Knowledge Representation

Knowledge Representation

Background

- One of the core problems in developing an intelligent system is **knowledge representation**, i.e., solving the problems of
  - (1) how to represent the knowledge one has about a problem domain, and
  - (2) how to reason using that knowledge in order to answer questions or make decisions
- Knowledge representation deals with the problem of how to model the world sufficiently for intelligent action

Background (2)

- Logic is one of the oldest representation languages studied for AI, and is the foundation for many existing systems that use logic as either inspiration or the basis for the tools in that system (e.g., rule-based expert systems and the Prolog programming language)
- For a knowledge-based intelligent agent, we need:
  - To represent knowledge about the world in a *formal language*
  - To reason about the world using *inferences* in the language
  - To decide what action to take by *inferring* that the selected action is good

Representation Languages

- Fundamental problem of designing a knowledge representation language is the fundamental tradeoff between:
  - (1) a language that is **expressive** enough to represent the important objects and relations in a problem domain, yet
  - (2) allows for a **tractable** (i.e., efficient) means of reasoning and answering questions about implicit information in a reasonable amount of time
- **Logic** is a well-studied, general-purpose language for describing what's true and false in the world, along with mechanical procedures that can operate on sentences in the language to perform reasoning (i.e., to determine what "implicitly follows" from what is explicitly represented)

Knowledge Representation

- A framework for storing knowledge and manipulating knowledge

Or

- A set of syntactic and semantic conventions that make it possible to describe things

- There is always a relationship between the form in which knowledge is represented and the way in which the knowledge is used.
  - You can use domain-specific or general-purpose representation.

What to represent

- Let us first consider what kinds of knowledge might need to be represented in AI systems:
  - **Objects** --- Facts about objects in our world domain. *e.g.* cars have wheels, cows are herbivores.
  - **Events** -- Actions that occur in our world. *e.g. The power-sharing deal was sealed on 28th February 2008*
  - **Performance** -- A behavior like *playing volleyball* involves knowledge about how to do things like serve, boost, spike and block.
  - **Meta-knowledge** -- knowledge about what we know.

What to represent

- Thus in solving problems in AI we must represent knowledge and
- There are two entities to deal with, i.e. facts and representation of these.
  - **Facts** -- truths about the real world and what we represent.
    - This can be regarded as the *knowledge level*
  - **Representation of the facts** which we manipulate.
    - This can be regarded as the *symbol level* since we usually define the representation in terms of symbols that can be manipulated by programs.

Uses of knowledge

Knowledge may be used in the following ways:

- **Learning**
  - acquiring knowledge. This is more than simply adding new facts to a knowledge base. New data may have to be *classified* prior to storage for easy *retrieval, etc.*. *Interaction* and *inference* with existing facts to avoid redundancy and replication in the knowledge and and also so that facts can be updated.
- **Retrieval**
  - The representation scheme used can have a critical effect on the *efficiency* of the method. Humans are very good at it.
  - Many AI methods have tried to model humans

Uses of knowledge (2)

- **Reasoning**
  - Infer facts from existing data.
  - If a system only knows: ***Daisy is a world class model. All world class models are beautiful***.
  - If things like ***Is Daisy a world class model****?* or ***Are world class models beautiful****?* are asked then the answer is readily obtained from the data structures and procedures. However a question like ***Is Daisy beautiful****?* requires reasoning.

- ***Note:*** The above are all related. For example, it is fairly obvious that learning and reasoning involve retrieval *etc*.

Properties of Knowledge Representation Systems

The following properties should be possessed by a knowledge representation system.

1) ***Representational Adequacy***
   - the ability to represent the required knowledge;
2) ***Inferential Adequacy***
   - the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Properties–ctnd…

3) *Inferential Efficiency*
   - the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

4) *Acquisitional Efficiency*
   - the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Properties-ctnd…

Note:

- To date no single system optimises all of the above.
- Currently, logic based representation is still the theoretical foundation of various kinds of knowledge representation.

Knowledge Representation Schemes

- A number of knowledge representation schemes (or formalisms) have been used to represent the knowledge of humans in a systematic manner. This knowledge is represented in a **knowledge base** such that it can be retrieved for solving problems. Amongst the well-established knowledge representation schemes are:
  - Natural language
  - Logic
    - Predicate and Propositional Logic
    - Conceptual or Terminological Logics
  - Production Rules
  - Semantic Networks
  - Frames
  - Conceptual Dependency Grammar
  - Conceptual Graphs
  - Ontology
  - XML / RDF

Logic Representation

Logic representation

- Logic is a formal system in which the formulas or sentences have true or false values
- A logic includes:
  - **Syntax**: Specifies the symbols in the language and how they can be combined to form sentences. Hence facts about the world are represented as sentences in logic
  - **Semantics**: Specifies what facts in the world, a sentence refers to. Hence, also specifies how you assign a truth value to a sentence based on its meaning in the world. A **fact** is a claim about the world, and may be true or false.
  - **Inference Procedure**: Mechanical method for computing (deriving) new (true) sentences from existing sentences

Logic representation (2)

- **Facts** are claims about the world that are True or False, whereas a **representation** is an expression (sentence) in some language that can be encoded in a computer program and stands for the objects and relations in the world
- We need to ensure that the representation is consistent with reality, so that the following figure holds:

```
                                   entails
Representation:     Sentences ---------------> Sentences
                   |                          |
                   |                          |
                   |  Semantics               |  Semantics
                   |  refer to                |  refer to
                   |                          |
                         \/        follows          \/
World:                 Facts -----------------> Facts
```

Logic representation (3)

- **Truth**: A sentence is True if the state of affairs it describes is actually the case in the world. So, truth can only be assessed with respect to the semantics. Yet the computer does not know the semantics of the knowledge representation language, so we need some way of performing inferences to derive valid conclusions even when the computer does not know what the semantics (the interpretation) is
- To build a logic-based representation:
  - User defines a set of primitive symbols and the associated semantics
  - Logic defines the ways of putting these symbols together so that the user can define legal sentences in the language that represent true facts in the world
  - Logic defines ways of inferring new sentences from existing ones

Representation, Reasoning, and Logic

- The objective of knowledge representation is to express knowledge in a computer-tractable form, so that agents can perform well.
- A representation that has clear syntax and semantics is a logic representation
- Logics include:
  - Propositional logic
  - Predicate logic (FOPC)
  - HOPC
  - Fuzzy Logic
  - Temporal logic
  - Description Logics
  - e.t.c.

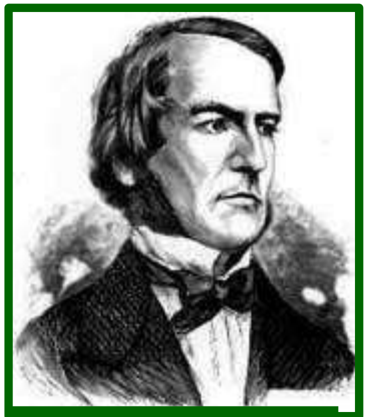
Propositional Logic

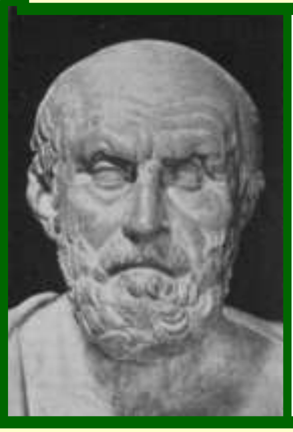
Propositional Logic

Propositional Logic is the logic of compound statements built from simpler statements using so-called *Boolean connectives.*

Some applications in computer science:

- Design of digital electronic circuits.
- Expressing conditions in programs.
- Queries to databases & search engines.


George Boole (1815-1864)


Chrysippus of Soli (ca. 281 B.C. – 205 B.C.)

Definition of a *Proposition*

Definition: A *proposition* (denoted *p*, *q*, *r*, …) is simply:

- a *statement* (*i.e.*, a declarative sentence)
  - *with some definite meaning*, (not vague or ambiguous)
- having a *truth value* that's either *true* (**T**) or *false* (**F**)
  - it is **never** both, neither, or somewhere "in between!"
    - However, you might not *know* the actual truth value,
    - and, the truth value might *depend* on the situation or context.

Propositional logic

- **Logical constants**: true, false
- **Propositional symbols**: P, Q, S, ... (**atomic sentences**)
- Wrapping **parentheses**: (…)
- Sentences are combined by **connectives**:

$\wedge$	...and	[conjunction]
$\vee$	...or	[disjunction]
$\Rightarrow$	...implies	[implication / conditional]
$\Leftrightarrow$	..is equivalent	[biconditional]
$\neg$	...not	[negation]

- **Literal**: atomic sentence or negated atomic sentence

Propositional Logic: Syntax

- **Symbols:**

  Logical constants: true (T), false (F)

  Propositional symbols: P, Q, S, ...

  logical connectives:

∧	...conjunction (and)	∨	...disjunction (or)
¬	...negation (not)	=>	...implication (if)
<=>	...logical equivalence (if and only if)		

  Wrapping **parentheses**: (…)

- A **proposition** (denoted by a proposition symbol) is a declarative statement which can be either true or false but not both or neither.
  - The moon is made of mursik (F)
  - Thika is closer to Nyeri than to Kisumu (T)

Syntax-2

- **Sentence**
  1. T or F itself is a sentence
  2. Individual proposition symbols P, Q, ... are sentences
  3. If S is a sentence, so is (S)
  4. If S1 and S2 are sentences, so are
     S1 ∧ S2, S1∨ S2, S1 => S2, S1 <=> S2, ¬ S1
  5. Nothing else is a sentence
- **Order of precedence** of logical connectors

¬ , ∧, ∨, => , <=>

- Atomic sentences: T, F, P, Q, ... i.e. any capital letter
- Literals: atomic sentences and their negations

Syntax-3

Example Sentences

- P means "It is hot"
- Q means "It is humid"
- R means "It is raining"
- P ^ Q => R

  "If it is hot and humid, then it is raining"
- Q => P

  "If it is humid, then it is hot"
- Q

  "It is humid."

Propositional Logic (PL): Semantics

- Need an **interpretation** of symbols for a given set of sentences
  - Proposition symbols do not have meaning by themselves
  - An interpretation connects proposition symbols to a statement about the world (which may be true or false in that world)
  - An interpretation in PL can be defined as an **assignment of truth values** to all proposition symbols involved
  - There are many interpretations for a given set of sentences (2^n if they involve n distinct proposition symbols)
  - *Example:* Use truth tables to provide semantics
  - Give a meaning to each logical connectives
  - Provide semantics

Interpretation and Truth Tables

- Truth table gives us operational definitions of important logical operators.
  - By using truth table, the truth values of well-formed formulae are calculated.
- Truth table elaborates all possible truth values of a formula.
- The meanings of the logical operators are given by the following truth table.

P	Q	~P	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Some definitions

- A **model** is an interpretation of a set of sentences such that every sentence is *True*. A model is just a formal mathematical structure that "stands in" for the world.
- A sentence is **satisfiable** if it is T*rue* under *some* interpretation(s)
- A **valid sentence** or **tautology** is a sentence that is *True* under *all* possible interpretations, no matter what the world is actually like or what the semantics is, e.g., *it is raining or it is not raining*
- An **inconsistent sentence** or **contradictio**n is a sentence that is *False* under *all* interpretations. The world is never like what it describes, e.g., *"it is raining and it is not raining"*
- **P entails Q** (Q is a **logical consequence** of P, Q **logically follows** P), written P |= Q, means that whenever P is True, so is Q. In other words, all models of P are also models of Q. e.g. ((P=>Q) AND P)=>Q

Some terms

- The meaning or **semantics** of a sentence determines its **interpretation**.
- Given the truth values of all symbols in a sentence, it can be "evaluated" to determine its **truth value** (True or False).
- A **model** for a KB is a "possible world" (assignment of truth values to propositional symbols) in which each sentence in the KB is True.
  - A **model** is an interpretation of a set of sentences such that every sentence is *True*. A model is just a formal mathematical structure that "stands in" for the world.

Inference Rules

- Logical Inference is used to create new sentences X that logically follow from a given set of sentences S (in KB and from input), i.e., S |- X
  - This kind of inference is also known as *deduction*. It can be done by constructing truth tables.
  - Below are some examples of rules of inference.
  - Each can be shown to be sound using a truth table -- a rule is sound if it's conclusion is true whenever the premise is true.

Inference rules-1

- An inference rule is **sound** if every sentence X produced by an inference rule operating on a KB logically follows from the KB. (That is, the inference rule does not create any contradictions)
- An inference rule is **complete** if it is able to produce every expression that logically follows from (is entailed by) the KB. (Note the analogy to complete search algorithms.)

Inference Rules-2

RULE	PREMISE	CONCLUSION
Modus Ponens	A, A => B	B
Modus Tollens	¬B, A => B	¬ A
And Introduction	A, B	A ^ B
And Elimination	A ^ B	A
Or Introduction	A	A v B
Double Negation	¬ ¬ A	A
Chaining	A => B, B => C	A => C
Resolution rule		
Unit Resolution	A v B, ¬ A	B
Resolution	A v B, ¬ B v C	A v C

Weakness of PL

- Hard to identify "**individuals**." E.g., Mary, 3
  - Individuals cannot be PL sentences themselves.
- Difficult to directly and clearly talk about **properties** of individuals or **relations** between individuals (hard to connect individuals to class properties).
  - E.g., property of being a human implies property of being mortal
- Generalizations, patterns, regularities can't easily be represented.
  - All members of a class have this property
  - Some member of a class have this property

Weaknesses of PL-2

- A better representation is needed to capture the relationship (and distinction) between objects and classes, including properties belonging to classes and individuals
- First-Order Logic (abbreviated FOL or FOPC) is expressive enough to concisely represent this kind of situation by separating classes and individuals
  - It has an Explicit representation of individuals and classes, x, Mary, 3, persons.
  - And also Adds relations, variables, and quantifiers, e.g.,
    - *"Every person is mortal"* Forall X: person(X) => mortal(X)
    - *"There is a white alligator"* There exists some X: Alligator(X) ^ white(X)

First Order Logic

- Also First Order predicate Calculus

First Order (Predicate) Logic (FOL)

- First-order logic is used to model the world in terms of
  - **objects** which are things with individual identities
    e.g., individual students, lecturers, companies, cars ...
  - **properties** of objects that distinguish them from other objects
    e.g., mortal, blue, oval, even, large, ...
  - **classes** of objects (often defined by properties)
    e.g., human, mammal, machine, ...
  - **relations** that hold among objects
    e.g., brother of, bigger than, outside, part of, has color, occurs after, owns, a member of, ...
  - **functions** which are a subset of the relations in which there is only one ``value'' for any given ``input''.
    e.g., father of, best friend, second half, one more than ...

Representing knowledge in FOL

- Typically, the user provides:
  - **Constant symbols,** which represent individuals in the world
    - Mary
    - 3
    - Green
  - **Function symbols,** which map individuals to individuals
    - father-of(Mary) = John
    - color-of(Sky) = Blue
  - **Predicate symbols,** which map individuals to truth values
    - greater(5,3)
    - green(Grass)
    - color(Grass, Green)

Representing Knowledge in FOL Provides

- Thereafter, FOL provides:
  - **Variable symbols**
    - E.g., x, y, z
  - **Connectives**
    - Same as in PL: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
  - **Quantifiers**
    - Universal $\forall$**x** or **(Ax)**
    - Existential $\exists$**x** or **(Ex)**
  - The features provided by FOL form part of its grammar

Syntax of FOL

- **Predicates:** P(x[1], ..., x[n])
  - P: **predicate name**; (x[1], ..., x[n]): **argument list**
  - A special function with range = {T, F};
  - Examples: human(x), /* x is a human */
    father(x, y) /* x is the father of y */
  - When all arguments of a predicate is assigned values (said to be **instantiated**), the predicate becomes either true or false, i.e., it becomes a proposition. Ex. Father(Fred, Joe)
  - A predicate, like a membership function, defines a set (or a class) of objects

Syntax of FOL-2

- **Terms** (arguments of predicates must be terms)
  - **Constants** are terms (e.g., Fred, a, Z, “red”, etc.)
  - **Variables** are terms (e.g., x, y, z, etc.), a variable is **instantiated** when it is assigned a constant as its value
  - **Functions** of terms are terms (e.g., f(x, y, z), f(x, g(a)), etc.)
  - A term is called a **ground** term if it does not involve variables
  - Predicates, though special functions, are not terms in FOL

Syntax of FOL-3

- **Quantifiers**

  Universal quantification ∀ (or *forall*)

  - (∀x)P(x) means that P holds for **all** values of x in the domain associated with that variable.
  - E.g., (∀x) luo(x) => kenyan(x)

    (∀x) kenyan(x) => african(x)
  - Universal quantifiers often used with "implication (=>)" to form "rules" about properties of a class

    (∀x) student(x) => smart(x) (All students are smart)

Syntax of FOL-4

- Often associated with English words “all”, “everyone”, “always”, etc.
- You rarely use universal quantification to make blanket statements about every individual in the world (because such statement is hardly true)

(∀x)student(x)^smart(x) means everyone in the world is a student and is smart.

Syntax of FOL-5

Existential quantification ∃

- (∃x)P(x) means that P holds for **some** value(s) of x in the domain associated with that variable.
- E.g., (∃x) mammal(x) ^ lays-eggs(x)-there is a mammal that lays eggs
- Existential quantifiers usually used with "^ (and)" to specify a list of properties about an individual.

  (∃x) student(x) ^ smart(x) (there is a student who is smart.)
- A common mistake is to represent this English sentence as the FOL sentence:

  (∃x) student(x) => smart(x)
- Often associated with English words "someone", "sometimes", etc.

- All variables need to be quantified.

Sentences are built from terms and atoms

- A **term** (denoting a individual in the world) is a constant symbol, a variable symbol, or a function of terms.
- An **atom** (atomic sentence) is a predicate P(x[1], ..., x[n])
  - Ground atom: all terms in its arguments are ground terms (does not involve variables)
  - A ground atom has value true or false (like a proposition in PL)
- A **literal** is either an atom or a negation of an atom
- A **sentence** is an atom, or,
  - ~P, P v Q, P ^ Q, P => Q, P <=> Q, (P) where P and Q are sentences
  - If P is a sentence and x is a variable, then (∀x)P and (∃x)P are sentences
- A **well-formed formula** (**wff**) is a sentence containing no "free" variables. i.e., all variables are "bound" by universal or existential quantifiers.

  (∀x)P(x,y) has x bound as a universally quantified variable, but y is free.

Translating English to FOL–Examples

- **Every gardener likes the sun.**
  (∀x) gardener(x) => likes(x,Sun)
- **Not Every gardener likes the sun.**
  ~((∀x) gardener(x) => likes(x,Sun))
- **You can fool some of the people all of the time.**
  (∃x)(∀t) person(x) ^ time(t) => can-be-fooled(x,t)
- **You can fool all of the people some of the time.**
  (∀x)(∃t) person(x) ^ time(t) => can-be-fooled(x,t)
  (the time people are fooled may be different)
- **You can fool all of the people at some time.**
  (∃t)(∀x) person(x) ^ time(t) => can-be-fooled(x,t)
  (all people are fooled at the same time)
- **You can not fool all of the people all of the time.**
  ~((∀x)(∀t) person(x) ^ time(t) => can-be-fooled(x,t))
- **Everyone is younger than his father**
  (∀x) person(x) => younger(x, father(x))

Examples-2

- **All purple mushrooms are poisonous.**
  (∀x) (mushroom(x) ^ purple(x)) => poisonous(x)
- **No purple mushroom is poisonous.**
  ~(∃x) purple(x) ^ mushroom(x) ^ poisonous(x)
  (∀x) (mushroom(x) ^ purple(x)) => ~poisonous(x)
- **There are exactly two purple mushrooms.**
  (∃x)(Ey) mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ~(x=y) ^
  (∀z) (mushroom(z) ^ purple(z)) => ((x=z) v (y=z))
- **Clinton is not tall.**
  ~tall(Clinton)
- **X is above Y if X is directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.**
  (∀x)(∀y) above(x,y) <=> (on(x,y) v (∃z) (on(x,z) ^ above(z,y)))

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there are infinite number of interpretations because |M| is infinite
- **Define semantics of logical connectives: ~, ^, v, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

Connections between All and Exists

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

$(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$

$\neg(\forall x)\ P \leftrightarrow (\exists x)\ \neg P(x)$

$(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$

$(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$

Quantified inference rules

- Universal instantiation
  - $\forall x\ P(x)\ \therefore\ P(A)$
- Universal generalization
  - $P(A) \wedge P(B) \ldots\ \therefore\ \forall x\ P(x)$
- Existential instantiation
  - $\exists x\ P(x)\ \therefore P(F)$ $\leftarrow$ skolem constant F
- Existential generalization
  - $P(A)\ \therefore\ \exists x\ P(x)$

Universal instantiation (a.k.a. universal elimination)

- If $(\forall x)\ P(x)$ is true, then P(C) is true, where C is *any* constant in the domain of x
- Example:

  $(\forall x)$ eats(Nevin, x) $\Rightarrow$ eats(Nevin, IceCream)
- The variable symbol can be replaced by any ground term, i.e., any constant symbol or function symbol applied to ground terms only

Existential instantiation (a.k.a. existential elimination)

- From (∃x) P(x) infer P(c)
- Example:
  - (∃x) eats(Garland, x) → eats(Garland, Stuff)
- Note that the variable is replaced by a **brand-new constant** not occurring in this or any other sentence in the KB
- Also known as skolemization; constant is a **skolem constant**
- In other words, we don’t want to accidentally draw other inferences about it by introducing the constant
- Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

Existential generalization (a.k.a. existential introduction)

- If P(c) is true, then (∃x) P(x) is inferred.
- Example

  eats(Nevin, IceCream) ⇒ (∃x) eats(Nevin, x)
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - (∀x,y) parent(x, y) ↔ child (y, x)
    (∀x,y) father(x, y) ↔ parent(x, y) ∧ male(x) (similarly for mother(x, y))
    (∀x,y) daughter(x, y) ↔ child(x, y) ∧ female(x) (similarly for son(x, y))
  - (∀x,y) husband(x, y) ↔ spouse(x, y) ∧ male(x) (similarly for wife(x, y))
    (∀x,y) spouse(x, y) ↔ spouse(y, x) (**spouse relation is symmetric**)
  - (∀x,y) parent(x, y) → ancestor(x, y)
    (∀x,y)(∃z) parent(x, z) ∧ ancestor(z, y) → ancestor(x, y)
  - (∀x,y) descendant(x, y) ↔ ancestor(y, x)
  - (∀x,y)(∃z) ancestor(z, x) ∧ ancestor(z, y) → relative(x, y)
    (related by common ancestry)
    (∀x,y) spouse(x, y) → relative(x, y) (related by marriage)
    (∀x,y)(∃z) relative(z, x) ∧ relative(z, y) → relative(x, y) (**transitive**)
    (∀x,y) relative(x, y) ↔ relative(y, x) (**symmetric**)
- **Queries**
  - ancestor(Jack, Fred) /* the answer is yes */
  - relative(Liz, Joe) /* the answer is yes */
  - relative(Nancy, Matthew)
    /* no answer in general, no if under closed world assumption */
  - (∃z) ancestor(z, Fred) ∧ ancestor(z, Liz)

Higher Order Logic

Higher order logic

- FOL only allows to quantify over variables, and variables can only range over objects.
- HOL allows us to quantify over relations
- Example: (quantify over functions)

  "two functions are equal iff they produce the same value for all arguments"

  $\forall f\ \forall g\ (f = g) <=> (\forall x\ f(x) = g(x))$
- Example: (quantify over predicates)

  $\forall r$ transitive(r) $=> (\forall xyz)\ r(x,y) \wedge r(y,z) => r(x,z))$
- More expressive, but undecidable.

Other KR Schemes

Production Rules

- The human mental process is internal, and it is too complex to be represented as an algorithm. However, most experts are capable of expressing their knowledge in the form of **rules** for problem solving.

IF	the 'traffic light' is green
THEN	the action is go
IF	the 'traffic light' is red
THEN	the action is stop

- Two types of production rules exist:
  - Inference-type production rule
    - Both antecedents and consequents are assertions about data in STM.
  - Situation action rules
    - Antecedent is a logical combination of assertions about the data in STM and consequent is a collection of actions that change STM.

Production Rules (2)

- The term **rule** in AI, which is the most commonly used type of knowledge representation, can be defined as an IF-THEN structure that relates given information or facts in the IF part to some action in the THEN part.
- A rule provides some description of how to solve a problem. Rules are relatively easy to create and understand.
- Any rule consists of two parts:
  - the IF part, called the **antecedent** (**premise** or **condition**)
  - and the THEN part called the **consequent** (**conclusion** or **action**).
- A rule can have multiple antecedents joined by the keywords AND (conjunction), OR (disjunction) or a combination of both.

Production Rules (3)

- Rules can represent relations, recommendations, directives, strategies and heuristics:
  - **Relation**

    IF the 'fuel tank' is empty THEN the car is dead
  - **Recommendation**

    IF the season is autumn AND the sky is cloudy AND the forecast is drizzle THEN the advice is 'take an umbrella'
  - **Directive**

    IF the car is dead AND the 'fuel tank' is empty

    THEN the action is 'refuel the car'

Production Rules (3)-b

- **Strategy**

IF the car is dead THEN the action is 'check the fuel tank'; step1 is complete

IF step1 is complete AND the 'fuel tank' is full

THEN the action is 'check the battery'; step2 is complete

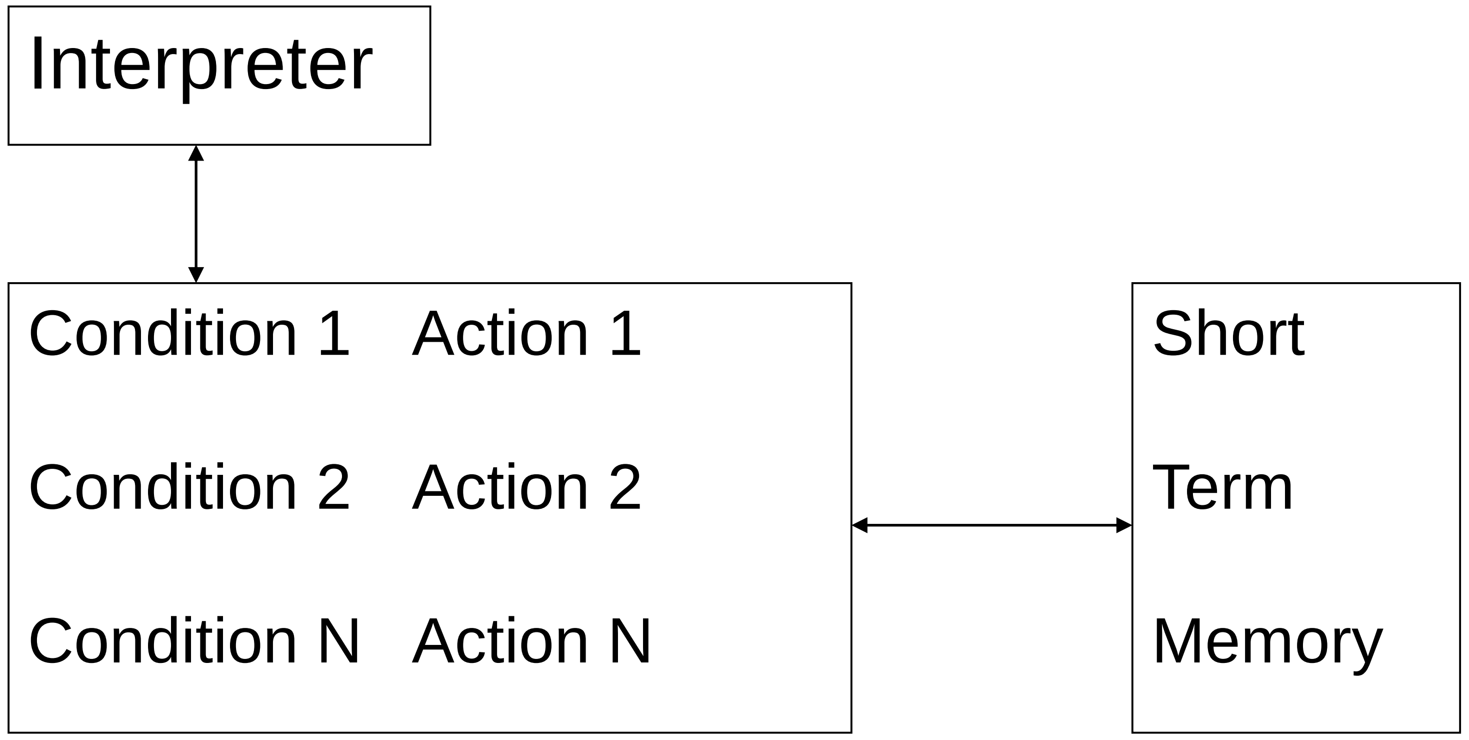
- **Heuristic**

IF the spill is liquid AND the 'spill pH' < 6 AND the 'spill smell' is vinegar THEN the 'spill material' is 'acetic acid'

Architecture of a Production System

- Consists of three (3) parts:
  - Short term memory of facts (working memory)
  - A rule base composed of a set of production rules of the form antecedent=> consequent.
    - The antecedent comprises conditions that characterize the short term memory and the consequent comprise actions that manipulate the short term memory.
    - If the antecedent of the production rule is satisfied by the short-term memory then the production rule may be executed (fired).
    - The consequent of the production rule will change the content of the short-term memory so that other rules will have their conditions satisfied.
  - An interpreter that represents a mechanism to examine the short term memory and to determine which rules to fire. The following diagram represents the architecture of the production system.

Diagram


Interpreter
Condition 1 Action 1
Condition 2 Action 2
Condition N Action N
Short
Term
Memory

Reasoning with Production Rules

- Production systems used as the basis for many rule-based expert systems
- Reasoning in:
  - forward (situation-action rules) or
  - backward chaining (situation-action and inference rules)

Reasoning with Production Rules

Basic Inference Procedure

- While changes are made to Working Memory do:
  - **Match** the current WM with the rule-base
    - Construct the Conflict Set -- the set of all possible (*rule, facts*) pairs where rule is from the rule-base, facts from WM that unify with the conditional part (i.e., LHS) of the rule.
  - **Conflict Resolution**: Instead of trying all applicable rules in the Conflict set, select one from the Conflict Set for execution. (**depth-first**)
  - **Act/fire**: Execute the actions associated with the conclusion part of the selected rule, after making variable substitutions determined by unification during match phase
  - **Stop** when conflict resolution fails to returns any (rule, facts) pair

Conflict Resolution Strategies

- **Refraction**
  - A rule can only be used once with the same set of facts in WM. This strategy prevents firing a single rule with the same facts over and over again (avoiding loops)
- **Recency**
- Use rules that match the facts that were added most recently to WM, providing a kind of "focus of attention" strategy.
- **Specificity**
  - Use the most specific rule, if one rule's LHS is a superset of the LHS of a second rule, then the first one is more specific
  - If one rule's LHS implies the LHS of a second rule, then the first one is more specific
- **Explicit priorities**
  - E.g., select rules by their pre-defined order/priority
- **Rule order**

Comparing Production Systems (PS) and FOL

- Advantages
  - Simplicity (both KR language and inference),
  - Inference more efficient
  - Modularity of knowledge (rules are considered, to a degree, independent of each other), easy to maintain and update
  - Similar to the way humans express their knowledge in many domains
  - Can handle simple default reasoning
- Disadvantages
  - No clearly defined semantics (may derive incorrect conclusions)
  - Inference is not complete (mainly due to the depth-first procedure)
  - Inference is sensitive to rule order, which may have unpredictable side effects
  - Less expressive (may not be suitable to some applications)
- ***No explicit structure among pieces of knowledge in BOTH FOL** (a un-ordered set of clauses) **and PS** (a list of rules)*

Semantic Networks

- A semantic network is a structure for representing knowledge as a pattern of interconnected **nodes** and **arcs**. Nodes in the net represent **concepts** of entities, attributes, events, values. Arcs in the network represent **relationships** that hold between the concepts.
  - A semantic network is a graph theoretic data structure whose nodes represent **word senses** and whose arcs express **semantic relationships** between these word senses.
  - Concepts can be represented as hierarchies of inter-connected concept nodes (e.g. animal, bird, canary)
  - Any concept has a number of associated attributes at a given level (e.g. animal --> has skin; eats etc.)
- Some concept nodes are **super class** of other nodes (e.g. animal > bird) and some are **sub classes** (canary < bird)
  - Often, sub classes **inherit** <u>all</u> the attributes of their super class concepts.
  - Some instances of a concept are **exempted** from the having certain attributes held by their concept classes (e.g. ostrich is excepted from flying)

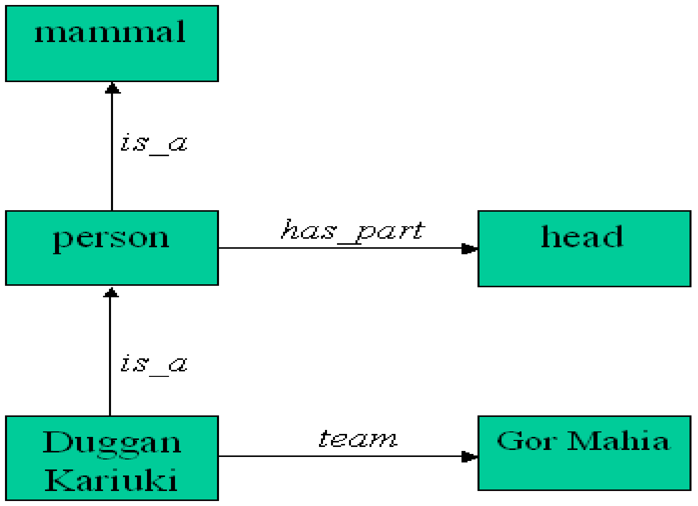
Semantic Networks– Exercises

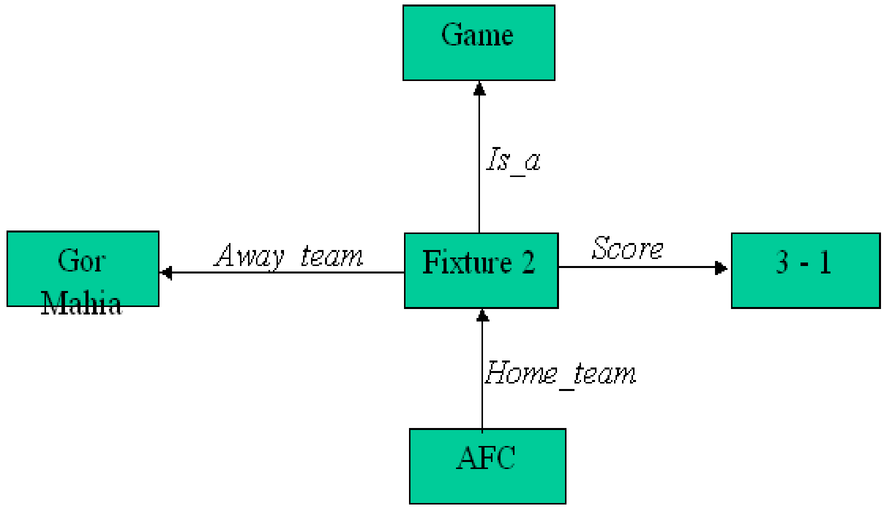
- Try to represent the following sentences using an appropriate semantic network diagram:
  - isa(person, mammal)
  - instance(Duggan Kariuki, person)
  - team(Duggan Kariuki, Cardiff) all in one graph

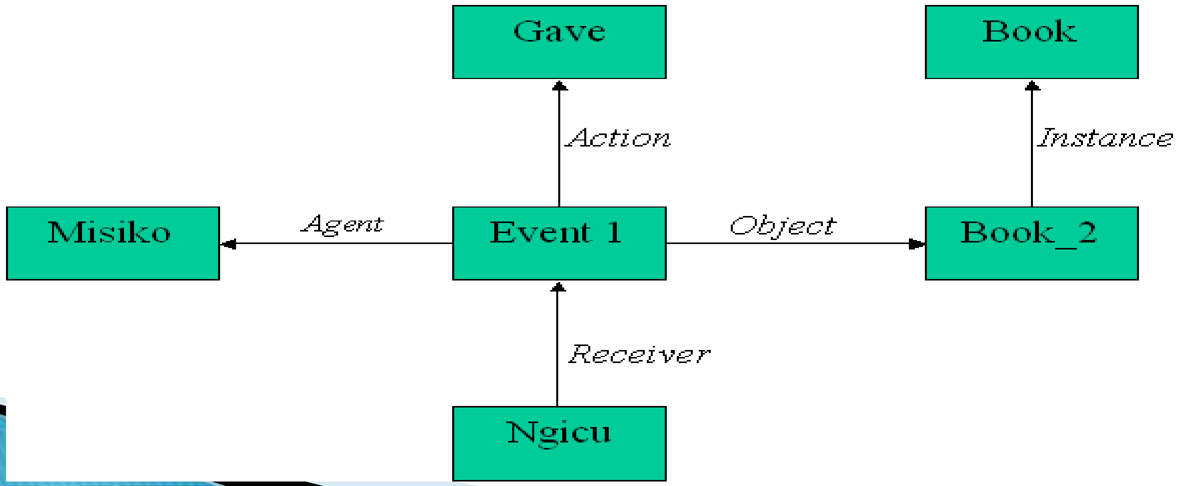
  - score(Gor Mahia, AFC, 3-1)

  - Misiko gave Ngicu the book

Semantic Networks– Solution







Advantages of Semantic Networks

Advantages

1) Easy to visualise and understand.
2) The knowledge engineer can arbitrarily define the relationships.
3) Related knowledge is easily categorised.
4) Efficient in space requirements.
5) Node objects represented only once.
6) Standard definitions of semantic networks have been developed.

Disadvantages of Semantic Networks

Limitations

The limitations of conventional semantic networks were studied extensively by a number of workers in AI:

1) Many believe that the basic notion is a powerful one and has to be complemented by, for example, **logic** to improve the notion's expressive power and robustness.
2) Others believe that the notion of semantic networks can be improved by incorporating **reasoning** used to describe events.
3) Limited in handling quantifiers e.g. "Every dog has bitten a postman"
4) Binary relations are usually easy to represent, but some times is difficult.e.g. try to represent the sentence:
    - "Nengo caused trouble to the party".

Frames

- stereotypical information on objects are represented- capture object attributes and then values.
- A * indicates attribute is only true of a typical member of class and not necessarily every number.
- Slots and slots values may also be frames.

Person
  isa: Mammal
  Cardinality:

Frames2

Adult-Male
  isa: Person
  Cardinality:

Football-Player
  isa: Adult-Male
  Cardinality:
  Height:
  Weight:
  Position:
  Team:
  Team-Colours:

Striker
  isa: Football-Player
  Cardinality:
  Goals:

Frames-3

- *Onditi*
  - *instance: Striker*
  - *Height: 6'0"*
  - *Position: Centre*
  - *Team: Gor Mahia*
  - *Team-Colours: Green/White*
- *Football-Team*
  - *isa: Team*
  - *Cardinality:*
  - *Team-size: 11*
  - *Coach:*
- *Gor Mahia*
  - *Instance:Foot-ball Team*
  - *Team size: 11*
  - *Coach: Haggai Nyang'*
  - *Players: {Onditi, Nyang, Jude, Mutero, Akidiva, Baraza,Patricia, Liz,…}*