Knowledge Representation

Introduction.

- The objective of research into intelligent machines is to produce systems which can reason with available knowledge and so behave intelligently.
- One of the major issues then is how to incorporate knowledge into these systems.

The Problem Of Knowledge Representation

- How is the whole abstract concept of knowledge reduced into forms which can be written into a computers memory.
- This is called the problem of Knowledge Representation.

Fields of Knowledge

- The concept of Knowledge is central to a number of fields of established academic study, including Philosophy, Psychology, Logic,and Education.
- Even in Mathematics and Physics people like Isaac Newton and Leibniz reflected that since physics has its foundations in a mathematical formalism, the laws of all nature should be similarly described.

Views of Knowledge

- The eighteenth Century Psychologist, Immanuel Kant wrote in his landmark "Critique of Pure Reason" that the mind has a priori principles and makes things outside conform to those principles.
- In other words as we absorb knowledge we impose some internal structure on it and tend to view the world in terms of this structure.
- The theories of the great twentieth century Educationalist Jean Piaget would conform to this view.

Knowledge Representation Before Computers

- With the exception of Logic, however, efforts made to formalize a knowledge representation tend to be disjointed and infrequent in the days before computers.
- However the revolutionary impact of the advent of computers proved a new motivating force and many researchers are now addressing the problem of Knowledge Representation.

Major Representation Schemes

- **Classical Representation schemes include**
- **Logic**
- **Procedural**
- **Semantic Nets**
- **Frames**
- **Direct**
- **Production Schemes**
- **Each Representation Scheme has a reasoning mechanism built in so we will treat these issues together and follow this with a discussion on Complexity.**

Logic

- Formal Logic is a classical approach of representing Knowledge.
- It was developed by philosophers and mathematicians as a calculus of the process of making inferences from facts.
- The simplest logic formalism is that of Propositional Calculus which is effectively an equivalent form of Boolean Logic, the basis of Computing.
- We can make Statements or Propositions which can either be atomic (i.e. they can stand alone) or composite which are sentences constructed using atomic statements joined by logical connectives like AND represented by ^ and OR represented by .
- The following is an example of a composite sentence.

  Fred_is_Big ^ Fred_is_Strong

- The semantics of Propositional Logic is based on Truth Assignments. Statements cane have either of the values, TRUE or FALSE assigned to them.
- We can also have rules of Inference. An Inference rule allows for the deduction of new sentences from existing sentences.

FOPC

- First Order Predicate Calculus (FOPC) is an extension of the notions of the propositional calculus.
- The basic notions of statements and logical connectives are retained but certain new features are allowed.
- These include assignments of specific objects in the Domain of Interpretation in addition to TRUE and FALSE and also the notion of predicates and functions.

- Predicates consist of a predicate symbol and a number of arguments called the arity of the predicate e.g. is_red(X) is a predicate of arity 1 in this case X and the predicate symbol is_red.
- A predicate is assigned the value TRUE or FALSE under the assignment of an Interpretation depending on the values of the arguments under the same interpretation.

Quantifiers

- We often have occasion to refer to facts that we know to be true for all or some members of a class.
- For this we use quantification and the notion of the quantifiers For all, and There Exists, . An example of a quantified sentence,

FOPC

- A Calculus is said to be first order if it allows quantification over terms but not over predicates or functions.
- First Order Logic is both sound (impossible to prove a false Statement) and complete (Any true statement has a proof). Obviously then FOPC is a good representation scheme

- Logic is a natural way to express certain notions. The expression of a problem in logic corresponds to our intuitive understanding of a domain. This gives a dimension of clarity to the representation.
- Another advantage of Logic is that it is precise. There are standard methods of establishing semantics of an expression in a logical scheme.
- Incorporating knowledge into a system is a long and changeable process. It is important that modifications be easily made to the knowledge base. In this respect the modularity and flexibility of logic represents a significant advantage.

- The major disadvantage of logic is that proofs for any real problems tend to be computationally unfeasible. So its reasoning power is limited by practical constraints.
- Its big failure as an expressive representation scheme is its failure to adequately represent time or higher order concepts needed in analagous reasoning, generalization and learning.

Procedural Representation

- Logic is what's known as a declarative representation, in that it expresses (declares) Knowledge without specifiying how it is to be used.
- Opponents of the declarative approach adhere to the notion of procedural representation where knowledge is intrinsically bound up in the routines and procedures which use it.
- These procedures and routines know how to do a particular task which would be regarded as intelligent.

SHRDLU

- A landmark Artificial Intelligence system which uses procedural knowledge is Winograds famous blocks world system SHRDLU.
- The knowledge of this system is represented in the PLANNER procedural language. The procedures of SHRDLU know how to recognize other instances of a specific concept, the status of that concept in a given sentence and such things as the conditions in which that concept exists and the consequence of that existence

Advantages

- The major advantage of procedural representation schemes is their ability to use knowledge.
- These systems have a marked directness in their approach to problem solving.
- They solve problems directly without wasting much time searching the problem space.
- Consequently they are usually much more efficient in dealing with the problems to which they are applied than other representations.

- Procedural representation lends itself to encoding using common programming languages.
- This avoids the need for system developers to concern themselves with a whole battery of difficult issues from theorem proving to problem space traversal.
- Hence the development process tends to be quicker and the skills pool inclusive of a wider range of professionals

Problems

- Two major problems which arise in a procedural approach concern completeness and consistency. Many procedural systems are not complete in the sense that given all the facts necessary to make certain deductions they fail to make these deductions. In addition
- Secondly a deductive system is consistent if all deductions are correct. However the use of default reasoning with procedural representation introduce inconsistencies into the deductive process.

- However completeness and consistency are not always fully desirable in AI systems because we humans often work with incomplete knowledge and are willing to make exceptions in certain cases.

Modularity

- Modularity is another feature that is sacrificed in procedural representation.
- It is not easy to modify procedural knowledge because it is usually intrinsically bound up in very complicated code.
- Not so much a problem with Object Oriented Languages

Explanations

- The flow of execution of a procedural system is often unclear and as such it becomes quite difficult to chart the development of a solution to a particular problem.

- Because of this it is often difficult to explain the knowledge and reasoning that went into making a particular decision.
- Many applications require explanations e.g. expert systems so procedural knowledge is at a big disadvantage in this respect.
- The pros and cons of the decalarative versus procedural approaches were the substance of one of the great debates in AI. However this row gave the whole concept of Knowledge Representation huge impetus and lead many researchers to focus their attention on the problem.
- Finally more recent research efforts tend to combine the best aspects of both approaches.

- The pros and cons of the decalarative versus procedural approaches were the substance of one of the great debates in AI.

- However this row gave the whole concept of Knowledge Representation huge impetus and lead many researchers to focus their attention on the problem.
- Finally more recent research efforts tend to combine the best aspects of both approaches.

Direct Knowledge

- There is a kind of representation scheme called analogical or direct reasoning.
- This class of scheme which includes representations such as maps, models , diagrams and sheet music, can represent knowledge about certain aspects of the world in especially natural ways.
- A street map, depicting any town or city, is a typical example of direct reasoning because a street on the map corresponds in size and orientation to the real street it represents.
- Also the distance between any two points on the map corresponds to the distance between the places they represent in the city.

Correspondence

- Correspondence is the key requirement in direct representations.
- There must be a correspondence between the important relations in the representing data structure and the relations in the represented situation.

Expressiveness

- Direct representations do not represent everything in a given situation. they are only direct with respect to certain things. For example a street map is direct with respect to location and distance but not usually to elevation.

Advantages

- For some problems direct representation is particularly advantageous.
- For one the problem of updating the representation to reflect changes in the world is usually far simpler than in other representations.
- An example of this to add a new town to a map we simply put it in the right place. It is not necessary to state explicitly its distance from the towns that are already there since distance on the map corresponds to distance in the real world.
- Another of the advantages of direct representations over their counterparts relates to the difference between observation and deduction.

More Advantages

- In some situations observation can be accomplished relatively cheaply in terms of computation using direct representation. For example it would be easier to see if three points are colinear using the direct representation of a diagram rather than geometric calculations.
- The use of direct representation can facilitate search constraints in the problem situation.
- Constraints are represented by constraints on the types of transformations applied to the representation so that impossible strategies are rejected immediately.

Difficulties

- However the advantages of efficiency and convenience during the actual processing of a problem must be weighed against the problems of setting up a direct representation in the first place.
- Direct representation schemes tend to represent specific instances and there are times when generality is needed. For example one map might show a town which has a university. However from that there is no way whether we could say this is a property of all towns.

More Difficulties

- There is the possibility too that some features in a direct representation might not hold in the actual situation and we might not know which ones these are.
- Mountains on a map for instance might be coloured bright yellow.It would be wrong to infer that the actual mountains are this colour.
- All this is part of the problem of knowing the limits of the representation schemes.
- Direct representations become unwieldy when we have to make inferences to fill in gaps in the knowledge.
- Sometimes we only know indirectly where something is to be entered into a direct representation and need to make complicated inferences to find the exact location. In such cases the power of direct representation is diminished.

Conclusion

- To conclude then direct representations are useful in some situations only. In others the problems which arise far outweigh the benefits.

Knowledge representation in Summary

- The choice of representation scheme is one of the most important issues concerning any intelligent system.
- This is so because a bad representation scheme can create many problems in both the design and execution of the system.
- The following are some general features that are desirable in a Knowledge Representation Scheme.

- It is important that the scheme chosen should be suited to the particular problem domain for which the program is designed.
- The representation should reflect the nature of knowledge associated with the problem domain and it should be easy to express this knowledge in the representation formalism.
- It is required that the representation scheme be modular.

More

- The representation scheme should be flexible enough to represent the many diverse forms of knowledge required in problem solving.
- The representation scheme should be mathematically sound and complete to guarantee the veracity of its inferences and its ability to make them.
- It is important tha the path to solution be clearly understood for explanation purposes and the representation scheme should not hinder this.

And Finally

- Efficient traversal of the problem search space is obviously central and the representation scheme should facilitate this.
- A system must know how to use its knowledge and the representation scheme must accommodate this.
- It may be that one single scheme will not provide all of these for a particular problem domain and a mixture of schemes must be used.

Outline

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL

Pros and cons of propositional logic

☺ Propositional logic is declarative

☺ Propositional logic allows partial/disjunctive/negated information
- (unlike most data structures and databases)
-

☺ Propositional logic is compositional:

☺
- meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
-

☺ Meaning in propositional logic is context-independent

- (unlike natural language, where meaning depends on context)
-

☹ Propositional logic has very limited expressive power

- (unlike natural language)
- E.g., cannot say "pits cause breezes in adjacent squares“
  - except by writing one sentence for each square

First-order logic

- Whereas propositional logic assumes the world contains facts,
- first-order logic (like natural language) assumes the world contains

  - Objects: people, houses, numbers, colors, baseball games, wars, …
  -
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

Syntax of FOL: Basic elements

- Constants KingJohn, 2, NUS,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
- Equality =
- Quantifiers $\forall$, $\exists$

Atomic sentences

Atomic sentence = *predicate* (*$term_1$*,...,*$term_n$*)
or *$term_1$* = *$term_2$*

Term = *function* (*$term_1$*,...,*$term_n$*)
or *constant* or *variable*

- E.g., *Brother(KingJohn,RichardTheLionheart)* > *(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences

- Complex sentences are made from atomic sentences using connectives

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$,

E.g. *Sibling(KingJohn,Richard)* $\Rightarrow$ *Sibling(Richard,KingJohn)*

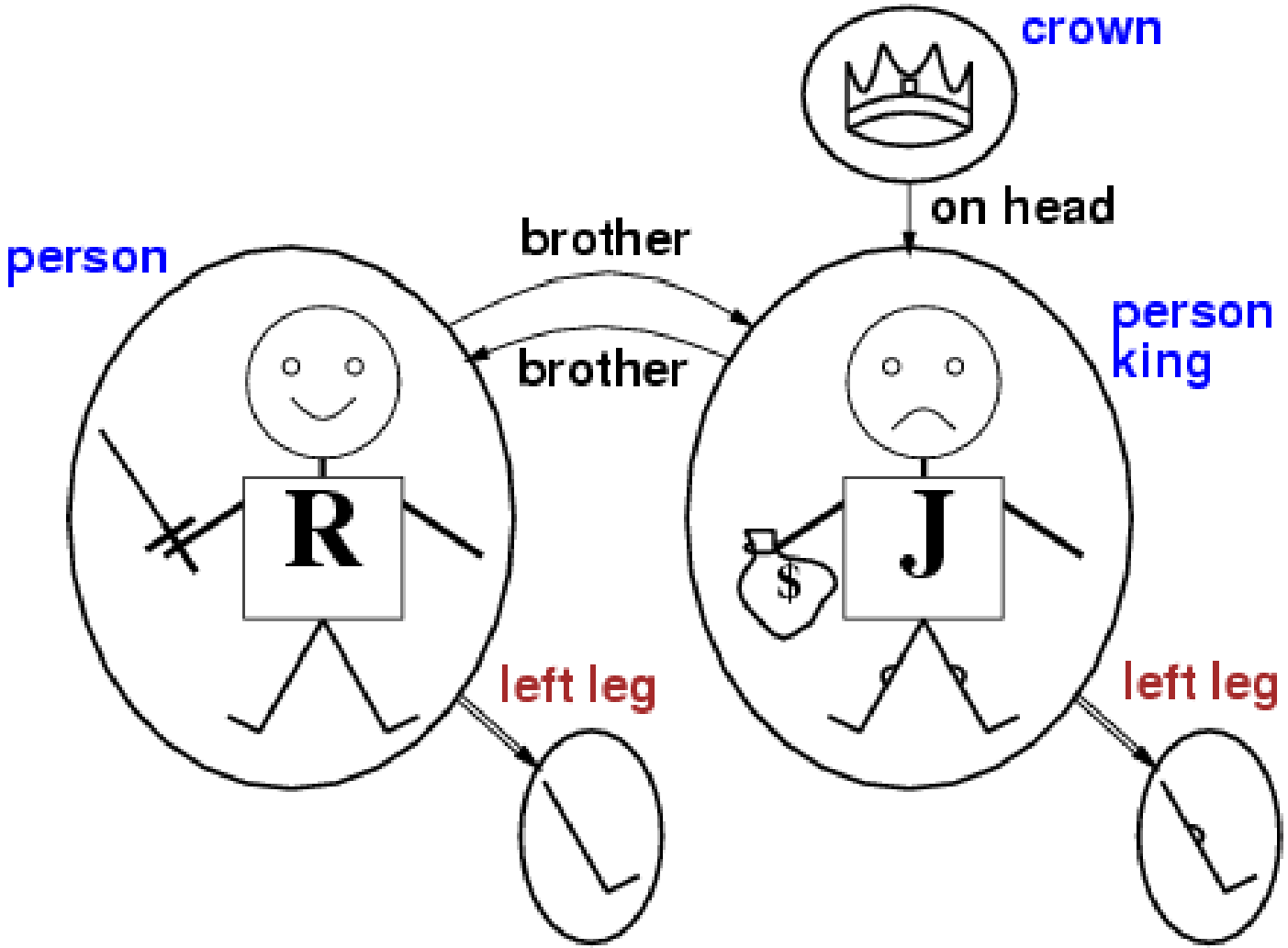
$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for
  constant symbols → objects
  predicate symbols → relations
  function symbols → functional relations
- An atomic sentence *predicate*($term_1$,...,$term_n$) is true iff the objects referred to by $term_1$,...,$term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Universal quantification

- $\forall$<*variables*> <*sentence*>

Everyone at NUS is smart:
$\forall x$ At(x,NUS) $\Rightarrow$ Smart(x)

- $\forall x$ *P* is true in a model *m* iff *P* is true with *x* being each possible object in the model

- Roughly speaking, equivalent to the conjunction of instantiations of *P*

  At(KingJohn,NUS) $\Rightarrow$ Smart(KingJohn)
  $\wedge$ At(Richard,NUS) $\Rightarrow$ Smart(Richard)
  $\wedge$ At(NUS,NUS) $\Rightarrow$ Smart(NUS)
  $\wedge$ ...

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

  $\forall x$ At(x,NUS) $\wedge$ Smart(x)

  means “Everyone is at NUS and everyone is smart”

Existential quantification

- ∃<*variables*> <*sentence*>

- Someone at NUS is smart:
- ∃*x* At(x,NUS) ∧ Smart(x)$

- ∃*x P* is true in a model *m* iff *P* is true with *x* being some possible object in the model

- Roughly speaking, equivalent to the disjunction of instantiations of *P*

  At(KingJohn,NUS) ∧ Smart(KingJohn)
  ∨ At(Richard,NUS) ∧ Smart(Richard)
  ∨ At(NUS,NUS) ∧ Smart(NUS)
  ∨ ...

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \; \text{At}(x,\text{NUS}) \Rightarrow \text{Smart}(x)$$

is true if there is anyone who is not at NUS!

Properties of quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$

- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
- $\exists x\ \forall y$ Loves(x,y)
  - "There is a person who loves everyone in the world"
  -
- $\forall y\ \exists x$ Loves(x,y)
  - "Everyone in the world is loved by at least one person"
  -

- Quantifier duality: each can be expressed using the other
- $\forall x$ Likes(x,IceCream) $\neg\exists x\ \neg$Likes(x,IceCream)
- $\exists x$ Likes(x,Broccoli) $\neg\forall x\ \neg$Likes(x,Broccoli)

Equality

- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

- E.g., definition of *Sibling* in terms of *Parent*:

  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \wedge \exists m,f \neg (m = f) \wedge Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$

Using FOL

The kinship domain:

- Brothers are siblings
  $\forall x,y\ Brother(x,y) \Leftrightarrow Sibling(x,y)$

- One's mother is one's female parent
  $\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$

- “Sibling” is symmetric
  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$

Using FOL

The set domain:

- $\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x|s_2\})$
- $\neg\exists x, s\ \{x|s\} = \{\}$
- $\forall x, s\ x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x, s\ x \in s \Leftrightarrow [\exists y, s_2\} (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
- $\forall s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1, s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x, s_1, s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- $\forall x, s_1, s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Interacting with FOL KBs

- Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at *t=5*:

  `Tell`(KB,Percept([Smell,Breeze,None],5))
  `Ask`(KB,∃a BestAction(a,5))
- I.e., does the KB entail some best action at *t=5*?

- Answer: *Yes*, {*a/Shoot*} ← substitution (binding list)

- Given a sentence *S* and a substitution σ,
- *S*σ denotes the result of plugging σ into *S*; e.g.,
  S = Smarter(x,y)
  σ = {x/Hillary,y/Bill}
  *S*σ = Smarter(Hillary,Bill)

- `Ask`(KB,S) returns some/all σ such that KB ⊨ σ

Knowledge base for the wumpus world

- Perception
  - $\forall t,s,b$ Percept([s,b,Glitter],t) $\Rightarrow$ Glitter(t)
  -
- Reflex
  - $\forall t$ Glitter(t) $\Rightarrow$ BestAction(Grab,t)

Deducing hidden properties

- $\forall x,y,a,b$ *Adjacent*$([x,y],[a,b]) \Leftrightarrow$
  $[a,b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

Properties of squares:

- $\forall s,t$ *At*$(\text{Agent},s,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$

Squares are breezy near a pit:

  - Diagnostic rule---infer cause from effect
    $\forall s$ Breezy(s) $\Rightarrow$ \Exi{r} Adjacent(r,s) $\wedge$ Pit(r)$

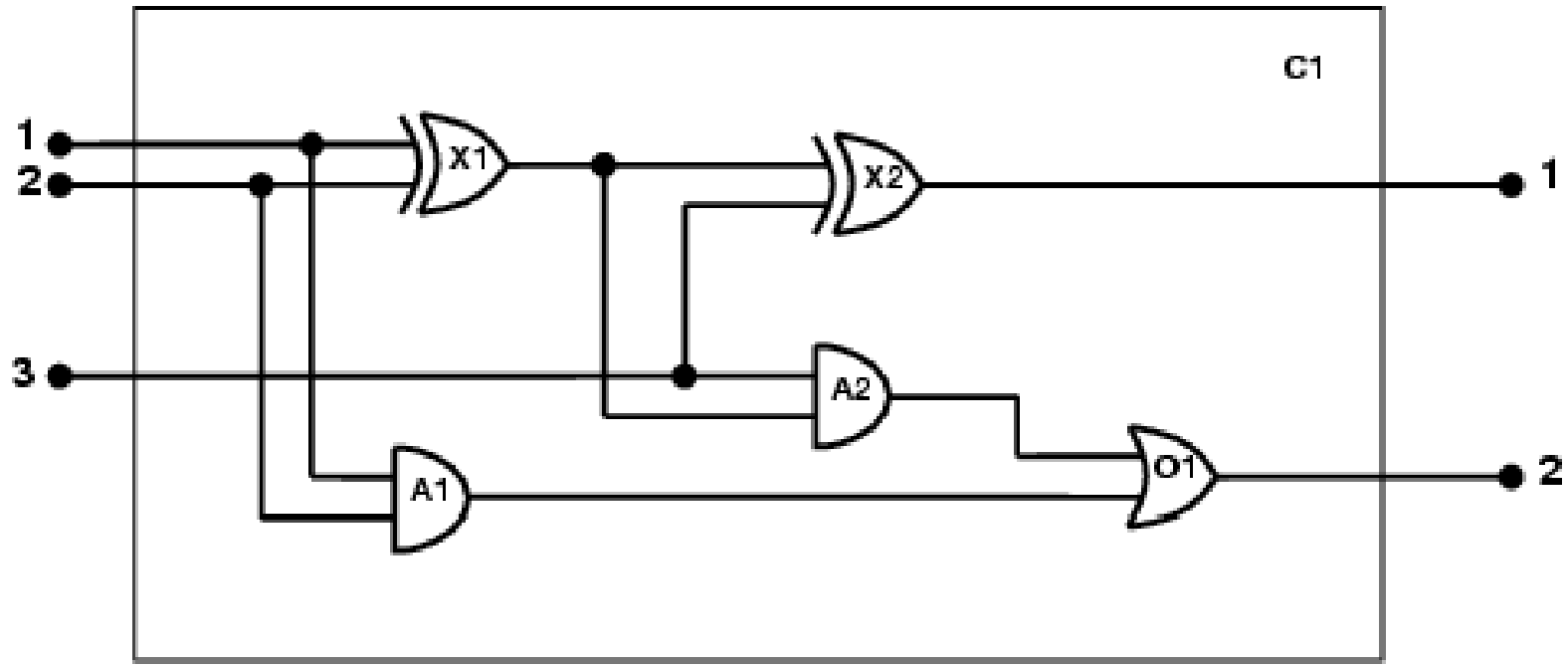
  - Causal rule---infer effect from cause
    $\forall r$ Pit(r) $\Rightarrow$ [$\forall s$ Adjacent(r,s) $\Rightarrow$ Breezy(s)$]

Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder



The electronic circuits domain

1. Identify the task
   - Does the circuit actually add properly? (circuit verification)

2. Assemble the relevant knowledge
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant: size, shape, color, cost of gates

3. Decide on a vocabulary
   - Alternatives:

     $\text{Type}(X_1) = \text{XOR}$

     $\text{Type}(X_1, \text{XOR})$
     $\text{XOR}(X_1)$

The electronic circuits domain

4. Encode general knowledge of the domain

$\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Signal(t_1) = Signal(t_2)

- $\forall t$ Signal(t) = 1 $\vee$ Signal(t) = 0
- $1 \neq 0$
- $\forall t_1,t_2$ Connected(t_1, t_2) $\Rightarrow$ Connected(t_2, t_1)
- $\forall g$ Type(g) = OR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 1
- $\forall g$ Type(g) = AND $\Rightarrow$ Signal(Out(1,g)) = 0 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 0
- $\forall g$ Type(g) = XOR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ Signal(In(1,g)) $\neq$ Signal(In(2,g))
- $\forall g$ Type(g) = NOT $\Rightarrow$ Signal(Out(1,g)) $\neq$ Signal(In(1,g))

The electronic circuits domain

5. Encode the specific problem instance

$Type(X_1) = XOR$ $Type(X_2) = XOR$

$Type(A_1) = AND$ $Type(A_2) = AND$

$Type(O_1) = OR$

$Connected(Out(1,X_1),In(1,X_2))$ $Connected(In(1,C_1),In(1,X_1))$

$Connected(Out(1,X_1),In(2,A_2))$ $Connected(In(1,C_1),In(1,A_1))$

$Connected(Out(1,A_2),In(1,O_1))$ $Connected(In(2,C_1),In(2,X_1))$

$Connected(Out(1,A_1),In(2,O_1))$ $Connected(In(2,C_1),In(2,A_1))$

$Connected(Out(1,X_2),Out(1,C_1))$ $Connected(In(3,C_1),In(2,X_2))$

$Connected(Out(1,O_1),Out(2,C_1))$ $Connected(In(3,C_1),In(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

$\exists i_1,i_2,i_3,o_1,o_2$ Signal(In(1,C_1)) = $i_1 \wedge$ Signal(In(2,C_1)) = $i_2 \wedge$
Signal(In(3,C_1)) = $i_3 \wedge$ Signal(Out(1,C_1)) = $o_1 \wedge$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

May have omitted assertions like $1 \neq 0$

Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers

- Increased expressive power: sufficient to define wumpus world