TDT4137 (Cognitive Architectures) Assignment Sheet 3

1 SOAR

1.1 General questions

> **Task 1.1**: Explain the notion of problem space in Soar.

The tasks in SOAR are represented as a collection of problem spaces. These problem spaces are made up of a state and operators that can be used to manipulate the states.

When SOAR wants to begin a task, it chooses one of these problem spaces and an initial statie in the space. Then, a single operator from the problem space are selected and applied to the agent's current state, leading to internal changes such as retrieval of knowledge, modifications or external actions. This is repeated until the task's *goal state* are reached.

> **Task 1.2**: What is an '*impasse*', and how does Soar handle the situation?

An *impasse* is a situation where when the knowledge to select or apply an operator is incomplete or uncertain. When an impasse happens, SOAR aitomatically creates a **sub-state** to the current state. A **sub-state** is a state where a temporary goal are created. This goal are concerned with retrieving or discovering knowledge, so that the decision making can continue. This process of problem solving are used recursively, which can lead to a stack of sub-states.

> **Task 1.4**: Describe the different ways Soar can learn from its problem solving experience.

It can learn in the *Reinforcement learning cycle*. The goal of this cycle is to "*learn an action-selection policy such as to maximize expected reciept of future reward.*". Reinforcement learning changes the rules in the procedural memory, that is, a type of long term memory, where knowledge are encoded as rules. So, in the reinforcement cycle, an action will be taken, and rewarded based on how good the action was.

Another way to learn are through *Semantic learning*, where it learns from capturing declarative statements (or facts) and through *Episodic learning*, where the architecture stores history of experiences.

> **Task 1.5**: Explain the 'symbol structures' that exist in Soar. Explain processes that change these symbol structures over time

SOAR operates with states and operators. The state stores information about the system's experiences, e.g through perception, and the operators consists of information on what action(s) to take when a new problem is encountered. The processes that changes these structures are for example thorugh perception to change the state, and through learning (previuosly explained) to change the operators.

1.2 Memory

> **Task 1.6**: Describe the three types of long term memory (LTM) structures in Soar and their roles in problem solving

There are three types og LTM in SOAR: Procedual, semantic and episodic. Theese three types of memory have different roles in problem solving. The procedual memory are accessed directly during the decision cycle, and the semantic and episodic memory are accessed deliberately through the creation of specific cues in the working memory,

> **Task 1.8**: How does the WM and LTM communicate / cooperate?

The working memory and long term memory work together in the sense that the contents of the working memory trigger retrieval from the LTM. That is, when the working memory percepts something, it triggers a retrieval from the long term memory to get information of the problem it's currently having. This is done through the three-phase decicion cycle. First, the WM inputs the elements it has, then the situation is elaborated though matching of the "if" parts of the rules in LTM and operators are evaluated based on their ability to be usefull in the current situation. Lastly, in the decision step, an operator are selected.

1.4 Practical

> **Task 1.10**: Write the following rule (presented in plain language) as a Soar production rule.

```
if we are in the ball-in-net problem space AND
we desire to get the ball in the net AND
the current state says the ball is not in the net
then propose an operator to apply to the current state AND
call this operator 'kick ball '
```

The rule as SOAR production rule:

```
sp {ball-in-net*propose*kick-ball
    (state <s> ^problem-space <p> ^desired <d>)
    (<p> ^name ball-in-net)
    (<d> ^ball-in-net yes)
    (<s> ^ball-in-net no)
    -->
    (<s> ^operator <o>)
    (<o> ^name kick-ball)}
```

2 ICARUS

> **Task 1.11**: Describe the conceptual inference process in ICARUS.

In the ICARUS architecture, conceptual inference is a way to interperet the environment around the system. On each cognitive cycle, ICARUS' conceptual inference module collects all percepts comming from the environment, finds consistent matches of conceptual clauses (symbolic predicates that the agent can use for this cognitive purpose) against these percepts, and infers beliefs (instances of concepts that describe the agent's situation) that correspond to these rules' heads and adds them to the belief memory.

> **Task 1.13**: How can ICARUS learn from its problem solving behavior?

ICARUS learns by previuosly solved problems. It usually achieves a goal by chaining off a skill clause. Whenever a new goal are reached, the skill are added to the skill clause as a subskill. The next time the system encounteres the same problem, it has the skill to reach the goal in the skill clause.

> **Task 1.15**: In ICARUS, the LISP programming language is used to create concepts, rules, and beliefs. LISP uses something that is known as prefix notation, explain what this is.

Prefix notation in LIPSP are also known as Cambridge Polish Notation. It means that the operator are put in front of the operands, rather than between them. For example:

`1+2*3+3+4` is the same as `( + (+ 1 ( * 2 3) ) 4 )` which can be interpreted as `((2*3) + 1) + 4`. (You start with the inner parentheses).

> **Task 1.16**: Provide at least one example of a syntactically valid clause in LISP which expresses a reasonable statement related to the real world which is not appearing in the lecture slides or the recommended papers.

Example of a valid LISP clause for calculating fibonacci numbers:

```
(defun fibonacci (N)
  (if (or (zerop N) (= N 1))
      1
    (+ (fibonacci (- N 1)) (fibonacci (- N 2)))))
```

3 ACT-R

> **Task 1.17**: How does ACT-R relate to the concept of symbolic vs. sub-symbolic cognitive architectures?

ACT-Rs production system is symbolic and it has a subsymbolic structure that is a set of processes that can be summarized by a number of mathematical equations. These subsymbolic equations control many of the symbolic processes, such as estimating a relative cost and benefit associated with different productions rules.

> **Task 1.19**: How does learning work in ACT-R?

Learning in ACT-R works in two different ways. Instance learning and utility learning.

Instance learning is learning by retrieving old experiences from the memory and *utility learning* is learning which of multiple available stratigies is optimal, by keeping track of costs and probabilty of success.

In instance learning, the architecture makes use of so called *chunks*. These chunks contain facts, such as "Oslo is the capital of Norway" or "1+1=2". When encountering a problem, a chunk that fits the current situation, and all related chunks are activated. Later, the odds of this chunk is chosen is higher than before. That way, the architecture has learned.

In utility learning, the procedual memory is used. The procedual memory contains productions rules and it is these that are retrieved when encountering a new problem. The probability of reaching the goal by choosing production rule P_i are given by $P_i = \frac{Successes}{Successes + failures}$. This means that every time a situation occurs, the

number of $successes$ and $failures$ will change, and the architecture has then learned. (The value of P) will change due to the change in $successes$ and $failures$

> **Task 1.20**: Describe how ACT-R retrieves relevant information in its cognition process.

Retrieval in ACR-R is based on different formulas. These formulas calculates the probability of success if this chunk is activated, or if this production rule are chosen. The formulas retrieves information based on how recently the chunk or production rule was used, and the correlation between the current situation and the information it has decided to retrieve.