Schema Independent Relational Learning

Jose Picado, Arash Termehchy, Alan Fern, Parisa Ataei

Information and Data Management and Analytics (IDEA) Lab

[Logo: Oregon State University]
A compound has **anti-HIV** activity if it has the following substructure:
Relational learning

- Leverages the structure of the relational database
- Learns a Datalog definition

**Training data:**

**anti-HIV**
- compound(c1, u)
- atom(u, N)
- compound(c1, v)
- atom(v, O)
- compound(c1, w)
- atom(w, N)
- bond(u, v, single)
- bond(v, w, single)

**no-anti-HIV**
- compound(c2, u)
- atom(u, N)
- compound(c2, v)
- atom(v, O)
- compound(c2, w)
- atom(w, N)
- bond(u, v, single)
- bond(v, w, single)
Relational learning has many applications in data analytics & management

• Model entities and relationships between entities

Drug design
What is the structure of compounds to fight a disease?
Concept
active(compound)

Marketing
How will new customers respond to an offer?
Concept
interestedInOffer(customer)

• Various applications in data management
  • E.g., information extraction, usable query interfaces, data integration/ exchange.
Benefits of relational learning

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

**FOIL, Progol, ...**  
**Castor** (new algorithm)

Existing algorithms

```
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).
```
### Schema 1

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authorId</td>
<td>id</td>
</tr>
<tr>
<td>p1</td>
<td>mad</td>
<td>mad</td>
</tr>
<tr>
<td>p1</td>
<td>bai</td>
<td>sto</td>
</tr>
<tr>
<td>p2</td>
<td>soc</td>
<td>soc</td>
</tr>
<tr>
<td>p2</td>
<td>man</td>
<td>man</td>
</tr>
<tr>
<td>p3</td>
<td>mad</td>
<td>bai</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
<th>paperYear</th>
<th>paperConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
<td>id</td>
</tr>
<tr>
<td>p1</td>
<td>MacroBase: Priori...</td>
<td>p1</td>
</tr>
<tr>
<td>p2</td>
<td>GloVe: Global Vect...</td>
<td>p2</td>
</tr>
</tbody>
</table>

Which authors are **collaborators**?

<table>
<thead>
<tr>
<th>collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>person1</td>
</tr>
<tr>
<td>Madden</td>
</tr>
<tr>
<td>Socher</td>
</tr>
<tr>
<td>Madden</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>non-collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>person1</td>
</tr>
<tr>
<td>Madden</td>
</tr>
<tr>
<td>Manning</td>
</tr>
</tbody>
</table>

**FOIL learning algorithm**
FOIL: relational learning algorithm

Schema 1

author
- id
- name

authorAffiliation
- id
- affiliation

paperAuthor
- paperId
- authorId

paperConf
- id
- conf

paper
- id
- title

paperYear
- id
- year

collaborators(x,y) :-

Scoring function $f: P \rightarrow N$

$P$: positive examples covered
$N$: negative examples covered

collaborators(x,y) :-
true.
FOIL: relational learning algorithm

Schema 1

```
author(id, name)
authorAffiliation(id, affiliation)
paperAuthor(paperId, authorId)
paperConf(id, conf)
paper(id, title)
paperYear(id, year)
```

collaborators(x, y) :-

f=0
author(z, x)
f=0
author(z, y)
f=-1

Scoring function \( f: P - N \)
P: positive examples covered
N: negative examples covered

collaborators(x, y) :- true.
FOIL: relational learning algorithm

Schema 1

```
author
   id   name

authorAffiliation
   id   affiliation

paperAuthor
   paperId  authorId

collaborators(x,y) :-
    author(z,x).

paperConf
   id   conf

paper
   id   title

paperYear
   id   year
```

Scoring function \( f: P - N \)

- \( P \): positive examples covered
- \( N \): negative examples covered

\[
collaborators(x,y) :-
   author(z,x).
\]
FOIL: relational learning algorithm

Schema 1

```
author(id, name)
authorAffiliation(id, affiliation)
paperAuthor(paperId, authorId)
paperConf(id, conf)
paper(id, title)
paperYear(id, year)
```

Scoring function $f$: $P - N$

$P$: positive examples covered
$N$: negative examples covered

```
collaborators(x, y) :-
    author(z, x).
```

```
collaborators(x, y) :-
    author(z, x).
```

```
collaborators(x, y) :-
    author(z, y).
```

```
collaborators(x, y) :-
    author(v, y).
```

```
f=0
```

```
f=1
```

```
f=-1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```

```
f=0
```

```
f=1
```
FOIL: relational learning algorithm

Schema 1

```
author
    id   name
authorAffiliation
    id   affiliation
paperAuthor
    paperId authorId
paperConf
    id   conf
paper
    id   title
paperYear
    id   year
```

Scoring function \( f: P - N \)

\( P \): positive examples covered
\( N \): negative examples covered

collaborators(x,y) :-
    author(z,x), author(v,y).

FOIL: relational learning algorithm

Schema 1

author
  id name

authorAffiliation
  id affiliation

paperAuthor
  paperId authorId

paperConf
  id conf

table
  id title

paperYear
  id year

Scoring function $f$: $P - N$

$P$: positive examples covered
$N$: negative examples covered

collaborators(x,y) :-
  author(z,x), author(v,y).
**FOIL: relational learning algorithm**

### Schema 1

<table>
<thead>
<tr>
<th>Table</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>author</td>
<td>id, name</td>
</tr>
<tr>
<td>authorAffiliation</td>
<td>id, affiliation</td>
</tr>
<tr>
<td>paperAuthor</td>
<td>paperId, authorId</td>
</tr>
<tr>
<td>paperConf</td>
<td>id, conf</td>
</tr>
<tr>
<td>paper</td>
<td>id, title</td>
</tr>
<tr>
<td>paperYear</td>
<td>id, year</td>
</tr>
</tbody>
</table>

### Scoring function $f$: $P - N$

- **P**: positive examples covered
- **N**: negative examples covered

```
collaborators(x,y) :-
    author(z,x), author(v,y),
paperAuthor(w,z).
```
FOIL: relational learning algorithm

Schema 1

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>id</td>
<td>affiliation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperConf</th>
<th>paper</th>
<th>paperYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>conf</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>title</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>year</td>
<td></td>
</tr>
</tbody>
</table>

Scoring function \( f: P - N \)

\( P \): positive examples covered
\( N \): negative examples covered

collaborators(x,y) :-
   author(z,x), author(v,y), paperAuthor(w,z).
FOIL: relational learning algorithm

Schema 1

```
Schema 1

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>id</td>
</tr>
<tr>
<td></td>
<td>affiliation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
</tr>
<tr>
<td>authorId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>conf</td>
</tr>
</tbody>
</table>

| paper |
| id    |
| title |

| paperYear |
| id        |
| year      |
```

Scoring function $f$: $P - N$

$P$: positive examples covered

$N$: negative examples covered

```
collaborators(x,y) :-
    author(z,x), author(v,y),
    paperAuthor(w,z), paperAuthor(w,v).
```
FOIL: relational learning algorithm

Schema 1

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
<td>id</td>
</tr>
<tr>
<td>paperId</td>
<td>authorId</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperConf</th>
<th>paper</th>
<th>paperYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>conf</td>
<td>id</td>
</tr>
</tbody>
</table>

Scoring function $f$: $P - N$

$P$: positive examples covered

$N$: negative examples covered

No improvement

collaborators(x,y) :-
    author(z,x), author(v,y),
    paperAuthor(w,z), paperAuthor(w,v).
collaborators(x,y) :-
  author(z,x), author(v,y),
  paperAuthor(w,z), paperAuthor(w,v).

Two people are collaborators if they are co-authors.
People represent the same data using different schemas

**author**
- **id**: mad, sto, soc, man, bai
- **name**: Madden, Stonebraker, Socher, Manning, Bailis

**authorAffiliation**
- **id**: mad, sto, soc, man, bai
- **affiliation**: MIT, MIT, Stanford, Stanford, Stanford

**paper**
- **id**: p1, p2
- **title**: MacroBase: Priori..., GloVe: Global Vect...

**paperYear**
- **id**: p1, p2
- **year**: 2017, 2014

**paperConf**
- **id**: p1, p2
- **conf**: SIGMOD, EMNLP

**author**
- **id**: mad, sto, soc, man, bai
- **name**: Madden, Stonebraker, Socher, Manning, Bailis

**affiliation**
- **mad**: MIT
- **sto**: MIT
- **soc**: Stanford
- **man**: Stanford
- **bai**: Stanford

**paper**
- **id**: p1, p2
- **title**: MacroBase: Priori..., GloVe: Global Vect...
- **year**: 2017, 2014
- **conference**: SIGMOD, EMNLP

**Composition Denormalization better performance**

DBA
Which authors are collaborators?

<table>
<thead>
<tr>
<th>Collaborators</th>
<th>Non-collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madden</td>
<td>Madden</td>
</tr>
<tr>
<td>Bailis</td>
<td>Socher</td>
</tr>
<tr>
<td>Socher</td>
<td>Manning</td>
</tr>
<tr>
<td>Madden</td>
<td>Stonebraker</td>
</tr>
<tr>
<td>Stonebraker</td>
<td>Madden</td>
</tr>
<tr>
<td>Socher</td>
<td>Socher</td>
</tr>
<tr>
<td>Manning</td>
<td>Bailis</td>
</tr>
</tbody>
</table>

Schema 2

- **paperAuthor**
  - | paperId | authorId |
  - | p1      | mad      |
  - | p1      | bai      |
  - | p2      | soc      |
  - | p2      | man      |
  - | p3      | mad      |

- **author**
  - | id  | name    | affiliation |
  - | mad | Madden  | MIT         |
  - | sto | Stonebraker | MIT       |
  - | soc | Socher  | Stanford    |
  - | man | Manning | Stanford    |
  - | bai | Bailis  | Stanford    |

- **paper**
  - | id  | title               | year | conference |
  - | p1  | MacroBase: Priori... | 2017 | SIGMOD     |
  - | p2  | GloVe: Global Vect... | 2014 | EMNLP      |

- FOIL learning algorithm
FOIL: relational learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

Scoring function $f$: $P - N$

$P$: positive examples covered
$N$: negative examples covered

collaborators(x,y) :-
true.
FOIL: relational learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>author</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
<td>affiliation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authorId</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
<td>year</td>
<td>conference</td>
</tr>
</tbody>
</table>

collaborators(x,y) :-

f=0

author(z,x,v)

f=0

author(z,y,v)

f=-1

collaborators(x,y) :-

true.

Scoring function \( f: P - N \)

\( P \): positive examples covered

\( N \): negative examples covered
FOIL: relational learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>author</th>
<th>paperAuthor</th>
<th>paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>paperId</td>
<td>id</td>
</tr>
<tr>
<td>name</td>
<td>authorId</td>
<td>title</td>
</tr>
<tr>
<td>affiliation</td>
<td></td>
<td>year</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>conference</td>
</tr>
</tbody>
</table>

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

$\text{collaborators}(x,y) :-$
- $\text{author}(z,x,v)$
- $\text{author}(z,y,v)$
FOIL: relational learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>author</th>
<th>id</th>
<th>name</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperAuthor</td>
<td>paperId</td>
<td>authorId</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
<th>id</th>
<th>title</th>
<th>year</th>
<th>conference</th>
</tr>
</thead>
</table>

Scoring function \( f: P - N \)

\( P \): positive examples covered  
\( N \): negative examples covered

\[
\text{collaborators}(x,y) :\quad \begin{cases} 
\text{author}(z,x,v) & f=0 \\
\text{author}(z,y,v) & f=0 \\
\text{author}(w,y,u) & f=-1 \\
\text{author}(w,y,v) & f=1 \\
\text{author}(w,y,v) & f=2 \\
\end{cases}
\]

\[
\text{collaborators}(x,y) : \quad \text{author}(z,x,v).
\]
FOIL: relational learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>author</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>affiliation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authorId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
</tr>
<tr>
<td></td>
<td>year</td>
</tr>
<tr>
<td></td>
<td>conference</td>
</tr>
</tbody>
</table>

Scoring function $f$: $P - N$

$P$: positive examples covered

$N$: negative examples covered

collaborators($x, y$) :-
  author($z, x, v$), author($w, y, v$).
FOIL: relational learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

Scoring function $f$: $P - N$

$P$: positive examples covered

$N$: negative examples covered

$\text{collaborators}(x, y) :-$ 
$\text{author}(z, x, v), \text{author}(w, y, v)$. 

No improvement
Two people are collaborators if they work in the same institution.

collaborators(x,y) :-
  author(z,x,v), author(w,y,v).

foil learning algorithm

Schema 2

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authorId</td>
</tr>
<tr>
<td>p1</td>
<td>mad</td>
</tr>
<tr>
<td>p1</td>
<td>bai</td>
</tr>
<tr>
<td>p2</td>
<td>soc</td>
</tr>
<tr>
<td>p2</td>
<td>man</td>
</tr>
<tr>
<td>p3</td>
<td>mad</td>
</tr>
</tbody>
</table>

Which authors are collaborators?

- Collaborators:
  - person1: Madden
  - person2: Bailis
  - person1: Socher
  - person2: Manning
  - person1: Madden
  - person2: Stonebraker

- Non-collaborators:
  - person1: Madden
  - person2: Socher
  - person1: Manning
  - person2: Bailis

f=2
Schema dependence: schema affects the learning outcomes

**Schema 1**

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
</tr>
<tr>
<td>soc</td>
<td>Socher</td>
</tr>
<tr>
<td>man</td>
<td>Manning</td>
</tr>
</tbody>
</table>

**Schema 2**

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
</tr>
<tr>
<td>soc</td>
<td>Socher</td>
</tr>
<tr>
<td>man</td>
<td>Manning</td>
</tr>
</tbody>
</table>

**FOIL learning algorithm**

- collaborators(x,y) :-
  author(z,x), author(v,y),
  paperAuthor(w,z), paperAuthor(w,v).

  Two people are collaborators if they are co-authors.

- collaborators(x,y) :-
  author(z,x,v), author(w,y,v).

  Two people are collaborators if they work in the same institution.
Current solutions

<table>
<thead>
<tr>
<th>Author</th>
<th>Author Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>mad</td>
<td>Madden MIT</td>
</tr>
</tbody>
</table>

Users must restructure databases

Restructure | Learn | Evaluate

Which is the best schema?

Expert attention
Definition of schema independence

**Algorithm A**

- **Schema 1**
  - **author**
    - id: mad
    - name: Madden
    - id: sto
    - name: Stonebraker
  - **authorAffiliation**
    - id: mad
    - affiliation: MIT
    - id: sto
    - affiliation: MIT

- **collaborators**
  - person1
  - person2

- **non-collaborators**
  - person1
  - person2

- **Algorithm A**

- **h_1**

- **Equivalent?**

- **Schema 2**
  - **author**
    - id: mad
    - name: Madden
    - affiliation: MIT
    - id: sto
    - name: Stonebraker
    - affiliation: MIT

- **Algorithm A**

- **h_2**
Definition of schema independence

Transformation $T$: Preserve information in the DB

<table>
<thead>
<tr>
<th>Schema 1</th>
<th>Schema 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>author</strong></td>
<td><strong>authorAffiliation</strong></td>
</tr>
<tr>
<td>id</td>
<td>id</td>
</tr>
<tr>
<td>mad</td>
<td>mad</td>
</tr>
<tr>
<td>sto</td>
<td>sto</td>
</tr>
<tr>
<td><strong>collaborators</strong></td>
<td><strong>non-collaborators</strong></td>
</tr>
<tr>
<td>person1</td>
<td>person2</td>
</tr>
<tr>
<td>person1</td>
<td>person2</td>
</tr>
<tr>
<td><strong>author</strong></td>
<td><strong>authorAffiliation</strong></td>
</tr>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
</tr>
</tbody>
</table>

Algorithm A

Equivalent?

$h_1$

$h_2$
Definition of schema independence

Algorithm A is **schema independent** under \( T \) iff for all pairs of databases \((I, J)\) and training examples \(E\), \(h_1\) and \(h_2\) are equivalent.
We focus on schema independence under composition/decomposition

• Most common schema transformations
• Used in normalization and denormalization
• We support combinations of compositions and decompositions

Inclusion dependencies (referential integrity constraints):
author[id] \(\subseteq\) authorAffiliation[id]
Current relational learning algorithms are NOT schema independent

Theorems:
- FOIL
- Progol
- ProGolem

are NOT schema independent under composition/decomposition

Reasons for schema dependence:
- Search process affected by schema
- Greedy search strategies
Our algorithm: **Castor**
schema independent algorithm

- Specific to general definitions
- Uses database constraints to achieve schema independence
Step 1: Create most specific definition

```
collaborators(v1, v2) :-
```

**Table: paperAuthor**
<table>
<thead>
<tr>
<th>paperId</th>
<th>authorId</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>mad</td>
</tr>
<tr>
<td>p1</td>
<td>bai</td>
</tr>
</tbody>
</table>

**Table: author**
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
</tr>
</tbody>
</table>

**Table: authorAffiliation**
<table>
<thead>
<tr>
<th>id</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>MIT</td>
</tr>
<tr>
<td>bai</td>
<td>Stanford</td>
</tr>
</tbody>
</table>

**Table: paper**
<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>MacroBase: Priori...</td>
</tr>
<tr>
<td>p2</td>
<td>GloVe: Global Vect...</td>
</tr>
</tbody>
</table>

**Table: paperYear**
<table>
<thead>
<tr>
<th>id</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>2017</td>
</tr>
<tr>
<td>p2</td>
<td>2014</td>
</tr>
</tbody>
</table>

**Table: paperConf**
<table>
<thead>
<tr>
<th>id</th>
<th>conf</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>SIGMOD</td>
</tr>
<tr>
<td>p2</td>
<td>EMNLP</td>
</tr>
</tbody>
</table>
Step 1: Create most specific definition

**Create most specific definition**

- Madden, Bailis

**Generalize to cover new example**

- **Did it improve?**
  - **Yes**
  - **No**

- **Reduce**

**collaborators**(v₁, v₂) :-
  author(v₃, v₁), author(v₄, v₂).

**paperAuthor**

<table>
<thead>
<tr>
<th>paperId</th>
<th>authorId</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>mad</td>
</tr>
<tr>
<td>p1</td>
<td>bai</td>
</tr>
</tbody>
</table>

**author**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
</tr>
</tbody>
</table>

**authorAffiliation**

<table>
<thead>
<tr>
<th>id</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>MIT</td>
</tr>
<tr>
<td>bai</td>
<td>Stanford</td>
</tr>
</tbody>
</table>

**paper**

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>MacroBase: Priori...</td>
</tr>
<tr>
<td>p2</td>
<td>GloVe: Global Vect...</td>
</tr>
</tbody>
</table>

**paperYear**

<table>
<thead>
<tr>
<th>id</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>2017</td>
</tr>
<tr>
<td>p2</td>
<td>2014</td>
</tr>
</tbody>
</table>

**paperConf**

<table>
<thead>
<tr>
<th>id</th>
<th>conf</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>SIGMOD</td>
</tr>
<tr>
<td>p2</td>
<td>EMNLP</td>
</tr>
</tbody>
</table>
**Step 1: Create most specific definition**

- **Start**: Create most specific definition
  - **Madden, Bailis**
  - **Generalize to cover new example**
  - **Did it improve?**
    - **Yes**
      - **Paper**: MacroBase: Priori...
        - **Year**: 2017
        - **Conference**: SIGMOD
    - **No**
      - **Reduce**

- **Paper Authors**:
  - **Madden**: MIT
  - **Bailis**: Stanford

- **Collaborators**:
  - \(v_1, v_2\)
  - \(v_3, v_4\)
  - \(v_5, v_6\)
  - \(author(v_3, v_1), author(v_4, v_2),\)
  - \(authorAffiliation(v_3, MIT), authorAffiliation(v_3, v_5),\)
  - \(authorAffiliation(v_4, Stanford), authorAffiliation(v_4, v_6).\)
Step 1: Create most specific definition

```
collaborators(v_1, v_2) :-
    author(v_3, v_1), author(v_4, v_2),
    authorAffiliation(v_3, MIT), authorAffiliation(v_3, v_5),
    authorAffiliation(v_4, Stanford), authorAffiliation(v_4, v_6),
    paperAuthor(v_7, v_3), paperAuthor(v_7, v_4).
```

```
collaborators:
<table>
<thead>
<tr>
<th>v_1</th>
<th>v_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>bai</td>
</tr>
</tbody>
</table>
```
Step 2: Generalize definition

v₁ -> Socher
v₂ -> Manning

v₁, v₂ : collaborators(v₁, v₂) :-
  author(v₃, v₁), author(v₄, v₂),
  authorAffiliation(v₃, MIT), authorAffiliation(v₃, v₅),
  authorAffiliation(v₄, Stanford), authorAffiliation(v₄, v₆),
  paperAuthor(v₇, v₃), paperAuthor(v₇, v₄).

f = P - N = 1

Create most specific definition

Generalize to cover new example

Did it improve?

Yes

No

Reduce

Madden, Bailis

Socher, Manning

Madden, Bailis

Socher, Manning

Start

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authorId</td>
<td></td>
</tr>
<tr>
<td></td>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>id</td>
<td>affiliation</td>
</tr>
<tr>
<td>p2</td>
<td>soc</td>
<td>Socher</td>
</tr>
<tr>
<td>p2</td>
<td>man</td>
<td>Manning</td>
</tr>
<tr>
<td></td>
<td>soc</td>
<td>Stanford</td>
</tr>
<tr>
<td></td>
<td>man</td>
<td>Stanford</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
<th>paperYear</th>
<th>paperConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
<td>id</td>
</tr>
<tr>
<td>p2</td>
<td>GloVe: Global Vect...</td>
<td>p2</td>
</tr>
</tbody>
</table>
Step 2: Generalize definition

```
Step 2: Generalize definition

Madden, Bailis

Create most specific definition

Socher, Manning

Generalize to cover new example

Did it improve?

Yes

Madden, Bailis

No

Reduce

Start

v_1 -> Socher
v_2 -> Manning

f = P - N = 2

v_1 \rightarrow Socher
v_2 \rightarrow Manning

\text{collaborators}(v_1, v_2) :-
  \text{author}(v_3, v_1), \text{author}(v_4, v_2),
  \text{authorAffiliation}(v_3, v_5),
  \text{authorAffiliation}(v_4, \text{Stanford}), \text{authorAffiliation}(v_4, v_6),
  \text{paperAuthor}(v_7, v_3), \text{paperAuthor}(v_7, v_4).
```
Step 2: Generalize definition

Create most specific definition

Socher, Manning

Generalize to cover new example

Madden, Bailis

Did it improve?

Yes

No

Reduce

f = P – N = 2

collaborators(v_1, v_2) :-
  author(v_3, v_1), author(v_4, v_2),
  authorAffiliation(v_3, v_5),
  authorAffiliation(v_4, Stanford), authorAffiliation(v_4, v_6),
  paperAuthor(v_7, v_3), paperAuthor(v_7, v_4).
Step 2: Generalize definition

Create most specific definition

Madden, Bailis

Generalize to cover new example

Madden, Stonebraker

Did it improve?

Yes

No

Reduce

repeat

f = P - N = 3

collaborators(v_1, v_2) :-
  author(v_3, v_1), author(v_4, v_2),
  authorAffiliation(v_3, v_5), authorAffiliation(v_4, v_6),
  paperAuthor(v_7, v_3), paperAuthor(v_7, v_4).

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authorId</td>
<td>id</td>
</tr>
<tr>
<td>p3</td>
<td>mad</td>
<td>mad</td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
<td>mad</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paper</th>
<th>paperYear</th>
<th>paperConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
<td>id</td>
</tr>
<tr>
<td>p3</td>
<td>The Data Civilizer...</td>
<td>p3</td>
</tr>
</tbody>
</table>
Step 3: Reduce definition

- Generalize even more to avoid overfitting
- Reduce definition using negative examples

流程图:
1. Start
2. Create most specific definition
3. Madden, Bailis
4. Generalize to cover new example
5. Madden, Stonebraker
6. Did it improve?
   - Yes
   - No
   - Reduce
Learned definition

Two people are collaborators if they are co-authors.

collaborators(v₁,v₂) :-
    author(v₃,v₁), author(v₄,v₂),
    paperAuthor(v₇,v₃), paperAuthor(v₇,v₄).

f = P – N = 3

Madden, Bailis

Madden, Stonebraker

Create most specific definition

Generalize to cover new example

Did it improve?

Yes

No

Reduce
Castor achieves schema independence by using database constraints.

<table>
<thead>
<tr>
<th>author</th>
<th></th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
<td>id</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
<td>mad</td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
<td>bai</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th></th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authId</td>
<td>id</td>
</tr>
<tr>
<td>p3</td>
<td>mad</td>
<td>mad</td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
<td>bai</td>
</tr>
</tbody>
</table>

The schema independence is achieved by using database constraints:

- `author[id] \subseteq authorAffiliation[id]`
- `author[id] \subseteq paperAuthor[authId]`
Step 1: Create most specific definition using database constraints

<table>
<thead>
<tr>
<th>author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>Madden</td>
<td></td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>authorAffiliation</th>
<th>id</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>MIT</td>
<td></td>
</tr>
<tr>
<td>bai</td>
<td>Stanford</td>
<td></td>
</tr>
</tbody>
</table>

Create most specific definition

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>paperId</th>
<th>authId</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>mad</td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
<td></td>
</tr>
</tbody>
</table>

collaborators(v₁, v₂) : -
Step 1: Create most specific definition using database constraints

<table>
<thead>
<tr>
<th>author</th>
<th></th>
<th>authorAffiliation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
<td>id</td>
<td>affiliation</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
<td>mad</td>
<td>MIT</td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
<td>bai</td>
<td>Stanford</td>
</tr>
</tbody>
</table>

Create most specific definition

<table>
<thead>
<tr>
<th>author</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authId</td>
</tr>
<tr>
<td>p3</td>
<td>mad</td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
</tr>
</tbody>
</table>

collaborators(v₁, v₂) :-
author(v₃,v₁), author(v₄,v₂).

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
<td>authId</td>
</tr>
<tr>
<td>p3</td>
<td>mad</td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
</tr>
</tbody>
</table>

collaborators(v₁, v₂) :-
author(v₃,v₁,MIT), author(v₄,v₂,Stanford).
Step 1: Create most specific definition using database constraints

<table>
<thead>
<tr>
<th>author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>Madden</td>
<td></td>
</tr>
<tr>
<td>bai</td>
<td>Bailis</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>authorAffiliation</th>
<th>id</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>MIT</td>
<td></td>
</tr>
<tr>
<td>bai</td>
<td>Stanford</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>paperId</th>
<th>authId</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>mad</td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
<td></td>
</tr>
</tbody>
</table>

collaborators($v_1$, $v_2$) :-
  author($v_3$, $v_1$), author($v_4$, $v_2$),
  authorAffiliation($v_3$, MIT), paperAuthor($v_3$, $v_5$),
  authorAffiliation($v_4$, Stanford), paperAuthor($v_4$, $v_6$).

Ensures that the algorithm accesses the same information over all schemas
Step 2 and 3: Generalization and reduction using database constraints

<table>
<thead>
<tr>
<th>author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>Madden</td>
<td></td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>authorAffiliation</th>
<th>id</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>MIT</td>
<td></td>
</tr>
<tr>
<td>sto</td>
<td>MIT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>paperId</th>
<th>authId</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>mad</td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
<td></td>
</tr>
</tbody>
</table>

Generalize to cover new example

<table>
<thead>
<tr>
<th>author</th>
<th>id</th>
<th>name</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>Madden</td>
<td>MIT</td>
<td></td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
<td>MIT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
<th>paperId</th>
<th>authId</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>mad</td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>sto</td>
<td></td>
</tr>
</tbody>
</table>

collaborators(v₁, v₂) :-
author(v₃, v₁, MIT), authorAffiliation(v₃, MIT),
author(v₄, v₂, Stanford).

collaborators(v₁, v₂) :-
author(v₃, v₁, MIT), author(v₄, v₂, Stanford).
Step 2 and 3: Generalization and reduction using database constraints

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad</td>
<td>MIT</td>
</tr>
<tr>
<td>sto</td>
<td>MIT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>author</th>
<th>authorAffiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>id</td>
<td>affiliation</td>
</tr>
<tr>
<td>mad</td>
<td>Madden</td>
</tr>
<tr>
<td>mad</td>
<td>MIT</td>
</tr>
<tr>
<td>sto</td>
<td>Stonebraker</td>
</tr>
<tr>
<td>sto</td>
<td>MIT</td>
</tr>
</tbody>
</table>

Generalize to cover new example

collaborators(v₁, v₂) :-
    author(v₃, v₁), authorAffiliation(v₃, MIT),
    author(v₄, v₂), authorAffiliation(v₄, Stanford).

collaborators(v₁, v₂) :-
    author(v₃, v₁, MIT), author(v₄, v₂, Stanford).

<table>
<thead>
<tr>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
</tr>
<tr>
<td>p3</td>
</tr>
<tr>
<td>p3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>paperAuthor</th>
</tr>
</thead>
<tbody>
<tr>
<td>paperId</td>
</tr>
<tr>
<td>p3</td>
</tr>
<tr>
<td>p3</td>
</tr>
</tbody>
</table>
Step 2 and 3: Generalization and reduction using database constraints

Generalize to cover new example

More details in the paper!
Step 2 and 3: Generalization and reduction using database constraints

Theorem: Castor is schema independent under composition / decomposition.
Techniques to achieve efficiency

1. Castor is implemented on top of the in-memory RDBMS VoltDB
   - Exploit RDBMS mechanisms
   - Part of the algorithm implemented in a stored procedure

2. Approximate and efficient definition minimization
Techniques to achieve efficiency

3. Castor efficiently checks whether a definition covers an example

Alternative approach:

**Datalog:**
collaborators(x, y) :-
    author(z, x), author(v, y), paperAuthor(w, z), paperAuthor(w, v).

**SQL:**
SELECT c.person1, c.person2
FROM collaborators c, author a1, author a2, paperAuthor pa1, paperAuthor pa2
WHERE c.person1 = a1.name AND c.person2 = a2.name AND a1.id = pa1.authorId
      AND a2.id = pa2.authorId AND pa1.id = pa2.id;

Castor’s approach:
1. Compute most specific definition $h_e$ for example $e$.
2. Definition $h$ covers example $e$ iff there is a substitution $\theta$ such that $h\theta \subseteq h_e$ (homomorphism).

✓ More efficient
Experimental results

- Database: UW-CSE – academic department
  - 9 relations, 2K tuples
  - 102 positive examples, 204 negative examples
- Target relation: `advisedBy(student, professor)`

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Metric</th>
<th>Schema 1</th>
<th>Schema 2</th>
<th>Schema 3</th>
<th>Schema 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOIL</td>
<td>F1-score</td>
<td>0.49</td>
<td>0.49</td>
<td>0.54</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Time (s)</td>
<td>18.7</td>
<td>20.8</td>
<td>30.7</td>
<td>30.6</td>
</tr>
<tr>
<td>Progol</td>
<td>F1-score</td>
<td><strong>0.68</strong></td>
<td>0.61</td>
<td>0.53</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>Time (s)</td>
<td>9.7</td>
<td>13.2</td>
<td>27.9</td>
<td>334.8</td>
</tr>
<tr>
<td>ProGolem</td>
<td>F1-score</td>
<td><strong>0.68</strong></td>
<td><strong>0.68</strong></td>
<td>0.60</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Time (s)</td>
<td>24.4</td>
<td>28.8</td>
<td>26.7</td>
<td>54.1</td>
</tr>
<tr>
<td>Castor</td>
<td>F1-score</td>
<td><strong>0.68</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.68</strong></td>
</tr>
<tr>
<td></td>
<td>Time (s)</td>
<td><strong>7.2</strong></td>
<td>7.4</td>
<td>7.9</td>
<td>12.4</td>
</tr>
</tbody>
</table>
Experimental results

• Database: HIV – structure of chemical compounds
  – 80 relations, 14M tuples
  – 5K positive examples, 36K negative examples

• Target relation: anti-HIV(compound)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Metric</th>
<th>Schema 1</th>
<th>Schema 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOIL</td>
<td>F1-score</td>
<td>0.49</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>Time (h)</td>
<td>3</td>
<td>0.9</td>
</tr>
<tr>
<td>Castor</td>
<td>F1-score</td>
<td>0.83</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>Time(h)</td>
<td>3.5</td>
<td>1.9</td>
</tr>
</tbody>
</table>

Progol and ProGolem do not terminate after 5 days
Conclusions and future work

• Relational learning algorithms leverage the structure of data to learn Datalog definitions
• Schema independence is a desired property
• Current algorithms are not schema independent
• Castor is schema independent, accurate and efficient

• Future work:
  – Achieve schema independence over other transformations
  – Learn over different data sources