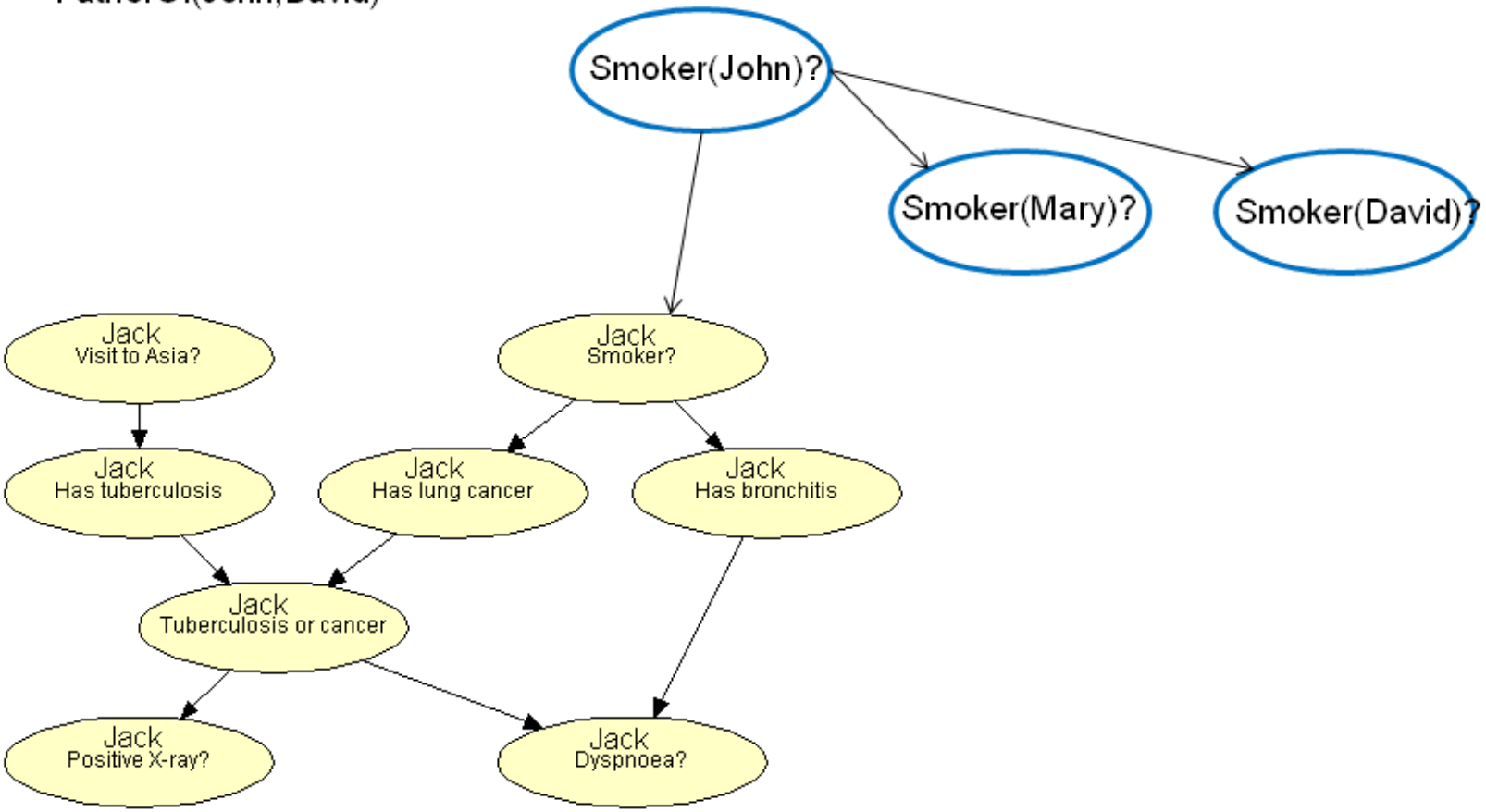
Bayes Nets for Relational Learning

Volker Tresp
Summer 2015

Bayes Nets for Relational Learning

- Consider the Chest Clinic Example: Let's assume that Jack's father is in the database and that he is a smoker. This should influence the prior probability of Jack being a smoker
- Now let's assume that we do not know if Jack's father is a smoker but that his other children are smokers. This can be seen as positive evidence that Jack's father is a smoker and thus that Jack is a smoker as well
- We will see that by considering binary relations such as *fatherOf* or *friendOf* we can get dependencies between nodes of potentially all patients, their friends and relatives: we cannot treat each patient independently of the others! *Technically, the whole observed world becomes one data point*

FatherOf(John, Jack)
FatherOf(John, Mary)
FatherOf(John,David)

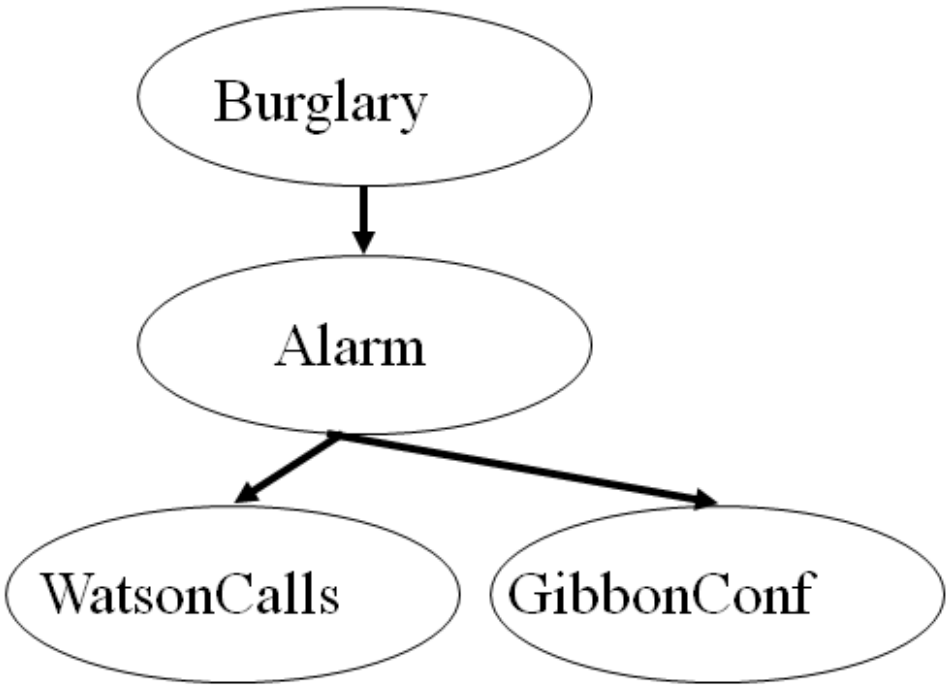


A Hierarchy of Bayes Nets

- Relationships are naturally being considered in first-order logic (FOL) and in relational databases (RDBs)
- Let's look at three relational complexities:
  - A propositional Bayes net
  - A Bayes net with unary relations/predicates (the classical learning scenario)
  - A Bayes net with binary (or higher-order) relations/predicates

I. Propositional Bayesian Nets

- The simplest case is a *propositional Bayesian network*
- Example: Holmes Network
- Nodes in a propositional Bayesian network represent atomic propositions as *Alarm*, *WatsonCalls*, *DaughterCalls*
- Closely related to propositional logic (PL)


Burglary
Alarm
WatsonCalls
GibbonConf

II. Template Bayes Nets with Unary Predicates

- This is a typical Bayes net used in applications
- A Bayesian network does not only make statements about a single person (Jack) but about a whole set of entities (all patients); a Bayesian network is a template
- Example:

$$\forall z.\ P(\textit{bronchitis}(z)|\textit{smoker}(z)) = \theta_{bron,t}$$

  where $t \in \{0, 1\}$ indicates if *smoker*(z) is true or false
- *Jack, John, ...* are constants (e.g., objects, entities)
- z is a variable that represents constants (here: *Jack, John ...*)
- $\forall$ (for all) is a quantifier; the other one is $\exists$ (there is)
- *smoker* is a predicate and *smoker*(z) is an atom
- We call *bronchitis*(z) the head and *smoker*(z) the body of the probabilistic rule

Rules with more than one Input

- Example:

$$\forall z.\ P(\mathit{dyspnea}(z)|\mathit{toc}(z), \mathit{bronchitis}(z)) = \theta_{dyspnea,t_1,t_2}$$

where $t_1, t_2 \in \{0, 1\}$ indicates if the corresponding terms in the body are true or false

Ground Bayes Net

- All possible ground atoms together with the probabilistic dependencies (directed links, quantifications of dependencies) derived from the template Bayesian network form a propositional Bayesian network, called the *ground Bayesian network*; in particular, there are no variables in the ground Bayes net

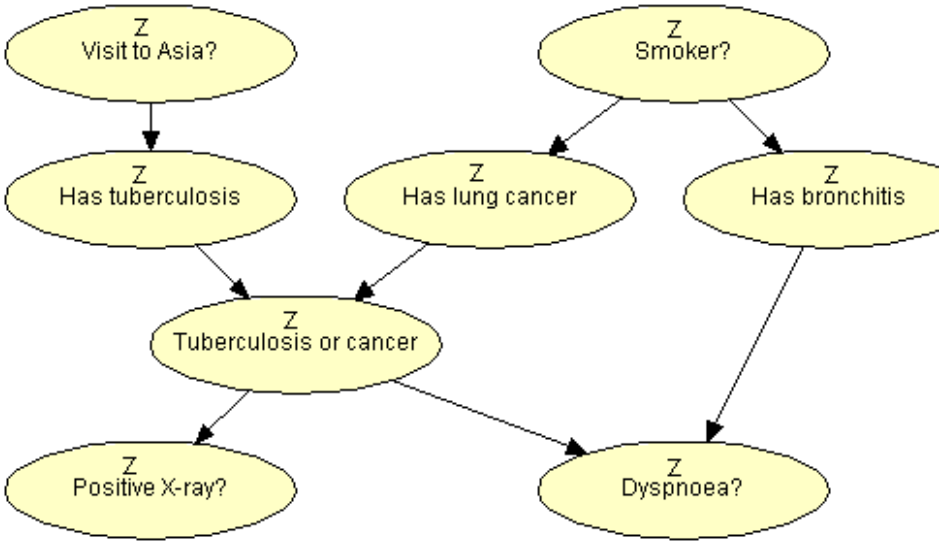
Template Bayes Net

$\forall z$

atom: *smoker(z)*

Probabilistc rule:

$$\forall z : P(bronchitis(z) \mid smoker(z)) = \theta_{bronch,t}$$

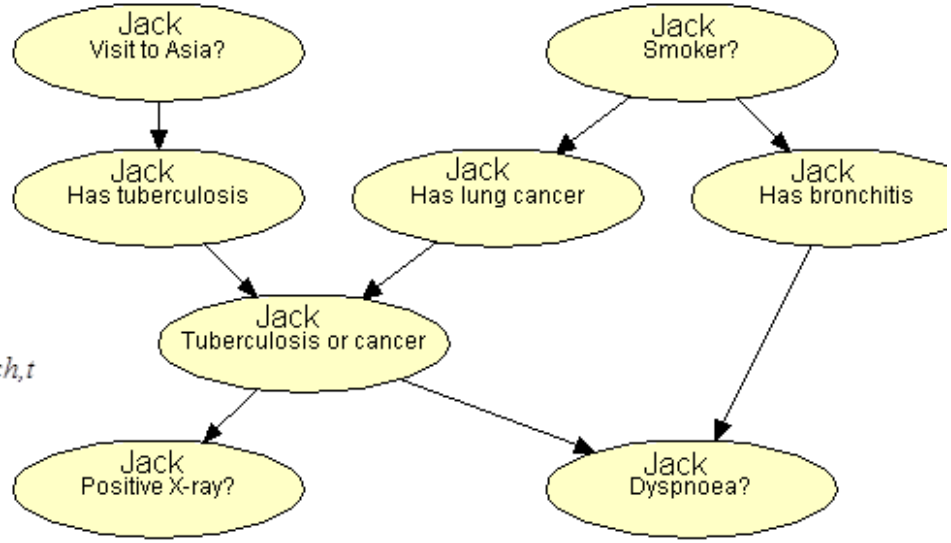


Ground Bayes Net

ground atom: *smoker(Jack)*

Probabilistc ground rule:

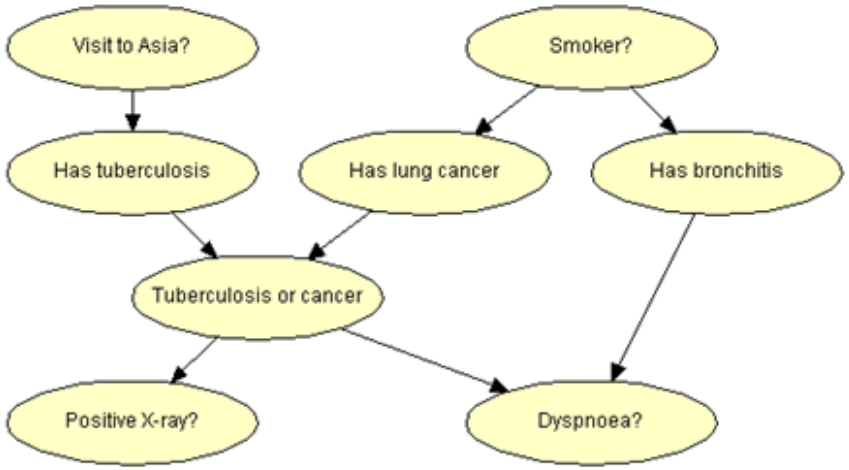
$$P(bronchitis(Jack) \mid smoker(Jack)) = \theta_{bronch,t}$$



Possible World

- $\mathcal{X}$ is now the set of all ground atoms that can be formed by all known constants (here: all patients) and all predicates
- $\mathcal{X} = x$ assigns truth values to each ground atom and is called a possible *world* (Herbrand interpretation). Thus $\mathcal{X} = x$ defines a complete training data set
- Each ground atom becomes a node in a Bayes net. With N constants and M_u unary predicates, there are NM_u ground atoms / nodes
- With only unary predicates, a world can be displayed as a binary matrix with entities as rows and unary relations as columns
- In Bayes Nets with only unary predicates, the ground Bayes net decomposes into independent Bayes nets, e.g., one for each patient. Under some assumptions, each patient might be interpreted as defining an independent data point

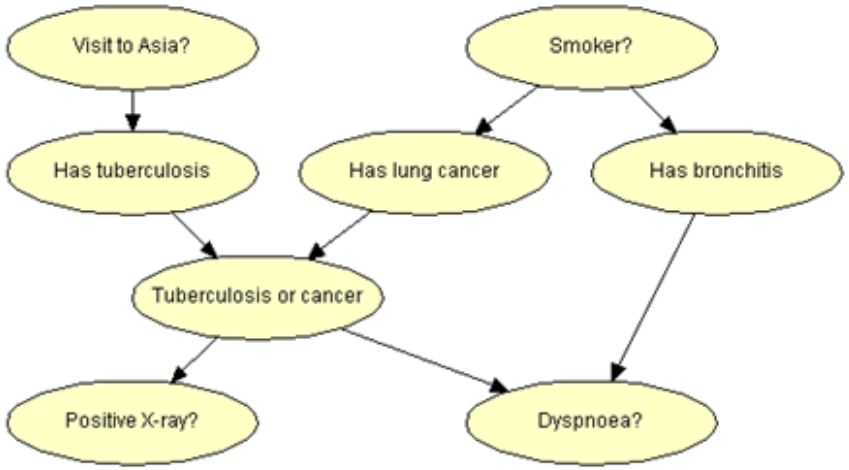
A Possible World



A relational representation of a possible world: relations (names) correspond to predicates and constants in the relations indicate for which constants the corresponding atoms are true. With a closed-world assumption all other ground atoms are assumed false. In an opern world assumption one would distinguish between grround atoms that are false and ground atoms whose truth value is unknown

A	S	T	C	B	ToC	X	D
Jack	John	Mary	Jack	Jack	Jack	Jack	John
John			Mary	Mary	Mary	Mary	

A Possible World

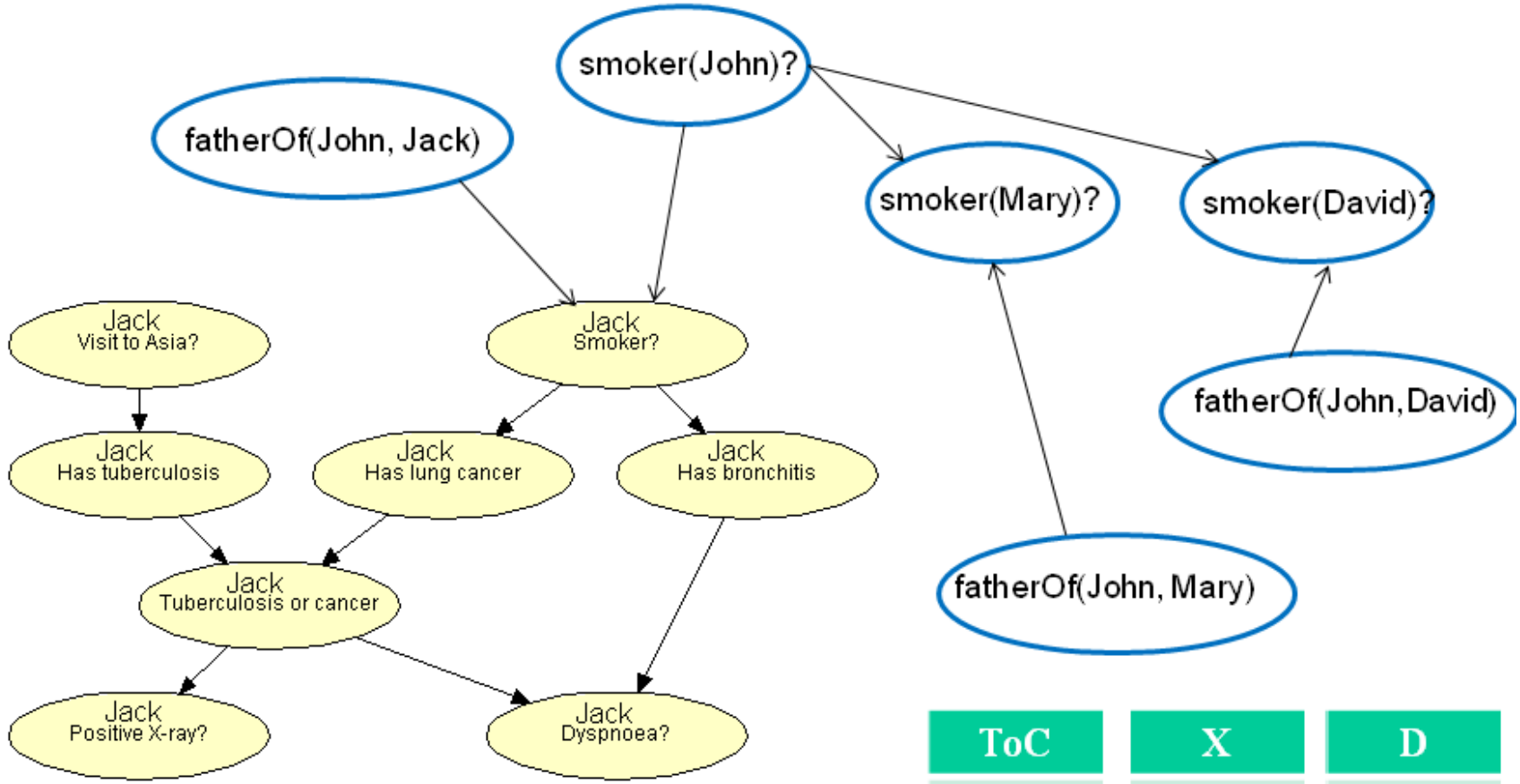


data matrix representation of a possible world

	A	S	T	C	B	ToC	X	D
Jack	1	0	0	1	1	1	1	0
John	1	1	0	0	0	0	0	1
Mary	0	0	1	1	1	1	1	0

III. Bayes Nets with Binary and Unary Predicates

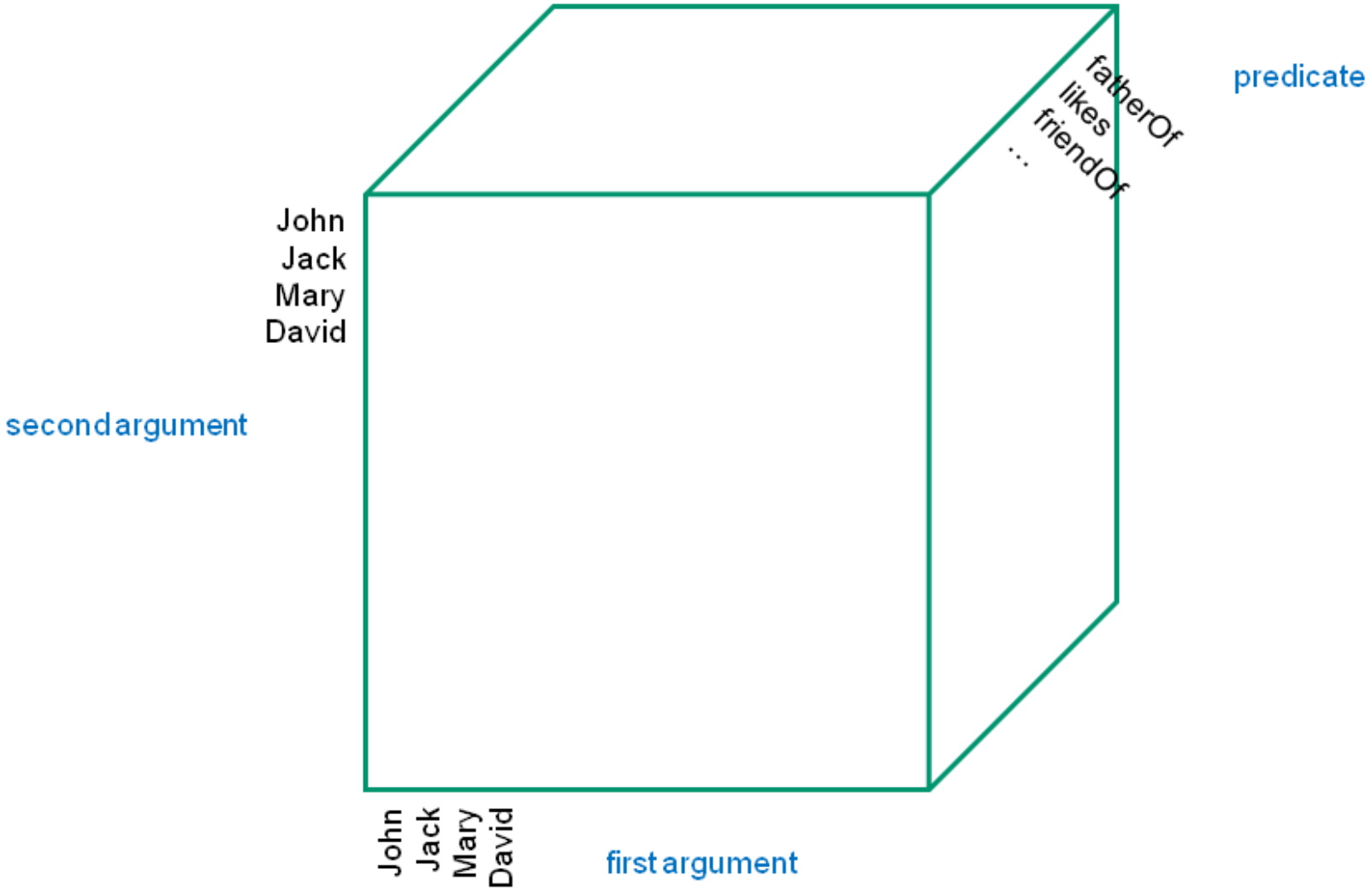
- Binary predicates represent relationships between entities: example: *fatherOf(z, y)* In the convention we are using the first argument is the subject and the second one the object
- Again: $\mathcal{X}$ is the set of all ground atoms that can be formed by all known constants and all predicates. But note that atoms can now have two arguments, e.g., *fatherOf(John, Jack)*
- In a possible world we obtain $NM_u + N^2 M_b$ ground atoms, where M_b is the number of binary relations
- For binary predicates binary relations are introduced
- A possible world can be represented as a tensor (3-way array)



A	S	T	C	B
Jack	John	Mary	Jack	Jack
John	David	David	Mary	Mary

ToC	X	D
Jack	Jack	John
Mary	Mary	

fatherOf	
John	Jack
John	Mary
John	David


predicate
fatherOf
likes
friendOf
...
John
Jack
Mary
David
second argument
John
Jack
Mary
David
first argument

Probabilistic Rules with Binary Predicates

- Smoking of ones father is predictive:

$$\forall z.\ P(\mathit{smoker}(z)|\exists t.\mathit{fatherOf}(t,z) \wedge \mathit{smoker}(t)) = \theta_{\mathit{smoker},t}$$

- Smoking of ones father is predictive and smoking of at least one friend is predictive:

$$\forall z.\ P(\mathit{smoker}(z)|\exists t.\mathit{fatherOf}(t,z) \wedge \mathit{smoker}(t), \exists u.\mathit{friendOf}(u,z) \wedge \mathit{smoker}(u))$$

$$= \theta_{\mathit{smoker},t_1,t_2}$$

- *friendOf* predicts *likes*

$$\forall z.\forall t.\ P(\mathit{likes}(z,t)|\mathit{friendsOf}(t,z)) = \theta_{\mathit{likes},t}$$

Type Constrains

- One needs to apply constraints such that there is only one rule applied to each head ground atom and that no loops in the ground Bayes net occurs
- Often the first constraint is implemented as a type constraint, leading to typed predicates
- The probabilistic *smoker* rule is only applied to patients

$$\forall z.patient(z) : \; P(\mathit{smoker}(z)|\exists \mathit{fatherOf}(t,z) \wedge \mathit{smoker}(t)) = \theta_{\mathit{smoker},t}$$

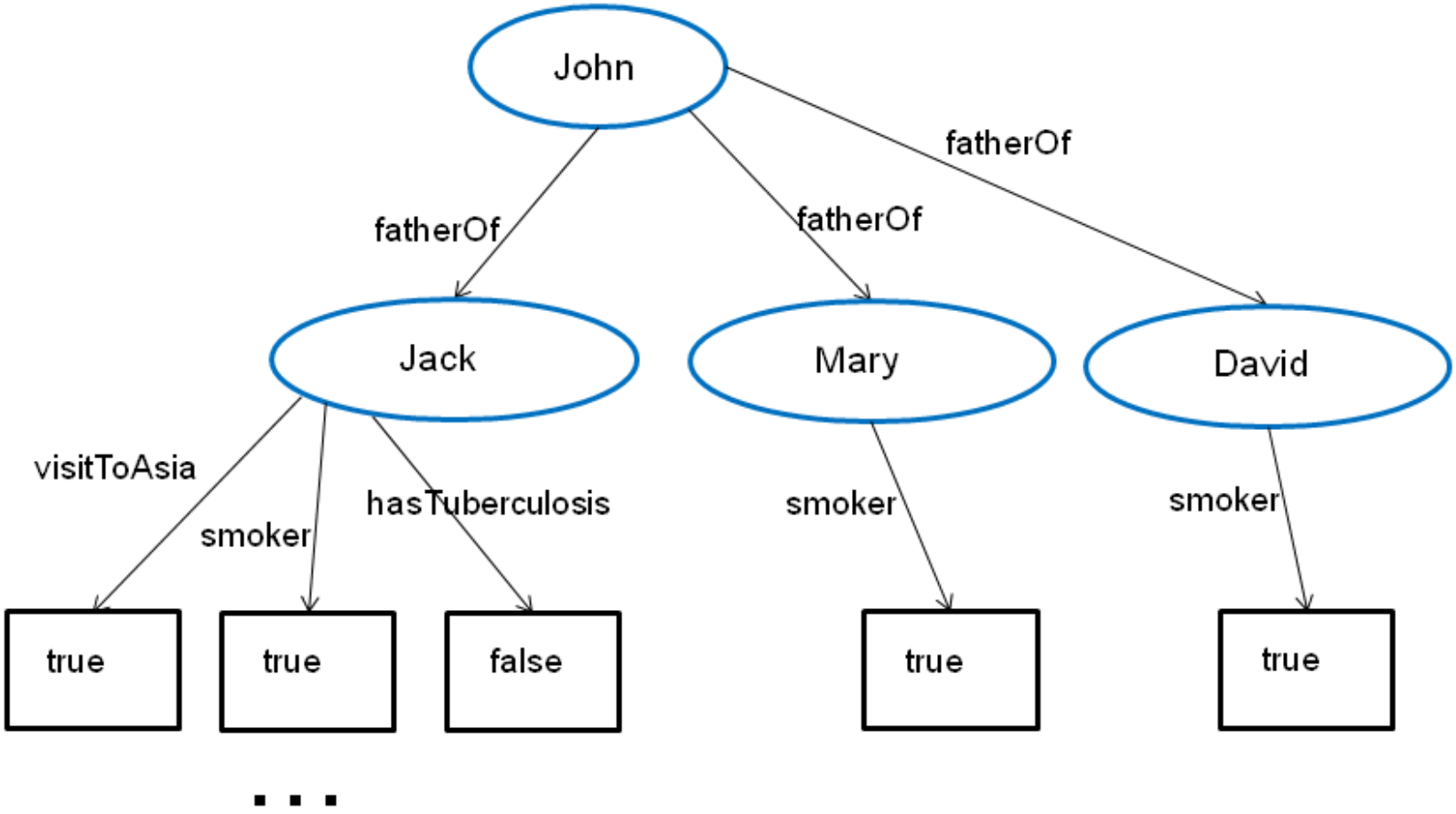
- The probabilistic *headacheMedication*rule is only applied to medication

$$\forall z.\mathit{medication}(z) : \; P(\mathit{headacheMedication}(z)|\mathit{containsAspirin}(z)) = \theta_{\mathit{headacheMedication},t}$$

- As an alternative (e.g., in Bayesian Logic Programming) one permits several probabilistic rules for a given head ground atom, and one then uses a form of a combination rule (e.g., Noisy-Or) (note that in logic programming the combination rule is very simple: if one rule "fires" and proofs the head to be true, then the head is true, no matter if the other rules fire)

Triple Graphs

- The training data is represented as a set of unary and binary tables (relations)
- Alternatively we write a ground atom as a $<$*subject*, *predicate*, *object*$>$ triple, for example *fatherOf(John, Jack)* becomes $<$*Jack*, *fatherOf*, *John*$>$ and form a graph where entities are nodes and a triple is represented as a directed link between subject and object. The link is labelled by the predicate
- The resulting graph is called a triple graph. Knowledge graphs (DBpedia, Yago, Freebase, Google Knowledge Graph) are special triple graphs. Another example is the RDF (resource description framework) graph used in the linked open data (LOD) cloud
- But don't confuse the triple graph with the ground Bayes net: The nodes of the Bayes net would correspond to the links in the triple graph!


John
fatherOf
fatherOf
fatherOf
Jack
Mary
David
visitToAsia
smoker
hasTuberculosis
smoker
smoker
true
true
false
true
true
. . .

Probabilistic Inference in Bayes Nets with Binary Relations

- Situation: the truth values of some ground atoms are unknown
- Recall that inference might propagate information in the whole ground Bayes net
- Exact inference is typically not feasible
- Usually, some form of approximate inference is used (loopy belief propagation, MCMC, mean field inference)

Missing Entities, Predicates, and New Worlds

- Important entities might be missing (Jack's father is not in the knowledge base)?
- Important predicates are missing?
- How can we apply an existing model to a new world?
- All these issues can be important in a particular application!

Parameter Learning

- Learning the θ-parameters with complete data is easy
- With missing information some form of EM-learning can be applied

Structural Learning

- In addition to the usual issues in structural learning, we are faced with the problem of searching for interesting candidate bodies
- The next figures shows two learned relational structures using the PRM formalism

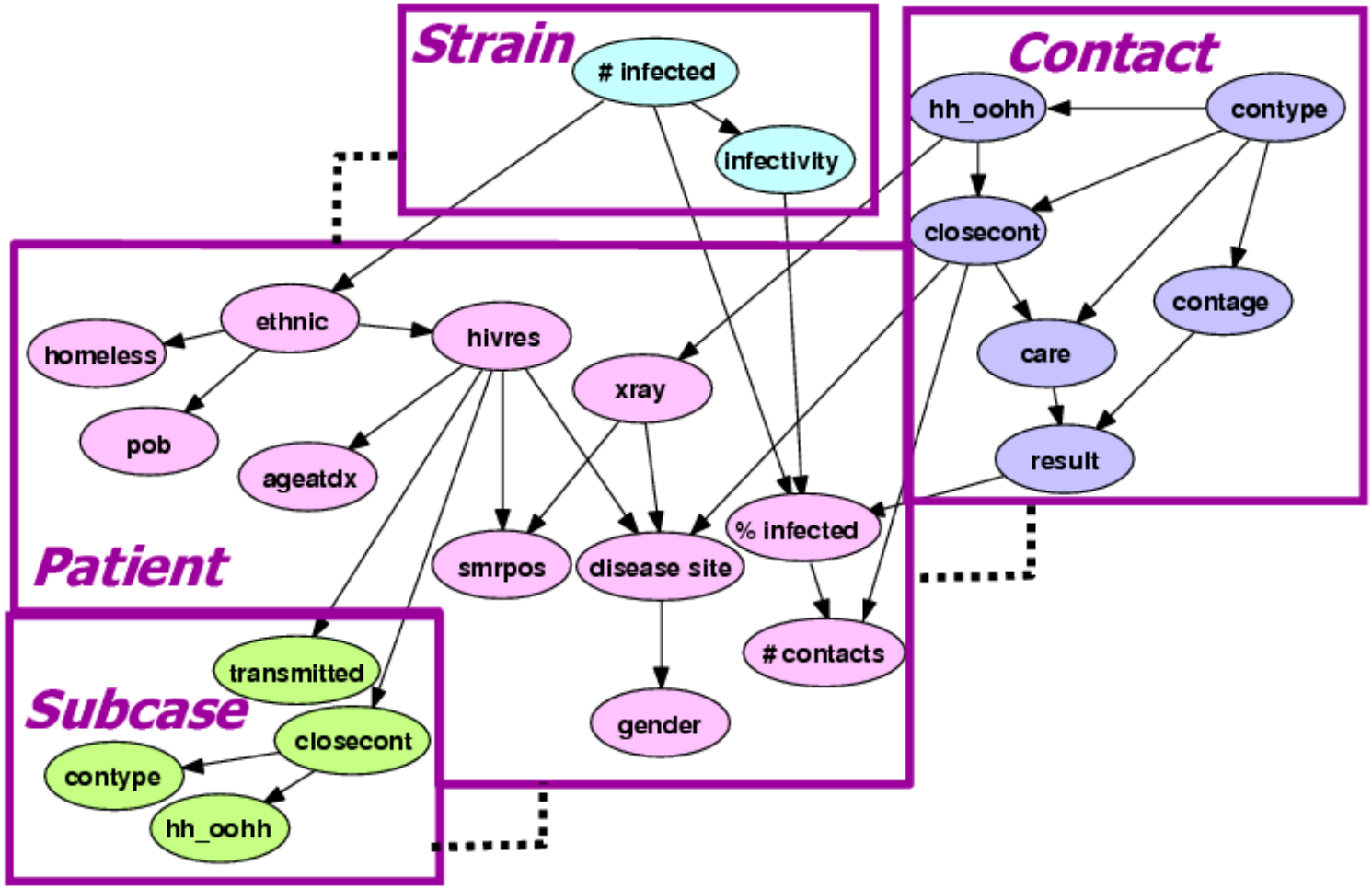


Fig. 1.7. The PRM structure for the TB domain.

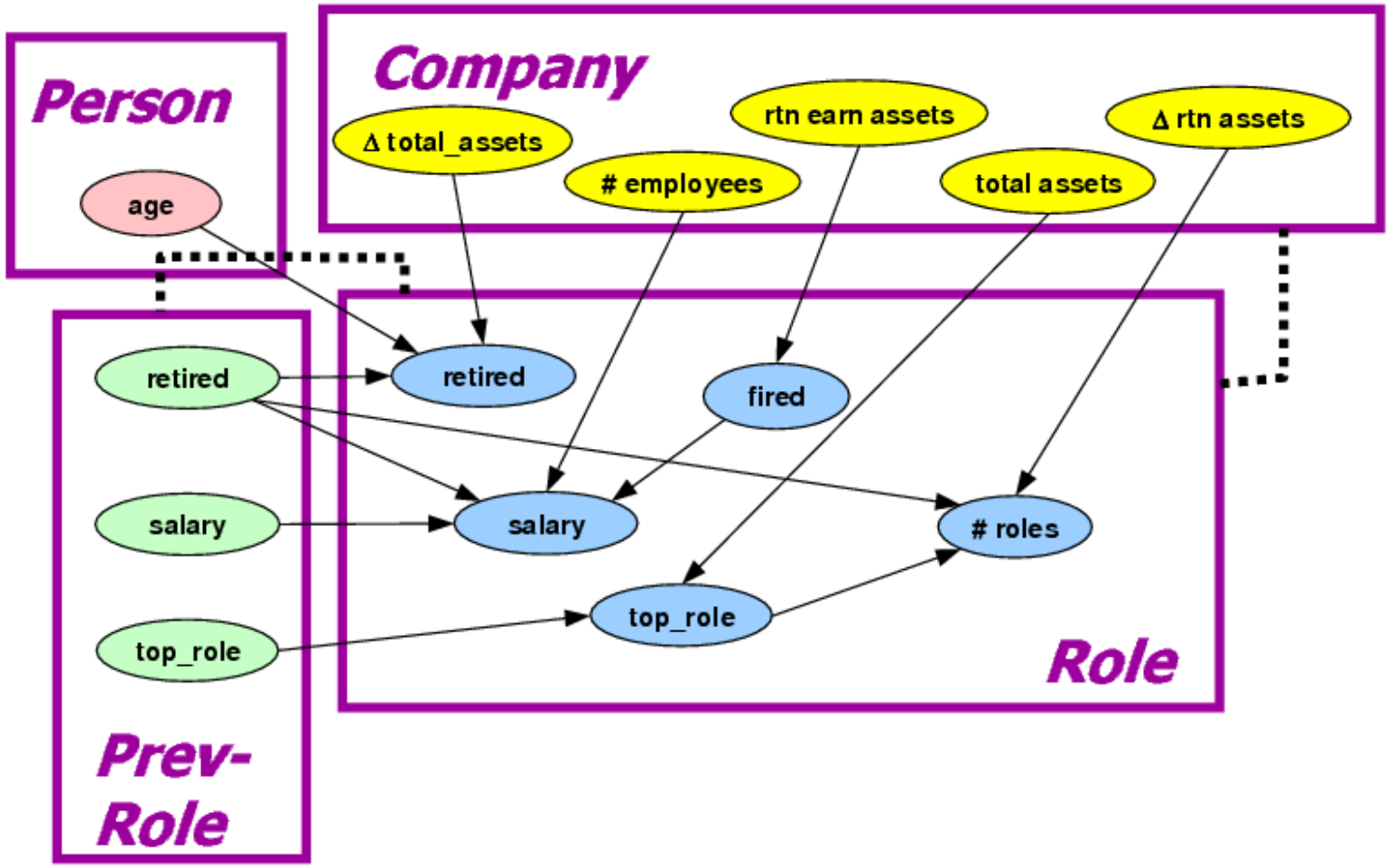


Fig. 1.8. The PRM structure for the Company domain.

Connection to Logics

- If we replace the probabilistic rules by deterministic ones we obtain *logic programming* and *rule-based systems*; logical constraints are also used in *knowledge representation*.
- A general logic framework is defined by first-order logic (FOL) where general logical expressions are used (not just rules)

Relational Learning

- Machine learning for deterministic or close-to deterministic domains: *Inductive Logic Programming (ILP), Inductive Databases, Inductive Reasoning*
- *Probabilistic Databases* deal with the representation and querying of uncertain facts
- Learning with probabilistic relational domains: *Probabilistic Relational Model (PRM), Probabilistic Entity-Relationshop Models (PER), Plate models, Markov Logic (ML), Stochastic Logic Programs, Bayesian Logic Programming, Relational Dependency Networks*

Concluding Remarks

- Relational Learning permits the application of machine learning to databases and knowledge representations
- So far the success in applications was limited: Markov Logic is the most popular approach being used
- Alternative approaches build simple generative models from raw data (Infinite Hidden Relational Models, RESCAL) and are quite promising
- RESCAL is build on a joint factorization of the relational matrix and the relational tensor and is used for learning with knowledge graphs (DBpedia, Google Knowledge Graph)