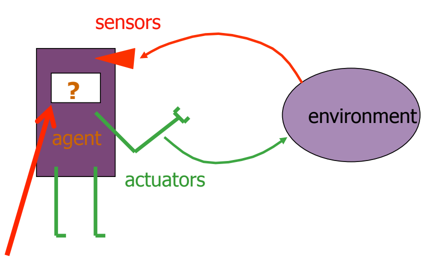
Knowledge Representation

CS311 Spring 2013
David Kauchak

Admin

- Video: http://www.cs.cmu.edu/~tom7/mario/
- Written problems 3 to be looked at by next Monday
- Status report 1 due on Friday
- Exam #2 next week
  - Take home
  - Review next Tuesday

Agent's knowledge representation



What have we seen so far for knowledge representation?

Agent's knowledge representation

procedural

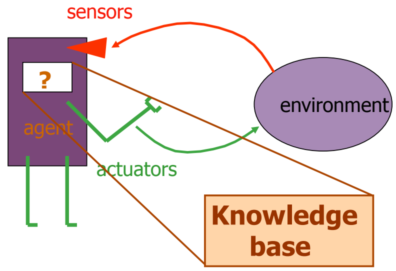
- methods that encode how to handle specific situations
  - chooseMoveMancala()
  - driveOnHighway()

model-based

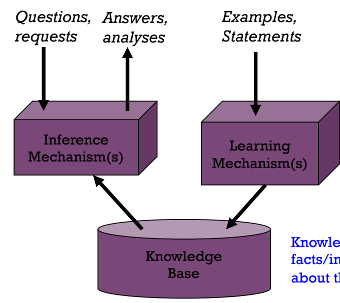
- bayesian network
- neural network
- decision tree

Is this how people do it?

+ Knowledge-based agent



+ Knowledge-based approach



Knowledge base stores facts/information/rules about the world

+ What is in a knowledge base?

Facts…

Specific:

- `Middlebury College is a private college`
- `Prof. Kauchak teaches at Middlebury College`
- `2+2 = 4`
- `The answer to the ultimate question of life is 42`

General:

- `All triangles have three sides`
- `All tomatoes are red`
- $n^2 = n * n$

+ Inference

Given facts, we'd like to ask questions

- Key: depending on how we store the facts, this can be easy or hard
- People do this naturally (though not perfectly)
  - For computers, we need specific rules

For example:

- `Johnny likes to program in C`
- `C is a hard programming language`
- `Computer scientists like to program in hard languages`

What can we infer?

Inference

For example:
- Johnny likes to program in C
- C is a hard programming language
- Computer scientists like to program in hard languages

Be careful!

we cannot infer that Johnny is a computer scientist

What about now:
- `All people who like to program in hard languages are computer scientists`

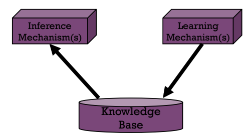
What can we infer?

Creating a knowledge-based agent

Representation: how are we going to store our facts?

Inference: How can we infer information from our facts? How can we ask questions?

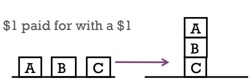
Learning: How will we populate our facts?



Your turn

Knowledge engineer
- representation: how are you storing facts?
- inference: how can you algorithmically query these facts?
- learning: you provide the facts ☺

Some problems to think about:
- Give change for some purchase < $1 paid for with a $1
- Block stacking problems
- Wumpus world
- How to make an omelette?
- How early should I leave for my flight?
- General reasoning agent (e.g. you)?



Things to think about:
- any approaches that you've seen previously useful?
- what are the challenges?
- what things are hard to represent?

Propositional logic

Statements are constructed from propositions

A proposition can be either true or false

Statements are made into larger statements using connectives

Example
- `JohnnyLikesC = true`
- `CisHard = true`
- `CisHard ∧ JohnnyLikesC => JohnnyIsCS`

Propositional logic

Negation: not, ¬, ~

Conjunction: and, ∧

Disjunction: or, ∨

Implication: implies, =>

Biconditional: iff, <=>

Propositional logic

A	B	A⇒B
F	F	
F	T	
T	F	
T	T	

A	B	A⇔B
F	F	
F	T	
T	F	
T	T	

Propositional logic

A	B	A⇒B
F	F	T
F	T	T
T	F	F
T	T	T

A=>B ≡ ¬A∨B

A	B	A⇔B
F	F	T
F	T	F
T	F	F
T	T	T

A<=>B ≡ (A=>B)∧(B=>A)

Inference with propositional logic

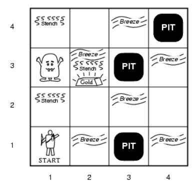
There are many rules that enable new propositions to be derived from existing propositions

- Modus Ponens: P => Q, P derive Q
- deMorgan's law: ¬(A∧B), derive ¬A∨¬B

View it as a search problem:

- **starting state**: current facts/KB
- **actions**: all ways of deriving new propositions from the current KB
- **result**: add the new proposition to the KB/state
- **goal**: when the KB/state contains the proposition we want

+ Propositional logic for Wumpus

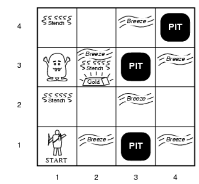


How can we model Wumpus world using propositional logic? Is propositional logic a good choice?

+ Propositional logic for Wumpus

Variable for each condition for each square

- $breeze_{1,1}$ = false, $breeze_{1,2}$ = true, …
- $breeze_{1,1}$ => $pit_{1,2}$ or $pit_{2,1}$, …



Have to enumerate all the states! Can't say if a square has a breeze then there is a pit next door

+ First order logic (aka predicate calculus)

Uses objects (entities) and relations/functions

Fixes two key problems with propositional logic

- Adds relations/functions
  - `Likes(John, C)`
  - `isA(Obama, person)`
  - `isA(Obama, USPresident)`
  - `programsIn(John, C)`
- This is much cleaner than:
  - `JohnLikeC`
  - `MaryLikesC`
  - `JohnLikesMary`
  - …

+ First order logic (aka predicate calculus)

Quantifiers

- "for all": written as an upside down 'A' - ∀
- "there exists": written as a backwards 'E' - ∃

For example:

- `Johnny likes to program in C`
- `C is a hard programming language`
- `All people who like to program in hard languages are computer scientists`

$$likes(Johnny, C)$$

$$isHard(C)$$

$$\forall x\ \exists y\ likes(x,y) \wedge isHard(y) => isA(x, CS)$$

+ From text to logic

There is a Middlebury Student from Hawaii.

Middlebury students live in Middlebury

+ More examples

All purple mushrooms are poisonous

No purple mushroom is poisonous

Every CS student knows a programming language.

A programming language is known by every CS student

+ How about…

$\forall x\ isA(x, Rose) \Rightarrow \exists y\ has(x, y) \wedge thorn(y)$

"Every rose has its thorn"

$\forall x\ \exists y\ \text{isPerson}(x) \wedge \text{isPerson}(y) \Rightarrow \text{loves}(x,y)$

"Everybody loves somebody"

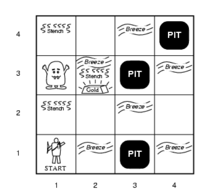
$\exists y\ \forall x\ \text{isPerson}(x) \wedge \text{isPerson}(y) \Rightarrow \text{loves}(x,y)$

"There is someone that everyone loves"

$\forall x\ \exists y\ \exists z\ \text{isPerson}(x) \wedge \text{isPerson}(y) \wedge \text{isTime}(z) \Rightarrow \text{loves}(x,y)$

"Everybody loves somebody, sometime"

+ First-order logic for Wumpus



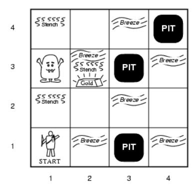
How can we model Wumpus world first order logic?

First-order logic for Wumpus

A little tricky, but much more condensed

$\forall s\ At(s) \wedge FeelBreeze(s) \Rightarrow Breezy(s)$

$\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(s,r) \wedge Pit(r)$



Inference with first-order logic

Similar to predicate logic, can define as a search problem

PROLOG is an example of an implementation of first-order logic

PROLOG

define a new method

```
change([H,Q,D,N,P]) :-
   member(H,[0,1,2]),
   member(Q,[0,1,2,3,4]),
   member(D,[0,1,2,3,4,5,6,7,8,9,10]),
   member(N,[0,1,2,3,4,5,6,7,8,9,10,
           11,12,13,14,15,16,17,18,19,20]),
   S is 50*H + 25*Q +10*D + 5*N,
   S =< 100,
   P is 100-S.
```

define range/possible values (the member lines)

facts (the last three lines)

What would: change([0,2,3,4,6]) give us?

PROLOG

define a new method

```
change([H,Q,D,N,P]) :-
   member(H,[0,1,2]),
   member(Q,[0,1,2,3,4]),
   member(D,[0,1,2,3,4,5,6,7,8,9,10]),
   member(N,[0,1,2,3,4,5,6,7,8,9,10,
           11,12,13,14,15,16,17,18,19,20]),
   S is 50*H + 25*Q +10*D + 5*N,
   S =< 100,
   P is 100-S.
```

define range/possible values (the member lines)

facts (the last three lines)

no solution

+ PROLOG

define a new method

```
change([H,Q,D,N,P]) :-
   member(H,[0,1,2]),
   member(Q,[0,1,2,3,4]),
   member(D,[0,1,2,3,4,5,6,7,8,9,10]),
   member(N,[0,1,2,3,4,5,6,7,8,9,10,
         11,12,13,14,15,16,17,18,19,20]),
   S is 50*H + 25*Q +10*D + 5*N,
   S =< 100,
   P is 100-S.
```

define range/possible values

facts

What would: change([0,2,3,2,P]) give us?

+ PROLOG

define a new method

```
change([H,Q,D,N,P]) :-
   member(H,[0,1,2]),
   member(Q,[0,1,2,3,4]),
   member(D,[0,1,2,3,4,5,6,7,8,9,10]),
   member(N,[0,1,2,3,4,5,6,7,8,9,10,
         11,12,13,14,15,16,17,18,19,20]),
   S is 50*H + 25*Q +10*D + 5*N,
   S =< 100,
   P is 100-S.
```

define range/possible values

facts

P=10 (we can make this work if P=10)

+ PROLOG

define a new method

```
change([H,Q,D,N,P]) :-
   member(H,[0,1,2]),
   member(Q,[0,1,2,3,4]),
   member(D,[0,1,2,3,4,5,6,7,8,9,10]),
   member(N,[0,1,2,3,4,5,6,7,8,9,10,
         11,12,13,14,15,16,17,18,19,20]),
   S is 50*H + 25*Q +10*D + 5*N,
   S =< 100,
   P is 100-S.
```

define range/possible values

facts

What would: change([H,Q,D,N,P]) give us?

+ PROLOG

define a new method

```
change([H,Q,D,N,P]) :-
   member(H,[0,1,2]),
   member(Q,[0,1,2,3,4]),
   member(D,[0,1,2,3,4,5,6,7,8,9,10]),
   member(N,[0,1,2,3,4,5,6,7,8,9,10,
         11,12,13,14,15,16,17,18,19,20]),
   S is 50*H + 25*Q +10*D + 5*N,
   S =< 100,
   P is 100-S.
```

define range/possible values

facts

All possible ways of making change for $1!

PROLOG: N-Queens

```
solve(P) :-
    perm([1,2,3,4,5,6,7,8],P),
    combine([1,2,3,4,5,6,7,8],P,S,D),
    all_diff(S),
    all_diff(D).

combine([X1|X],[Y1|Y],[S1|S],[D1|D]) :-
    S1 is X1 +Y1,
    D1 is X1 -Y1,
    combine(X,Y,S,D).
combine([],[],[],[]).

all_diff([X|Y]) :-  \+member(X,Y), all_diff(Y).
all_diff([X]).
```

http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html

Logic, the good and the bad

Good:

- Mathematicians have been working on it for a while
- Logical reasoning is straightforward
  - tools (like PROLOG) exist to help us out

Bad:

- Dealing with exceptions is hard
  - not all tomatoes are red
  - sometimes our weather rock is wet, even though its not raining
- Can be unintuitive for people
- Going from language to logic is very challenging
- Many restrictions on what you can do

Challenges

General domain reasoning is hard!

- ACTIONS
- TIME
- BELIEFS

Chapt 12 in the book talks about a lot of these challenges

- organizing objects into a hierarchy (shared/inherited properties… like inheritance in programming)
- dealing with measurements
- …

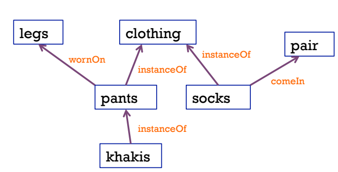
At the end of the day, these don't work very well

Ontology

First-order logic states relationships between objects

One easy way to represent a similar concept is with a graph

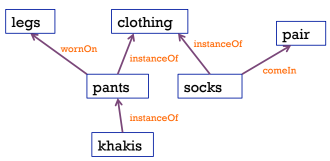
- nodes are the objects
- edges represent relationships between nodes
- some of the quantifier capability is lost



Ontology

Intuitive representation for people

Can pose questions as graph traversals which is often more comfortable/efficient



Opencyc

http://sw.opencyc.org/

The good:
- hundreds of thousands of terms
- millions of relationships
- includes proper nouns
- includes links to outside information (wikipedia)

The bad:
- still limited coverage
- limited/fixed relationships

WordNet

http://wordnet.princeton.edu/

The good:
- 155K words
- word senses (and lots of them)
- part of speech
- example usage
- definitions
- frequency information

- some interesting uses already
  - word similarity based on graph distances
  - word sense disambiguation

WordNet

The bad:
- limited relationships
  - only "linguistic" relationships
  - hyponym (is-a)
  - hypernym (parent of is-a)
  - synonym
  - holonym (part/whole)
- sometimes too many senses/too fine a granularity

+ Open mind common sense

Use the intellect of the masses!

http://openmind.media.mit.edu/

The good:

- much broader set of relationships
- lots of human labeling
  - can collect lots of data
  - human labeled
  - reduces spam
- more general statement engine

+ Open mind common sense

The bad:

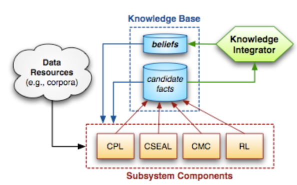
- relies on the user
- still a limited vocabulary
- only scoring is voting
- limited coverage in many domains

+ NELL

NELL: Never-Ending Language Learning

- http://rtw.ml.cmu.edu/rtw/
- continuously crawls the web to grab new data
- learns entities and relationships from this data
  - started with a seed set
  - uses learning techniques based on current KB to learn new information

+ NELL



4 different approaches to learning relationships

Combine these in the knowledge integrator

- idea: using different approaches will avoid overfitting

Initially was wholly unsupervised, now some human supervision

- cookies are food => internet cookies are food => files are food

+ An example learner: coupled pattern learner (CPL)

Cities:

Los Angeles
San Francisco
New York
Seattle
…

… city of X …
… the official guide to X …
… only in X …
… what to do in X …
… mayor of X …

… mayor of X …

extract occurrences of group

statistical co-occurrence test

+ CPL

… mayor of <CITY> …

extract other cities from the data

Albequerque
Springfield
…

+ CPL

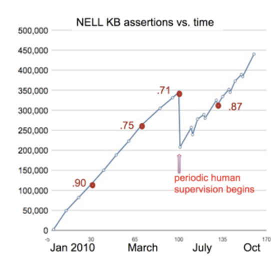
Can also learn patterns with multiple groups

… X is the mayor of Y …
… X plays for Y …
… X is a player of Y …

can extract other groups, but also relationships

Antonio Villaraigosa —mayor of→ Los Angeles

+ NELL performance



estimated accuracy in red

For more details: http://rtw.ml.cmu.edu/papers/carlson-aaai10.pdf

+ NELL

The good:
- Continuously learns
- Uses the web (a huge data source)
- Learns generic relationships
- Combines multiple approaches for noise reduction

The bad:
- makes mistakes (overall accuracy still may be problematic for real world use)
- does require some human intervention
- still many general phenomena won't be captured