Data Structure Estimation and Semantic Web

Martin Řimnáč

rimnacm@cs.cas.cz

VŠE – KEG

Knowledge Engineering Group

April 27, 2006

Institute of Computer Science

Data Structure Estimation and Semantic Web

Martin Řimnáč
rimnacm@cs.cas.cz

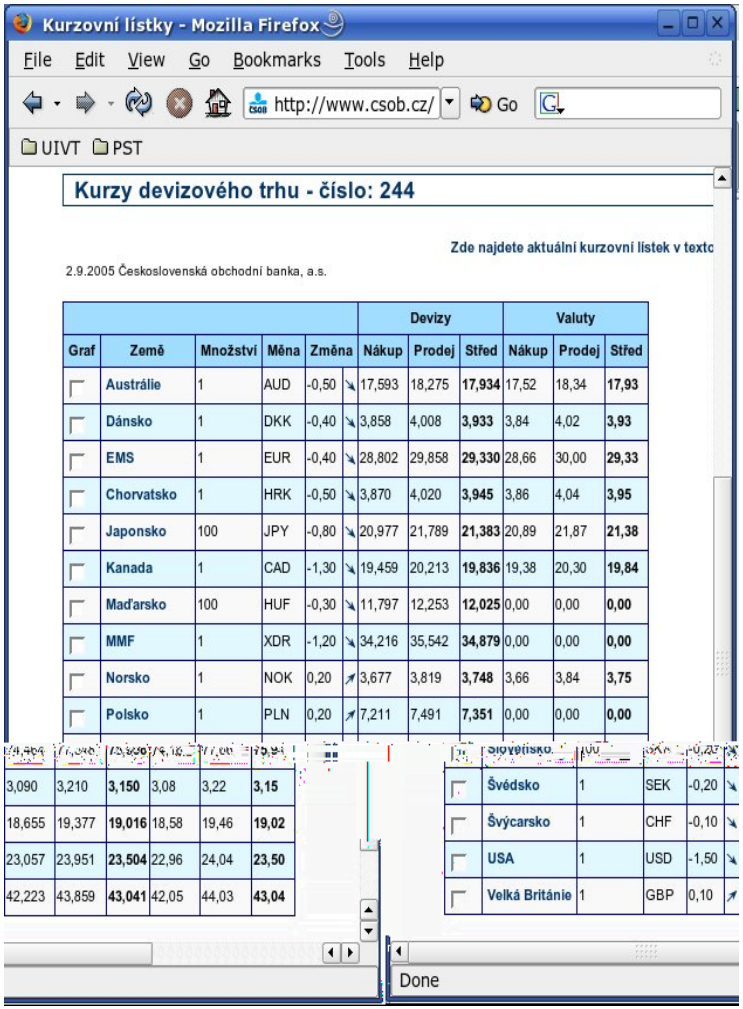
VŠE
Knowledge E

Apri

- Data Structure Estimation
- Inductive Logic Programming
- Semantic Web Portal Building
- Binary Matrix Notion

Institute of Computer Science

Data Structure Estimation - Motivation



Graf	Země	Množství	Měna	Změna	Devizy Nákup	Devizy Prodej	Devizy Střed	Valuty Nákup	Valuty Prodej	Valuty Střed
	Austrálie	1	AUD	-0,50	17,593	18,275	**17,934**	17,52	18,34	**17,93**
	Dánsko	1	DKK	-0,40	3,858	4,008	**3,933**	3,84	4,02	**3,93**
	EMS	1	EUR	-0,40	28,802	29,858	**29,330**	28,66	30,00	**29,33**
	Chorvatsko	1	HRK	-0,50	3,870	4,020	**3,945**	3,86	4,04	**3,95**
	Japonsko	100	JPY	-0,80	20,977	21,789	**21,383**	20,89	21,87	**21,38**
	Kanada	1	CAD	-1,30	19,459	20,213	**19,836**	19,38	20,30	**19,84**
	Maďarsko	100	HUF	-0,30	11,797	12,253	**12,025**	0,00	0,00	**0,00**
	MMF	1	XDR	-1,20	34,216	35,542	**34,879**	0,00	0,00	**0,00**
	Norsko	1	NOK	0,20	3,677	3,819	**3,748**	3,66	3,84	**3,75**
	Polsko	1	PLN	0,20	7,211	7,491	**7,351**	0,00	0,00	**0,00**

3,090	3,210	**3,150**	3,08	3,22	**3,15**
18,655	19,377	**19,016**	18,58	19,46	**19,02**
23,057	23,951	**23,504**	22,96	24,04	**23,50**
42,223	43,859	**43,041**	42,05	44,03	**43,04**

Švédsko	1	SEK	-0,20
Švýcarsko	1	CHF	-0,10
USA	1	USD	-1,50
Velká Británie	1	GBP	0,10

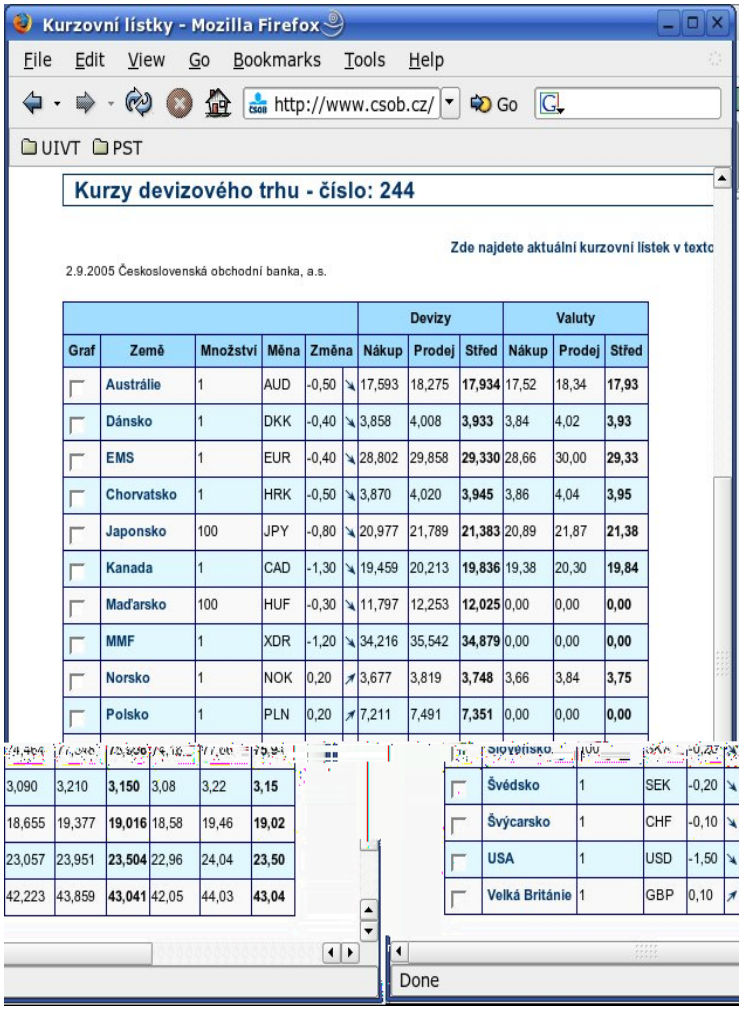
Web pages contain a lot of information.
Many of them present a view into database systems.

The work aim:
- Extract information from a web page
- Store it back by an effective way into:
  - a database system
  - a semantic web portfolio (portal)

To provide this:
- A data structure has to be known.

(Normal Database Form, RDF/OWL structure)

Data Structure Estimation - Similarities



Graf	Země	Množství	Měna	Změna	Devizy Nákup	Devizy Prodej	Devizy Střed	Valuty Nákup	Valuty Prodej	Valuty Střed
	Austrálie	1	AUD	-0,50	17,593	18,275	**17,934**	17,52	18,34	**17,93**
	Dánsko	1	DKK	-0,40	3,858	4,008	**3,933**	3,84	4,02	**3,93**
	EMS	1	EUR	-0,40	28,802	29,858	**29,330**	28,66	30,00	**29,33**
	Chorvatsko	1	HRK	-0,50	3,870	4,020	**3,945**	3,86	4,04	**3,95**
	Japonsko	100	JPY	-0,80	20,977	21,789	**21,383**	20,89	21,87	**21,38**
	Kanada	1	CAD	-1,30	19,459	20,213	**19,836**	19,38	20,30	**19,84**
	Maďarsko	100	HUF	-0,30	11,797	12,253	**12,025**	0,00	0,00	**0,00**
	MMF	1	XDR	-1,20	34,216	35,542	**34,879**	0,00	0,00	**0,00**
	Norsko	1	NOK	0,20	3,677	3,819	**3,748**	3,66	3,84	**3,75**
	Polsko	1	PLN	0,20	7,211	7,491	**7,351**	0,00	0,00	**0,00**

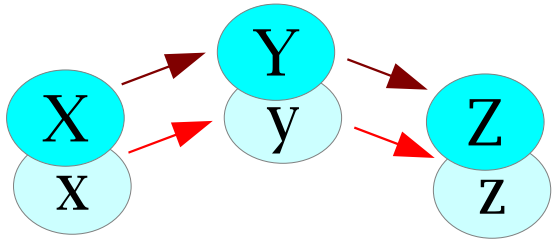
3,090	3,210	**3,150**	3,08	3,22	**3,15**
18,655	19,377	**19,016**	18,58	19,46	**19,02**
23,057	23,951	**23,504**	22,96	24,04	**23,50**
42,223	43,859	**43,041**	42,05	44,03	**43,04**

Švédsko	1	SEK	-0,20	
Švýcarsko	1	CHF	-0,10	
USA	1	USD	-1,50	
Velká Británie	1	GBP	0,10	

Data Structure Estimation

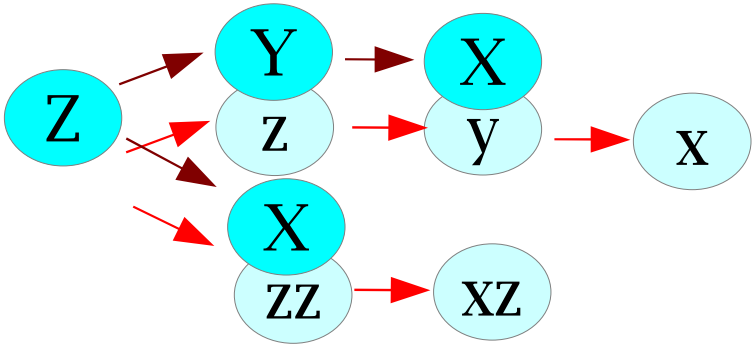
- machine learning method
  - using a global criterion

(a functional dependency system)

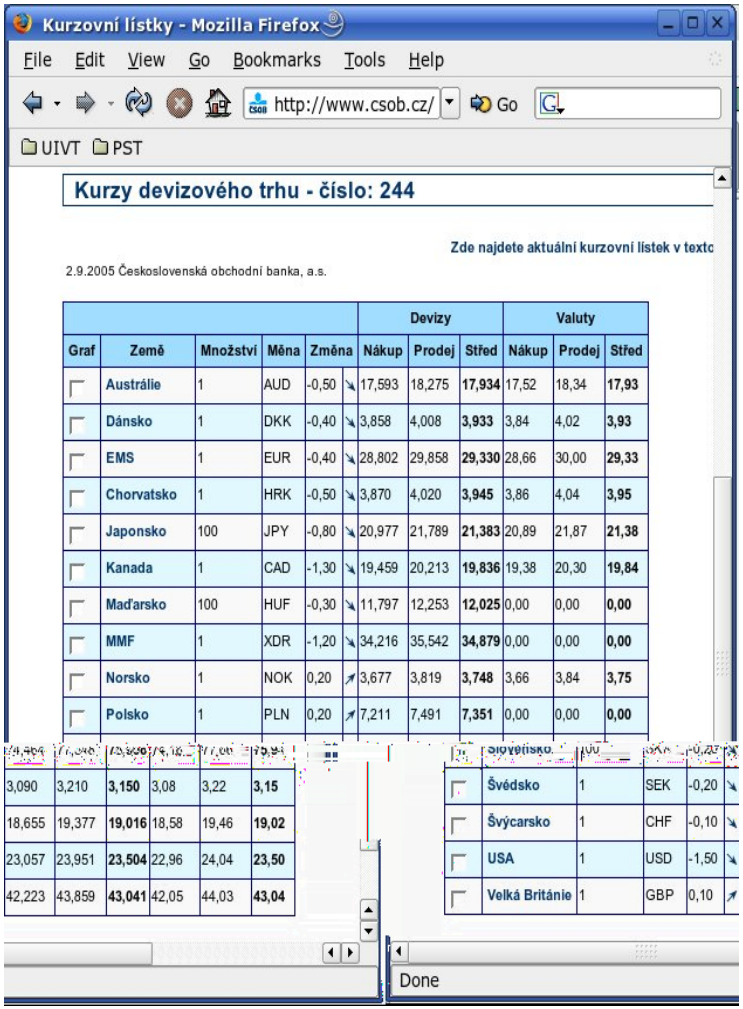


- similar to a decision trees making
  - using a local criterion

(entropy)



Data Structure Estimation Aim



Graf	Země	Množství	Měna	Změna	Devizy Nákup	Devizy Prodej	Devizy Střed	Valuty Nákup	Valuty Prodej	Valuty Střed
	Austrálie	1	AUD	-0,50	17,593	18,275	**17,934**	17,52	18,34	**17,93**
	Dánsko	1	DKK	-0,40	3,858	4,008	**3,933**	3,84	4,02	**3,93**
	EMS	1	EUR	-0,40	28,802	29,858	**29,330**	28,66	30,00	**29,33**
	Chorvatsko	1	HRK	-0,50	3,870	4,020	**3,945**	3,86	4,04	**3,95**
	Japonsko	100	JPY	-0,80	20,977	21,789	**21,383**	20,89	21,87	**21,38**
	Kanada	1	CAD	-1,30	19,459	20,213	**19,836**	19,38	20,30	**19,84**
	Maďarsko	100	HUF	-0,30	11,797	12,253	**12,025**	0,00	0,00	**0,00**
	MMF	1	XDR	-1,20	34,216	35,542	**34,879**	0,00	0,00	**0,00**
	Norsko	1	NOK	0,20	3,677	3,819	**3,748**	3,66	3,84	**3,75**
	Polsko	1	PLN	0,20	7,211	7,491	**7,351**	0,00	0,00	**0,00**

3,090	3,210	**3,150**	3,08	3,22	**3,15**		Švédsko	1	SEK	-0,20
18,655	19,377	**19,016**	18,58	19,46	**19,02**		Švýcarsko	1	CHF	-0,10
23,057	23,951	**23,504**	22,96	24,04	**23,50**		USA	1	USD	-1,50
42,223	43,859	**43,041**	42,05	44,03	**43,04**		Velká Británie	1	GBP	0,10

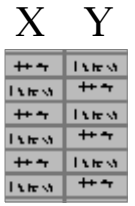
Extract data
from a web page

Estimate
extensional functional dependency system
respecting
an input data set

Store data
according to
estimated model

Data Structure Description

A description by a model of functional dependencies
(same as in relational database theory)



The Y attribute is functionally depended on the X attribute
(denote X->Y)
iff
the Y attribute value can be unambiguously derived from the X attribute value.

Properties:

- Transitivity — X->Y and Y->Z, then X->Z
- Hierarchy — X->Y, then also trivial {X,Z}->Y

Effective data storage:

The set of functional dependencies is in the 3NF.

Naïve algorithm

Computation of dependencies holding relation R:

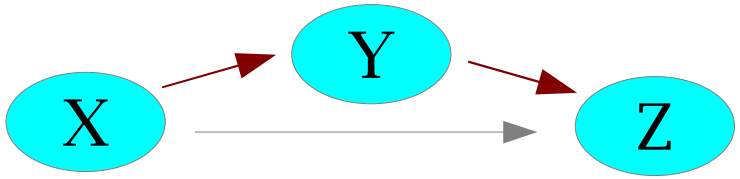
$$
\begin{aligned}
&M = \emptyset \\
&\text{for } \forall \text{ attributes } A \in R \\
&\quad \text{for } \forall \text{ subsets } X \subseteq R \setminus \{A\} \\
&\quad\quad \text{if } \exists \wp : X \rightarrow A \text{ then} \\
&\quad\quad\quad F = F \cup \{X \rightarrow A\}
\end{aligned}
$$

Complexity

$$O(n2^{(n-1)} * n * m^2)$$

Model Skeleton

Because of transitivity and hierarchy,
consideration of all functional dependencies is redundant.



The model skeleton is a minimal set of valid functional dependencies.
All other functional dependencies can be derived from this set.
The model skeleton set represents a kernel of functional dependency system.
The model skeleton covers all elementary relationships presented in the model.

To find a kernel of the set:

NP-complete issue.

Solution:

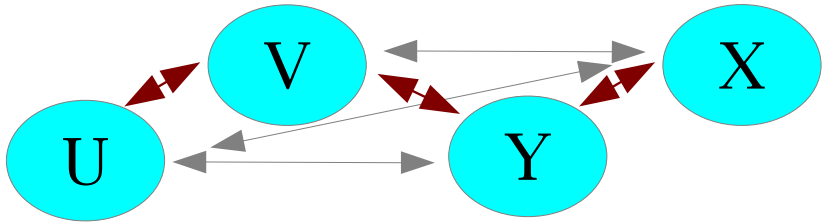
Create when initialising by a trivial way and
update after each new tuple is added.

Initialising Model Skeleton

Each tuple is of one entity type having **n** of attributes.
There are:

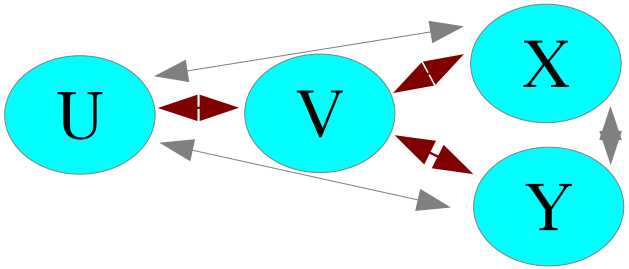
- $\mathbf{2^{(n-1)}}$ of functional dependencies (all covered ones).
- $\mathbf{n^2}$ of functional dependencies in the model (non-trivial).
- **2(n-1)** of functional dependencies in the model skeleton.

After the first tuple is inserted into a storage,
all attribute value can be derived from any other one.



Linear model skeleton

Max cycle length: **2(n-1)**
No. of combinations: **n!**



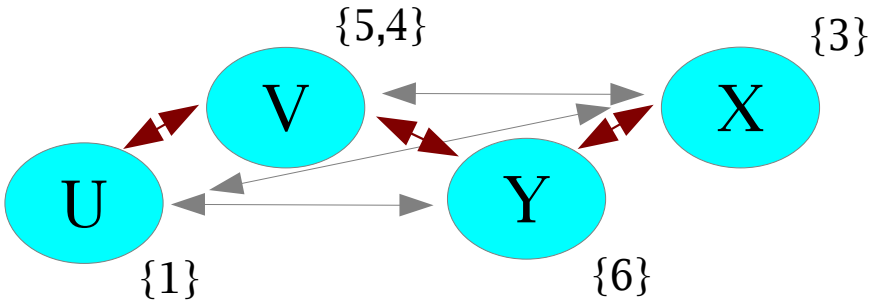
Star model skeleton

Max cycle length: **4**
No. of combinations: **n**

All sets being a model skeleton must be a combination of these.

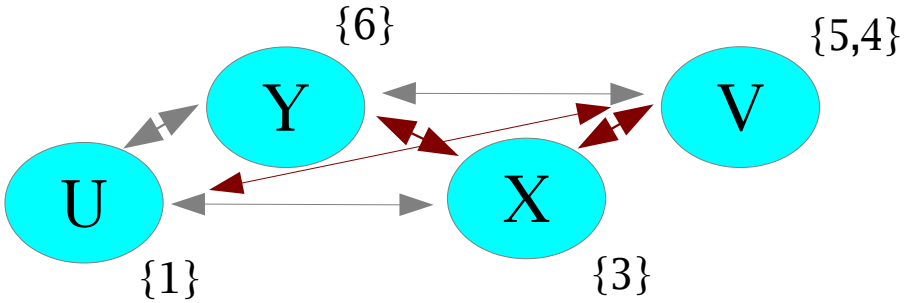
Updating Skeleton – Order Change

The model skeleton contains all functional dependencies between attributes with "*minimal distance*".



The attribute order can be defined as non-degreasing order of active attribute domains.

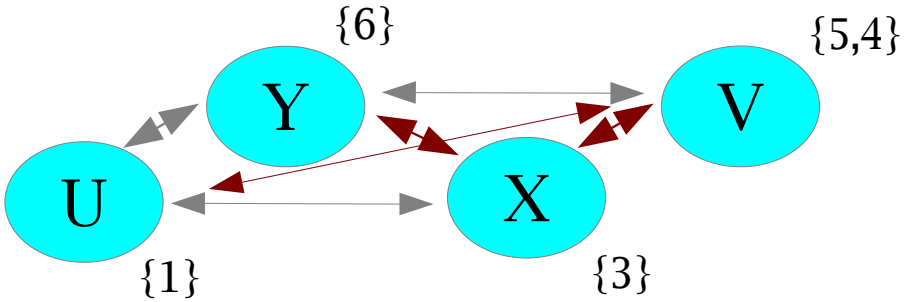
if D(Ai) > D(Aj) then i > j



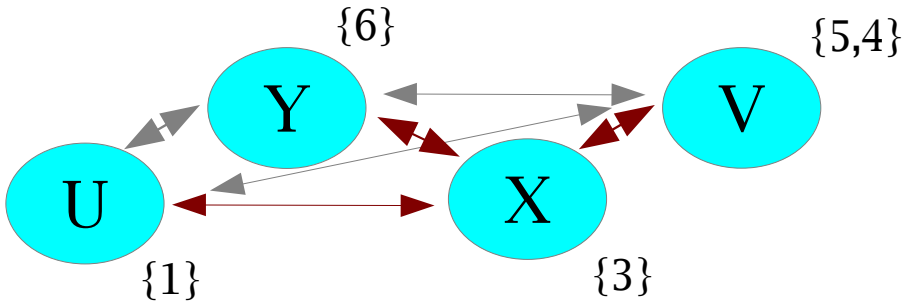
After each new tuple is added, if any attribute active domain size is changed, the model skeleton update is required.

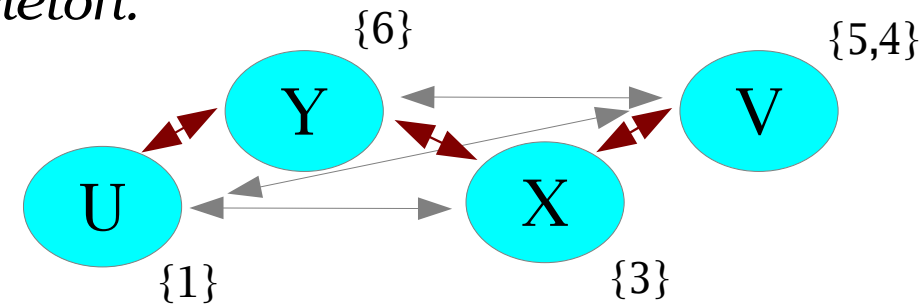
Updating Skeleton – Swapping

Because of the attribute active domain size is changed,
the attributes can be swapped and the skeleton must be updated.



If there exist functional dependencies U->V and X->V in the skeleton and the U->X functional dependency out of the skeleton and a distance U to V is greather than U to X, remove U->V from the skeleton and add U->X into the skeleton.

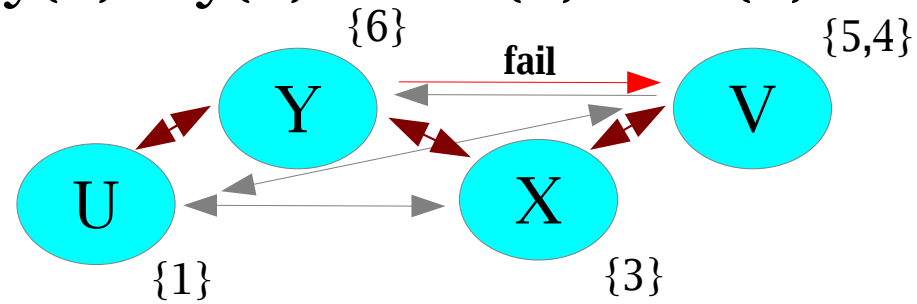




Analogically for U->X , X->Y in and U->Y out in the second step.
Analogically for mutally functional dependencies.

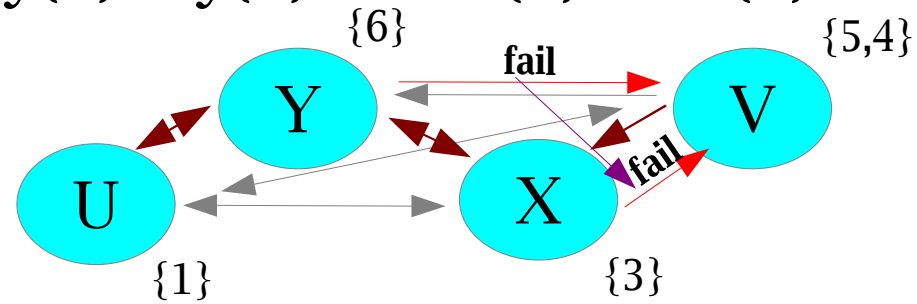
Functional Dependency Corruption

The next tuple corrupts the **Y->V** extensional functional dependency, ie. y(1) = y(2) and v(1) <> v(2).



Functional Dependency Corruption

The next tuple corrupts the **Y->V** extensional functional dependency, ie. y(1) = y(2) and v(1) <> v(2).

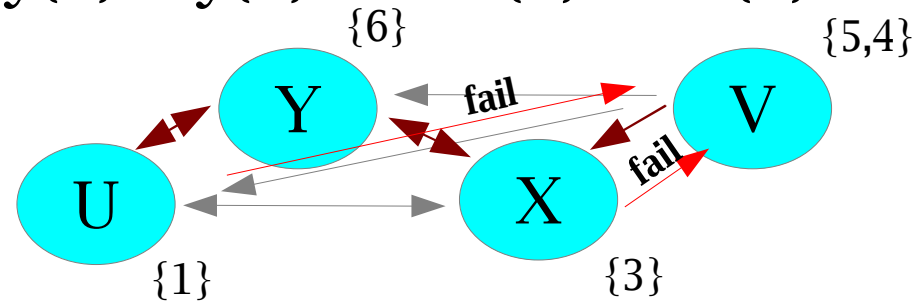


If any functional dependency excluded the skeleton is corrupted, some functional dependency included in the skeleton must be corrupted.

After each new tuple is added,
all functional dependencies in the skeleton must be tested.

Functional Dependency Corruption

The next tuple corrupts the **Y->V** extensional functional dependency, ie. y(1) = y(2) and v(1) <> v(2).



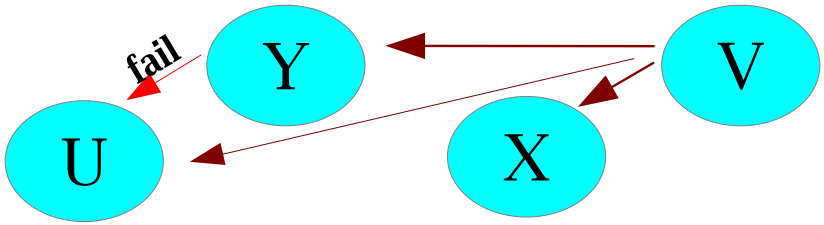
If any functional dependency excluded the skeleton is corrupted, some functional dependency included in the skeleton must be corrupted.

If any functional dependency is corrupted in the model skeleton, all functional dependencies with the same left or right side must be tested. There are given into model skeleton and the test is recursively called.

After each new tuple is added,
all functional dependencies in the skeleton must be tested.

Complex Attribute Creation

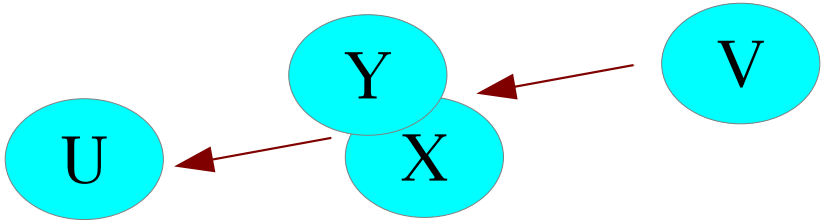
When the “single” functional dependencies are corrupted, there may exist complex attribute.



If there exist m attributes with no dependency between, no dependency to U and one of them Y->U is being corrupted, there may exist a complex attribute (with arity less m), which satisfies the {X,Y}->U functional dependency.

Complex Attribute Creation

When the “single” functional dependencies are corrupted, there may exist complex attribute.



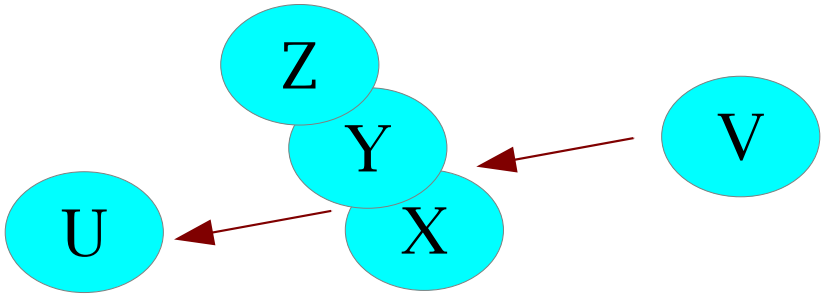
If there exist m attributes with no dependency between, no dependency to U and one of them Y->U is being corrupted, there may exist a complex attribute (with arity less m), which satisfies the {X,Y}->U functional dependency.

This complex attribute is added as the virtual attribute (representing a Cartesian product of all single attributes)

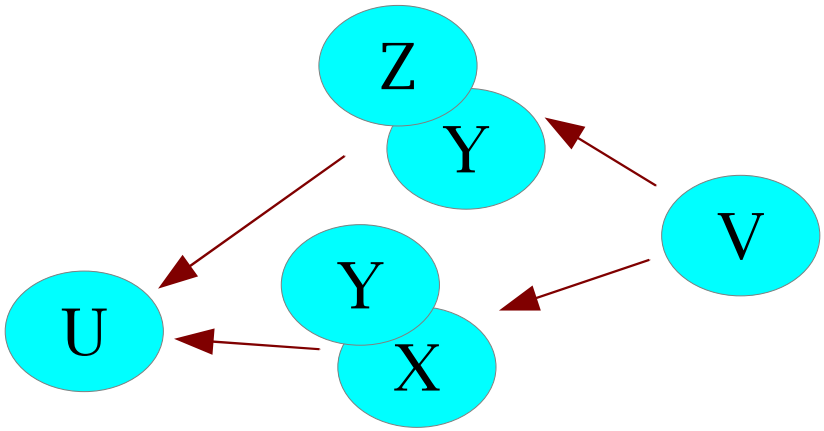
The new virtual attribute values are derived from the V attribute value. It is why the unique identificator of tuple must be presented in the model.

Complex Attribute Creation

Finding of the new complex attribute non-trivial dependency
a NP issue for m>2



For testing of a complex attribute of arity m, there is
$m! / (m/2)!^2$
number of possible combinations of attributes.
Only non-trivial are used to be considered.

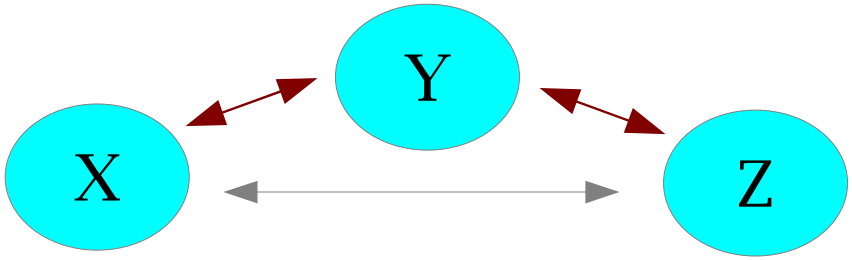


This makes the model estimation process NP complex issue.

Model Monotonicity

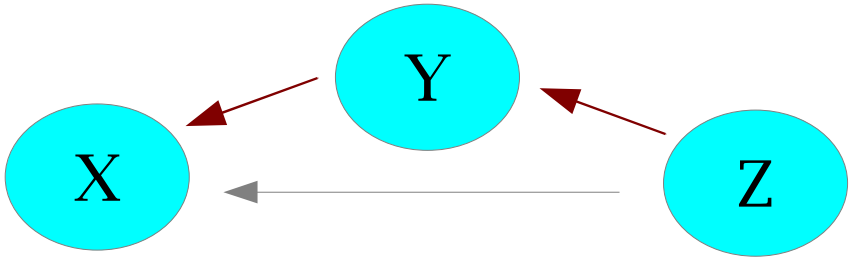
The model estimation process is monotonic.

The initial model M_0 contains n! of functional dependencies.



Several functional dependencies are corrupted during the estimation process.

A number of functional dependencies is down going.



The models can be ordered by a number of functional dependencies.

After i-th tuple:

$$M_0 \geq M_i \geq M_S \geq M_\infty$$

Principle Limitation

Why can the inducted model be different to the logical one?

Reason:

- *Data Representativeness*
  (classical issue of machine learning methods)

- *Source Limitations*
  (data or a structure may depend on their source)

- *Granuality / Uncountable Set*
  (a finite number of tuples versus infinite domains)

- *Extensional functional dependency meaning*
  (it is why to speak about an estimation)

ILP and Data Structure Estimation

Data Structure Estimation

Question:

On which **attributes** depend a **given attribute value**?

Similar questions, similar issues?

Inductive Logic Programming

Question:

From which **predicates** can be inducted a **given predicate**?

Inductive Logic Programming

Inductive logic programming system
induces logic programs
as hypotheses from ground facts.

Inductive logic programming

Question:

From which **predicates** can be inducted a **given predicate**?

GOLEM

Using a generalisation principle.

(relative least general generalisation)

1) Transform literals into their models:

parent(Tom,Marie} *into* *Parent({Tom},{Marie})*

2) Join the same name models as general as possible with not to cover any negative example:

Parent({Tom},{Marie}), Parent({Joe},{Alice}), Parent({Tom},{Eve})

into

Parent({Tom,Joe},{Alice,Eve})

3) Join models with the same constant sets:

Parent({Tom,Joe},{Alice,Eve}), Female({Alice,Eve})

into

Parent({Tom,Joe},{Alice,Eve}) and *Female({Alice,Eve})*

4) Transform back model into clauses.

FOIL

Using a specialisation principle.

Find induction of the P predicate:
While positive example set is not empty
1) Find a new body Q
a) Give all training set examples into Q related set
b) While Q covers any negative example
1) Find literal b well separating negative/positive examples related to Q.
2) Add b into Q.
3) Actualise Q related example set

2) Postprocess P <- Q

3) Add P <- Q into a hypothese set

4) Remove all positive examples covered by Q from a training set

PROGOL

Find induction of the P predicate:

1) Find the most specific clause for each training example.

2) Apply a refinement operator on these clauses.

3) Search the subsumption lattice.

Data Structure Estimation as ILP

Find induction of the P predicate:

1) Transform knowledge base into a table (propositional form)
The variables and predicates correspond to attributes
The literal present in a base corresponds to **true**, *else* **false** *values.*

2) Estimate a given table data structure.

3) Transform back a functional dependency system into clauses.

Data Structure Estimation as ILP (step 1)

1) Transform knowledge base into a table (propositional form)

The variables and predicates correspond to attributes

The literal present in a base corresponds to **true**, *else* **false** *values.*

Row number
Variables

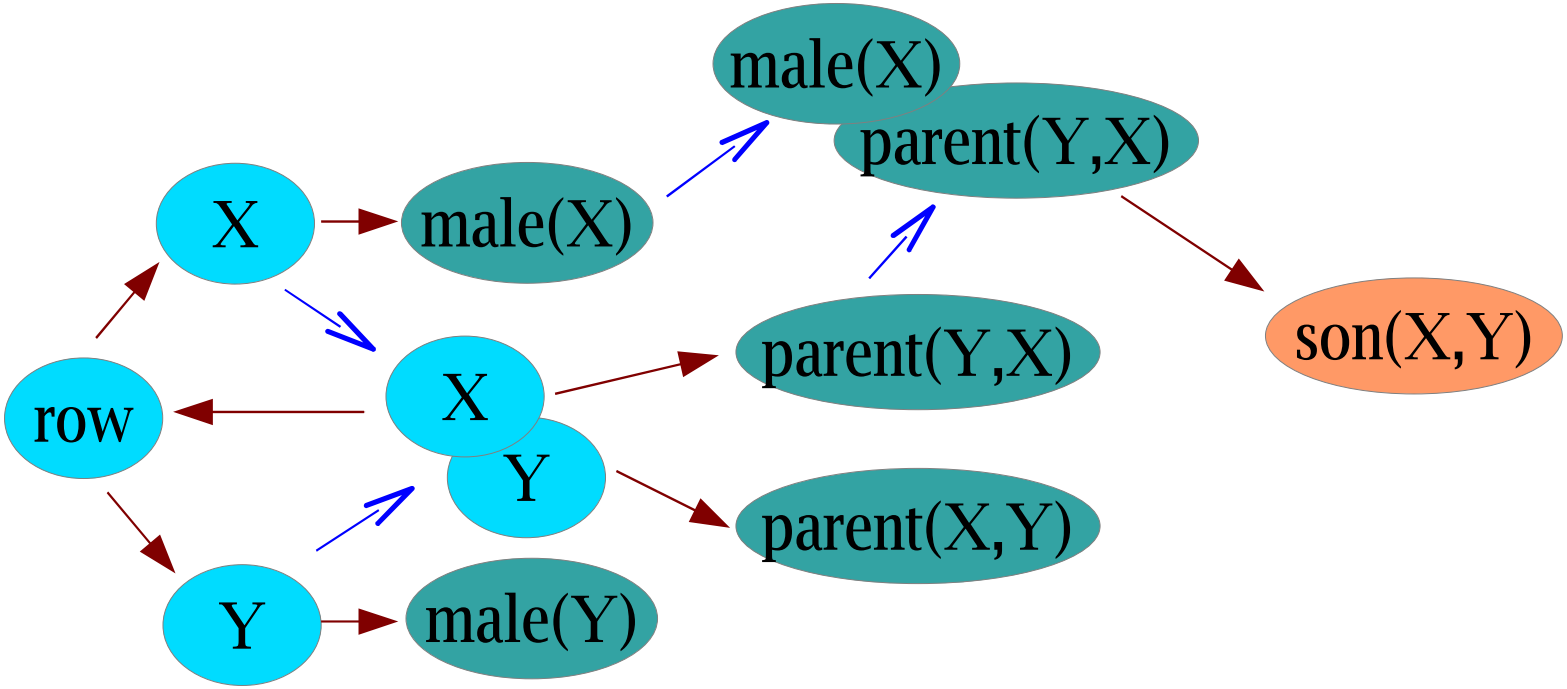
Literals in
the knowledge base

Predicate
to be searched

ROW	X	Y	male(X)	male(Y)	parent(X,Y)	parent(Y,X)	son(X,Y)
1	Tom	Joe	true	true	false	true	true
2	Tom	Alice	true	false	false	true	false
3	Eve	Joe	false	true	false	true	false
4	Alice	Eve	false	false	true	false	false
…	…	…	…	…	…	…	false

Data Structure Estimation as ILP (step 2)

2) Estimate a given table data structure.

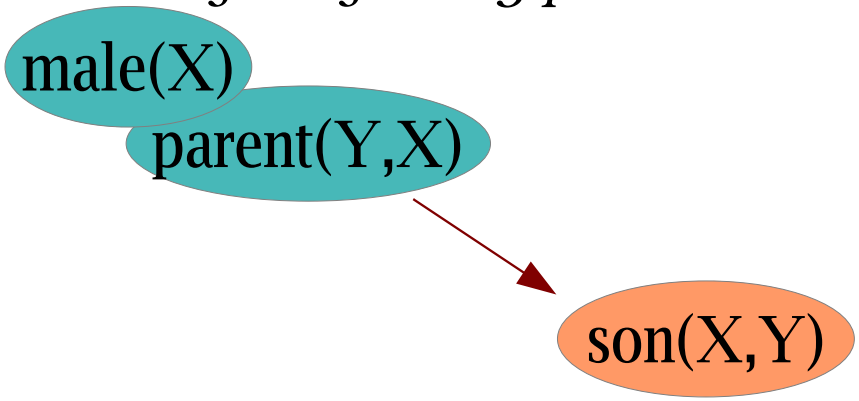


ROW	X	Y	male(X)	male(Y)	parent(X,Y)	parent(Y,X)	son(X,Y)
1	Tom	Joe	true	true	false	true	true
2	Tom	Alice	true	false	false	true	false
3	Eve	Joe	false	true	false	true	false
4	Alice	Eve	false	false	true	false	false
…	…	…	…	…	…	…	false

Data Structure Estimation as ILP (step 3)

3) Transform back the functional dependency system into clauses.

a) Remove all dependencies out of the finding predicate attribute.

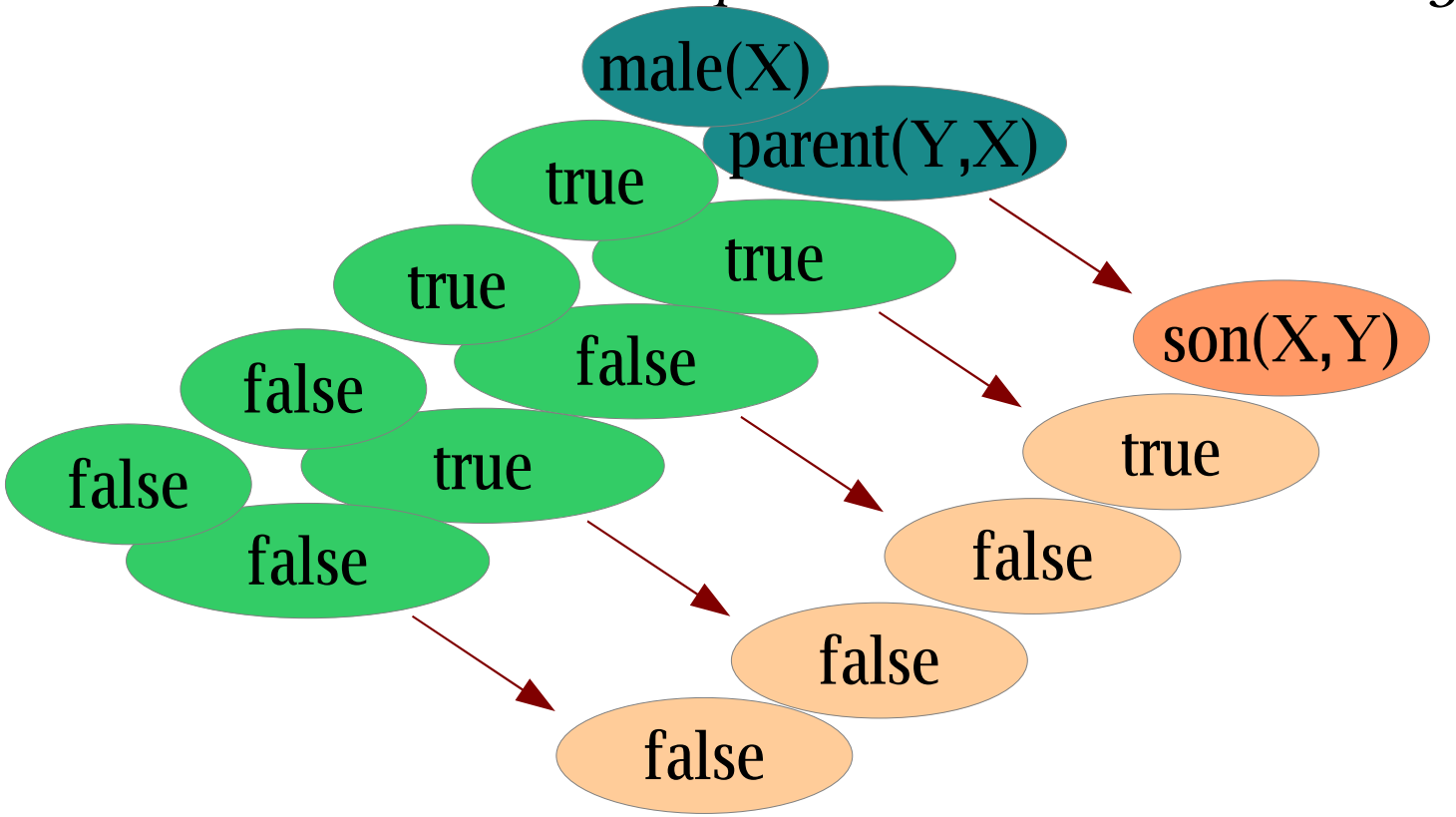


ROW	X	Y	male(X)	male(Y)	parent(X,Y)	parent(Y,X)	son(X,Y)
1	Tom	Joe	true	true	false	true	true
2	Tom	Alice	true	false	false	true	false
3	Eve	Joe	false	true	false	true	false
4	Alice	Eve	false	false	true	false	false
...	...	...	...	...	...	...	false

Data Structure Estimation as ILP (step 3)

3) Transform back the functional dependency system into clauses.

b) Select instances which has positive valued the searching predicate.

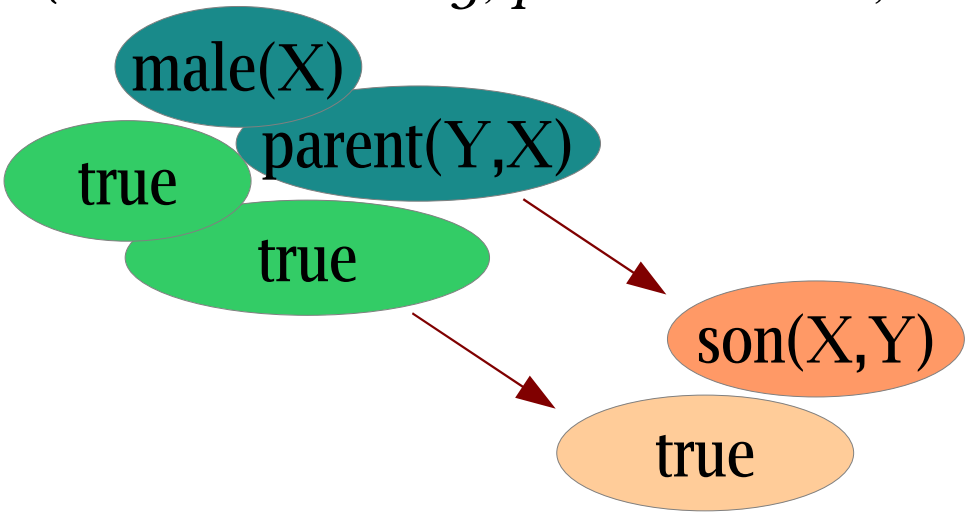


ROW	X	Y	male(X)	male(Y)	parent(X,Y)	parent(Y,X)	son(X,Y)
1	Tom	Joe	true	true	false	true	true
2	Tom	Alice	true	false	false	true	false
3	Eve	Joe	false	true	false	true	false
4	Alice	Eve	false	false	true	false	false
…	…	…	…	…	…	…	false

Data Structure Estimation as ILP (step 3)

3) Transform back the functional dependency system into clauses.

*c) Reconstruct clause. (**true** in the body, positive literal, else negative)*



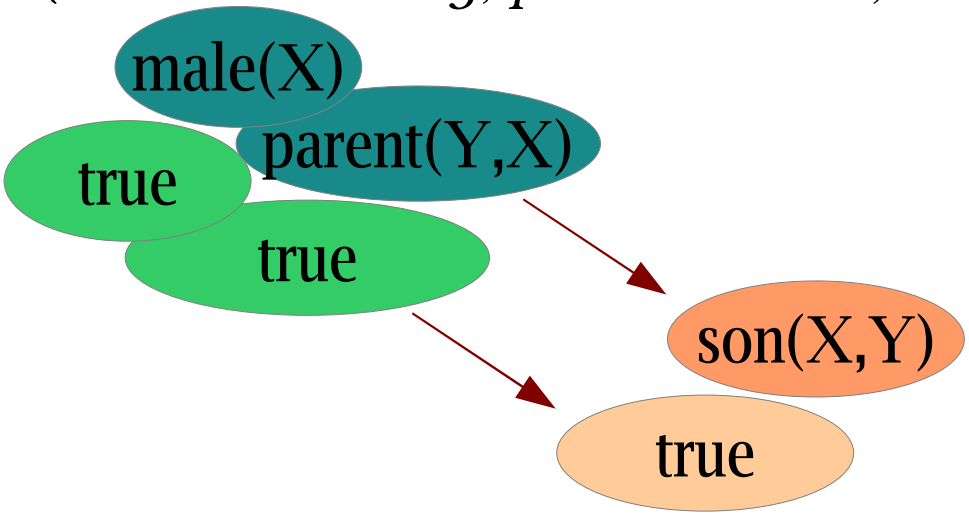
son(X,Y) <- male(X), parent(Y,X)

ROW	X	Y	male(X)	male(Y)	parent(X,Y)	parent(Y,X)	son(X,Y)
1	Tom	Joe	true	true	false	true	true
2	Tom	Alice	true	false	false	true	false
3	Eve	Joe	false	true	false	true	false
4	Alice	Eve	false	false	true	false	false
…	…	…	…	…	…	…	false

Data Structure Estimation as ILP (step 3)

3) Transform back the functional dependency system into clauses.

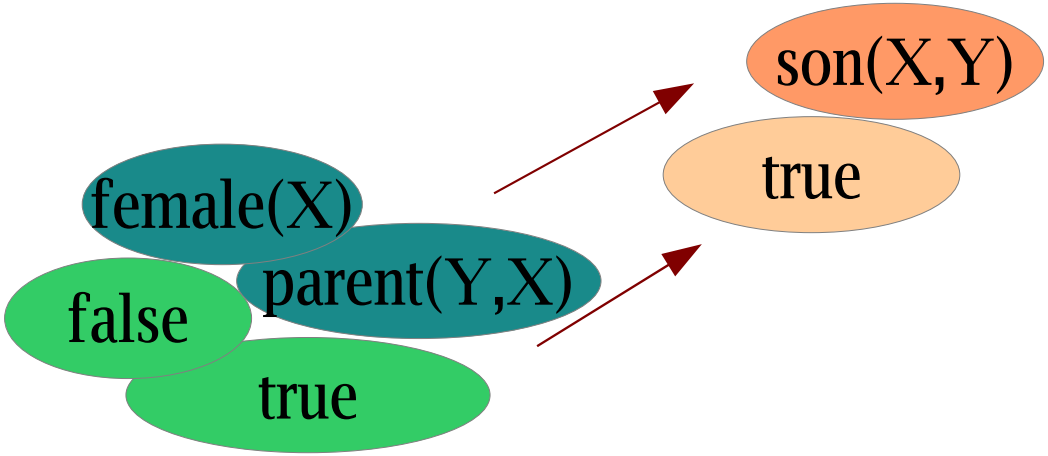
*c) Reconstruct clause. (**true** in the body, positive literal, else negative)*



son(X,Y) <- male(X), parent(Y,X)

Another example:

son(X,Y) <- not(female(X)), parent(Y,X)



Advantages/Disadvantages

Only basic variant is mentioned:

- *no recursion, no non-deterministic values, free-function clauses.*

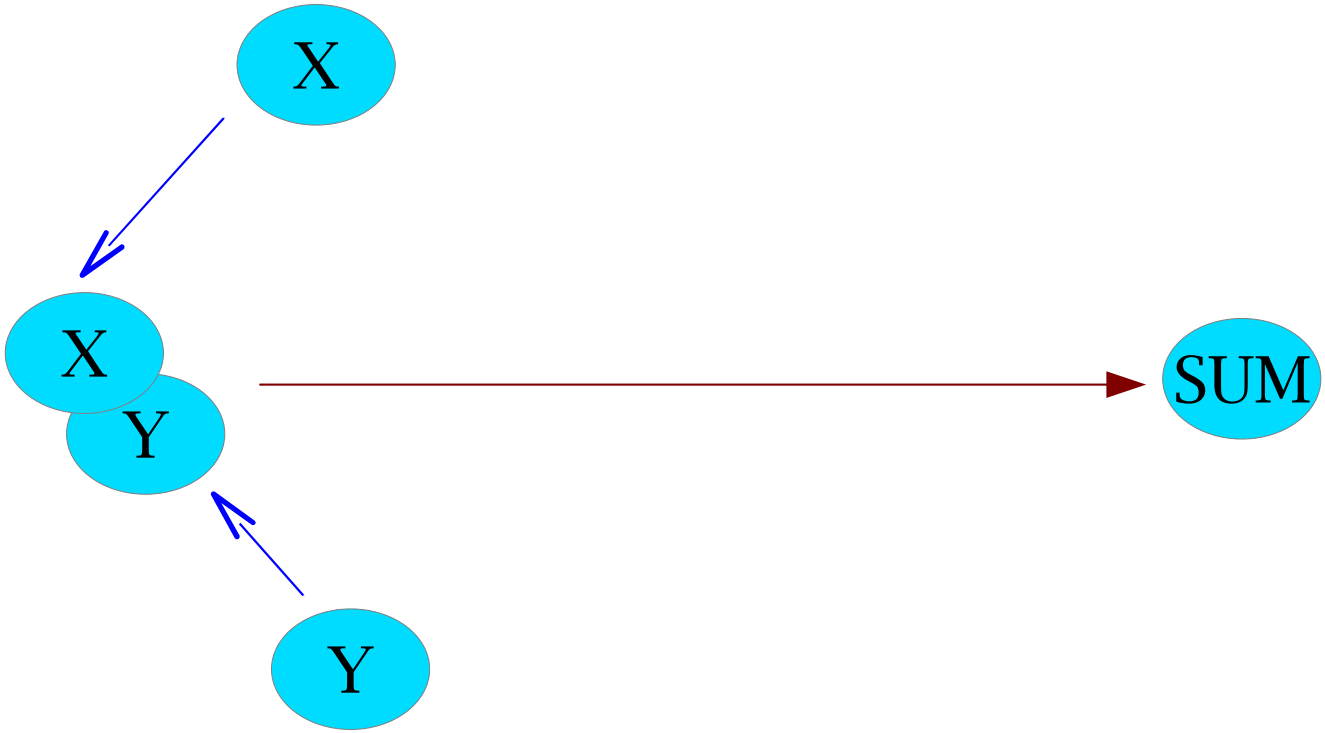
Advantages

- *A set of predicates can be inducted by one method calling.*
- *Effective pruning of possible hypothesse state space.*
- *No dependency on a training set example order (Progol).*
- *All (paralel) hypotheses are given.*

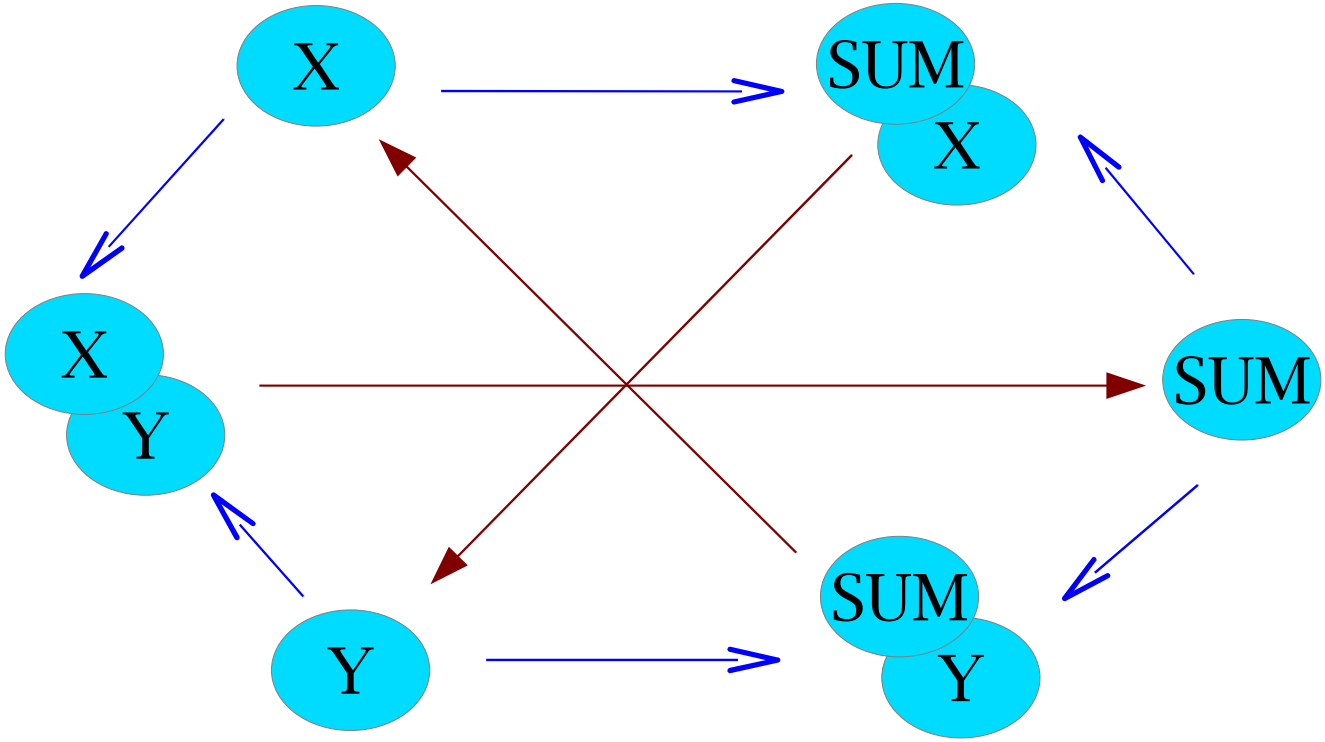
Disadvantages

- *Propositional form required.*
- *Only "crisp" hypotheses – no noise handling support.*

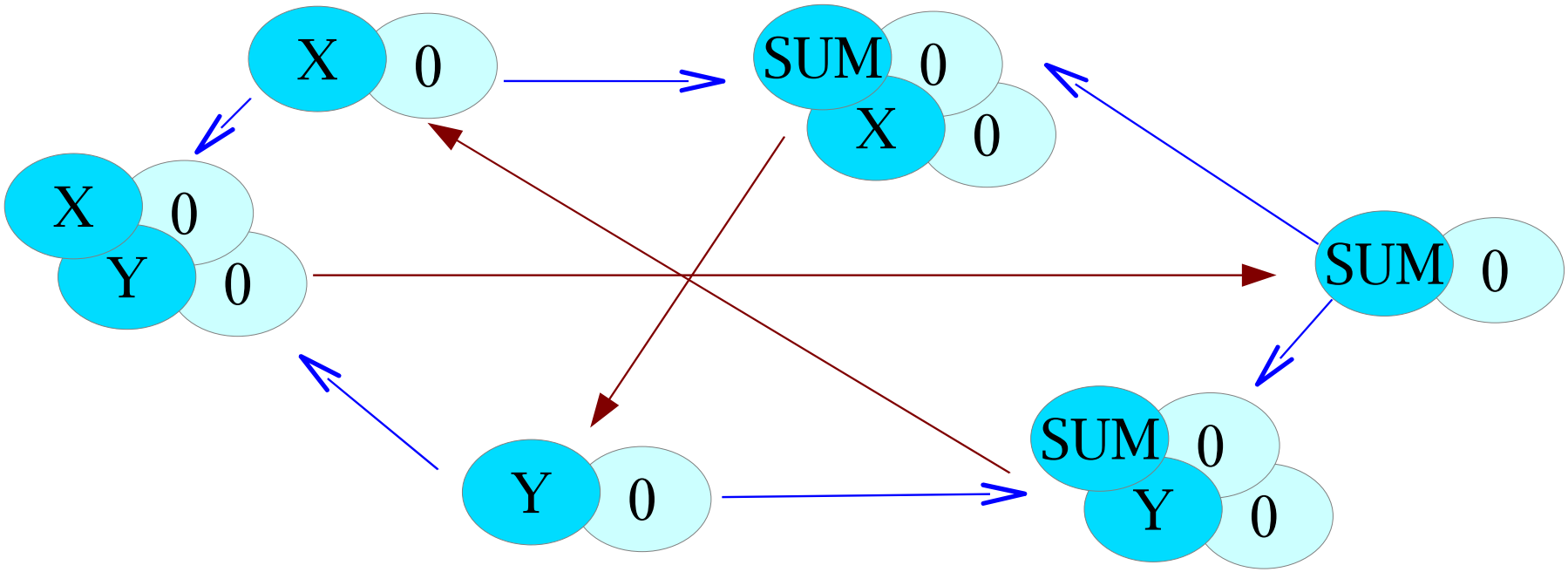
Next Example – X + Y = SUM



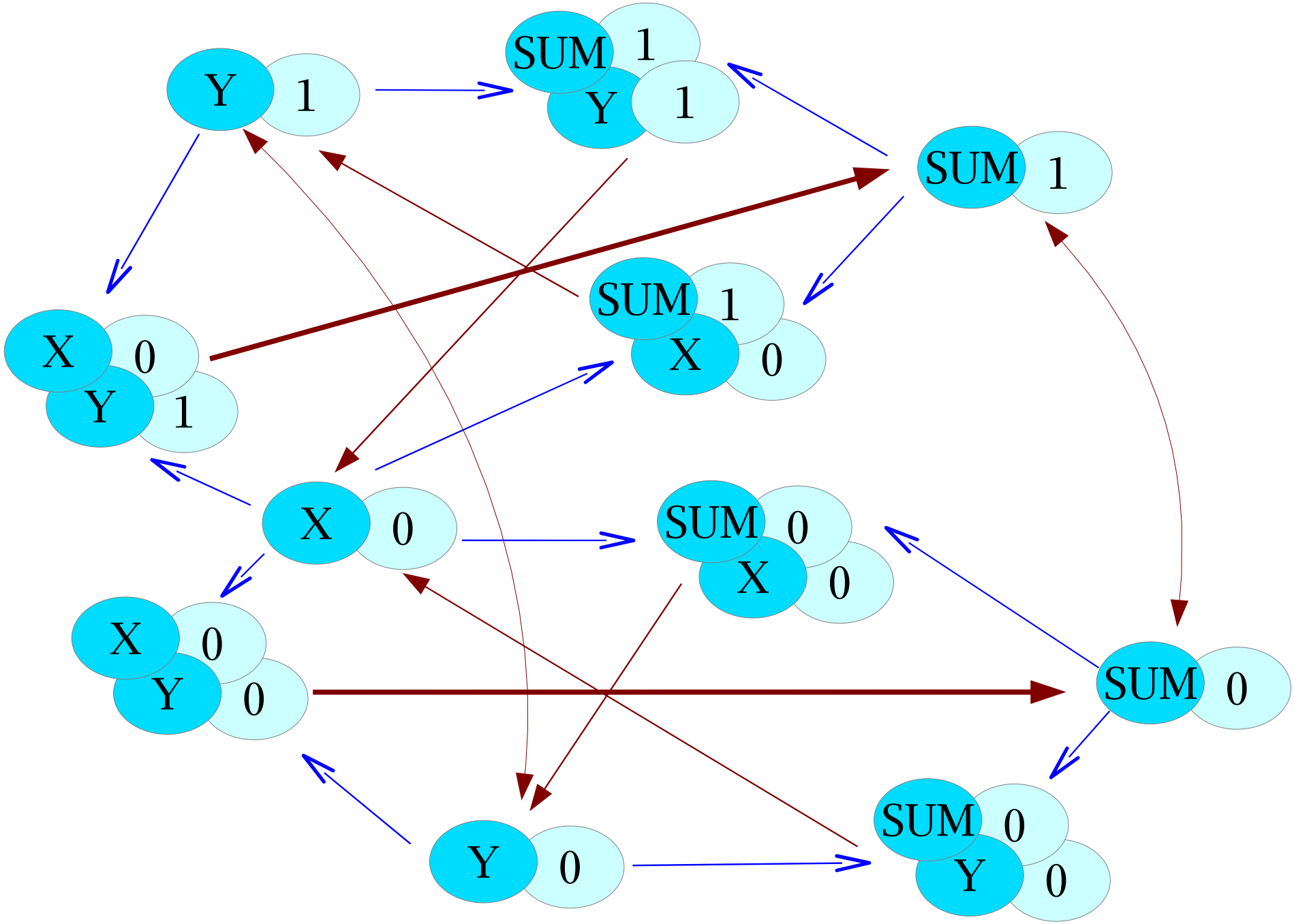
Next Example - Note – All is a Cycle



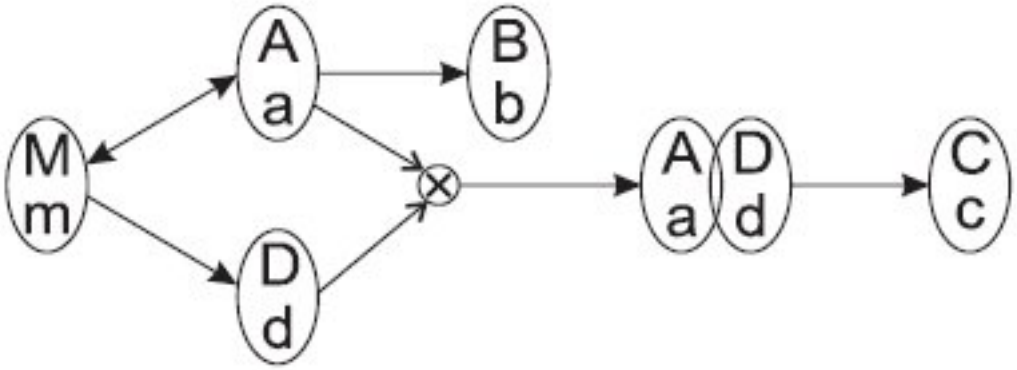
To do – Recursive Definitions



To do – Recursive Definitions



Interpreting Skeleton



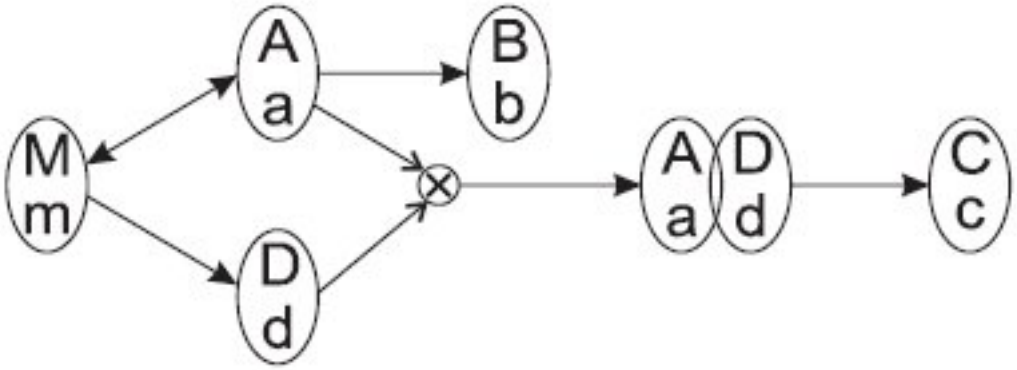
Interpretation as:

- Relational Schema

- Semantic Web

  - Basic Element – (attribute – value) pair
  - RDF / OWL variant

Interpreting Skeleton – Relations



Transformation Rule

$$(A_i \to A_j) \in S \wedge (A_i \to A_k) \in S \rightsquigarrow \begin{array}{l}(A_i \to A_j) \notin S' \wedge (A_i \to A_k) \notin S' \\ (A_i \to \{A_j, A_k\}) \in S'\end{array}$$

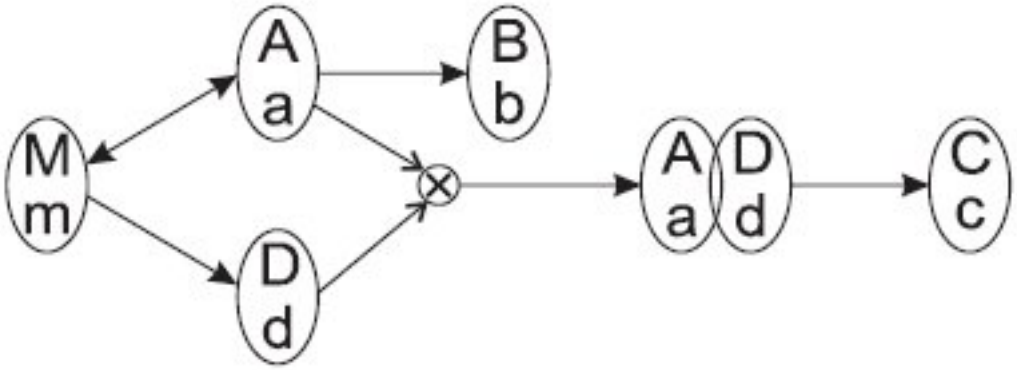
Result

$$\begin{array}{lr|l} table_M & \{M^{PK} : A, D\} & \{m : a, d\} \\ table_A & \{A^{PK} : M, B\} & \{a : m, b\} \\ table_{AD} & \{A^{PK}, D^{PK} : C\} & \{a, d : c\} \end{array}$$

Back Transformation Rule

$$(A_i \to \{A_j, A_k\}) \in S' \rightsquigarrow (A_i \to A_j) \in S \wedge (A_i \to A_k) \in S$$

Interpreting Skeleton - RDF



Transformation Rule

$$(A_i \to A_j) \in S \wedge (A_i \to A_k) \in S \rightsquigarrow \begin{matrix}(A_i \to A_j) \notin S' \wedge (A_i \to A_k) \notin S' \\ (A_i \to \{A_j, A_k\}) \in S'\end{matrix}$$

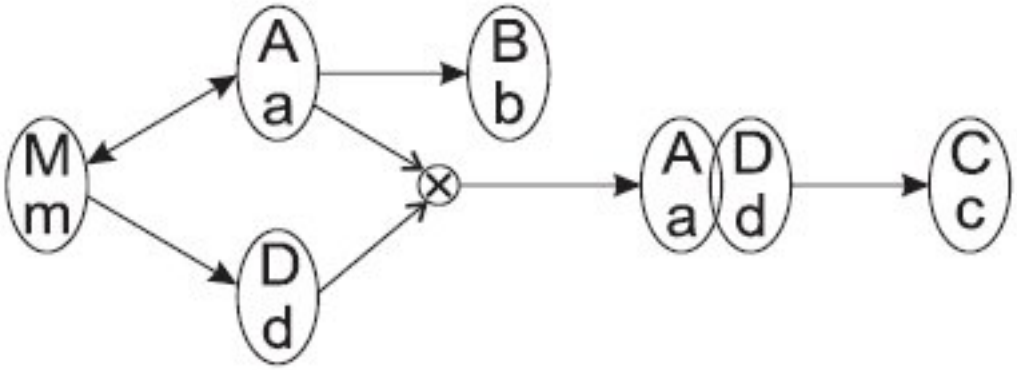
Result

```
<dse:M   rdf:about="m"   dse:A="a" dse:D="d" />
<dse:A   rdf:about="a"   dse:B="b" dse:M="m" />
<dse:A-D rdf:about="a-d" dse:A="a" dse:D="d" dse:C="c" />
```

Back Transformation Rule

$$(A_i \to \{A_j, A_k\}) \in S' \rightsquigarrow (A_i \to A_j) \in S \wedge (A_i \to A_k) \in S$$

Interpreting Skeleton - OWL



Transformation Rule

$$(A_i \to A_j) \in S \wedge (a_i \to a_j) \in \mathscr{I}_{A_i \to A_j} \rightsquigarrow A_i^{a_i} \sqsubseteq A_j^{a_j}$$

Result

$$\begin{array}{ll} A^a \sqsubseteq M^m & M^m \sqsubseteq A^a \\ D^d \sqsubseteq M^m & B^b \sqsubseteq A^a \\ C^c \sqsubseteq A^a \sqcap D^d & M^m \sqsubseteq \{\langle \mathfrak{m} \rangle\} \end{array}$$

Back Transformation Rule

$$A_i^{a_i} \sqsubseteq A_j^{a_j} \rightsquigarrow \begin{array}{l} (A_i \to A_j) \in S \\ a_i \in \mathscr{D}_\alpha(A_i),\ a_j \in \mathscr{D}_\alpha(A_j) \wedge (a_i \to a_j) \in \mathscr{I}_{A_i \to A_j} \end{array}$$

RDF Oriented Data Repository

Using binary matrices

- design as simple as possible

- searching the repository ~ RDF reasoners

- training the repository ~ ?

    - data structure estimation

Repository – Matrix Definition

Repository *R*: $r_{ij} = 1$ if element *i* implies element *j*

$r_{ij} = 0$ otherwise

Dependency System *S*: $s_{ij} = 1$ if attribute *i* implies *j*

$s_{ij} = 0$ otherwise

Domain *D*: $d_{ij} = 1$ if element *i* corresponds to attribute j

$d_{ij} = 0$ otherwise

Repository – Generalisation

- search for facts (elements) valid for objects satisfying the query x_0

- Generalisation algorithm:

$$\text{while}\left(x_{k+1} \neq x_k\right)$$
$$x_{k+1} = R \cdot x_k$$

- **Reached in**

max |A|+1 steps

Repository – Specialisation

- search for objects satisfying the query x_o

- Specialisation algorithm:

$$\text{while}\left(x_{k+1} \neq x_k\right)$$
$$x_{k+1} = R^T \cdot x_k$$

- **Issue**

  - several objects may be returned -> explode the vector
  - allow activation of elements corresponding to new attributes only

Repository – Training

- training the repository for information x_k
- **Training algorithm**:
  - verify domain matrix D
  - update functional dependency system matrix

$$S'_{k+1} = S_k + (1 - \overline{S_k}) \circ [(D_{k+1} \cdot x_k)^T \cdot (D_{k+1} \cdot x_k)]$$

  - update repository matrix

$$R'_{k+1} = R_k + (D^T_{k+1} \cdot S'_{k+1} \cdot D_{k+1}) \circ (x^T_k \cdot x_k)$$

  - check corrupted functional dependencies

$$Z = D_{k+1} \cdot ((R^T \cdot D^T_{k+1}) > 1)$$

$$S_{k+1} = S'_{k+1} - Z \qquad \overline{S'_{k+1}} = \overline{S_{k+1}} + Z$$

  - filter repository matrix

$$R_{k+1} = R'_{k+1} \circ (D^T_{k+1} \cdot S_{k+1} \cdot D_{k+1})$$

Repository – Decomposition

- similar to the model skeleton

- **Decomposition algorithm**:

  - to find non-trivial decomposition of

$$R = R' \cdot \ldots \cdot R'$$

  - reduced items, but not complexity
    - sparse matrices?

  - not unique solution
    - minimalising the expressiveness criterion

Conclusion

Data Structure Estimation

- polynomial complexity algorithm for simple fds
- nonpolynomial complexity if complex attribute support
  - a complex attribute decomposition

- Incremental algorithm proposed

The aim:

- to process current web sources by semantic web tools with the best approximation of description.

The binary matrix notion

- discover (meta)relationships generated by data and back effect of (meta)relationships on the stored data.