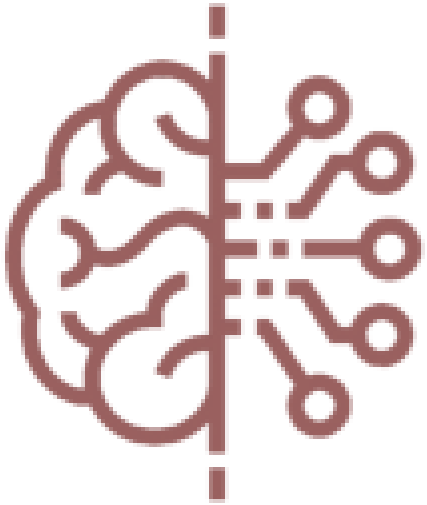
Building Control Algorithm for State Space Search



SCJ3553
Artificial Intelligence

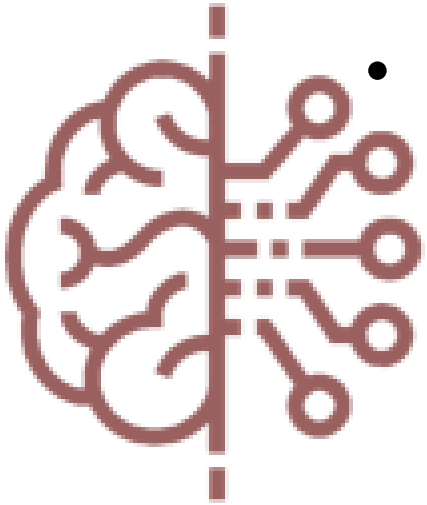
Faculty of Computing
Universiti Teknologi Malaysia

Outline

- Recursion-based system
- Production Systems
- Control of Search in Production System
- Blackboard architecture

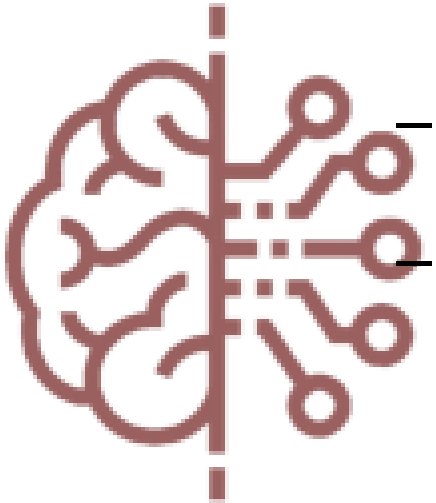
Recursion-based system

- Important tool for programming graph search
- A recursive procedure consists of:
  - Recursive step-procedure calls itself
  - Terminating condition that stops the procedure from recurring endlessly
- A natural control construct for data structures
  - i.e. lists, trees and graph



Recursion-based system

- Pattern-based Searching Technique
  - The major advantage of using general methods such as unification and modus ponens to generate states is the resulting algorithm may search any space of logical inferences.
  - The specifics of a problem are described using predicate calculus assertions.
  - Thus, we have a means of separating problem-solving knowledge from its control and implementation.
  - One example of implementation of the separation of knowledge and control is pattern-search.



Recursion-based system

- Pattern-based Searching Technique

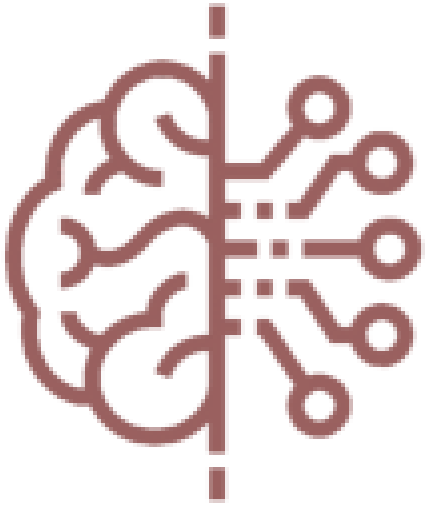
  - Recursive search is applied to a space of logical inferences
  - result in a general search procedure for predicate calculus based problem specifications
  - eg. In goal-directed search, modus ponens define the transition between states.

  - i.e. a goal p(a), the algorithm uses UNIFICATION to select the implications whose conclusions match the goal (q(x) -> p(x))
  - Implications are simply called rules
  - After unification, rule premise becomes new subgoal q(a)
  - The algo recurs on the subgoal
  - If subgoal matches fact in KB, search terminates

```
function pattern_search (current_goal);

begin
  if current_goal is a member of closed                          % test for loops
    then return FAIL
    else add current_goal to closed;
  while there remain in data base unifying facts or rules do
    begin
      case
        current_goal unifies with a fact:
          return SUCCESS;
        current_goal is a conjunction (p ∧ ...):
          begin
            for each conjunct do
              call pattern_search on conjunct;
            if pattern_search succeeds for all conjuncts
              then return SUCCESS
              else return FAIL
          end;
        current_goal unifies with rule conclusion (p in q → p):
          begin
            apply goal unifying substitutions to premise (q);
            call pattern_search on premise;
            if pattern_search succeeds
              then return SUCCESS
              else return FAIL
          end;
      end;                                                       % end case
    end;
    return FAIL
  end.
```

Production Systems



- Alternative control structure for pattern-directed search
- Model of computation in implementing search algorithm and modeling human problem solving
- Provides pattern-directed control of a problem-solving process
- Consists of a set of production rules, working memory and recognise-act control cycle.
- Architecture for knowledge-based systems

Production Systems

DEFINITION

PRODUCTION SYSTEM

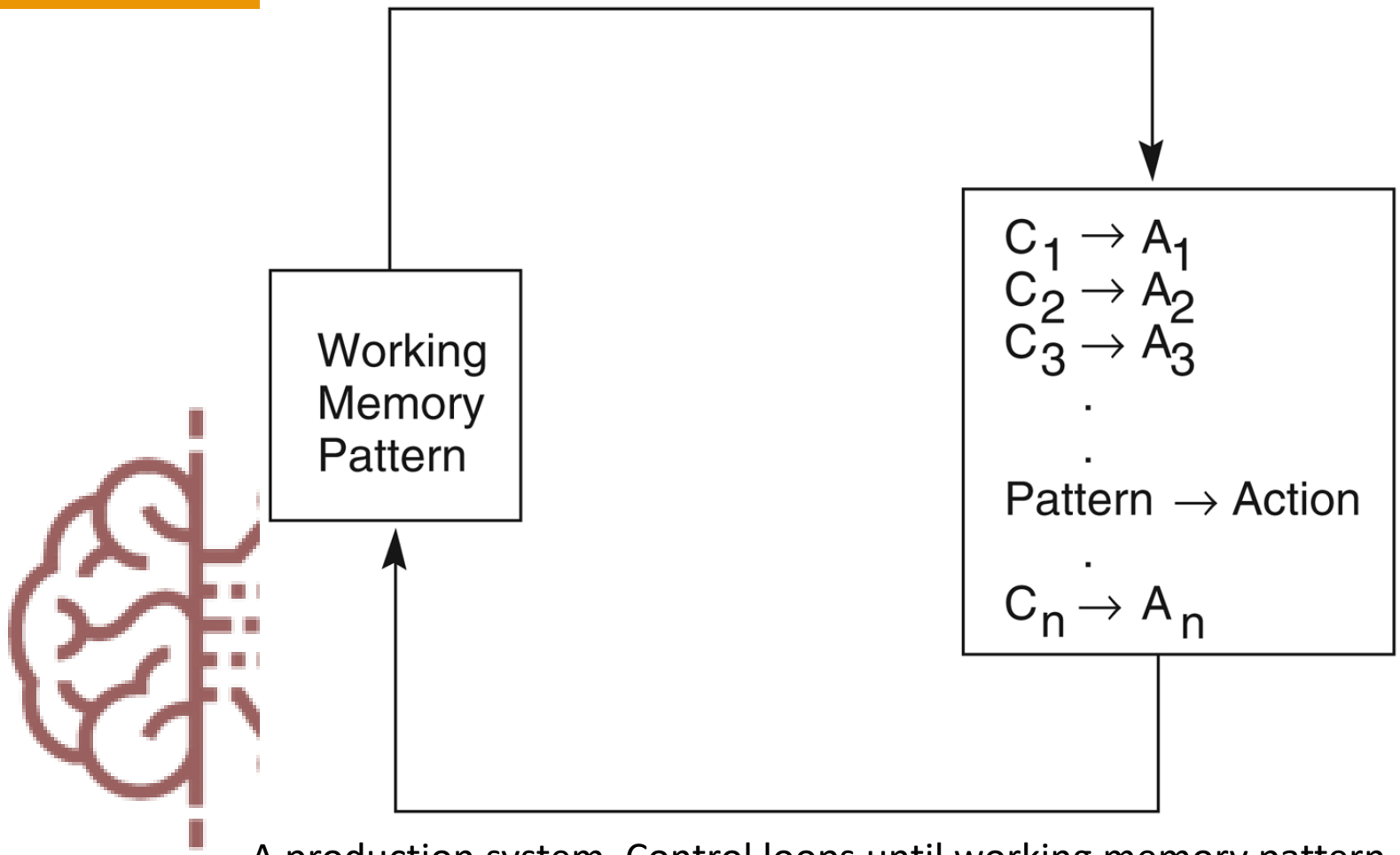
A *production system* is defined by:

1. *The set of production rules*. These are often simply called *productions*. A production is a *condition–action* pair and defines a single chunk of problem-solving knowledge.

2. *Working memory* contains a description of the *current state of the world* in a reasoning process.

3. *The recognize–act cycle*. The control structure for a production system is simple: *working memory* is initialized with the beginning problem description. The current state of the problem-solving is maintained as a set of patterns in working memory. These patterns are matched against the conditions of the production rules; this produces a subset of the production rules, called the *conflict set*, whose conditions match the patterns in working memory. The productions in the conflict set are said to be *enabled*. One of the productions in the conflict set is then selected (*conflict resolution*) and the production is *fired*.

Production Systems



A production system. Control loops until working memory pattern no longer matches the conditions of any productions.

Production Systems

Below is an example of production systems used to sort a string in ascending order. You are required to change the rules in the production set so that the system can sort the string in descending order then, do the sorting process until the string is sorted.

Production set:

1. ba → ab
2. ca → ac
3. cb → bc

- eg: A production system program for sorting a string composed of letters a, b and c
- A production is enabled if its condition matches part of string in working memory.
- When a rule is fired, the substring that matched the rule condition is replaced by the string on the right-hand side of the rule.
- Productions are invoked by the 'pattern' of a particular problem instances

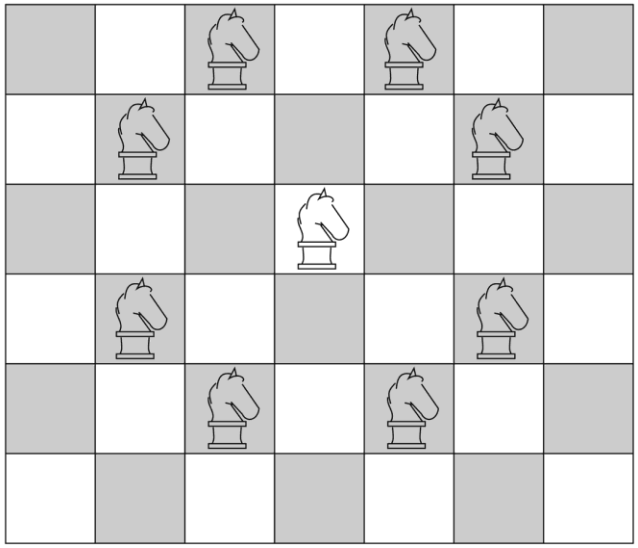
Iteration #	Working memory	Conflict set	Rule fired
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2, 3	2
3	acbac	1, 3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	Ø	Halt

Production Systems

Use production system to make legal move from 1 to 2 based on 3 x 3 knight's tour problem.

move(1,8) R1	move(6,1) R9
move(1,6) R2	move(6,7) R10
move(2,9) R3	move(7,2) R11
move(2,7) R4	move(7,6) R12
move(3,4) R5	move(8,3) R13
move(3,8) R6	move(8,1) R14
move(4,9) R7	move(9,2) R15
move(4,3) R8	move(9,4) R16

1	2	3
4	5	6
7	8	9



Production Systems

A production system solution to the 3 x 3 knight's tour problem to make legal move from 1 to 2

move(1,8) R1 move(6,1) R9
move(1,6) R2 move(6,7) R10
move(2,9) R3 move(7,2) R11
move(2,7) R4 move(7,6) R12
move(3,4) R5 move(8,3) R13
move(3,8) R6 move(8,1) R14
move(4,9) R7 move(9,2) R15
move(4,3) R8 move(9,4) R16

1	2	3
4	5	6
7	8	9

Iteration #	Working memory		Conflict set (rule #'s)	Fire rule
	Current square	Goal square		
0	1	2	1, 2	1
1	8	2	13, 14	13
2	3	2	5, 6	5
3	4	2	7, 8	7
4	9	2	15, 16	15
5	2	2		Halt

Control of Search in Production System

- Production system offers opportunities for adding heuristic control to search algorithm, including:-
  - Choice of data-driven or goal-driven strategies
  - Structure of rules
  - Choice of strategies for conflict resolution

Control of Search in Production System

Data-driven search in a production system

- Begin with problem description
- Infers new knowledge from data through rules inferencing, legal moves etc.
- Continues until goal is reached
- Productions have

CONDITION->ACTION form creating new state of graph

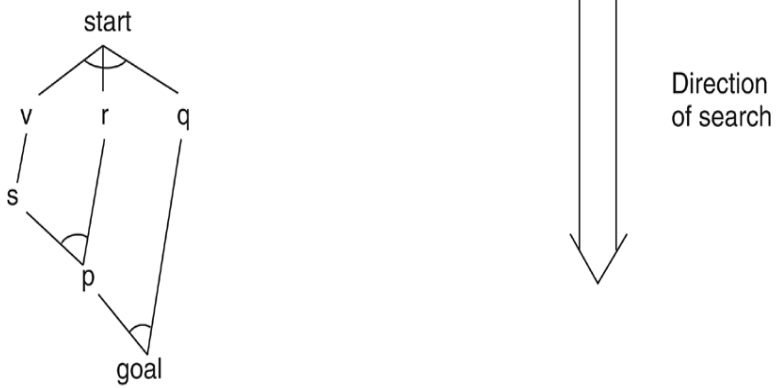
Production set:

1. $p \wedge q \rightarrow goal$
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow q$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. $start \rightarrow v \wedge r \wedge q$

Trace of execution:

Iteration #	Working memory	Conflict set	Rule fired
0	start	6	6
1	start, v, r, q	6, 5	5
2	start, v, r, q, s	6, 5, 2	2
3	start, v, r, q, s, p	6, 5, 2, 1	1
4	start, v, r, q, s, p, goal	6, 5, 2, 1	halt

Space searched by execution:



Control of Search in Production System

Goal-driven search in a production system

- Begins with a goal, work backward to facts satisfying the goals
- In production system,

-Goal is placed in working memory

ACTIONS <- CONDITIONS

- Continues until facts are found

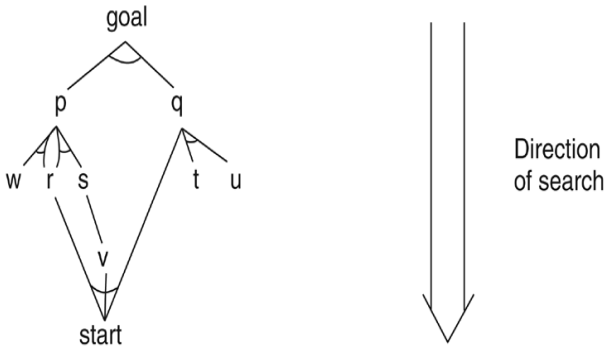
Production set:

1. $p \wedge q \rightarrow goal$
2. $r \wedge s \rightarrow p$
3. $w \wedge r \rightarrow p$
4. $t \wedge u \rightarrow q$
5. $v \rightarrow s$
6. $start \rightarrow v \wedge r \wedge q$

Trace of execution:

Iteration #	Working memory	Conflict set	Rule fired
0	goal	1	1
1	goal, p, q	1, 2, 3, 4	2
2	goal, p, q, r, s	1, 2, 3, 4, 5	3
3	goal, p, q, r, s, w	1, 2, 3, 4, 5	4
4	goal, p, q, r, s, w, t, u	1, 2, 3, 4, 5	5
5	goal, p, q, r, s, w, t, u, v	1, 2, 3, 4, 5, 6	6
6	goal, p, q, r, s, w, t, u, v, start	1, 2, 3, 4, 5, 6	halt

Space searched by execution:



Control of Search in Production System

- Structure of rules :-
  - programmer may control search through the structure (distinction between condition and action) and order of rules in production set.
  - order of premises encodes important procedural information

Control of Search in Production System

- Heuristic control through conflict resolution
  - i.e. choose first rule matches the working memory
  - By Brownston et al. 1985
    - Refraction – once a rule has fired, it may not fire again until working memory element that match its condition have been modified to discourage looping
    - Recency – choose rules whose condition match with the pattern most recently added to the working memory
    - Specificity – use a more specific problem-solving rule rather than general one

Control of Search in Production System

Major advantages of production systems for artificial intelligence

i) Separation of Knowledge and Control- control (recognize-act cyle) and knowledge in the rule, can maintain separately

ii) A Natural Mapping onto State Space Search- working memory = nodes of states, production rules=transitions between states, conflict resolution=branch

iii) Modularity of Production Rules –syntactic independence, easy adding, deleting or changing knowledge of system

iv) Pattern-Directed Control

v) Opportunities for Heuristic Control of Search

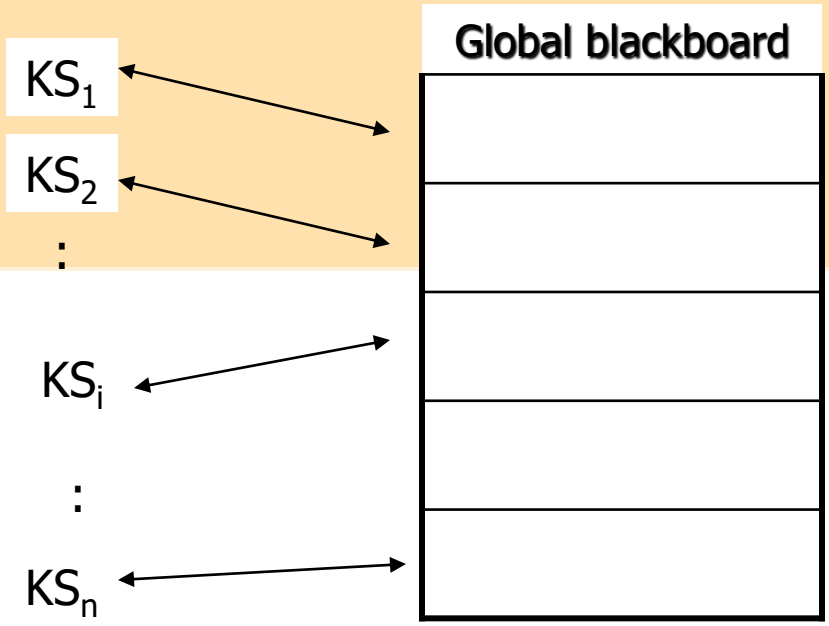
Blackboard Architecture

- Extends production systems by allowing organizing production memory into separate modules, each correspond to different set of rules
- Blackboard integrate separate sets of rules and coordinate the actions of multiple agents (knowledge source -KS) within a single global structure , blackboard
- i.e. speech understanding program
  - First, manipulate utterance represented as digitized waveform
  - Then find words in this utterance form a sentence
  - Finally, produce semantic representation of utterance meaning

Blackboard Architecture

- Blackboard architecture – model of control requires coordination of multiple processes or knowledge sources (KS).

- Blackboard- a central global database for communication of independent asynchronous knowledge sources focusing on related aspects of particular problem



Each level of analysis is processed by different class of knowledge source (KS)

-waveform is entered at lowest level

-KS for processing this entry is enabled and post interpretation to blackboard, to be picked up by appropriate process

-all processes are asynchronous and data driven

-they act when they have input data, continue acting until they have finished their task, then post their results and wait for next task

KS_1	The waveform of the acoustic signal.
KS_2	The phonemes or possible sound segments of the acoustic signal.
KS_3	The syllables that the phonemes could produce.
KS_4	The possible words as analyzed by one KS.
KS_5	The possible words as analyzed by a second KS (usually considering words from different parts of the data).
KS_6	A KS to try to generate possible word sequences.
KS_7	A KS that puts word sequences into possible phrases.