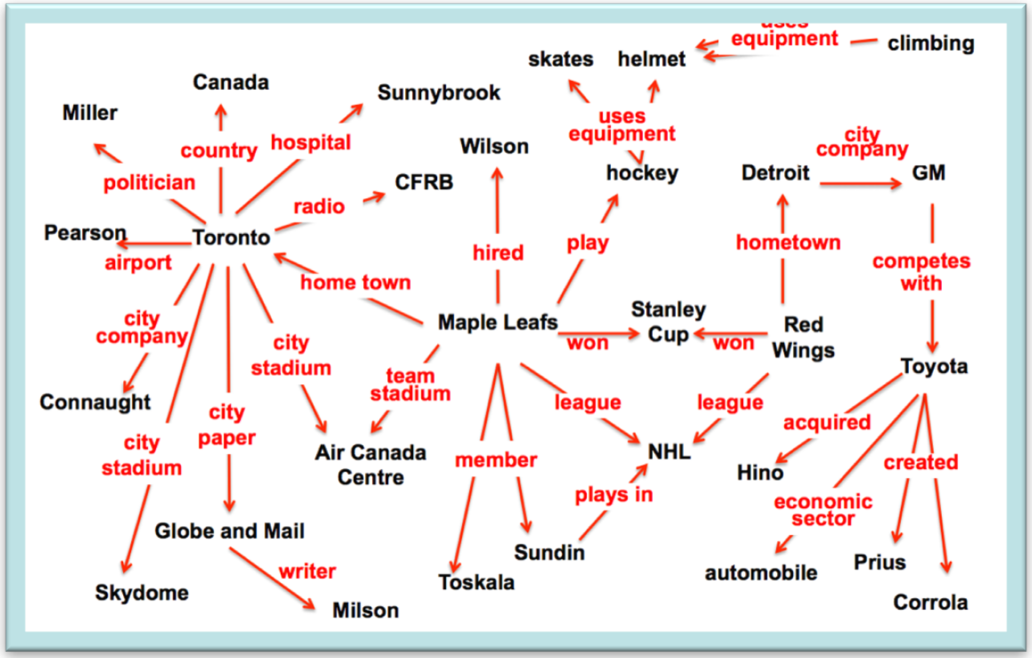
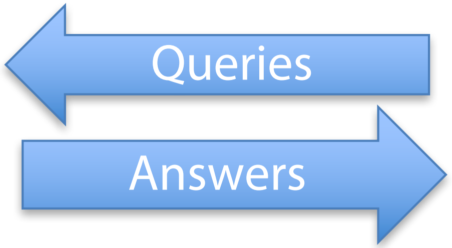
Web-Mining Agents

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Tanya Braun (Übungen)

Accessing Document Annotation Databases





. . .

What if the DB/KB or domain models are imperfect?

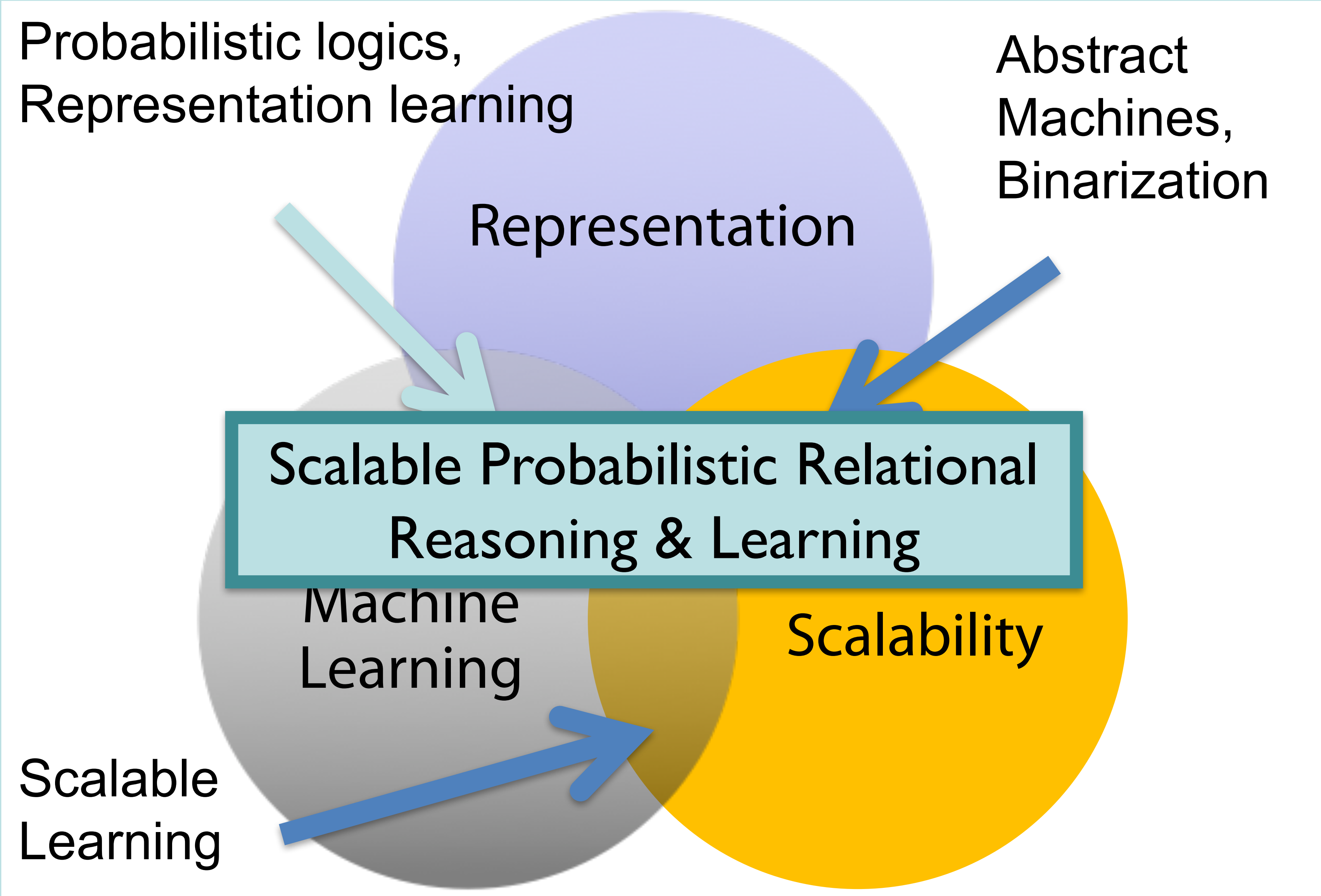
Challenges:

- **Robustness**: noise, incompleteness, ambiguity (“*Sunnybrook*”), statistical information (“*foundInRoom(bathtub, bathroom)*”), …
- **Complex queries:** “which Canadian hockey teams have won the Stanley Cup?”
- **Extensions to annotations required** (exploit domain knowledge)
- **Learning**: how to *acquire and maintain* domain models as well as how to use it

Current state of the art

“**Expressive, probabilistic, efficient**: pick any two”

Three Areas of Data Science



Datalog for Extending Annotation DBs

- A program defines a unique least Herbrand model
- Example program:

  grandparent(X,Y):-parent(X,Z),parent(Z,Y).

  parent(alice,bob). parent(bob,chip). parent(bob,dana).

The least Herbrand model also includes grandparent(alice,dana) and grandparent(alice,chip).

Finding the least Herbrand model: theorem proving…

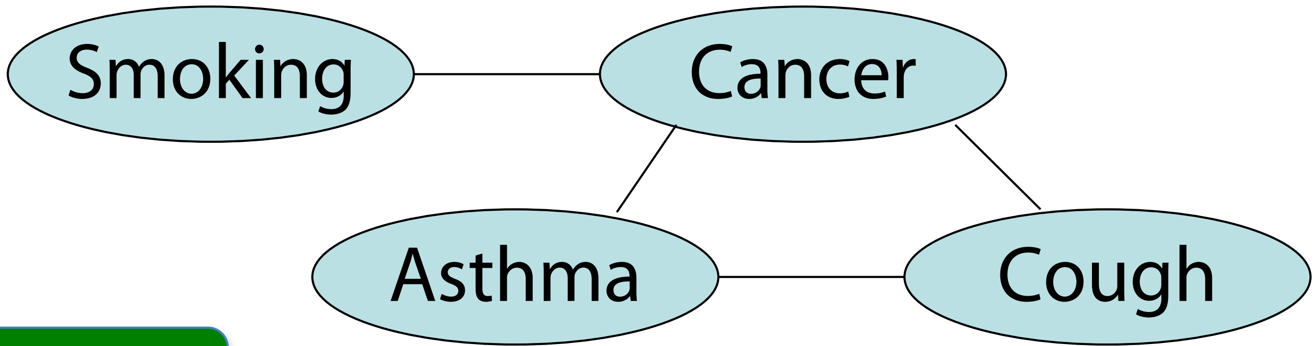
Usually we care about answering queries:

What are values of W: grandparent(alice,W) ?

Markov Networks

- **Undirected** graphical models

[h/t Pedro Domingos]



x = vector

$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$= \frac{1}{Z} \exp\left(\sum_i \Phi_c(x_c) \right)$$

Smoking	Cancer	Φ(S,C)
False	False	4.5
False	True	4.5
True	**False**	**2.7**
True	True	4.5

A soft constraint that smoking → cancer

Markov Logic Networks (MLNs): Intuition

[Domingos et al]

- QA w.r.t. is a set of **hard constraints** on the set of possible worlds constrained to be deductively closed
- Let's make closure a **soft constraint**: When a world is not deductively closed, it becomes less probable
- Give each rule a weight which is a reward for satisfying it: (Higher weight $\Rightarrow$ Stronger constraint)

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

Markov Logic Networks (MLNs): 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 – each element of the Herbrand base
  - One feature for each grounding of each **formula** F in the MLN, with the corresponding weight w

Example: Friends & Smokers

Smoking causes cancer.
Friends have similar smoking habits.

Example: Friends & Smokers

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

Example: Friends & Smokers

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

Example: Friends & Smokers

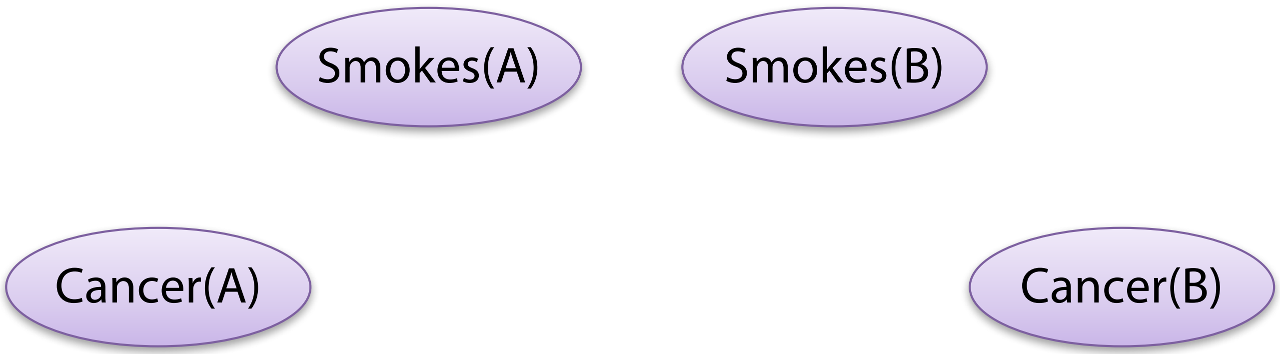
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

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

Example: Friends & Smokers

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

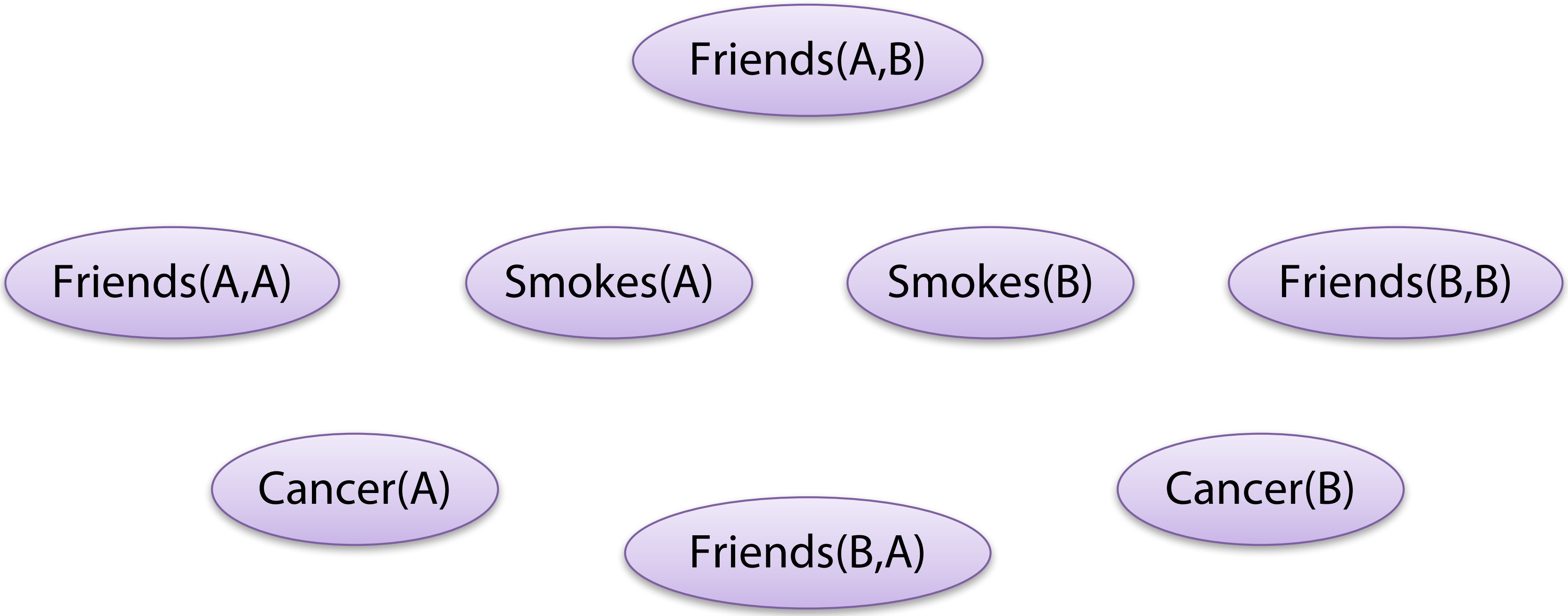
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

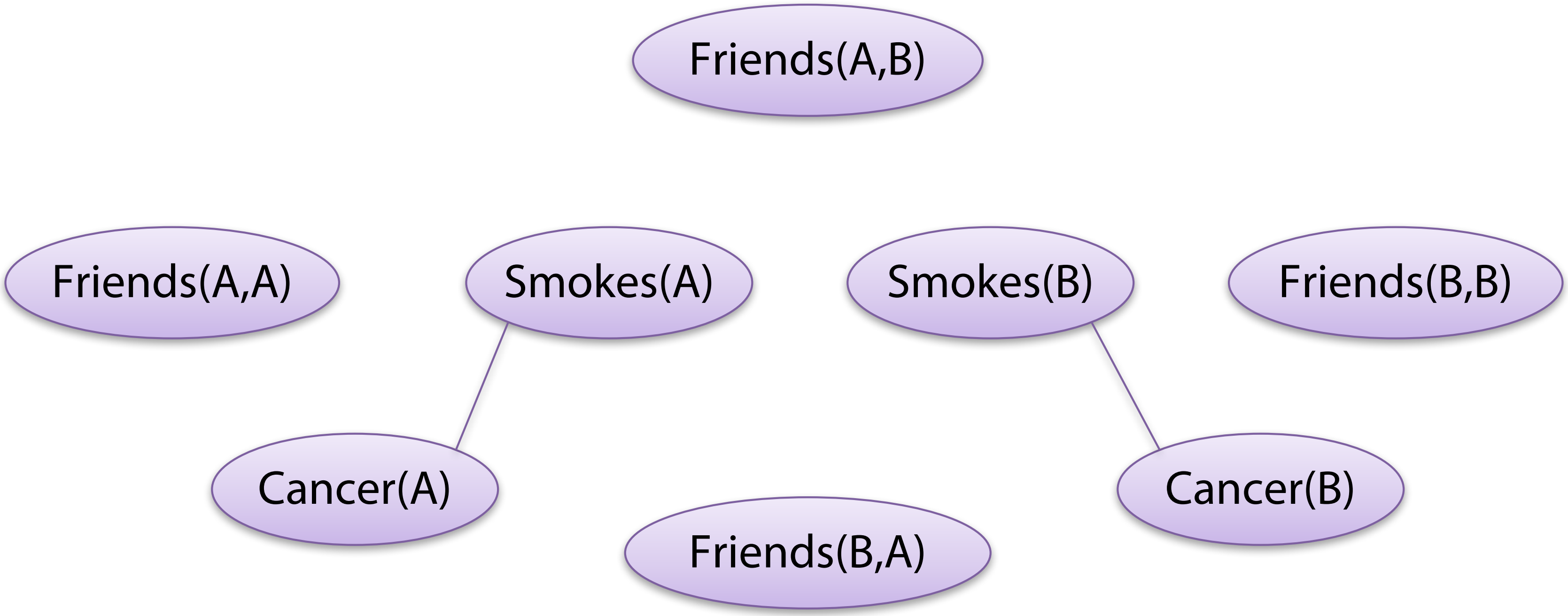
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

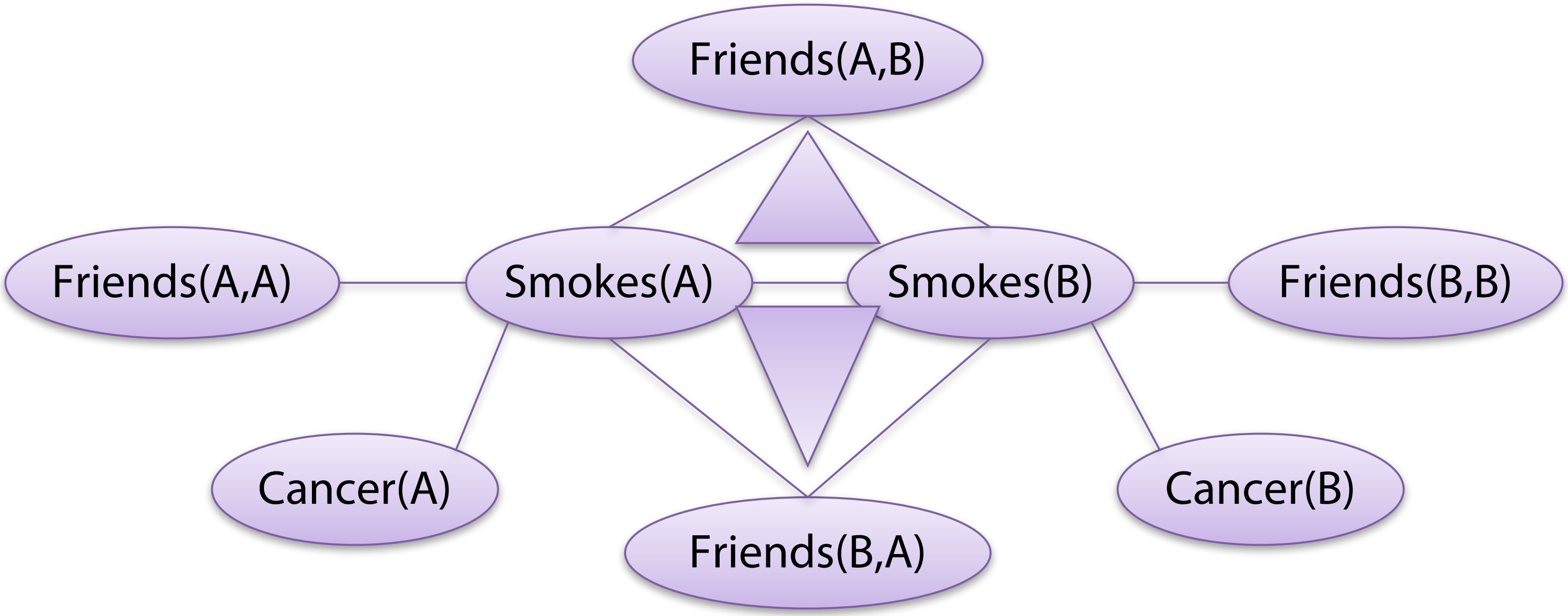
Two constants: **Anna** (A) and **Bob** (B)



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

- MLN is **template** for ground Markov nets
- Probability of a world ***X***:

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

Weight of formula *i*

No. of true groundings of formula *i* in *x*

Recall for ordinary Markov net

$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$= \frac{1}{Z} \exp\left(\sum_i \Phi'_c(x_c) \right)$$

H/T: Pedro Domingos

MLNs generalize many statistical models ☺

- Special cases:
  - Markov networks
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields

- Obtained by making all predicates zero-arity

- Markov logic allows objects to be interdependent (non-i.i.d.)

H/T: Pedro Domingos

MLNs generalize logic programs ☺

- Subsets of Herbrand base ~ domain of joint distribution
- Interpretation ~ element of the joint
- Consistency with all clauses A:-B1,...,Bk , i.e. "model of program" ~ compatibility with program as determined by clique potentials

- Reaches logic in the limit when potentials are infinite (sort of)

H/T: Pedro Domingos

MLNs are expensive ☹

- Inference done by *explicitly building a ground MLN*
  - Herbrand base is huge for reasonable programs: It grows *faster* than the size of the DB of facts
  - You'd like to able to use a huge DB—NELL is $O(10M)$
- After that, inference on an arbitrary MLN is *expensive*: #P-complete
  - It's not obvious how to restrict the template so the MLNs will be tractable

Use Probabilistic Databases for Scalability?

Old trick: If you want to weight a **rule** you can introduce a **rule-specific fact**….

1. `uncle(X,Y):-child(X,W),brother(W,Y).`
2. `uncle(X,Y):-aunt(X,W),husband(W,Y).`
3. ~~`status(X,tired):-child(W,X),infant(W).`~~

~~r3. status(X,tired) :- child(W,X), infant(W), **weighted(r3).**~~

r3. status(X,T) :- child(W,X), infant(W), assign_tired(T), **weighted(r3).**

assign_tired(tired),1

`child(liam,eve)`,0.99	`infant(liam)`,0.7
`child(dave,eve)`,0.99	`infant(dave)`,0.1
`child(liam,bob)`,0.75	`aunt(joe,eve)`,0.9
`husband(eve,bob)`,0.9	`brother(eve,chip)`,0.9
	weighted(r3),0.88

So learning rule weights is a **special case** of learning weights for **selected DB facts**.

PDBs: Problems

- Not clear if expanding queries with respect to rules yields safe queries (safe queries can be answered with SQL)
- Rules can be cyclic (no expansion possible)
- Queries get very large due to expansion (n-way join order optimization has its limits)
  - Preprocessing is at least not easy
  - Better approach: Query data w.r.t. model
- How to learn a model?
  - Learn datalog rules
  - Learn more complex logical formulas

Inductive Logic Programming

- Combines inductive methods with the power of first-order representations
- Offers a rigorous approach to the learning problem
- Offers complete algorithms for inducing general, first-order theories from examples

E.Y. Shapiro., The model inference system. Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2. Morgan Kaufmann Publishers Inc., **1981**

E.Y. Shapiro., Inductive inference of theories from facts, Research Report 192, Yale University, Department of Computer Science, **1981**. Reprinted in J.-L. Lassez, G. Plotkin (Eds.), Computational Logic, The MIT Press, Cambridge, MA, 1991, pp. 199–254.

J.R. Quinlan. Learning Logical Definitions from Relations. Machine Learning, Volume 5, Number 3, **1990**

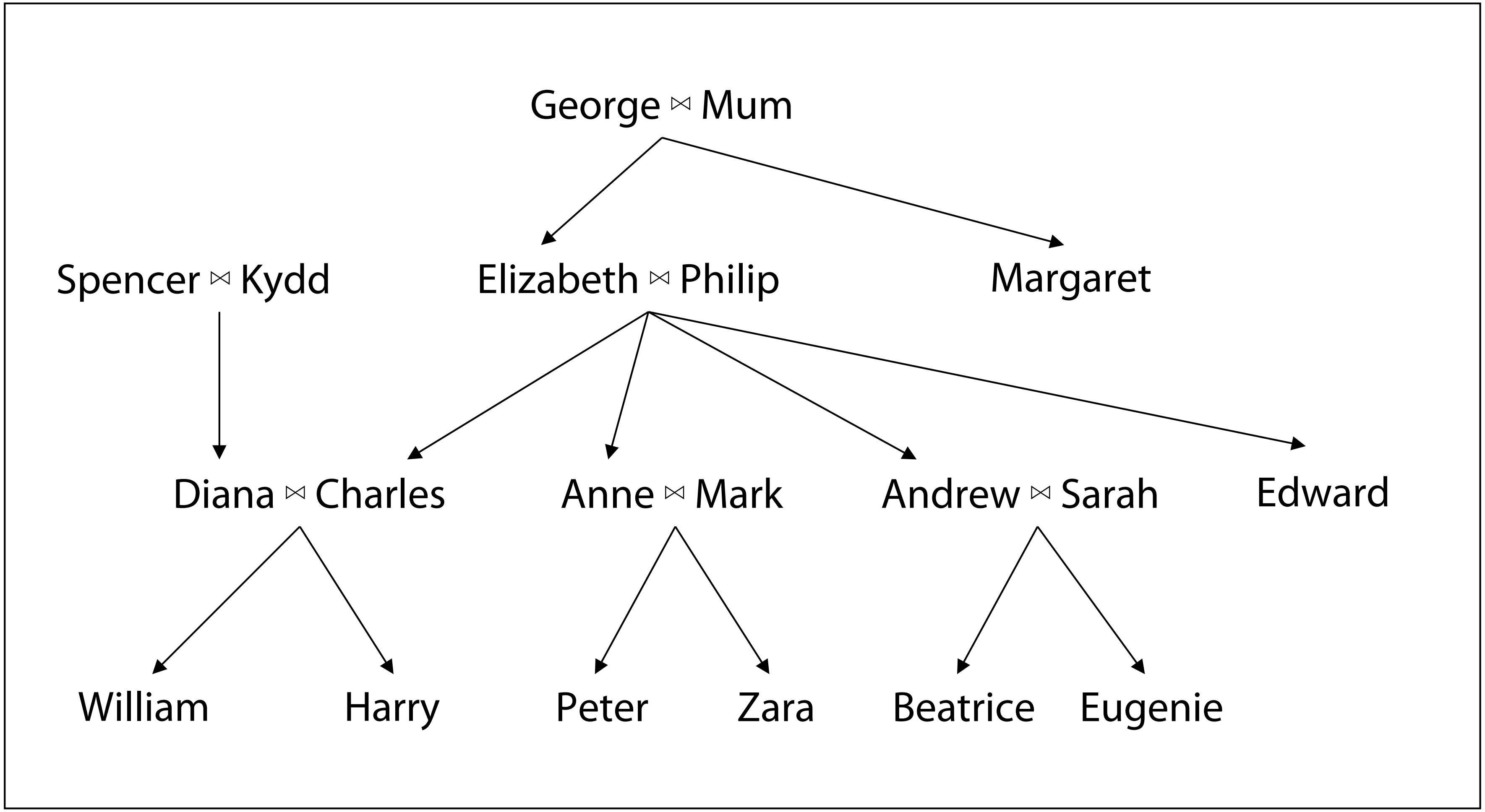
Muggleton, S.; De Raedt, L., Inductive Logic Programming: Theory and methods. The Journal of Logic Programming. 19-20: 629–679, **1994**

Lavrac, N.; Dzeroski, S., Inductive Logic Programming: Techniques and Applications. New York: Ellis Horwood, **1994**

ILP: An example

- Example: Learning family relations from examples
  - Observations are an extended family tree
    - Mother, Father and Married relations
    - Male and Female properties
  - Target predicates: Grandparent, BrotherInLaw, Ancestor

Example (not up to date)



Example

- Descriptions include facts like
  - Father(Philip, Charles)
  - Mother(Mum, Margaret)
  - Married(Diana, Charles)
  - Male(Philip)
  - Female(Beatrice)
- Sentences in Classifcations depend on the target concept being learned (in the example: 12 positive, 388 negative)
  - Grandparent(Mum, Charles)
  - ¬Grandparent(Mum, Harry)
- Goal: find a set of sentences for Hypothesis such that the entailment constraint is satisfied
  - Without background knowledge this is for example

$$
\begin{aligned}
Grandparent(x,y) \Leftrightarrow & \left[\exists_z Mother(x,z) \wedge Mother(z,y)\right] \\
& \vee \left[\exists_z Mother(x,z) \wedge Father(z,y)\right] \\
& \vee \left[\exists_z Father(x,z) \wedge Mother(z,y)\right] \\
& \vee \left[\exists_z Father(x,z) \wedge Father(z,y)\right]
\end{aligned}
$$

Background knowledge

- A little bit of background knowledge helps a lot
  - Background knowledge contains
    $$Parent(x,y) \Leftrightarrow [Mother(x,y) \vee Father(x,y)]$$
  - Grandparent is now reduced to
    $$Grandparent(x,y) \Leftrightarrow [\exists_z Parent(x,z) \wedge Parent(z,y)]$$
- Constructive induction algorithm
  - Create new predicates to facilitate the expression of explanatory hypotheses
  - Example: introduce a predicate Parent to simplify the definitions of the target predicates

Top-Down Inductive Learning: FOIL

- Split positive and negative examples
  - Positive: <George, Anne>, <Philip, Peter>, <Spencer, Harry>
  - Negative: <George, Elizabeth>, <Harry, Zara>, <Charles, Philip>
- Construct a set of Horn clauses with Grandfather(x,y) as the head with the positive examples instances of the Grandfather relationship
  - Start with a clause with an empty body

    $\Rightarrow$ Grandfather(x,y)
  - All examples are now classified as positive, so specialize to rule out the negative examples: Here are 3 potential additions:

    1) Father(x,y) $\Rightarrow$ Grandfather(x,y)
    2) Parent(x,z) $\Rightarrow$ Grandfather(x,y)
    3) Father(x,z) $\Rightarrow$ Grandfather(x,y)
  - The first one incorrectly classifies the 12 positive examples
  - The second one is incorrect on a larger part of the negative examples
  - Prefer the third clause and specialize

    Father(x,z) $\wedge$ Parent(z,y) $\Rightarrow$ Grandfather(x,y)

FOIL

function Foil(examples, target) **returns** a set of Horn clauses
inputs: examples, set of examples
target, a literal for the goal predicate
local variables: clauses, set of clauses, initially empty
while examples contains positive examples **do**
clause ← New-Clause(examples, target)
remove examples covered by clause from examples
add clause to clauses
return clauses

FOIL

function New-Clause(examples, target) **returns** a Horn clause
local variables:
clause, a clause with target as head and an empty body
l, a literal to be added to the clause
extended-examples, a set of examples with values for new variables
extended-examples ← examples
while extended-examples contains negative examples **do**
l ← Choose-Literal(New-Literals(clause), extended-examples)
append l to the body of clause
extended-examples ← set of examples created by applying Extend-Example to each example in extended-examples
return clause

FOIL

function Extend-Example(example, literal) **returns**
  if example satisfies literal
    then return the set of examples created
      by extending example with each
      possible constant value for each new
      variable in literal
  else return the empty set

FOIL

- New-Literals
  - Takes a clause and constructs all possible “useful” literals
- Example: Father(x,z) $\Rightarrow$ Grandfather(x,y)
  - Add literals using predicates
    - Negated or unnegated
    - Use any existing predicate (including the goal)
    - Arguments must be variables
    - Each literal must include at least one variable from an earlier literal or from the head of the clause
    - Valid: Mother(z,u), Married(z,z), Grandfather(v,x)
    - Invalid: Married(u,v)
  - Equality and inequality literals
    - E.g. $z \neq x$, empty list
  - Arithmetic comparisons
    - E.g. $x > y$, threshold values

FOIL

- The way New-Literal changes the clauses leads to a very large branching factor
- Improve performance by using type information
  - E.g., Parent(x,n) where x is a person and n is a number
- Choose-Literal uses a heuristic similar to information gain
- Ockham's razor to eliminate hypotheses
  - If the clause becomes longer than the total length of the positive examples that the clause explains, this clause is not a valid hypothesis
- Most impressive demonstration
  - Learn the correct definition of list-processing functions in Prolog from a small set of examples, using previously learned functions as background knowledge

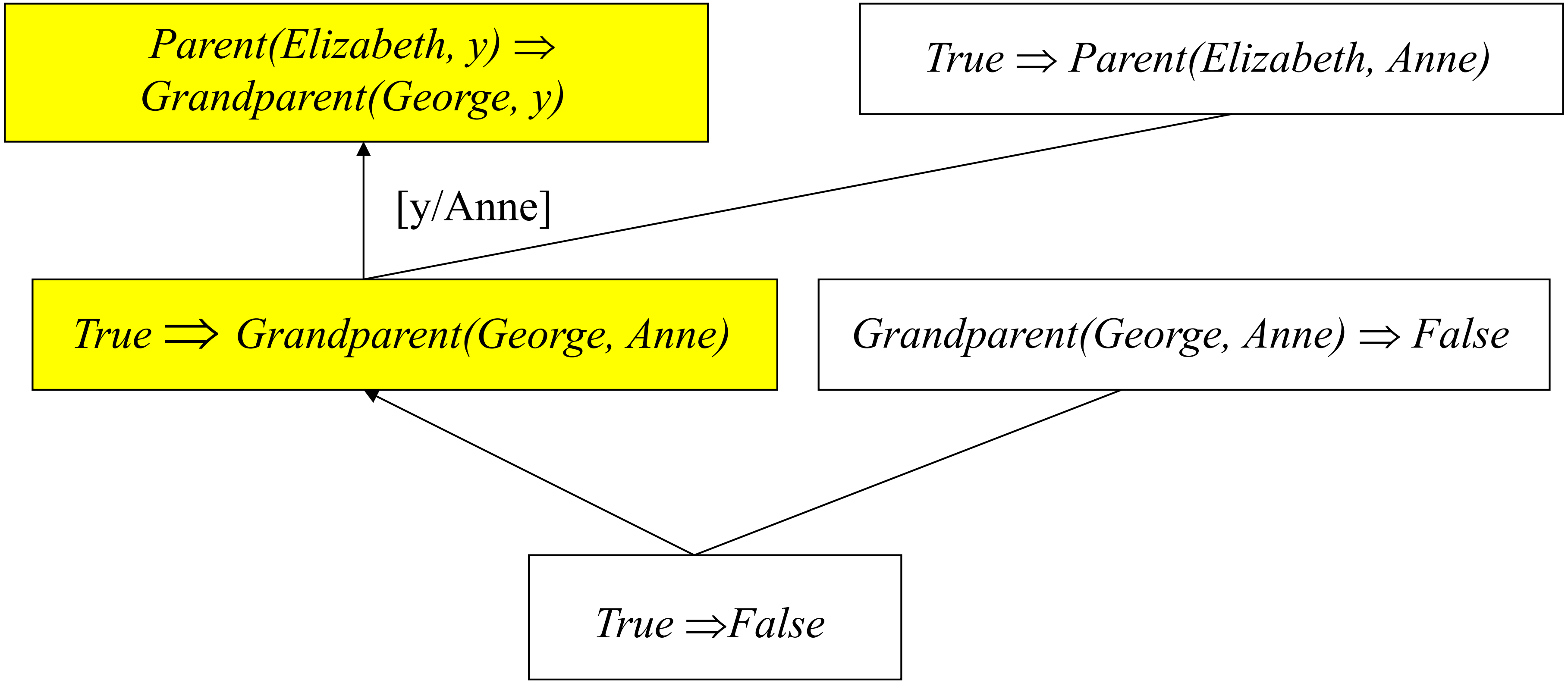
Inverse Resolution

- Inverse resolution
  - Run a proof backwards to find Hypothesis
  - Problem: How to run the proof backwards?

Generating Inverse Proofs

- Ordinary resolution
  - Take two clauses C_1 and C_2 and resolve them to produce the resolvent C
- Inverse resolution
  - Take resolvent C and produce two clauses C_1 and C_2
  - Take C and C_1 and produce C_2

Generating Inverse Proofs



Generating Inverse Proofs

- Inverse resolution is a search
  - For any C and C_1 there can be several or even an infinite number of clauses C_2
    - Instead of Parent(Elizabeth,y) $\Rightarrow$ Grandparent(George,y) there were numerous alternatives

      Parent(Elizabeth,Anne) $\Rightarrow$ Grandparent(George,Anne)

      Parent(z,Anne) $\Rightarrow$ Grandparent(George,Anne)

      Parent(z,y) $\Rightarrow$ Grandparent(George,y)
  - The clauses C_1 that participate in each step can be chosen from Background, Descriptions, Classifications or from hypothesized clauses already generated
- ILP needs restrictions to make the search manageable
  - Eliminate function symbols
  - Generate only the most specific hypotheses
  - Use Horn clauses
  - All hypothesized clauses must be consistent with each other
  - Each hypothesized clause must agree with the observations

New Predicates and New Knowledge

- An inverse resolution procedure is a complete algorithm for learning first-order theories
  - If some unknown Hypothesis generates a set of examples, then an inverse resolution procedure can generate Hypothesis from the examples
- Can inverse resolution infer the law of gravity from examples of falling bodies?
  - Yes, given suitable background mathematics
- Monkey and typewriter problem: How to overcome the large branching factor and the lack of structure in the search space?

New Predicates and New Knowledge

- Inverse resolution is capable of generating new predicates
  - Resolution of C_1 and C_2 into C eliminates a literal that C_1 and C_2 share
  - This literal might contain a predicate that does not appear in C
  - When working backwards, one possibility is to generate a new predicate from which to construct the missing literal

New Predicates and New Knowledge

$$Father(George,y) \Rightarrow P(x,y) \qquad P(George,y) \Rightarrow Ancestor(George,y)$$

$$[x/George]$$

$$Father(George,y) \Rightarrow Ancestor(George,y)$$

- P can be used in later inverse resolution steps
  - Example: Mother(x,y) ⇒ P(x,y) or Father(x,y) ⇒ P(x,y) leading to the “Parent” relationship
- Inventing new predicates is important to reduce the size of the definition of the goal predicate
  - Some of the deepest revolutions in science come from the invention of new predicates (e.g. Galileo’s invention of acceleration)

Learning of "Weights"

- Use similar trick as for PDBs:
  - Introduce atoms weighted(r_k) in rules and respective facts with probabilities
- Learn probabilities of weighted facts such that training data are most likely generated (ML, MAP)
- Various approaches known

- Use MLNs

Problems with MLN QA

- Grounding

Leads to research about lifted inference:

- Probabilistic relational models (PRMs)
- Dynamic probabilistic relational models (DPRMs)