UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

3547 Intelligent Agents & Reinforcement Learning

Module 4: Planning and Knowledge Representation

Course Plan

Module Titles
Module 1 – Introduction to Intelligent Agents
Module 2 – Current Focus: Search
Module 3 – Logical Inference
Module 4 – **Planning and Knowledge Representation**
Module 5 – Probabilistic Reasoning
Module 6 – Intro to Reinforcement Learning and Finite Markov Decision Processes
Module 7 – Dynamic Programming and Monte Carlo Methods
Module 8 – Temporal Difference Learning
Module 9 – Function Approximation for RL
Module 10 – Deep Reinforcement Learning and Policy Gradient Methods
Module 11 – Introduction to Advanced DRL
Module 12 – Presentations (no content)

Learning Outcomes for this Module

- Introduce you to expert systems
- Extend your skills working with Propositional Logic to First Order Logic (FOL)
- Introduce Prolog
- Enable you to model the passage of time in logic
- Use logic programming to plan sequences of action for an agent
- Show you how to model knowledge using ontologies and relationships

Topics for this Module

- **4.1** Expert Systems and First Order Logic
- **4.2** Introduction to Prolog
- **4.3** Modeling the Flow of Time
- **4.4** Planning
- **4.5** Knowledge Representation
- **4.6** Resources and Wrap-up

Module 4 – Section 1

Expert Systems and First Order Logic

Expert Systems

- Need to be able to deal with uncertainty and be transparent
- Components
  - Problem Solving: DB + inference engine
  - User Interaction: UI + query capability
- If-then rules are a natural choice
- MYCIN
- AL3
- Knowledge acquisition is the bottleneck

Terminology for First Order Logic

- **Constant**: A symbol that stands for an object
- **Variable**: A symbol representing a variable term (and in FOL can serve as an argument to a function)
- **Predicate**: A symbol which stands for a relation of form $p(t_1 \ldots t_n)$ where p is a predicate symbol and each t_i is a term (called an **argument** to the predicate)
- **Function**: A symbol that stands for a function of form $\mathrm{f}(t_1 \ldots t_n)$ and represents an object
- **Term**: A constant, a variable or a function; a logical expression that refers to an object
- **Atom** (or **atomic symbol**): a formula of form p
- **Expression**: Either a term, an atom, a definite clause, or a query
- **Ground**: An expression is ground if doesn't contain any variables
- **Interpretation**: Specifies which objects, relations, and functions are referred to by the constant, predicate and function symbols
- The definitions of definite clause, knowledge base, rule and query remain the same as for propositional calculus

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 4 – Section 2

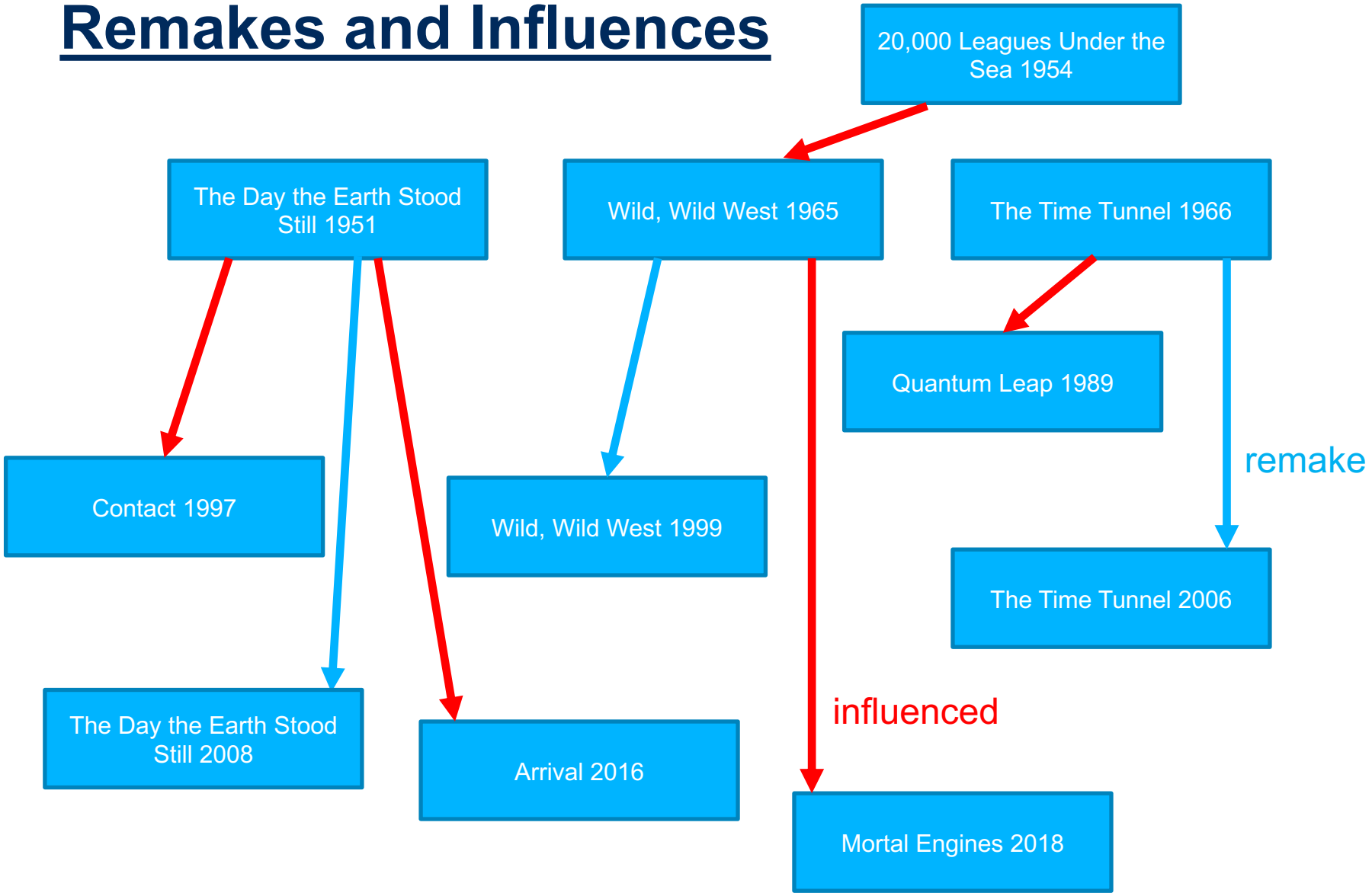
Introduction to Prolog

Prolog Hands-On

- https://swish.swi-prolog.org/

```
a :- b, c.
b :- d, e.
c :- e.
d.
e.
f :- a, g.
```

Remakes and Influences



All opinions on influences are the slide author's

Prolog Relations

- https://swish.swi-prolog.org/

```
remade(day_earth_51, day_earth_08).
remade(wild_west_65, wild_west_99).
remade(time_tunnel_66, time_tunnel_06).
influenced(leagues_54, wild_west_65).
influenced(day_earth_51, arrival_16).
influenced(wild_west_65, mortal_engines_18).
influenced(time_tunnel_66, quantum_leap_89).
influenced(day_earth_51, contact_97).
```

Prolog Queries

?-remade(wild_west_65, wild_west_99).

?-remade(wild_west_65, quantum_leap_89).

?-directed(wild_west_65, quantum_leap_89).

?-influenced(leagues_54, arrival_16).

?-influenced(X, mortal_engines_18).

?-influenced(X, mortal_engines_18), influenced(A, X).

Now: Formulate a query to answer:

- Which movie influenced both Arrival and Contact?

Facts and Rules

excellent(arrival_16).

- How would you express, (using :-)

  For all X and Y,

  X is excellent if

  X influenced Y, and Y was excellent.

- How would you ask what movies were also influenced by the movies that influenced Arrival?

Recursive Rules

led_to(X,Y):-influenced(X,Y).

led_to(X,Y):-influenced(X,A), influenced(A,Y).

led_to(X,Y):-influenced(X,A), influenced(A,B), influenced(B,Y).

led_to(X,Y):-influenced(X,A), influenced(A,B), influenced(B,C), influenced(C,Y).

…

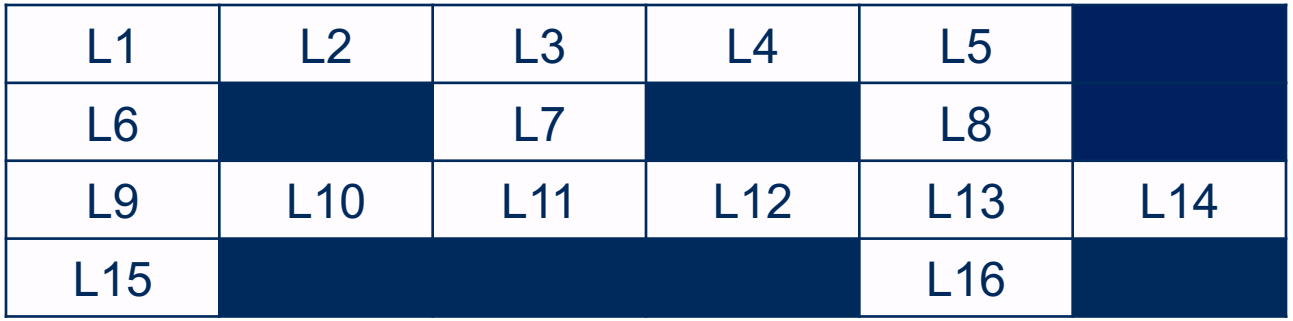
Define with two rules:

led_to(X,Y):-influenced(X,Y).

led_to(X,Y):-influenced(X,A),led_to(A,Y).

Try This:

```
word(d,o,g).
word(r,u,n).
word(t,o,p).
word(f,i,v,e).
word(f,o,u,r).
word(l,o,s,t).
word(m,e,s,s).
word(u,n,i,t).
word(b,a,k,e,r).
word(f,o,r,u,m).
word(g,r,e,e,n).
word(s,u,p,e,r).
word(p,r,o,l,o,g).
word(v,a,n,i,s,h).
word(w,o,n,d,e,r).
word(y,e,l,l,o,w).
solution(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16) :-
    word(L1,L2,L3,L4,L5),
    word(L9,L10,L11,L12,L13,L14),
    word(L1,L6,L9,L15),
    word(L3,L7,L11),
    word(L5,L8,L13,L16).
?- solution(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16).
```



Source: Bratko

Declarative and Procedural Meaning

- Consider X :- A, B where X, A, B are terms
- We can read it declaratively:
  - if A is true and B is true then X is true
  - A and B imply X
- Or procedurally:
  - To solve problem X, *first* solve subproblem A *then* subproblem B
  - To satisfy X, *first* satisfy A *then* B

Disjunction in Prolog

- Prolog also supports disjunction (or):
  - Use semicolon rather than colon
  - e.g. X :- A; B.
  - Has the same meaning as:

    X :- A.

    X :- B.
- Comma binds stronger than semicolon
- e.g. X :- M, N; A, B, C. means X :- (M,N);(A,B,C).

How Prolog Scans

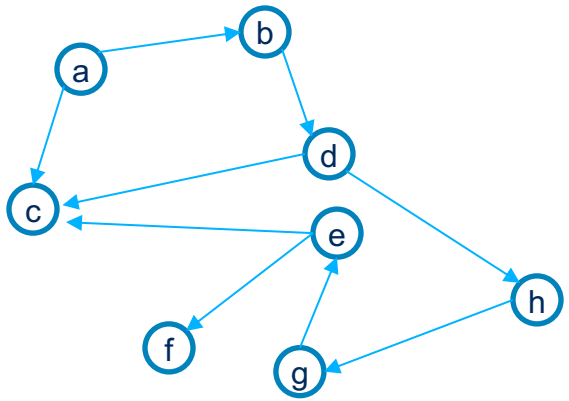
```
movie(oblivion, cruise).
movie(vanilla_sky, cruise).
movie(edge_of_tomorrow,cruise).
movie(paycheck, affleck).
movie(dumb_and_dumber, carrey).
movie(shakespeare_in_love, affleck).
nationality(affleck, american).
nationality(cruise, american).
nationality(carrey, canadian).

movie(X,Y),nationality(Y,canadian).
```

Other Prolog Features

- Lists
  - [item_one, item_two, item_three]
  - Constructor is **.** e.g. **.**(item_one, (time_two, (item_three, [ ])))
  - Head and tail operator is | e.g. L = [a|Tail]
  - Concatenate with conc(a, b, c).
- Arithmetic
  - Use *is* to evaluate e.g. X is 1 + 2
  - Use =:= to compare numbers by evaluating them

Graph Search in Prolog




```
link(a,b).
link(a,c).
link(b,d).
link(d,c).
link(d,h).
link(h,g).
link(g,e).
link(e,c).
link(e,f).
```

```
path(Node, Node).
path(StartNode, EndNode) :- link(StartNode, NextNode), path(NextNode, EndNode).
```

WumpusWorld in Prolog

https://www.metalevel.at/wumpusworld/wumpus.pl

https://github.com/rlegendi/wumpus-prolog/blob/master/wumpus.pl

Module 4 – Section 3

Modeling the Flow of Time

Agent Percepts Over Time

- Now we have a problem: which percepts' occurrence are dependent on time? (why not just index by the location of the agent when the percept occurred?)
- So we need a way of indexing percepts by time
- We have the same need for the agent's state (location, pose, whether it still has the arrow) and the status of the gold and Wumpus
- We call an aspect of the world that changes over time a **fluent**
- It would be convenient to create a **transition model** to capture how fluents change between timesteps as a result of actions taken by the agent

Percepts and the Transition Model

$$agentLocation_{x,y}^{t} \Rightarrow (breeze^{t} \Leftrightarrow breezePerceptible_{x,y})$$
$$agentLocation_{x,y}^{t} \Rightarrow (stench^{t} \Leftrightarrow stenchPerceptible_{x,y})$$

Transition Model:

$$agentLocation_{1,1}^{0} \wedge facingEast^{0} \wedge forward^{0}$$
$$\Rightarrow \left(agentLocation_{2,1}^{1} \wedge \neg agentLocation_{1,1}^{1}\right).$$

…

Question: How many of these **effects axioms** would we need?

Successor-State Axioms

Better to describe each fluent in terms of how it evolves over time:

$$f^{t+1} \Leftrightarrow actionCausesF^t \vee (f^t \wedge \neg actionCausesNotF^t)$$

e.g.

$$haveArrow^{t+1} \Leftrightarrow (haveArrow^t \wedge \neg shoot^t)$$

$$agentLocation_{1,1}^{t+1}$$

$$\Leftrightarrow \left(agentLocation_{1,1}^t \wedge (\neg forward^t \vee bump^{t+1})\right)$$

$$\vee \left(agentLocation_{1,2}^t \wedge (south^t \wedge forward^t)\right)$$

$$\vee \left(agentLocation_{2,1}^t \wedge (west^t \wedge forward^t)\right).$$

Module 4 – Section 4

Planning

Planning

- So far, our agent has used domain-independent heuristics that would allow it to generate a plan but it would need to handle a potentially huge space of states and actions
- For example, for WumpusWorld:

  4 agent orientations x max # of time steps x 16 possible locations
- We can deal better with this space if we factor it
- Our approach: Represent each state as a conjunction of ground, functionless fluents that describe its features

Assumptions for the Planning Method to Follow

- Single, deterministic agent that predicts the consequences of its actions
- No events outside its control
- The environment is fully observable
- Time progresses discretely from one state to the next
- Goals are predicates of states to be achieved

Representation Using Planning Languages

- Current State
- Action
  - Precondition
  - Effect
- Resulting State

$goal(\neg firePresent)$

$action(activate)$,

$PRECOND: at(station) \wedge \neg batteryLow \wedge firePresent$

$EFFECT: \neg at(station) \wedge at(enroute)$

Forward Planning

- Create a graph of the relevant space
- Use search such as A*
- Optimal planning is PSPACE which is worse than NP but often finding a non-optimal solution can be P if we have good heuristics
- Common heuristics:
  - Ignore preconditions
  - Ignore delete lists
  - State abstraction
  - Decomposition

Planning With Multiple Agents

- Multi-agent vs. multi-body
- Coordination and communication
- Hierarchical control
- Team play
  - Cohesion
  - Separation
  - Alignment

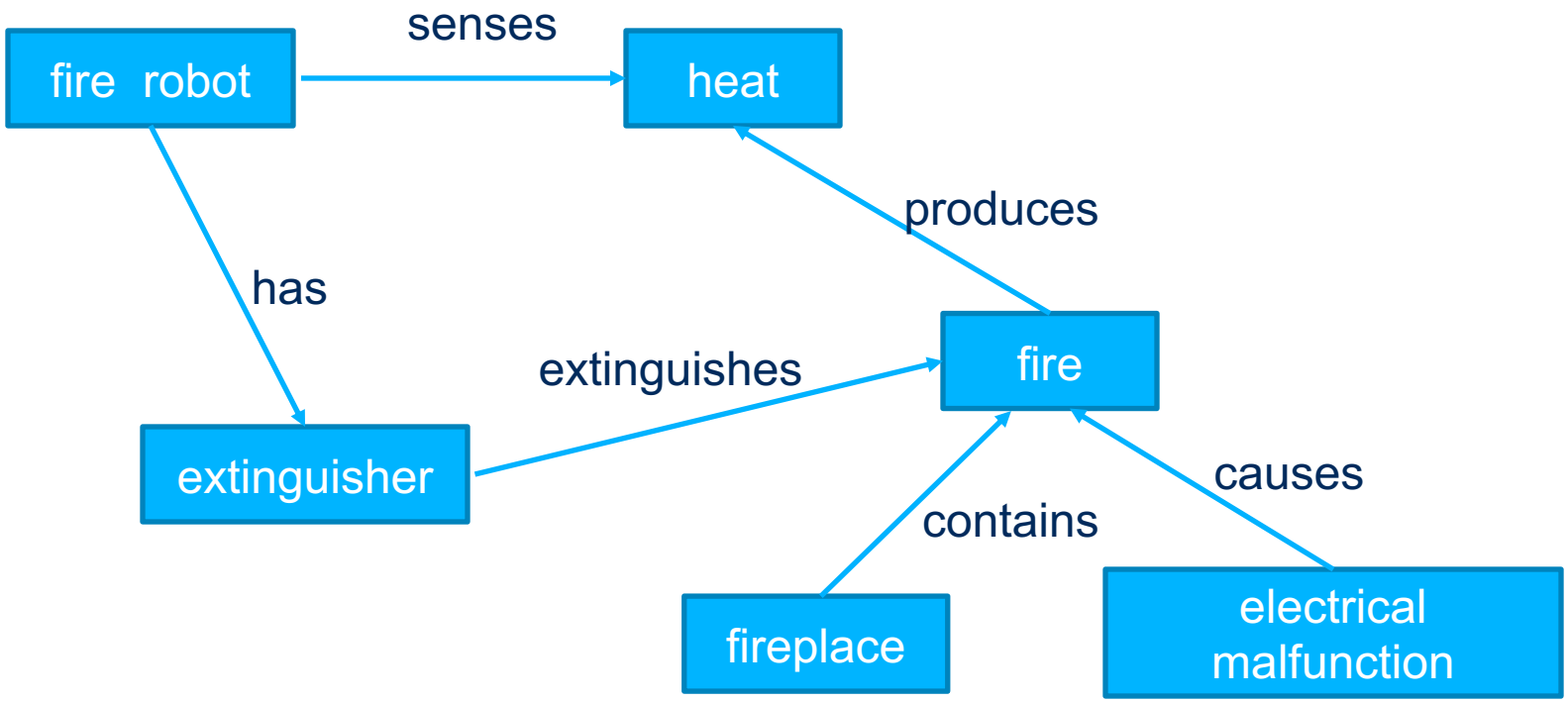
Module 4 – Section 5

Knowledge Representation

Ontologies

- An ontology is a specification of the meanings of the symbols in a knowledge base or information system
- We want to model
  - Things
  - Relationships
  - Properties of things
- A popular form is to model relationships as a **triple**: $prop(Individual, Property, Value)$
- Triples correspond to $subject\ verb\ object$ form and can also be written $prop(subject, verb, object)$ or $verb(subject, object)$
- Triples can be interpreted as forming a directed graph called a **semantic network** or **knowledge graph**

An Example Semantic Network

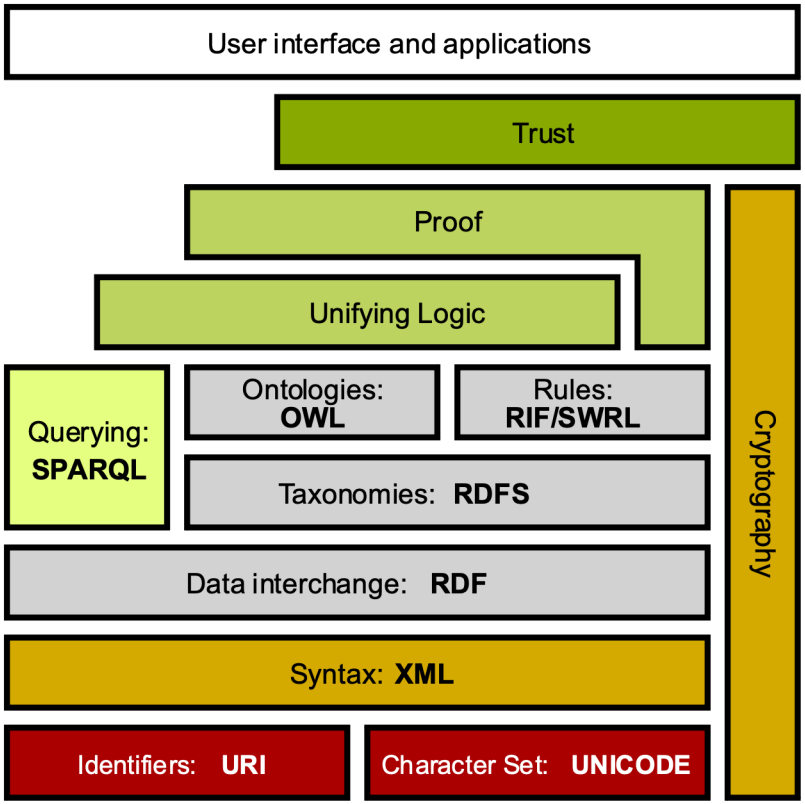


Classes and Part-of

- It is a natural extension to recognize groups of things that are similar
- We can introduce a distinguished Property **is_type_of**
- Similarly, we may distinguish **is_part_of**

Semantic Web

- Berners-Lee's vision of the WWW is a world-wide semantic network
- Based on triples and semantic markup



UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 4 – Section 6

Resources and Wrap-up

Resources

- Intro to Prolog: https://metalevel.at/prolog
- Bratko, Ivan. Prolog Programming for Artificial Intelligence. 4th Ed. 2012.
- SWI-Prolog: https://www.swi-prolog.org/
- Gnu Prolog: http://www.gprolog.org/
- Lambda Prolog: http://www.lix.polytechnique.fr/Labo/Dale.Miller/lProlog/
- https://en.wikipedia.org/wiki/DPLL_algorithm
- Scryer Prolog: https://github.com/mthom/scryer-prolog
- Datalog vs. Prolog: http://www.cs.tau.ac.il/~msagiv/courses/pl14/prolog.pdf

Summary

- Logic programming *was* “AI” until Deep Learning appeared
- Deep Learning is good at pattern recognition and soft logic but not reasoning*
- Logic programming enables declarative problem-solving
- Planning can be expressed as a search problem and solved using logic programming but is computationally a “Hard” problem
- Knowledge can be represented with very simple constructs (e.g. triples) and given meaning by ontologies

* yet?

Next Week

- Logic programming treats propositions as true/false dichotomies: *black and white thinking…*
- We live in an uncertain world, so: we'll use Bayesian techniques to enable probabilistic inference

Any questions?

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Thank You

Thank you for choosing the University of Toronto School of Continuing Studies