Knowledge Representation Issues

CSE-345: Artificial Intelligence

Introduction

- Discussed: Search-based problem solving programs
  - power is limited because of their generality
- Knowledge representation models
  - allow for more specific, more powerful problem-solving mechanisms

Knowledge

- Can be defined as the body of facts & principles accumulated by humankind or the act, fact, or state of knowing.
- True, but incomplete, much more than this
- It is having a familiarity with language, concepts, procedures, rules, ideas, abstractions, places, customs, facts, & associations, coupled with an ability to use these notions effectively in modeling different aspects of the world.

Knowledge

- The meaning of knowledge is closely related to the meaning of intelligence.
- Intelligence requires the possession of an access to knowledge
- And a characteristic of intelligent people is that they posses much knowledge
- Knowledge is likely stored as complex structures of interconnected neurons.
- Symbolic representation

Human brain	**Computer**
3.3 lbs	100 gms
10^{12} neurons	magnetic spots & voltage states
10^{14} bits storage	10^{12} bits doubling about every 3~4 years

- The gap between human & computer storage capacities is narrowing rapidly
- Still wide gape between representation schemes & efficiencies

Knowledge

- Declarative vs. Procedural
  - **Procedural:** compiled knowledge related to the performance of some tasks
    - The steps used to solve an algebraic equation
  - **Declarative:** passive knowledge expressed as statements of facts about the world.
    - Personal data is a database
- **Heuristic Knowledge**: special type of knowledge used by humans to solve complex problems.
  - The knowledge used to make good judgments, or strategies, tricks, or 'rules of thumb' used to simply the solution of problems.
  - Heuristic s are usually acquired with much experience
    - Fault in a TV set
      - an experienced technician will not start by making numerous voltage checks when it is clear that the sound is present but the picture is not
      - The high voltage fly back transformer or related component is the culprit
  - May not always be correct
  - But frequently/quickly can find a solution

Knowledge and Data

- Knowledge should not confused with data
- Physician treating a patient use both Knowledge & Data
  - **Data:** record: history, measurement of vital sign, drugs given, response to drugs,……
  - **Knowledge:** what Physician learned from medical school, internship, residency, specialization, practice.
- Knowledge includes & requires the use of data & information
- It combines relationship, correlations, dependencies, & notion of gestalt with data & information

Belief, Hypothesis, & Knowledge

- **Belief:** define as essentially any meaningful & coherent expression that can be represented
  - It may be true or false
- **Hypothesis:** define as a justified belief that is not known to be true
  - Thus a hypothesis is a belief which is backed up with some supporting evidence, but it may still be false
- **Knowledge:** define as true justified belief

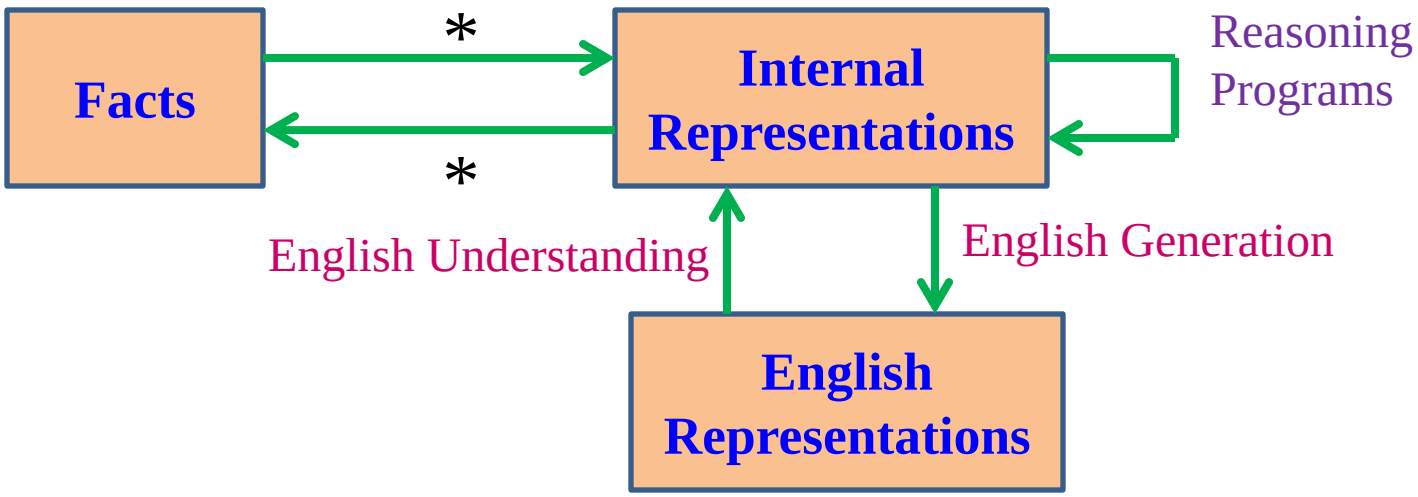
Representations and Mappings

- In order to solve the complex problems encountered in AI
  - -needs a large amount of knowledge
  - -some mechanisms for manipulating that knowledge to create solutions to new problems
- **Two entities:**
  - **Facts:** truths in some relevant world. These are the things we want to represent
  - Representations of facts in some chosen formalism. These are the things we will actually be able to manipulate.

Representations and Mappings

- **Structuring entities:**
  - The *knowledge level*, at which facts are described
  - The *symbol level*, at which representations of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs

Mappings between Facts & Representations



- **Representation mappings:**
  - Forward representation mappings (facts-representation)
  - Backward representation mappings (representation-facts)

Mappings between Facts & Representations

An example using mathematical logic as the representational formalism:

- Spot is a dog: dog(Spot)
- Every dog has a tail: $\forall x: dog(x) \rightarrow hastail(x)$

Use the deductive mechanisms of logic:

hastail(Spot) : Spot has a tail

Mappings between Facts & Representations

- The available mapping functions are not one-to-one
- They are not even functions but rather many-to-many relations
- Each object in the domain may map to several elements in the range
- Several elements in the domain may map to the same element in the range

Mappings between Facts & Representations

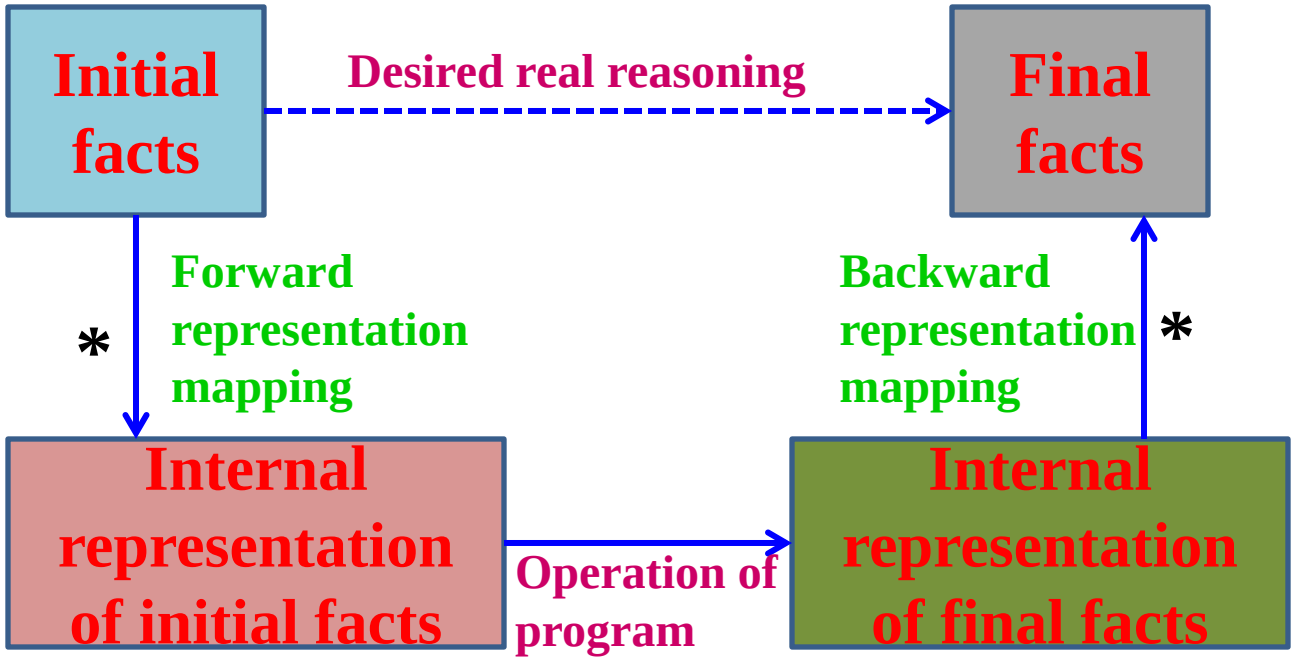
"All dogs have tails" & "Every dog has a tail"

- Both represent the same fact (every dog has at least one tail)
- **But**
  - **Former**: *every dog has at least one tail/each dog has several tails*
  - **Latter:** *every dog has at least one tail/there is a tail that every dog has*
- When we try to convert English sentences into some other representation (such as logical propositions) , we must first decide what facts the sentence represent & then convert those facts into the new representation

Mappings between Facts & Representations

- What an AI program does is to manipulate the internal representation of the facts it is given
- This manipulation should result in new structures than can also be interpreted as the internal representations of facts
- More precisely, these structures should be the internal representations of facts that correspond to the answer to the problem described by the starting set of facts

Representation of Facts: Expanded view



- Dotted line: abstract reasoning process that a program is intended to model
- Solid line: concrete reasoning process that a particular program performs

Representation of Facts: Expanded view

- This program successfully models the abstract process to the extent that, when the backward representation mapping is applied to the program's output, the appropriate final facts are actually generated
- If either the program operation/one of the representation mappings is not faithful to the problem that is being modeled, then the final facts will probably not be the desired ones
- If no good mapping can be defined for a problem, then no matter how good the program to solve the problem is, it will not be able to produce answers that correspond to real answers to the problem

Approaches to Knowledge Representation

- Good system for representation of knowledge should possess 04 properties:
  - **Representational Adequacy**
  - **Inferential Adequacy**
  - **Inferential Efficiency**
  - **Acquisitional Efficiency**
- No single system that optimizes all of the capabilities for all kinds of knowledge has yet been found.
  - Multiple techniques for KR exist.

Approaches to Knowledge Representation

- Simple Relational Knowledge
  - Simplest way to represent declarative facts is as a set of relations of the same sort used in database systems.

Player	Height	Weight	Bats-Throws
Babul	6-0	180	Right-Right
Alim	5-10	170	Right-Right
Didar	6-2	215	Left-Left
Rubel	6-3	205	Left-Right

  - Representation is simple, provides very weak inferential capabilities

Simple Relational Knowledge

- Knowledge represented in this form may serve as the input to more powerful inference engines
- It is not possible even to answer the simple question, "*Who is the heaviest player*?"
- But if a procedure for finding the heaviest player is provided, then these facts will enable the procedure to compute an answer.
- We are provided with a set of rules for deciding which hitter to put up against pitcher (based on right- & left-handedness), then this same relation can provide at least some of the info required by those rules

Approaches to Knowledge Representation

- Inheritable knowledge
  - Relational knowledge corresponds to a set of attributes & associated values that together describe the objectives of the knowledge base

Player	Height	Weight	Bats-Throws
Babul	6-0	180	Right-Right
Alim	5-10	170	Right-Right
Didar	6-2	215	Left-Left
Rubel	6-3	205	Left-Right

  - It is possible to augment the basic representation with inference mechanisms that operate on the structure of the representation

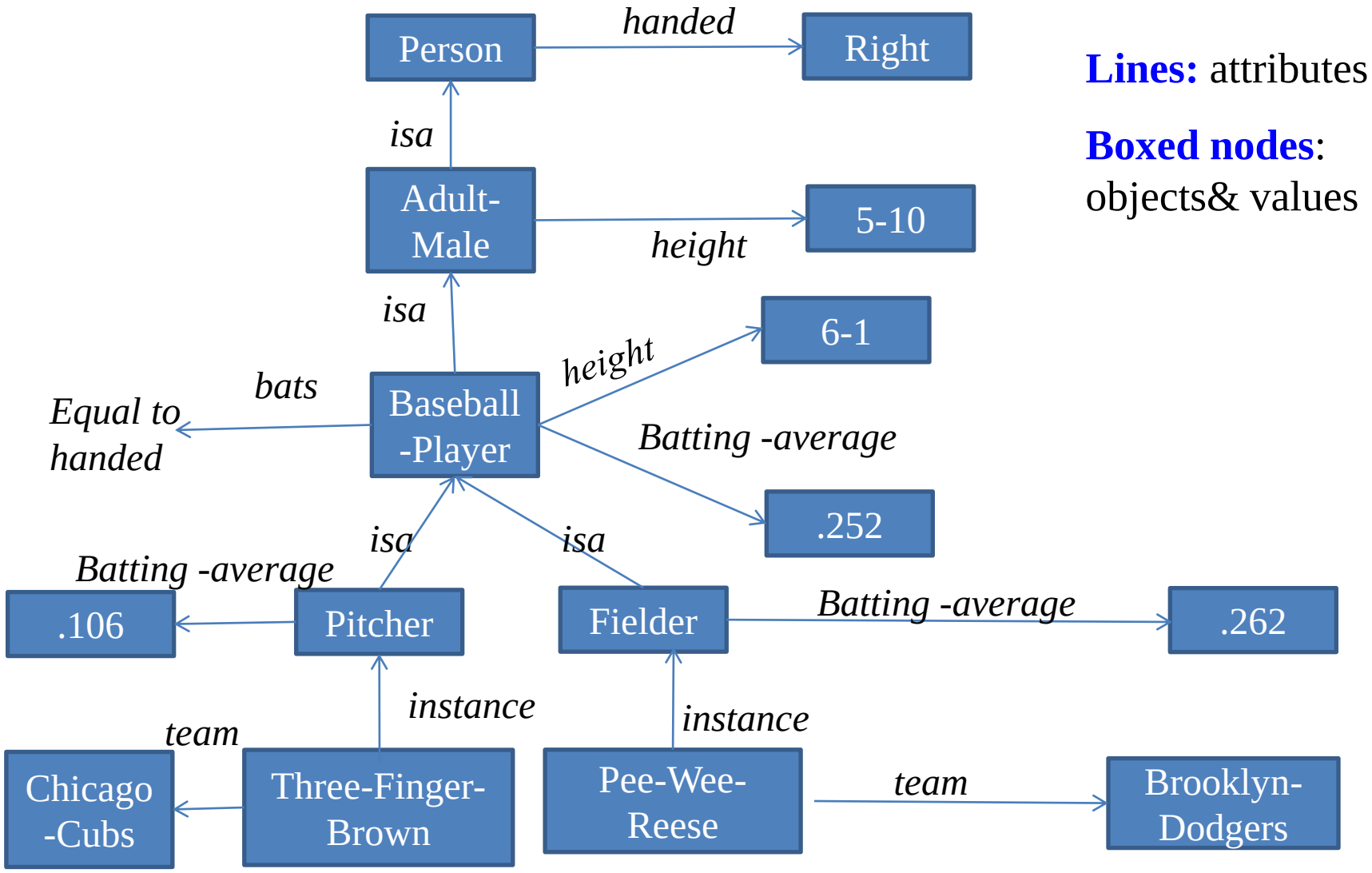
Inheritable knowledge

- For this to be effective, the structure must be designed to correspond to the inference mechanisms that are desired
- One of the most useful forms of inference is *property inheritance*, in which elements of specific classes inherit attributes & values from more general classes in which they are included
- In order to support property inheritance, objects must be organized into classes & classes must be arranged in a generalization hierarchy.

Inheritable knowledge

- Some additional baseball knowledge inserted into a structure that is so arranged.
  - **Lines:** attributes
  - **Boxed nodes**: objects& values
  - Values can also be viewed as objects to its value along the corresponding attribute line
- *Slot-and-filler structure*
- *Semantic network*/a collection of *frames*

Baseball Player: Semantic Net



Lines: attributes

Boxed nodes: objects& values

Baseball Player: Frame

Baseball-player

isa: Adult-Male

bats: (EQUAL handed)

height: 6-1

batting-average: .252

Viewing a Node as a Frame

-All of the objects & most of the attributes to correspond to the baseball domain.

Two Exceptions:

-attribute *isa* show class inclusion

-attribute *instance* show class membership

-these attributes provide the basis for property inheritance as an inference technique

Algorithm: Property Inheritance

To retrieve a value V for attribute A of an instance object O:

1. Find O in the knowledge base
2. If there is a value there for the attribute A, report that value
3. Otherwise, see if there is a value for the attribute instance. If not, then fail
4. Otherwise move to the node corresponding to that value & look for a value for the attribute A. if one found, report it
5. Otherwise, do until there is no value for the ***isa*** attribute or until an answer found:

a) Get the value of the *isa* attribute & move to that node

b) See if there is a value for the attribute A. if there is, report it

Example

- *Team (Pee-Wee-Reese)=Brooklyn-Dodgers*. The attribute had a value stored explicitly in the knowledge base
- *batting-average (Three-Finger-Brown)=.106*. Since there is no value for batting average stored explicitly for Three-Finger-Brown, we follow the *instance* attribute to Pitcher & extract the value stored there.
- *height (Pee-Wee-Reese) = 6-1.*

This represent another default inference.

Note: we get to it first, the more specific fact about the height of baseball players overrides a more general fact about the height of adult males

Example

- *bats (Three-Finger-Brown)=Right*.

To get a value for the attribute bats required going up the *isa* hierarchy to the class Baseball-Player.

But what we found there was not a value but a rule for computing a value

This rule required another value (that for handed) as input

So the entire process must be begun again recursively to find a value *handed*

This time, it is necessary to go all the way up to Person to discover that the default value for handedness for people is *Right.*

Now the rule for *bats* can be applied, producing the result *Right*, that turns out to be wrong, since Brown is a switch hitter (he can hit both left-and-right-handed)

Approaches to Knowledge Representation

- **Inferential knowledge**
  - Facts represented in a logical form (e.g. First-Order Logic: FOL), which facilitates reasoning.
  - This knowledge is useless unless there is also an inference procedure that can exploit it
  - The required inference procedure now is one that implements the standard logical rules of inference
  - There are many such procedures, some of which reason forward from given facts to conclusions, others of which reason backward from desired conclusions to given facts.
  - **Resolution** most commonly used

Approaches to Knowledge Representation

- **Procedural knowledge**
  - Representation of "how to make it" rather than "what it is"
  - Procedural knowledge can be represented in programs in many ways:
    - Code in some programming language, such as Lisp

      →May have inferential efficiency, no inferential adequacy (difficult to write a program that can reason about another program's behaviour), acquisitional efficiency (b/c of the process of updating and debugging large pieces of code)

The Frame Problem

- How to represent efficiently *sequences* of problem states that arise from a search process
- For complex ill-structured problems, this can be a serious matter
- Consider the world of a household robot.
- There are many objects & relationships in the world, & a state description must somehow included facts

  On(Plant, Table), under(Table, Window), and in(Table, Room)

  -one strategy is to store each state description as a list of such facts

The Frame Problem

- But what happens during the problem solving process if each of those descriptions is very long?
- Most of the facts will not change from one state to another, yet each fact will be represented once at every node & will quickly run out of memory
- Furthermore, spend the majority of time creating these nodes & copying these facts-most of which do not change often-from one node to another.
- Spend a lot of time recording *above(Ceiling,Floor)* at every node

  -All of these is, of course, in addition to the real problem figuring out which facts *should* be different at each node

The Frame Problem

- This whole problem of representing the facts that change as well as that do not is known as the *frame problem*.
- In some domains, the only hard part is representing all the facts
- In other, though, figuring out which ones change is nontrivial

Robot World: *there might be a table with a plant on it under the window. Suppose we move the table to the center of the room. We must also infer that the plant is now in the center of the room too but the window is not.*

The Frame Problem

- To support this kind of reasoning, some systems make use of an explicit set of axioms (Frame Axioms).
  - describes all the things that do not change when a particular operator is applied in state n to produce state n+1 (the things that do change must be mentioned as part of the operator itself)

$$color(x,y,s_1) \wedge move(x,s_1,s_2) \rightarrow color(x,y,s_2)$$

The Frame Problem

- Simply start with a description of initial state and then making changes to that description as indicated by the rules applied.
- But when back tracking how to know what change to problem statement need to undone.
- Two solution:
  - Do not modify the initial state description at all. At each node, store an indication of the specific changes that should be made at this node. Whenever it is necessary to refer to the description of the current problem state, look at the initial state description and also look back through all the nodes on the path from the start state to the current state.
  - Modify the initial state description as appropriate, nut also record at each node an indication of what to do undo the move it ever be necessary to back track through the ode. Then whenever it is necessary to back track, check each node along the way and perform the indicated operation on the state description.

The END