

Lernen mit Logik und Relationen

Volker Tresp

Mustererkennung

- Betrachten wir Klassifikatoren, deren Zielgröße mit Sicherheit vorhergesagt werden kann, so kann man diese als Wenn-Dann Regel interpretieren:
WENN die Eingangsmerkmale gewisse Werte annehmen, DANN gehört das Muster zu Klasse i
- Im deterministischen Grenzfall können Bayes Netze als ein System von deterministischen Klassifikatoren betrachtet werden: für jede Variable X_i :
WENN $\text{par}(X_i) = j$, DANN ist $X_i = j$
- Ein Datenpunkt im Klassifikator oder auch im Bayes Netz bezieht sich auf ein Typus von Objekt. Beispiele: : Patient, Student, Kunde, ...
- In der Logik nennt man solche Regeln *propositional rules*: Sie enthalten keine Variablen und keine Quantifizierer (\forall, \exists)
- Erst sogenannte Regeln erster Ordnung enthalten Variable (d.g. nehmen explizit Bezug auf Objekte); so besteht die Turing äquivalente Programmiersprache PROLOG aus Regeln erster Ordnung; Foil (Quinlan: Foil is a first order learning system that uses

information in a collection of relations to construct theories expressed in a dialect of prolog)

- Inductive Logic Programming (ILP) hat zum Ziel, solche Regeln zu lernen; dies bedeutet, dass man ein PROLOG Programm lernt, bzw. ein Programm einer Turing Maschine erlernt
- Kognitiver Bezug: denkbar ist, dass Turing mächtige Maschinen eine Rolle im Sprachzentrum, Bewegungszentrum und anderen höheren kognitiven Fähigkeiten spielen; sicherlich spielt Lernen in der Einstellung dieser Maschinen eine wichtige Rolle

Beispiel

- Regeln erster Ordnung enthalten Variable und erhalten dadurch erhöhte Mächtigkeit:
- Beispiel:

IF Parent(x, y) THEN Ancestor(x, y)

IF Parent(x, z) AND Ancestor(z, y) THEN Ancestor(x, y)

Wie kann man dies mit einem Entscheidungsbaum darstellen?

Vorteile regelbasierter Systeme

- Mächtigkeit
- Eine Konklusion muss nur einmal als wahr bewiesen werden und kann dann von weiteren Regeln verwandt werden
- Man muss nur einen möglichst großen Satz logischer Regeln aufstellen; man muss jedoch nicht ein gesamtes Weltmodell bauen
- Interpretierbarkeit

Nachteile regelbasierter Systeme

- Kein eleganter Umgang mit Unsicherheit
- Man wird nicht gezwungen, ein komplettes Weltmodell aufzubauen
- Schlussfolgerungen sind nur erlaubt von Wenn zu Dann und nicht invers (wie bei Bayes Netzen)

Sequential Covering Algorithms

- Zunächst beschäftigen wir uns mit dem Erlernen von logischen propositionaler Regel (ohne Variable)
- Ziele: Regeln des Types:

WENN (Bedingung = wahr) DANN (Folgerung = wahr)

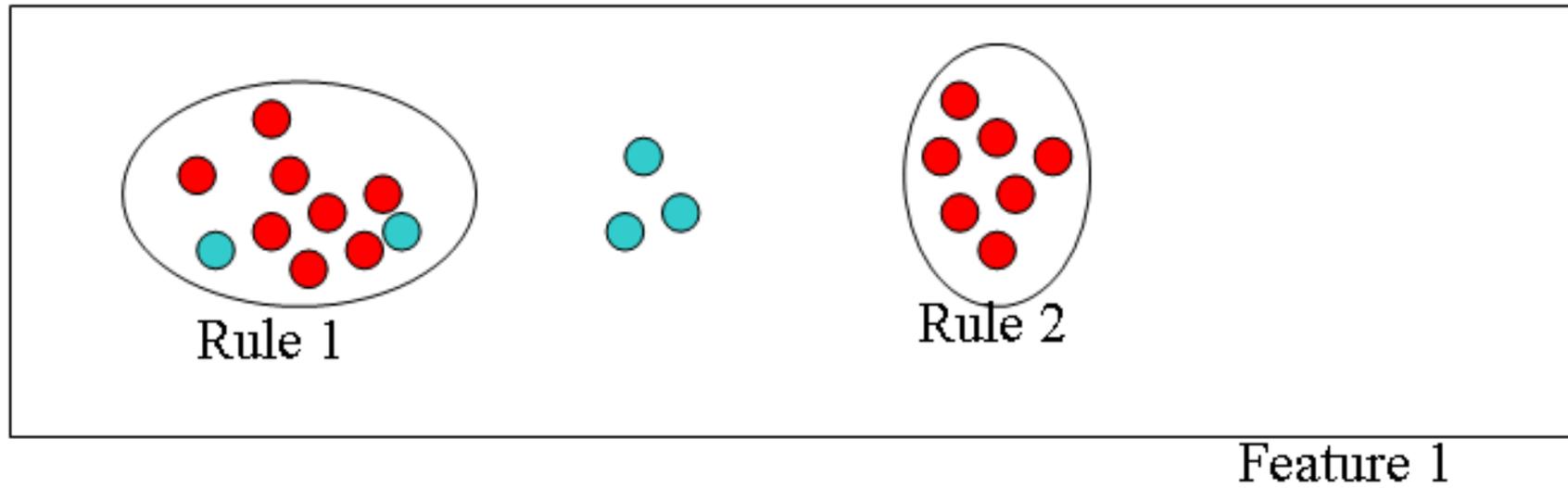
Lerne eine Regel

- Vorgegeben: positive und negative Beispiele
- Ergebnis: eine Regel, die möglichst viele der positiven Beispiele abdeckt (cover), aber keine (oder wenige) der negativen Beispiele
- Anforderung: Hohe Akkuratheit (aber nicht notwendigerweise hohe Abdeckung)

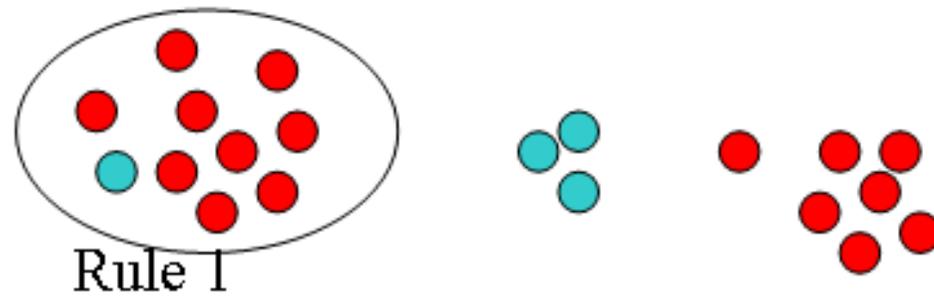
Lerne viele Regeln

- Lerne eine Regel, beseitige die Daten, die die Regel abdeckt (cover) und iteriere (Sequential Covering Algorithm)

Feature 2



Zwei Regeln des Sequential Covering Algorithm



Regel 1 hat 90% Akkuratheit und 50% Abdeckung
Im Allgemeinen kann die Abdeckung klein sein,
solange die Akkuratheit groß ist

Sequential-Covering- Algorithmus

- Sequential-Covering(class,attributes,examples,threshold T)
- RuleSet = 0
- Rule = Learn-one-rule(class,attributes,examples)
- While (performance(Rule) > T) do
 - a. RuleSet += Rule
 - b. Examples = Examples \ { ex. classified correctly by Rule }
 - c. Rule = Learn-one-rule(class,attributes,examples)
- Sort RuleSet based on the performance of the rules
- Return RuleSet

Bemerkungen

- Die gierige Suche findet nicht notwendigerweise den besten Regelsatz
- Man lernt einen Satz von Regeln, den man schreiben kann als

$$\text{Ziel} \leftarrow (A_{1,1} \wedge A_{1,2} \dots) \vee (A_{2,1} \wedge A_{2,2} \dots) \vee (A_{3,1} \wedge A_{3,2} \dots) \dots$$

Dies ist eine Oder-Verknüpfung von Und-Verknüpfungen (*Disjunction of Conjuncts*)?

- Sequentiell werden die positiven Beispiele abgedeckt (bis zu einem Schwellwert)
- (Dasselbe wie die Datenbank:

$$[\text{Ziel} \leftarrow (A_{1,1} \wedge A_{1,2} \dots)] \wedge [\text{Ziel} \leftarrow (A_{2,1} \wedge A_{2,2} \dots)] \wedge$$

$$[\text{Ziel} \leftarrow (A_{3,1} \wedge A_{3,2} \dots)] \dots$$

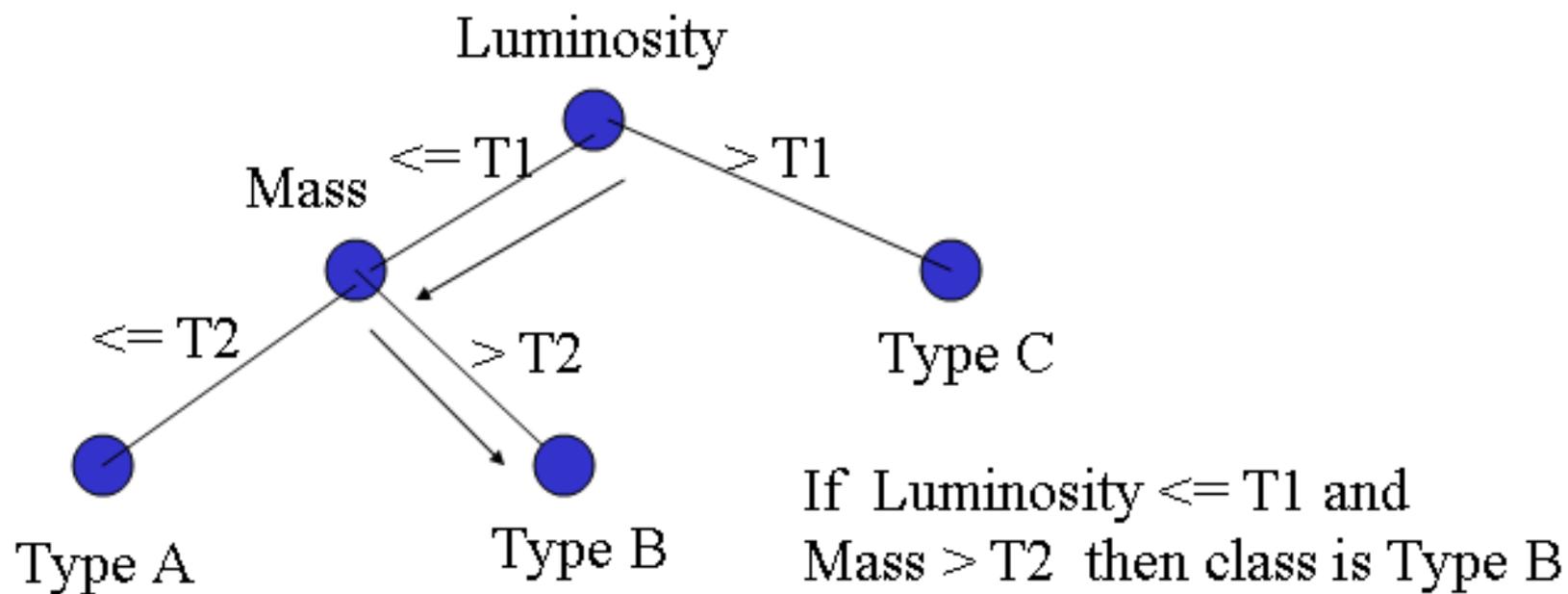
)

Strategie

- Ein möglicher Algorithmus: Entscheidungsbäume wie CART (oder ID3), aber folge nur dem Zweig mit den meisten positiven Beispielen (d.h., es wird kein ganzer Baum aufgebaut!)
- Besser, Beam search: der Algorithmus behält eine Liste der besten Alternativen
- Viele Variationen dieses einfachen Algorithmus sind vorgeschlagen worden; Beispiel CN2-Algorithmus (Clark und Niblet)

Lerne eine Regel

Lerne einen Entscheidungsbaum, aber folge nur dem Zweig mit dem besten Kosten



Relationen und Logik

- Beispiel, was man mit propositional Regeln nicht (so einfach) lernen kann

$$\textit{Enkelin}(x, y) \leftarrow \textit{Vater}(y, z) \wedge \textit{Vater}(z, x) \wedge \textit{Weiblich}(y)$$

- Hier sind
 - Hier sind *Enkelin*, *Vater* Prädikate und *x*, *y*, *z* Variable
 - Terme sind Konstanten (*Mary*, *green*, *23*, ...), Variablen (oder Funktionen)
 - *Enkelin*(*x*, *y*), *Vater*(*y*, *z*), ..., sind sogenannte Atome (Atome und ihre Negationen heißen Literale). Die Argumente in Atomen sind Terme; arity spezifiziert die Anzahl der Argumente
 - Ein *ground literal* enthält keine Variablen; ein ground atom ist ein ground fact
- Wir nehmen an, dass das als wahr erkannte Wissen in einer relationalen Datenbank abgelegt wird

Neu:

- Target: For all pairs of persons x, y who are in the relation GrandDaughter: The granddaughter of x is y .
- Start: For all pairs of persons x, y it is true that they are in the target relation Granddaughter
- For all persons y who have a father (i.e., who are in the first position of the relation *Father*), it is true that the granddaughter of x is y ; the father of y is z .
- For all persons y who have a father z and for all persons x who are the father of t and $z = t$, and if y is in the relation Female it is true that the granddaughter of x is y

Training set contains the rows: (x, y) is in Granddaughter table (1/0), x, y, y has a father (1/0), z (the father), x is a father (1/0), t is the descendent, $t = x$ (1/0), y is female (1/0)

Wenn positive Beispiele übrig bleiben wird FOIL eine weitere Suche starten: das Endergebnis ist wieder eine disjunction of conjuncts

Relationale Datenbank

- Eine Relation R ist eine 2-dimensionale Tabelle (Beispiel: *Filme*); der Name einer Relation entspricht einem Prädikat
- Ein Attribut ist eine Spalte einer Relation (Beispiele: Titel, Jahr, Länge, Filmtyp); (Argumente der Prädikate)
- (Attribute sind Attribute von Objekten (Entities))
- Ein Schema ist der Name der Relation mit ihren Attributen. Beispiel:

Film(Titel, Jahr, Länge, Filmtyp)

- Ein relationales Datenbankschema ist die Gesamtheit der Schema in einer Datenbank
- Ein Tupel ist eine Zeile einer Relation und entspricht einem *ground fact* (*ground atom*)
- Mit Hilfe den Operationen einer relationalen Algebra können neue Relationen aus vorgegebenen Relationen abgeleitet werden: Projektionen, Selektion, Verbindungen, ...

Beispiel:

- Relation: Movie

Titel	Jahr	Länge	inFarbe	StudioName	ProduzentC#
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890
Wayne's world	1992	95	true	Paramount	99999

- Offensichtliche Operationen: Union $R \cup S$, Intersektion $R \cap S$, Differenz $R - S$
- Projektion : $\pi_{Titel, Jahr, Länge}(Movie)$

Titel	Jahr	Länge
Star Wars	1977	124
Mighty Ducks	1991	104
Wayne's world	1992	95

Selektion

- Selektion $\sigma_{Länge \geq 100}(Movie)$

Titel	Jahr	Länge	inFarbe	StudioName	ProduzentC#
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890

Cartesisches Produkt und Joins

- Cartesisches Produkt

R:	<table border="1"><tr><th>A</th><th>B</th></tr><tr><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td></tr></table>	A	B	1	2	3	4
A	B						
1	2						
3	4						

S:	<table border="1"><tr><th>B</th><th>C</th><th>D</th></tr><tr><td>2</td><td>5</td><td>6</td></tr><tr><td>4</td><td>7</td><td>8</td></tr><tr><td>9</td><td>10</td><td>11</td></tr></table>	B	C	D	2	5	6	4	7	8	9	10	11
B	C	D											
2	5	6											
4	7	8											
9	10	11											

Dann ist $R \times S$

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

- Natürliches Join: $R \bowtie S$ (nur die Tupel werden gepaart, die in den gemeinsamen Attributen übereinstimmen, hier: B)

A	B	C	D
1	2	5	6
3	4	7	8

Datalog

- Diese Operationen können ebenso mit einer relationalen Logik *Datalog* definiert werden
- Relationen werden als Prädikate P dargestellt
- Ein Prädikat mit Argumenten nennt man ein Atom; Argumente können Konstante sein $P(\textit{StarWars})$ oder Variablen $P(m_name)$;
- In datalog: Das Atom

$$R(a_1, \dots, a_n)$$

ist wahr, wenn a_1, \dots, a_n ein Tupel der Relation R ist

Regeln heißen Klauseln

- Eine Wenn-Dann-Regel (Klausel) hat die Form:

$$Kopf \longleftarrow Körper$$

- Kopf (head) ist ein Atom und der Körper (body) ist eine Konjunktion (UND verknüpft) von Atomen, sogenannten Unterzielen (sub goals); vor einem Unterziel kann auch der logische Operator *NOT* stehen (Atome und ihre Negationen werden als Literale bezeichnet)
- Beispiel:

$$LongMovie(t, y) \longleftarrow Movie(t, y, \dots) \text{ AND } l \geq 100$$

entspricht der Zuordnung $LongMovie := \pi_{Titel, Jahr, Länge}(\sigma_{Länge \geq 100}(Movie))$

- (Horn clause: Wenn im Körper nur positive Literale (Atome) auftreten; Äquivalent zu: $H \vee \neg L_1 \vee \neg L_2 \vee \dots \vee \neg L_n$ (disjunction of literals!))
- **Wenn der Körper einer Regel wahr ist, wird das Tupel zur Relation hinzugefügt, die im Kopf auftaucht**

- Ein Prädikat, welches durch eine Regel definiert wird, heisst intensional (sonst: extensional)
- Nebenbedingung: jede Variable einer Regel muss in einem nicht-negierten Unterziel mindestens einmal erscheinen

Rekursionen in Datalog

- Datalog erlaubt Rekursionen

$$\textit{Erreicht}(x, y) \longleftarrow \textit{Flug}(a, x, y, d, r)$$

$$\textit{Erreicht}(x, y) \longleftarrow \textit{Erreicht}(x, z) \text{ AND } \textit{Erreicht}(z, y)$$

- Beachte: die Rolle der Merkmale im normalen maschinellen Lernen nehmen jetzt Prädikate (b.z.w Atome) ein, und diese sind logische Funktionen angewandt auf Attribute (eines oder mehrerer Objekte)

Beispiel: Lernen von Regeln erster Ordnung

Beispiel:

- Zielkonzept $Tochter(x, y)$
- Datenbank enthält: $Person(Name, Mutter, Vater, männlich, weiblich)$

Name	Mutter	Vater	männlich	weiblich
Sharon	Louise	Bob	Falsch	Wahr
Bob	Nora	Victor	Wahr	Falsch

- Im Zielkonzept gibt es nur einen Eintrag $Tochter(Bob, Sharon)$
- Ziel ist das Erlernen der Regel

$$Tochter(x, y) \leftarrow Vater(y, x) \wedge Weiblich(y)$$

Foil

- Der Foil Algorithmus arbeitet sehr ähnlich wie der Sequential Covering Algorithmus, nur dass die Zielgröße ein Atom ist und dass die Merkmale ebenfalls Atome sind, die mit UND bzw UND NOT verknüpft werden; dies bedeutet, dass FOIL Datalog Regeln lernen kann (beachte: Literale sind Atome und ihre Negation)
- Wie im *sequential covering algorithm*: Foil started mit einem Kopf (target literal) und erweitert den Körper, bis eine erwünschte Akkuratheit erreicht ist (*conjunction of literals*) (innere Schleife). Die Suche geht vom Allgemeinen zum Spezifischen.
- Ähnlich werden neue Regeln generiert, bis eine genügende Abdeckung erreicht ist . Die Suche beginnt beim Spezifischen (nur eine innere Regel) zum Allgemeinen (*disjunction of conjuncts*).

Foil: innere Schleife

- FOIL addiert neue Literale zur Regel

$$P(x_1, x_2, \dots, x_k) \leftarrow L_1, \dots, L_n$$

- Ein neues Literal L_{n+1} wird generiert aus
 - $Q(v_1, \dots, v_r)$, wobei Q eine Relation ist (typischerweise eine Relation in der Datenbank oder eine Relation die durch einfache Projektion erzeugt werden kann); mindestens eine Variable aus v_1, \dots, v_r muss schon in der Regel existieren
 - $Equal(x_j, x_k)$, wobei beide Variablen in der Regel bereits existieren
 - Oder: Negation der vorangegangenen Atome

Beispiel:

- Lerne das Konzept: $Enkelin(x, y)$
- Man beginnt mit (die Enkelin von x ist y)

$$Enkelin(x, y) \leftarrow$$

so dass, $Enkelin$ wahr ist für alle x, y

- Betrachte die Kandidaten-Literale:

$$Equal(x, y), Weiblich(x), Weiblich(y), Vater(x, y), Vater(y, x),$$

$$Vater(x, z), Vater(z, x), Vater(y, z), Vater(z, y)$$

und deren Negationen

- Foil selektiert $Vater(y, z)$, (der Vater von y ist z) so dass

$$Enkelin(x, y) \leftarrow Vater(y, z)$$

Dies ist möglich, da y in beiden Atomen enthalten ist

- Im nächsten Schritt kommen alle vorangegangenen Literale in die Auswahl und zusätzlich

$Weiblich(z), Equal(z, x), Equal(z, y),$

$Vater(z, w), Vater(w, z)$

und deren Negationen (da z nun in der Regel auftaucht). Beachte: w erscheint neu in der Regel

- FOIL fügt hinzu: $Vater(z, x)$ und dann $Weiblich(y)$ und terminiert die innere Schleife mit

$Enkelin(x, y) \leftarrow Vater(y, z) \wedge Vater(z, x) \wedge Weiblich(y)$

Wie Regelsuche in FOIL geführt wird

- Konvention: $P(x, y)$ bedeutet: Das P von x ist y .
- Datenbank: $Enkelin(Victor, Sharon)$, $Weiblich(Sharon)$ und

Kind	Vater
Sharon	Bob
Tom	Bob
Bob	Victor

- Geschlossene-Welt-Annahme: alle anderen Literale mit $Enkelin$, $Weiblich$, $Vater$ mit den Konstanten $Victor$, $Sharon$, Bob , Tom sind falsch
- Start: Wir berücksichtigen alle Möglichkeiten (bindings) in welchen die Variablen x, y in

$$Enkelin(x, y) \leftarrow$$

gebunden werden können, mit den Konstanten $Victor$, $Sharon$, Bob , Tom (insgesamt 16). Es gibt ein positives und 15 negative Beispiele.

- Der Gewinn eines neuen Literals L ist definiert als

$$Foil_Gain(L, R) = t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

Sei R' die Regel, die entsteht, wenn man L zu R hinzunimmt

- p_0 : Anzahl der positiven Bindungen von R
- n_0 : Anzahl der negativen Bindungen von R ,
- p_1 : Anzahl der positiven Bindungen von R' ,
- n_1 : Anzahl der negativen Bindungen von R' ,
- t : Anzahl der positiven Bindungen der Regel R , nachdem L zu R hinzugenommen wird.

...

- Man startet mit $p = 1$ und $n = 15$
- Mit $Vater(y, z)$, muss y als Kind in der Relation $Vater$ auftauchen. D.h., dass $Victor \neq y$ und $p = 1, n = 11$
- Mit $Vater(z, x)$, muss z der Vater von x sein. Dies ist nur möglich wenn $x = Victor$ und $y = Sharon$ oder $y = Tom$; nun ist $p = 1$ und $n = 1$
- Mit $Weiblich(y)$, folgt dass $y = Sharon$ ist $p = 1$ und $n = 0$; wir haben 100 % Akkuratheit und ebenfalls 100 % Abdeckung

Weitere Regeln

- Wenn Abdeckung noch nicht erreicht werden wie beim Sequential Covering Algorithmus weitere Regeln hinzugefügt,

Rekursive Regeln

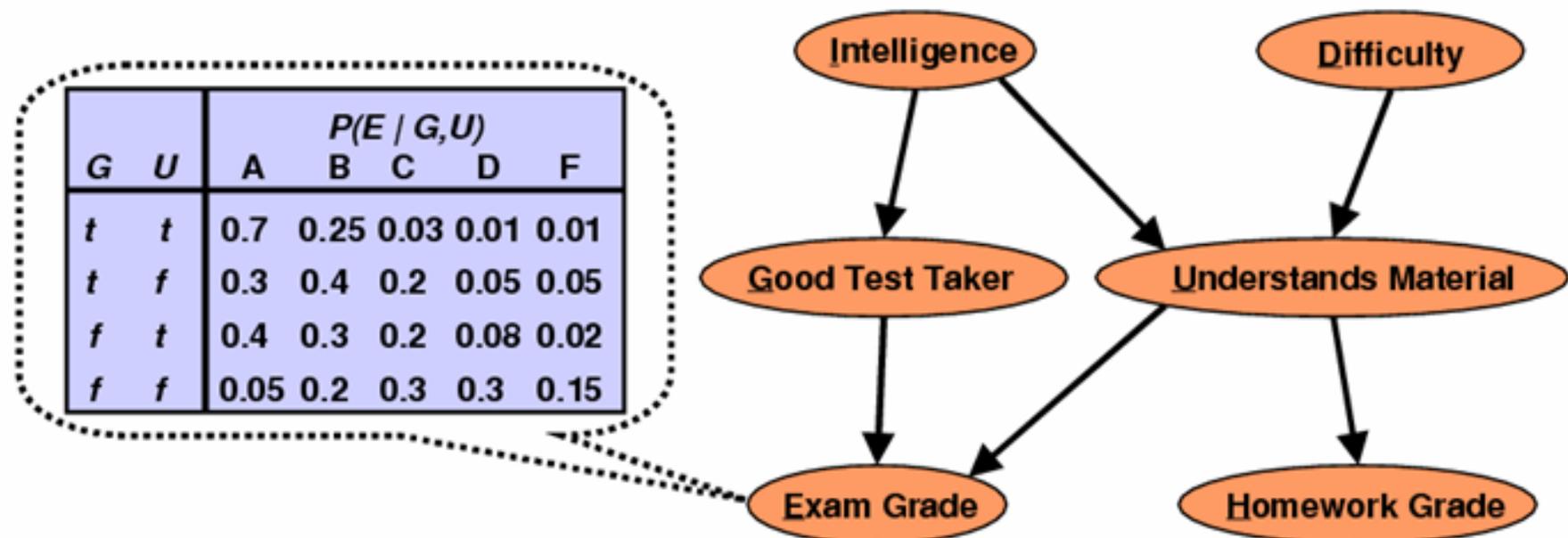
- Mit Foil können ebenfalls rekursive Regeln gelernt werden, also Regeln, die die gleichen Prädikate im Kopf und im Körper enthalten
- Beispiel

$$\textit{Ancestor}(x, y) \leftarrow \textit{Parent}(x, y)$$

$$\textit{Ancestor}(x, y) \leftarrow \textit{Parent}(x, z) \wedge \textit{Ancestor}(z, y)$$

Probabilistisch Relationale Modelle

- Ziel ist die Generierung kompletter probabilistischer Modelle basierend unter Beibehaltung der Mächtigkeit relationaler Modellierung
- Im Grunde bestehen die Ansätze darin, die Wahrheit des Zielatoms mit einer Wahrscheinlichkeit zu belegen, die bedingt ist auf die Zustände der Atome im Körper
- Wie in einem Bayes'schen Netz entsteht ein zyklensfreies probabilistisches System
- Es sind eine Reihe von Ansätzen entstanden, die Logik, Lernen und Wahrscheinlichkeiten verknüpfen (**inductive logic programming**, ILP)
- Persönlich finde ich die Ansätze am interessantesten, die sich an relationalen Datenbanken orientieren; dies sind der PRM Ansatz (probabilistic relation models) und der DAPER Ansatz (Directed Acyclic Probabilistic Entity Relationship Models)



$$P(I, D, G, U, E, H) = P(I)P(D)P(G | I)P(U | I, D)P(E | G, U)P(H | U)$$

Fig. 1.1. A simple Bayesian network for the student performance domain, the decomposition of the joint distribution into a product of CPDs, and the CPD for one of the nodes in the network

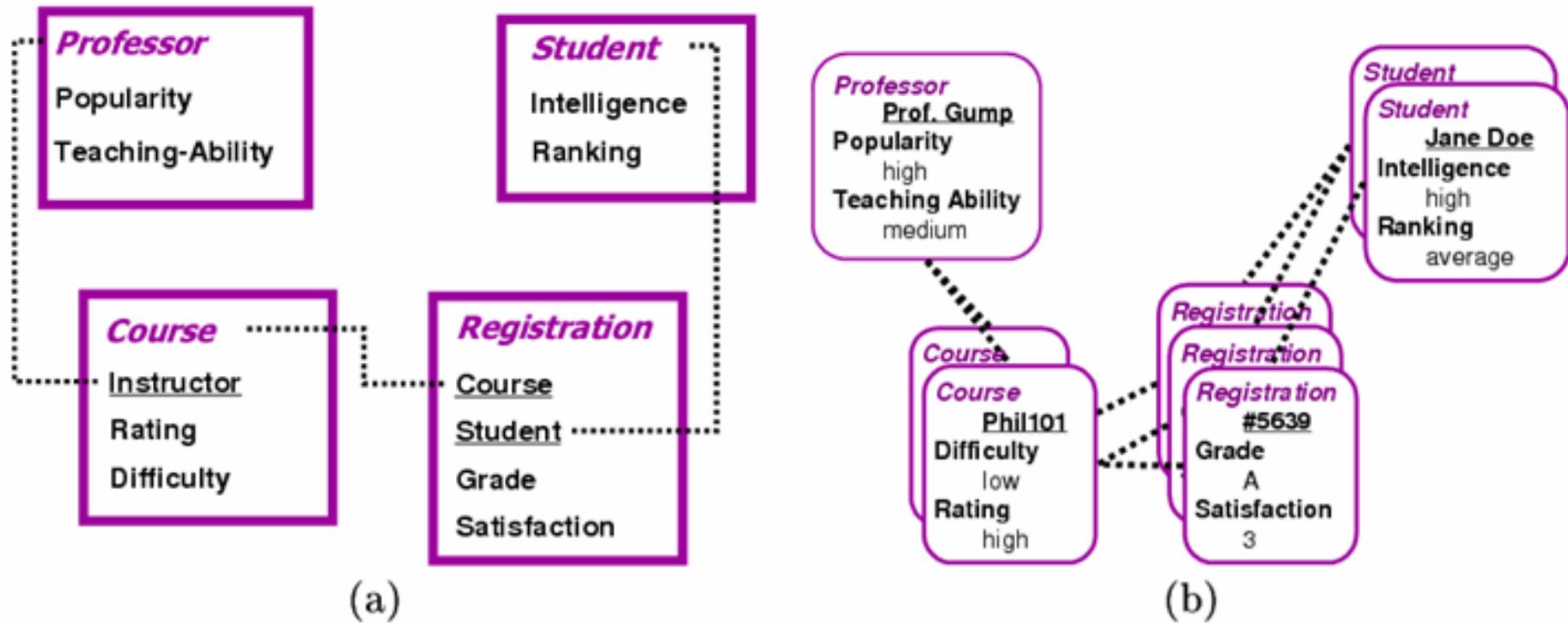


Fig. 1.2. (a) A relational schema for a simple university domain. The underlined attributes are reference slots of the class and the dashed lines indicate the types of objects referenced. (b) An example instance of this schema. Here we do not show the reference slots, we use dashed lines to indicate the relationships that hold between objects.

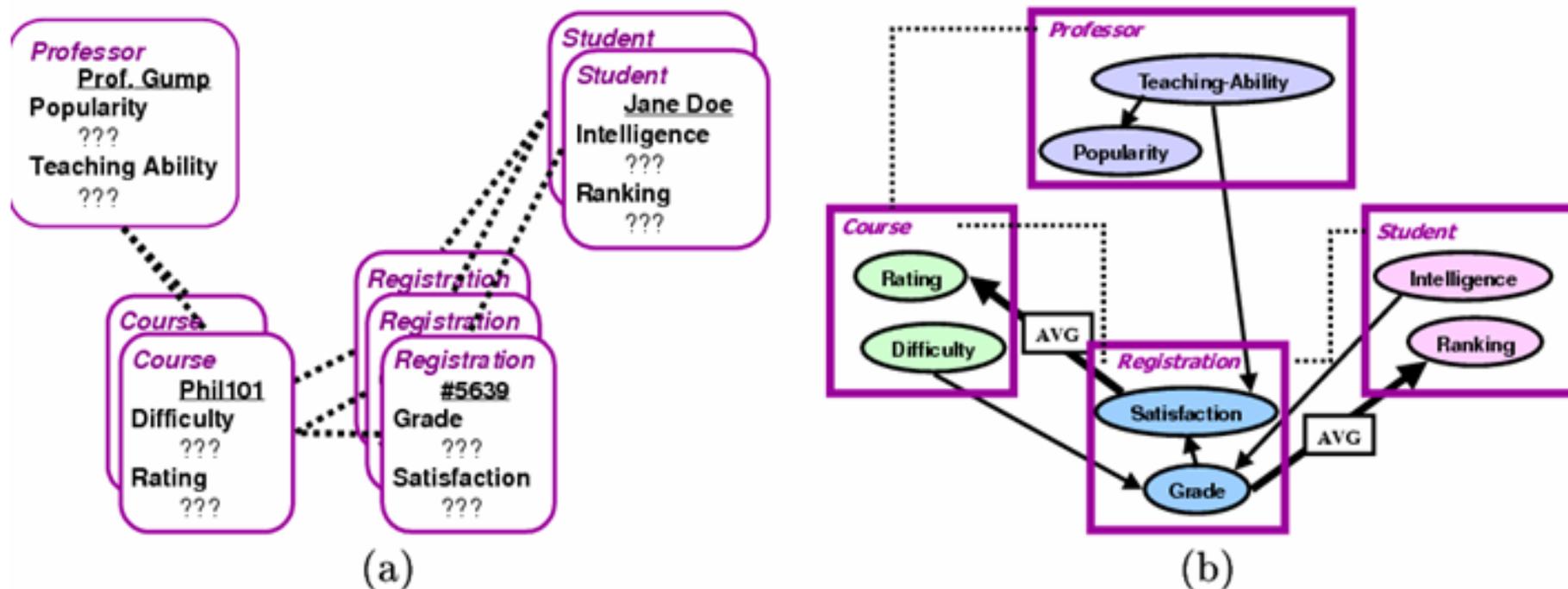
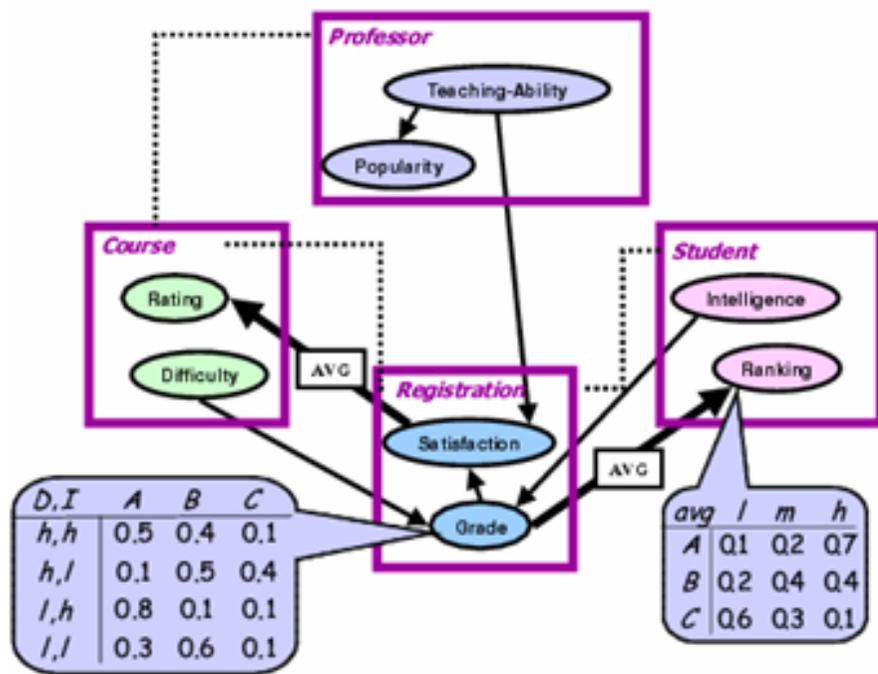
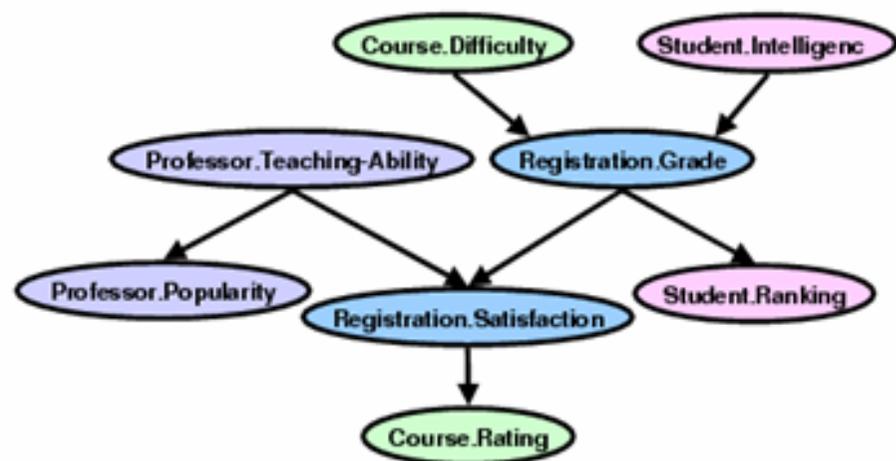


Fig. 1.3. (a) An example relational skeleton for the school domain. Here we know only the objects in our domain and the relationships that hold between them. (b) A PRM structure for the school domain. Edges correspond to probabilistic dependency. Edges from one class to another are routed through slot-chains (chains of references). For clarity these are not explicitly stated in the diagram (see text).



(a)



(b)

Fig. 1.4. (a) The CPD for *Registration.Grade* and the CPD for an aggregate dependency of *Student.Rank* on *Student.Registered-In.Grade*. (b) The dependency graph for the school PRM.

$$\begin{aligned} P(\mathcal{I} \mid \sigma, \mathcal{S}, \theta_{\mathcal{S}}) &= \prod_{x \in \sigma} \prod_{A \in \mathcal{A}(x)} P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\text{Pa}(x.A)}) \\ &= \prod_{X_i} \prod_{A \in \mathcal{A}(X_i)} \prod_{x \in \mathcal{O}^{\sigma}(X_i)} P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\text{Pa}(x.A)}) \end{aligned}$$

Definition 1: A *probabilistic relational model (PRM)* Π for a relational schema \mathcal{R} is defined as follows. For each class $X \in \mathcal{X}$ and each descriptive attribute $A \in \mathcal{A}(X)$, we have:

- a set of *parents* $\text{Pa}(X.A) = \{U_1, \dots, U_l\}$, where each U_i has the form $X.B$ or $\gamma(X.\tau.B)$, where τ is a slot chain;
- a *conditional probability distribution (CPD)* that represents $P_\Pi(X.A \mid \text{Pa}(X.A))$. ■

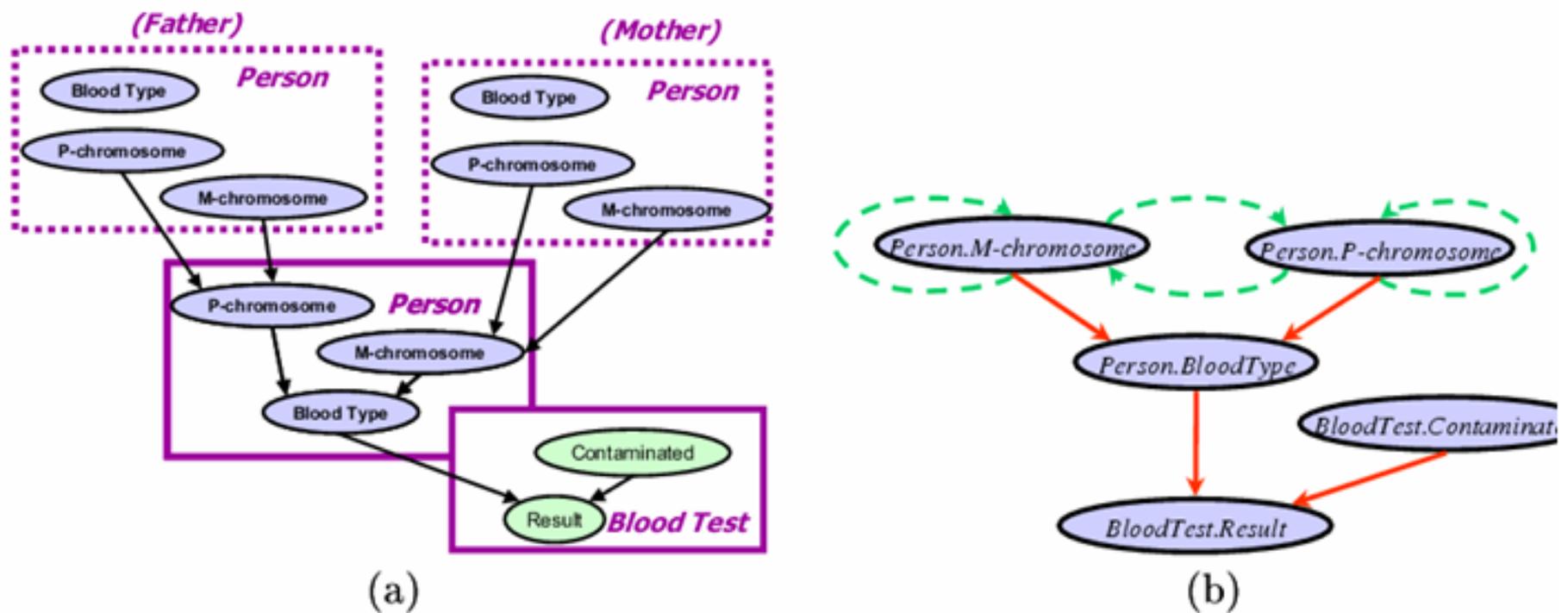


Fig. 1.5. (a) A simple PRM for the genetics domain. (b) the corresponding dependency graph. Dashed edges correspond to “guaranteed acyclic” dependencies.

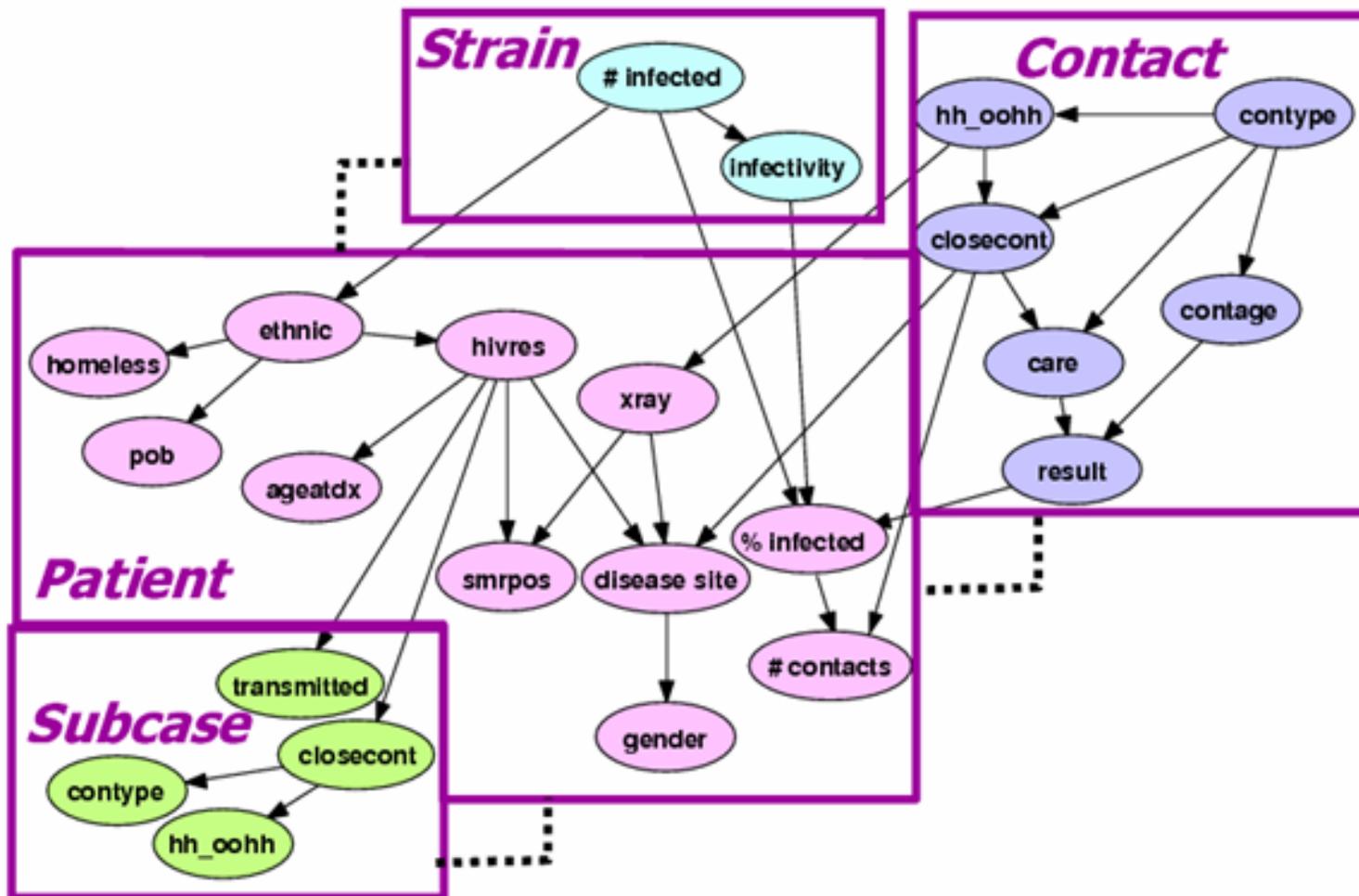


Fig. 1.7. The PRM structure for the TB domain.

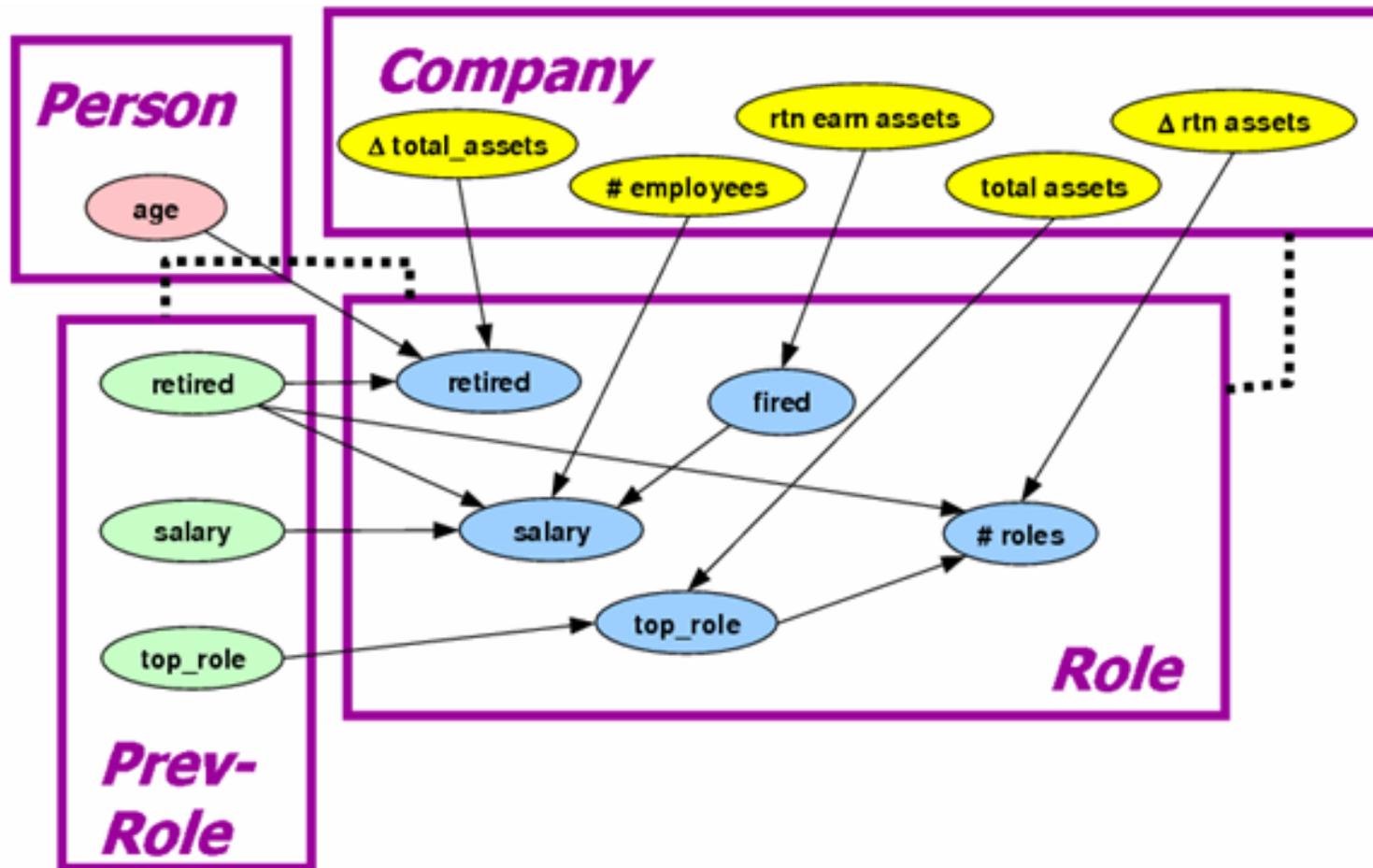


Fig. 1.8. The PRM structure for the Company domain.