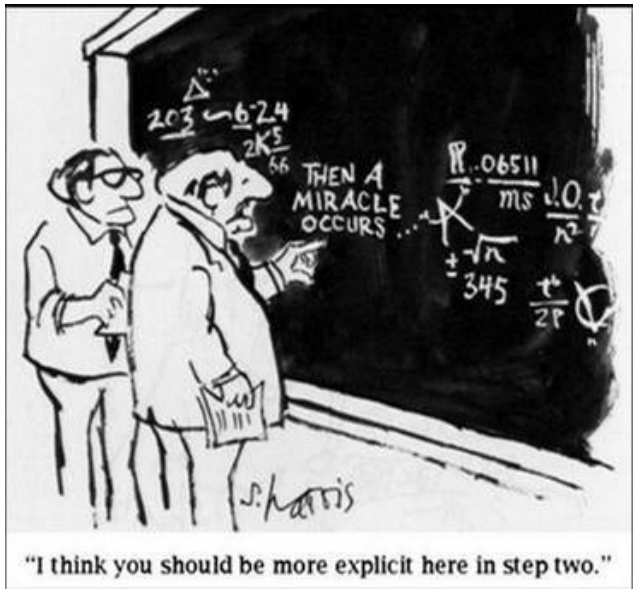
Logic-based Learning

Alessandra Russo and Mark Law

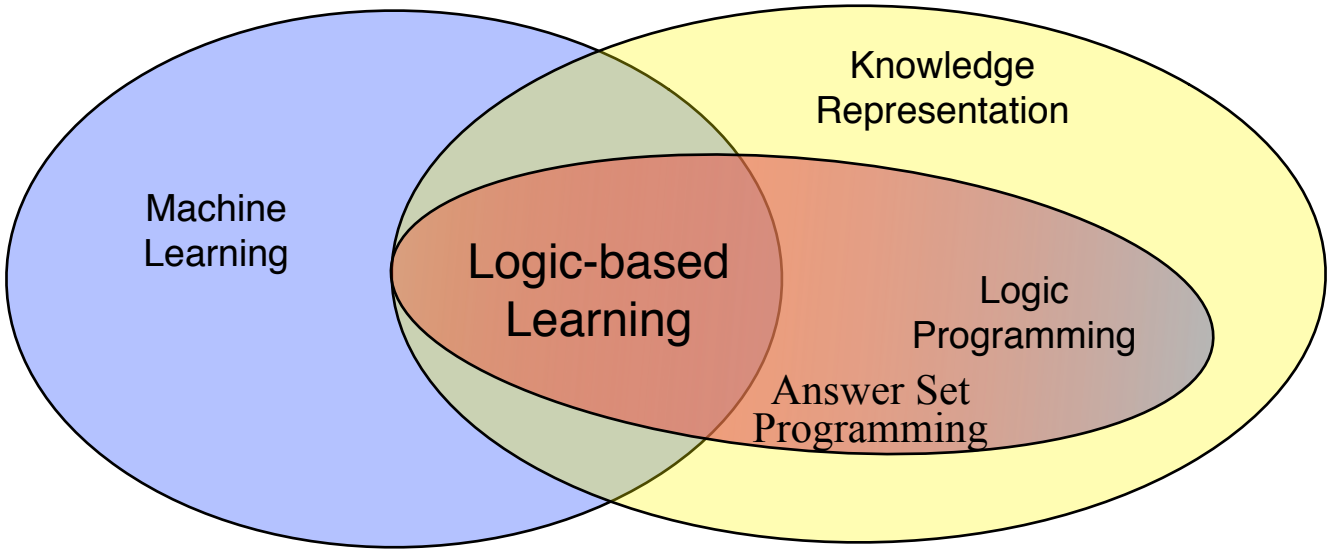
{a.russo, mark.law}@imperial.ac.uk



Objectives

- Modelling complex problems using logic.
  - Why logic. (e.g. deductive, abductive and inductive inference).
  - How logic-based learning differs from other machine learning techniques.
  - Different formalisms and semantics (e.g. (non-)monotonicity, (non-)determinism)
- Logic-based learning frameworks
  - Different formalisms and semantics (logic programming) (e.g. (non-)monotonicity, (non-)determinism, learning from entailments)
  - Algorithms and refinement operators (e.g. top-down, bottom-up, meta-level)
  - Extended learning frameworks, enabling learning more expressive programs (answer set programs including constraints, choices and preferences)
- Evaluating and enriching the goodness of the learned models
  - Learning from noisy data
  - Probabilistic logic programming(e.g. ProbLog)
  - Learning probabilistic logic programs (ProbFOIL)

What is logic-based learning?



- Knowledge extraction from observations
- Predictions about unseen data
- Ability to improve behavior over time.

- Declarative representation
- Clear semantics
- Inference mechanisms

Learning logic-based programs from
observations and given background knowledge

Why is it important?

Early Machine Learning techniques are limited in representation of data and learning outcomes:

- assume fixed number of features
- use propositional representation of data (e.g. set of features' values)
- learning outcomes depend on selection of features
- learning outcomes are in terms of single relationships between data and set of features.

Propositional representation is insufficient for domains with a variable number of features and multi-relations between features.

Logic-based learning is targeted to learning tasks in domains with a variable number of entities and multi-relations between entities.

Why is it important?

Imagine that we have two tables: a customer can make multiple purchases and we would like to characterise customers that spend a lot.

customer

CID	Name	Age	SpendALot

purchase

CID	ProdID	Date	Value	PaymentMode

purchase1

CID	Name	Age	SpendALot	ProdID	Date	Value	PaymentMode

We cannot analyse information wrt customers

customer1

CID	Name	Age	NofPurchases	TotalValue	SpendALot

Loss of information during aggregation

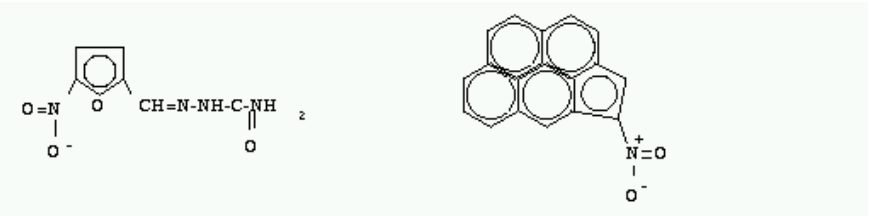
```
Customer(CID, Name, Age, yes) :-
        Age > 30,
        purchase(CID, PID, D, Value, PM),
        PM = creditcard,
        Value > 100.
```

Example1: Mutagenicity prediction

Some molecules (e.g. mutagenic) can cause mutation in DNA, others don't. Mutations can lead to cancer.

How to predict which molecules are active mutagenic?
Can we use just information about their molecular structure?

Active molecules

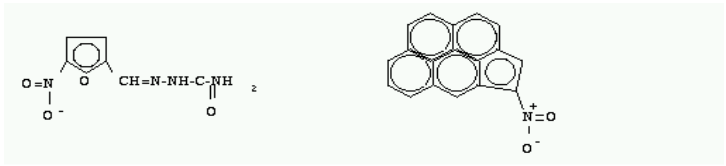


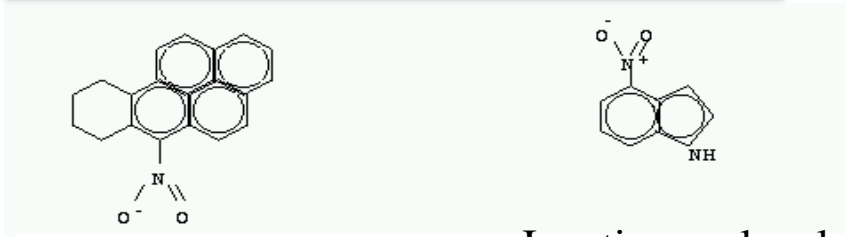
Inactive molecules

Compound	Attr1	Attr2	Attr3	Class
ID1	true	false	true	Active
ID2	true	true	true	Active
ID3	false	false	true	Inactive
……	……	……	……	……

Example1: Mutagenicity prediction

Active molecules





Inactive molecules

active(f1).
atom(f1, $f1_1$, c, 21, 0.817).
atom(f1, $f1_2$, c, 21, -0.143).
atom(f1, $f1_3$, c, 21, -0.143).
…….
bond(f1, $f1_1$, $f1_2$, 7).
bond(f1, $f1_2$, $f1_3$, 7).
bond(f1, $f1_3$, $f1_4$, 7).
……
logmutag(f1, 0.64).
lumo(f1, -1.785).
logp(f1, 1.01).

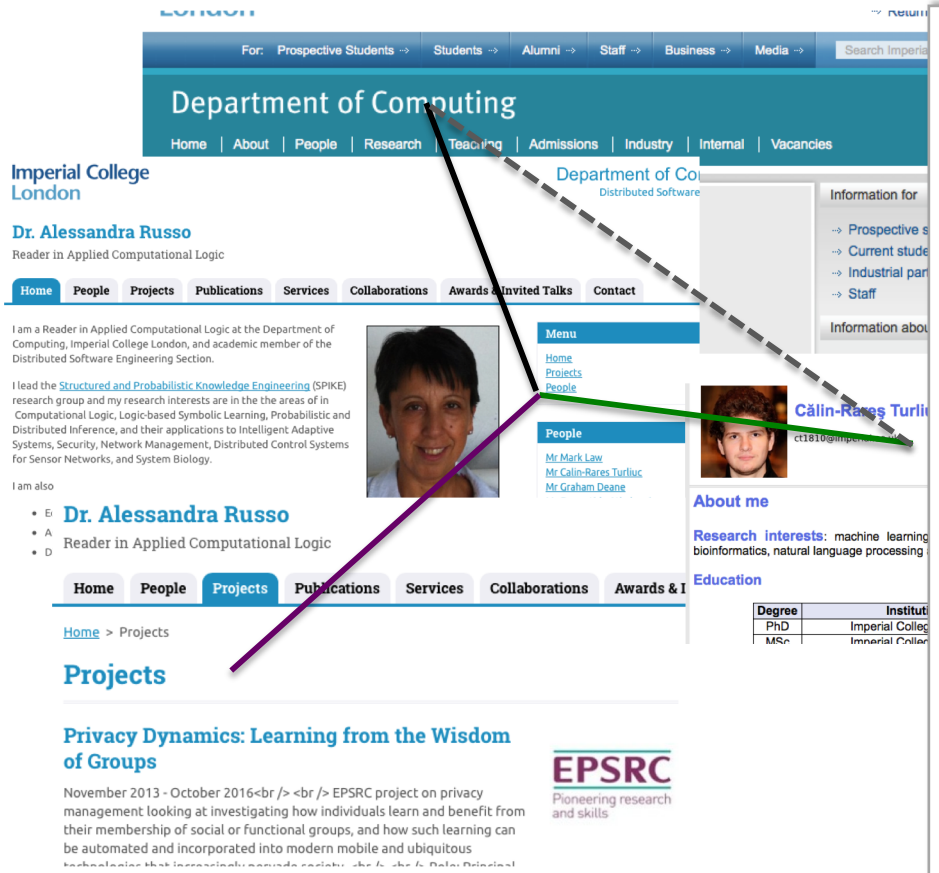
ring_size5(f1, [$f1_5$, $f1_1$, $f1_2$, $f1_3$, $f1_4$]).
……..

mutagenic(M) ← ring_size5(M, L),
atom(M, A1, _,_,_),
atom (M,A2,_,_,_),
member(A1,L), bond(M, A1, A2, 2)

Example2: Web Mining

Learning single large graphs or networks

e.g. protein networks, social networks, Web-based link network



staff(url1, alessandra)
staff(url2, stephen)
project(url4, privacy)
student(url6 rares)
student(url7 mark)
department(url9, computing)
.....
project(alessandra,privacy)
staffOf(alessandra, computing)
people(privacy, rares)
.....

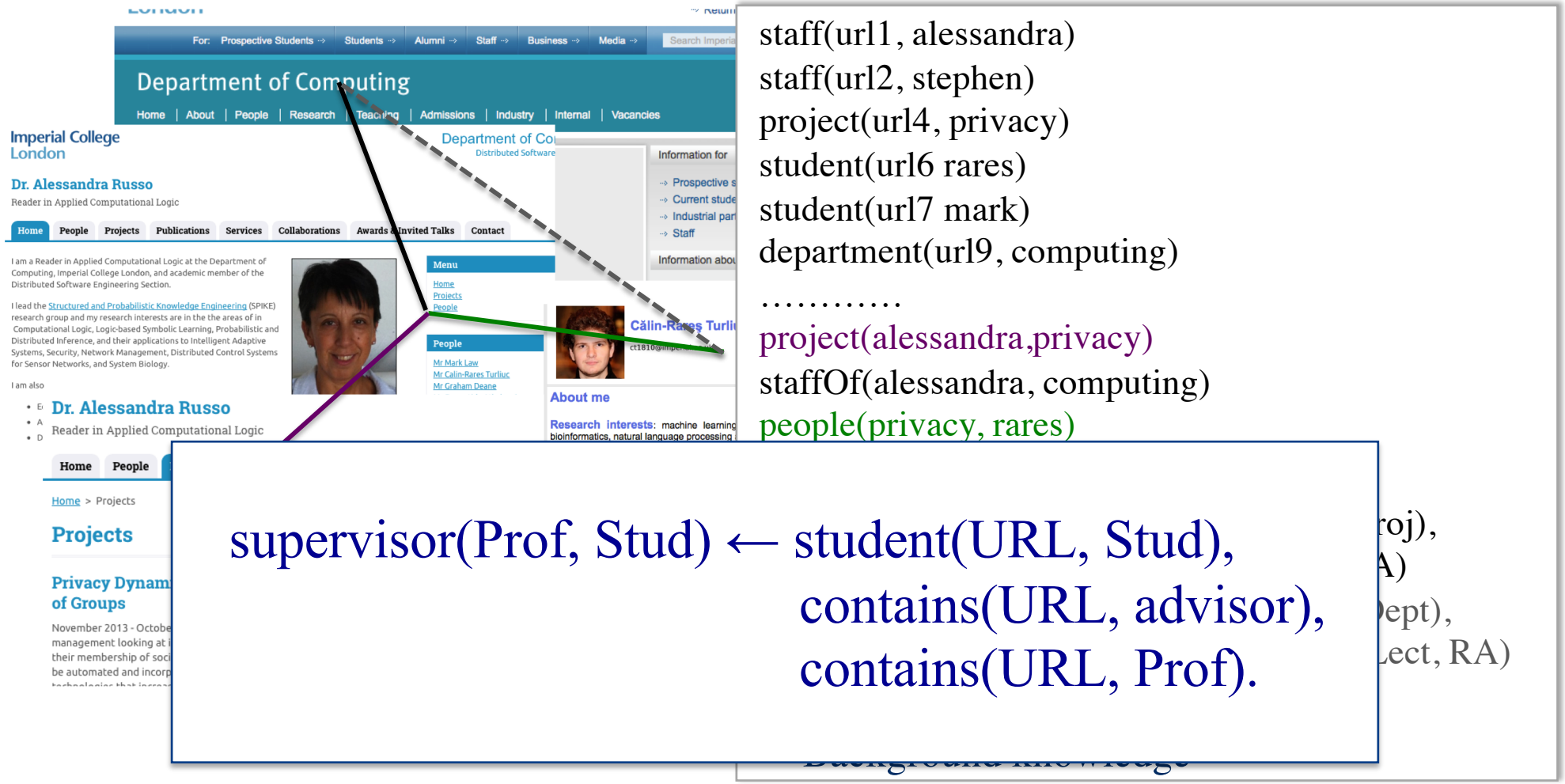
researcherOf(Lect, RA) ← project(Lect, Proj),
people(Proj, RA)

researcherAt(RA, Dept) ← staffOf(Lect, Dept),
researcherOf(Lect, RA)

Background knowledge

Example2: Web Mining

Learning task aims at learning single large graphs or networks
e.g. protein networks, social networks, Web-based link network



Example3: Learning Language in Logic

NLP ideal for logic-based learning: structured data, recursive definitions and background knowledge

Expressivity and flexibility of logic makes it suitable for NLP.

Hand-craft logic-based description of NL is too costly and not very robust.

Statistical approaches lead to descriptions that are difficult to interpret.

Learning task aims at learning grammar rules from a background knowledge of a partial grammar and (positive only or positive and negative) examples

- Learn missing grammatical concepts
- Revise existing grammar rules

Recursion

Multiple rules

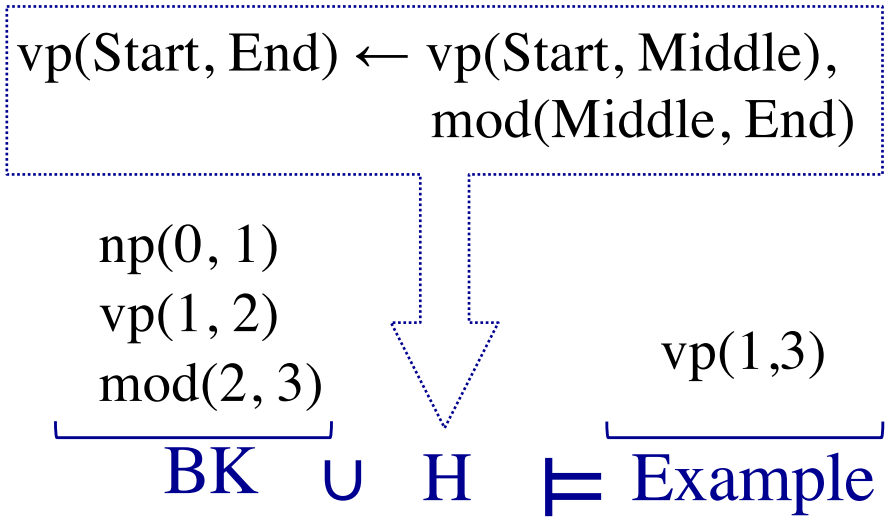
Example3: Learning Language in Logic

VP ::= VP MOD

$_0$ She $_1$ ran $_2$ quickly $_3$

However many other clauses could also explain the example

- vp(Start, End)
- vp(Start, End) ← vp(Start, Middle)



Negative examples are necessary to avoid over generalization

Training Examples				Background knowledge
vp(1, 3)	+	vp(0, 1)	-	np(0, 1)
		vp(0, 2)	-	vp(1, 2)
		vp(0, 3)	-	mod(2, 3)
		vp(2, 3)	-	

Example3: Learning Language in Logic

What about learning multiple target concepts?

What about learning concepts that are different from given example?

Secondary examples

Training Examples		Background knowledge (BK)	
s(0, 3)	+	np(X, Y) ← word("She", X, Y)	word("She", 0, 1)
		mod(X, Y) ← word(quickly, X, Y)	word(quickly, 2, 3)
		s(X, Y) ← np(X, Z), vp(Z, Y)	word(ran, 1, 2)
		vp(X, Y) ← v(X, Y)	← v(1, 3)

vp(Start, End) ← vp(Start, Middle),
mod(Middle,End)

v(1, 2) ←

BK ∪ H ⊨ s(0,3) Example

Example 4: Learning from human-robot dialogue

ANN and DNN have recently been able to support highly accurate NLP

- SyntaxNet (from Google)
- Core-NLP (from Stanford)

But, limited in extracting common-sense and domain expert knowledge, needed for deeper semantic Understanding.

https://1drv.ms/v/s!Aq-g0J2JpSjPox7CC5YSCvXLYNgl

Example5: Machine Comprehension of Text

- End-to-end semantic representation of English text into interpretable modes.
- Learning common-sense knowledge from Q&A

Text comprehension + Given correct and incorrect answers

Story:

1. John went to the local restaurant.
2. The waiter brought John a glass of water and took the order.
3. As John was waiting, he took out the book and began to read it.
4. The steak which he ordered finally arrived.
5. After John had finished the meal, he took the jacket but he forgot to take the book.
6. He paid the bill and went back to the hotel.

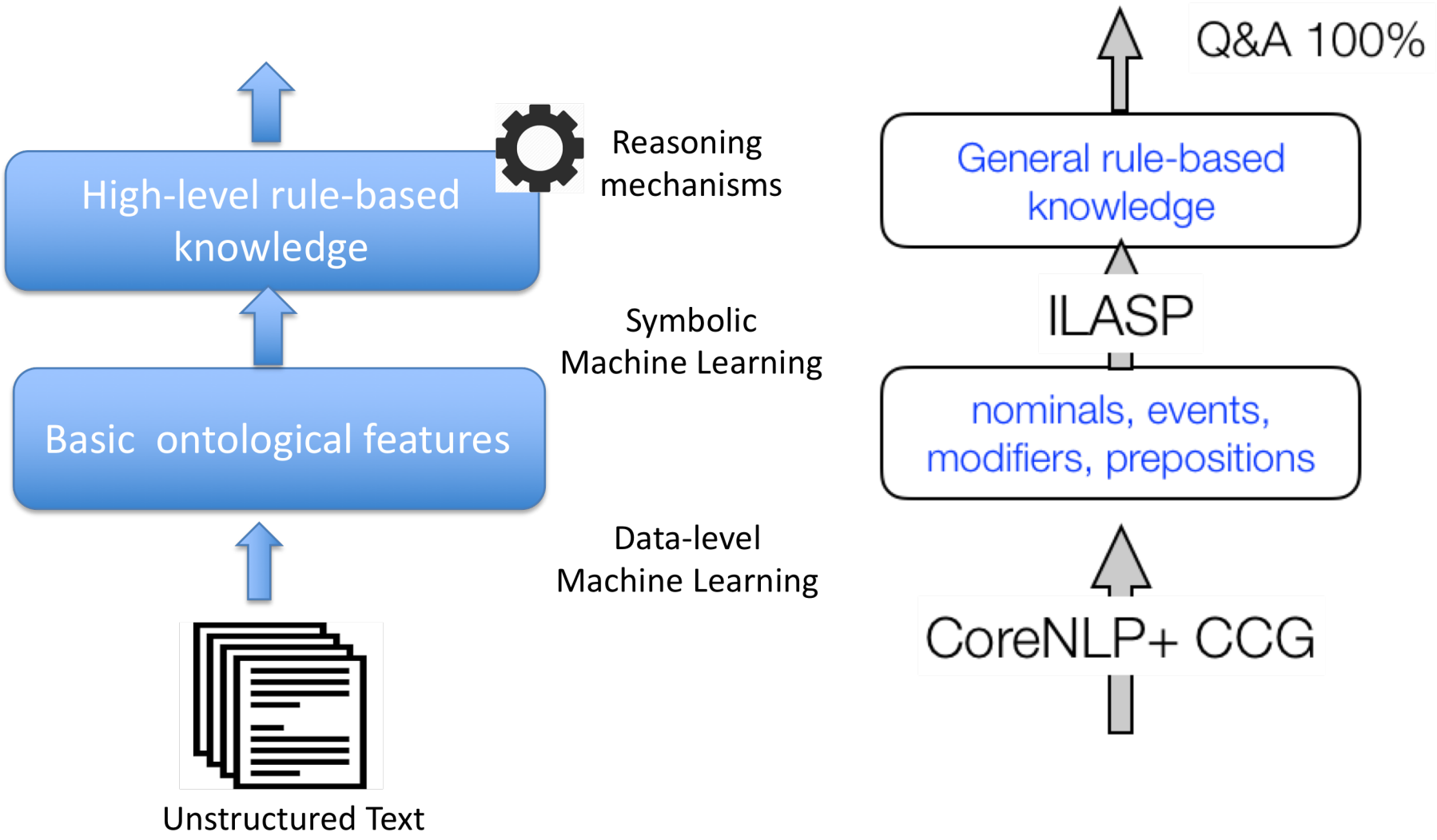
Questions:

1. Where is John?
2. Where was John before he went to the hotel?
3. Who took the order?
4. Who received a glass of water?
5. Did John have to wait?
6. What did John read?
7. What did John choose?
8. Where is the book?
9. Where is the jacket?

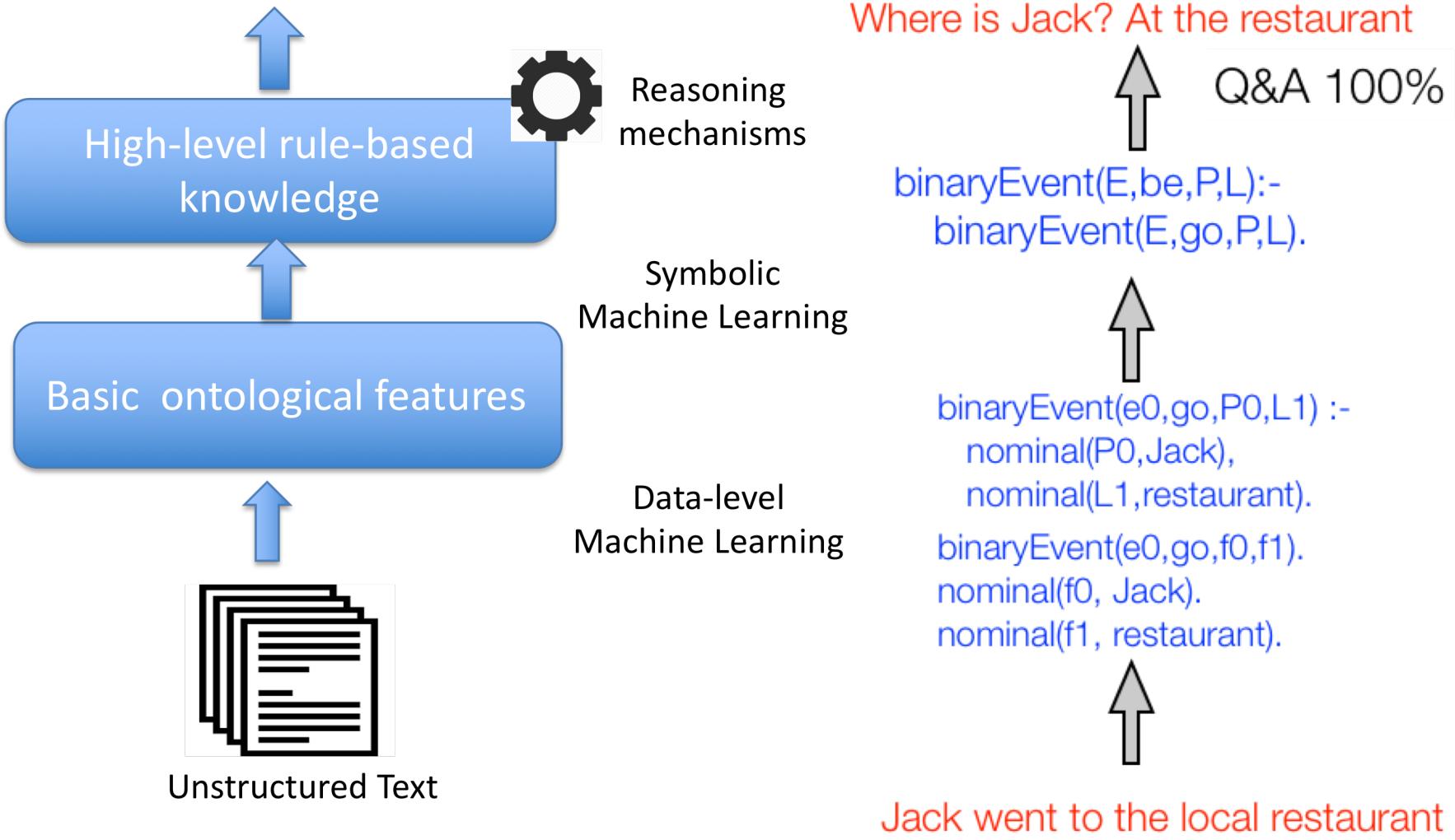
Learned Knowledge

- If a person goes somewhere then he/she will be at that location.
- If a person picks an object than he/she will carry that object
- If a person carries an object and he/she goes to a location then the object will be at that location

Example5: Machine Comprehension of Text



Example5: Machine Comprehension of Text



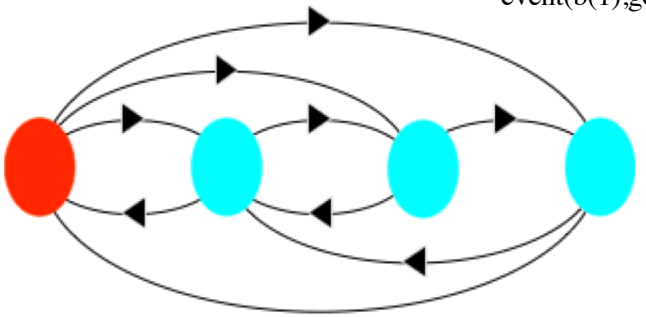
Example 6: Learning Software Models

Informal requirements

Execution traces

```
event(b(1),start,complete,0,[],noAgentIdAvailable).
event(b(1),applyForLicense,complete,1,[],noAgentIdAvailable).
event(b(1),attendClassesRideMotorBikes,complete,2,[],noAgentIdAvailable).
event(b(1),doTheoreticalExam,complete,3,[],noAgentIdAvailable).
event(b(1),obtainSpecialInsurance,complete,4,[],noAgentIdAvailable).
event(b(1),doPracticalExamRideMotorBikes,complete,5,[],noAgentIdAvailable).
event(b(1),getResult,complete,6,[],noAgentIdAvailable).
```

Model Elaboration



Model Evolution

Model Refinement

Logic-based learning can enable **elaboration** and **evolution** of software models

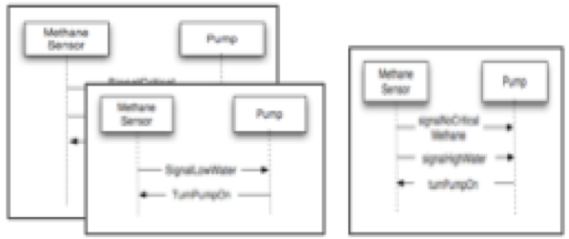
$\Box$(critical Methane $\wedge$pumpOn $\rightarrow$ $\bigcirc$ turnPumpOn)

System goals

Example 6: Software Model Elaboration

Learning task aims at learning software models from

- scenarios provided by the user
- domain knowledge



Undesired behaviours

Desired behaviours

Logic-based Learning

□(pumpOn→pumpOn W turnPumpOff)
□(¬pumpOn→¬pumpOn W turnPumpOn)
□(turnPumpOn→ pumpOn)
□(turnPumpOff→¬pumpOn)
□(criticalMethane ∧ pumpOn → ◯turnPumpOff)
□(criticalMethane → ◯ turnPumpOff)
□(¬highWater → ◯ ¬turnPumpOn)

Elaborated software model

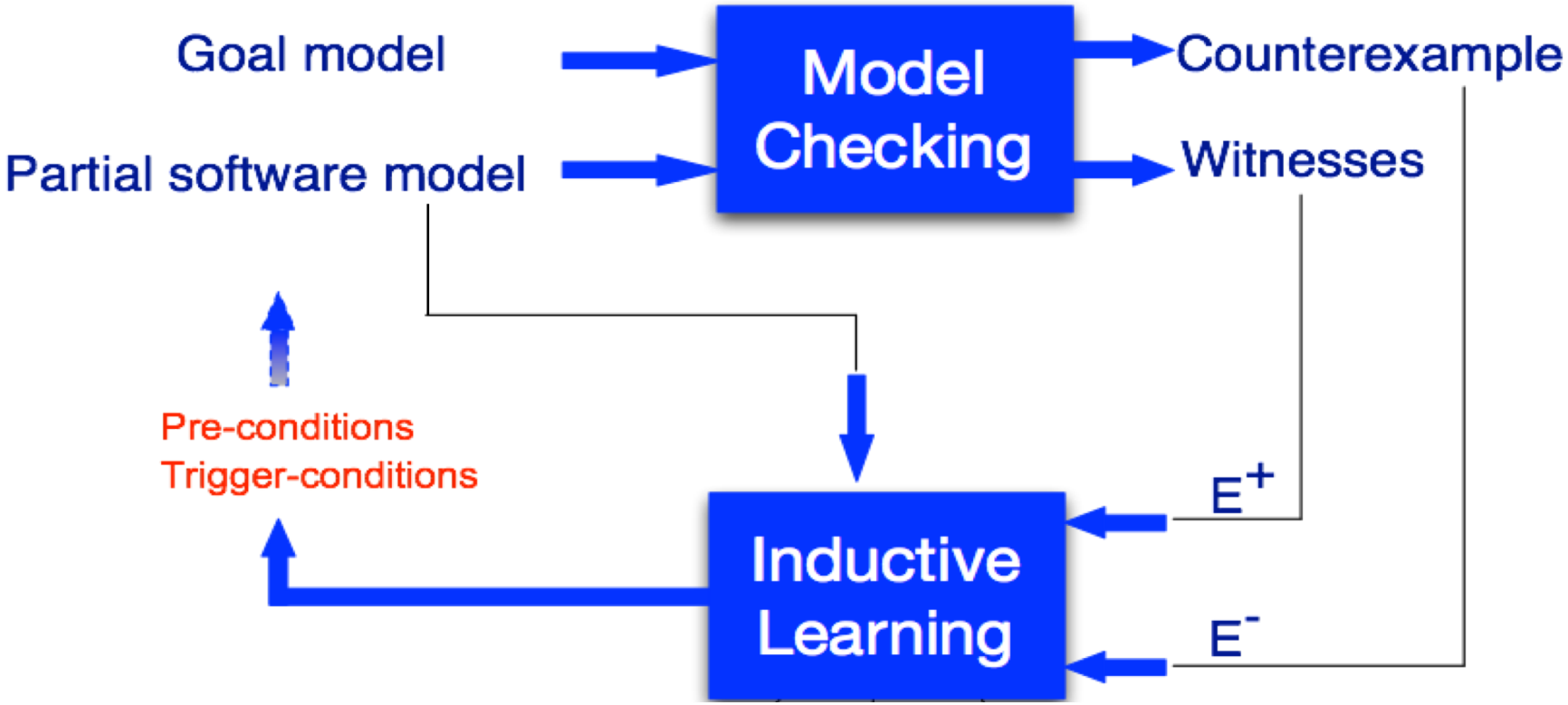
□(pumpOn→pumpOn W turnPumpOff)
□(¬pumpOn→¬pumpOn W turnPumpOn)
□(turnPumpOn→ pumpOn)
□(turnPumpOff→¬pumpOn)
□(criticalMethane ∧ pumpOn → ◯turnPumpOff)
.....

Partial software model

Example 6: Software Model Refinement

Learning task aims at refining and/or revising software models from

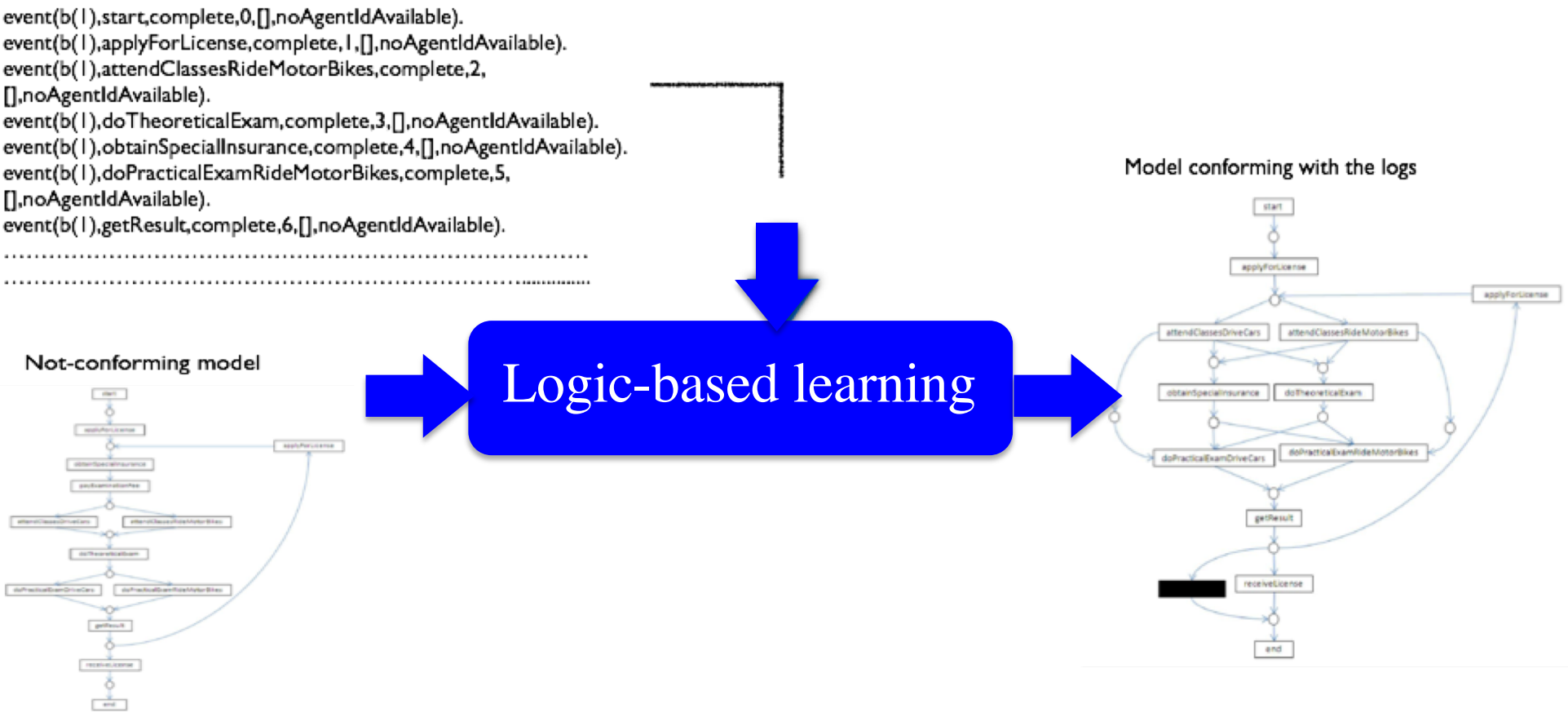
- counterexamples of system properties
- system goals



Example 6: Reverse Engineering Models

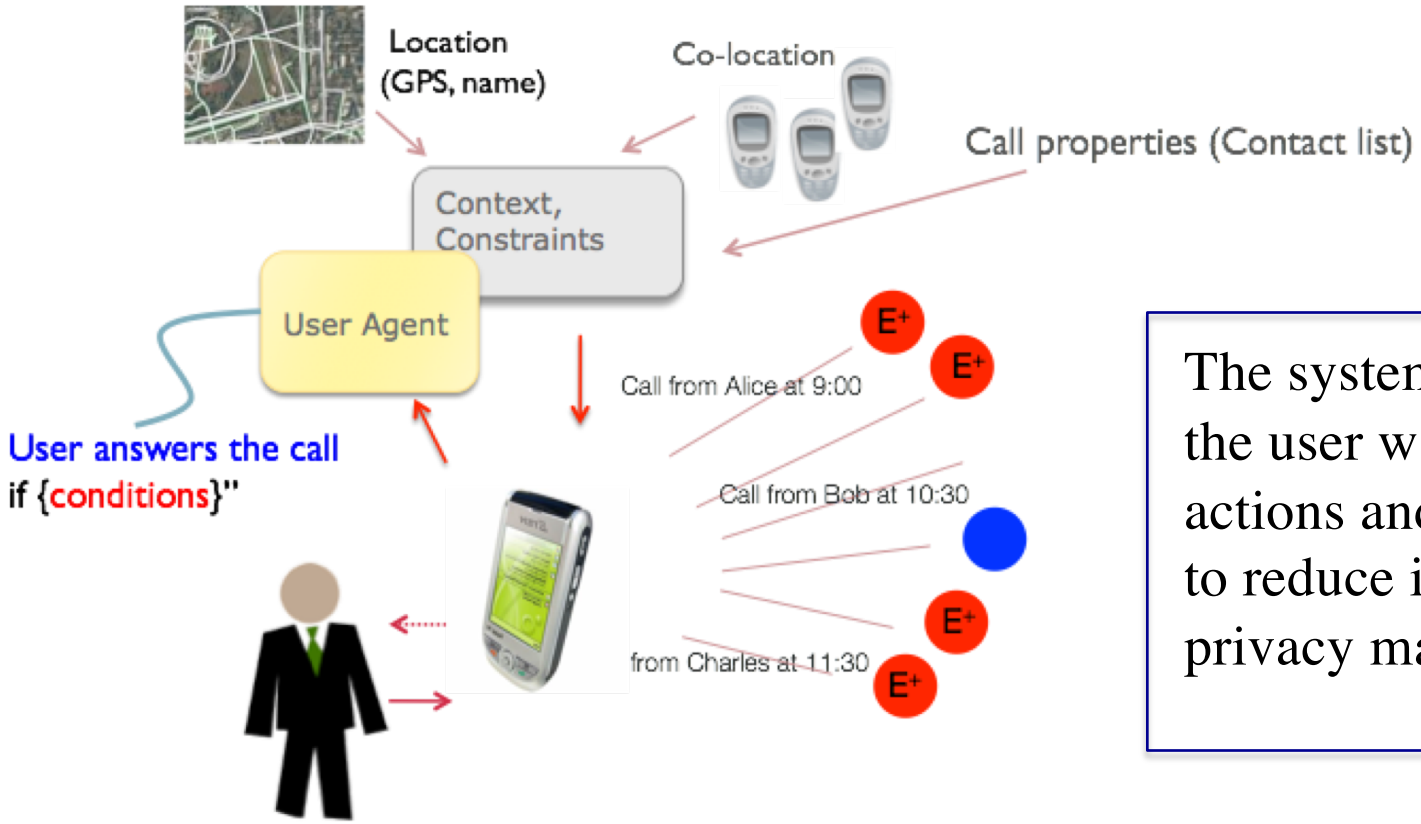
Learning task aims at reverse engineering software models from execution traces

- systems logs are positive examples
- generating negative examples from execution traces and Close World Assumption (CWA)



Example 7: Learning User Behaviours

- Improve a system automatically with experience
- Reverse engineering from execution traces
- Indirect management of policies (learning instead of specifying rules)



The system learns what the user wants from past actions and uses model to reduce intervention in privacy management.

Example 7: Learning User Behaviours

Find a set of rules of the type

accept(…) IF condition$_{1,1}$, …, condition$_{max_1,1}$

accept(…) IF condition$_{1,2}$, …, condition$_{max_2,2}$

……..

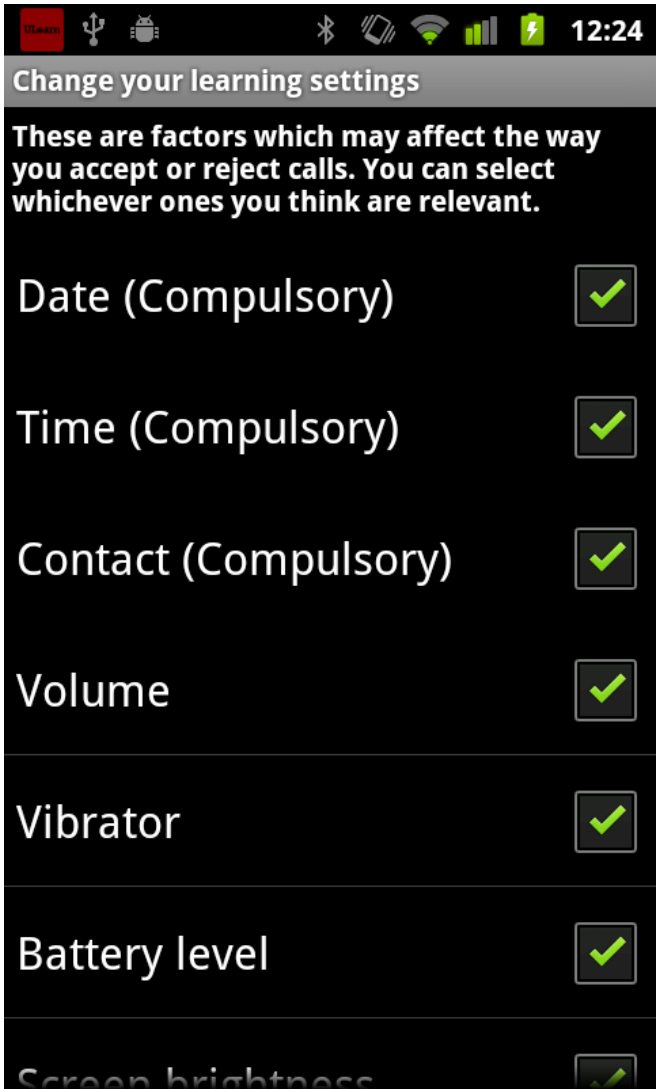
How many?

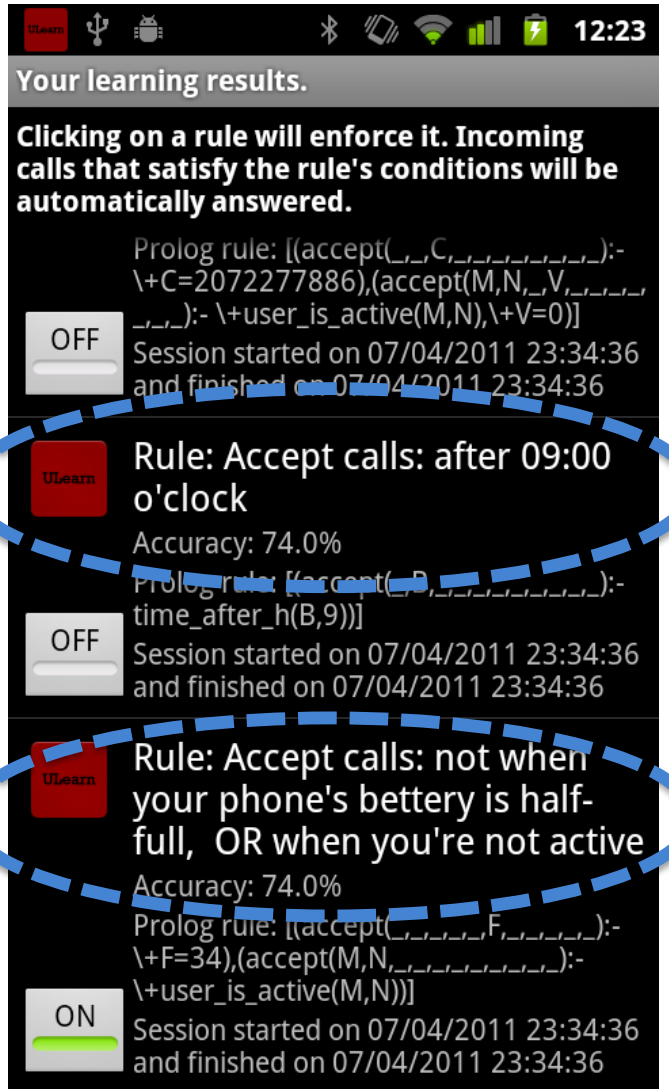
Battery_level ~ 10^2
Contacts ~ 10^1
Devices ~ 10^3
Cells ~ 10^2
Date*Time ~ 10^3
Other options ~ 10

- modeh(accept(+date, +time, +contact, +volume, +vibrator, +battery_level, +screen_brightness, +headset, +screen_status, +light_level, +battery_charging)). 1
- modeb(=(+contact, #contact), [no_ground_constants, name(c)]). 200
- modeb(=(+volume, #volume), [no_ground_constants, name(vol)]). 20
- modeb(=(+vibrator, #vibrator), [no_ground_constants, name(vib)]). 20
- modeb(=(+battery_level, #battery_level), [no_ground_constants, name(bl)]). 200
- modeb(=(+screen_brightness, #screen_brightness), [no_ground_constants, name(scb)]). 20
- modeb(=(+headset, #headset), [no_ground_constants, name(hs)]). 20
- modeb(=(+screen_status, #screen_status), [no_ground_constants, name(ss)]). 20
- modeb(=(+light_level, #light_level), [no_ground_constants, name(ll)]).
- modeb(=(+battery_charging, #battery_charging), [no_ground_constants, name(bc)]). 200
- modeb(weekday(+date)). 2(Positive, Negative)
- modeb(weekend(+date)). 2
- modeb(evening(+time)). 2
- modeb(morning(+time)). 2
- modeb(afternoon(+time)). 2
- modeb(in_call(+date, +time)). 2
- modeb(at(+date, +time, #cell)). 200
- modeb(nearDevice(+date, +time, #device)). 2000
- modeb(neighbourhood(+cell, #cell)). 200
- modeb(user_been_in(+date, +time, +cell)). 2
- modeb(user_is_active(+date, +time)). 2
- modeb(phone_charging(+date, +time)). 2
- modeb(phone_on(+date, +time)). 2
- modeb(user_is_using_app(+date, +time, #app)). 20
- modeb(time_before_h(+time, #hour), [no_ground_constants,name(before)]). 100
- modeb(time_after_h(+time, #hour), [no_ground_constants,name(after)]). 100

Around 5000 choices for condition

Example 7: My mobile knows me!





In summary

- Logic-based learning implies *inductive inference*
  - Generalise specific facts into general principles
- Aims at developing general purpose learning systems
  - Reasoning about structured data
  - Capable of learning complex, structured and readable hypothesis
  - Able to make use of existing relevant background knowledge.
- Needs positive and negative examples
  - Positive examples only, could cause over generalisation
  - Negative examples generated from problem domain or close word assumption
- Can be used for classification and knowledge acquisition.

A Brief History

Finds its origin in philosophy of science

Use of predicate logic for studying machine learning problem	1960
Plotkin's generalisation and specialisation properties	1970
Computational models of scientific discovery	1978
INDUCE system (Michalski) – induction as inverse of deduction	1980
Shapiro's model inference system for program synthesis	1983
Inductive Logic Programming (Muggleton)	1991
e.g FOIL, GOLEM, PROGOL — more focus on ML and Data Mining	
Learning in probabilistic logic setting	1993
Hybrid abductive and inductive learning	2004
Non-monotonic inductive logic programming	2010
Meta-level learning	2011
Inductive Learning of Answer Set Programs	2014

Reading Material

- *Prolog Programming for Artificial Intelligence*, Ivan Bratko Pearson 2012.
- *Logical and Relational Learning*, Luc de Raedt, Springer 2008.
- *Answer Set Solving in Practice* http://www.cs.uni-potsdam.de/~torsten/Potassco/Slides/asp.pdf
- *Knowledge Representation, Reasoning, and the Design of Intelligent Agents* – Michael Gelfond & Yulia Gelfond Kahl Januray 2014
- Collection of research papers and PhD thesis

Slides and notes, complemented with information given during lectures and tutorials.

Course Structure

What is logic-based learning and why it is important

- Deductive, Abductive and Inductive Reasoning
- Monotonic Inductive Logic Programming
- Learning from entailment
  - » Bottom-up learning and incompleteness of PROGOL5
  - » Hybrid abductive and inductive learning (HAIL)
  - » Meta-level learning (TAL)
- Non-monotonic Inductive Logic Programming
  - Answer Set Programming and Stable Model Semantics
    - » Abductive learning ASPAL
    - » Meta-level learning ILASP
  - Context and Preference Learning
  - Learning from Noisy Examples
- Probabilistic Logic Programming and Probabilistic Rule Learning