CLP(*BN*)

Bayesian networks as constraints in logic programming

Outline

- Motivation
- Introduction
- Example
- Syntax & Semantics of a CLP(*BN*) program
  - 2 views: operational (query resolution) and model-theoretic
- Examples for how to make use of logic programming
  - Non-deterministic aggregates
  - Recursion
- PRM $\subset$ CLP(*BN*)
- Learning (the ILP style)
- References

Motivation

- Relational DB's foundations lie in FOL
  - e. g. $\forall x \; \exists y$ Registration(x) $\rightarrow$ Registration_Grade(x,y)
- Problem: Relational DBs with unknown fields
- Our goal:
  - Estimate probabilities for possible values (like in PRM)
  - Make this information accessible in logic programs

Introduction

- Skolemization of existentially quantified fields (non-key data)
  - e. g. ∀x Registration(x) → Registration_Grade(x,*grade*(x))
- In CLP(*BN*) Skolem functions are represented as Skolem terms
- CLP(*BN*) expresses probabilities over these Skolem terms as constraints
- It's still a logic program → clauses

Example: DB

- Black attributes in italics are keys (arbitrary unique IDs)
- Attributes with blue links are foreign keys
- Red attributes are random variables (RVs) (= mostly unknown fields)
- Arcs express dependencies between these RVs and therefore form a Bayesian net

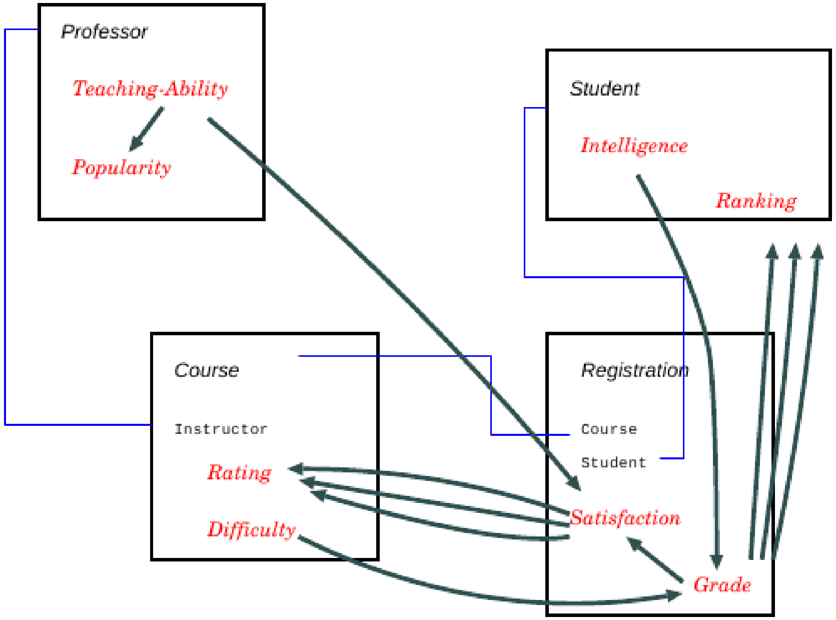
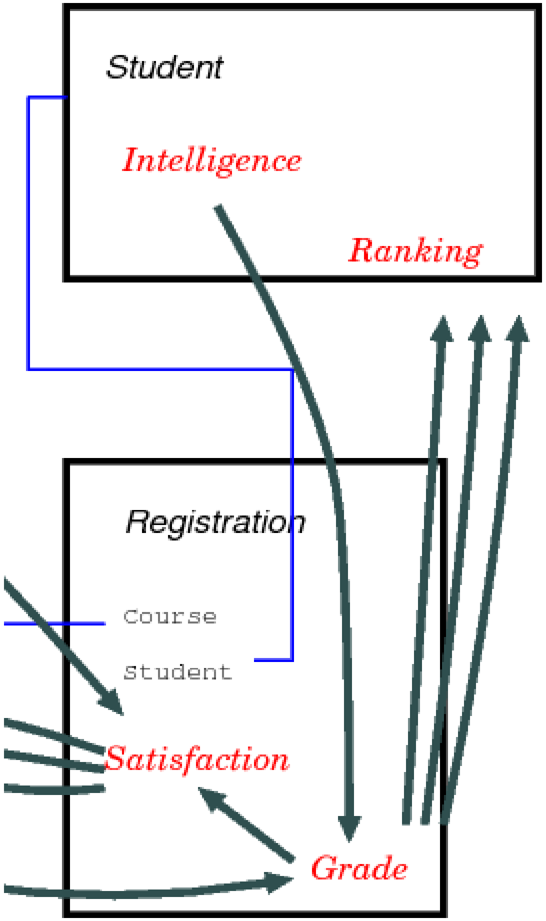


Figure 1: The School Database

Example: CLP(*BN*) clauses

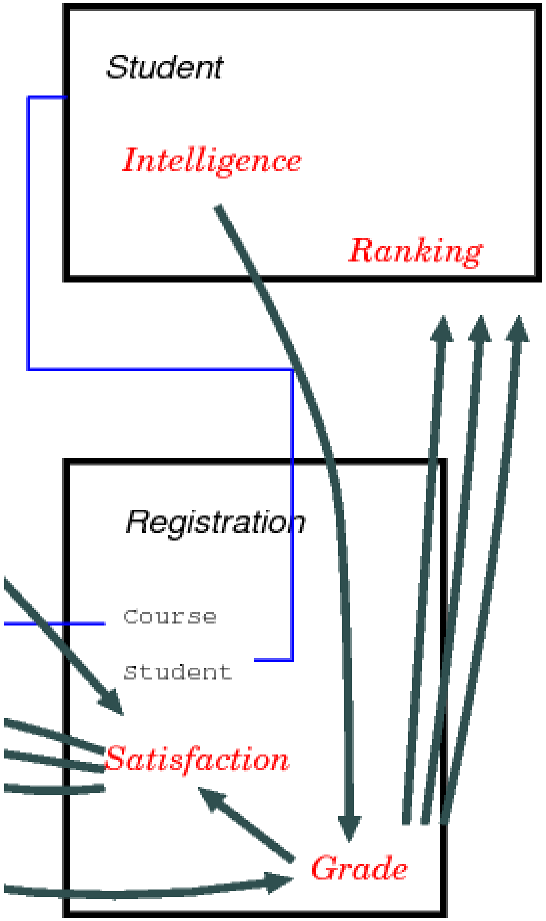
```
intelligence(S,Int) :-
 {Int = i(S) with p([h,l],[0.7,0.3],[])}.
```



Example: CLP(*BN*) clauses

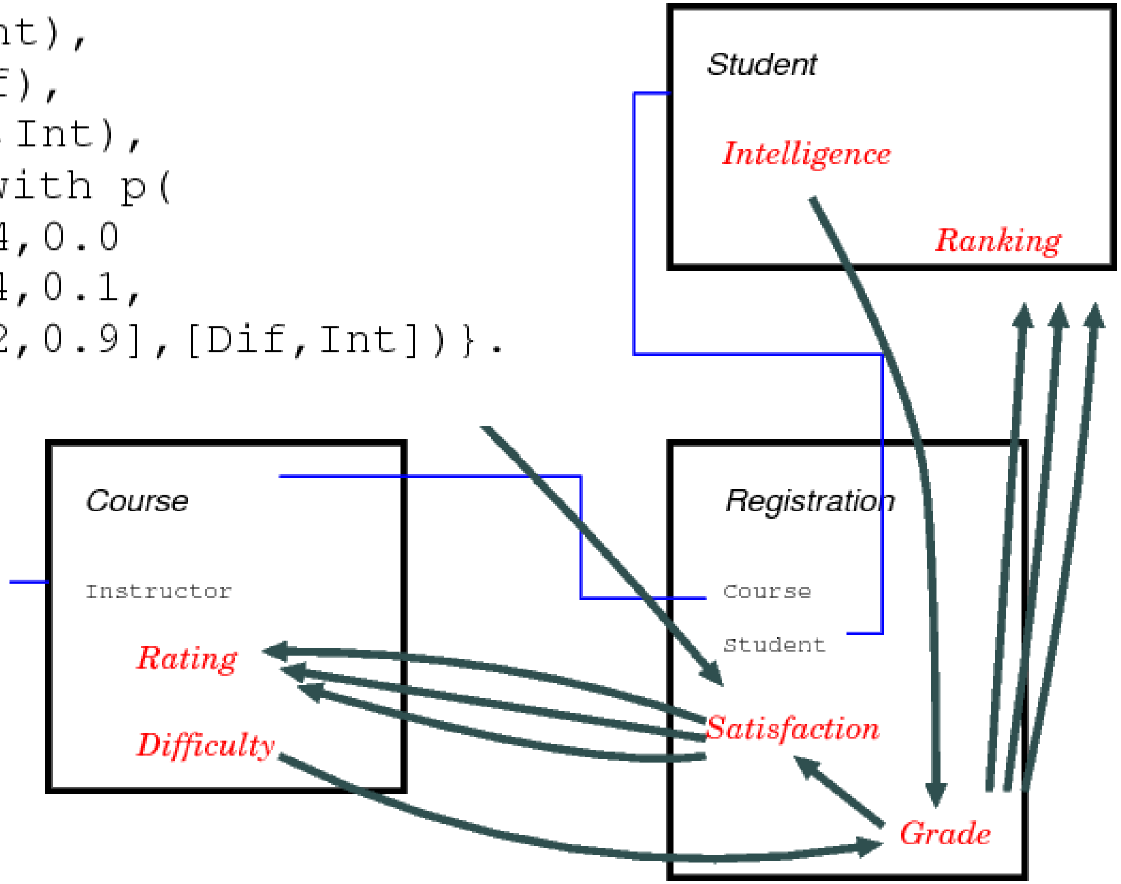
```
intelligence(S,Int) :-
  int_table(S, Dist),
  {Int = i(S) with p([h,l],Dist,[])}.

int_table(bob, [0.3, 0.9]) :- !.
int_table(mike, [0.8, 0.2]) :- !.
int_table( _, [0.7,0.3]).
```



Example: CLP(*BN*) clauses

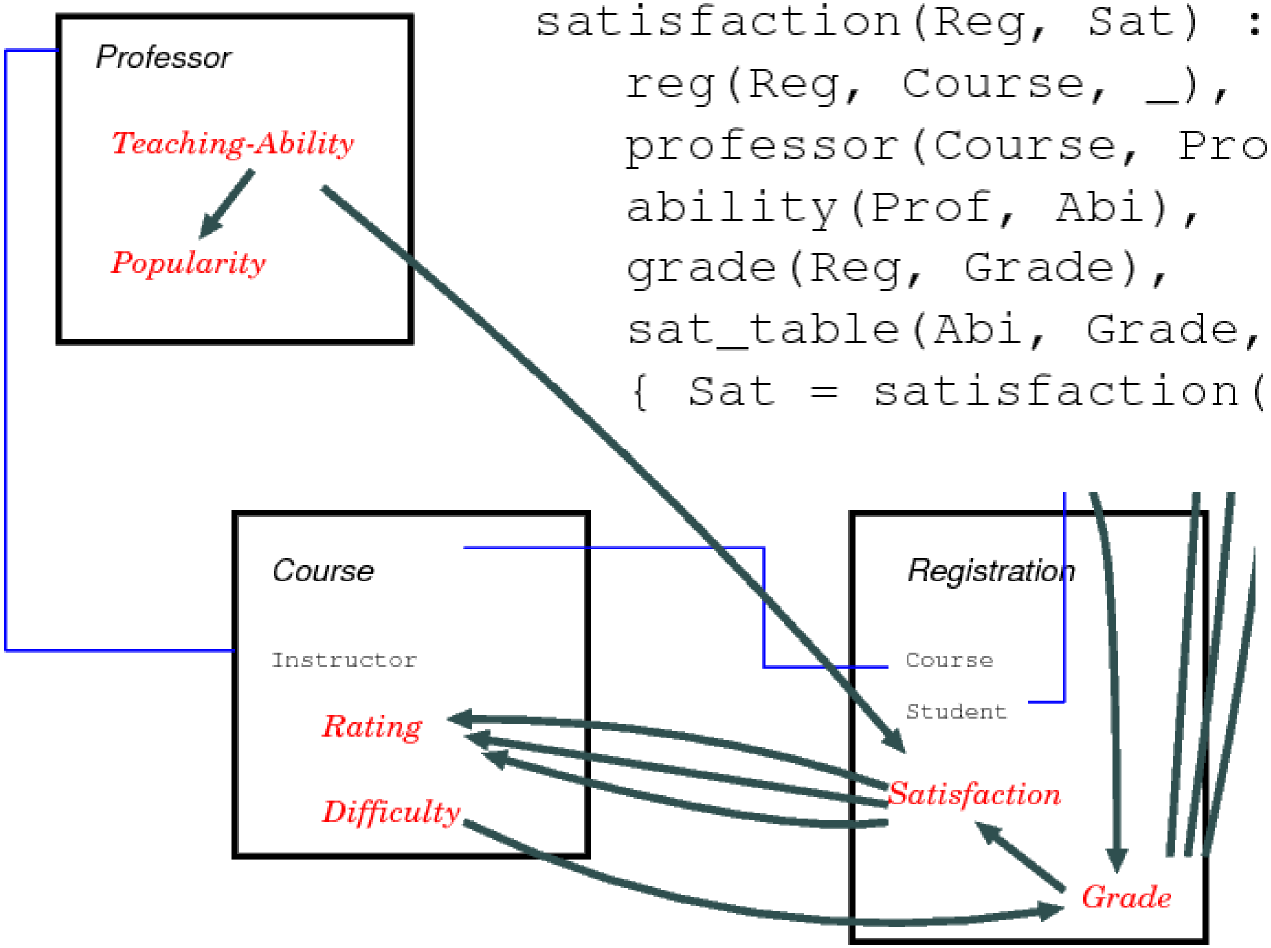
```
grade(Reg, Grade) :-
  reg(Reg,Course,Student),
  difficulty(Course,Dif),
  intelligence(Student,Int),
  {Grade = grade(Reg) with p(
   [a,b,c],[0.4,0.9,0.4,0.0
            0.4,0.1,0.4,0.1,
            0.2,0.0,0.2,0.9],[Dif,Int])}.
```



Example: CLP(*BN*) clauses

TECHNISCHE UNIVERSITÄT DARMSTADT

```
satisfaction(Reg, Sat) :-
    reg(Reg, Course, _),
    professor(Course, Prof),
    ability(Prof, Abi),
    grade(Reg, Grade),
    sat_table(Abi, Grade, Table),
    { Sat = satisfaction(Reg) with Table }.
```



Example: Queries to RVs

```
?- grade(r2,Grade).
```

Step	*Bindings*	*Skolem Terms*
0	$\{R = \mathtt{r2}\}$	$\{\}$
1	$\cup\{S = \mathtt{s1}, \mathtt{C} = \mathtt{c3}\}$	
2		$\cup\{D(\mathtt{c3})\}$
3		$\cup\{I(\mathtt{s1})\}$
4		$\cup\{G(r2, D(\mathtt{c3}), \mathtt{I}(\mathtt{s1}))\}$

```
   ?- registration_grade(r2,Grade).
p(Grade=a)=0.4115,
p(Grade=b)=0.356,
p(Grade=c)=0.16575,
p(Grade=d)=0.06675 ?
yes
   ?-
```

Example: Queries with evidence


```
?- grade(r2,X), satisfaction(r2,h).
```

Step	*Bindings*	*Skolem Terms*
5	$\{C' = \mathrm{c2}\}$	
6	$\cup\{P' = \mathrm{p7}\}$	
7		$\cup\{A(\mathrm{p7})\}$
8 – 12		
13		$\cup\{S(\mathrm{r2}, \mathtt{A}(\mathrm{p7}), \mathtt{G}(\mathrm{r2}, \ldots))\}$
14		$\cup\{S(\mathrm{r2}, \ldots) = \mathrm{h}\}$

```
   ?- registration_grade(r2,Grade), registration_satisfaction(r2,h).
p(Grade=a)=0.533096689359442,
p(Grade=b)=0.322837654892765,
p(Grade=c)=0.114760451955444,
p(Grade=d)=0.0293052037923492 ?
yes
   ?-
```

Example: Evidence only

- Often large databases with lots of observed evidence
- Queries would become very large
- CLP(BN) offers a way to feed in evidence at compile time which is processed at query execution (if needed)
- Grounded Skolem term with empty constraint

```
grade(r2, a) :- {}.
```

Syntax: Definitions

- Alphabet of CLP(BN) is the alphabet of logic programming
- Skolem functors := Subset of valid functors
- Skolem term := term whose primary functor is a Skolem functor (gained during process of Skolemization)
- Skolem functor Sk/n → its Skolem term has the form $Sk(W_1,\ldots,W_n)$
- CLP(BN) program := $\{ H_i \leftarrow A_i \,/\, B_i \mid 1 \leq i \leq N \}$
  - H_i and A_i as in Prolog (=: logical portion C_i of clause i)
  - B_i := possibly empty conjunction of { V = Sk with CPT }
    - B_i empty → clause is called a Prolog clause (→ Prolog ⊂ CLP(BN))
    - these B_i are our Bayesian constraints to Variable V
    - CPT is term of the form p(**D**, **T**, **P**)
    - **D** = **D**omain, **T** = Probability **t**able, **P** = **P**arents in BN

Syntax: Well-formed constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

A CLP($\mathcal{BN}$) constraint B_i is well-formed if and only if:

1. all variables in B_i appear in C;
2. $Sk's$ functor is unique in the program; and,
3. there is at least one substitution σ such that $CPT\sigma = \mathbf{p}(\mathbf{D}\sigma, \mathbf{T}\sigma, \mathbf{P}\sigma)$, and **(a)** $D\sigma$ is a ground list, all members of the list are different, and no subterm of a term in the list is a Skolem term; **(b)** $P\sigma$ is a ground list, all members of the list are different, and all members of the list are Skolem terms; and **(c)** $T\sigma$ is a ground list, all members of $T\sigma$ are numbers p such that $0 \leq p \leq 1$, and the size of $T\sigma$ is a multiple of the size of $D\sigma$.

Semantics: Answer queries

- Queries as in Prolog
- Proofs constructed by resolution
- Two clauses may be unified at any step in the proof
- If both these clauses participate in *BN* constraints, unify the corresponding nodes
- Check for cycles (recursive occurrences in CPT)
- Marginalize away unknown nodes (except the one to be queried)

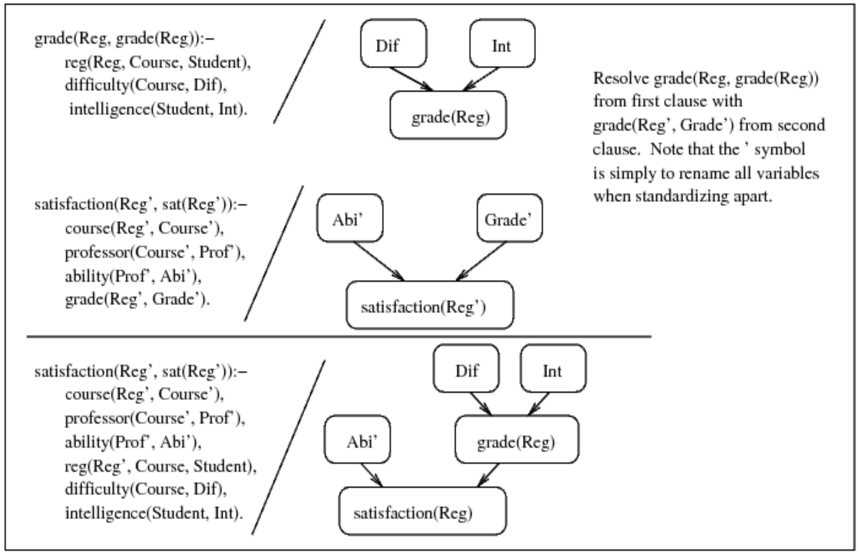


Figure 5: Resolution.

Semantics: Model-theoretic

- Let *P* be a CLP(*BN*) program
- *P* defines a joint probability distribution *PD(P)* over all ground Skolem terms (= RVs) as follows:
- These RVs with the corresponding grounded constraints (containing ground CPTs and parents) build a (possibly infinite) Bayesian net *BN* for *P*
- *BN* is acyclic, as for Skolemization, each Skolem term may appear only in one clause

Semantics: Model-theoretic

TECHNISCHE UNIVERSITÄT DARMSTADT

- Build a Herbrand quotient model:
  - Take the least Herbrand model *H* of the logical portion *C* of *P*
  - Every non-Skolem constant in *H* represents one equivalence class
  - Add every ground Skolem term in *P* to exactly one equivalence class
- *S* := Set of all possible quotient models
- *D* := Any Probability distribution over *S* that is consistent with *BN*
- *D* consistent with *BN* ↔ P(t = c | *P*) = $\sum_{h \in S, t \equiv c \text{ in } h} P(h \mid D)$
  - for any ground Skolem term t and non-Skolem constant c
- Our models for *P* are such pairs $\langle D, S \rangle$

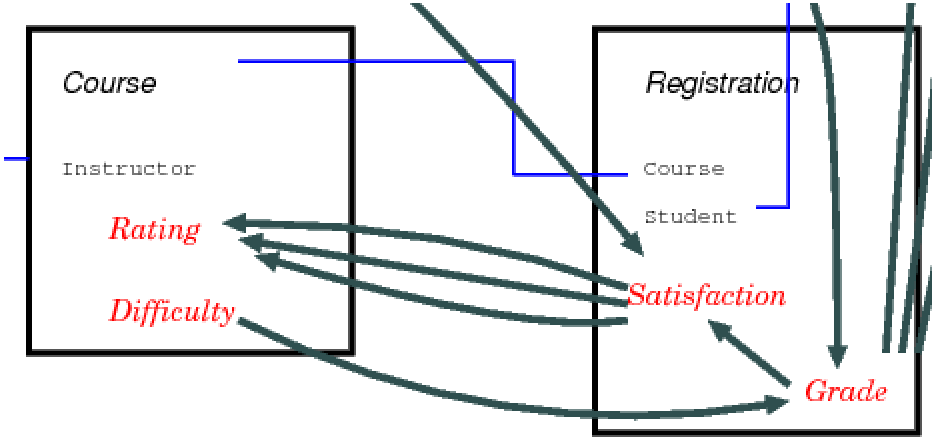
Semantics: Match the 2 views

- Theorem:
  - Given any CLP(*BN*) program *P*, any derivation *D* from *P*, any to *D* attached ground Bayes net *BN*' and any query *Q* to *BN*', the answer to *Q* is the same as would be given by *PD(P)* (defined from the full Bayes net *BN* of P)
- Proof (sketch):
  - Assume answer from *BN*' to query P(q | *E*) is different from answer from *BN* to same query for some evidence *E*
  - *BN*' ⊆ *BN*
  - As answers differ, there must flow evidence through q in *BN*, but not in *BN*'
  - By Lemma given in [1] this is impossible

Non-deterministic aggregates

- Aggregate all Skolem terms of interest (setof/3)
- Apply deterministic functions (like average/2) on their CPTs
- Compute CPT for goal and use it in its constraint

```
rating(C, Rat) :-
  setof(S,R^(registration(R,C),
             satisfaction(R,S)), Sats),
  average(Sats, CPT),
  {Rat = rating(C) with CPT}.
```



Non-deterministic aggregates

- Concept already part of Prolog framework
- Only problem: CPTs grow exponentially fast with number of a node's parents
- 2 approaches:
  - More intelligent data structure (binary trees with aggregating nodes)
  - Approximative inference on Bayesian nets

Recursion

- Can encode sequences of events or observations (Hidden Markov Models)
- Example scenario:
  - Send spy to enemy
  - 2 possible watchmen (careful & lax)
  - Only information: watchman at time I is likely to be watchman at time I+1
  - p(I): probability who is watching at time I
  - c(I): probability for the spy to be caught by time I

```
caught(0,Caught) :- !,
   {Caught = c(0) with p([t,f],[0.0,1.0],[])}.
caught(I,Caught) :-
   I1 is I-1, caught(I1, Caught0),
   watch(I, Police),
   caught(I,Caught0, Police, Caught).

watch(0, P) :- !,
   {P = p(0) with p([m,l],[0.5,0.5],[])}.
watch(I, P) :-
   I1 is I-1, watch(I1, P0),
   {P = p(I) with
       p([m,l],[0.8,0.2,0.2,0.8],[P0])}.

caught(I, C0, P, C) :-
   {C = c(I) with
      p([t,f],[1.0,1.0,0.05,0.001,
               0.0,0.0,0.95,0.099],[C0,P])}.
```

PRM to CLP(*BN*)

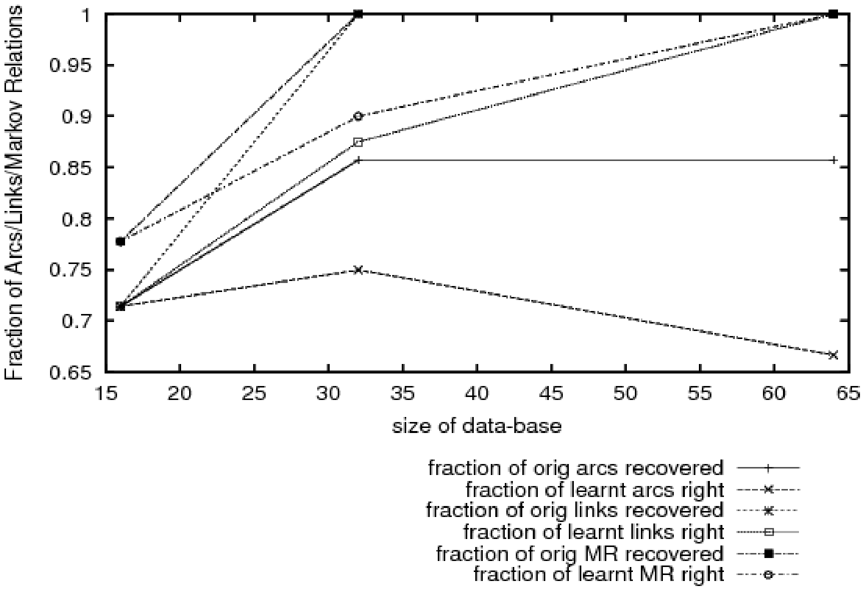
- Binarization of PRM tables (many <key,attribute> tables)
- Slot-chains via unification of foreign key variables
- Aggregates as shown
- Now the parents in the BN are found it remains to give the CPT in another literal

$$registration_3(RegKey, StudentKey),$$
$$registration_2(RegKey, CourseKey),$$
$$course_2(CourseKey, ProfKey),$$
$$professor_2(ProfKey, Ability)$$

$$\begin{aligned} findall(Ability,\ (\ & registration_2(RegKey, CourseKey), \\ & course_2(CourseKey, ProfKey), \\ & professor_2(ProfKey, Ability),\ \ L\ ), \end{aligned}$$
$$mean(L, X)$$

Learning CLP(BN) programs

- Simplifying assumption: predicates can be defined by just one clause
- Examples contain no missing data
- Use ILP learning algorithm (like ALEPH [3]) with Bayesian Information criterion (BIC) to find dependencies
- As most of these algorithms learn rules independently: remove cycles in a post learning process (authors recommend greedy algorithm with BIC)



Learning CLP(BN) programs

- Benchmarked on KDD01 Task 2 training data
- 2 class problem: Does a certain gene code for metabolism?
- Not significantly better than ordinary logic program learned with ALEPH [3]
- Advantage of CLP(BN): probabilities give a ranking classifier → can draw ROC curve and trade-off between F.P.R. & T.P.R.

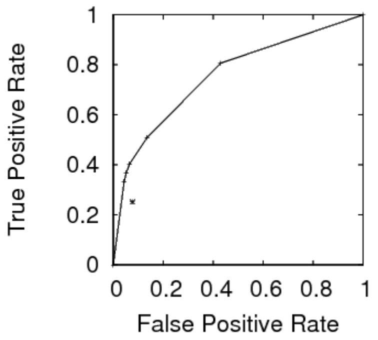


Figure 8: ROC Curve for Metabolism Application

References

- [1] Luc De Raedt, Paolo Frasconi, Kristian Kersting, Stephen Muggleton (eds.): Probabilistic Inductive Logic Programming, Springer-Ver-lag, 2009
- [2] Vítor Santos Costa, David Page, Maleeha Qazi, James Cussens: CLP(BN): Constraint Logic Programming for Probabilistic Knowledge, UAI 2003
- [3] Ashwin Srinivasan: The Aleph Manual, (last visit: Dec. 2009)