Frequent Pattern Discovery from OWL DLP Knowledge Bases

Joanna Józefowska,
Agnieszka Ławrynowicz,
Tomasz Łukaszewski

Institute of Computing Science
Poznań University of Technology

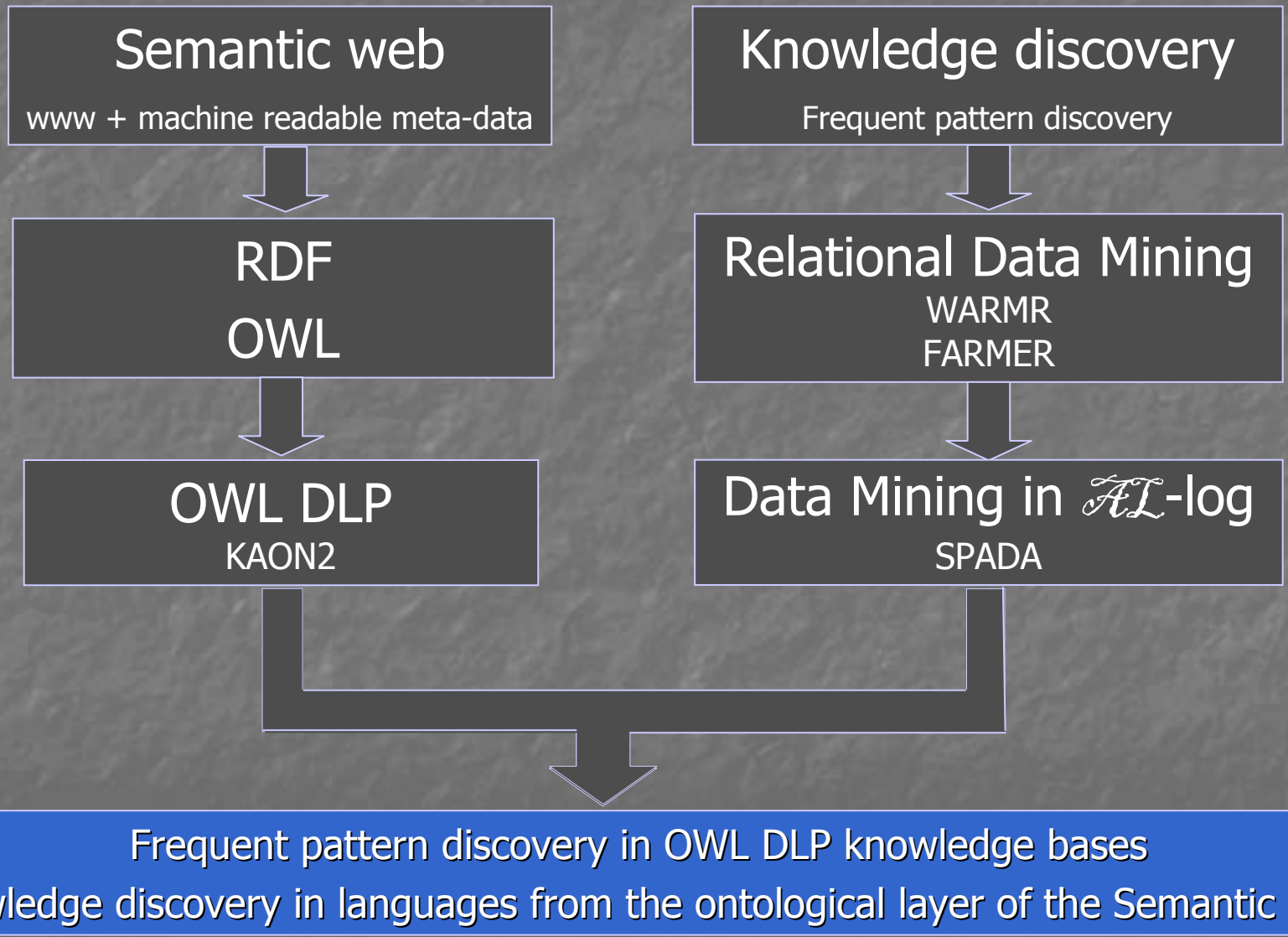
Outline

- Motivation
- Related research
- Problem formulation
- Algorithm
- Experiment
- Conclusions and further research

Motivation

- data is complex by nature in many domains,
- domain knowledge is available, but often implicit or not formalized,
- **using domain knowledge represented in languages with formal semantics may:**
  - improve the efficiency of data mining,
  - enable easier interpretation of results,
  - improve knowledge reorganization.

Related research



Related work - discussion

- **RDM methods**
  - generality relation used, θ-subsumption, only approximation of logical implication
  - problem with hierarchical/structural knowledge
- **SPADA**
  - works on restricted version of *AL*-log (DL component is restricted)
  - either the patterns that can be found contain concepts only from the same level of taxonomy or some concepts are replicated in some, lower levels of taxonomy

General problem

Data mining:

- using background/domain knowledge
- represented in expressive languages

from the intersection/combination of OWL DL and Logic Programming

Frequent patterns

Ontology

TBox

$$Man \sqsubseteq Client$$
$$Woman \sqsubseteq Client$$
$$Gold \sqsubseteq CreditCard$$
$$Classic \sqsubseteq CreditCard$$

$$\top \sqsubseteq \forall hasCreditCard.Client$$
$$\top \sqsubseteq \forall hasCreditCard^{-}.CreditCard$$

ABox

Client
Kowalski
Nowak
Zalewska

Man
Kowalski
Nowak

Woman
Zalewska

hasCreditCard	
Kowalski	c123
Kowalski	c456
Zalewska	c789

Gold
c123
c456

Classic
c789

Example pattern (query)

Q(x):-Client(x), Man(x), hasCreditCard(x, y), Gold(y)

key variable (the only distinguished one)

Frequent patterns

Ontology

TBox

$$Man \sqsubseteq Client$$
$$Woman \sqsubseteq Client$$
$$Gold \sqsubseteq CreditCard$$
$$Classic \sqsubseteq CreditCard$$

$$\top \sqsubseteq \forall hasCreditCard.Client$$
$$\top \sqsubseteq \forall hasCreditCard^{-}.CreditCard$$

ABox

Client
Kowalski
Nowak
Zalewska

Man
Kowalski
Nowak

Woman
Zalewska

hasCreditCard	
Kowalski	c123
Kowalski	c456
Zalewska	c789

Gold
c123
c456

Classic
c789

Example pattern

Q(x):-Client(x), Man(x), hasCreditCard(x, y), Gold(y)

key variable (the only distinguished one)

reference concept

Frequent patterns

Ontology

TBox

$$Man \sqsubseteq Client$$
$$Woman \sqsubseteq Client$$
$$Gold \sqsubseteq CreditCard$$
$$Classic \sqsubseteq CreditCard$$

$$\top \sqsubseteq \forall hasCreditCard.Client$$
$$\top \sqsubseteq \forall hasCreditCard^{-}.CreditCard$$

ABox

Client
Kowalski
Nowak
Zalewska

Man
Kowalski
Nowak

Woman
Zalewska

hasCreditCard	
Kowalski	c123
Kowalski	c456
Zalewska	c789

Gold
c123
c456

Classic
c789

Example pattern

Q(**x**):-**Client**(**x**), Man(**x**), hasCreditCard(**x**, y), Gold(y)

key variable (the only distinguished one)

reference concept

Q_{ref}(x):-Client(x)

Reference query

$$support(\hat{C}, Q, \mathcal{KB}) = \frac{|answerset(\hat{C}, Q, \mathcal{KB})|}{|answerset(\hat{C}, Q_{ref}, \mathcal{KB})|} = \frac{1}{3} \approx 0.33$$

Task

Given ontology O with TBox and ABox

Find a set of queries, $\mathcal{Q}$ that capture the characteristics of ABox – frequent patterns

Pattern discovery task

Given

- $\mathcal{KB}$ – a knowledge base in OWL DLP,
- a set of patterns in the language $\mathcal{L}$ of queries Q that all contain a reference concept $\hat{C}$,
- a minimum support threshold *minsup* specified by the user,

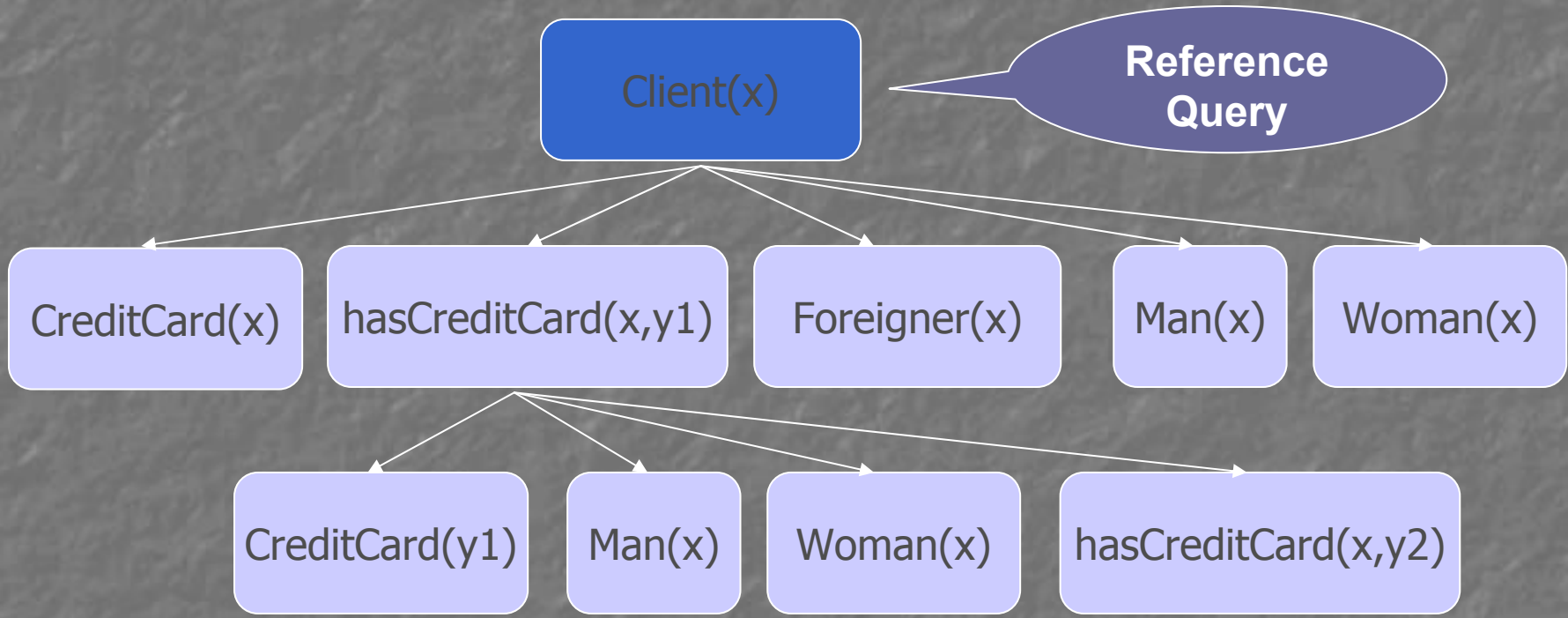
  and assuming that queries with support s are frequent in $\mathcal{KB}$ given $\hat{C}$ if $s \geq$ *minsup*,

  the task of frequent pattern discovery is to find the set $\mathcal{F}$ of frequent queries.

Constructing the Query Set

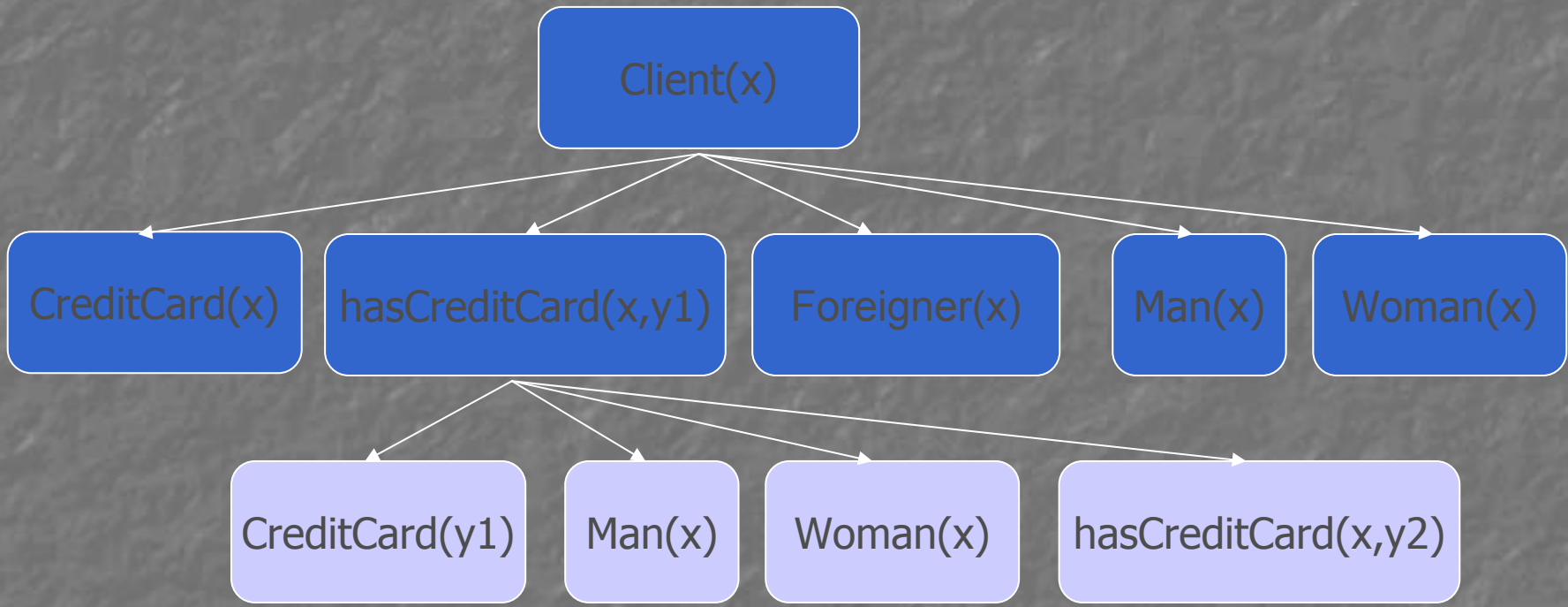
1. Define the threshold – minsup;
2. Start with a reference query Q_{ref};
   Run the reference query, count the number of instances;
3. Create a tree of conjunctive queries rooted at Q_{ref}
   1. Create new query by adding new constraint based on TBox;
   2. Check consistency by reasoning over TBox (if not consistent than do not keep it in the tree);
   3. Ask Query (if below minsup threshold than do not keep it in the tree).

Constructing the Query Set



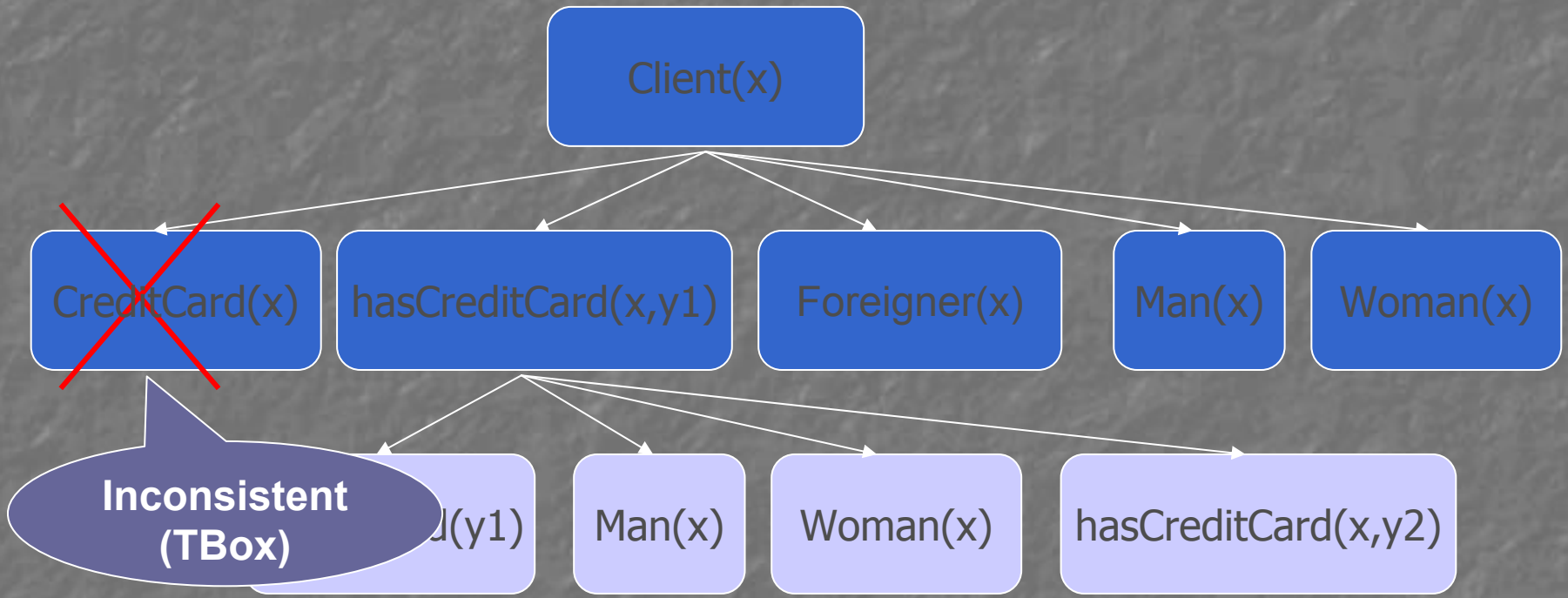
- nodes correspond to atoms of the query
- every path from the root to a node corresponds to a query

Constructing the Query Set



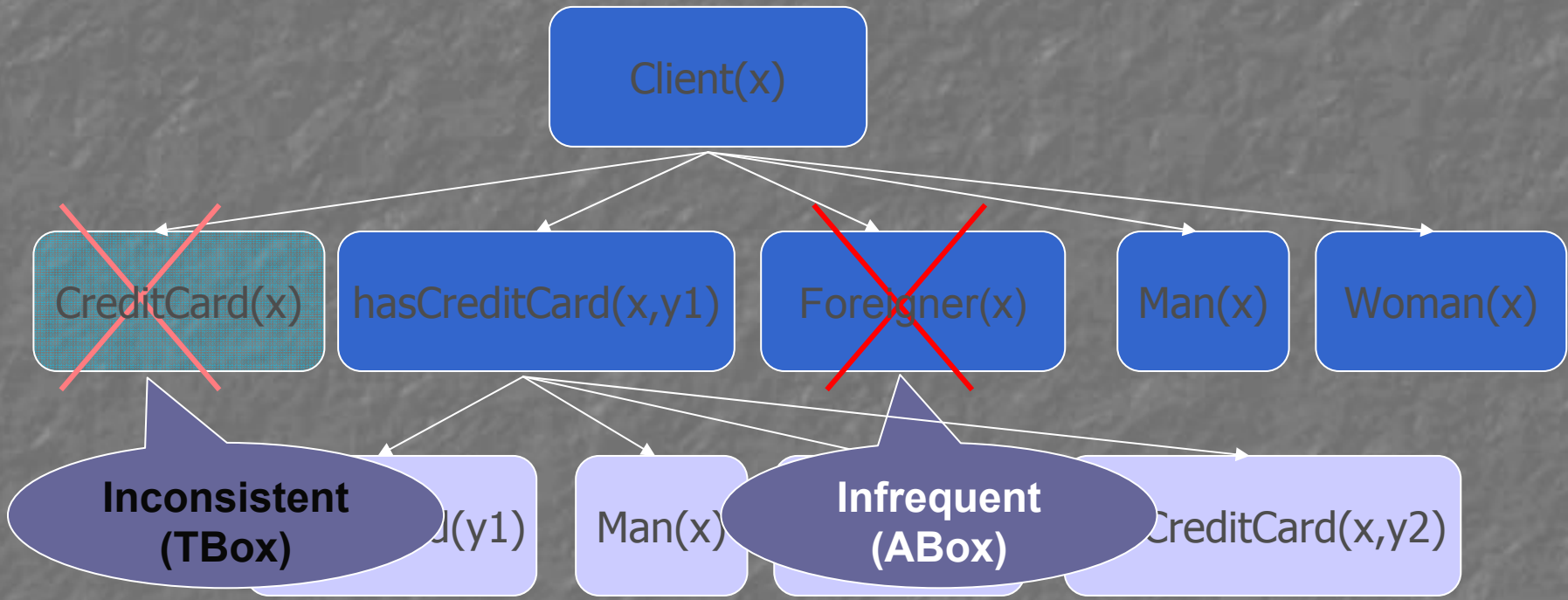
- nodes correspond to atoms of the query
- every path from the root to a node corresponds to a query

Constructing the Query Set



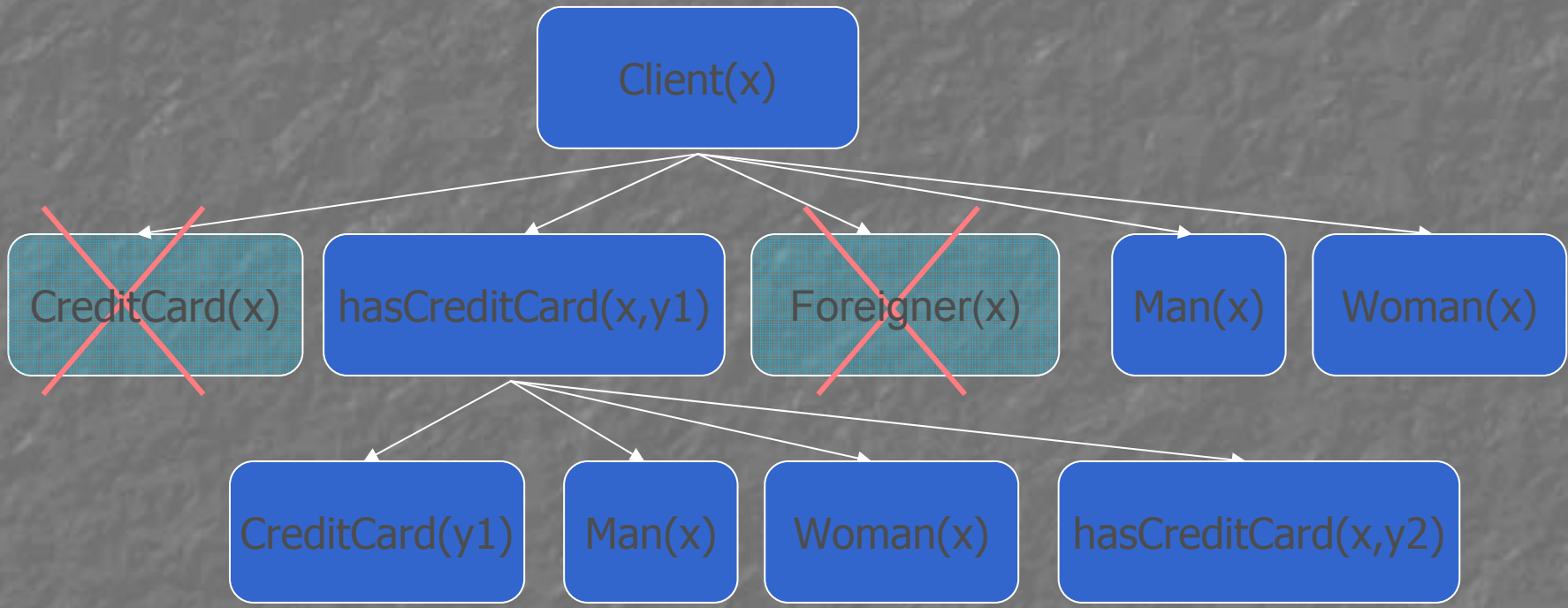
- nodes correspond to atoms of the query
- every path from the root to a node corresponds to a query

Constructing the Query Set



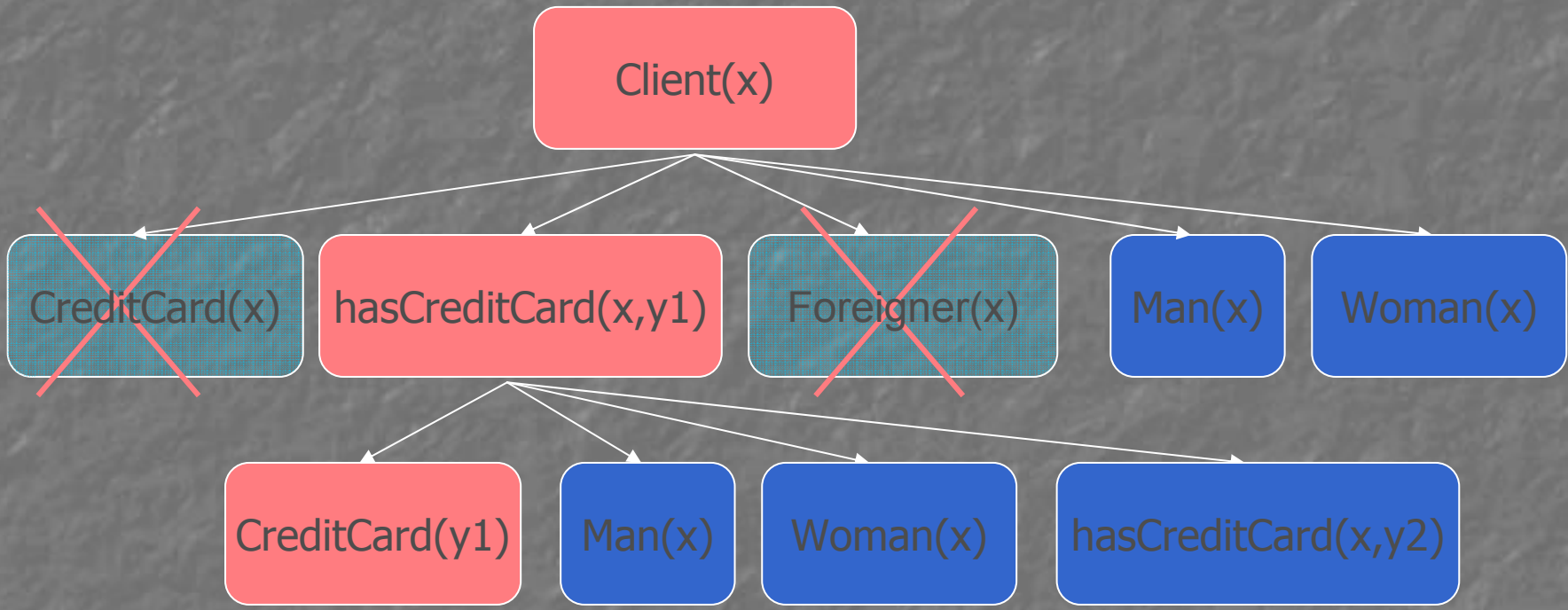
- nodes correspond to atoms of the query
- every path from the root to a node corresponds to a query

Constructing the Query Set



- nodes correspond to atoms of the query
- every path from the root to a node corresponds to a query

Constructing the Query Set



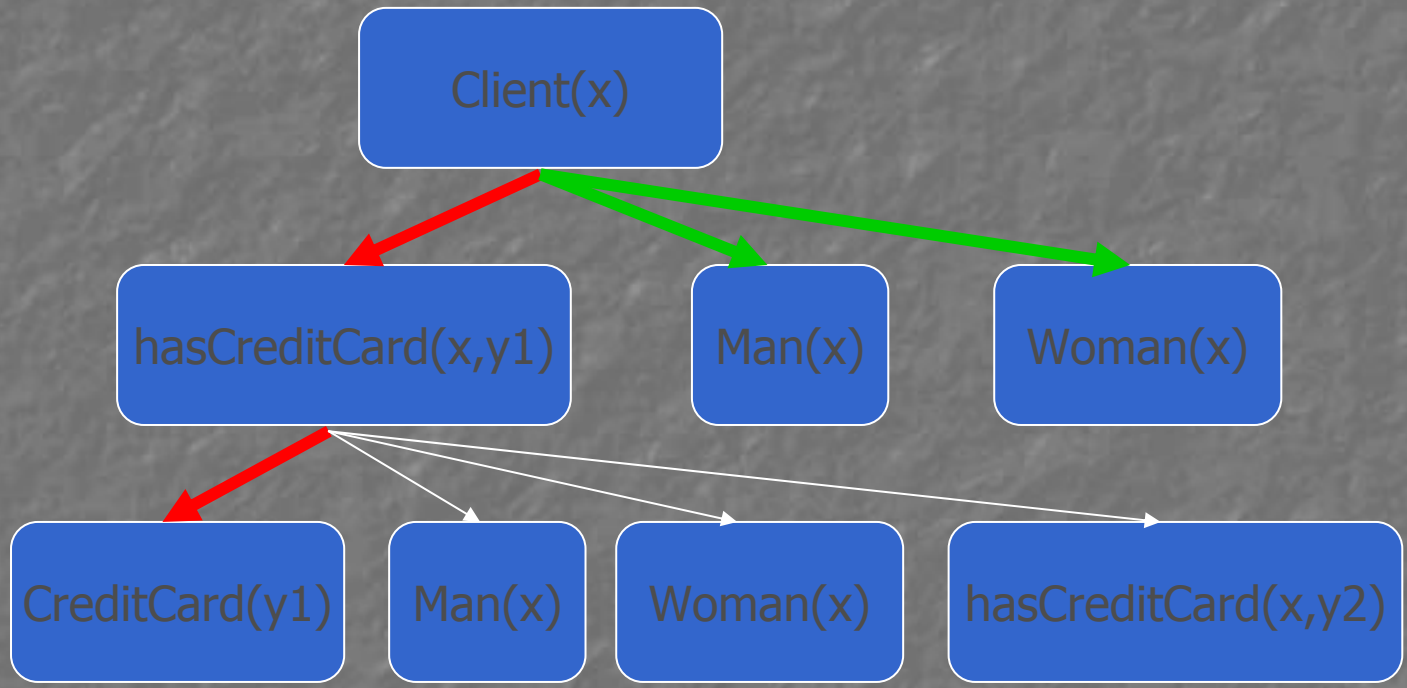
- nodes correspond to atoms of the query
- every path from the root to a node corresponds to a query:

q(x):-Client(x), hasCreditCard(x,y1), CreditCard(y1)

Overview of the method

- taxonomy classification
- for every predicate when first appears in the tree compute the list of ***admissible predicates***, according to TBox
  - e.g. for **Man:**
    - admissible: Foreigner, hasCreditCard ….
      - *Man(x), Foreigner(x) -> satisfiable*
    - not admissible: Woman, CreditCard…
      - *Man(x), Woman(x) ->unsatisfiable*
- use refinement rules for tree expansion

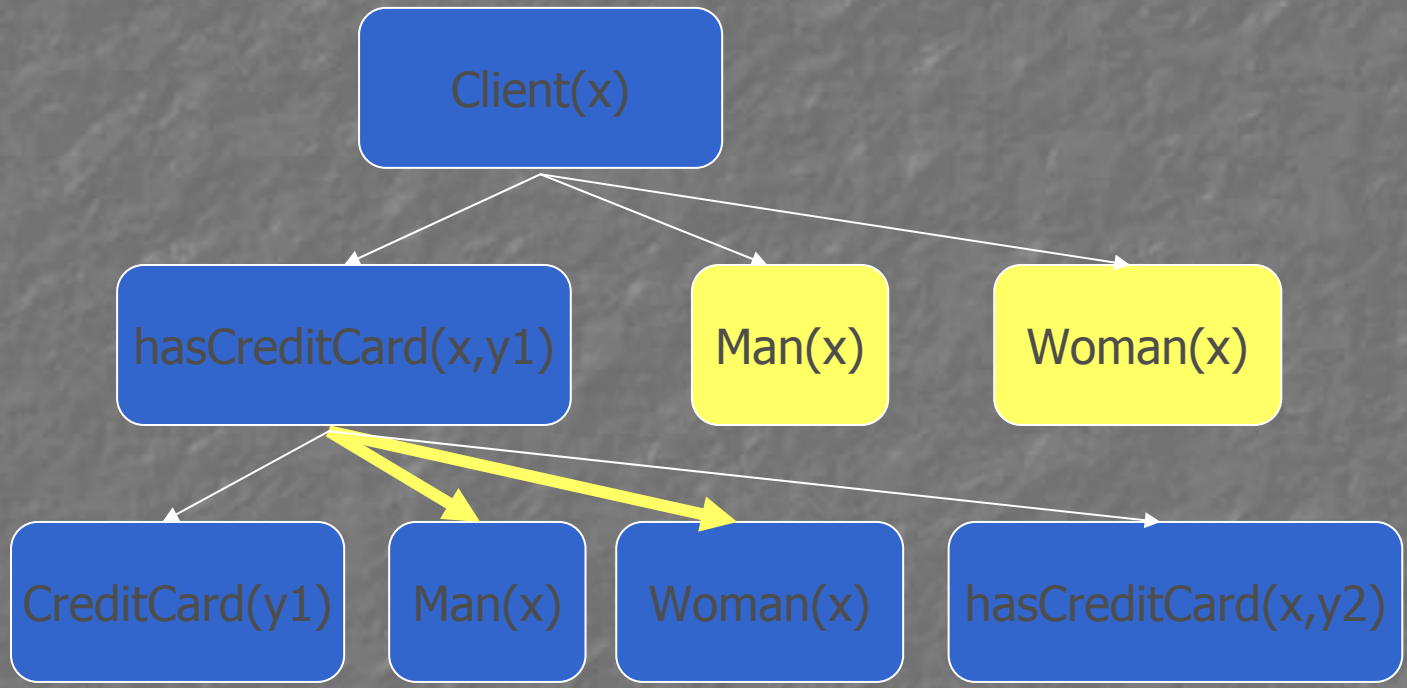
Refinement rules - tree expansion



1. dependent atoms of a given node (sharing variable):
- top-level concepts/properties
- direct subconcepts/subproperties

Only admissible predicates for given predicate are considered

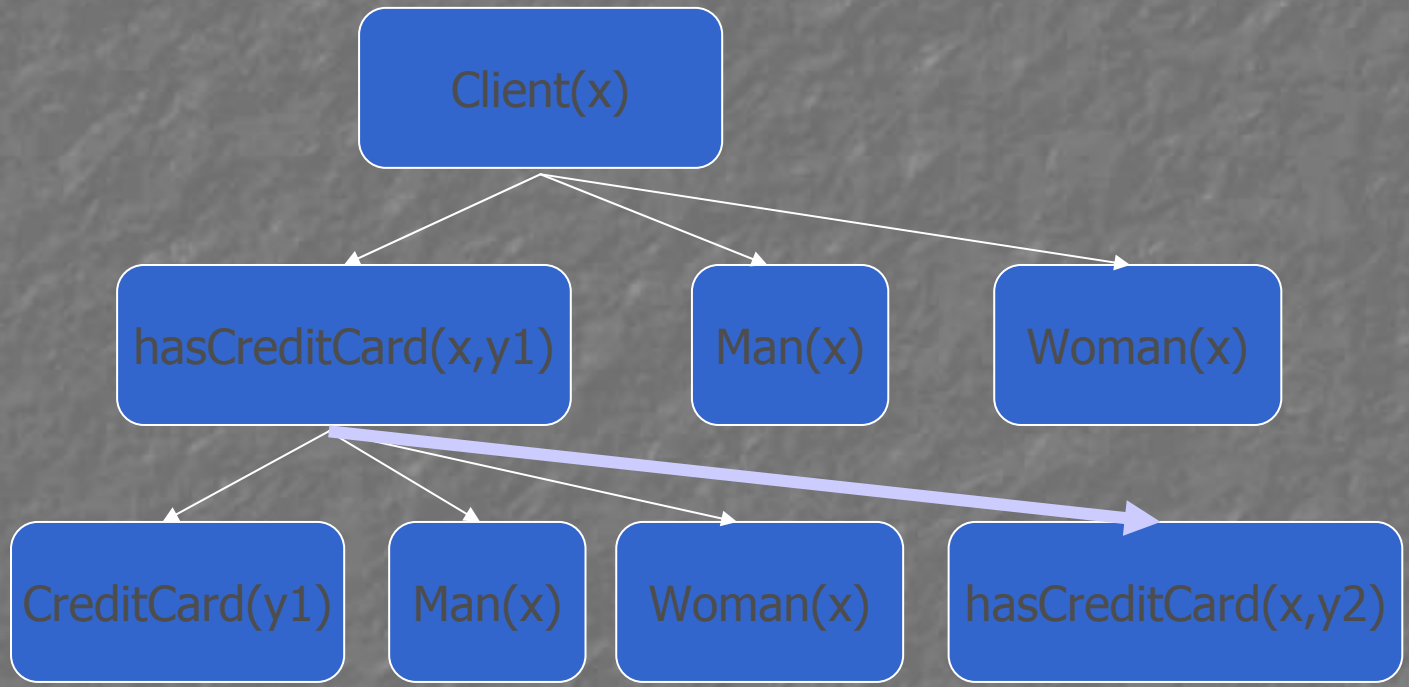
Refinement rules - tree expansion



2. right brothers of a given node

Right brother copying mechanism takes care that all possible subsets are generated and only one permutation out of a set of dependent atoms is considered.

Refinement rules - tree expansion



3. a copy of the given node

Experimental setup

- Test data
  - *ontology* defined on the basis of relational dataset from PKDD Discovery Challenge'99 (financial services), published online
- Implementation: Java
- Reasoning engine: KAON2
  - recently published tests (among others on our ontology): when small TBox and big ABox – faster then other reasoning engines

Experimental results

	The original method				No functionality check				Naive approach			
Level	Cnd	Pt	Pt/Cnd	Time	Cnd	Pt	Pt/Cnd	Time	Cnd	Pt	Pt/Cnd	Time
1	4	2	50,00%	4s	4	2	50,00%	5s	50	2	4,00%	22s
2	6	5	83,33%	2s	6	5	83,33%	3s	155	11	7,10%	57s
3	28	25	89,29%	11s	31	28	90,32%	12s	418	98	23,44%	148s
4	140	139	99,29%	65s	184	181	98,37%	99s	1731	953	55,05%	1182s
5	786	773	98,35%	511s								
Total	**964**	**944**	**97,93%**									

Cnd - Number of candidates

Pt - Number of patterns

- experiment interrupted due to the high computation time

Conclusions & Future work

- in our approach domain knowledge can be taken into consideration in the knowledge discovery process
- further intensive experimental tests needed
- problem with convergence
  - without UNA chains like:

    q(x):-Client(x),isOwnerOf(x,y1), isOwnerOf(x,y2), isOwnerOf(x,y3),…

    where to *y1*, *y2*, *y3…*the same individual can be assigned

Further research:

- techniques for improving efficiency of the exact algorithm
- considering more expressive languages within the DL-safe rules approach
- heuristic algorithms

Thank you for your attention!