedBy(x,y) :-
        paperAuthor(z,x).
```

Generic relational learning algorithm

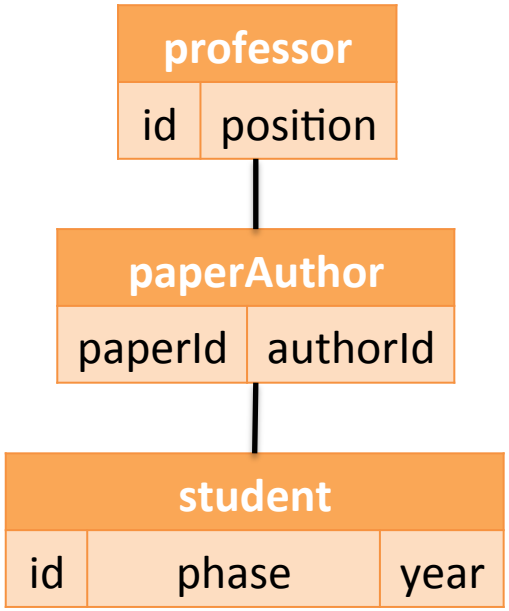


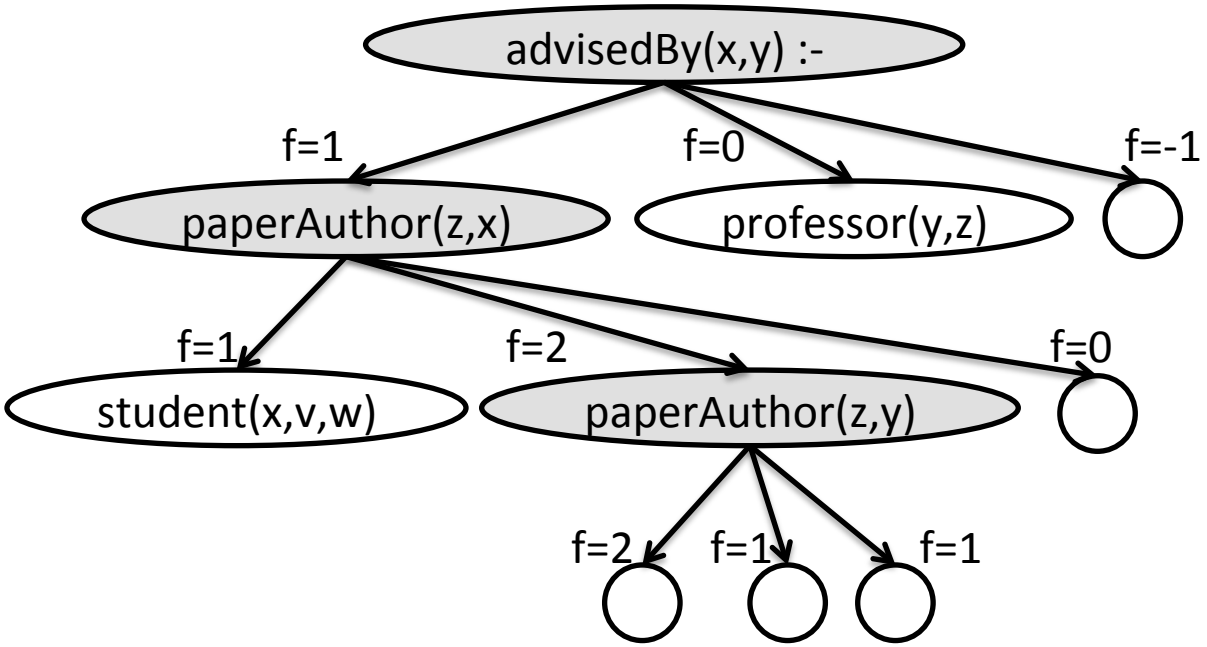


Scoring function **f**: P - N
P: positive examples covered
N: negative examples covered

```
advisedBy(x,y) :-
        paperAuthor(z,x), paperAuthor(z,y).
```

Generic relational learning algorithm



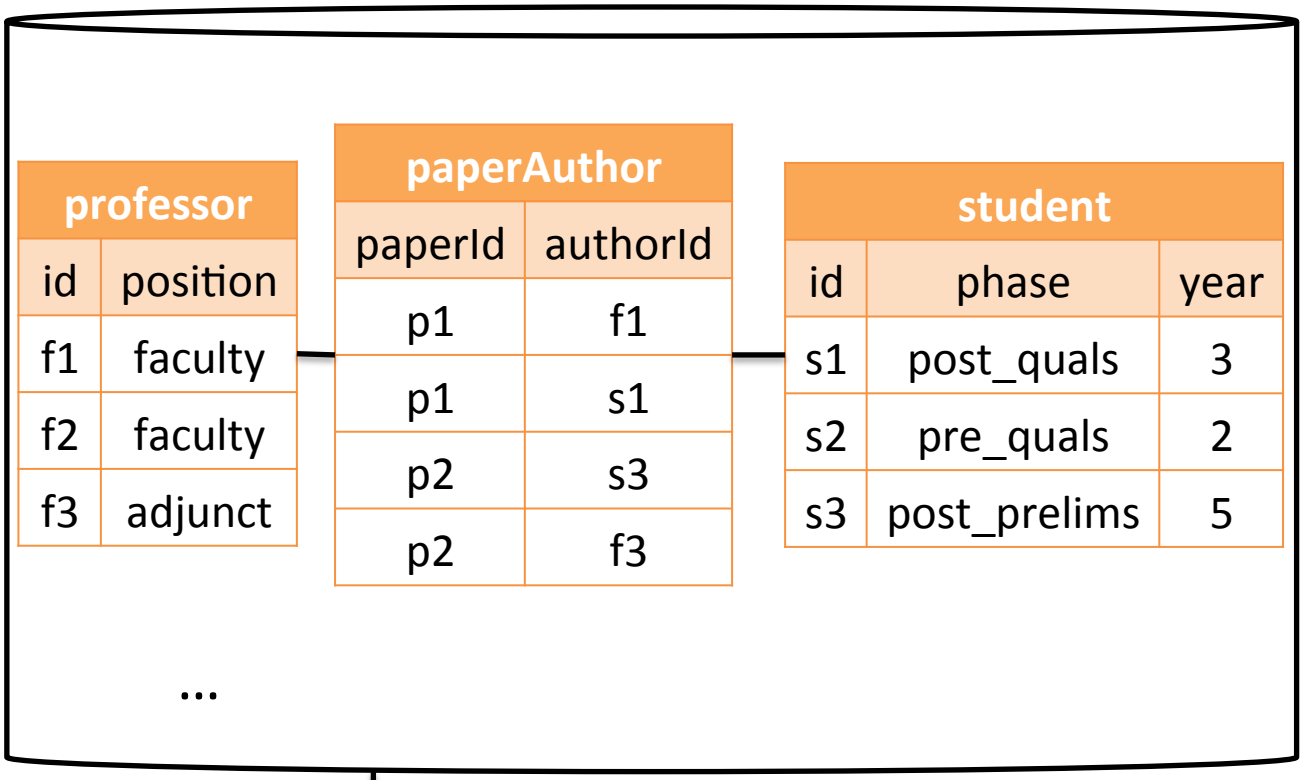


No improvement

Scoring function **f**: P - N
P: positive examples covered
N: negative examples covered

```
advisedBy(x,y) :-
        paperAuthor(z,x), paperAuthor(z,y).
```

Learned definition



What is the definition of the **advisedBy** relation?

advisedBy	
studId	profId
s1	f1
s3	f3

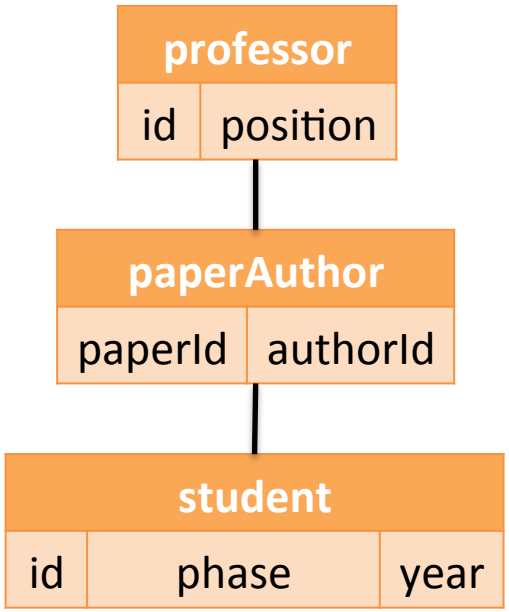
not-advisedBy	
studId	profId
s2	f3
s1	f3

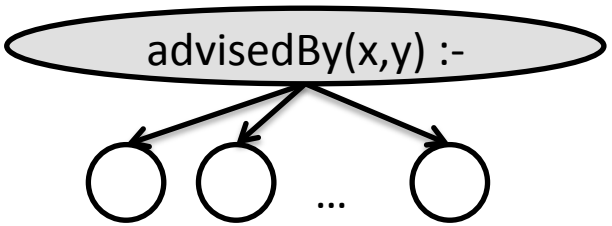
Relational learning algorithm

```
advisedBy(x,y) :-
    paperAuthor(z,x), paperAuthor(z,y).
```

Hypothesis space in relational learning algorithms is huge

- Hypothesis space: all Datalog definitions containing relations in the schema
- Current solution: users must set language bias to restrict the hypothesis space

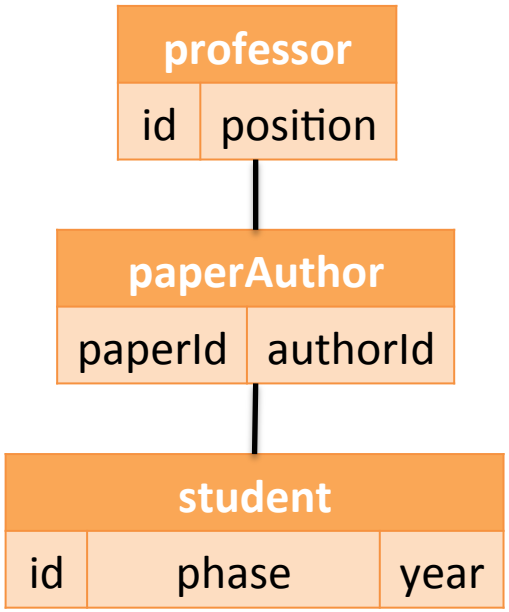




paperAuthor(x,x)	professor(x,z)
paperAuthor(z,x)	professor(x,y)
paperAuthor(z,y)	student(x,v,w)
paperAuthor(x,y)	student(x,y,z)
paperAuthor(z,v)	...

Syntactic bias restricts the structure of learned Datalog definitions

- Which relations to query?
- Which relations to join and over which attributes?
- Should an attribute be a constant or a variable?



join paperId with professor id?

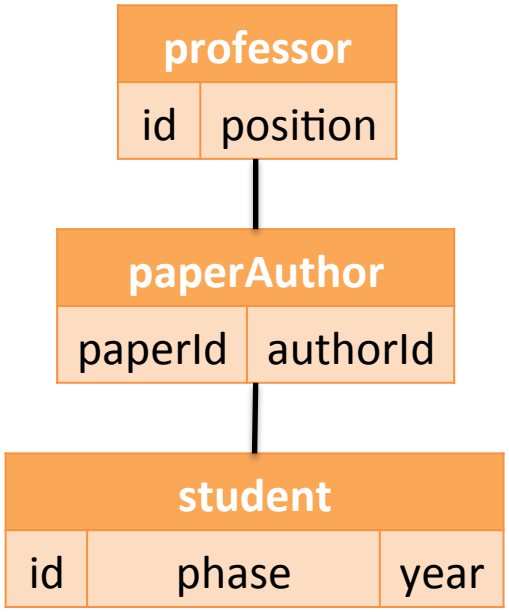
```
advisedBy(x,y) :-
      paperAuthor(z,x), professor(z,v).

advisedBy(x,y) :-
      professor(y,z), professor(y,faculty).
```

variable

constant

Predicate definitions

- Assign types to each attribute in every relation
- Only attributes with same type can join



attribute	type
professor[id]	professor
professor[position]	position
paperAuthor[paperId]	paper
paperAuthor[authorId]	student
paperAuthor[authorId]	professor
student[id]	student
…	

Predicate definitions

- Assign types to each attribute in every relation
- Only attributes with same type can join

attribute	type
professor[id]	professor
professor[position]	position
paperAuthor[paperId]	paper
paperAuthor[authorId]	student
paperAuthor[authorId]	professor
student[id]	student
…	

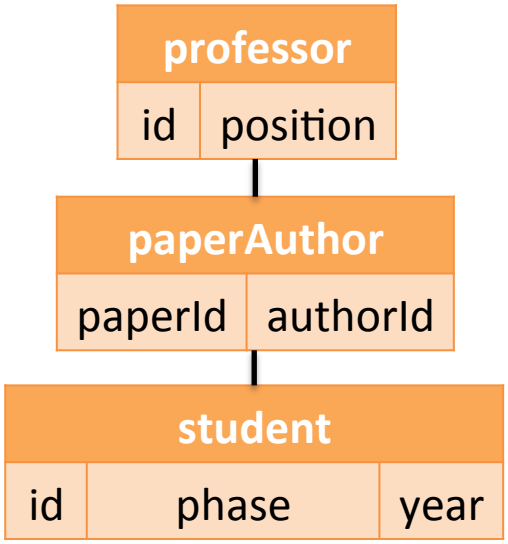
input to the algorithm

professor(professor,position)
paperAuthor(paper,student)
paperAuthor(paper,professor)
student(student,phase,year)
…

advisedBy(x,y) :-
paperAuthor(z,x), professor(z,v).

Mode definitions

- Define the mode to call relations and create literals
- Each attribute can be:
  - an existing variable (+)
  - an existing or new variable (-)
  - a constant (#)



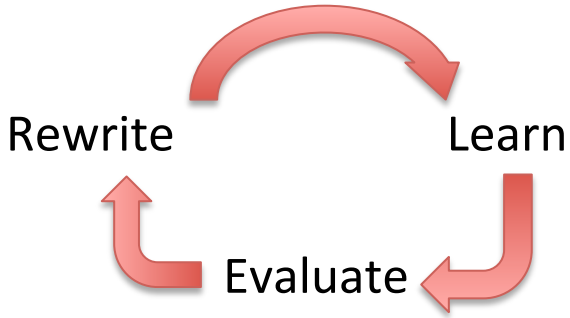
input to the algorithm

```
professor(+,-)
professor(-,+)
professor(+,#)
...
```

Predicate and mode definitions are the "black magic" of relational learning

- All relational learning algorithms require syntactic bias
- Manually written by the user

Requires expertise



Trial-and-error

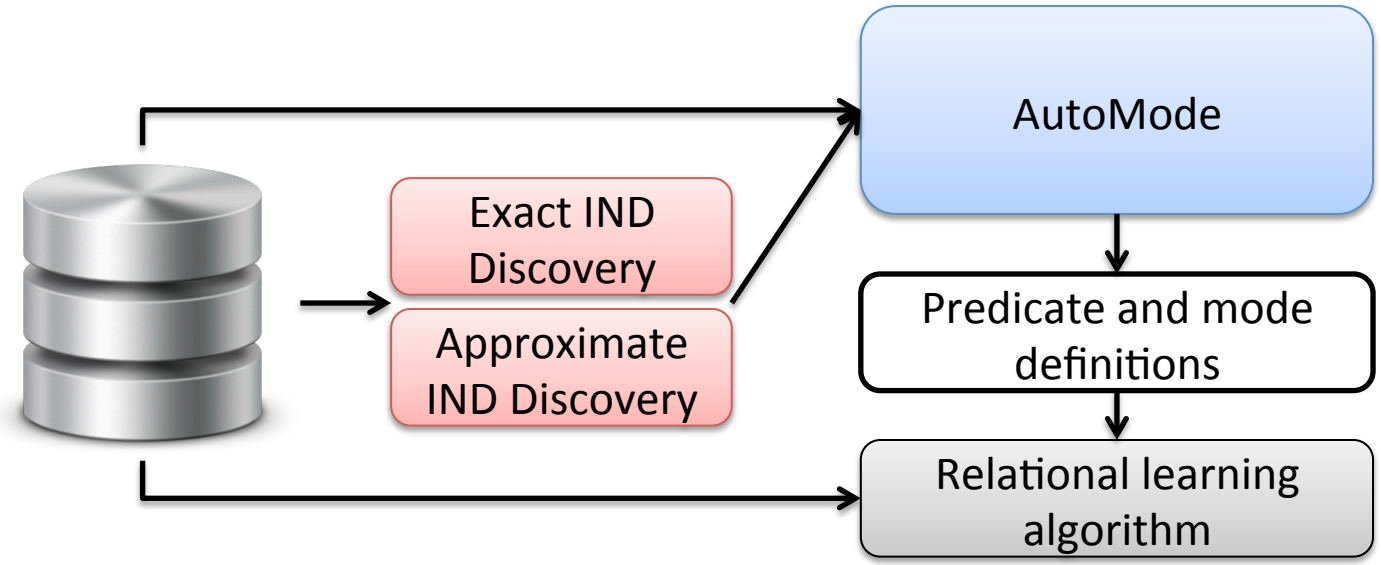
Difficult and time-consuming

Many lines of code to specify definitions

movies(+movieid,-title,-year)
movies2genres(+movieid,-genreid)
movies2prodcompanies(+movieid,-prodcompanyid)
movies2colors(+,movieid,-colorid)
movies2directors(+movieid,-director)
movies2directors(-movieid,+director)
movies2producers(+movieid,-producer)
movies2producers(-movieid,+producer)
producers(+producer,-name)
directors(+director,-name)
colorinfo(+colorid,-color)
colorinfo(+colorid,#color)
movies2writers(+movieid,-writer)
movies2writers(-movieid,+writer)
writers(+writer,-name)
movies2actors(+movieid,-actor,-character)
actors(+actor,-name,-sex)
actors(+actor,-name,#sex)
movies2cinematgrs(+movieid,-cinemat)
movies2cinematgrs(-movieid,+cinemat)
cinematgrs(+cinemat,-name)
movies2composers(+movieid,-composer)
movies2composers(-movieid,+composer)
composers(+composer,-name)
movies2costdes(+movieid,-costdes)
movies2costdes(-movieid,+costdes)
costdesigners(+costdes,-name)
movies2editors(+movieid,-editor)
movies2editors(-movieid,+editor)
editors(+editor,-name)
movies2misc(+movieid,-misc)
misc(+misc,-name)
movies2proddes(+movieid,-proddes)
movies2proddes(-movieid,+proddes)
proddesigners(+proddes,-name)
genres(+genreid,-genre)
genres(+genreid,#genre)
prodcompanies(+prodcompanyid,-prodcompany)
ratings(+movieid,-rank,-votes)
certificates(+movieid,-country,-certificate)
certificates(+movieid,#country,-certificate)
certificates(+movieid,-country,#certificate)
certificates(+movieid,#country,#certificate)
countries(+countryid,-country)
countries(+countryid,#country)
runningtimes(+movieid,-time)
runningtimes(+movieid,#time)
akatitles(+movieid,-languageid,-title)
akanames(+name,-name)
altversions(+movieid,-text)
business(+movieid,-text)
plots(+movieid,-text)
biographies(+bio,-name,-text)
distributors(+movieid,-name)
mpaaratings(+movieid,-text)
mpaaratings(+movieid,#text)
releasedates(+movieid,-countryid,-date)
releasedates(+movieid,-countryid,#date)
technical(+movieid,-text)
technical(+movieid,#text)
language(+languageid,-language)
language(+languageid,#language)
movies2languages(+movieid,-languageid)
movies2countries(+movieid,-countryid)

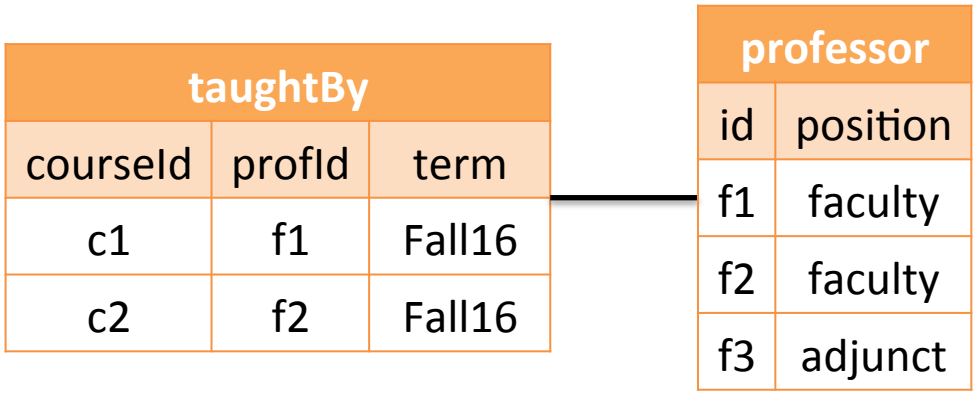
AutoMode: automatically induce syntactic bias

- Leverage information in the schema and content of the database



AutoMode: generate predicate definitions

- Use inclusion dependencies (referential integrity constraints) to find types of attributes
- Key idea: the most frequently used joins are the ones over the attributes that participate in an IND
  - E.g., primary-key to foreign-key relationship



taughtBy		
courseId	profId	term
c1	f1	Fall16
c2	f2	Fall16

professor	
id	position
f1	faculty
f2	faculty
f3	adjunct

$taughtBy[profId] \subseteq professor[id]$

AutoMode: generate predicate definitions

1. Get inclusion dependencies (INDs)
   - Read INDs from schema, if available
   - Discover exact INDs (Binder) and approximate INDs (AutoMode)

Exact IND Discovery

Approximate IND Discovery

taughtBy[profId] ⊆ professor[id]
ta[studId] ⊆ student[id]
paperAuthor[authorId] ⊆ professor[id]
paperAuthor[authorId] ⊆ student[id]
…

paperAuthor	
paperId	authorId
p1	f1
p1	s1

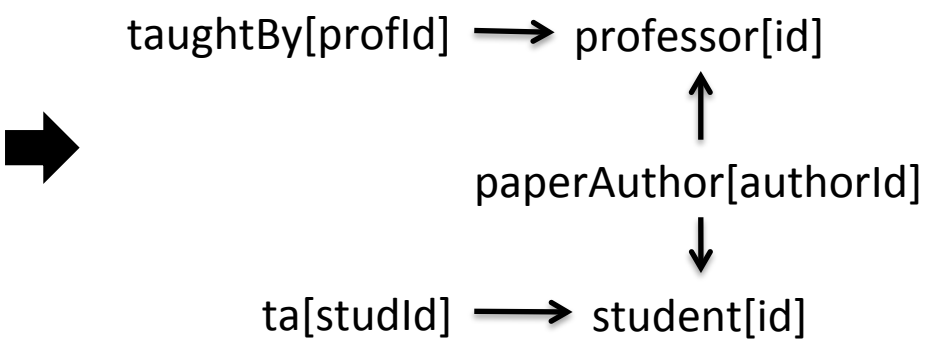
professor	
id	position
f1	faculty
f2	faculty

paperAuthor[authorId] ⊆ professor[id], α

AutoMode: generate predicate definitions

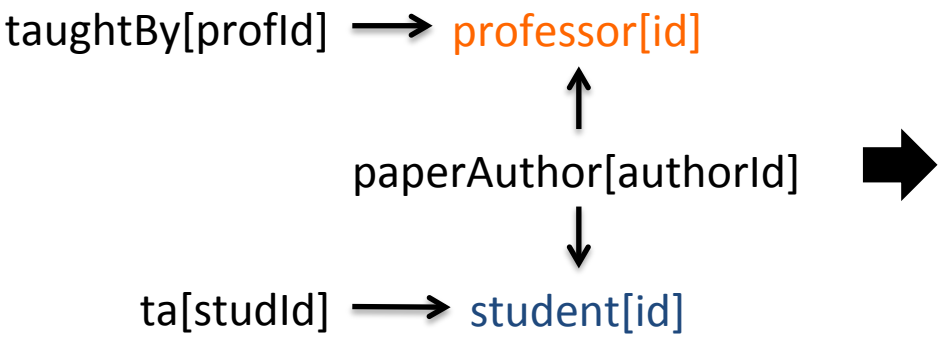
2. Generate a graph where:
   - nodes are attributes of the relations
   - there is an edge from A to B iff IND A ⊆ B exists

taughtBy[profId] ⊆ professor[id]
ta[studId] ⊆ student[id]
paperAuthor[authorId] ⊆ professor[id]
paperAuthor[authorId] ⊆ student[id]
…



AutoMode: generate predicate definitions

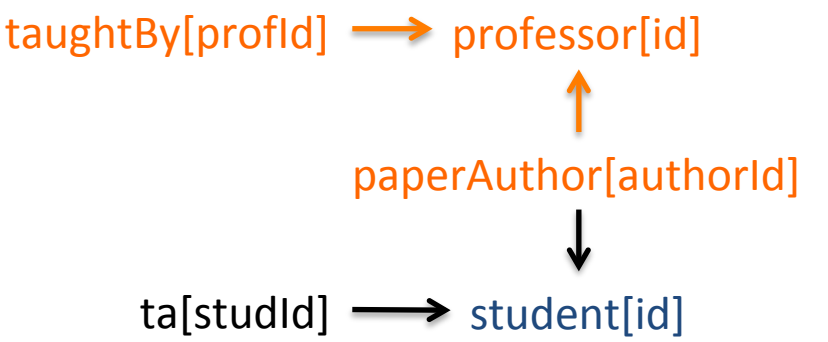
3. Assign a unique type to each node without outgoing edge



attribute	type
professor[id]	professor
student[id]	student

AutoMode: generate predicate definitions

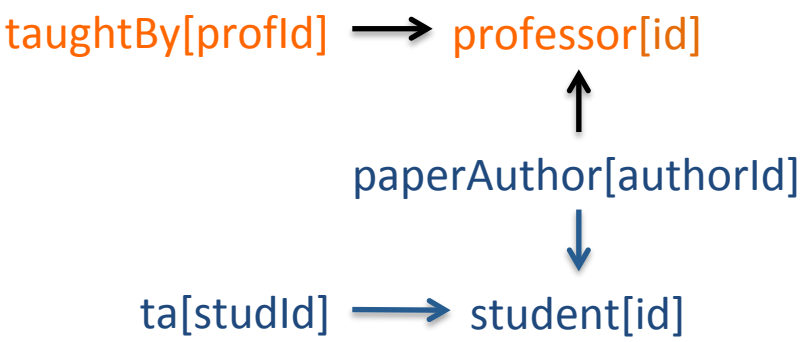
4. Propagate types backwards



attribute	type
professor[id]	professor
student[id]	student
taughtBy[profId]	professor
paperAuthor[authorId]	professor

AutoMode: generate predicate definitions

4. Propagate types backwards



attribute	type
professor[id]	professor
student[id]	student
taughtBy[profId]	professor
paperAuthor[authorId]	professor
paperAuthor[authorId]	student
ta[studId]	student

AutoMode: generate predicate definitions

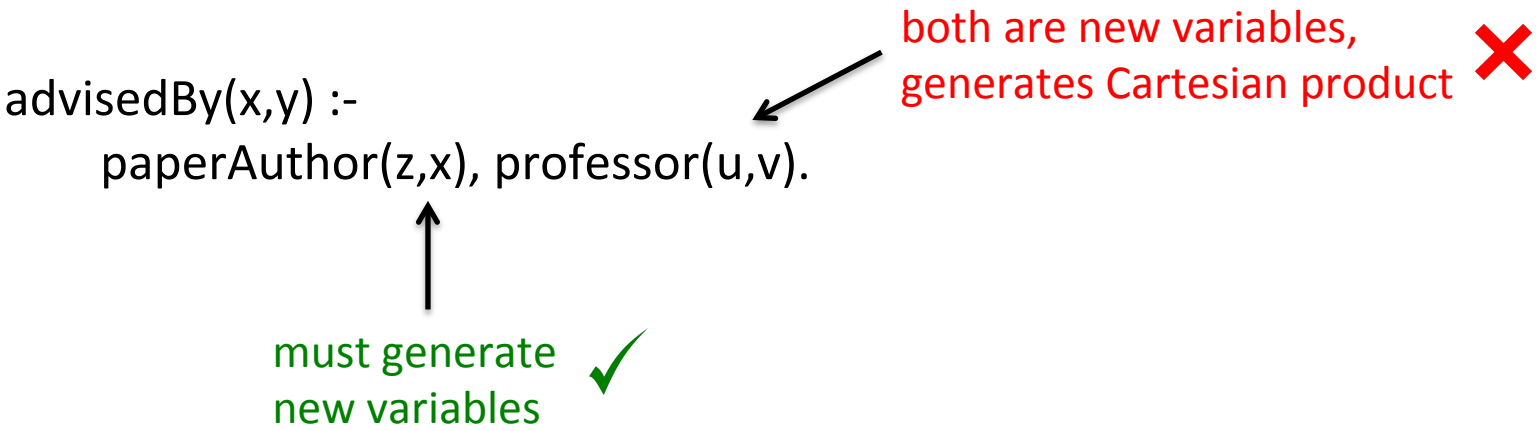
5. Assign unique types to attributes not in INDs and generate predicate definitions

attribute	type
professor[id]	professor
student[id]	student
taughtBy[profId]	professor
paperAuthor[authorId]	professor
paperAuthor[authorId]	student
ta[studId]	student
professor[position]	t0
paperAuthor[paperId]	t1
…	

professor(professor,t0)
paperAuthor(t1,professor)
paperAuthor(t1,student)
student(student,t4,t5)
taughtBy(t2,professor,t3)
…

AutoMode: generate mode definitions

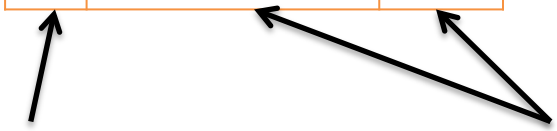
- Every attribute of every relation can be a variable
- Exactly one variable is an existing variable, rest can be new or existing variables



AutoMode: generate mode definitions

- Attributes can be constants if:
  - number of distinct values in the attribute is less than some threshold

student		
id	phase	year
s1	post_quals	3
s2	pre_quals	2
s3	post_prelims	5
s4	post_quals	3
s5	pre_quals	2



cannot be constants can be constants

Experimental settings

- Run relational learning system Castor[1] (SIGMOD'17) with different methods of generating syntactic bias
- Baseline:
  - All attributes are of the same type -> all joins possible
  - Every attribute can be a constant
- Baseline w/o constants:
  - Datalog definitions do not contain constants
- Manual tuning:
  - Syntactic bias written by an expert
- AutoMode

[1]Jose Picado et al. Schema Independent Relational Learning. SIGMOD 2017.

Databases

- IMDb: database about movies
- HIV: database about chemical compounds
- UW-CSE: database about an academic department

Database	Target relation	# relations	# tuples	# positive examples	# negative examples
IMDb	dramaDirector(dir)	46	8M	1.8K	3.6K
HIV	anti-HIV(comp)	80	14M	5.8K	36K
UW-CSE	advisedBy(stud,prof)	9	1.8K	102	204

Pre-processing step to generate predicate and mode definitions

- Baselines: no time
- Manual tuning: time taken by expert
- AutoMode (only done once for a dataset):
  - Extract exact INDs using Binder:
    - IMDb: 18 seconds
    - HIV: 40 seconds
    - UW-CSE: 1s
  - Extract approximate INDs:
    - IMDb: 53 minutes
    - HIV: 45 minutes
    - UW-CSE: 2 seconds

Experimental results

- F1-score: weighted average of precision and recall
- Time: **learning time** taken by Castor

Dataset	Measure	Baseline	Baseline w/o constants	Manual tuning	AutoMode
IMDb	F1-score	-	0.58	1	1
	Time	crashed	9.2h	2.7m	6.9m
HIV	F1-score	-	0.80	0.83	0.83
	Time	>36h	20h	23.7m	25.9m
UW-CSE	F1-score	0.60	0.64	0.68	0.67
	Time	30s	3.8s	8s	44s

Conclusions and future work

- Relational learning algorithms require language bias to be used effectively and efficiently
- It is time-consuming and difficult for users to write language bias
- AutoMode is able automatically generate language bias, and obtain similar results as manual tuning

- Future work:
  - Optimize pre-processing time
  - Automate relational learning: hyper-parameter tuning

Thank you