Knowledge Representation

CompSci 367
Assoc. Prof. Ian Watson

Knowledge Representation

- Types of Knowledge
- Rules
- Frames
- Objects
- Semantic Nets
- Conceptual Graphs
- Scripts
- Cases

The AI Wars

- Isn't logic programming better than any inherently less formal approach?
- Are rule-based systems better/worse than frame-based systems?
- Are frame systems better than more general semantic networks?
- Are frames useful if there is no associated description logic?
- Who wants a K representation that may not allow for decidable inferencing?

Allen Newell's AAAI Presidential Address (1980)

- We should stop bickering about representation
- What really matters is the knowledge that a system has, not how that knowledge is represented
- Knowledge is what an observer attributes to an agent to allow the observer to call that agent intelligent

The important distinction

- Knowledge representations are symbols such that, when some process is applied, an observer attributes intelligence to the emergent behavior
- Knowledge representations acquire meaning only when there is some process that is applied to them
- representations are symbols that must be interpreted

The important distinction

- Knowledge is a label for intelligent behaviour
- Knowledge is inferred by observing an agent's behaviours
- knowledge ultimately is something that is experienced and attributed
- Not something that exists in isolation

An analogy:

- The notes are not the music

Knowledge has some structure

- The goals that an agent has
- The set of actions of which an agent is capable
- How the agent selects actions to help it achieve its goals

The “Knowledge Level”

(Alan Newell, 1982 The AI Journal)

- Computer systems can be viewed at discrete, hierarchical levels, where each level consists of
  - A medium that is processed
  - Components that provide primitive processing
  - Laws of composition
  - Laws of behaviour
  - Each level can be defined either
  - Autonomously or
  - In terms of the components of the level below it

A hierarchy of computer-system levels

- Knowledge Level
  - Intelligent behaviour
- Symbol level
  - Knowledge representations
- Hardware level
  - Machine-level instructions

The Symbol Level

- Systems: Computer programs
- Medium: Symbols, expressions
- Components: Memory stores, operations
- Behaviour laws: Sequential interpretation

The Knowledge Level

- Systems: Agents
- Medium: Knowledge
- Components: Goals, actions, bodies of knowledge
- Behaviour laws: The principle of rationality
  - If an agent has the knowledge to obtain a goal it will do so

Newell's thoughts on the knowledge level

- Knowledge and rationality are intimately tied together
- Splitting what was once a single level into two allows each one to be addressed separately
- Knowledge is not representable by a structure at the symbol level - it requires both structures and processes
- Knowledge is an abstraction that can never be had in hand

Knowledge-level analysis

- Ability to understand intelligent behaviour in terms of
  - Goals
  - Actions
  - Bodies of knowledge
- Makes the underlying knowledge representation irrelevant

Rules

- IF "A" THEN "B"
- called IF-THEN Rules
- or Antecedent/Consequent Rules
- or Production Rules

Rules

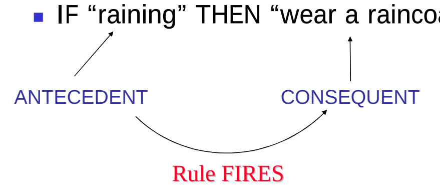
- Expert Systems built using rules are called:
- Rule-Based Systems
- Knowledge-Based Systems
- or Production Systems

Rules

- Rules are good at representing Heuristics
- IF "raining" THEN "wear a raincoat"
- IF "animal is warm blooded" and "has pouch" THEN "animal is marsupial"
- Rules can be understood by "real" people
- not just programmers

Rules

- IF "raining" THEN "wear a raincoat"

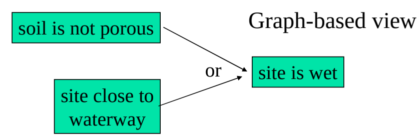


Rules

- Rules can chain together
- IF “A” THEN “B”
- IF “B” THEN “C”
- IF “C” THEN “X”

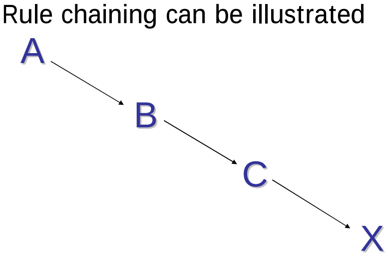
2 Views of Rules

- Language-based view
  - IF (soil is not porous)
    OR (site is close to waterway)
    THEN (site is wet)

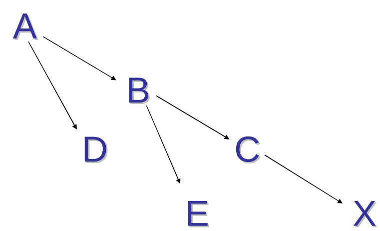


Rules

- Rule chaining can be illustrated



Rules



Rules

- IF “A” THEN “B”
- IF “A” THEN “D”
- IF “B” THEN “C”
- IF “B” THEN “E”
- IF “C” THEN “X”

The Search Space

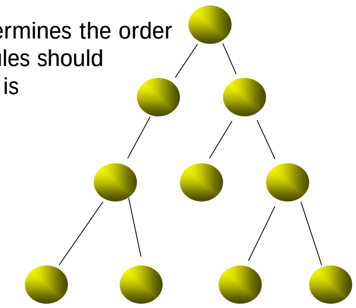
- the *search space* is a way of thinking about how rules chain together
- IF “A” THEN “B”
- IF “B” THEN “C”
- IF “B” THEN “E”
- which of the IF “B” rules should fire first?

Inferencing Methods

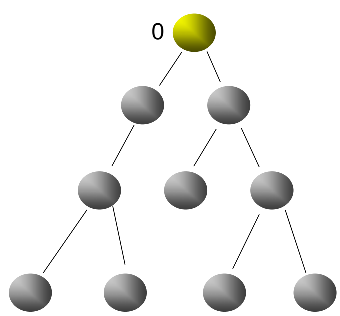
- Inferencing methods determine how search is controlled
- DEPTH FIRST
- BREADTH FIRST
- BEST FIRST

Depth First Search

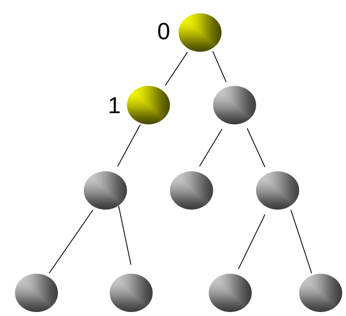
- Search determines the order in which rules should fire if there is a choice



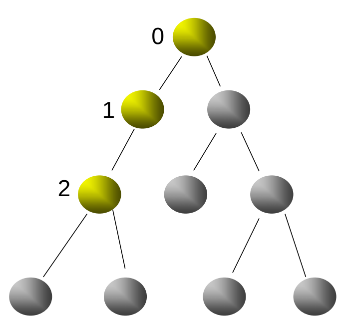
Depth First Search



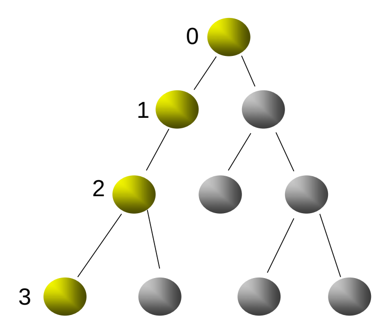
Depth First Search



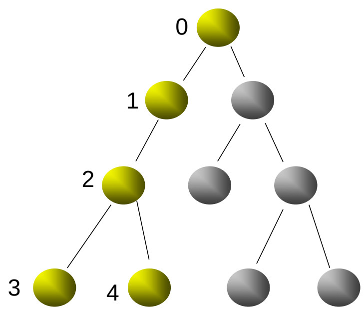
Depth First Search



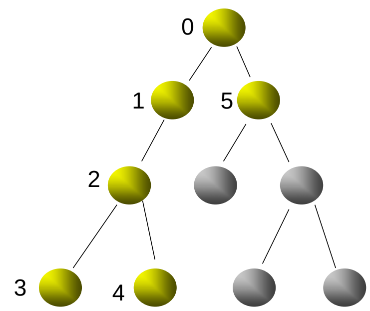
Depth First Search



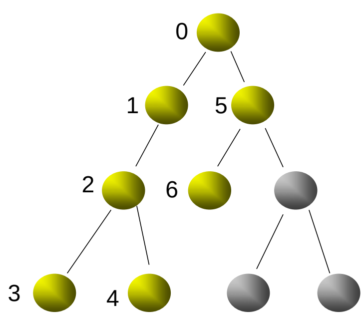
Depth First Search



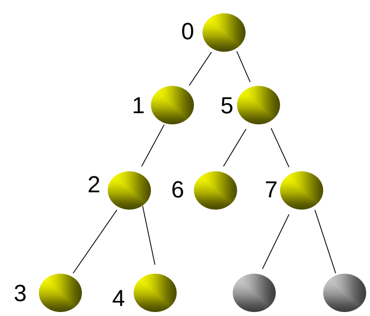
Depth First Search



Depth First Search



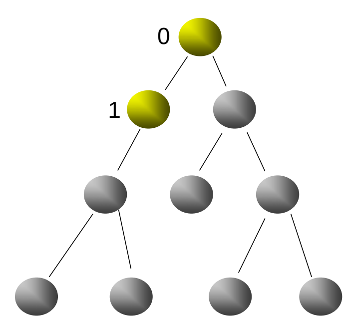
Depth First Search



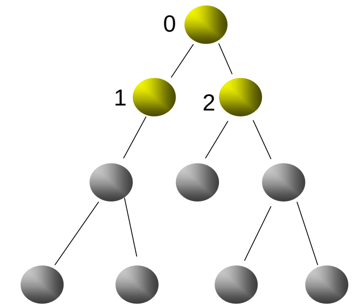
Depth First Search

- ADVANTAGES
- logical ordering of questions, program seems to follow a single line of reasoning
- computationally efficient
- DISADVANTAGES
- inefficient if large number of alternative solutions - long winded

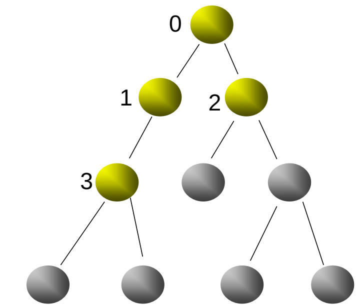
Breadth First Search



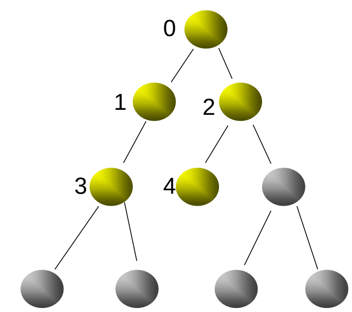
Breadth First Search



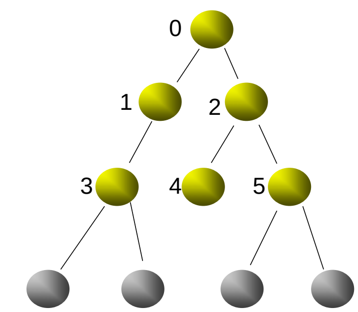
Breadth First Search



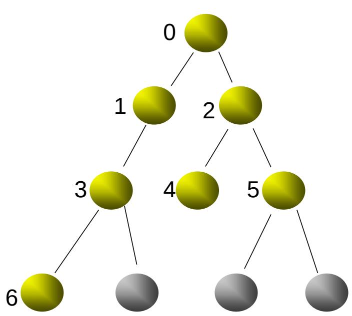
Breadth First Search



Breadth First Search



Breadth First Search

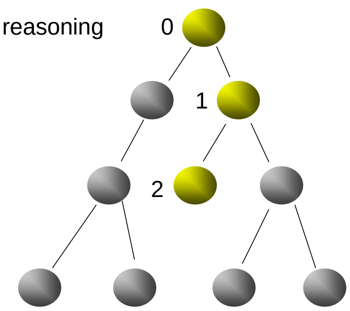


Breadth First Search

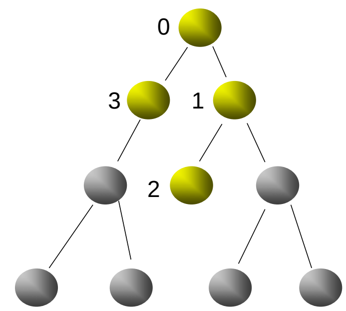
- ADVANTAGES
- efficient if many alternatives & search space is not deep
- DISADVANTAGES
- illogical ordering of questions, program is considering many hypotheses at once
- computationally inefficient

Best First Search

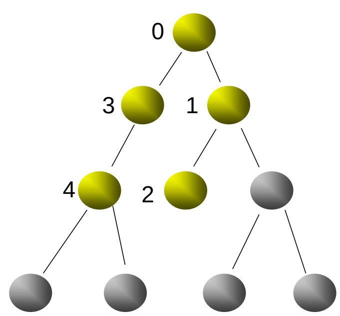
mixed mode reasoning



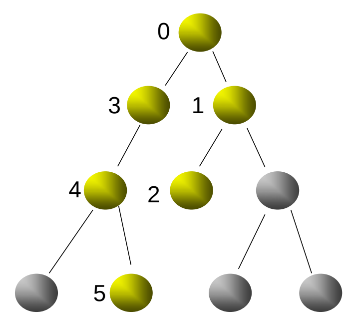
Best First Search



Best First Search



Best First Search



Best First Search

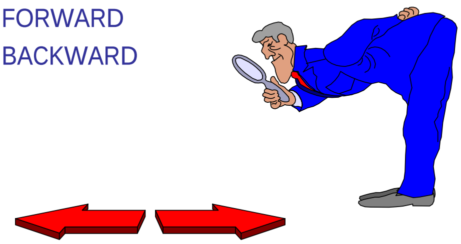
- ADVANTAGES
- flexible
- mimics human reasoning - uses domain knowledge to guide search
- DISADVANTAGES
- complex to control - needs meta-rules
- inefficient

Meta-Rules

- knowledge about knowledge
- needs knowledge of “context”
- IF context = “Z” THEN DEPTH FIRST
- IF context = “Y” THEN BREADTH FIRST
- provides control over how rules are used
- meta-rules, meta-meta-rules, meta-meta-...

Search Direction

- search can be applied in two directions
- FORWARD
- BACKWARD

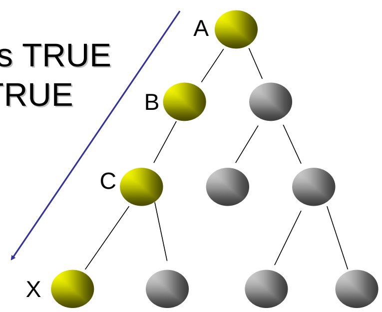


Forward Chaining

- infers NEW facts from known facts
- IF “A” THEN “B” THEN “C” THEN “X”
- DATA DRIVEN INFERENCING

Forward Chaining

IF “A” is TRUE
“X” is TRUE

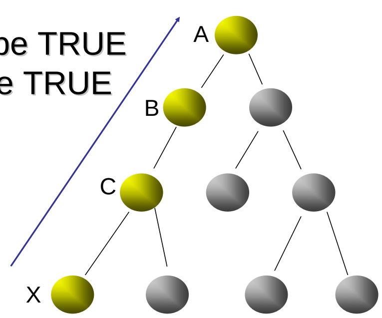


Backward Chaining

- determines what needs to be true to support a hypothesis
- IF “A” THEN “B” THEN “C” THEN “X”
- GOAL DRIVEN INFERENCING

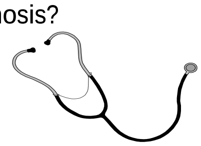
Backward Chaining

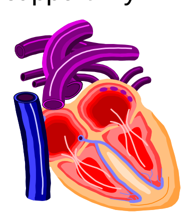
For “X” to be TRUE
“A” must be TRUE



Backward Chaining

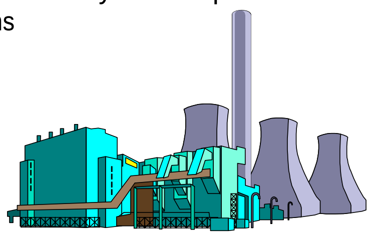
- commonly used in diagnosis
- what needs to be true to support my diagnosis?





Forward Chaining

- often used to see what effect incoming information may have in process control systems



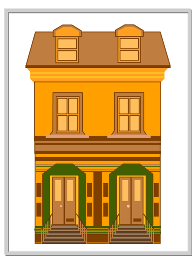
Frames

- Psychology suggests knowledge chunking
- knowledge is not always inferred
- we use stereotypes and expectations to understand and reason
- FRAMES (schemas, scripts)
  - richer data structures

© University of Auckland www.cs.auckland.ac.nz/~ian/ ian@cs.auckland.ac.nz

Frames

- organise the properties/features of a thing or event into a prototype
- FRAMES contain
  - slots and fillers
- FRAME: House
  - walls:
  - roof:
  - door:
  - windows:



© University of Auckland www.cs.auckland.ac.nz/~ian/ ian@cs.auckland.ac.nz

Frames

- describe default or typical knowledge
- FRAME: House
  - walls: brick
  - roof: tile
  - doors: wooden
  - windows: wooden [6]
- slots can contain data & processes

© University of Auckland www.cs.auckland.ac.nz/~ian/ ian@cs.auckland.ac.nz

Frames

- can be instantiated to specific individuals
- FRAME: House: My_House
  - walls: brick*
  - roof: shingles
  - doors: wooden*
  - windows: aluminum

* = inherited

© University of Auckland www.cs.auckland.ac.nz/~ian/ ian@cs.auckland.ac.nz

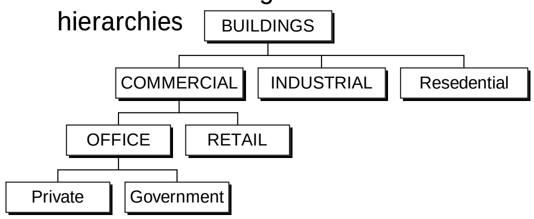
Frames

- help us deal with uncertainty
- we assume an individual conforms to its defaults unless told otherwise
- frames are often combined with rules
- frames store data
- rules or demons perform inferences
- Leonardo, ART, KEE

© University of Auckland www.cs.auckland.ac.nz/~ian/ ian@cs.auckland.ac.nz

Hierarchies

- Frames can be organised into hierarchies



Hierarchies

- Hierarchies are organised into
- classes
- subclasses
- instances
- the hierarchies have inheritance
- Frames became OBJECTS
- Frame based systems became OO

OOPS

- objects show these properties
  - abstraction
  - inheritance
  - encapsulation
  - polymorphism

OOPS

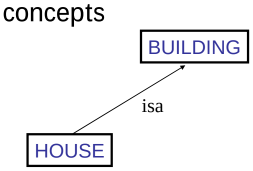
- OOPS is used in
- C++, Java etc...
- Used in AI tools: SmallTalk, Kappa, ART*Enterprise, Nexpert Object,
- these are more *dynamic* than Java
  - Objects (e.g. classes, methods) can be modified during runtime
  - Also provide a higher-level programming environments
- more suitable for AI programming

Expressiveness

- Object or Frames
  - support sharing of common properties
  - & overriding of properties - e.g. “birds fly, penguins don’t”
  - supports declarative and procedural knowledge
  - well structured knowledge bases
  - no guarantees of truth
  - requires discipline of the programmer

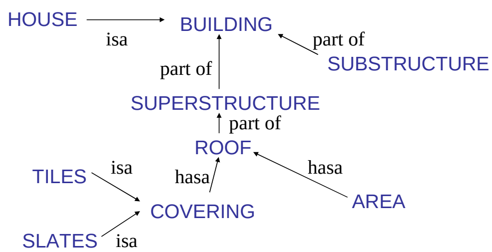
Semantic Nets

- Graphs made up of vertices and edges
- vertices describe entities/concepts
- labeled edges describe relationships between concepts



Semantic Nets

- can define complex models



Semantic Nets

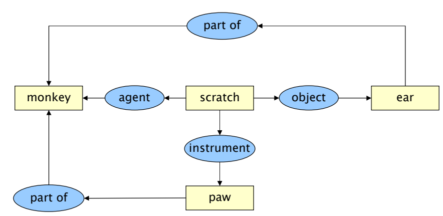
- easily represented in AI languages
- e.g., PROLOG
  isa(house,building).
  isa(slates,covering).
  isa(tiles,covering).
  partof(substructure,building).
  partof(superstructure,building).
  partof(roof,superstructure).
  hasa(covering,roof).
  hasa(area,roof).

Conceptual Graphs

- Formalised semantic network knowledge representation
- rooted in Sowa's association theory of meaning
- A Conceptual Graph is a finite bipartite directed graph
- each vertex is either a concept or a relation between two concepts
- each concept may represent another conceptual graph

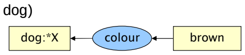
Conceptual Graphs

A monkey scratches its ear with a paw

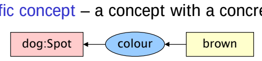


Conceptual Graphs

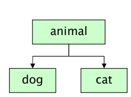
- each concept has a type
  general concept – a concept with a wildcard instance (there exists a dog)



  specific concept – a concept with a concrete instance



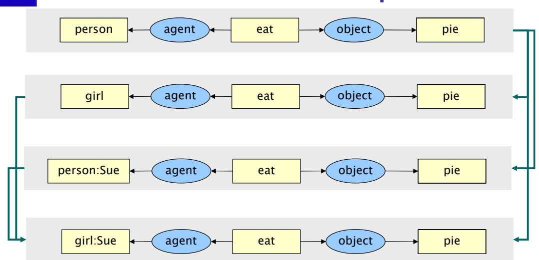
- there is a hierarchy of types & subtypes:
- concept w is specialisation of concept v if type(v)>type(w)



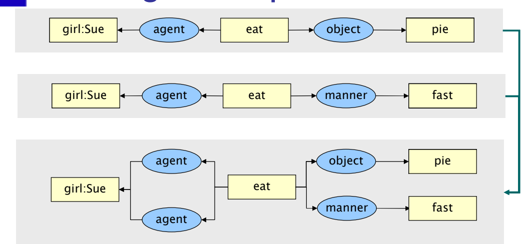
Conceptual Graphs

- canonic conceptual graph is sensible representation of knowledge that can, but does not have to, be true
- canonic formation rules formalise rules of inference between two graphs while preserving canonicity
  - copy – identical cloning of a graph
  - restriction – substituting a concept in a graph with its specialisation
  - join – joining two graphs via shared concept
  - simplification – deleting identical relations

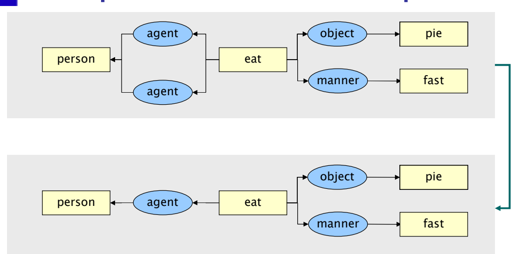
Restriction of Concepts



Joining Concepts



Simplification of Concepts



Conceptual Graphs

- FOPL transformation to CG
  - for each node → predicate
  - general concept → variable, specific concept → atom
    type:instance → type(instance)
  - relation → n-ary predicate relation(in1, in2, ..., inn) with arguments connecting neighbouring concepts
  - CG is existentionally quantified conjunction of these predicates

$\exists X\ (dog(Spot) \wedge colour(Spot,X) \wedge brown(X))$

Prolog:

dog(spot).
colour(spot,brown).