An Introduction to Artificial Cognitive Systems

David Vernon
Institute for Cognitive Systems
Technical University of Munich

www.vernon.eu
Lecture 4

Cognitive Architectures

A Survey of Cognitive Architectures
Cognitive Architectures

1. The function and characteristics of a cognitive architecture
   - The cognitivist and emergent perspectives
   - Desirable characteristics
   - Facets of a cognitive architecture:
     + component functionality
     + component interconnectivity
     + system dynamics

2. A survey of cognitive architectures

3. Requirements for a developmental cognitive architecture

4. A case study: the iCub cognitive architecture
Example Architectures

Surveys:


A Survey of Cognitive and Agent Architectures, University of Michigan, http://ai.eecs.umich.edu/cogarch0/ (12 cognitive architectures)
Example Architectures

Surveys:

W. Duch, R. J. Oentaryo, and M. Pasquier.
“Cognitive Architectures: Where do we go from here?”,
(17 cognitive architectures)

D. Vernon, G. Metta, and G. Sandini,
"A Survey of Artificial Cognitive Systems: Implications for the Autonomous
Development of Mental Capabilities in Computational Agents",
(14 cognitive architectures)

D. Vernon, C. von Hofsten, and L. Fadiga.
"A Roadmap for Cognitive Development in Humanoid Robots",
Cognitive Systems Monographs (COSMOS), Vol. 11, Springer
Chapter 5 and Appendix I
(20 cognitive architectures)
Example Architectures

COGNITION

Cognitivist Systems

Hybrid Systems

Emergent Systems

Soar [Newell 1996]
EPIC [Kieras & Meyer 1997]
ICARUS [Langley 05, Langley 2009]
GLAIR [Shapiro & Bona 2009]
CoSy [Hawes & Wyatt 2008]
CLARION [Sun 2007]
ACT-R [Anderson et al. 2004]
KTH [2009]
ICARUS [Shanahan 2006]
CoSy [Hawes & Wyatt 2008]
PACO-PLUS [Kraft et al. 2008]
SASE [Weng 2004]
Darwin [Krichmar et al. 2005]
Cognitive Affect [2006]
Cognitive Architectures

Soar [Newell 96]
- (sitemaker.umich.edu/soar)
- Newell’s candidate UTC
- 1983 – 2005 ... (v 8.5)
- Production system
- Cyclic operation
  - Production firing (all)
  - Decision (cf preferences)
- Fine-grained knowledge representation
- Universal sub-goaling (dealing with impasse)
- General-purpose learning (encapsulates resolution of impasse)

http://cogarch.org/index.php/Soar/Architecture

(Reused from Figure 3.1, p. 20, The Soar’s User Manual, Version 6)
Cognitive Architectures

EPIC [Kieras & Meyer 97]

- Executive Process Interactive Control

- Link high-fidelity models of perception and motor mechanisms with a production system
  - Only the timing!

- Knowledge in production rules

- Perceptual-motor parameters

- All processors run in parallel
  - No learning

Copyright © 2012 David Vernon
Cognitive Architectures

ACT-R 5.0 [Anderson et al. 04]
- Adaptive Character of Thought [96]–>
- Adaptive Control of Thought-Rational [04]
- Production system
- Execute on production per cycle
  - Arbitration
- Declarative memory
  - Symbols (cf. Soar)
  - Activation values
  - Probability of reaching goal
  - Time cost of firing
  - Combined to find best trade-off
- Activation based on Bayesian analysis of probability
- Learning (‘Rational Analysis’)
  - Includes sub-symbolic:
    P(Goal), C(fire), Activation level, context association
Cognitive Architectures

ICARUS [Langley 05, Langley 06]

- Cognition is grounded in perception and action
- Concepts and skills are distinct cognitive structures
- Skill and concept hierarchies are acquired cumulatively
- Long-term memory is organized hierarchically
- LT & ST structures have a strong correspondence
- Symbolic cognitive structures are modulated with numeric functions
Cognitive Architectures

Shanahan’s Global Workspace Architecture

[Shanahan06, ShanahanBaars06, Shanahan05a, Shanahan05b]

- Anticipation and planning achieved through internal simulation
- Action selection (internal and external) mediated by affect
- Analogical representation (→ small semantic gap & easier grounding)
- Global workspace model: parallelism is a fundamental component of the architecture, not an implementation issue

---

SC Sensory Cortex
MC Motor Cortex
BG Basal Ganglia (action selection)
AC Association Cortex
Am Amygdala (affect)
Cognitive Architectures

Global workspace model: sequence of states emerge from multiple competing and cooperating parallel processes
Cognitive Architectures

Shanahan’s Global Workspace Architecture

[Shanahan06, ShanahanBaars06, Shanahan05a, Shanahan05b]

- Implemented using G-RAMS (generalized random access memories)
- Global workspace and cortical assemblies define an attractor landscape
- Perceptual categories define attractors
- Higher-order loop allows the GW to visit these attractors

SC Sensory Cortex
MC Motor Cortex
BG Basal Ganglia (action selection)
AC Association Cortex
Am Amygdala (affect)
Cognitive Architectures

Another But …

Architectures not focussed on development in the sense of the gradual acquisition of cognitive skills over an extended period

[Weng 02, Weng & Zhang 02, Weng 04a, Weng 04b]

(but also consider Anderson 04 /ACT-R 5.0)
Cognitive Architectures

AMD Autonomous Mental Development
[Weng et al. 01, Weng 02, Weng & Zhang 02, Weng 04a, Weng 04b]

Self-aware self-effecting (SASE) agent
Example Architectures

Review using 7 criteria:

Embodiment
Perception
Action
Anticipation
Adaptation
Motivation
Autonomy

[Vernon, von Hofsten, Fadiga 2010]
<table>
<thead>
<tr>
<th>Cognitive Architecture</th>
<th>Embodiment</th>
<th>Perception</th>
<th>Action</th>
<th>Anticipation</th>
<th>Adaptation</th>
<th>Motivation</th>
<th>Autonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cognitivist</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Soar</td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACT-R</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>ICARUS</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>ADAPT</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLAIR</td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>CoSy</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td><strong>Emergent</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAR</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Global Workspace</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>I-C SDAL</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SASE</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Darwin</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Cognitive-Affective</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Hybrid</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HUMANOID</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Cerebus</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cog: Theory of Mind</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kismet</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>LIDA</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>CLARION</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>PACO-PLUS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
Example Architectures

- **Cognitivist Systems**
- **Hybrid Systems**
- **Emergent Systems**

Only GLAIR addresses autonomy.
Only CoSy Architecture Schema addresses motivation.
Only ADAPT makes any strong commitment to embodiment (cf. functionalism).
Only ICARUS addresses strong adaptation (development of new models).

All address most of the 7 characteristics.
IC-SDAL, SASE, and Cognitive-Affective address all 7.

Only Global Workspace and Cognitive-Affective address anticipation in depth.
Only SASE and Cognitive-Affective address adaptation in a strong manner.
Some Architectures in More Depth
CLARION


CLARION

• Hybrid cognitive architecture
  – Symbolic representations
  – Connectionist representations

• Four sub-systems
  – ACS – Action-centred subsystem
  – NACS – Non-action-centred subsystem
  – MS – Motivational subsystem
  – MCS – meta-cognitive subsystem
All four subsystems have two levels of knowledge representation

- Implicit connectionist bottom level
- Explicit symbolic top level
- Implicit and explicit levels interact and cooperate both in action selection and in learning

Able to learn with or without a priori domain-specific knowledge

Able to learn continuously from on-going experience
CLARION

• Action-centred Subsystem (ACS)
  – Controls actions
    • External physical movements
    • Internal mental operations
CLARION

• Action-centred Subsystem (ACS)
  – Given some observational state, i.e. a set of sensory features $x$
    • The bottom level evaluates the desirability ("quality") of all possible actions
      \[ Q(x, a_1), Q(x, a_2), \ldots, Q(x, a_n) \]
    • The top level identifies possible actions from a rule network based on the input $x$ sent up from the bottom level
      \[ (b_1, b_2, \ldots, b_m) \]
CLARION

• Action-centred Subsystem (ACS)
  – The bottom-level actions $a_i$ and top-level actions $b_j$ are compared and the most appropriate top-level action $b$ is selected
  – Action $b$ is performed and the outcome is observed
    • The next state $y$ and (possibly) a reinforcement $r$ are determined
    • The $Q$ values at the bottom level are updated using the $Q$-Learning-Backpropagation algorithm
    • The top-level rules are also updated using the Rule-Extraction-Refinement algorithm
  – This process continues indefinitely
CLARION

• Action-centred Subsystem (ACS)
  – The bottom level comprises several modules of small neural networks
    • Each adapted to a distinct sensory modality or task
    • These modules can be developed by the system
      – based on experience (i.e. through ontogenesis) through trial-and-error exploration
      – or they can be specified a priori and hard-wired into the cognitive architecture (i.e. as the system phylogeny).
CLARION

• Action-centred Subsystem (ACS)
  – In the top level, explicit symbolic conceptual knowledge is captured in the form of symbolic rules
  – Explicit knowledge can be learned in several ways
    • Independent experiential hypothesis-testing learning
    • Mediation of implicit knowledge: bottom-up learning … Autonomous Generation of Explicit Conceptual Structures
CLARION

• Action-centred Subsystem (ACS)
  – The implicit bottom level & the explicit top level representations interact to effect bottom-up learning
  
  – If an action selected by the bottom level is successful
    
    • the system extracts an explicit rule that corresponds to the sensory features and the selected action
    
    • adds the rule to its top level rule network
CLARION

• Action-centred Subsystem (ACS)
  – The system subsequently verifies the extracted rule by considering the outcome of applying the rule
    • If the outcome is successful, the rule is generalized (made more universal and applicable to other situations)
    • If the outcome is unsuccessful, the rule is refined (made more specific and exclusive of the current situation)
  – i.e. autonomous generation of explicit conceptual structures by exploiting implicit knowledge acquired by trial-and-error learning
CLARION

• Action-centred Subsystem (ACS)
  – Assimilation of externally-given conceptual structures
    • Internalizing externally-provided knowledge in the form of explicit rule-based conceptual structures with existing conceptual structures at the top-level
    • Assimilating these into the bottom level implicit representation … top-down learning
CLARION

• Non-Action-centred Subsystem (NACS)
  – Maintains the system’s general knowledge
    • Implicit knowledge in connectionist form
      – Associative memory networks (mapping input to output)
    • Explicit knowledge in symbolic form
      – A network of nodes
      – Each node corresponds to an entity-specific chunk comprising
        » an entity identifier (e.g. table_1)
        » a vector of feature dimensions / feature value pairs (e.g. (size, large) ... (colour, white), (number_of_legs, 4))
CLARION

- Non-Action-centred Subsystem (NACS)
  - Maintains the system’s general knowledge
    - The feature values are represented by nodes in the bottom level associative memory
    - Chunks are linked through association rules
  - Both bottom-up and top-down learning can take place
    - Extract explicit knowledge in the top level from the implicit knowledge in the bottom level
    - Assimilate explicit knowledge of the top level into implicit knowledge in the bottom level
CLARION

• Motivational Subsystems (MS)
  
  – Provides
    
    • The drives that determines what the agent does
    
    • Evaluates the feedback
      (were the outcomes of an action satisfactory or not)
CLARION

• Motivational Subsystems (MS)
  
  – Provides the ACS with goals derived from
    
    • Low-level drives concerning physiological needs (e.g. need for food, need for water, need to avoid danger, need to avoid boredom, …)
    
    • High-level drives (e.g. desire for social approval, desire for following social norms, desire for reciprocation, desire for imitation of other people, …)
      
      – Primary hard-wired drives (cf. Maslow’s hierarchy of needs)
      
      – Secondary derived drives (changeable, acquired mostly in the process of satisfying primary drives)
CLARION

• Meta-cognitive Subsystem (MCS)
  – Monitors, regulates, and modify the overall behaviour of the cognitive system to improve cognitive performance
    • By setting goals for the action-centred subsystem
    • By setting essential parameter values the action-centred and non-action-centred subsystems
    • For example, setting reinforcement functions
    • Can be achieved by setting drive states in the motivational subsystem
  – Also comprises a top level (explicit) and bottom level (implicit)
Soar

(Based on Figure 3.1, pg 20, The Soar User Manual, Version 5)

Copyright © 2012 David Vernon
Soar


Soar

Symbolic Long-Term Memories

Procedural

Semantic

Episodic

Reinforcement Learning

Chunking

Semantic Learning

Episodic Learning

Symbolic Working Memory

Perceptual STM

LT Visual Memory

Perception

Mental Imagery

Action

Decision Procedure

Body

+ Activation

Copyright © 2012 David Vernon
Soar

• Newell’s candidate for a Unified Theory of Cognition

• Archetypal and iconic cognitivist cognitive architecture

• Production (or rule-based) system
  – Production: effectively an IF-THEN condition-action pair
  – A production system:
    • A set of production rules
    • A computational engine for interpreting or executing productions
Soar

• Operates in a cyclic manner
  – Production cycle
    • All productions that match the contents of declarative (working) memory fire
      – A production that fires may alter the state of declarative memory
      – and cause other productions to fire
    • This continues until no more productions fire.
  – Decision cycle
    • a single action from several possible actions is selected
    • The selection is based on stored action preferences.
Soar

• For each decision cycle, may have been many production cycles

• Productions in Soar are low-level
  – knowledge is encapsulated at a very small grain size
Universal sub-goaling

- There no guarantee that the action preferences will lead to
  - a unique action or
  - any action

- In this case, the decision cycle may lead to an ‘impasse’
  - Soar sets up an new state in a new problem space — sub-goaling — with the goal of resolving the impasse.
  - Resolving one impasse may cause other impasses and the sub-goaling process continues

- It is assumed that degenerate cases can be dealt with
  - e.g. if all else fails, choose randomly between two actions
Soar

- Whenever an impasse is resolved
  - Soar creates a new production rule which summarizes the processing that occurred in the sub-state in solving the sub-goal
  - Resolving an impasse alters the system super-state
    - This change is called a result
    - It becomes the outcome of the production rule
    - The condition for the production rule to fire is derived from a dependency analysis
      - finding what declarative memory items matched in the course of determining the result
  - This change in state is a form of learning
    - It is the only form that occurs in Soar
    - i.e. Soar only learns new production rules
The Cognitive-Affective Architecture Schematic
The Cognitive-Affective Architecture Schematic
The Cognitive-Affective Architecture Schematic


The Cognitive-Affective Architecture Schematic

• Explicitly addresses the role of emotion in a cognitive system

• Based on the architecture and physiology of the mammalian brain

• Referred to both as
  
  − A schema for an “Enactive Organizational Constraints Hierarchy” [Morse, Lowe, and Ziemke 2008]
  
  − “Cognitive-Affective Architecture Schematic” [Ziemke and Lowe 2009]

• Schema

  − Identifies the principal characteristics of the architecture without providing a detailed design of the component parts or the dynamics of their interaction
The Cognitive-Affective Architecture Schematic

• The autonomy of the agent is effected through a hierarchy of homeostatic self-regulatory processes which exploit a progression of associated emotions
  – Basic reflexes linked to metabolic regulation
  – Drives and motivations
  – Emotions and feelings linked to “higher” cognitive functions

• This follows closely Damasio’s hierarchy of levels of homeostatic regulation

• “The emotional aspect of cognition, providing motivation and value to an otherwise neutral world, ... is a fundamental part of the make-up of and organism with respect to sensorimotor learning”.
The Cognitive-Affective Architecture Schematic

- The Enactive Organizational Constraints Hierarchy traverses two dimensions
  1. Constitutive Organization
  2. Behavioural Organization
The Enactive Organizational Constraints Hierarchy traverses two dimensions

1. *Constitutive Organization*

   Internal dynamics to maintain autonomy in the face of perturbation by various stimuli

   a) Metabolic homeostatic self-regulation
   b) Stimulus valence evaluation
   c) Somatic state response
   d) Content evaluation
The Enactive Organizational Constraints Hierarchy traverses two dimensions

1. **Constitutive Organization**

   Increasing …

   - Organizational complexity
   - Decoupling between stimulus and response
   - Degree of appraisal and associated adaptivity
The Cognitive-Affective Architecture Schematic

- The Enactive Organizational Constraints Hierarchy traverses two dimensions

2. *Behavioural Organization*

   Each level is matched by an associated level in the behavioural organization dimension:

   a) approach-avoidance
   b) sequenced behaviour
   c) multi-sequenced behaviour

- Coupled by sensorimotor perception
The Cognitive-Affective Architecture Schematic

- Cognitive-Affective Architecture Schematic is a later version

  - Reflect the coupling by a *single* space of constitutive organization which is viewed from two perspectives:

1. Internal organization

2. Behavioural organization
The Cognitive-Affective Architecture Schematic

Constitutive Organization

Realized by recruitment of a progression of emotions from reflexes, through drives and motivations, to emotions-proper and feelings
The Cognitive-Affective Architecture Schematic

**Constitutive Organization**

Each level is associated on the *Internal Organization axis* with an increasing level of *homeostatic autonomy-preserving self-maintenance*:

a) Basic metabolic processes
b) Reactive sensorimotor activity (pre-somatic effects)
c) Associative learning and prediction (somatic modulation)
d) Interoception and internal simulation of behaviour prior to action
The Cognitive-Affective Architecture Schematic

Constitutive Organization

Each level is associated on the Behavioural Organization axis with an increasing level of complexity in behaviour.

a) Approach-avoidance
b) Sequenced behaviours
c) Multi-sequenced behaviours
The key idea under-pinning the Cognitive-Affective Architecture

- Different levels of cognitive function and behavioural complexity are associated with, and are brought about by, different levels of emotion

- Each is linked to affective homeostatic processes ranging from reflexes right through to internal simulation
Soar in more detail …
Soar in more detail …

The following is abstracted directly from

Soar in more detail …

• Recall:
  – Soar is a *Unified Theory of Cognition (UTC)*
  – Behaviour = Architecture x Content

• Every architecture is a theory about what is common to the content it processes

• Soar is a theory of what cognitive behaviours have in common
Soar in more detail …

• The Soar theory asserts that cognitive behaviour has at least the following characteristics:

1. It is goal-oriented

2. It reflects a rich, complex, detailed environment

3. It requires a large amount of knowledge

4. It requires the use of symbols and abstractions
   • cf. the concept of something vs. something in particular (a car vs. this particular car)
   • “… some of the knowledge you have can be elicited by something other than your perceptions in all their detail. We’ll call that thing a symbol (or set of symbols)
Soar in more detail …

• The Soar theory asserts that cognitive behaviour has at least the following characteristics:

5. It is flexible, and a function of the environment

  • “… human cognition isn’t just a matter of thinking ahead, it’s also a matter of thinking in step with the world”

  • … but just ahead of it, too! cf. the different timescales of anticipation

6. It requires learning from the environment and experience
Soar in more detail …

• If you want to be goal-oriented, you have to be goal-oriented about something if you want to produce behaviour

• An example (which we’ll use throughout this section)
  
  – Joe Rookie is a hypothetical pitcher with the Pittsburg Pirates, about to throw the first pitch of his major league career. He chooses to throw a curve ball. The batter, Sam Pro, hits the ball, but Joe is able to catch it after it bounces once between home plate and the pitching mound. Then, he quickly turns and throws the batter out at first base
Soar in more detail …

Soar in more detail …
Soar in more detail …

• “…Joe displays many of the characteristics of intelligent behavior …”:

1. Behaves in a goal-oriented manner
   • Overriding goal: win the game
   • Sub-goals
     – Get the batter out
     – Striking the batter out with a curve ball
     – Throwing the batter out at first base

2. Operates in a rich, complex, detailed environment
   • Positions and movement of the batter and other member of the team
   • Number of balls and strikes
   • The sound of the ball striking the bat
   • The angle his body makes with the first baseman as he turns to throw
   • …
Soar in more detail …

• “…Joe displays many of the characteristics of intelligent behavior …”:

3. Uses a large amount of knowledge
   • In deciding on the pitch, draws on statistics about his team, his pitching record, Sam Pro’s batting record, …

4. Uses symbols and abstractions
   • Joe has never played this particular game before
   • So, he can draw on his previous experience only by abstracting away from this day and place
Soar in more detail …

• “…Joe displays many of the characteristics of intelligent behavior …”:

5. Behaves flexibly as a function of the environment

  • In choosing his pitch, Joe responds to his own perceptions of the environment
    – Is it windy?
    – Is the batter left- or right-handed
    – …
  • When the ball is hit, Joe must show flexibility again
    – Change his sub-goal to the new situation

6. Learns from the environment and experience

  • “Learning is the acquisition of knowledge that can change your future behaviour”
  • Next time, he’s better throw Sam a fast ball!
Soar in more detail …

- Behaviour = Architecture x Content (i.e. Knowledge)

- How can we express the knowledge so that the model of Joe acts in a goal-oriented way?

K1: Knowledge of the objects in the game
   e.g. baseball, infield, base line, inning, out, ball/strike count

K2: Knowledge of abstract events and particular episodes
   e.g. how batters hit, how this guy batted last time he was up

K3: Knowledge of the rules of the game
   e.g. number of outs, balk, infield fly

K4: Knowledge of objectives
   e.g. get the batter out, throw strikes

K5: Knowledge of actions or methods for attaining objectives
   e.g. use a curve ball, throw to first, walk batter

K6: Knowledge of when to choose actions or methods
   e.g. if behind in the count, throw a fast ball

K7: Knowledge of the component physical actions
   e.g. how to throw a curve ball, catch, run
Soar in more detail …

• Behaviour as movement through problem spaces
Soar in more detail …

- Behaviour as movement through problem spaces
  - Joe must make his decisions with respect to the current situation, based on
    - That situation
    - What he recalls from the past
    - What he anticipates about the future
  - But at all the moments that can arise
  - Problem spaces partition knowledge in goal-relevant ways
    - Everything in the previous diagram was concerned with the goal of getting Sam out
Soar in more detail …

• Behaviour as movement through problem spaces
  – State “Sam Pro is at bat” can be represented by two features and associated feature values
    • Batter name *Sam Pro*
    • Batter status *not out*
  – The state is a representation of *all* of the aspects of the situation – internal and external – that the model may need to chose its next action
  – To model a particular behaviour, we need
    • Initial state $S_0$
    • Desired goal state or states
    • All specified by set of features and associated feature values
      – E.g. goal states: feature $f_2$ has value $v_6$ (irrespective of the values of other features)
      – If $f_2$ is batter status and value $v_6$ is *out*, then the goal state has been achieved
Soar in more detail …

• Behaviour as movement through problem spaces
  
  – Goal (circle)
  – Problem space: expanding set of possibilities that can unfold over time (triangle)
  – States (rectangles)
    • Vocabulary of features (bold)
    • Their possible values (italics) … values can also be a set of features
  – State transition (arrows) … operators reflecting internal or external behaviour
Soar in more detail …

• Behaviour as movement through problem spaces
  
  – Movement from the current state to a new state occurs through the application of an operator to the current state
    
    • The operator changes some of the features and values
  
  – To keep behaviour goal-directed
    
    • The succession of operators that are applied to the state and the resulting state transformations must be guided by

    The principle of rationality:

    *If an agent has knowledge that the application of an operator will lead to one of its goals then the agent will select that operator*
Soar in more detail …

• Behaviour = Architecture x Content (i.e. Knowledge)

• How can we express the knowledge so that the model of Joe acts in a goal-oriented way?

“By representing the knowledge in terms of states and operators and guiding the choice of which operator to apply by the principle of rationality”

• Mapping knowledge into states and operators is the first step towards tying the content of Joe’s world to the Soar cognitive architecture; but …

  – What knowledge becomes part of the state?
  – What knowledge becomes part of the operators?
  – How do we avoid having every operator available in every state?
  – How do we know what the application of an operator will do?
  – How do we know when the goal has been achieved?
Soar in more detail …

• Tying the content to the architecture
  
  – Soar’s architectural mechanisms process the four types of objects
    
    • Goals
    • Problem spaces
    • States
    • Operators

  
  A particular instance of these four kinds of objects is organized into a structure call the goal context, with 4 slots, one for each of the objects.
Soar in more detail …

- Tying the content to the architecture

**Knowledge about things in the world**
- \( K1 \): Knowledge of the objects in the game
  - e.g. baseball, infield, base line, inning, out, ball/strike count

**Knowledge about abstract ideas**
- \( K2 \): Knowledge of abstract events and particular episodes
  - e.g. how batters hit, how this guy batted last time he was up

**Knowledge about the rules of the game**
- \( K3 \): Knowledge of the rules of the game
  - e.g. number of outs, balk, infield fly

**Knowledge about objectives**
- \( K4 \): Knowledge of objectives
  - e.g. get the batter out, throw strikes

**Knowledge about actions or methods for attaining objectives**
- \( K5 \): Knowledge of actions or methods for attaining objectives
  - e.g. use a curve ball, throw to first, walk batter

**Knowledge of when to choose actions or methods**
- \( K6 \): Knowledge of when to choose actions or methods
  - e.g. if behind in the count, throw a fast ball

**Knowledge of the component physical actions**
- \( K7 \): Knowledge of the component physical actions
  - e.g. how to throw a curve ball, catch, run
Soar in more detail …

• Tying the content to the architecture
  – Use
    • Knowledge of objectives for goals (e.g. K4)
    • Knowledge about actions to define operators (e.g. K5 and K7)
    • Domain knowledge of people and objects (e.g. K1) is represented by
      – Features
      – Values
    – By organizing domain knowledge into multiple problem spaces
      • Limit the set of operators to be considered in searching for a goal state
Soar in more detail …

• Tying the content to the architecture
  
  – How do we know what an operator will do?

    • By executing the operator (action) in the real world and observing the outcome (e.g. execute K7 and observe result)

    • Or by knowledge of abstract events or particular episodes (e.g. K2)

      – i.e. acting or thinking about acting
Soar in more detail …

• Tying the content to the architecture

  – Still some unanswered questions:

    • How should knowledge in its general form be represented?

    • How should it be represented within a goal context?

    • What are the mechanisms for perceiving and acting on the external world?

    • What are the architectural processes for using knowledge to create and change the goal context?
Soar in more detail …

• Memory, Perception, Action, and Cognition

  – The difference between general knowledge and specific applications of that knowledge
    • Bicycles, in general, vs.
    • My personal bicycle

  – Captured in Soar by two different memory systems

    • Long-term Memory (LTM)
      – Knowledge that exists independently of the current goal context
      – Describes what is true in general

    • Working Memory (WM)
      – Describes particular and specific occurrences of that knowledge (bikes are made of carbon, steel, aluminium, or titanium).
      – Describes what the cognitive model hold to be true in a particular situation (my bike is made of titanium)
Soar in more detail …

• Memory, Perception, Action, and Cognition
  
  – Behaviour can only occur after we tie the content of the domain to a goal context

  – The content of the domain resides in LTM
    • Expressed in general terms
    • Independently of any particular situation

  – The goal context resides in WM
    • Comprises exactly the four slots of the goal context
      – Current goal, problems space, state, and operator
      – Features and values that make up the state

  – Knowledge moves from its general form in LTM to its specific form in WM by a process called the decision cycle
Soar in more detail …

• Memory, Perception, Action, and Cognition
  
  – Long-term Memory

  • Knowledge is represented as associations called production rules

    (a1) If I perceived I am at the mound then suggest a goal to get the batter out

    (a2) If there is a goal in WM to get the batter out then suggest achieving it using the Pitch problem space with an initial state having balls 0 and strikes 0

    (a3) If using the Pitch problem space and I perceive a new batter who is left- (or right-) handed then add batter not out and batter left-handed (or right-handed) to state

    (a4) If using the Pitch problem space and the batter is not out then suggest the throw-curve operator
Soar in more detail …

• Memory, Perception, Action, and Cognition
  – Long-term Memory
    • Knowledge is represented as associations called production rules
      (a5) If using the Pitch problem space and the batter is not out and the batter is left-handed then suggest the throw-fast-ball operator
      (a6) If both throw-fast-ball and throw-curve are suggested then consider throw-curve to be better than throw-fast-ball
      (a7) If the throw-curve operator has been selected in the Pitch problem space then send throw-curve to the motor system and add pitch thrown to the state
      (a8) If using the Pitch problem space and the pitch was thrown and I perceive a hit then add pitch hit to the state
Soar in more detail …

• Memory, Perception, Action, and Cognition
  
  – Working Memory
    
    • WM is just goal context
    
    • The “if” part of each LTM association tests either
      – A perception, or
      – Elements of the goal context in WM
    
    • If there is a match between the “if” part of an association and elements in the WM → the association has been triggered
    
    • This triggering causes the “then” part to fire
      – Either by sending a message to the motor system (a7)
      – Or suggesting changes to the goal context
    
    • Thus, each matching association maps from current goal context elements to new goal context elements
Soar in more detail …

• Memory, Perception, Action, and Cognition
  – Working Memory
    • There can be dependencies between the associations (production rules)
      – For example, a2 will not match the current state until a1 has matched and fired
      – a4 will not match until a1 through a3 have matched and fired
    • The dependencies are part of the semantics of the domain of baseball
    • Soar doesn’t recognize the existence of these dependencies; it just matches the “if” parts and performs the “then” parts
Soar in more detail …

- Memory, Perception, Action, and Cognition
  - Working Memory
    - Since all the elements in the associations are expressed in terms of
      - perceptions,
      - actions, and
      - goal context elements
    - They are processed
      - by perception or
      - through associations that fire during the decision cycle
    - Perceptual input enters working memory (as additional feature-value information added to the state) irrespective of the decision cycle
Soar in more detail …

• Memory, Perception, Action, and Cognition

  – Decision cycle

  • The processing component that generates behaviour out of the content that resides in LTM and WM

  • Purpose of the decision cycle is to change the value in one of the four slots of the goal context

  • Typically, this is done by changing the value in the operator slot of the goal context

  • Why?

  • Because goal-directed behaviour corresponds to movement in a problem space from the current state to a new state through the application of an operator to the current state
Soar in more detail …

• Memory, Perception, Action, and Cognition
  – Decision cycle
    • Two phases
      1. Elaboration
        » The contents of working memory are matched against the “if” parts of the association in LTM
        » All associations that can fire do fire (in parallel)
        » This results in changes to the features and values of the state in addition to suggestions (preferences) for changes in the context slots
        » As a result, new associations may fire
        » Elaboration continues in parallel waves of association firings until no more associations fire
Soar in more detail …

• Memory, Perception, Action, and Cognition
  
  – Decision cycle

  • Two phases

    2. Decision

      » Preferences (for changes to the context) are added to the WM

      » Evaluated by a fixed architectural decision procedure

      » Applied to the preferences and context slots, independent of the domain semantics (i.e. in terms of operators and operator preferences and not in terms of specific domain actions)

• The outcome of the decision cycle is the change of a single context slot: a new goal, a new problem space, a new state, or a new operator is selected
Soar in more detail …

- Memory, Perception, Action, and Cognition
  - Decision cycle

  - For example, how does the Soar cognitive model choose to throw a curve ball?

Decision cycles 1 & 2:

Initial WM after associations a1 and a2 fired:

Get-batter-out goal;
Pitch problems space;
initial state: 0/0 for **balls 0** and **strikes 0**

The … in state represents other working memory elements resulting from perception
Soar in more detail …

- Memory, Perception, Action, and Cognition
  - Decision cycle

- For example, how does the Soar cognitive model choose to throw a curve ball?

Elaboration phase of Decision Cycle 3

Results in state changes as a result of a3 (adds information about the handedness of the batter to the state) …

bnl - **batter not out** and **batter left-handed**

This change triggers a4 and a5, producing proposal for throw-curve and throw-fast-ball

The operator proposals trigger a6 producing a preference for throwing the curve ball

No more associations fire (elaboration over)
Soar in more detail …

- Memory, Perception, Action, and Cognition
  - Decision cycle
    - For example, how does the Soar cognitive model choose to throw a curve ball?

Decision phase of Cycle 3
Resolves the preferences for operator proposals,
Selects the throw-curve operator as the new value for the operator slot of the goal context
Soar in more detail …

- Memory, Perception, Action, and Cognition
  - Decision cycle

  - For example, how does the Soar cognitive model choose to throw a curve ball?

  Elaboration phase of Decision Cycle 4

  Results in the application of the throw-curve operator (as a result of a7)

  and subsequent state changes (as a result of a8)
Soar in more detail …

• Memory, Perception, Action, and Cognition
  – Decision cycle
    • Why are changes to the context slots called suggestions or preferences?
    • To make sure all the knowledge available in the current context if brought to bear before the decision is made
      – A decision to change a context slot is only made after all the associations in the LTM that match the current context have been retrieved
      – It the principle of rationality: the model should use all its knowledge to achieve its goals
      – Preferences allow the architecture to collect all the available evidence for potential changes before making any changes (i.e. after elaboration is complete)
  
• But what happens if the decision cycle can’t reach a decision?
Soar in more detail …

• Detecting a lack of knowledge

  – What would happen if a6 was not part of the domain content?

    (a6) If both throw-fast-ball and throw-curve are suggested
    then consider throw-curve to be better than throw-fast-ball

  – At the end of the elaboration cycle in which a3, a4, and a5 fire, two
    operators would be proposed but there would be no way to decide
    between them (that’s what a6 would have done)

    • The knowledge in LTM that can be accessed by the current context is
      insufficient to resolve the tie between the two proposed pitch operators

    • Processing reaches an impasse
Soar in more detail …

• Detecting a lack of knowledge

  – “An impasse is an architectural structure that arises whenever the decision process cannot resolve the preferences in working memory to produce a single change in the context slots”
  
  – “An impasse is what happens when the decision cycle can’t decide”
  
  – It happens because the current context doesn’t have access to sufficient knowledge in the LTM

• Soar uses multiple problem spaces to partition knowledge and limit the operators the need to be considered when searching for a goal state

• So far, we’ve used just one problem space, Pitch

• But there are others (there have to be in order to realize Joe’s complete behaviour)
Soar in more detail …

• Detecting a lack of knowledge
  
  – The impasse may mean that the required knowledge is not associated with the *current context*
  
  – When an impasse occurs, the architecture automatically establishes the goal of eliciting the knowledge needed to continue processing, i.e. to make the required decision

  • Here, the impasse is an operator tie in the Pitch problem

  • The goal is to resolve the tie

  • The domain content must fill in the remainder of the new goal context using LTM associations: select a problem space, state, and operator
Soar in more detail …

- Detecting a lack of knowledge
  - When the decision cycle can’t decide
    - The architecture creates an impasse structure (denoted by a crescent in the diagram below)
    - Which links the current goal (get-batter-out) to a new subgoal (resolve-tie)
    - Producing a goal-subgoal hierarchy
Soar in more detail …

- Detecting a lack of knowledge
  - When the decision cycle can’t decide
    - Knowledge in the domain content then determines the rest of the goal context
      - The problem space to work in to achieve the goal (Recall)
      - The initial state and the operators that are applicable to it
    - The decision cycle may now result in a change to goal or subgoal context
      - Processing normally continues in the lower problem space (Recall)
      - If the decision cycle suggests a change to a slot in the higher goal context (Pitch), that takes precedence and the lower context disappears
Soar in more detail …

• Detecting a lack of knowledge
  – When the decision cycle can’t decide
    • The LTM must also have sufficient knowledge to resolve the tie
    • Here, we assume that the way Joe would make this decision is by recalling how successful he’s been with curve ball and fast ball pitches in the past
    • So, we add the following association to the model’s LTM
      (a9) If there’s a goal to resolve an operator tie in the Pitch problem space Then suggest achieving it using the Recall problem space with and initial state containing the tied operators
Soar in more detail …

• Detecting a lack of knowledge

  – In general, working memory consists of a goal-subgoal hierarchy where each subgoal exists to resolve an impasse in the problem space above it

    • In fact, every goal in Soar is established by the architecture

    • The highest goal is created automatically

    • All subsequent goals result from impasses

    • The goal “get-batter-out” is really the subgoal to resolve the impasse on the get-batter-out operator in a higher problem space (which we haven’t considered here)

  – The hierarchy grows as impasses arise

  – The hierarchy shrinks as impasses are resolved by accessing the required knowledge
Soar in more detail …

• Detecting a lack of knowledge

  – An operator tie is just one way in which the decision cycle can fail: there are others

  – Note that the full set of impasse types defined in Soar is fixed and domain-independent

    • Just as there is a fixed vocabulary of preferences

    • There is a fixed vocabulary of impasses

    “so that it can process domain content without understanding the meaning of that content”
Soar in more detail …

• Learning
  – When it resolves an impasse, Soar learns new associations

• An impasse indicates a lack of available knowledge
  – Acquiring this knowledge from LTM through subgoaling provides an opportunity for learning
  – Association a9 creates Recall’s initial state with nothing but the information on the tied operators
  – To accomplish the Recall goal, we need to access associations in LTM that encapsulate Joe’s past experiences and allow the tie to be resolved
Soar in more detail …

• Learning
  
  – First, an aside:

  Learning could lead to removing associations, as well as adding them.

  However, Soar (as a theory) assumes that long-term memory only grows

  So, in Soar, forgetting only results when a new association prevents an old association from firing
Soar in more detail …

• Learning

  – We add two kinds of knowledge to the model

    • Knowledge that encodes memories of past events (episodic memory)

    • Knowledge that defines two new operators

      – Augment

      – Evaluate

(a10) If using the Recall problem space and the batter is left-handed and the weather is windy then add event116 to Recall’s state with substructure part-of game22, isa homerun, pitch throw-fast, etc.

This association does not fire in the Pitch problem space (it fires in the Recall problem space)
Soar in more detail …

• Learning

  – To elicit the memory of event116, Recall’s state must contain the
    features and values in the “if” part of association a10

• To do this, we use augment operators that copy a single feature and its
  values from Pitch’s state to Recall’s state

  (a11) If there is a feature in Pitch’s state that is not in Recall’s state
       then suggest an augment operator for that feature

  (a12) All augment operators are equally acceptable

  (a13) If the augment operator has been selected in Recall for some feature
       then add that feature and its values to Recall’s state
Soar in more detail …

• Learning

  – Association a11 proposes one augment operator for every feature that is in Pitch’s state and not yet in Recall’s state

  – Association a12 provides preference information about the proposed operators to the decision process (resulting in random selection of a single instance of Augment at the end of the decision cycle, in this case)

  – Association a13 does the actual copying
Soar in more detail …

Pre-impasse environment from the get-batter-out goal

Processing in the resolve-tie subgoal
Soar in more detail …

• Learning

  – Bear in mind that processing in the Recall problem space is driven by the goal of resolving the tie between the throw-curve and throw-fast-ball operators in the Pitch problem space

  – Since no useful knowledge is elicited by the batter information alone

    • associations a11 and a12 fire again

    • New feature and value is copied from Pitch

    • This continues and, eventually, an augment operator is selected that copies the weather feature

    • Then, the contents of WM trigger association a10: episodic memory of event116
Soar in more detail …

Pre-impasse environment from the get-batter-out goal

Processing in the resolve-tie subgoal
Soar in more detail …

- Learning
  - WM now contains enough knowledge to resolve the tie
  - But we need three more associations (in LTM) to define the evaluate operator

(a14) If using the Recall problem space and there are two tied operators and an event with a pitch that is one of the tied operators then suggest an evaluate operators

(a15) Prefer evaluate operators over augment operators

(a16) If applying an evaluate operator to a state that has two tied operators and an event that is a homerun with a pitch that is one of the operators then prefer the other operator
Soar in more detail …

• Learning

  – Association a14 proposes the evaluate operator on the basis of a relevant event

  – Association a15 is included to prefer an evaluation just in case there are other argument operators that are still possible

  – Association 16 adds a preference to WM that results in applying the evaluate operator

  – In this example, a14-a16 means that if Joe remembers a time when someone threw a fast ball that led to a homerun under similar circumstances, then he should prefer to throw a curve
Soar in more detail …

Pre-impasse environment from the get-batter-out goal

Processing in the resolve-tie subgoal
Soar in more detail …

• Learning
  – So far
    • The operator-tie impasse arises because the original goal did not provide cues for eliciting the LTM knowledge to choose between the two pitches
    • A subgoal to find the knowledge is created by the architecture
    • Domain content then expands this subgoal into a second goal context involving the Recall problem space … the total context that can access LTM knowledge is expanded
    • As a result of processing Recall (and achieving the Recall goal state), the knowledge needed in Pitch becomes available and the impasse is resolved
Soar in more detail …

• Learning
  
  – Now we have an opportunity for learning a new association

  • This new association makes the required knowledge immediately available in Pitch under the circumstances that originally led to the impasse.

  • The Soar architecture automatically forms a new association whenever results are generated from an impasse

  • The learned associations are called chunks

  • The learning process is called chunking

(c1) If using the Pitch problem space and both the throw-curve and throw-fast-ball operators are suggested and the batter is left-handed and the weather is windy then prefer the throw-curve operator
Soar in more detail …

• Learning

  – The “if” part of c1 includes only elements in the context before the impasse arose (as you’d expect)

  – The “then” part mirrors the knowledge elicited.

  – With c1 in LTM, the model will not encounter and impasse the next time it is in a similar state

  – Instead, the chunk will fire, the curve ball will be preferred, and processing will continue in Pitch
Soar in more detail …

• Learning
  – Chunking is essentially a deductive compositional learning mechanism
    • The preference for the curve ball represents a kind of deduction from prior knowledge
    • The new association is composed of a “then” part containing the deduction and an “if” part containing the knowledge that contributed to the deduction
  – This recombination of existing knowledge makes it accessible in new contexts via LTM associations
  – Chunking overcomes some of the partitioning function of problems spaces, making knowledge available in a problem space that wasn’t available before learning
    • But only when experience requires it
Soar in more detail …

• Learning
  
  – As a theory, Soar says chunking happens all the time
    
    • It is a ubiquitous mechanism
    
    • Requires no intention to learn on the part of the agent
  
  – Impasses arise automatically where there is a lack of knowledge
  
  – Chunks arise automatically when knowledge becomes available that helps resolve the impasse
Soar in more detail …

• Learning
  
  – Although chunking is the only architectural mechanism for learning, nothing about Soar prevents other styles of learning (as content theories that rest on top of the architecture)

  • Learning by induction
  
  • Learning by abduction
  
  • Learning by instruction
Soar in more detail …

- A Soar Model of Joe Rookie

  - So far, we’ve described only a small part of the knowledge required
  
  - To expand the model to the full functionality, we need to

    - Specify the domain knowledge Joe needs (content)
    
    - Tie the different types of domain knowledge to the different parts of the goal context
      
        - Goals
        - problem spaces
        - state structures including percepts and any working memory elements that trigger actions
        - operators
      
    - Specify the relationships between problem spaces by the impasses that will arise and the kinds of knowledge that will be missing and consequently learned
Soar in more detail …

- A Soar Model of Joe Rookie
  - The Top goal and Top problem space are given by the architecture
    - These are the only goal and problem space that persists over time
  - All perceptions and actions occur though the Top state
    - Thus, they are part of the pre-impasse environment for all subgoals
    - This ensures that tests of percepts and actions will be preserved in any chunks that are learned
  - Operators in the Top problem space correspond to tasks Joe has to perform
    - Getting batter out
    - Comprehending language and gestures
    - …
Soar in more detail …

- A Soar Model of Joe Rookie
Soar in more detail …

• A Soar Model of Joe Rookie
  – Thus, the Pitch problem space is actually evoked in response to an impasse in Soar’s Top problem space
  – Knowledge may be spread out among subspaces in various ways
    • But it is integrated as a result of chunking in response to impasses
Soar in more detail …

• Summary
  – The Goal Context
    • The key structure in architecture
    • Every goal context is defined by four slots and their values:
      – The goal (ensures goal-directed behavior)
      – The problem space (partitions or organizes knowledge in a goal-related way)
      – The state (an internal representation of the situation)
      – The operator (maps from state to state)
Soar in more detail …

• Summary

  – Working Memory (WM)

    • Contains the current situation (including past states and hypothetical states if they are important for reasoning) in the form of one or more goal contexts

    • Also holds the results of perception as features and values in the Top state

    • The contents of working memory trigger both associations in LTM and motor actions
Soar in more detail …

• Summary

  – Long-term Memory (LTM)
    • The repository for domain content that is processed by the architecture to produce behaviour
    • The form of knowledge in LTM is associations that map from the current goal context in WM to a new goal context
    • Because the mapping is from context to context, the triggering of an association can be accomplished by a simple match process against working memory
    • In addition to being associational, Soar’s long-term memory is impenetrable
      – a Soar system cannot examine its own associations directly
      – its only window into LTM is through the changes to working memory that are the results of association firings.
Soar in more detail …

• Summary
  – The Perception/Motor Interface
    • The mechanism for defining mappings
      – from the external world to the internal representation in working memory
      – From the internal representation back out to action in the external world
    • Through this interface, perception and action can occur asynchronously and in parallel with cognition
Soar in more detail …

• Summary
  – The Decision Cycle
    • The basic architectural process supporting cognition
    • It is composed of two phases
      – Elaboration phase: parallel access to LTM during the elaboration phase changes the features and values that define the state and suggests new values for context slots
      – Decision phase: quiescence defines the moment at the end of the elaboration phase and the beginning of the decision phase when all the knowledge that can be elicited by the current context has been
Soar in more detail …

• Summary
  – The Decision Cycle
    • During the second phase, the decision procedure interprets the domain-independent language of preferences and suggestions for changes to the context
    • The result of the decision procedure is either
      – a single change to the goal context
      – or an impasse if no single change has been suggested
    • The application of a single operator per decision cycle imposes a cognitive bottleneck in the architecture, that is, a limit on how much work cognition can do at once
Soar in more detail …

• Summary

  – Impasses

  • Signal a lack of knowledge, and therefore, an opportunity for learning

  • An impasse occurs automatically whenever the knowledge elicited by the current context isn’t enough for the decision procedure to resolve the preferences in working memory to a single change in the context

  • The language of impasses, like the language of preferences, is defined independently of any domain

  • When an impasse arises, the architecture also automatically begins the creation of a new subgoal context whose goal is to resolve the impasse

  • In this way, impasses impose a goal/subgoal hierarchy on the contexts in working memory
Soar in more detail …

• Summary
  
  – Chunking

  • The pervasive architectural learning mechanism

  • Chunking automatically creates new associations in LTM whenever results are generated from an impasse

  • The new associations map the relevant pre-impasse WM elements into WM changes that prevent that impasse in future similar situations

  • Although basically a deductive or compositional mechanism, chunking serves many purposes.
Soar in more detail …

• Summary
  – Chunking
    • It can integrate different types of knowledge that have been spread over multiple problem spaces
    • It can speed up behavior by compiling many steps through many subspaces into a single step in the pre-impasse problem space
    • It can be used as the basis of inductive learning, analogical reasoning, and so on
    • Because it is the only architectural mechanism for changing LTM, it is assumed to be the basis of all types of learning in people
Soar continues to evolve … [Laird 09]

- Perception
- Action
- Mental imagery (internal simulation)
- Procedural memory & reinforcement learning
- Semantic memory & learning
- Episodic memory & learning
Recommended Reading

