Complex Data Mining & Workflow Mining

An Introduction to Multi-Relational Data Mining

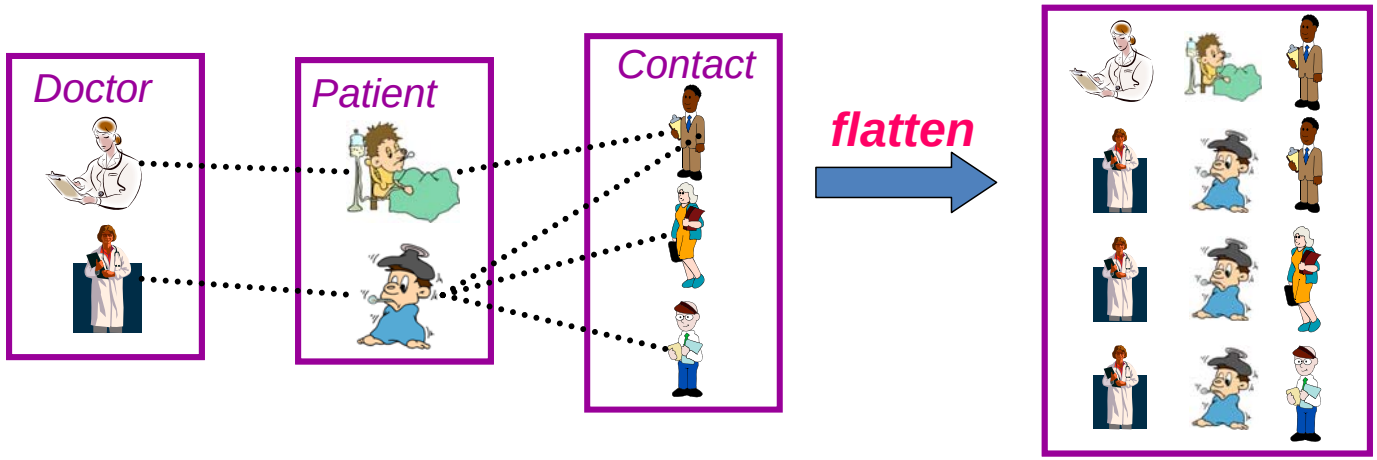
ICAR CNR
Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Outline

- Introduzione e concetti di base
  - Motivazioni, applicazioni
  - Concetti di base nell'analisi dei dati complessi
- Web/Text Mining
  - Concetti di base sul Text Mining
  - Tecniche di data mining su dati testuali
- Graph Mining
  - Introduzione alla graph theory
  - Principali tecniche e applicazioni
- Multi-Relational data mining
  - Motivazioni: da singole tabelle a strutture complesse
  - Alcune delle tecniche principali
- Workflow Mining
  - I workflow: grafi con vincoli
  - Frequent pattern discovery su workflow: motivazioni, metodi, applicazioni

Traditional Data Mining

- Works on single "flat" relations



  - ***Single table assumption:*** Each row represents an object and columns represent properties of objects

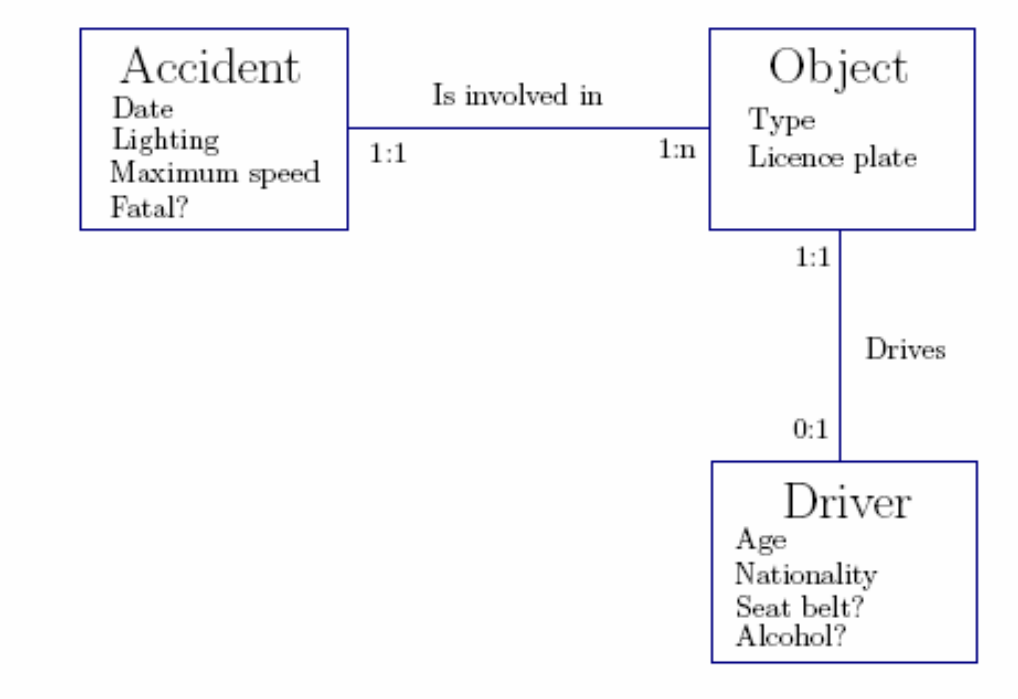
- Drawbacks:
  - Lose information of linkages and relationships
  - Cannot utilize information of database structures or schemas

Multi-Relational Data Mining (MRDM)

- (Multi-)Relational data mining algorithms can analyze data distributed in multiple relations, as they are available in relational database systems.
  - These algorithms come from the field of inductive logic programming (ILP)
  - ILP has been concerned with finding patterns expressed as logic programs
- Motivations
  - Most structured data are stored in relational databases
  - MRDM can utilize linkage and structural information
- Knowledge discovery in multi-relational environments
  - Multi-relational rules
  - Multi-relational clustering
  - Multi-relational classification
  - Multi-relational linkage analysis
  - …

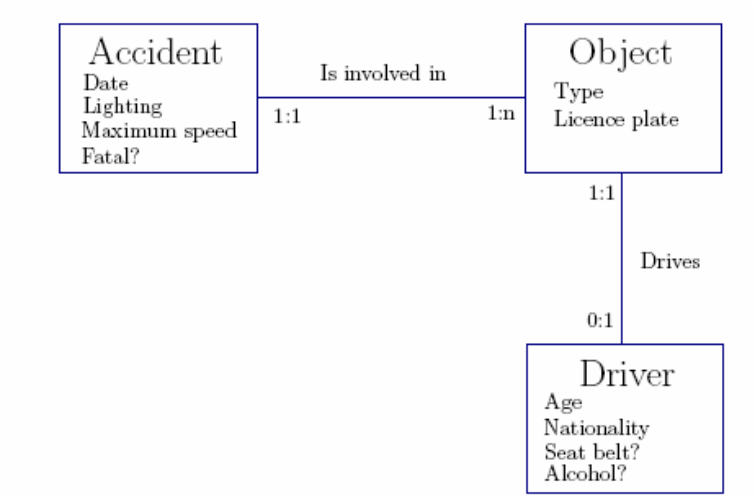
Why MRDM?

- An example: accidents



Which accidents are likely to be fatal?

- How can we find a subgroup like:
  - *If an accident takes place in a road with maximum speed of 100km/h, and involves a car whose driver is not wearing a seat-belt, then the accident is likely to be fatal*
  - The description uses information for all three tables



Example 2: customers

ID	*Name*	*First Name*	*Street*	*City*	*Sex*	*Social Status*	*Income*	*Age*	*Response*
3478	Smith	John	38 Lake St	Seattle	M	single	160k	32	Y
3479	Doe	Jane	45 Sea St	Venice	F	married	180k	45	N
…	…	…	…	…	…	…	…	…	…

Example 2: Standard DM

- In the customer table we can add as many attributes about our customers as we like.
  - *A person's number of children*
- For other kinds of information the single-table assumption turns out to be a significant limitation
  - *Add information about orders placed by a customer, in particular*
    - *Delivery and payment modes*
    - *With which kind of store the order was placed (size, ownership, location)*
  - *For simplicity, no information on the goods ordered*

ID	*Name*	*First Name*	*…*	*Respo nse*	*Delivery mode*	*Payment mode*	*Store size*	*Store type*	*Locati on*
3478	Smith	John	…	Y	regular	cash	small	franchis	city
3479	Doe	Jane	…	N	express	credit	large	indep	rural
…	…	…	…	…	…	…	…	…	…

Example 2: Standard DM (II)

- This solution works fine for once-only customers
- What if our business has repeat customers?
- Under the single-table assumption we can make one entry for each order in our customer table

ID	*Name*	*First Name*	*…*	*Respo nse*	*Delivery mode*	*Payment mode*	*Store size*	*Store type*	*Locati on*
3478	Smith	John	…	Y	regular	cash	small	franchis	city
3478	Smith	John	…	Y	express	check	small	franchis	city
…	…	…	…	…	…	…	…	…	…

- We have usual problems of non-normalized tables
  - Redundancy, anomalies, ...

Example 2: Standard DM (III)

- one line per order → analysis results will really be about orders, not customers, which is not what we might want!
- Aggregate order data into a single tuple per customer.

ID	*Name*	*First Name*	*…*	*Response*	*No. of orders*	*No. of stores*
3478	Smith	John	…	Y	3	2
3479	Doe	Jane	…	N	2	2
…	…	…	…	…	…	…

- No redundancy. Standard DM methods work fine, but
- There is a lot less information in the new table
  - What if the payment mode and the store type are important?

Example 2: Relational Data

- A database designer would represent the information in our problem as a set of tables (or relations)

ID	*Name*	*First Name*	*Street*	*City*	*Sex*	*Social Status*	*Income*	*Age*	*Response*
3478	Smith	John	38 Lake St	Seattle	M	single	160k	32	Y
3479	Doe	Jane	45 Sea St	Venice	F	married	180k	45	N
…	…	…	…	…	…	…	…	…	…

Cust ID	*Order ID*	*Store ID*	*Delivery mode*	*Payment mode*
3478	213444	12	regular	cash
3478	372347	19	regular	cash
3478	334555	12	express	check
…	…	…	…	…

Store ID	*size*	*Type*	*Location*
12	small	franchis	city
19	large	indep	rural
…	…		…

Example 2: Relational patterns

- *Relational patterns* involve multiple relations from a relational DB
- They are typically stated in a more expressive language than patterns defined on a single data table.
  - Relational classification rules
  - Relational regression trees
  - Relational association rules

IF Customer(C1,N1,FN1,Str1,City1,Zip1,Sex1,SoSt1, In1,Age1,Resp1)
AND order(C1,O1,S1,Deliv1, Pay1)
AND Pay1 = credit_card
AND In1 $\geq$ 108000
THEN Resp1 = Yes

Relational patterns

IF Customer(C1,N1,FN1,Str1,City1,Zip1,Sex1,SoSt1, In1,Age1,Resp1)
AND order(C1,O1,S1,Deliv1, Pay1)
AND Pay1 = credit_card
AND In1 $\geq$ 108000
THEN Resp1 = Yes

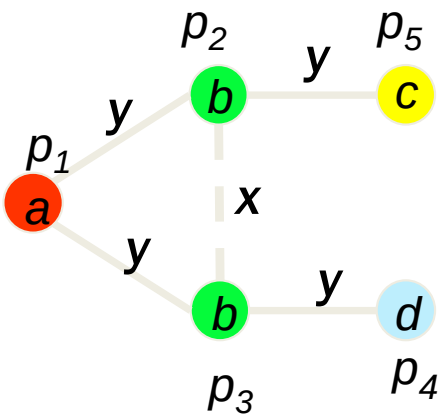
good_customer(C1) $\leftarrow$
customer(C1, N1,FN1,Str1,City1,Zip1,Sex1,SoSt1, In1,Age1,Resp1) $\wedge$
order(C1,O1,S1,Deliv1, credit_card) $\wedge$
In1 $\geq$ 108000

This relational pattern is expressed in a subset of first-order logic!

A relation in a relational database corresponds to a predicate in predicate logic (see ***deductive databases***)

Why is MRDM of interest?

- Graph databases
  - Two relations:



node

ID	Label
p_1	a
p_2	b
p_3	b
p_4	c
p_5	d

edge

Src	Dst	weight
p1	p2	y
p1	p3	y
p2	p5	y
p2	p3	x
p3	p4	y

- Workflows
  - Can extend with further information

MRDM tasks

- Multi-relational Classification
  - Classify objects based on properties spread through multiple tables
- Multi-relational Clustering Analysis
  - Clustering objects with multi-relational information
- Probabilistic Relational Models
  - Model cross-relational probabilistic distributions

Inductive Logic Programming (ILP): general framework

- Find a hypothesis that is consistent with background knowledge (training data)
  - FOIL, Golem, Progol, TILDE, …
- Background knowledge
  - Relations (predicates), Tuples (ground facts)

Training examples

Daughter(mary, ann)	+
Daughter(eve, tom)	+
Daughter(tom, ann)	–
Daughter(eve, ann)	–

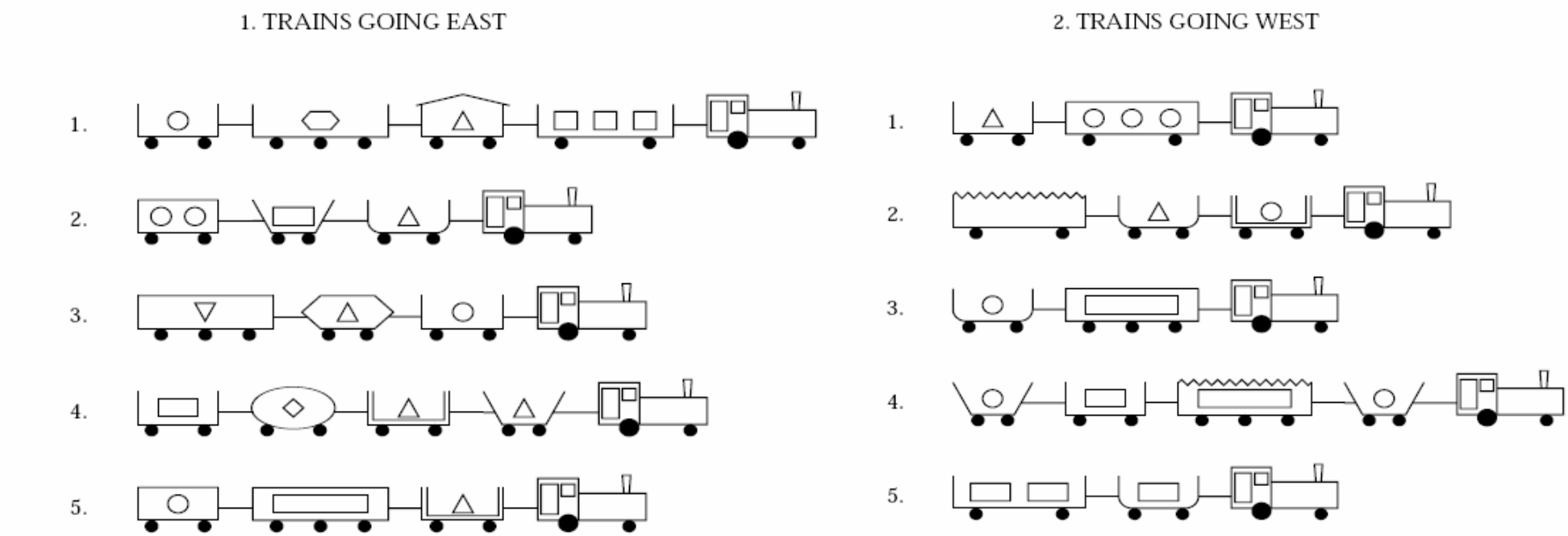
Background knowledge

Parent(ann, mary)
Parent(ann, tom)
Parent(tom, eve)
Parent(tom, ian)

Female(ann)
Female(mary)
Female(eve)

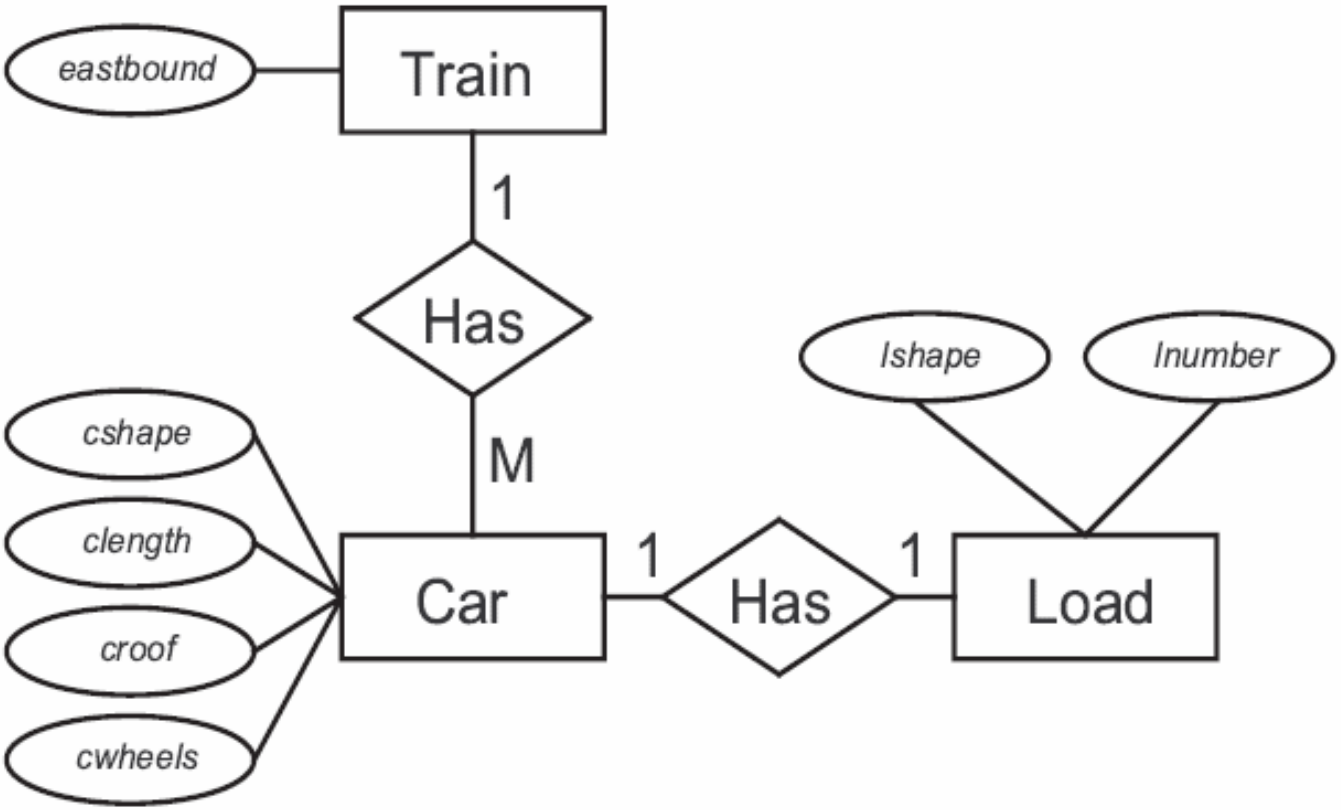
- Hypothesis
  - The hypothesis is usually a set of rules, which can predict certain attributes in certain relations
  - Daughter(X,Y) ← female(X), parent(Y,X)

ILP setting: an example

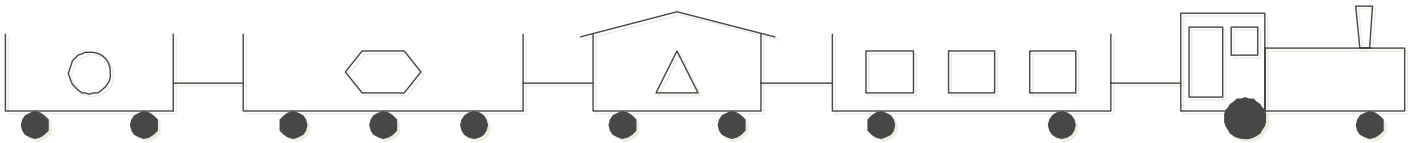


- How do we distinguish eastbound from westbound trains?
  - A train is eastbound if it contains a short closed car

Trains: the data model (II)



Trains: FO representation



- Example:

  `eastbound(t1).`

- Background theory:

```
car(t1,c1).    car(t1,c2).    car(t1,c3).    car(t1,c4).
rectangle(c1). rectangle(c2). rectangle(c3). rectangle(c4).
short(c1).     long(c2).      short(c3).     long(c4).
none(c1).      none(c2).      peaked(c3).    none(c4).
two_wheels(c1). three_wheels(c2). two_wheels(c3). two_wheels(c4).
load(c1,l1).   load(c2,l2).   load(c3,l3).   load(c4,l4).
circle(l1).    hexagon(l2).   triangle(l3).  rectangle(l4).
one_load(l1).  one_load(l2).  one_load(l3).  three_loads(l4).
```

ILP Approaches

- Top-down Approaches (e.g. FOIL)

  while(enough examples left)

  generate a rule

  remove examples satisfying this rule

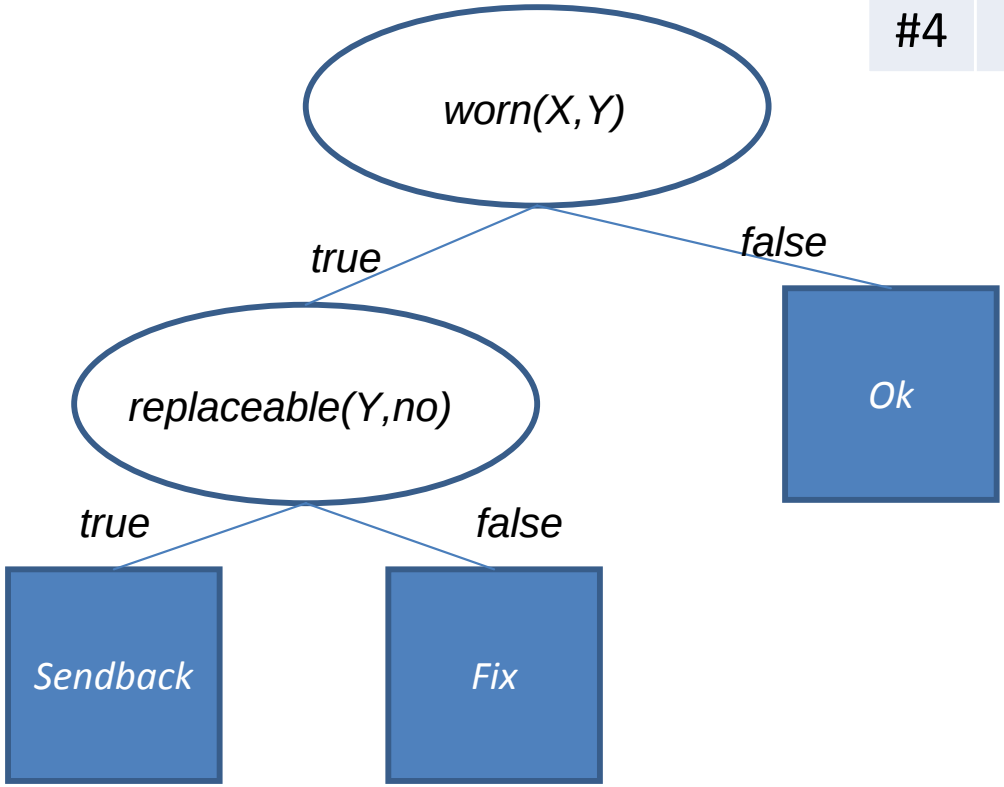
- Bottom-up Approaches (e.g. Golem)

  Use each example as a rule

  Generalize rules by merging rules

- Decision Tree Approaches (e.g. TILDE)

TILDE: Relational decision trees



instance

ID	Class
#1	Fix
#2	Sendback
#3	Sendback
#4	Ok

worn

ID	Worn
#1	Gear
#1	Chain
#2	Engine
#2	Chain
#3	wheel

replaceable

Component	Replaceable
Gear	Yes
Chain	Yes
Engine	No
Wheel	no

Multi-relational Clustering

- RDBC
  - Distance-based agglomerative clustering
- First-order K-Means clustering
  - Distance-based K-Means clustering
- Relational distance measure
  - Measure distance between two objects by their attributes and their neighbor objects in relational databases

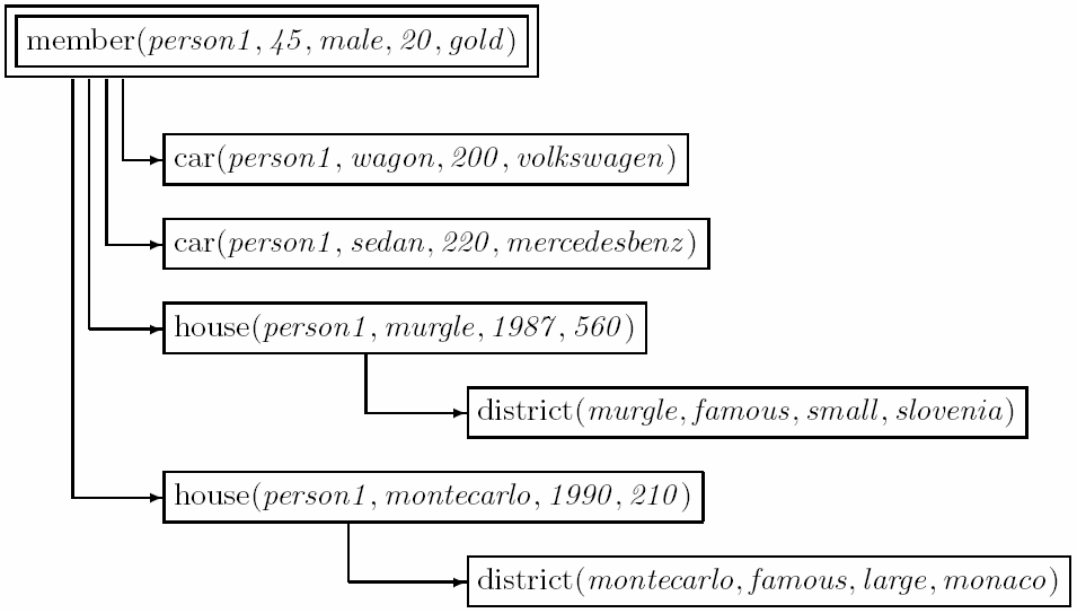
Relational Distance Measure

- RIBL (Relational Instance-Based Learning)
  - To measure distance between objects O_1 and O_2, neighbor objects of O_1 and O_2 are also considered.

Relational data

member(person1 ; 45 ; male; 20 ; gold)
member(person2 ; 30 ; female; 10 ; platinum)
car(person1 ; wagon; 200 ; volkswagen)
car(person1 ; sedan; 220 ; mercedesbenz)
car(person2 ; roadster; 240 ; audi)
car(person2 ; coupe; 260 ; bmw)
house(person1 ; murgle; 1987 ; 560)
house(person1 ; montecarlo; 1990 ; 210)
house(person2 ; murgle; 1999 ; 430)
district(montecarlo; famous; large; monaco)
district(murgle; famous; small ; slovenia)

Neighbor data of level 2

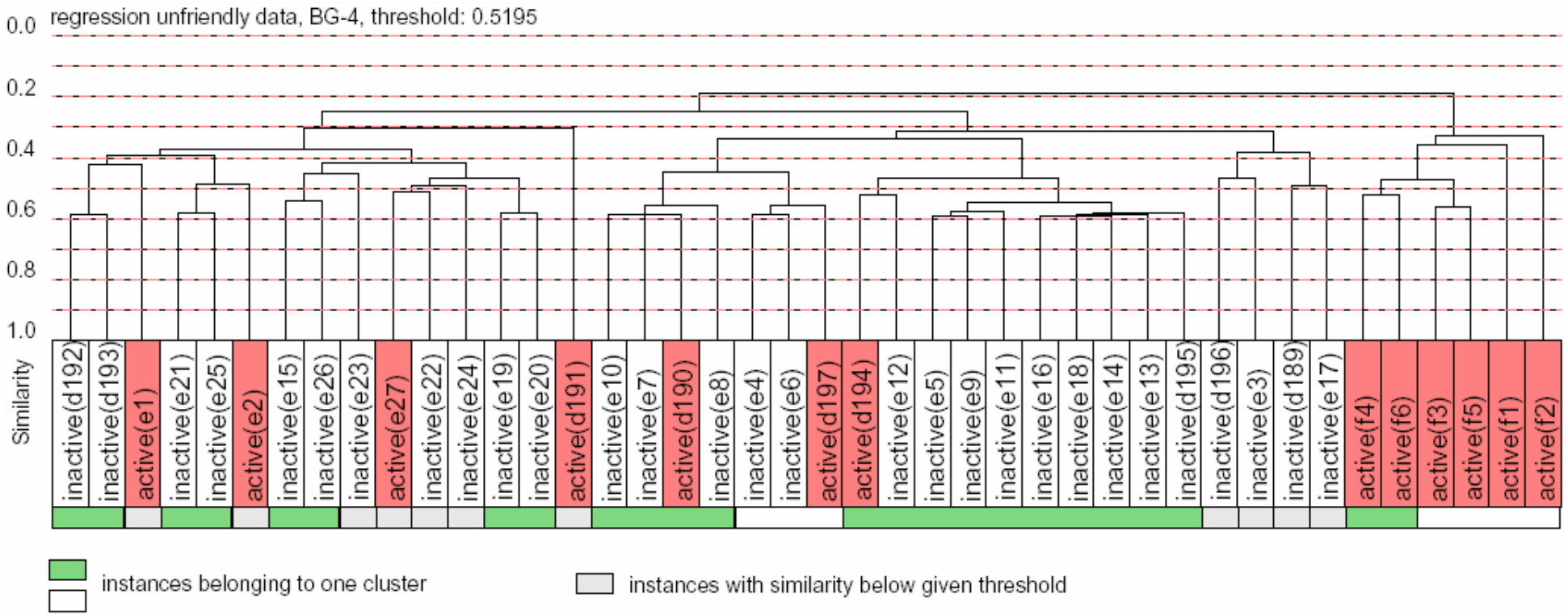


Relational Distance Measure (cont.)

- Distance between two objects O_1 and O_2 is defined by
  - Attributes of O_1 and O_2:
    - Discrete attribute: distance = 1 if equal; 0 otherwise.
    - Numerical attribute: distance = diff / range
  - Neighbor objects of O_1 and O_2:
    - Defined recursively
- Comments
  - Advantage: considering related objects in distance measure
  - Disadvantage: very expensive to compute, because of the huge number of related objects

RDBC: Relational Distance-Based Clustering

- Use distance measure of RIBL
- Agglomerative clustering approach
  - Every object is used as a cluster at beginning
  - Keep merging clusters that are most similar



First-order K-Means Clustering

- K-Means algorithm
  1. Select *k* initial objects as cluster centers
  2. Assign objects to nearest clusters
  3. Repeat step 2 until stable
- K-Means is very expensive
  - Computing distance between an object and a cluster is very expensive
- K-Means can be replaced by K-Medoids
  - For each cluster, use an object that is nearest to all objects in this cluster as the center

Multi-relational Clustering: Summary

- Extend clustering algorithms to multi-relational environments
- Use distance measures that consider related objects
- Very expensive because the numbers of related objects are usually very large

Multi-relational association rule

has

KID	OBJECT
Joni	ice-cream
Joni	dolphin
Elliot	piglet
Elliot	gnu
Elliot	lion

likes

KID	OBJECT
Joni	ice-cream
Joni	piglet
Elliot	ice-cream

prefers

KID	OBJECT	TO
Joni	ice-cream	pudding
Joni	pudding	raisins
Joni	giraffe	gnu
Elliot	lion	ice-cream
Elliot	piglet	dolphin

likes(KID, piglet), likes(KID, ice-cream)

→ likes (KID, dolphin) (9%, 85%)

likes(KID, A), has(KID, B) → prefers (KID, A, B) (70%, 98%)

Mining relational associations

Problem statement

Given:

- a deductive relational database *D*
- a couple of thresholds, *minsup* and *minconf*

Find

all association rules that have support and confidence greater than *minsup* and *minconf* respectively.

Mining relational associations (II)

Problem decomposition

- Find *large* (or frequent) atomsets
- Generate *highly-confident* association rules

Representation issues

A deductive relational database is a relational database which may be represented in *first-order logic* as follows:

- Relation $\Leftrightarrow$ Set of ground facts (EDB)
- View $\Leftrightarrow$ Set of rules (IDB)

Finding frequent atomsets (I)

likes(joni, ice-cream) **atom**

has

KID	OBJECT
Joni	ice-cream
Joni	dolphin
Elliot	piglet
Elliot	gnu
Elliot	lion

likes

KID	OBJECT
Joni	ice-cream
Joni	piglet
Elliot	ice-cream

prefers

KID	OBJECT	TO
Joni	ice-cream	pudding
Joni	pudding	raisins
Joni	giraffe	gnu
Elliot	lion	ice-cream
Elliot	piglet	dolphin

likes(KID, piglet), likes(KID, ice-cream) **atomset**

$\rightarrow$ *likes (KID, dolphin) (9%, 85%)*

likes(KID, A), has(KID, B) $\rightarrow$ *prefers (KID, A, B) (70%, 98%)*

Finding frequent atomsets (II)

Pattern Space

false

$\leq_\theta$

$Q_1 \equiv \exists$ is_a(X, large_town)
$\wedge$ intersects(X, R)
$\wedge$ is_a(R, road)

$\leq_\theta$

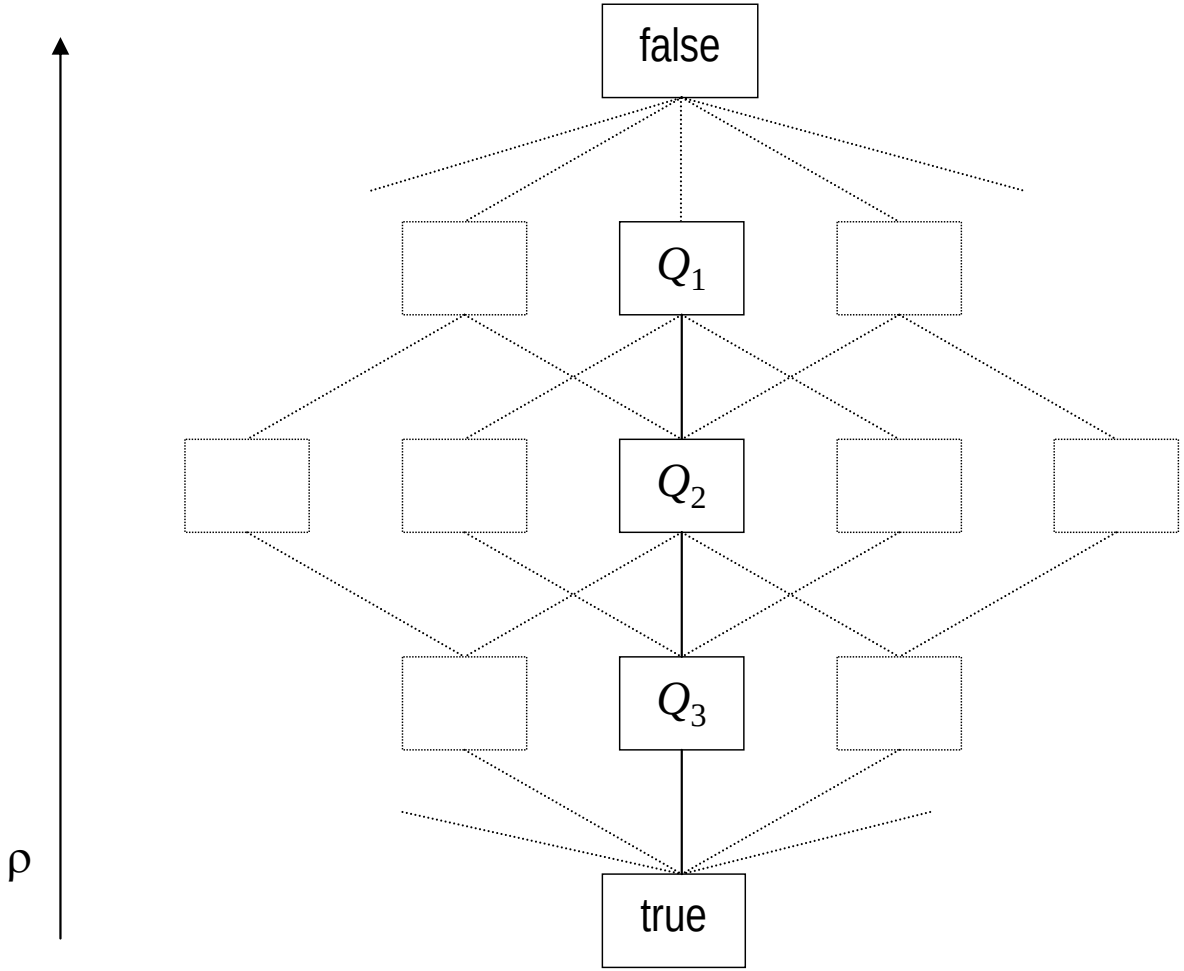
$Q_2 \equiv \exists$ is_a(X, large_town)
$\wedge$ intersects(X,Y)

$\leq_\theta$

$Q_3 \equiv \exists$ is_a(X, large_town)

$\leq_\theta$

true



Finding frequent atomsets (III)

The **WARMR** algorithm

Compute large 1-atomsets

Cycle on the size (k>1) of the atomsets

- **WARMR-gen** Generate candidate *k*-atomsets from large (*k*-1)-atomsets
- Generate large *k*-atomsets from candidate *k*-atomsets (cycle on the observations loaded from D)

until no more large atomsets are found.

Finding frequent atomsets (IV)

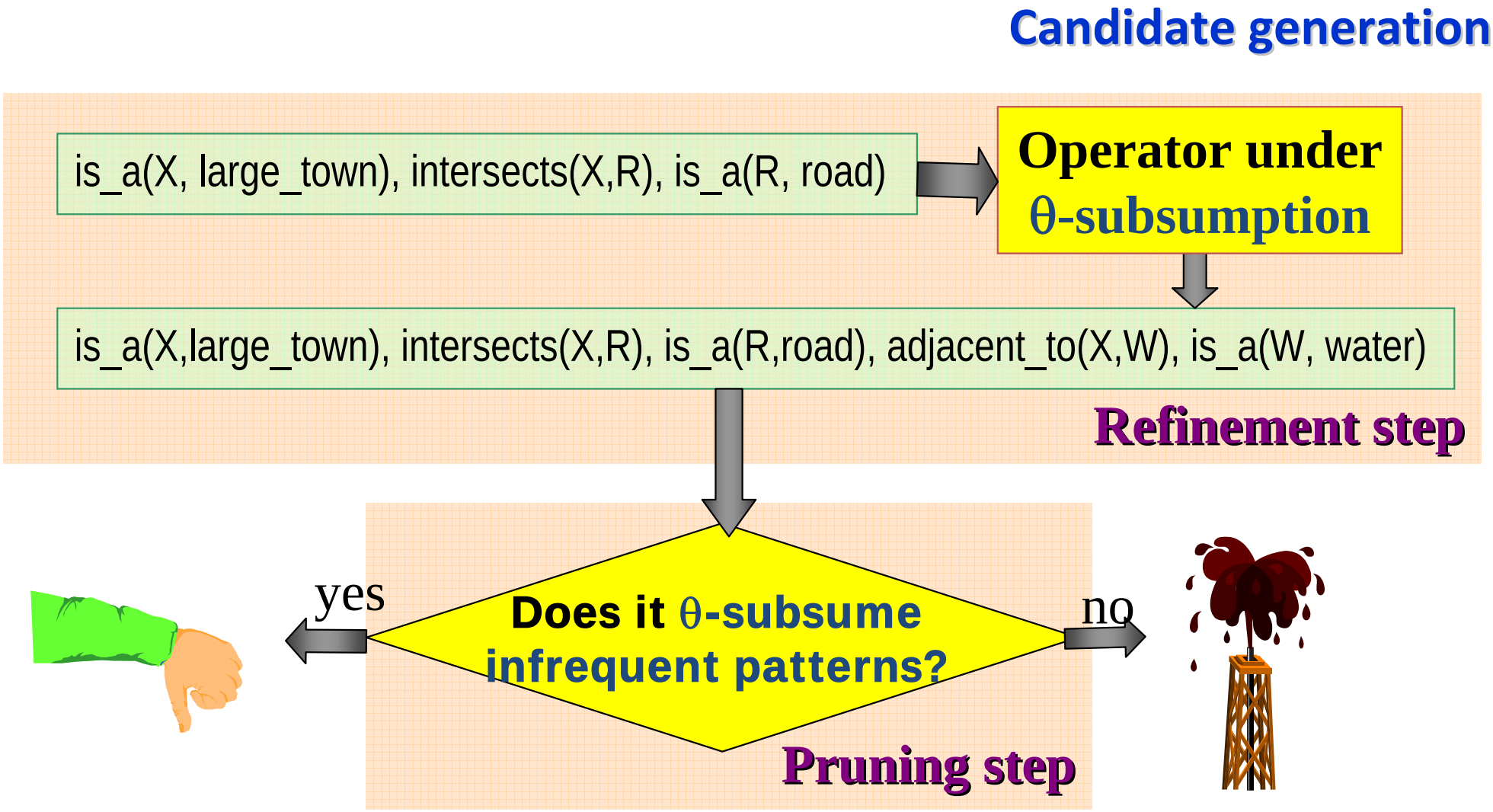
WARMR

- Breadth-first search on the atomset lattice
- Loading of an observation *o* from *D* (query result)
- Largeness of candidate atomsets computed by a coverage test

APRIORI

- *Breadth-first search on the itemset lattice*
- *Loading of a transaction t from D (tuple)*
- *Largeness of candidate itemsets computed by a subset check*

Mining relational association rules: Example (I)

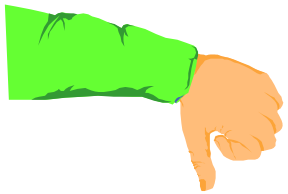


Mining relational association rules: Example (II)

Candidate evaluation

is_a(X, large_town), intersects(X,R), is_a(R, road), adjacent_to(X,W), is_a(W, water)

?- is_a(X, large_town), intersects(X,R), is_a(R, road), adjacent_to(X,W), is_a(W, water)



D

<X=barletta,R=a14,W=adriatico>
<X=bari,R=ss16bis,W=adriatico>
...

Large?

no

yes

Mining relational association rules: Example (III)

is_a(X, large_town), intersects(X,R), is_a(R, road), adjacent_to(X,W), is_a(W, water)

Rule generation

is_a(X, large_town), intersects(X,R), is_a(R, road), is_a(W, water)
→ adjacent_to(X,W) (62%, 86%)

yes

High confidence?

no