Statistical Relational Learning

www.biostat.wisc.edu/~dpage/cs760/

Goals for the lecture

you should understand the following concepts

- Markov networks
- Markov logic networks (MLNs)
- the parameter learning task in MLNs
- the structure learning task in MLNs
- probabilistic relational models (PRMs)
- plate models
- parameter learning and structure learning inherited from BNs

Statistical relational learning (SRL)

- two lectures ago, we saw the representational advantages of relational learning methods
  - ability to represent relationships among objects
  - ability to incorporate background knowledge

- but there are disadvantages of these methods
  - not well suited for representing uncertainty
  - not very robust given noisy data

- an active area in machine learning is developing statistical relational methods that aim to preserve the advantages while alleviating the disadvantages

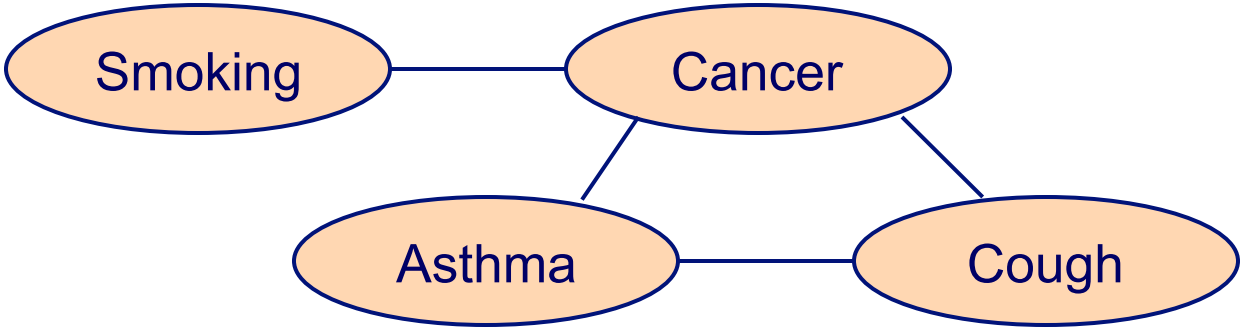
- the real world is complex and uncertain
  - logic handles the complexity
  - probability handles the uncertainty

Many SRL approaches have been developed

- knowledge-based model construction [Wellman et al., 1992]
- stochastic logic programs [Muggleton, 1996]
- probabilistic relational models [Friedman et al., 1999]
- relational Markov networks [Taskar et al., 2002]
- constraint logic programming for probabilistic knowledge [Santos Costa et al., *UAI* 2003]
- Bayesian logic [Milch et al., 2005]
- ✓ Markov logic [Richardson & Domingos, *Machine Learning* 2006]
- etc.

Recall: Markov networks

- a Markov network is an undirected graphical model



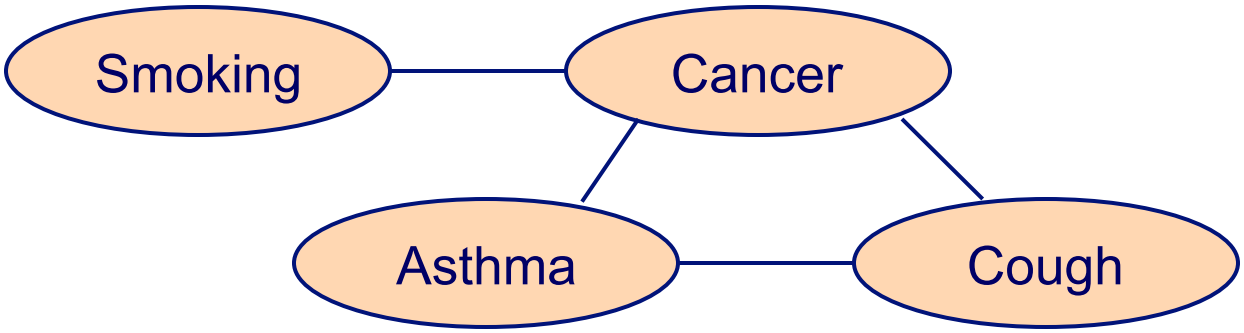
- *potential functions* are defined over cliques

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_{c \in Cliques} \Phi_c(\boldsymbol{x}_c)$$

$$Z = \sum_{\boldsymbol{x}} \prod_{c} \Phi_c(\boldsymbol{x}_c)$$

Smoking	Cancer	$\Phi(S, C)$
false	false	4.5
false	true	4.5
true	false	2.7
true	true	4.5

Recall: Markov networks



- potentials can be represented using log-linear models over features

$$P(\boldsymbol{x}) = \frac{1}{Z}\exp\left(\sum_i w_i \, f_i(\boldsymbol{x})\right)$$

weight of feature i

feature i

$$f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg \text{ Smoking } \vee \text{ Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 1.5$$

First-order logic

- Constants, variables, functions, predicates
  e.g. Anna, x, MotherOf(x), Friends(x, y)

- **Formulas:** constructed from constants, variables, functions, predicates
  e.g. Friends(x, MotherOf(Anna)), Friends(x, y) $\wedge$ Friends(x, z)

- **Grounding:** Replace all variables by constants
  e.g. Friends (Anna, Bob)

- **World** (model, interpretation):
  Assignment of truth values to all ground predicates

Markov logic: intuition

- a logical knowledge base is a set of **hard constraints** on the set of possible worlds
- let's make them **soft constraints**: when a world violates a formula, it becomes less probable, not impossible
- give each formula a weight (higher weight → stronger constraint)

$$\mathrm{P}(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$

MLN definition

- a *Markov Logic Network* (MLN) is a set of pairs (F, w) where
  F is a formula in first-order logic
  w is a real number

- together with a set of constants, it defines a Markov network with
  – one node for each grounding of each predicate in the MLN
  – one feature for each grounding of each formula F in the MLN, with the corresponding weight w

MLN example: friends & smokers

Smoking causes cancer.

Friends have similar smoking habits.

MLN example: friends & smokers

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$

$$\forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

MLN example: friends & smokers

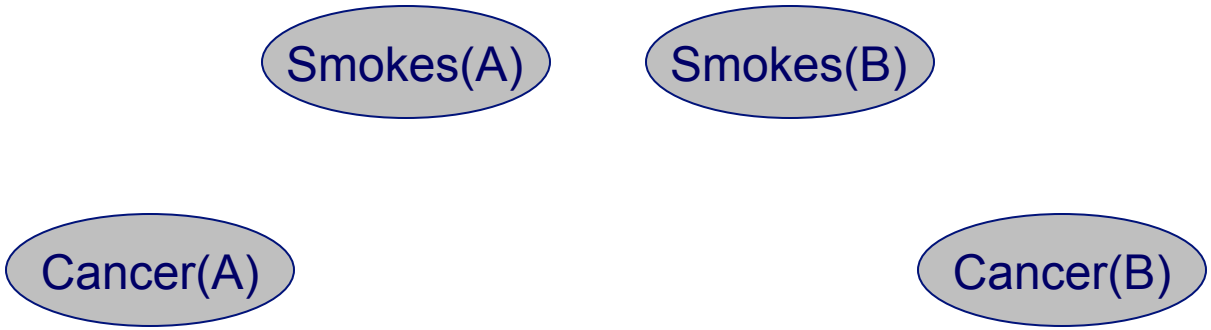
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Two constants: **Anna** (A) and **Bob** (B)

MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

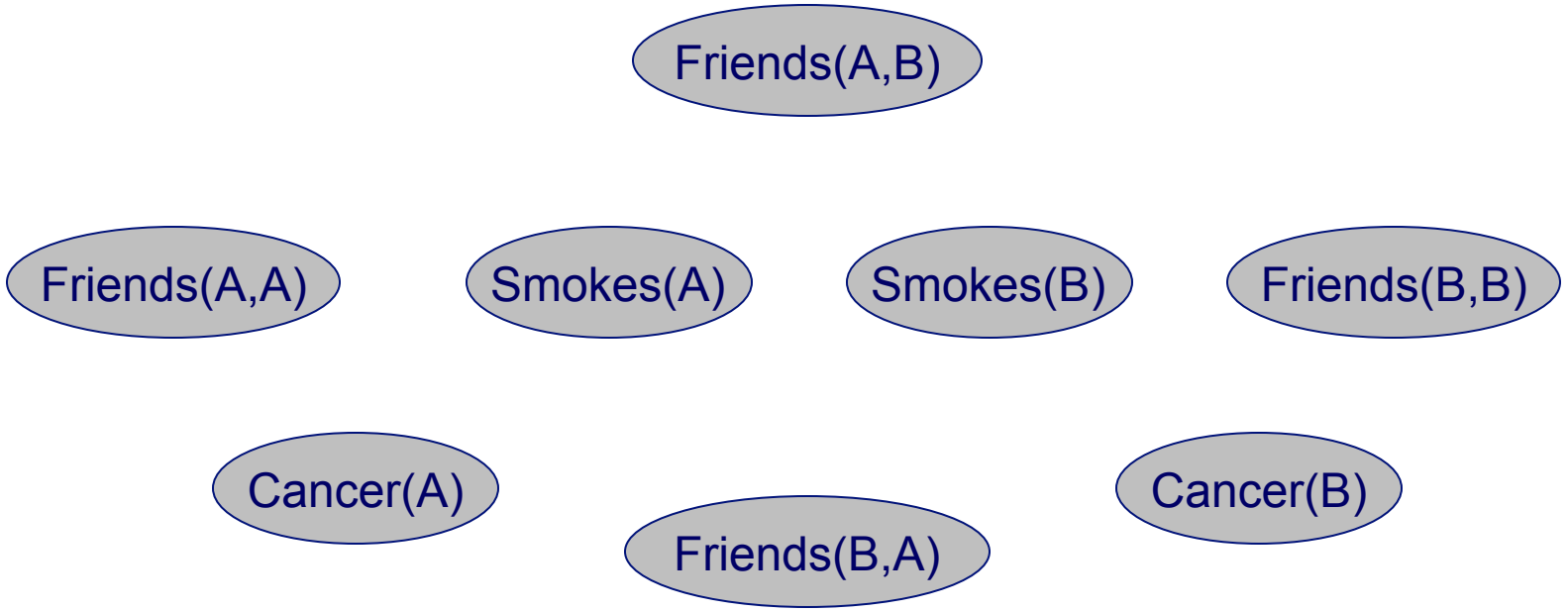
Two constants: **Anna** (A) and **Bob** (B)



MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

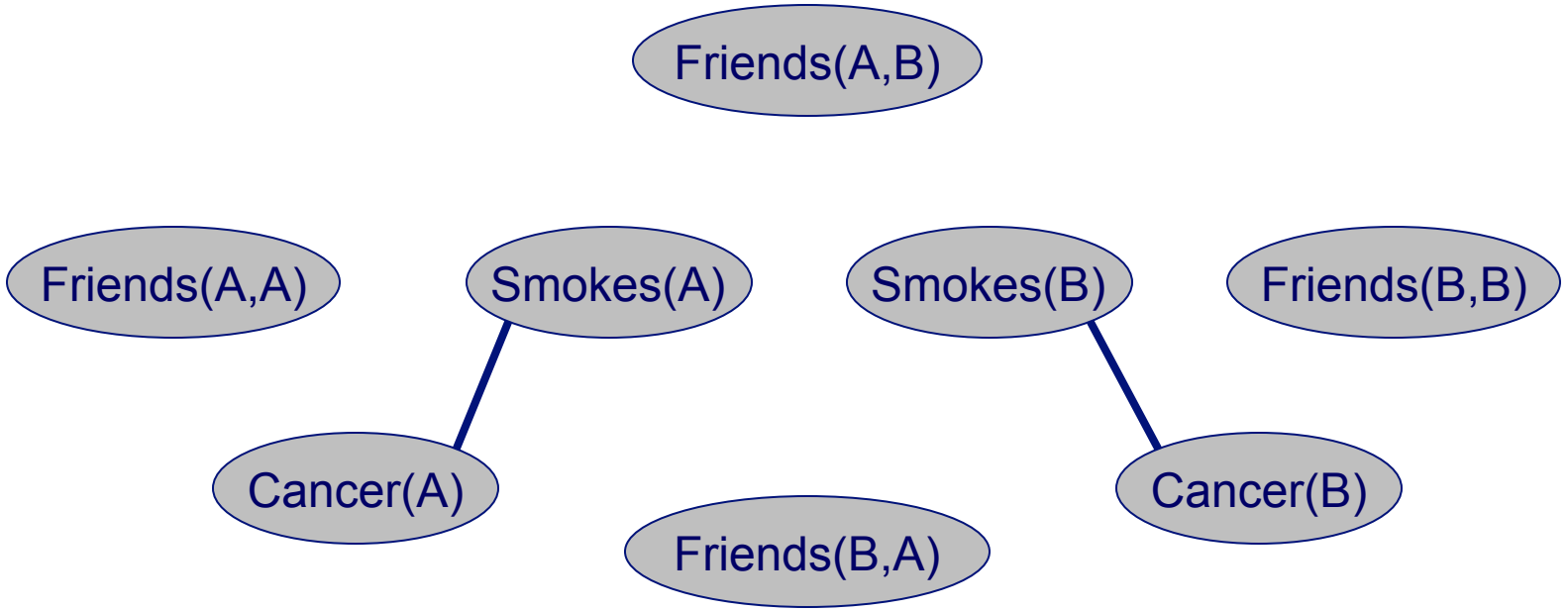
Two constants: **Anna** (A) and **Bob** (B)



MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

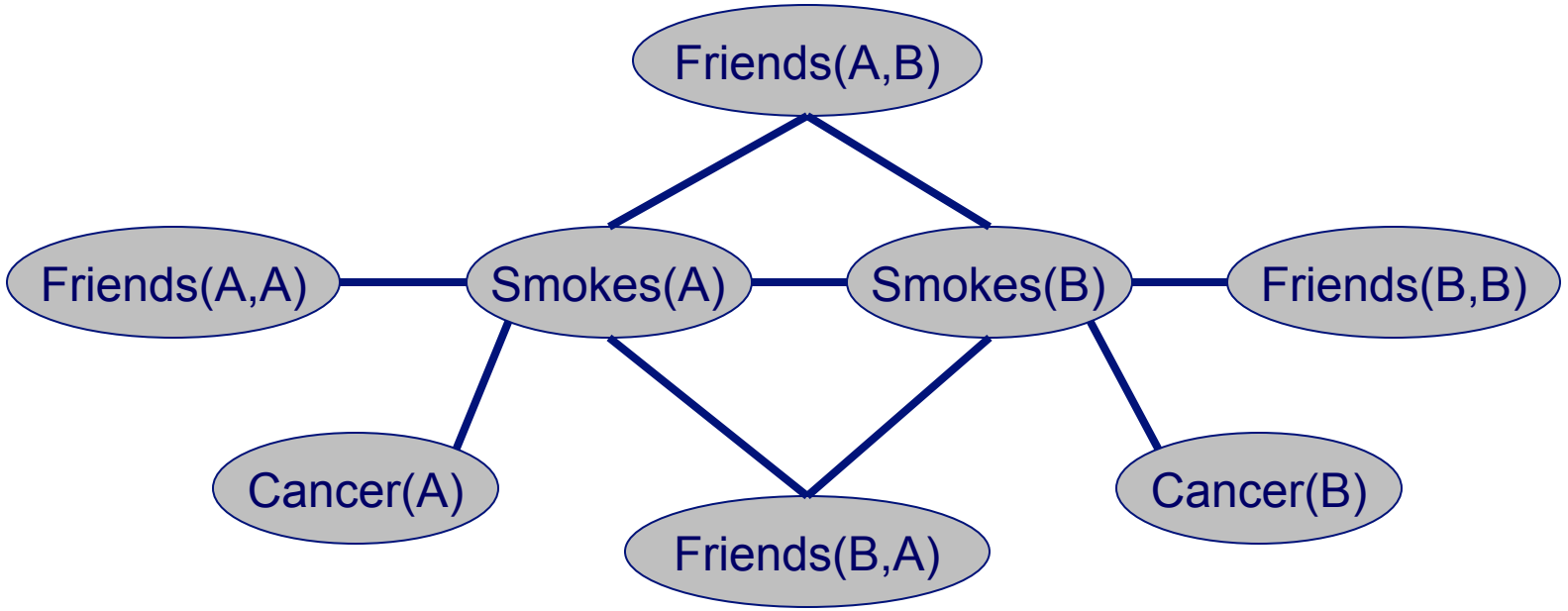
Two constants: **Anna** (A) and **Bob** (B)



MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)



Markov logic networks

- a MLN is a **template** for ground Markov nets
  - the logic determines the form of the cliques
  - but if we had one more constant (say, **Larry**), we'd get a different Markov net

- we can determine the probability of a world $\boldsymbol{v}$ (assignment of truth values to ground predicates) by

$$P(\boldsymbol{v}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\boldsymbol{v}) \right)$$

weight of formula i

# of true groundings of formula i in $\boldsymbol{v}$

Example translated to Markov Network

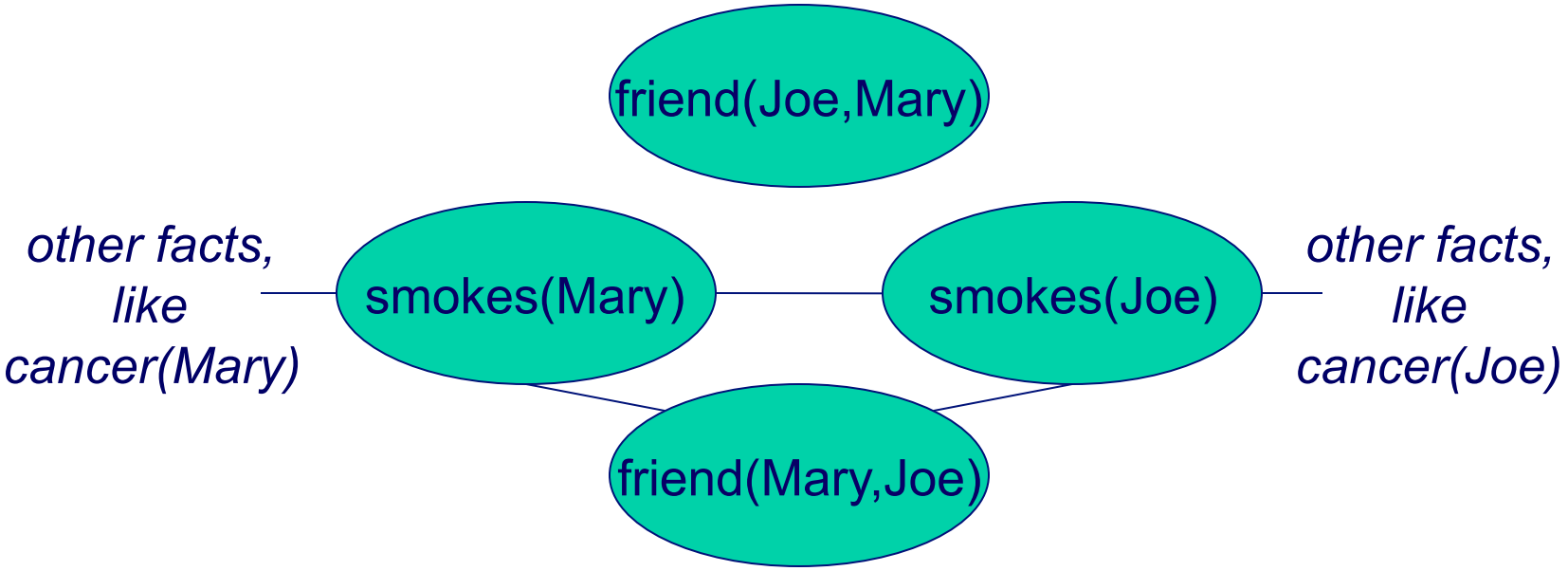
- Facts:

$$smokes(Mary)$$
$$smokes(Joe)$$
$$friend(Joe, Mary)$$

- Rules:

Friend(x,y) ∧ Smokes(x) ->
Smokes(y)

$$\forall x \forall y\ friends(x, y) \wedge smokes(x) \rightarrow smokes(y)$$



Computing weights

Consider the effect of rule R: Friend(x,y) ∧ Smokes(x) -> Smokes(y)

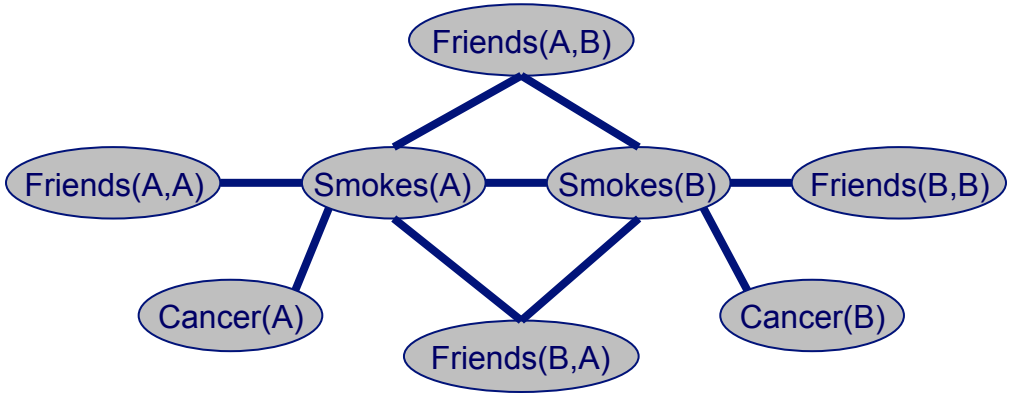
weight
1.1

	smokes(Mary)		¬smokes(Mary)	
	smokes(Joe)	¬smokes(Joe)	smokes(Joe)	¬smokes(Joe)
friends(Mary,Joe)	$e^{1.1}$	1	$e^{1.1}$	$e^{1.1}$
¬friends(Mary,Joe)	$e^{1.1}$	$e^{1.1}$	$e^{1.1}$	$e^{1.1}$

This is the only setting that does not satisfy R
Thus, it is given value 1, while the others are
Given value exp(weight(R))

Probability of a world in an MLN

$$\boldsymbol{v} = \begin{bmatrix} \text{Friends(A,A)} = \text{T} \\ \text{Friends(A,B)} = \text{T} \\ \text{Friends(B,A)} = \text{T} \\ \text{Friends(B,B)} = \text{T} \\ \text{Smokes(A)} = \text{F} \\ \text{Smokes(B)} = \text{T} \\ \text{Cancer(A)} = \text{F} \\ \text{Cancer(B)} = \text{F} \end{bmatrix}$$



$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$

$x = \text{A}$ T

$x = \text{B}$ F

$n_1(\boldsymbol{v}) = 1$

$$\forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

$x = \text{A}, y = \text{A}$ T

$x = \text{A}, y = \text{B}$ F

$x = \text{B}, y = \text{A}$ F

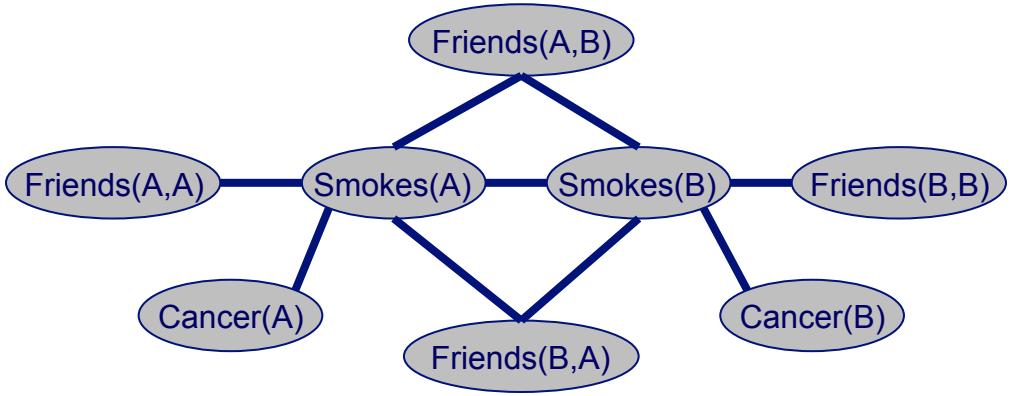
$x = \text{B}, y = \text{B}$ T

$n_2(\boldsymbol{v}) = 2$

# of true groundings of formula 1 in $\boldsymbol{v}$

Probability of a world in an MLN

$$\boldsymbol{v} = \begin{bmatrix} \text{Friends(A,A)} = \text{T} \\ \text{Friends(A,B)} = \text{T} \\ \text{Friends(B,A)} = \text{T} \\ \text{Friends(B,B)} = \text{T} \\ \text{Smokes(A)} = \text{F} \\ \text{Smokes(B)} = \text{T} \\ \text{Cancer(A)} = \text{F} \\ \text{Cancer(B)} = \text{F} \end{bmatrix}$$



$$P(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(\boldsymbol{v})\right)$$

$$= \frac{1}{Z}\exp\left(1.5(1) + 1.1(2)\right)$$

Three MLN tasks

- inference: can use the toolbox of inference methods developed for ordinary Markov networks
  - Markov chain Monte Carlo methods, including persistent contrastive divergence (PCD) where we iterate MCMC and gradient ascent steps (don't wait for MCMC to converge)
  - belief propagation
  - variational methods

  in tandem with weighted SAT solver (e.g., MaxWalkSAT [Kautz et al., 1997])

- parameter learning

- structure learning: can use ordinary relational learning methods like FOIL or other ILP algorithms to learn new formula

MLN learning tasks

- the input to the learning process is a relational database of ground atoms

Friends(x, y)	Smokes(x)	Cancer(x)
Anna, Anna	Bob	Bob
Anna, Bob		
Bob, Anna		
Bob, Bob		

- the closed world assumption is used to infer the truth values of atoms not present in the DB

Parameter learning

- parameters (weights on formulas) can be learned using gradient ascent

$$\frac{\partial}{\partial w_i} \log P_w(\boldsymbol{v}) = n_i(\boldsymbol{v}) - E_w\left[n_i(\boldsymbol{v})\right]$$

# of times clause i is true in data

Expected # times clause i is true according to MLN

- approximation methods may be needed to estimate both terms

MLN experiment

- testbed: a DB describing Univ. of Washington CS department
  - 12 predicates

    Professor(*person*)

    Student(*person*)

    Area(*x*, *area*)

    AuthorOf(*publication*, *person*)

    AdvisedBy(*person*, *person*)

    etc.

  - 2707 constants

    publication (342)

    person (442)

    course (176)

    project (153)

    etc.

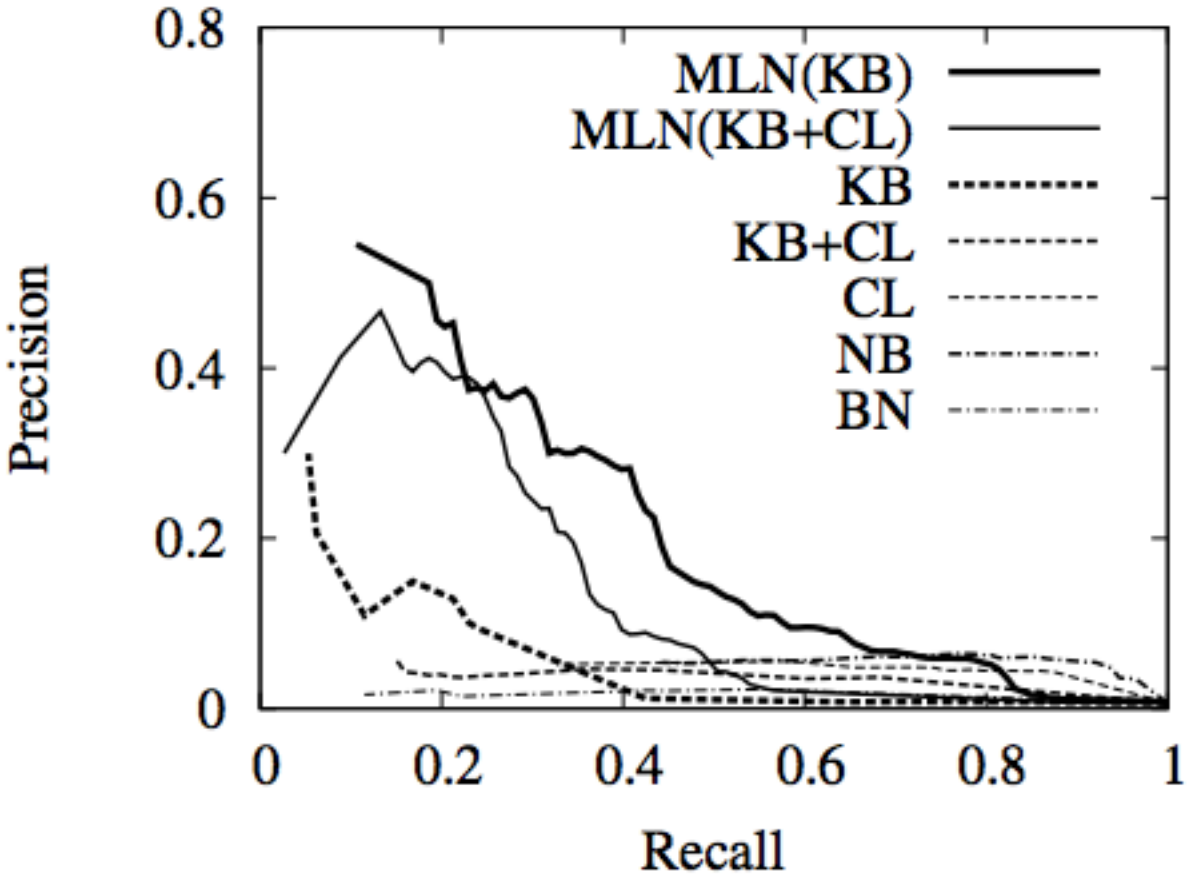
MLN experiment

- obtained knowledge base by having four subjects provide a set of formulas in first-order logic describing the domain

- the formulas in the KB represent statements such as
  - students are not professors
  - each student has at most one advisor
  - if a student is an author of a paper, so is her advisor
  - at most one author of a given publication is a professor
  - etc.

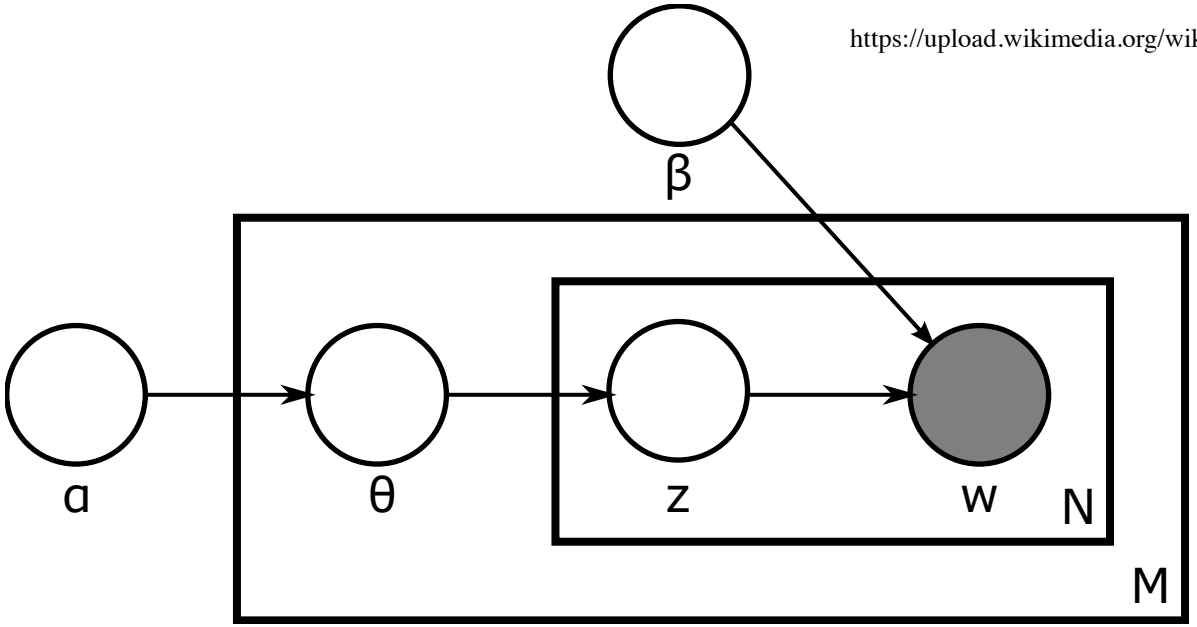
- note that the KB is not consistent

Learning to predict the AdvisedBy(x, y) relation



MLN w/ original KB
MLN w/ KB + ILP learned rules
KB alone
KB + ILP learned rules
ILP learned rules
naïve Bayes
Bayes net learner

Plates



https://upload.wikimedia.org/wikipedia/commons/d/d3/Latent_Dirich...

A rectangle labeled “N” denotes N copies of the Bayes net inside.
Typically arcs go *into* rectangles, but we relax to allow outgoing next…

PRMs

- Bayes nets over relational databases, with BN structure and parameters defined at the level of the database schema
- PRMs allow properties of an entity to depend probabilistically on properties of other related entities
- Extension to typical plate models: allow arcs from inside a rectangle to outside (or to inside another rectangle), requiring aggregation

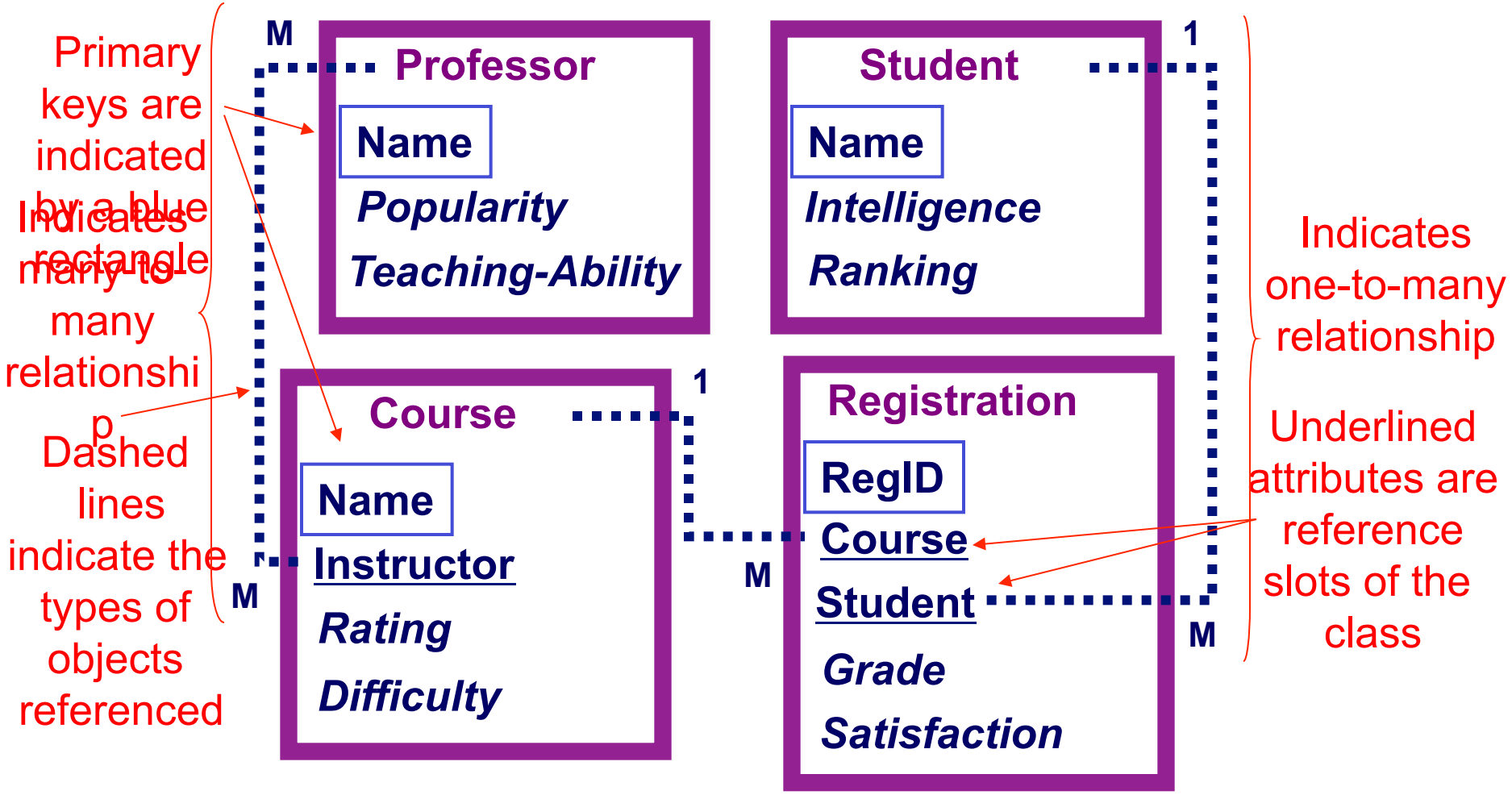
Mapping PRMs from Relational Models

- The representation of PRMs is a direct mapping from that of relational databases
- A relational model consists of a set of *classes* $X_1,\ldots,X_n$ and a set of *relations* $R_1,\ldots,R_m$, where each relation R_i is typed
- Each class or entity type (corresponding to a single relational table) is associated with a set of *attributes* $\mathcal{A}(X_i)$ and a set of *reference slots* $\mathcal{R}(X)$

PRM Semantics

- Reference slots correspond to attributes that are foreign keys (key attributes of another table)
- $X.\rho$, is used to denote reference slot ρ of X. Each reference slot ρ is typed according to the relation that it references

University Domain Example - Relational Schema



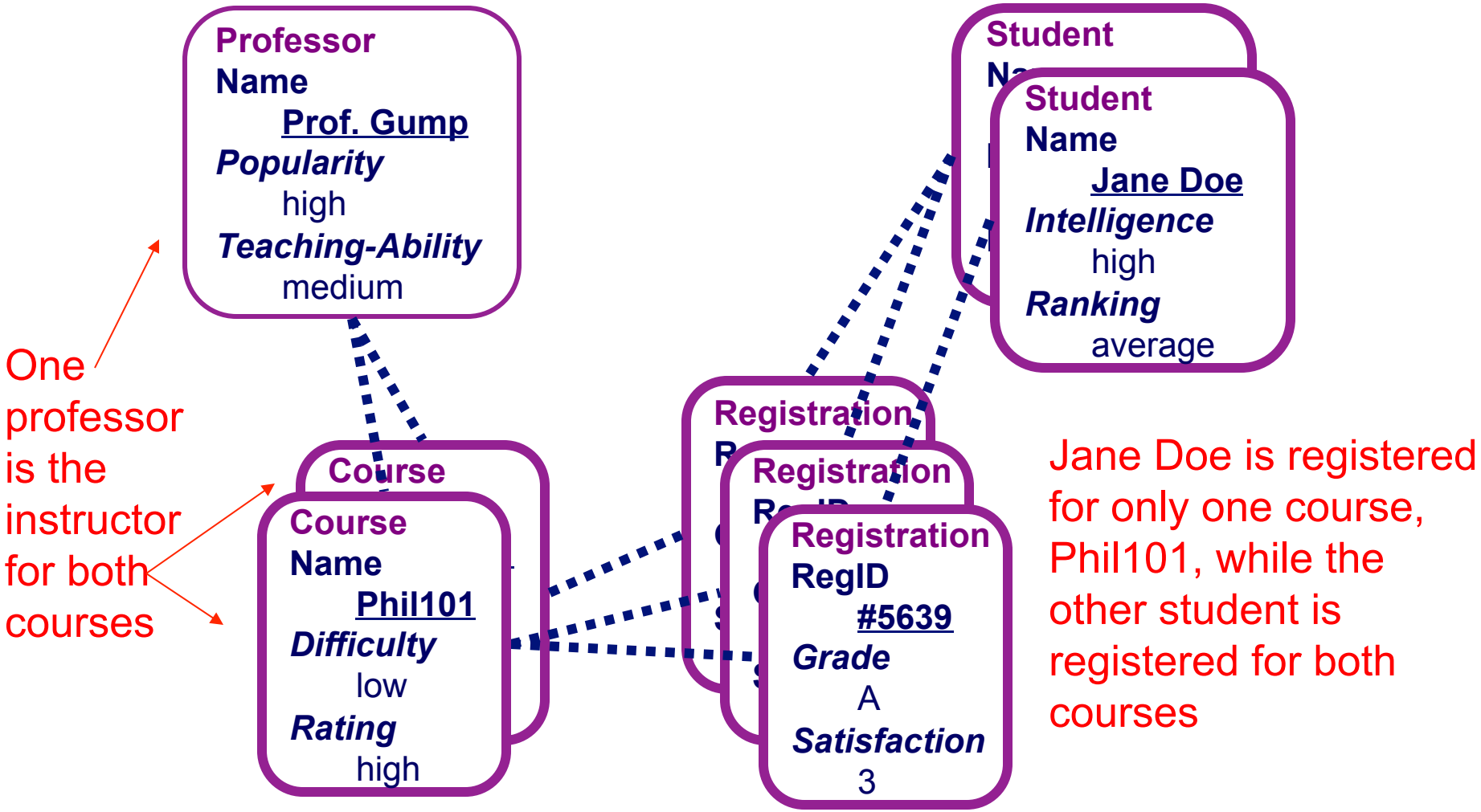
PRM Semantics Continued

- Each attribute $A_j \in \mathcal{A}(X_i)$ takes on values in some fixed domain of possible values denoted $V(A_j)$. We assume that value spaces are finite
- Attribute *A* of class *X* is denoted *X.A*
- For example, the **Student** class has an *Intelligence* attribute and the value space or domain for **Student**.*Intelligence* might be {*high*, *low*}

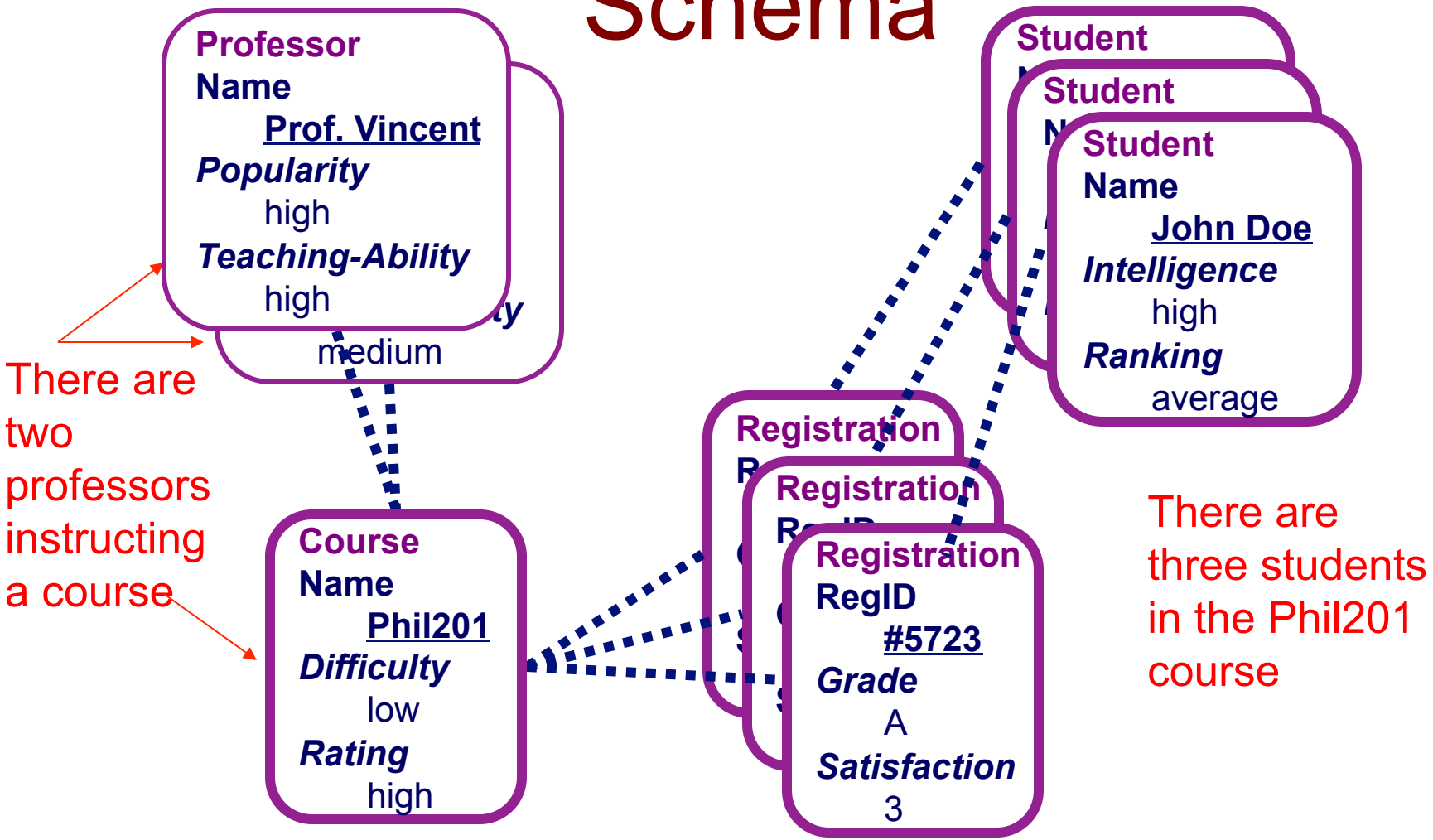
PRM Semantics Continued

- An *instance* $\mathcal{I}$ of a schema specifies a set of objects x, partitioned into classes, such that there is a value for each attribute $x.A$ and a value for each reference slot $x.\rho$
- $\mathcal{A}(x)$ is used interchangeably with $\mathcal{A}(X)$, where x is of class X. For each object x in the instance and each of its attributes A, we use $I_{x.A}$ to denote the value of $x.A$ in I

University Domain Example – An Instance of the Schema



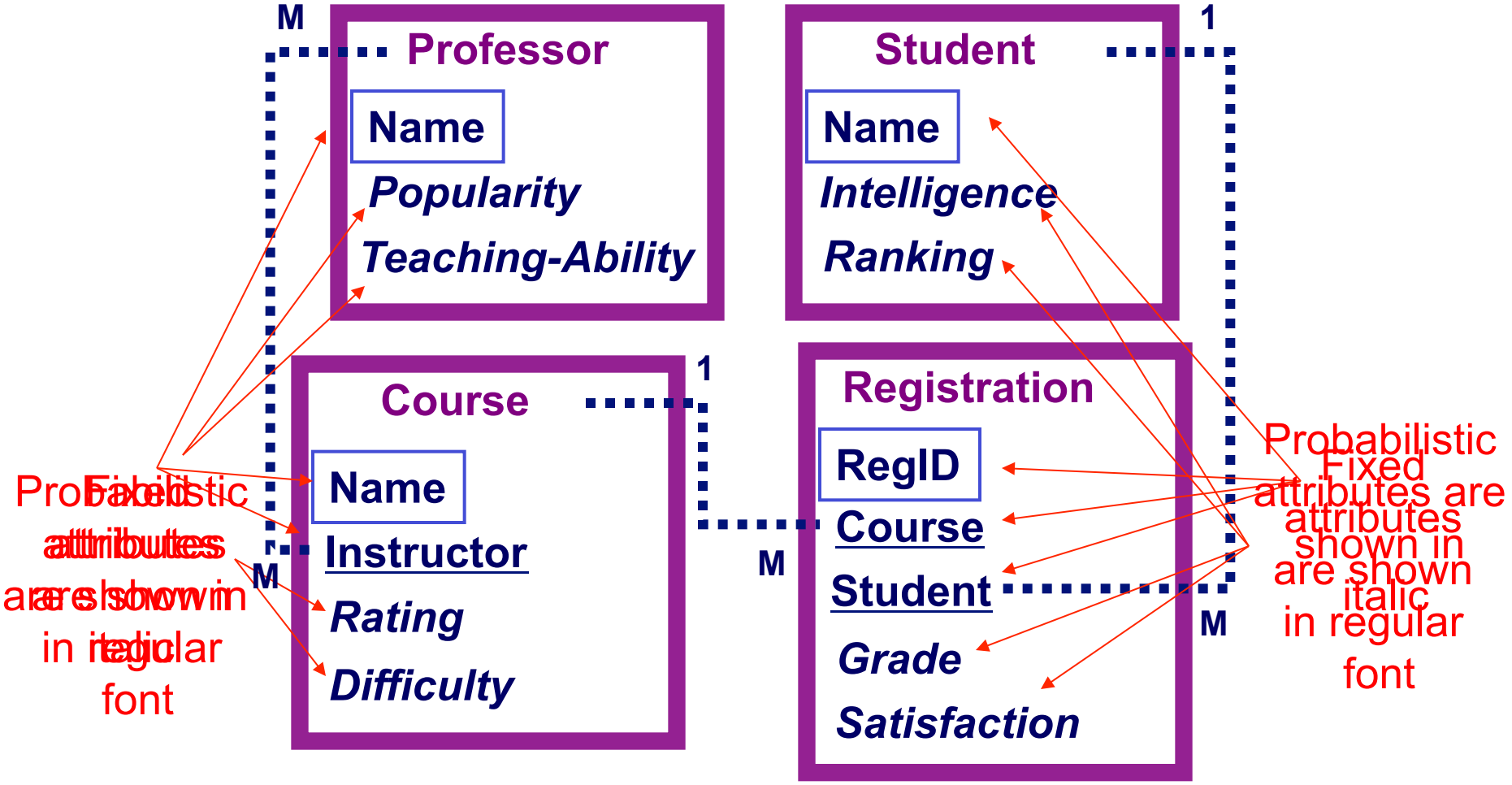
University Domain Example – Another Instance of the Schema



PRM Semantics Continued

- Some attributes, such as name or social security number, are fully determined. Such attributes are labeled as *fixed*. Assume that they are known in any instantiation of the schema
- The other attributes are called *probabilistic*

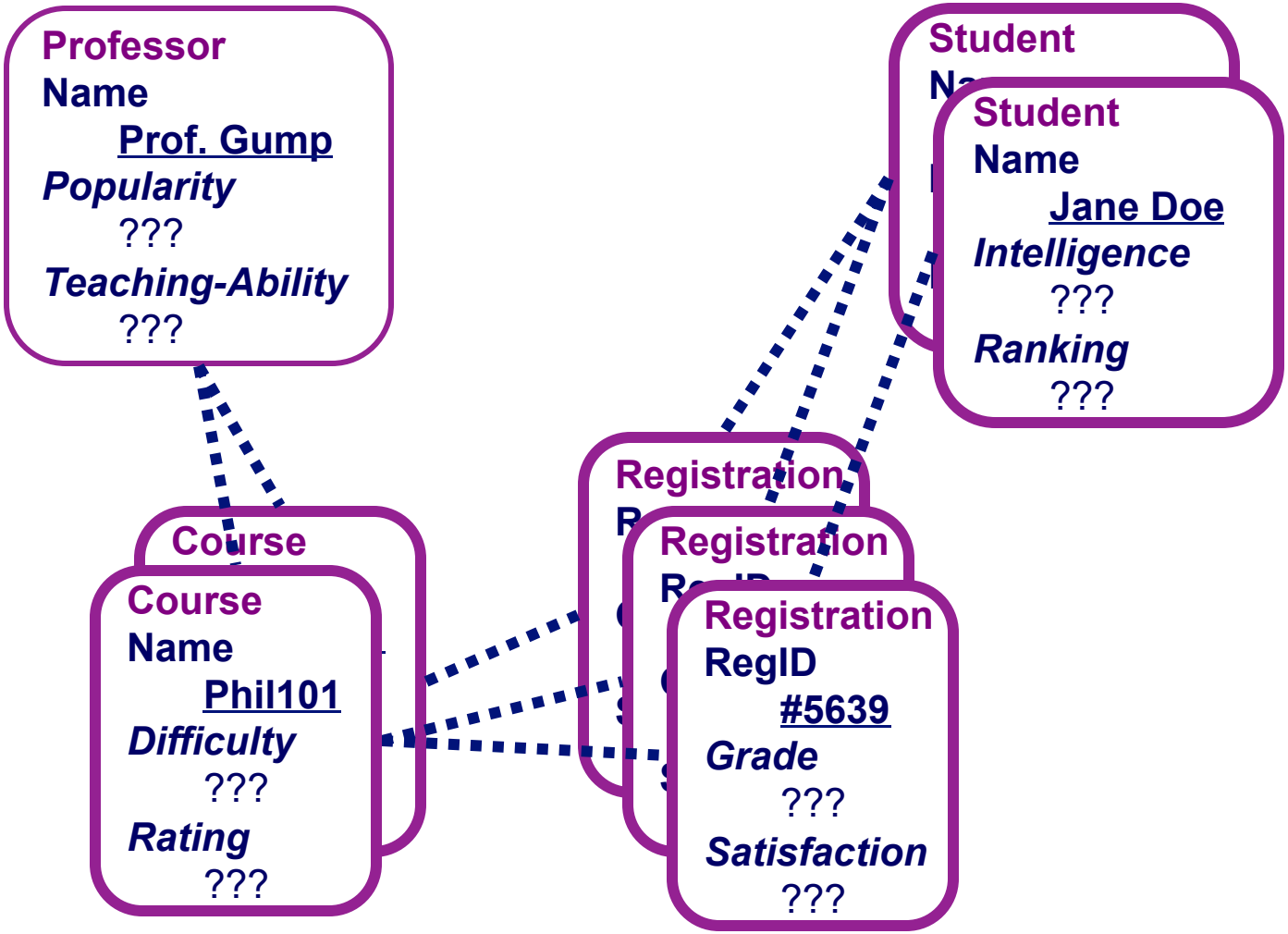
University Domain Example - Relational Schema



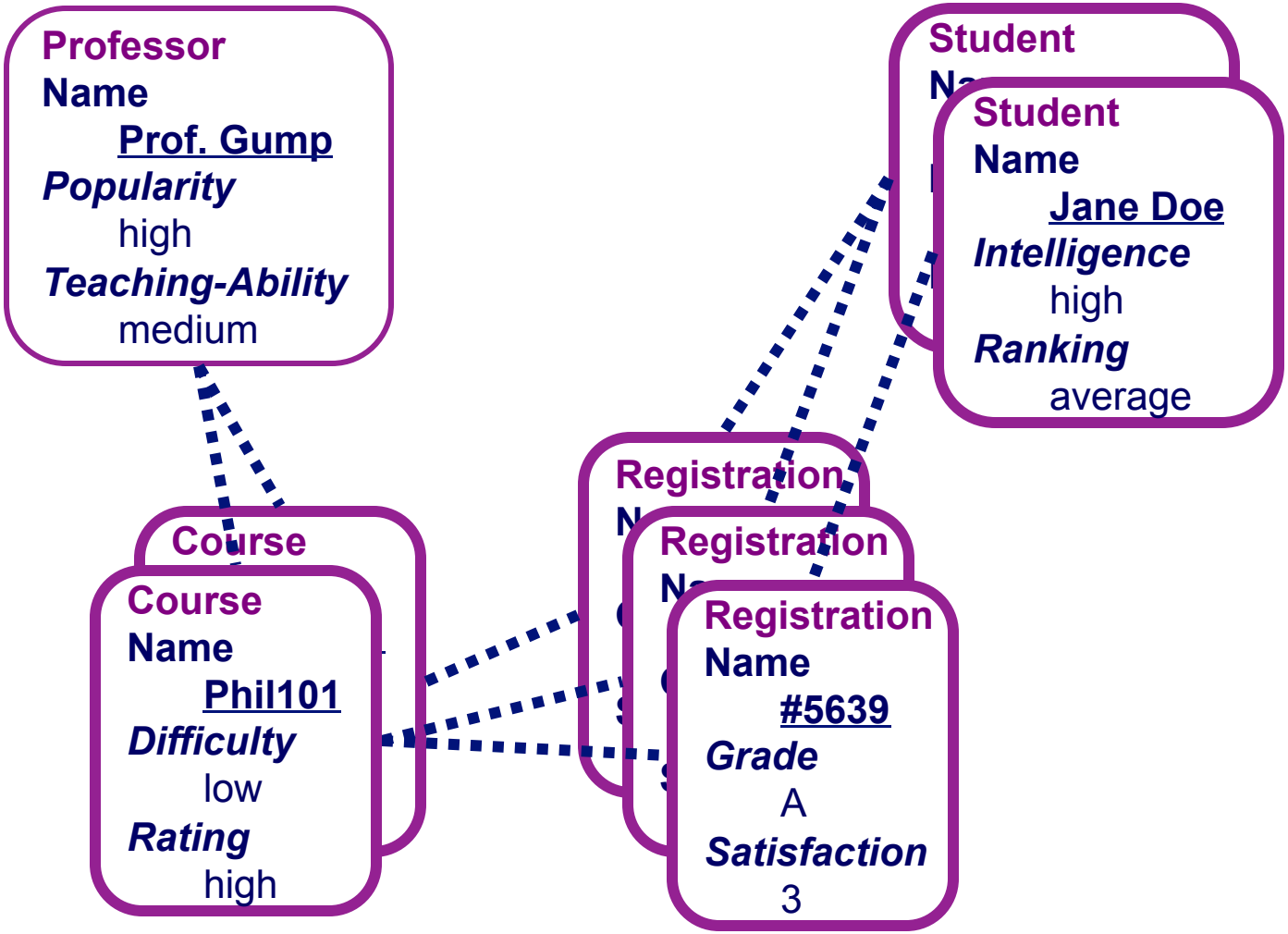
PRM Semantics Continued

- A *skeleton structure* σ of a relational schema is a partial specification of an instance of the schema. It specifies the set of objects $\sigma(X_i)$ for each class, the values of the fixed attributes of these objects, and the relations that hold between the objects
- The values of probabilistic attributes are left unspecified
- A *completion I* of the skeleton structure σ extends the skeleton by also specifying the values of the probabilistic attributes

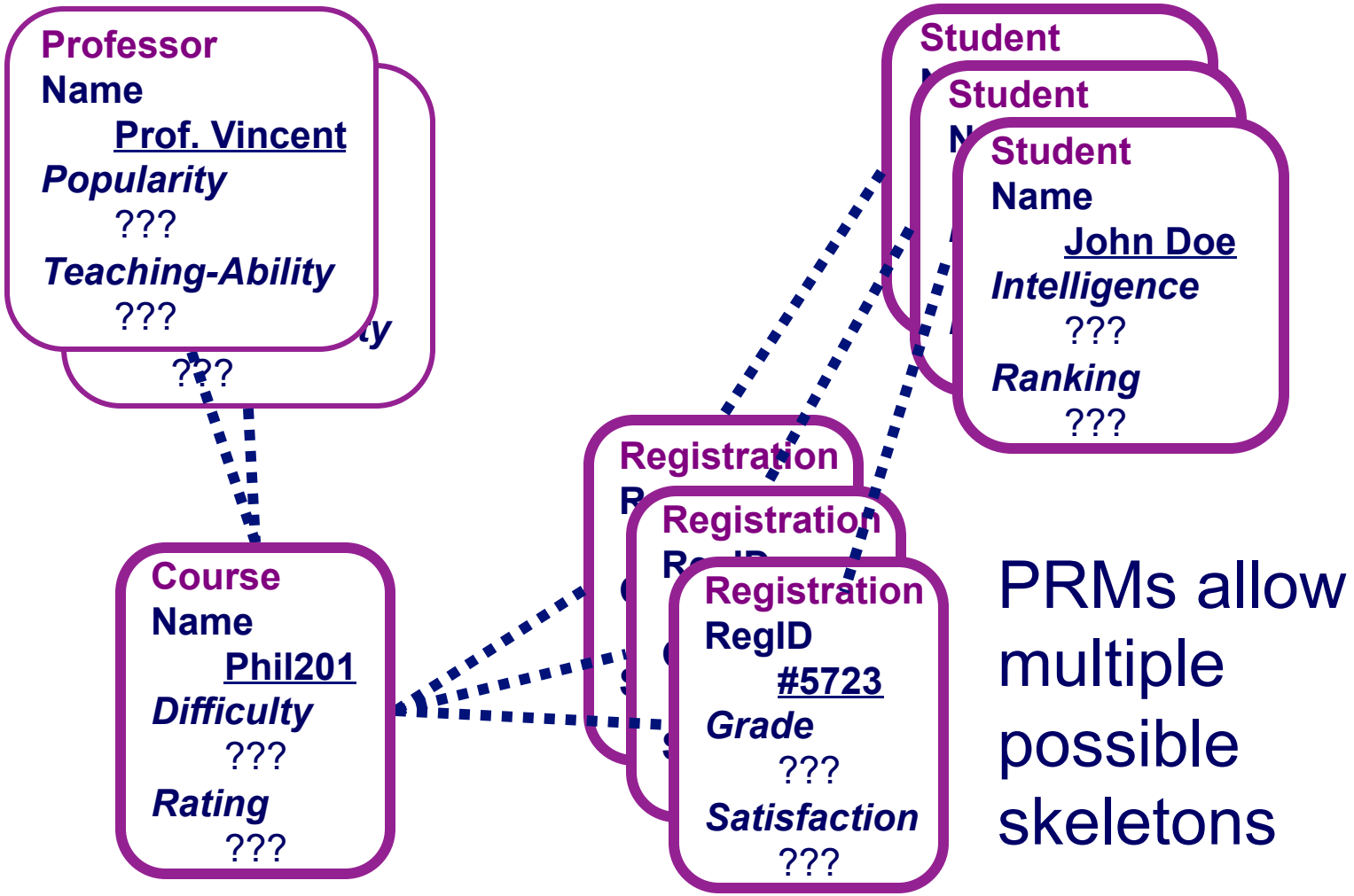
University Domain Example – Relational Skeleton



University Domain Example – The Completion Instance *I*

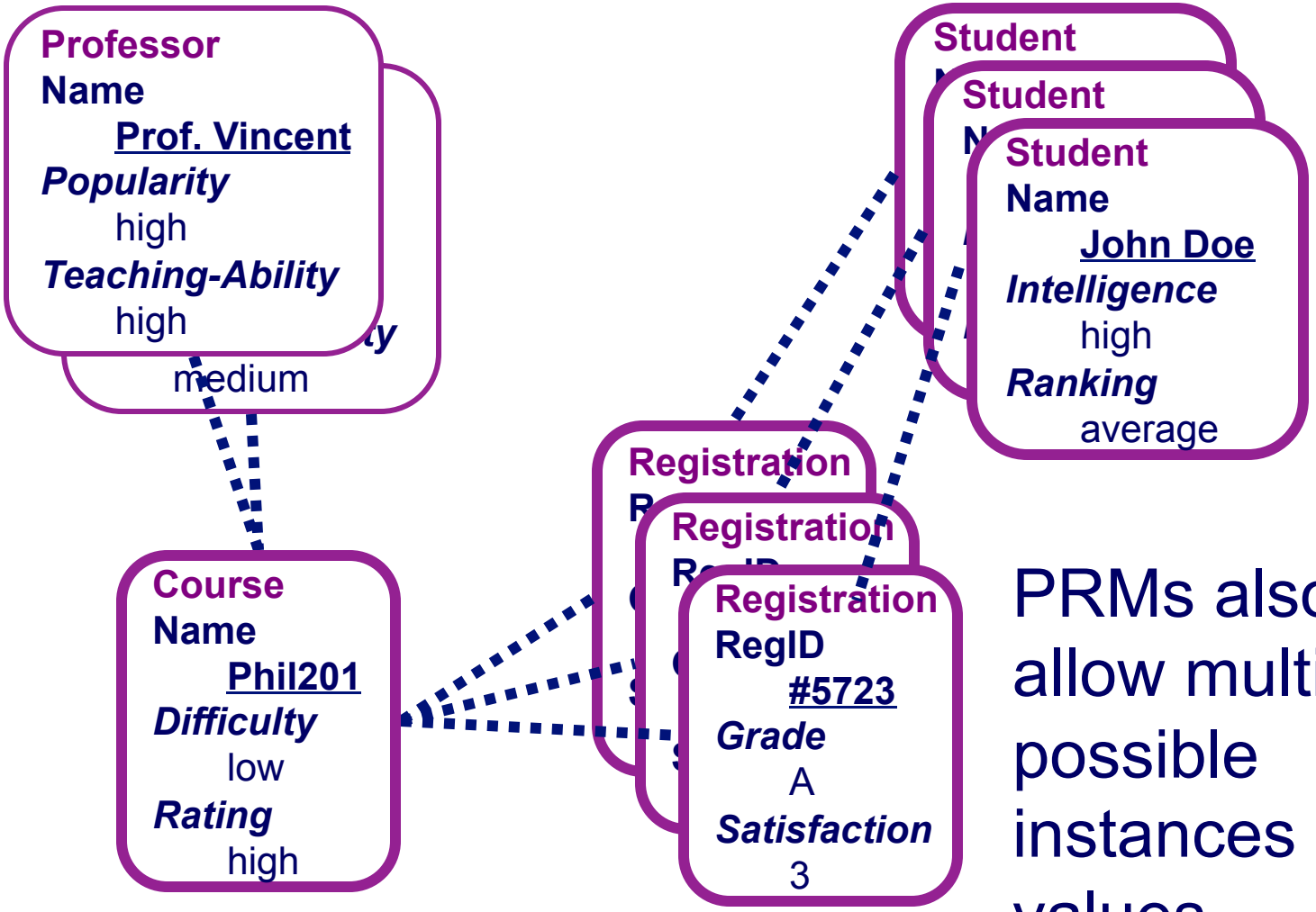


University Domain Example – Another Relational Skeleton



PRMs allow multiple possible skeletons

University Domain Example – The Completion Instance *I*



PRMs also allow multiple possible instances and values

More PRM Semantics

- For each reference slot ρ, we define an inverse slot, ρ^{-1}, which is the inverse function of ρ
- For example, we can define an inverse slot for the *Student* slot of **Registration** and call it *Registered-In*. Since the original relation is a one-to-many relation, it returns a set of **Registration** objects
- A final definition is the notion of a *slot chain* $\tau=\rho_1..\rho_m$, which is a sequence of reference slots that defines functions from objects to other objects to which they are indirectly related. For example, **Student**.*Registered-In*.*Course*.*Instructor* can be used to denote a student's set of instructors

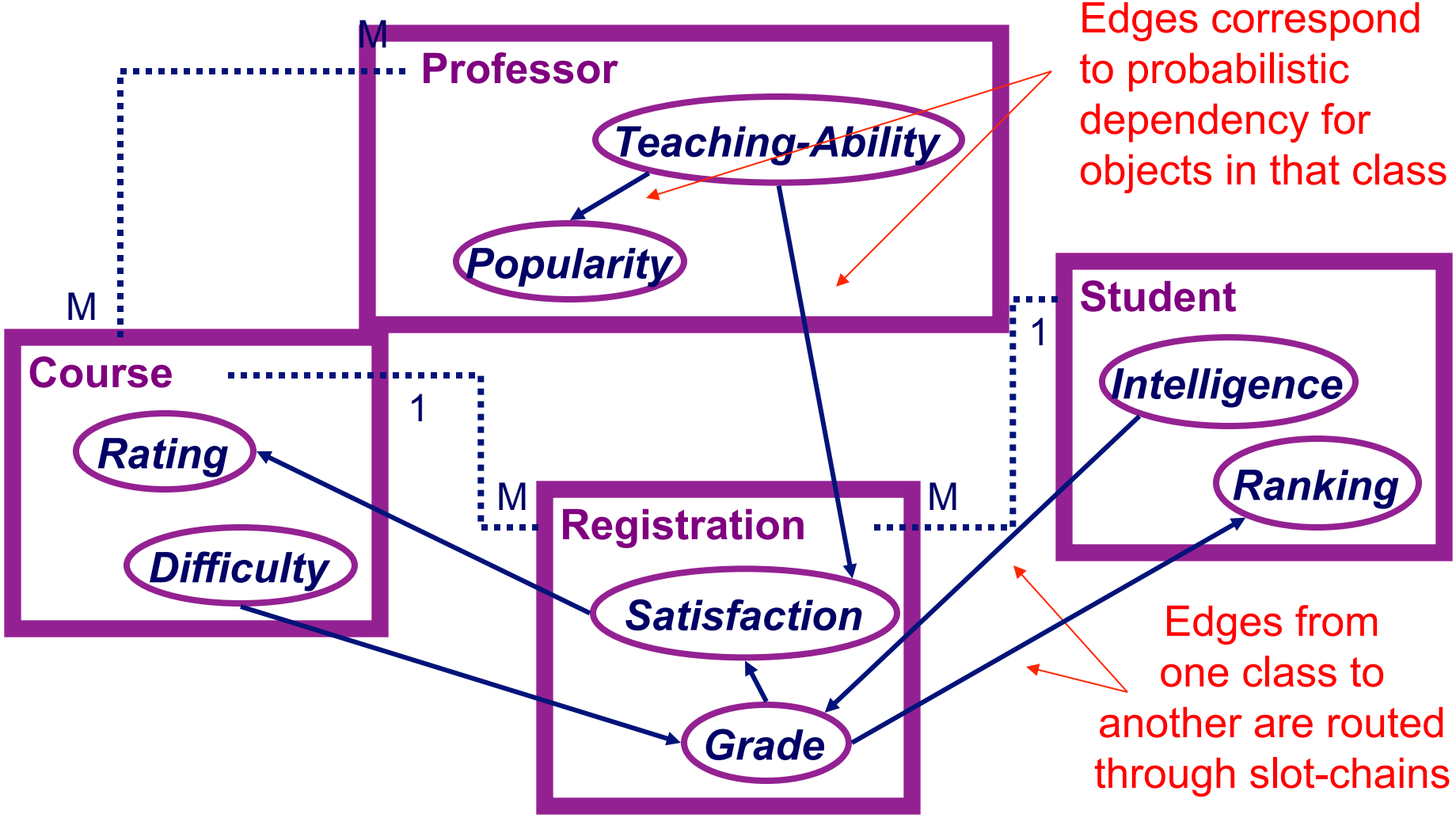
Definition of PRMs

- The probabilistic model consists of two components: the qualitative dependency structure, S, and the parameters associated with it, θ_S
- The dependency structure is defined by associating with each attribute *X.A* a set of *parents* Pa(X.A); parents are attributes that are "direct influences" on *X.A*. This dependency holds for any object of class *X*

Definition of PRMs Cont'd

- The attribute *X.A* can depend on another probabilistic attribute *B* of *X*. This dependence induces a corresponding dependency for individual objects
- The attribute *X.A* can also depend on attributes of related objects *X.τ.B*, where *τ* is a slot chain
- For example, given any **Registration** object *r* and the corresponding **Professor** object *p* for that instance, *r.Satisfaction* will depend probabilistically on *r.Grade* as well as *p.Teaching-Ability*

PRM Dependency Structure for the University Domain



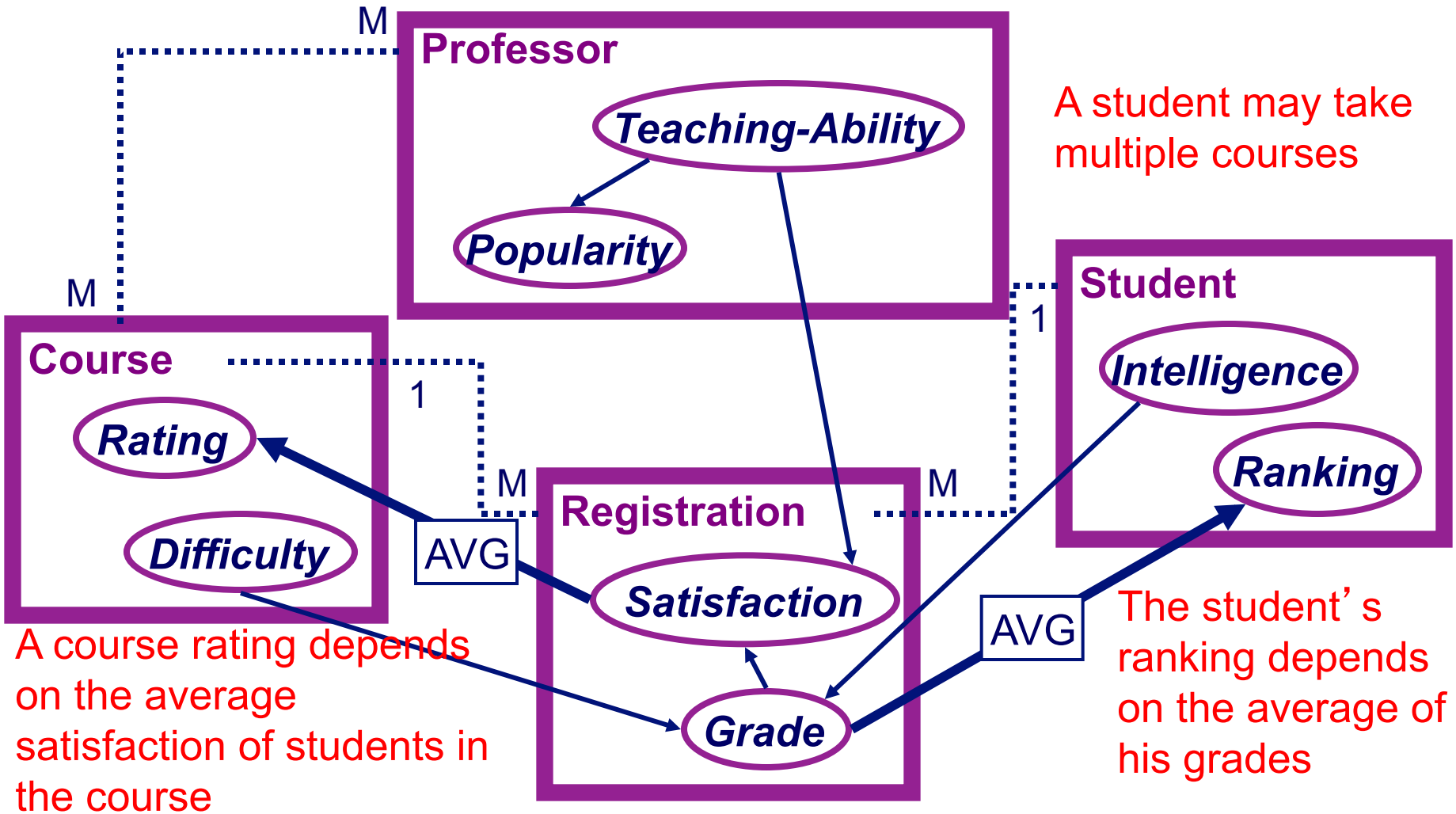
Dependency Structure in PRMs

- As mentioned earlier, $x.\tau$ represents the set of objects that are τ-relatives of x. Except in cases where the slot chain is guaranteed to be single-valued, we must specify the probabilistic dependence of $x.A$ on the multiset $\{y.B : y \in x.\tau\}$
- The notion of *aggregation* from database theory gives us the tool to address this issue; i.e., $x.a$ will depend probabilistically on some aggregate property of this multiset

Aggregation in PRMs

- Examples of aggregation are: the mode of the set (most frequently occurring value); mean value of the set (if values are numerical); median, maximum, or minimum (if values are ordered); cardinality of the set; etc
- An aggregate essentially takes a multiset of values of some ground type and returns a summary of it
- The type of the aggregate can be the same as that of its arguments, or any type returned by an aggregate. $X.A$ can have $\gamma(X.\tau.B)$ as a parent; the semantics is that for any $x \in X$, $x.a$ will depend on the value of $\gamma(x.\tau.b)$, $V(\gamma(x.\tau.b))$

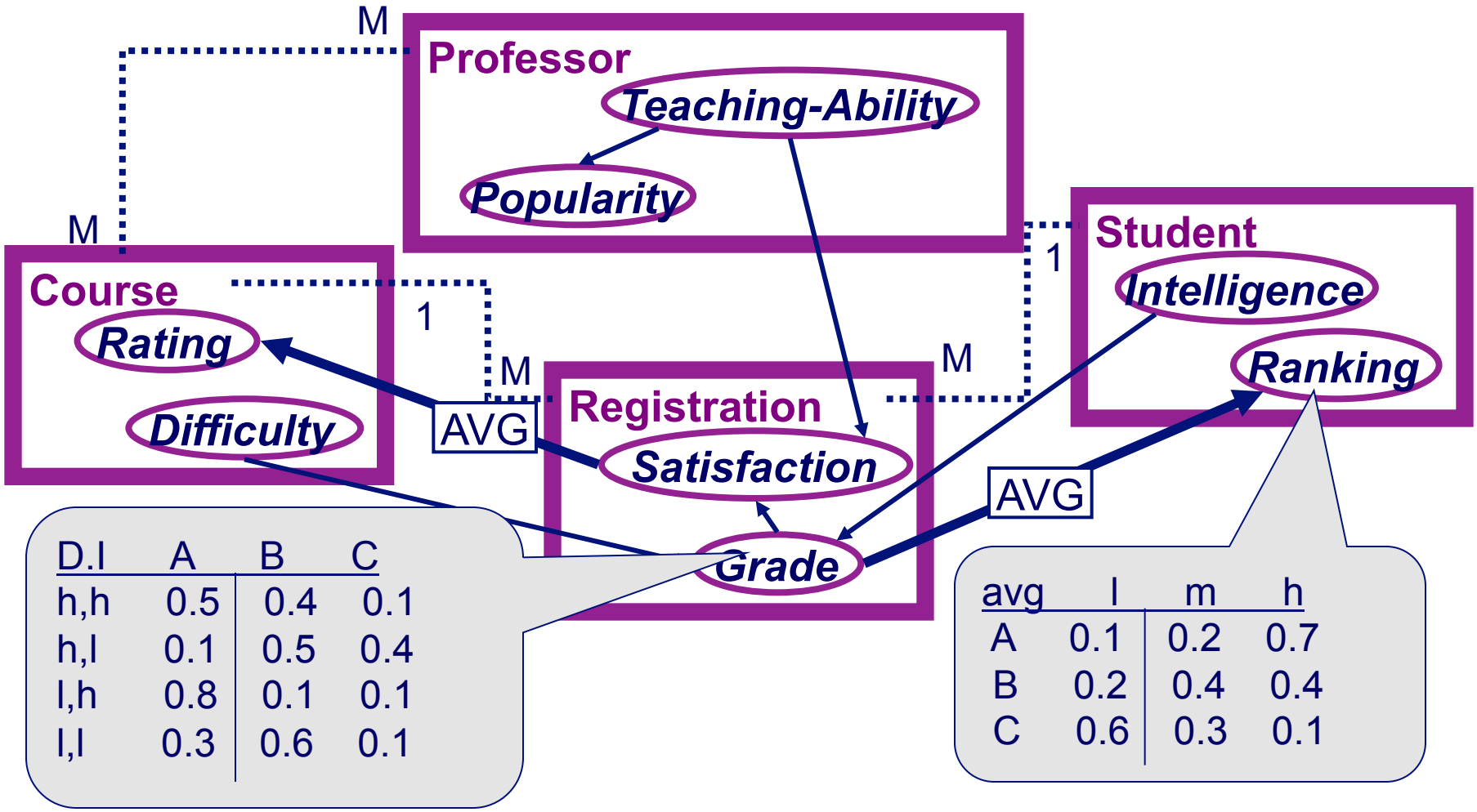
PRM Dependency Structure



Parameters of PRMs

- A PRM contains a *conditional probability distribution* (CPD) $P(X.A|\text{Pa}(X.A))$ for each attribute $X.A$ of each class
- More precisely, let **U** be the set of parents of X.A. For each tuple of values $\mathbf{u} \in V(\mathbf{U})$, the CPD specifies a distribution $P(X.A|\mathbf{u})$ over $V(X.A)$. The parameters in all of these CPDs comprise θ_S

CPDs in PRMs


M
Professor
Teaching-Ability
Popularity
M
Course
Rating
Difficulty
1
M
AVG
Registration
Satisfaction
Grade
M
1
Student
Intelligence
Ranking
AVG
D.I A B C
h,h 0.5 0.4 0.1
h,l 0.1 0.5 0.4
l,h 0.8 0.1 0.1
l,l 0.3 0.6 0.1
avg l m h
A 0.1 0.2 0.7
B 0.2 0.4 0.4
C 0.6 0.3 0.1

Parameters of PRMs

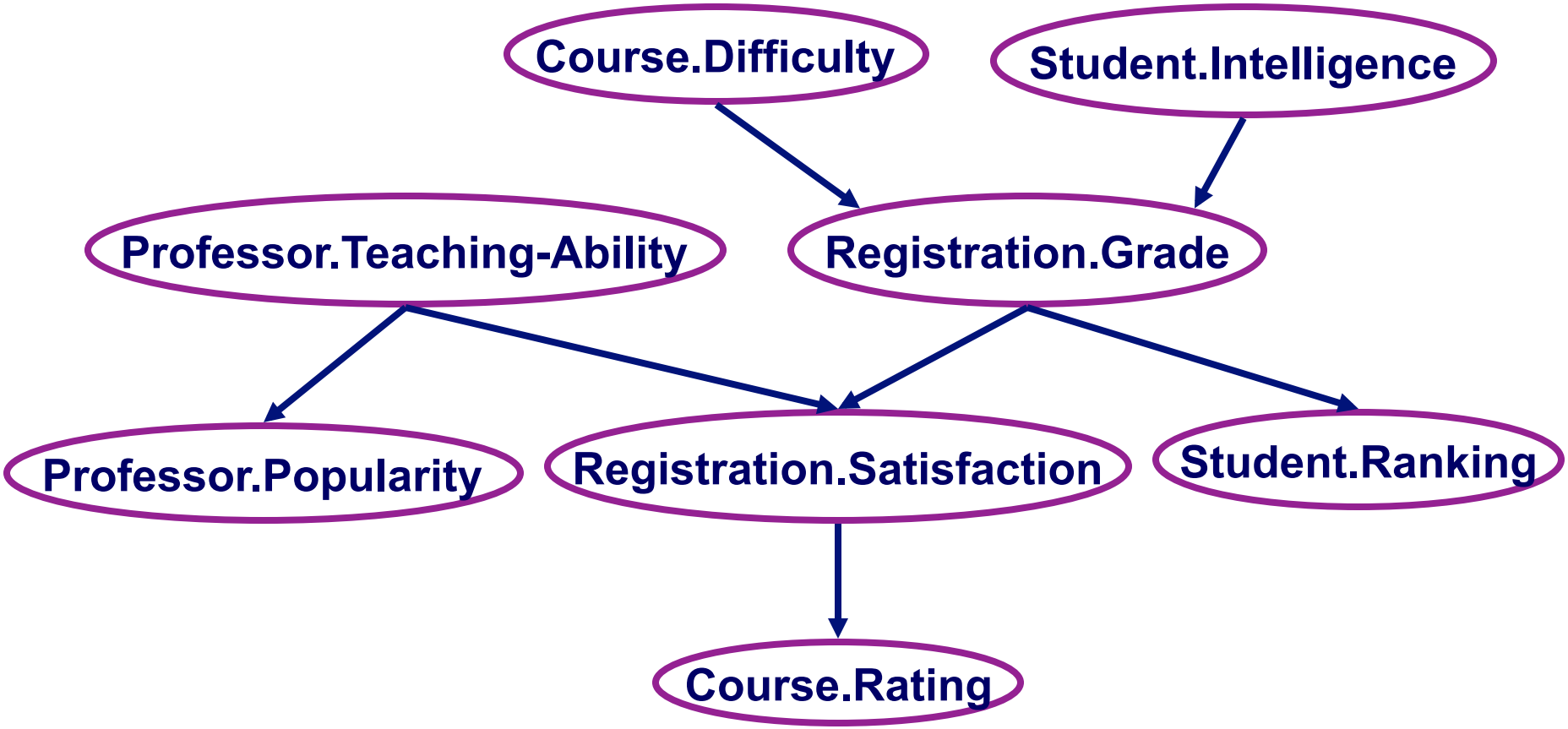
- Given a skeleton structure for our schema, we want to use these local probability models to define a probability distribution over all completions of the skeleton
- Note that the objects and relations between objects in a skeleton are always specified by σ, hence we are disallowing uncertainty over the relational structure of the model

Parameters of PRMs Continued

- To define a coherent probabilistic model, we must ensure that our probabilistic dependencies are acyclic, so that a random variable does not depend, directly or indirectly, on its own value
- A dependency structure $\mathcal{S}$ is acyclic relative to a skeleton σ if the directed graph over all the parents of the variables $x.A$ is acyclic
- If $\mathcal{S}$ is acyclic relative to σ, then the following defines a distribution over completions I of σ:

$$P(I|\sigma,\mathcal{S},\theta_{\mathcal{S}}) = \prod_{X_i} \prod_{A \in A(X_i)} \prod_{x \in O^{\sigma}(X_i)} P(I_{x.a} | I_{Pa(x.a)})$$

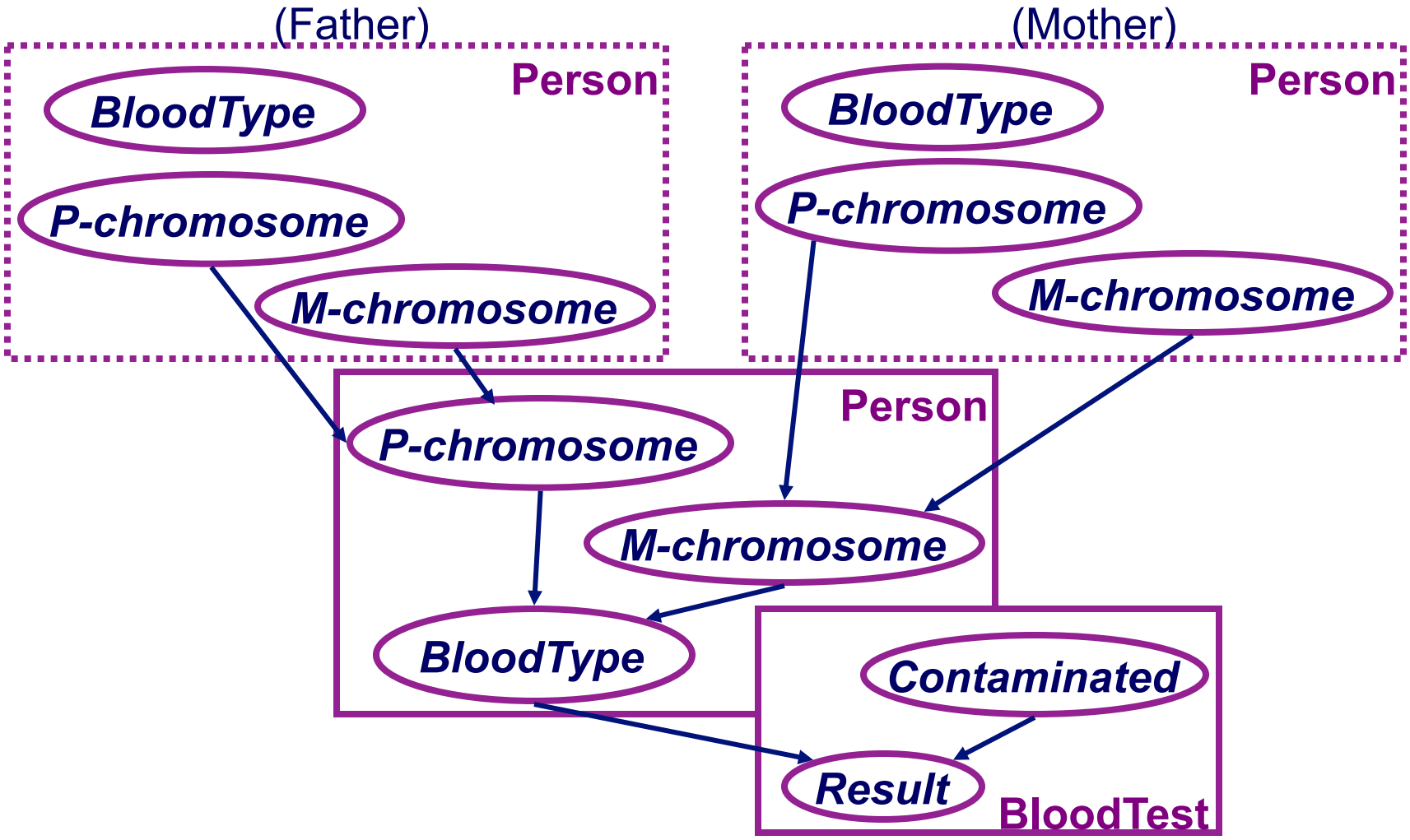
Class Dependency Graph for the University Domain



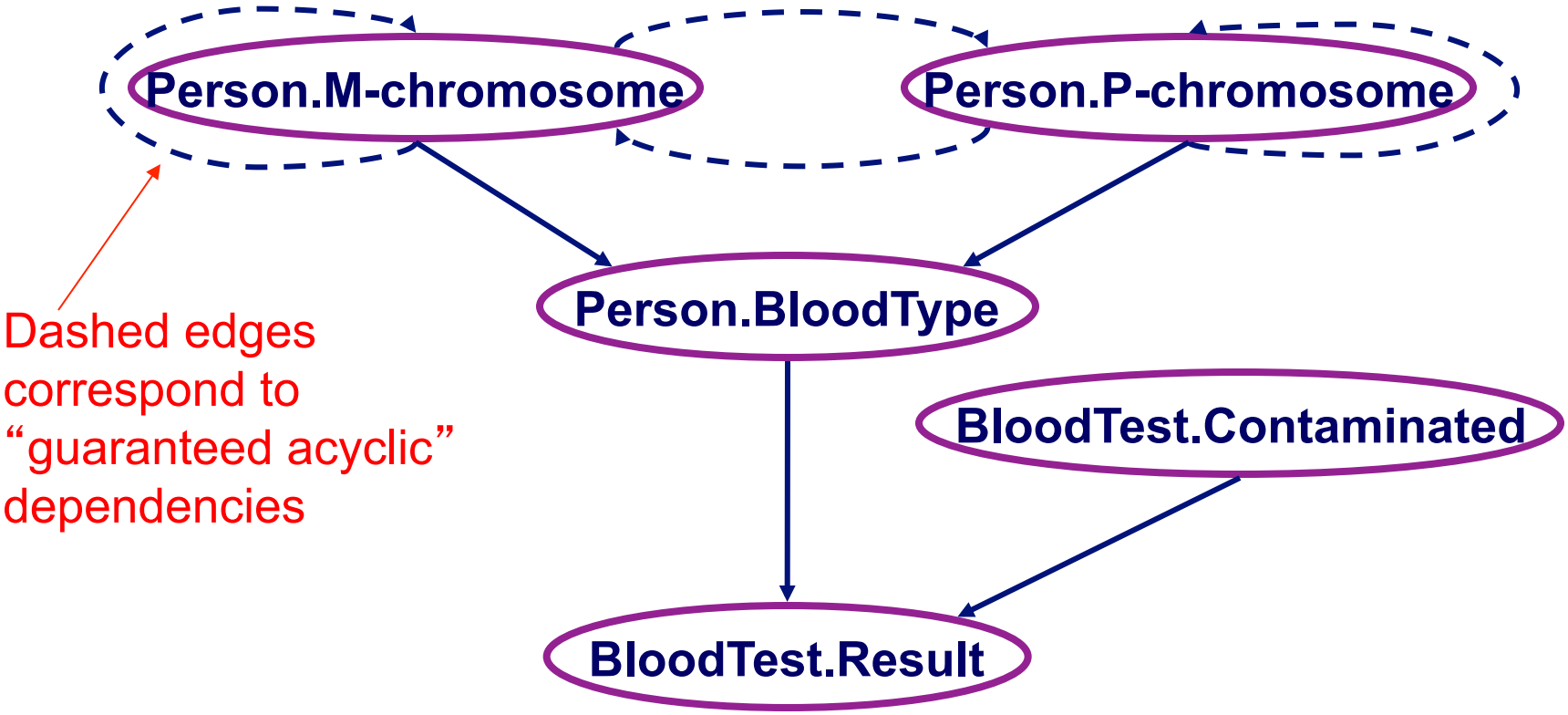
Ensuring Acyclic Dependencies

- In general, however, a cycle in the class dependency graph does not imply that all skeletons induce cyclic dependencies
- A model may appear to be cyclic at the class level, however, this cyclicity is always resolved at the level of individual objects
- The ability to guarantee that the cyclicity is resolved relies on some prior knowledge about the domain. The user can specify that certain slots are *guaranteed acyclic*

PRM for the Genetics Domain



Dependency Graph for Genetics Domain



Applications

- The learning algorithm was tested on one synthetic dataset and two real ones
- Genetics domain – a artificial genetic database similar to the example mentioned earlier was used to test the learning algorithm
- Training sets of size 200 to 800, with 10 training sets of each size were used. An independent test database of size 10,000 was also generated
- A dataset size of n consists of a family tree containing n people, with an average of 0.6 blood tests per person

Applications

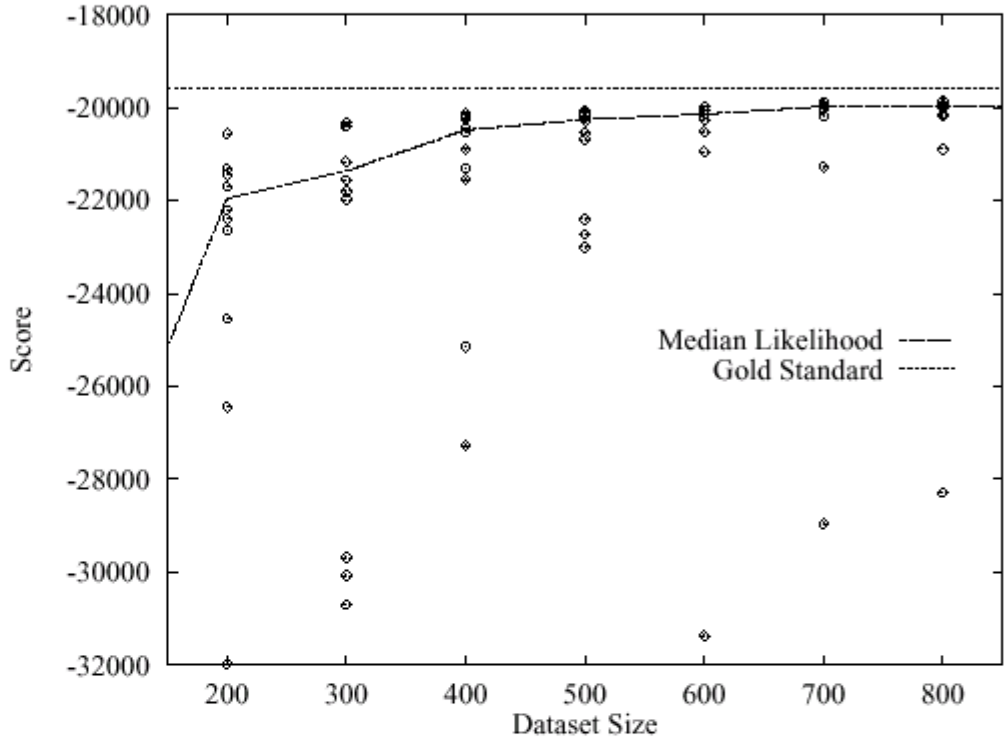


Fig. 1.6. Learning curve showing the generalization performance of PRMs learned in the genetic domain. The x-axis shows the training set size; the y-axis shows log-likelihood of a test set of size 10,000. For each sample size, we show learning experiments on ten different independent training sets of that size. The curve shows median log-likelihood of the models as a function of the sample size.

Applications

- Tuberculosis patient domain – drawn from a database of epidemiological data for 1300 patients from the SF tuberculosis (TB) clinic, and their 2300 contacts
- Relational dependencies, along with other interesting dependencies, were discovered: there is a dependence between the patient's HIV result and whether he transmits the disease to a contact; there is a correlation between the ethnicity of the patient and the number of patients infected by the strain

Applications

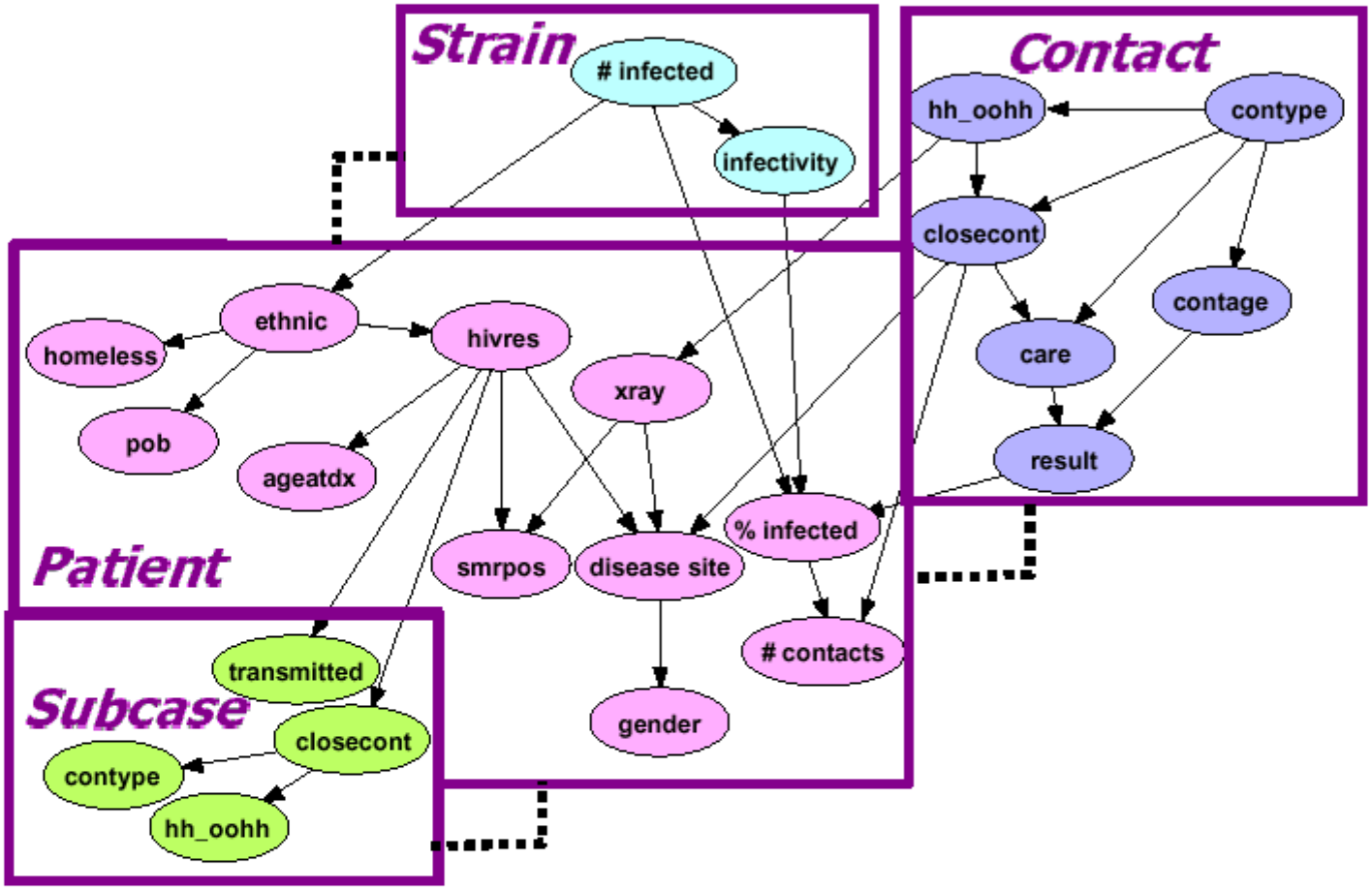


Fig. 1.7. The PRM structure for the TB domain.

Applications

- Company domain – a dataset of company and company officers obtained from Security and Exchange Commission (SEC) data
- The dataset includes information, gathered over a five year period, about companies, corporate officers in the companies, and the role that the person plays in the company
- For testing, the following classes and table sizes were used: **Company** (20,000), **Person** (40,000), and **Role** (120,000)

Applications

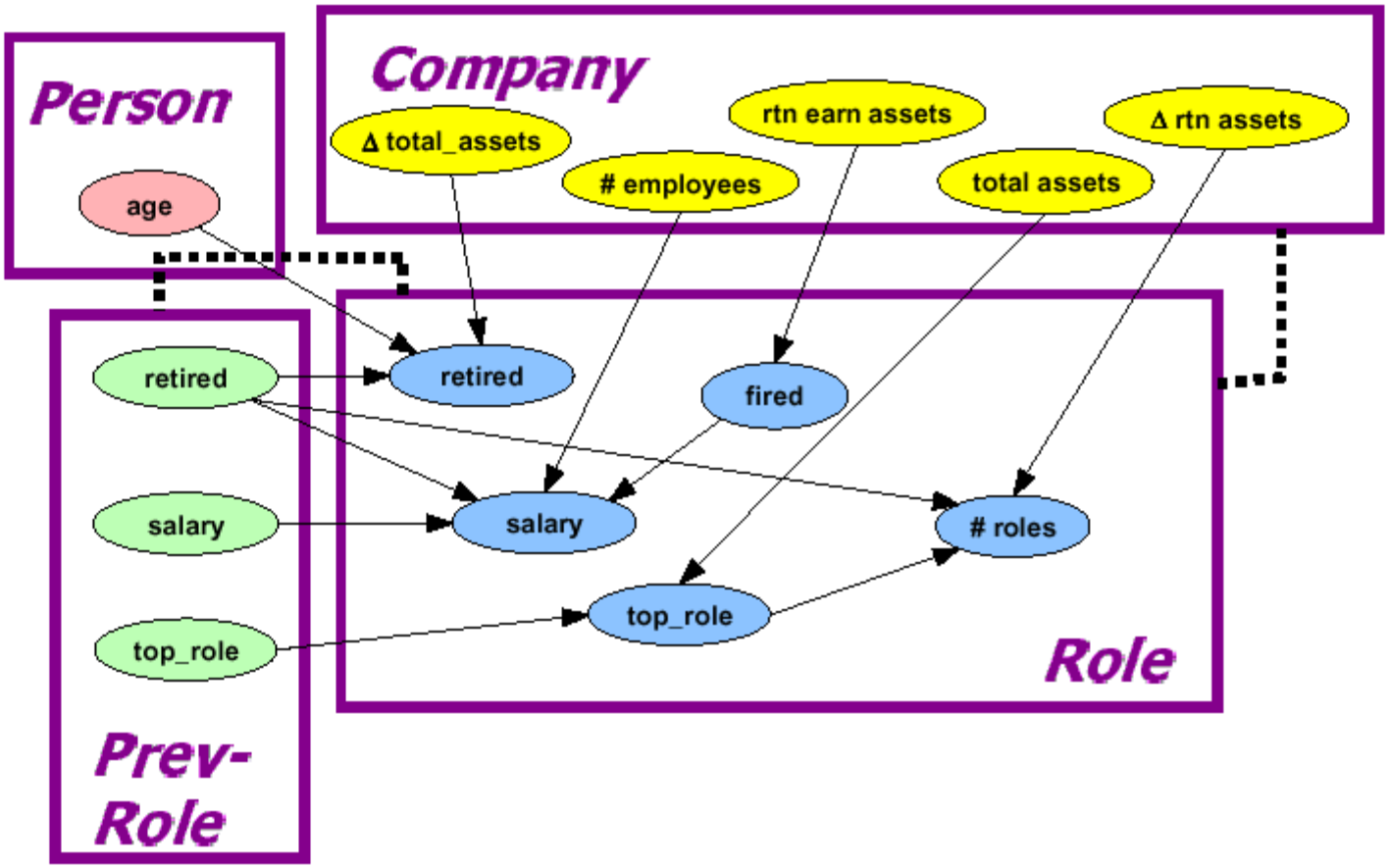


Fig. 1.8. The PRM structure for the Company domain.

Comments on statistical relational learning

- predicate calculus lets us handle rich data representations and complex concepts without all the “right” features pre-defined
- but predicate calculus labels every statement true or false: no uncertainty
- SRL combines:
  - ability to learn from relational data: graphs, relational databases
  - explicit representation of uncertainty: learn a probability distribution
- Markov logic and plate models (e.g., PRMs) are two successful approaches
- an advantage of MLNs is ability to change representation, or “view” of a relational database (change the fields and/or tables)
- advantages of plate models are faster inference and, consequently, faster learning
- other representations and approaches as well