Contents

1 Introduction — Soar **2**

2 What is a unified theory of cognition? **3**

3 Is Soar *the* unified theory of cognition? **3**

4 The hypotheses behind Soar **4**

5 How does Soar work? Problem spaces. **4**
5.1 Problem spaces in natural-language understanding 5

6 How Soar works **5**

7 ACT* **6**

AI lecture 5

Jocelyn Ireson-Paine

February 14, 1996

1 Introduction — Soar

Last week I talked about the "classical" production system models. This week, the latest and most ambitious: Allen Newell's Soar. This is the culmination of Newell's interests (he died in 1992), and one can see in it the themes that were foreshadowed in his earlier work — described, for example, in *You can't play 20 questions with nature and win*, in *Visual Information Processing*, edited by Chase (PSY BC:C 39).

What's the place of Soar in AI and psychology? For a good two-page summary, see the *Soar* article in the *Encyclopaedia of AI*. Main points:

- Original goal of the research: to investigate an AI architecture that used *problem spaces* for solving all its tasks. In terms of the definition of AI that I gave in the first lecture, this would be "(b) the theoretical study of possible intelligences, and the general principles underlying them". If you believe the article, Soar wasn't intended as a psychological theory at that stage. However, it was created by two of Newell's students, so the clean separation of historical stages I've cited here may not be entirely correct.
- These aims were later broadened to
  - An integrated intelligent system. I.e. a system that can do all the things you'd expect expect of an AI system: knowledge representation, problem-solving, planning, learning, natural language understanding, ability to cope with dynamic environments. In contrast to most AI programs, which explore only one or two of these competances/mechanisms, but not whether they can be combined with the others. Cf page 305 of *20 questions*. [1]
  - A unified theory of cognition. In contrast, most psychological theories consider only a narrow range of phenomena. Cf page 284 of *20 questions*.
- Following the last point, note that several of Soar's mechanisms originate in psychology: problem spaces, production systems, chunking. (And note also that all these have been earlier interests of Newell.)
- At this stage of our knowledge, it seems there are many possible kinds of functional architecture for a general intelligence. The main assumption behind Soar is that its architecture is the best.

[1] That's one meaning of "integrated": incorporating all the themes that are commonly deemed part of AI. A second meaning is to be able to carry out a wide variety of different types of task. A third meaning is to do all of the first or second using a single underlying mechanism. Soar would appear to encompass all three meanings.

It's worth noting that other people, both from the classical and nouvelle AI sides, are beginning to realise that AI should try building integrated systems. For another example of one, see the simulated submarine described by Vere in *A Basic Agent*, AI photocopy V15, and once dubbed "the most complete artificial intelligence built to date". It's also described in a recent AI article, *Silicon Babies* in *Scientific American*, December 1991. Unlike Soar, this only satisfies the first two meanings: it doesn't have a single underlying mechanism.

- Has now been used to model many phenomena: language comprehension, visual attention, planning, attention-switching in driving, as well as "classical" puzzle problems.

  I have not looked at these models in detail, but some critics claim that they are too ad hoc. Soar is Turing-complete (see next lecture): in other words, there's a way to program it to do any task that any other programming language (or computer) could do. So, say the critics, it's certainly possible to program planning, language, etc. into Soar: but these programs do not embody enough psychological constraints to act as models of our mental processes.

2 What is a unified theory of cognition?

For a full account, see *Unified Theories of Cognition* by Newell (PSY BH:N 044). book. Many of these points, as I said, are foreshadowed in the *20 questions* paper. I find the book's style somewhat rambling: it's a lot easier to read a précis in *Behavioural and Brain Sciences* volume 15 page 425. This is followed by replies from Newell's critics.

According to Newell, a UTC is a single set of mechanisms that account for all of cognition (using the term broadly to include perception and motor control). For example, memory, problem solving, dreaming. Some of the things a UTC must explain are:

- How intelligent organisms respond flexibly to the environment.
- How they exhibit goal-directed behaviour and choose goals rationally (and in response to interrupts: see previous point).
- How they use symbols.
- How they learn from experience.

Why should psychologists want a UTC?

- The mind is a single system.
- The cognitive models we can devise are vastly underdetermined by experimental data: we need to add extra constraints. (I think what's meant by constraints here are the assumptions behind Soar's functional architecture.)
- If we have a UTC, then we can apply theoretical constructs to a much wider range of phenomena.

3 Is Soar *the* unified theory of cognition?

According to Newell in this precis, not. The point of his book is that it's time for each researcher to stop working on little tiny isolated bits of psychology that only explain a few phenomena and that have little or no theoretical content in common with the work of other researchers. See the entertainingly presented examples in *20 questions*. Newell strongly believes this: but if he just tells everybody "you must work on a UTC", they won't listen. So don't talk, act. Soar is an *example* of a UTC, but it is not intended as *the* UTC, and it may not even be a candidate. It may seem presumptuous, but as Warren McCullough said, "Don't bite my finger, look where I'm pointing".

4 The hypotheses behind Soar

See *Artificial Intelligence* volume 33 page 58 (1987). Some of these points will become clearer from the examples.

- *Physical symbol system hypothesis.* A general intelligence must be realised with a symbolic system. I described what this meant in the first lecture.
- *Goal structure hypothesis.* Control in a general intelligence is maintained by manipulating symbolic representations of goals. For a simple example, the goal-stack in Young's seriation PS, last week.
- *Uniform elementary-representation hypothesis.* There is a single elementary representation for declarative knowledge. The term *declarative* is usually opposed with *procedural*. Declarative knowledge can be inspected and decoded; procedural knowledge is "implicit", skill-type knowledge, and can only be obeyed. In one slogan, "declarative knowledge is what you read; procedural knowledge is what you run", using these words in the computing sense.
- *Problem space hypothesis.* Problem spaces are the fundamental organisational unit of all goal-directed behaviour.
- *Production system hypothesis.* Production systems are the appropriate method of encoding all long-term knowledge.
- *Universal-subgoaling hypothesis.* Any decision can be an object of goal-oriented attention.
- *Automatic-subgoaling hypothesis.* All goals are created dynamically in response to impasses and are generated automatically by the architecture. Unlike the classical production systems, where goals are created by rules, and the architecture doesn't have any special mechanisms for them.
- *Control-knowledge hypothesis.* Any decision can be controlled by indefinite amounts of knowledge, both domain- dependent and -independent.
- *Weak-method hypothesis.* The weak methods form the basic methods of intelligence.
- *Weak-method emergence hypothesis.* The weak methods arise directly from the system responding based on its knowledge of the task.
- *Uniform-learning hypothesis.* Goal-based chunking is the general learning mechanism.

5 How does Soar work? Problem spaces.

None of our references explain concisely what Soar does, and some (e.g. Winston) only explain a tiny part of it. So I've put together the following sections from a number of different sources. This was not, incidentally, particularly easy.

In the problem-space paradigm, our problem consists of a number of *states*, and we have various *operators* for converting one state into another.

Example: the eight-puzzle. This is a 3 by 3 frame which can hold eight tiles, numbered one through eight. Instead of a ninth tile, there's a gap. The problem is to move the tiles so that they end up in a specified configuration, e.g. all the numbers in order. The states are the various different configurations of the puzzle. The operators are actions like "move the gap right", "move the gap up", etc.

Example: a robot in a room with a box, e.g. the monkey and banana puzzle. The states are the relative positions of robot, box, and other objects. The operators are actions like "robot, move to box", "robot, push box", etc.

Note. Some problems can have many different ways of describing operations. E.g. in the eight-puzzle, the operations could be more specific - "move tile 9 left", etc.

5.1 Problem spaces in natural-language understanding

Taken from page 461, *Proceedings of the 13th Annual Conference of the Cognitive Science Society*. (In the Oversize section of the library.)

States are the possible mental representations one can build of sentences etc, plus some notion of what the focus of attention is.

Example of how a sentence might be comprehended:

1. Initial state: looking at sentence "John is rich".
2. Apply the operator "Attend to next word". New state now includes a focus on the word "rich".
3. Apply the operator "Look up meanings". New state now includes the following information: "(1) = has lots of money (for people); (2) = well-supplied with fat, sugar etc (of food)". It also includes the following partial mental model, from interpreting "John has": "John (person) has (some property)".
4. Apply the operator "Add a meaning to the mental model". This changes the mental model to "John (person) has (property) lots of money (of people)".
5. Apply the operator "check semantics". Are the qualifiers "(person)" and "(of people)" compatible? In this case, they are.

6 How Soar works

For an overview, see *The approach through symbols* by Newell, Young and Polk, in *The simulation of human intelligence* edited by Broadbent (PSY AD:B 078).

Soar starts off by being given a goal which determines the current context it's in. It then has to choose a problem space, an initial state, and an operator. Once it does this, it can apply the operator.

Example. We have some rooms and passages:

```
    !
A---B---D-
    !
    C
    !
```

where A, B, C, D are rooms. A is a dead-end, the others have both entrances and exits. The problem-solver (robot) starts in A and has to get to D. The states in this problem space are the robot's positions. The operators are "left", "forward", "right", "back". I'll call the problem space "maze".

So Soar starts off with the goal "get to D", and I'll assume it's chosen (or been told) to start at A and to operate in the maze problem space. It now has to choose one of the four operators, and then apply it.

Like a classical PS, Soar has an LTM of rules and an STM or working memory. Hopefully, the LTM will contain some knowledge about which operators to use.

Depending on how much Soar has learn, this could be strong knowledge (leading to a definite choice of "forward" in this case) or weak knowledge (just the names of the possible operators).

As I said, Soar first has to choose one of the operators. It does this by going through two phases. In the first, *elaborate*, it runs the rules in LTM repeatedly (just like a classical PS) until no more fire. These rules can do many things, but should eventually propose one or more operators with a *preference* attached to each. In the second phase, it enters a *decision stage*, and tries to resolve the preferences. There's a built-in algorithm to do that, and if it can't proceed (comes to an *impasse*), it automatically creates a subgoal to handle the job.

Let's assume Soar only knows the weakest possible knowledge: that all these operators "left", "forward", "right", "back" are possible. This is encoded by associating a preference with each operator in the rules in LTM:

```
Operator = left (Preference = accept) if
    ProblemSpace = maze.
Operator = forward (Preference = accept) if
    ProblemSpace = maze.
Operator = right (Preference = accept) if
    ProblemSpace = maze.
Operator = back (Preference = accept) if
    ProblemSpace = maze.
```

In the elaborate phase, Soar runs these rules and ends up with four operators, each having preference "accept". It then enters the decision stage. The preference-resolution algorithm reaches an impasse, because all the operators have the same preference. There's a good account of this on pp 173–176 of Winston 3rd edition, but note that he doesn't describe the rest of Soar.

Because Soar has reached an impasse, the *automatic subgoaling* mechanism takes over. This creates a subgoal *resolve tie*. This sets up a new context for Soar to work in, and it has to now choose an initial state, a problem space, and an operator.

Let's assume that as well as the "maze" problem space, we have another problem space which is all about selecting operators in other problem spaces. This could have its own operators such as "make random choice", "evaluate by speed", "evaluate by distance to goal", etc. Soar first chooses this as its problem space for the new context. It then chooses its initial state. This is the set of four operators proposed above. It then must pick an operator. Let's assume it chooses "evaluate by distance to goal". Each of these choices is made by entering an elaboration phase and then a decision phase. In each case, the decision phase might hit an impasse, requiring descent into yet another problem space.

Now Soar knows the initial state, problem space, and operator for the new context. It now enters another elaboration phase to apply "evaluate by distance to goal". Assume this yields "forward". The context above is now complete (it has an operator), so Soar can drop back into it and apply "forward".

As part of the automatic-subgoaling, a chunking mechanism now takes effect and creates a new rule to go into LTM. This "remembers" the context and state in which "forward" was being chosen, and states that under the same conditions, "forward" should be selected again.

7 ACT*

An earlier PS than Soar, and perhaps the first candidate for a UTC. Its architecture is quite different from both the classical PSs and from Soar. Most recent account:

Rules of the Mind by John Anderson (LEA 1993; PSY BH:A 547). It's interesting to note his comments about Soar: see the index entries.