Introduction

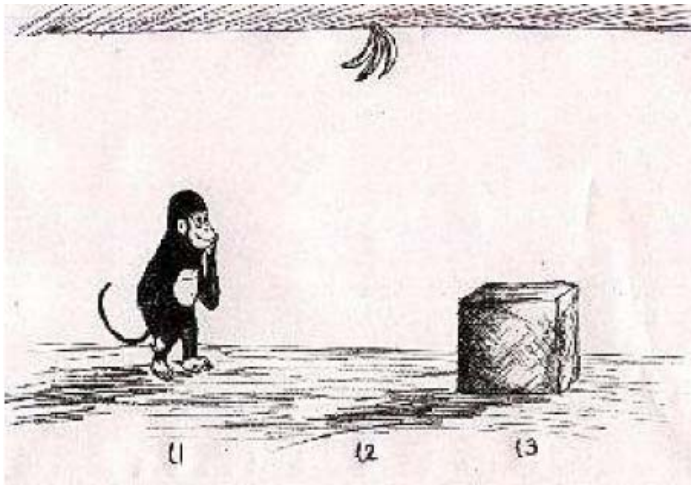
CSE591 KRR

- Knowledge Representation and Reasoning is at the heart of AI
  - To understand the nature of intelligence and cognition so well that computers can be made to exhibit human-like abilities
  - researchers assume that (artificial) intelligence could be formalized as symbolic reasoing with explicit representations of knowledge, and that the core research challenge is to figure out how to represent knowledge in computers, and to use it algorithmically to solve problems

Many KRR formalisms

- General methods
  - SAT, description logics, constraint programming, conceptual graphs, nonmonotonic logics, answer set programming, belief revision, ...
- Specialized methods: time, space, causation, action
  - temporal reasoning, knowledge and belief, action formalims
- Applications
  - query answering, semantic web, planning, cognitive robotics, multiagent systems

Monkey and Bananas



Missionaries and Cannibals

McCarthy's Elaborations

- The boat can carry three.
- Only one missionary and one cannibal can row.
- Three missionaries alone with a cannibal can convert him into a missionary.
- One of the missionaries is Jesus Christ.
- There are four missionaries and four cannibals.

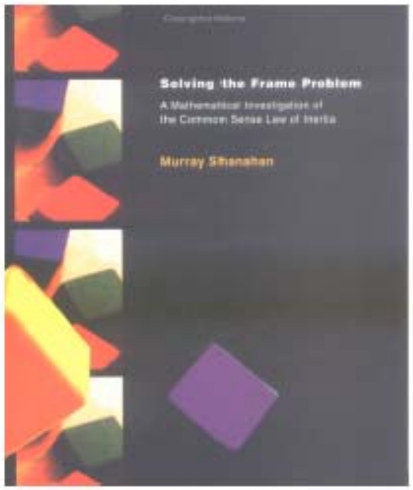
Advice Taker

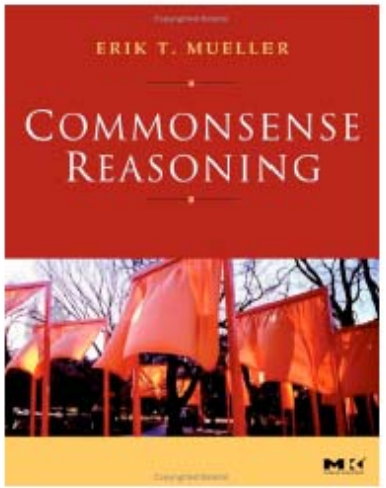
Assume that I am seated at my desk at home and I wish to go to the airport. My car is at my home also. How shall I get to the airport?

Advice Taker: draws a relevant conclusion from the set of formal statements of premises; changes are made by updating the premises, rather than by rewriting the program's internal code [McCarthy, "Programs with common sense," 1959].

The Frame Problem: describing what remains unchanged after executing an action.

Frame Problem





Shanahan, Solving the Frame Problem, 1997, MIT Press.
Mueller, Commonsense Reasoning, 2006, Morgan Kaufmann.

Classical Logic vs. Nonmonotonic Logics

- The needs for defeasible reasoning: conclusions are drawn tentatively and can be retracted in the light of further information
- Classical logic (propositional logic, first-order logic, ...) is monotonic: if $\Gamma \vdash A$ then $\Gamma \cup \Delta \vdash A$
- Nonmonotonic logics: $\Gamma \vdash A$ then $\Gamma \cup \Delta \nvdash A$
  - circumscription [McCarthy, 1980]
  - default logic [Reiger, 1980]
  - nonmonotonic modal logic [McDermott & Doyle, 1980]

Yale Shooting Problem

- Initially the turkey is alive and the gun is unloaded. Then the gun is loaded and after wait shot the turkey. Is the turkey alive?
- A naive attempt to formalize the commonsense law of inertia did not work
- It was NOT the limitation of nonmonotonic logics. Rather it is about how to use them in a proper way. More structured high level action formalims are desirable

More KR Formalisms

-

Situation Calculus [McCarthy & Hayes, 1969] Event Calculus [Kowalski & Sergot, 1986] Temporal Action Logics [Doherty, 1996] ...	Action Languages $\mathcal{A}$ [Gelfond & Lifschitz, 1993] $\mathcal{C}$ [Giunchiglia & Lifschitz, 1998] $\mathcal{C}+$ [Giunchiglia *et al.*, 2004] ...
Circumscription [McCarthy, 1980;1986] Completion [Clark, 1978]	Stable Model Semantics [Gelfond & Lifschitz, 1988] Nonmonotonic Causal Theories [Giunchiglia *et al.*, 2004]
Classical Logic	Nonmonotonic Logics

Yale Shooting in Event Calculus

- T : $Initiates(Load, Loaded, t)$
  $HoldsAt(Loaded, t) \rightarrow Terminates(Shoot, Alive, t)$
  $Terminates(Shoot, Loaded, t)$

  $HoldsAt(Alive, 0)$
  $\neg HoldsAt(Loaded, 0)$
  $Happens(Load, 0)$
  $\neg Happens(e, 1)$
  $Happens(Shoot, 2)$
- Plus adding domain independent axioms and minimize extents of Initiates, Terminates, Happens

Yale Shooting in CCalc

```
:- sorts
  turkey.

:- objects
  turkey1                 :: turkey.

:- variables
  T                       :: turkey.

:- constants
  loaded,
  alive(turkey)           :: inertialFluent;
  target                  :: inertialFluent(turkey+none);
  load,
  aim(turkey),
  shoot                   :: exogenousAction.
```

```
load causes loaded.
load causes target=none.

aim(T) causes target=T.

shoot causes -alive(T) if target=T.
shoot causes -loaded.
nonexecutable shoot if -loaded.

nonexecutable aim(T) & shoot.


:- query
maxstep :: 3;
0: [/\T | alive(T)],
   target=none,
   -loaded;
maxstep:
   [/\T | -alive(T)].
```

Monkey and Bananas in CCalc

```
:- sorts
  thing; location.

:- objects
  monkey,bananas,box      :: thing;
  loc1,loc2,loc3          :: location.

:- constants
  hasBananas,onBox        :: inertialFluent;
  loc(thing)              :: inertialFluent(location);

  walk(location),
  pushBox(location),
  climbOn,climbOff,
  graspBananas            :: exogenousAction.
```

```
:- variables
  L                              :: location.

walk(L) causes loc(monkey)=L.
nonexecutable walk(L) if loc(monkey)=L.
nonexecutable walk(L) if onBox.

pushBox(L) causes loc(box)=L.
pushBox(L) causes loc(monkey)=L.
nonexecutable pushBox(L) if loc(monkey)=L.
nonexecutable pushBox(L) if onBox.
nonexecutable pushBox(L) if loc(monkey)\=loc(box).
```

```
% Calling mChaff spelt3...  done.
% Reading output file(s) from SAT solver...  done.
% Solution time:  0.01 seconds

0: loc(monkey)=loc1  loc(bananas)=loc2  loc(box)=loc3

ACTIONS: walk(loc3)

1: loc(monkey)=loc3  loc(bananas)=loc2  loc(box)=loc3

ACTIONS: pushBox(loc2)

2: loc(monkey)=loc2  loc(bananas)=loc2  loc(box)=loc2

ACTIONS: climbOn

3: onBox  loc(monkey)=loc2  loc(bananas)=loc2  loc(box)=loc2

ACTIONS: graspBananas

4: hasBananas  onBox  loc(monkey)=loc2  loc(bananas)=loc2
   loc(box)=loc2
```

Missionaries and Cannibals in CCalc

```
% File 'basic'

cross(V) causes loc(V)=L
  if destination(V)=L unless abCross(V).

cross(V) increments num(G,L) by N
  if destination(V)=L & howmany(V,G)=N unless abCross(V).

constraint -outnumbered(num(mi,L),num(ca,L))
  unless ab1(L).

nonexecutable cross(V)
  if outnumbered(howmany(V,mi),howmany(V,ca))
  unless ab4(V).

constraint capacity(boat)=2 unless ab5.
```

"The boat can carry three. Four [pairs] can cross but not five."

```
% File 'basic'
...
constraint capacity(boat)=2 unless ab5.
```

```
% File 'jmc4'
:- include 'basic'.

caused ab5.
constraint capacity(boat)=3 unless ab6.
```

Correction to McCarthy

4. The boat can carry three. Four can cross but not five. If the boat can carry four an arbitrary number can cross. [2003 Sept: This is mistaken. Joohyung Lee showed that if the boat holds three, five can cross.]

[John McCarthy, "Elaboration Tolerance", 1998; updated in 2003]

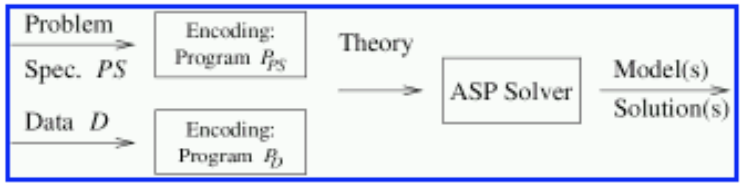
Applications

- Query answering
- Ontology reasoning: semantic Web, medical ontology
- Cognitive robotics

Answer Set Programming

A new form of declarative programming oriented towards combinatorial search problems and knowledge-intensive applications.

In answer set programming we obtain the answers by declaring the properties of the answers, using logic rules. It differs from procedural languages like C/C++ and Java where we specify the sequence of operations to be executed to generate an answer.



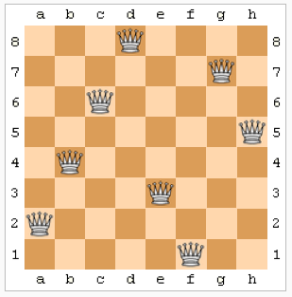
The idea of ASP is to represent a given search problem as the problem of finding an answer set for some logic program, and then find a solution using an answer set solver.

ASP Applications

http://www.kr.tuwien.ac.at/research/projects/WASP/showcase.html

- configuration
- information integration
- security analysis
- agent systems
- semantic web
- planning
- auctions
- policy description
- linguistics
- bioinformatics
- ...

Example: 8 Queens Problem



"Each row has exactly one queen"

$$1 \leq \{q_{i,1}, \ldots, q_{i,8}\} \leq 1 \quad (1 \leq i \leq 8).$$

"Two queens cannot stay on the same column"

$$\bot \leftarrow q_{i,j}, q_{i',j} \quad (1 \leq i < i' \leq 8; 1 \leq j \leq 8).$$

"Two queens cannot stay on the same diagonal"

$$\bot \leftarrow q_{i,j}, q_{i',j'} \quad (1 \leq i < i' \leq 8; 1 \leq j, j' \leq 8; i' - i = |j' - j|).$$

Papers on ASP

- *Answer Set Programming and Plan Generation.*
- *A Logic Programming Approach to Knowledge-state Planning.*
- *Reconstructing the Evolutionary History of Indo-European Languages Using Answer Set Programming.*
- *Character-based Cladistics and Answer Set Programming.*
- *Rectilinear Steiner Tree Construction Using Answer Set Programming.*
- *An A-Prolog Decision Support System for the Space Shuttle* (RCS/USA-Advisor : decision support system for shuttle controllers).
- *Developing a Declarative Rule Language for Applications in Product Configuration* (variantum : spin-off).
- *Bounded LTL Model Checking with Stable Models.*

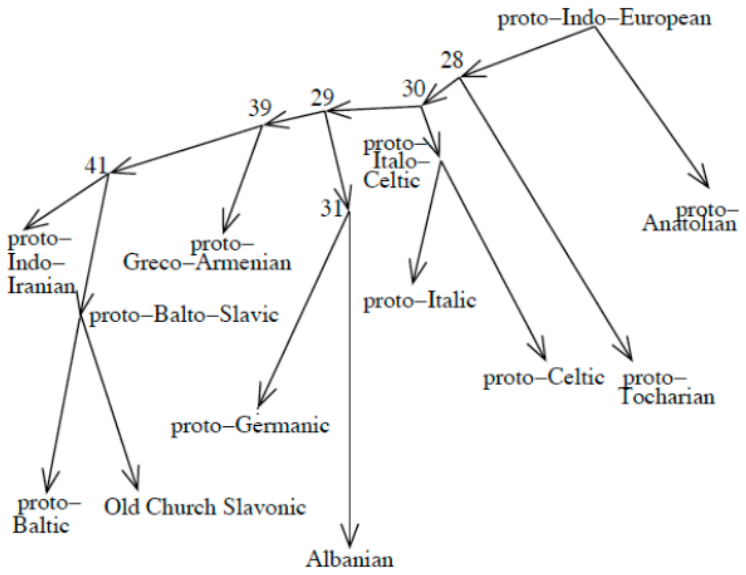
- *Using Logic Programs with Stable Model Semantics to Solve Deadlock and Reachability Problems for 1-Safe Petri Nets.*
- *Data Integration: A Challenging ASP Application.*
- *Efficient Haplotype Inference with Answer Set Programming*
- *Automatic Composition of Melodic and Harmonic Music by Answer Set Programming* — Listen!
- *A Logic Programming Approach to Home Monitoring for Risk Prevention in Assisted Living*

```
%% At every time step the note must change
%% It changes by stepping (moving one note in the scale) or
%% leaping (moving more than one note)
%% These can either be upwards or downwards
1 { stepAt(P,T), leapAt(P,T) } 1 :- T != t.
1 { downAt(P,T),   upAt(P,T) } 1 :- T != t.

stepDown(P,T) :- stepAt(P,T), downAt(P,T).
stepUp(P,T)   :- stepAt(P,T),   upAt(P,T).

leapDown(P,T) :- leapAt(P,T), downAt(P,T).
leapUp(P,T)   :- leapAt(P,T),   upAt(P,T).
```

Inferring Phylogenetic Trees

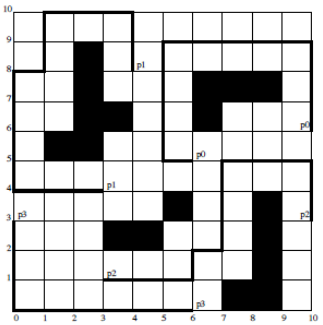
[Erdem, Lifschitz, Nakhleh, Ringe, "Reconstructing the Evolutionary History of Indo-European Languages Using Answer Set Programming", 2003]

	'one'	'arm'	'beard'	'free'	'pour'	'tear'
proto-Indo-Iranian		5	1			11
proto-Baltic	11	8	5		6	11
Old Church Slavonic			5		6	
proto-Greco-Armenian	2		1	3	3	2
proto-Germanic	11	8	5	10	14	2
Albanian	2		1			
proto-Italic	11		5	3	14	2
proto-Celtic	11			10		2
proto-Tocharian	2	5			3	11
proto-Anatolian			1			



Wire Routing

Wire routing is the problem of determining the physical locations of all wires interconnecting the circuit components on a chip.



Challenges

- Expressiveness
- Computability