SYMBOLIC PROCESSING FOR INTELLIGENCE

Fourth Lecture
On unified theories of cognition
The William James Lectures
Harvard University

Allen Newell
Computer Science and Psychology
Carnegie-Mellon University

11 March 1987

REPRISE: HUMAN COGNITIVE ARCHITECTURE

What aspects are dictated by the nature of its world?

The real-time constraint on cognition

- From neural technology, to get mind-like behavior
- Only two small system levels available

The yield —

1. Neural, cognitive, rational, social timescales
2. Computational symbolic systems
3. Four levels of the cognitive band
    - Architecture — symbolic access @~10 ms
    - Elementary deliberation (automatic) @~100 ms
    - Selection of prepared operators (controlled) @~1 s
    - Composed operators (full cognition) @~10 s
4. Recognition-based architecture
5. Continual shift to recognition (learning)

PLAN OF THE LECTURE

Present a specific architecture for cognition — Soar

- The basis for a unified theory of cognition

Focus is on functionality (for this lecture)

- How Soar attains intelligent behavior
- How the requirements dictate its structure
  - The architectural features derived in Lecture 3
  - Also the details of making it be intelligent

1. Architecture for central cognition
2. Learning from experience
3. The total cognitive system
4. Functionality and ability
5. Qualitative aspects of human cognition

MAJOR FEATURES OF CENTRAL ARCHITECTURE

Cognition, but not perception or motor behavior

1. Problem spaces to represent all tasks

    Little knowledge yields search, lots yields direct path

    Problem-solving architecture (no process substrate)

2. Productions for all long-term memory (symbols)

    Search control, operators, declarative knowledge

3. Attribute/value representation medium for all things

4. Preference-based procedure for all decisions

    Preference language: accept/reject, better/worse

5. Goals (and goal stack) to direct all behavior

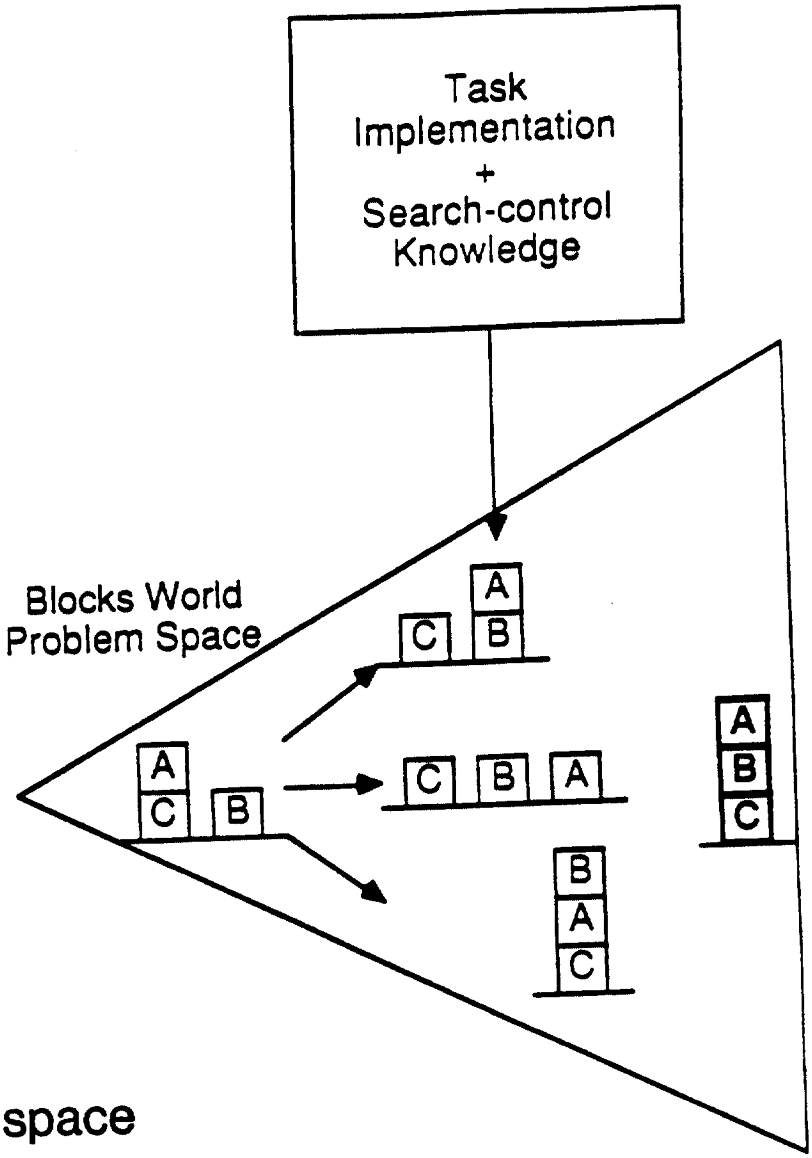
    Goals are created by the system itself

        At performance time from impasses, not in plans

    Operators perform the function of deliberate goals

6. Chunking of all goal-results occurs continuously

PROBLEM SPACE ARCHITECTURE



Primitive functions

- Select a problem space
- Select a state from those directly available
- Select an operator
- Apply the operator to obtain new state

The <u>deliberative</u> acts of architecture

PRODUCTION SYSTEM

Familiar view — Collection of condition-action rules

Better — Content-addressed memory, recognition system

Soar production system (OPS5-like)

```
                                    C,C    ⟶ A
    Conditions are patterns         C,C,C  ⟶ A
                                    C,C    ⟶ A,A
        Obtain all instantiations   [W,W,W, ... ]
```

Actions only add elements to working memory

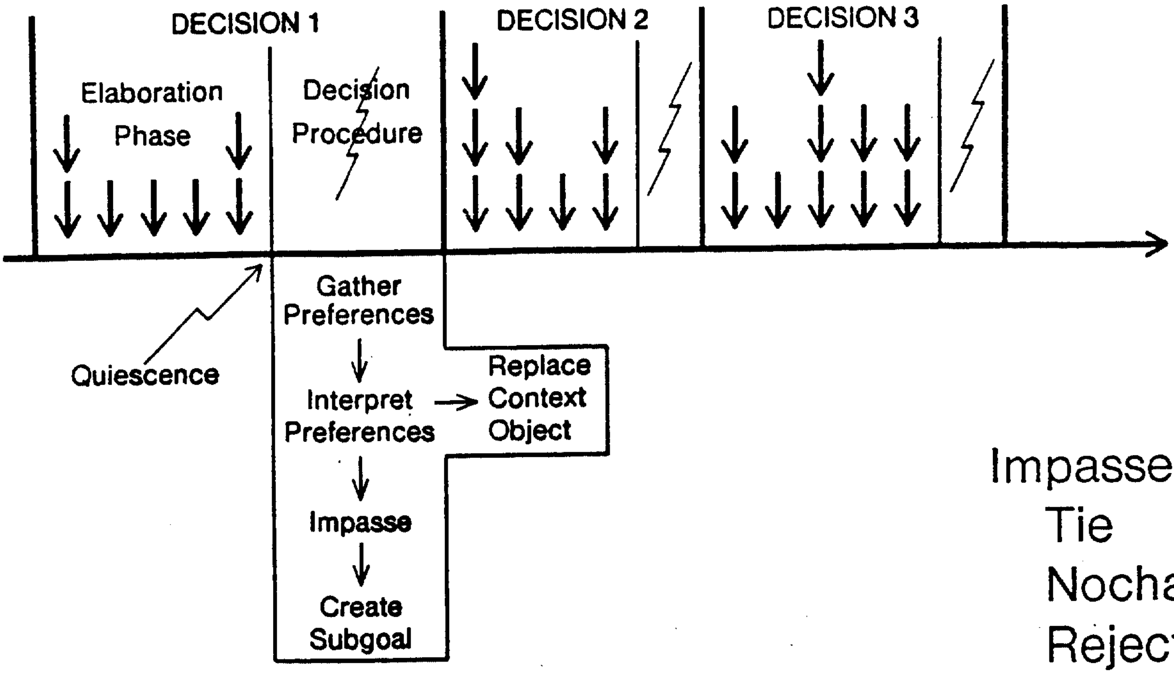
Elements leave when no longer accessible

No conflict resolution — Entirely parallel

Example Soar production

```
  (sp propose-operator*comprehend
   (goal <g> ↑problem-space <p> ↑state <s>)
   (problem-space <p> ↑name base-level-space)
   (state <s> ↑object <b> ↑input <i>)
   (box <b> ↑on table ↑on-top nothing)
 — (signal <i> ↑attention yes)
 ⟶
   (operator <o> ↑name comprehend)
   (preference <o> ↑role operator ↑value acceptable
    ↑goal <g> ↑problem-space <p> ↑state <s>))
```

DECISION CYCLE


DECISION 1
DECISION 2
DECISION 3
Elaboration Phase
Decision Procedure
Quiescence
Gather Preferences
Interpret Preferences
Replace Context Object
Impasse
Create Subgoal

Impasses
- Tie
- Nochange
- Reject
- Conflict

Elaboration phase produces preferences

(S13 acceptable for supergoal state)

(S13 rejected for supergoal state)

(Q2 acceptable for operator)

(Q7 acceptable for operator)

(Q7 better than Q2 for operator)

(Q9 indifferent to Q6 for operator)

Context

$|G_1|P_1|S_1|O$

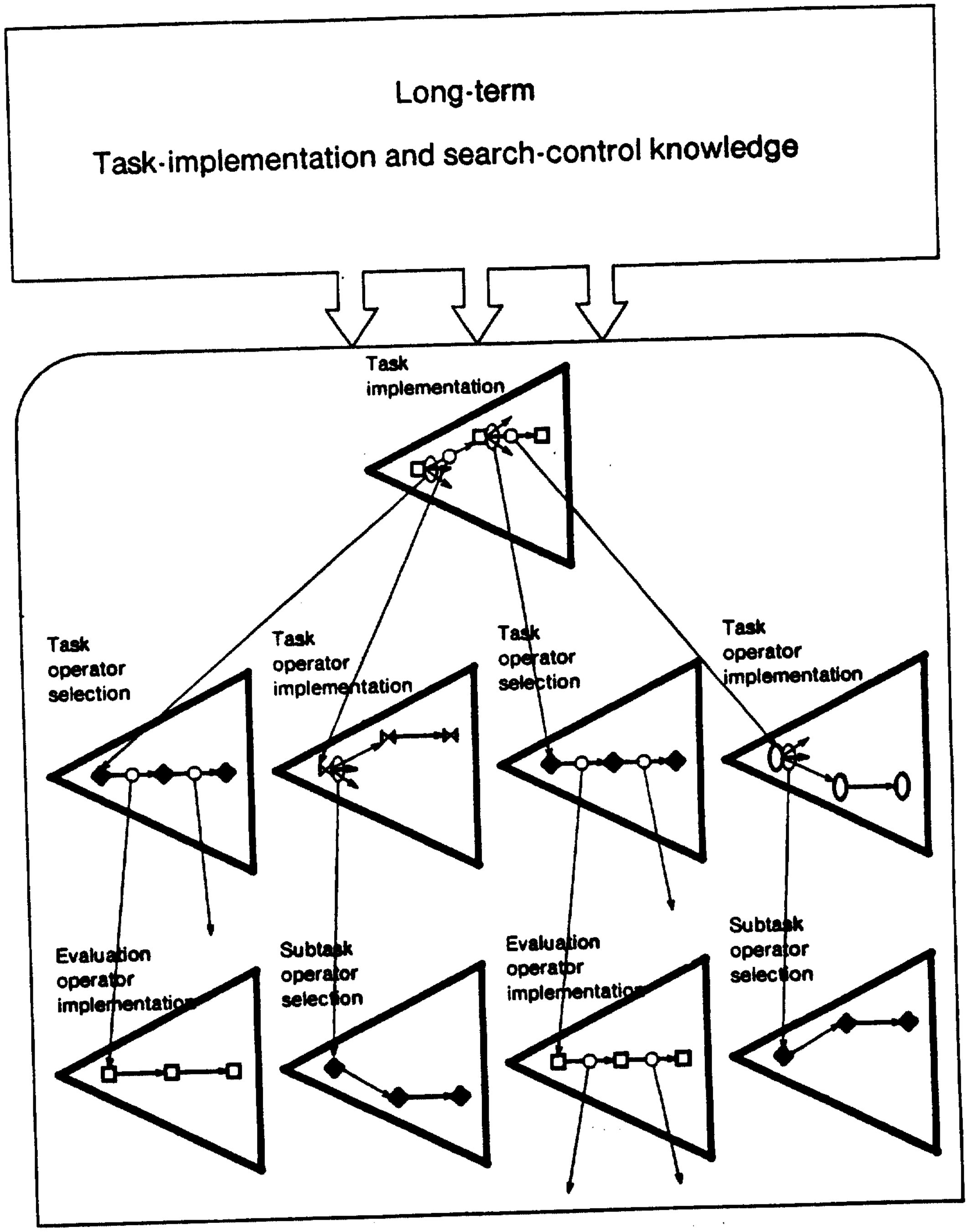
Nochange

$\rightarrow |G_2|P_2|S_2|{-}{-}|$

Tie

$\rightarrow |G_3|P_3|S_3|O$

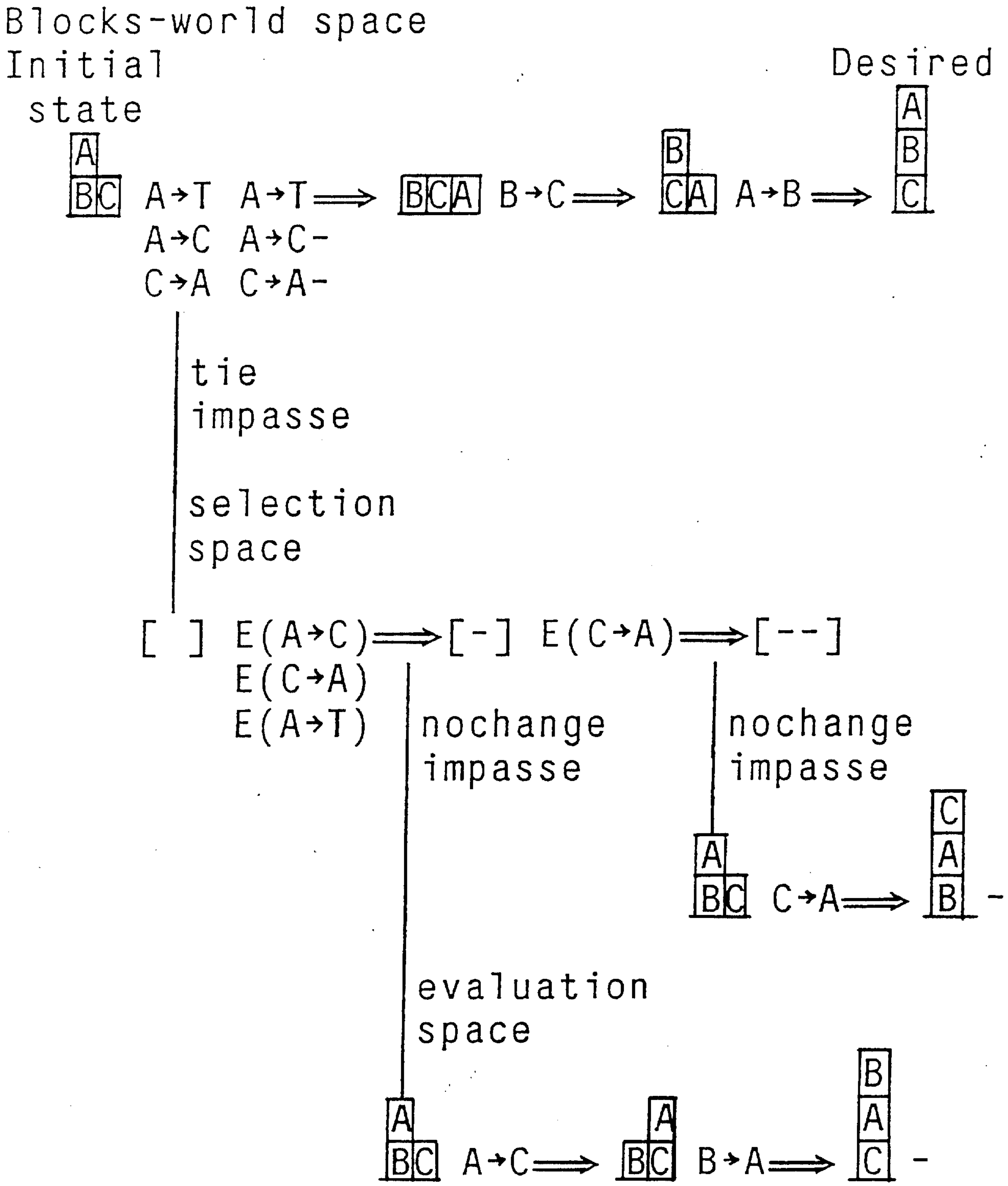
IMPASSES AND PROBLEM SPACES ALL THE WAY DOWN



Long-term

Task-implementation and search-control knowledge

EXAMPLE OF OPERATION



Blocks-world space

Initial state

A→T A→T ⟹ B→C ⟹ A→B ⟹ Desired

A→C A→C-

C→A C→A-

tie impasse

selection space

[ ] E(A→C) ⟹ [-] E(C→A) ⟹ [--]

E(C→A)

E(A→T)

nochange impasse

nochange impasse

C→A ⟹ -

evaluation space

A→C ⟹ B→A ⟹ -

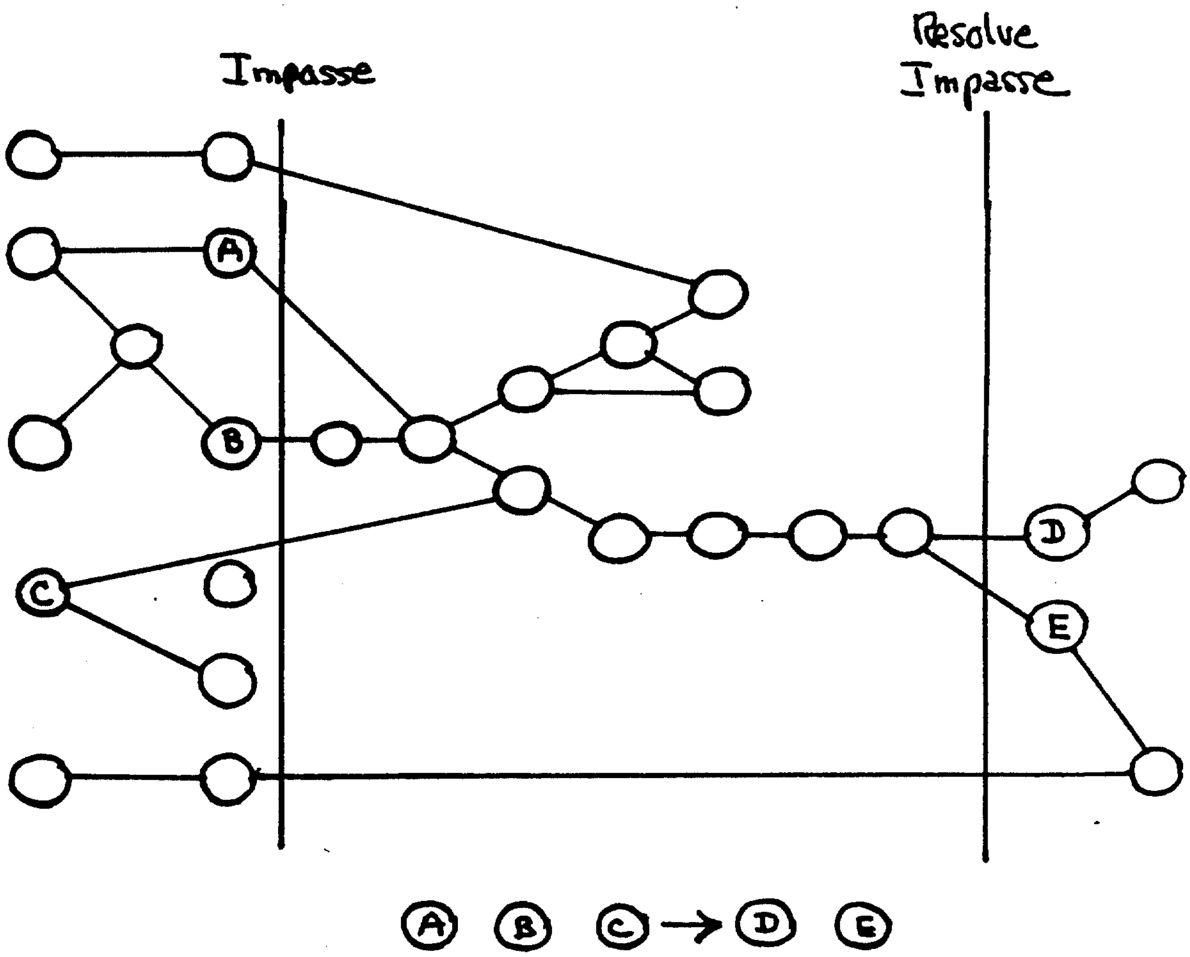
CHUNKING — LEARNING FROM EXPERIENCE

Converts goal-based problem solving into productions

- Actions — Based on the results of the subgoal
- Conditions — Based on the pre-impasse situation
  - The aspects necessary to produce the actions

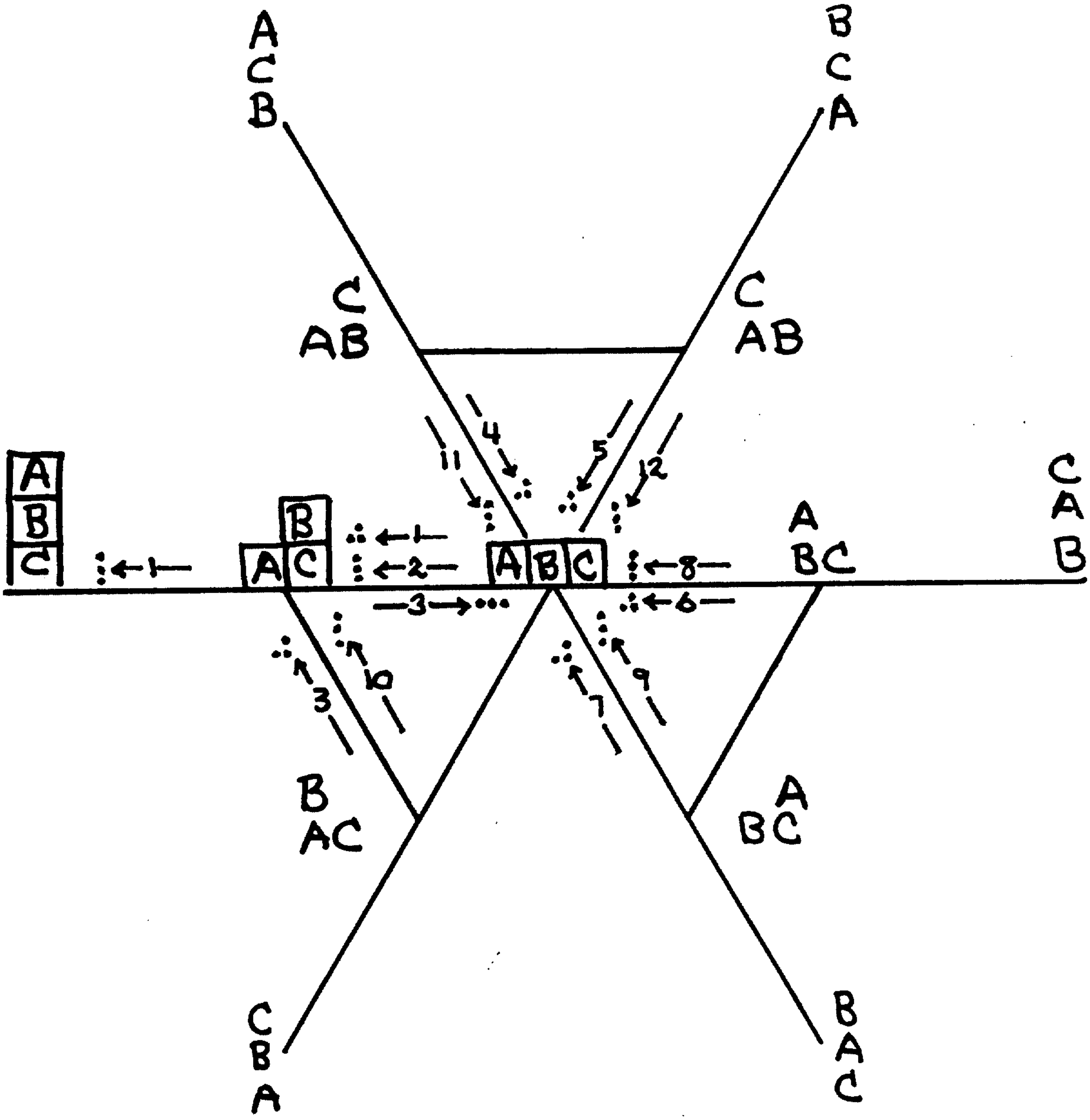
1. Chunks are productions — processes not data
2. A form of permanent goal-based caching
3. Chunks generalized implicitly
   - Ignore whatever the problem solving ignored
4. Learning occurs during problem solving
5. Chunking applies to <u>all</u> subgoals
   - Search control, operator implementation, ...
   - Whenever knowledge is incomplete or inconsistent
6. Learns only what system experiences
7. General mechanism for moving up the P-D isobar

CHUNKING — ILLUSTRATION



Chunk1:
If the problem-space is simple-blocks-world and
	the state is one proposition different than the goal and
	the state has block1 and block2 clear and
		block1 is on the table and
	the desired state has block1 on block2
then make a best preference for the operator that
	moves block1 onto block2.

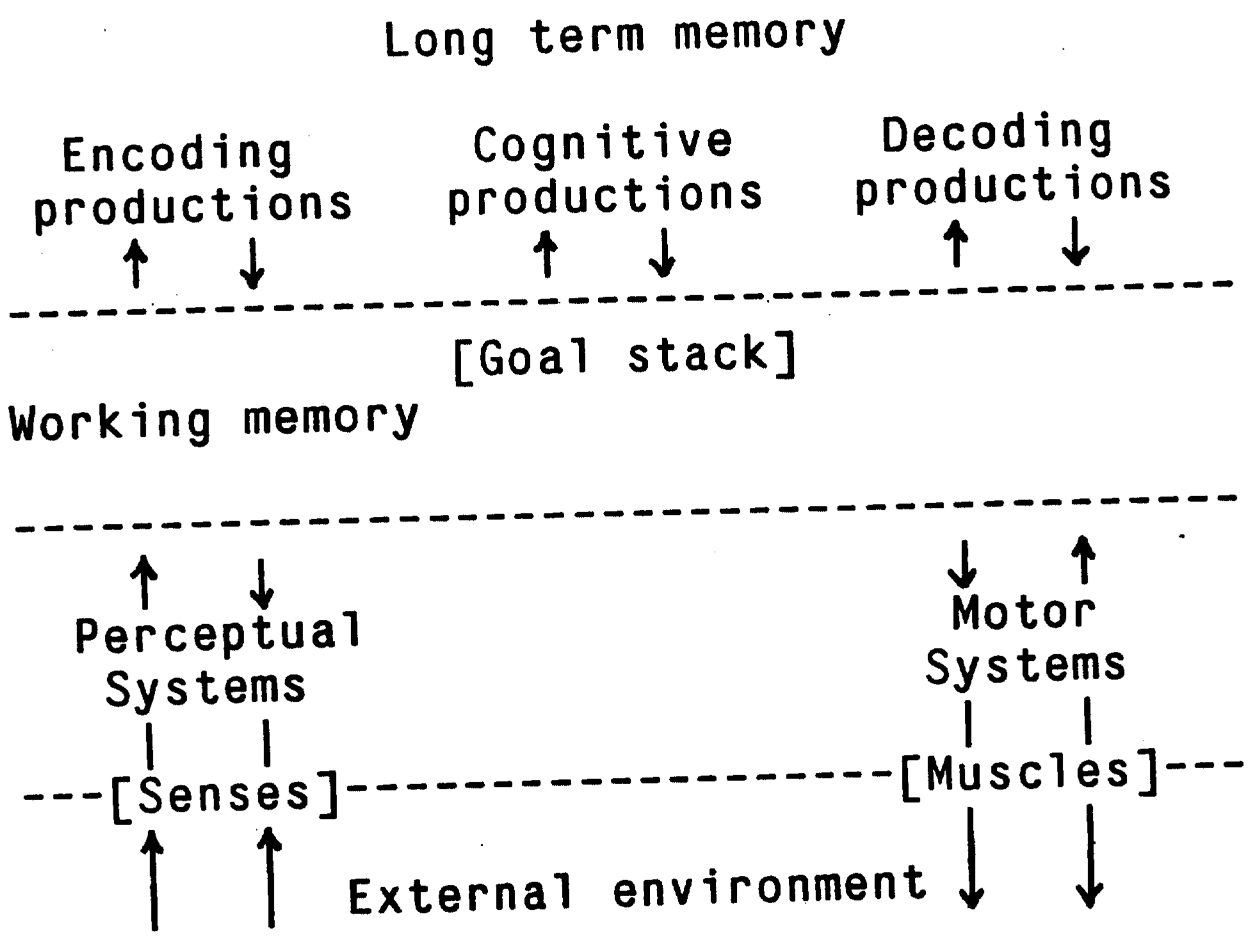
CHUNKING IN THE BLOCKS WORLD



TOTAL COGNITIVE SYSTEM

Brief overview now, more later

Basic concern — To get interface right



Long term memory

Encoding productions

Cognitive productions

Decoding productions

[Goal stack]

Working memory

Perceptual Systems

Motor Systems

---[Senses]---------------------[Muscles]---

External environment

In terms of peformance

[P ⟶ E] ⟶ C ⟶ [D ⟶ M]

In terms of structure and learning

[P] ⟶ [E ⟶ C ⟶ D] ⟶ [M]

R1-SOAR: CONFIGURATION TASK

R1 expert system (McDermott, 1980; DEC)

Input: An order for a Vax computer (a dozen items)

- Processor, bus, primary memory, disks, graphics, ...

Output: Information to assemble the system (ten pages)

- Filled out and verified order
- Spatial layout in cabinets with all connections

Take into account many factors

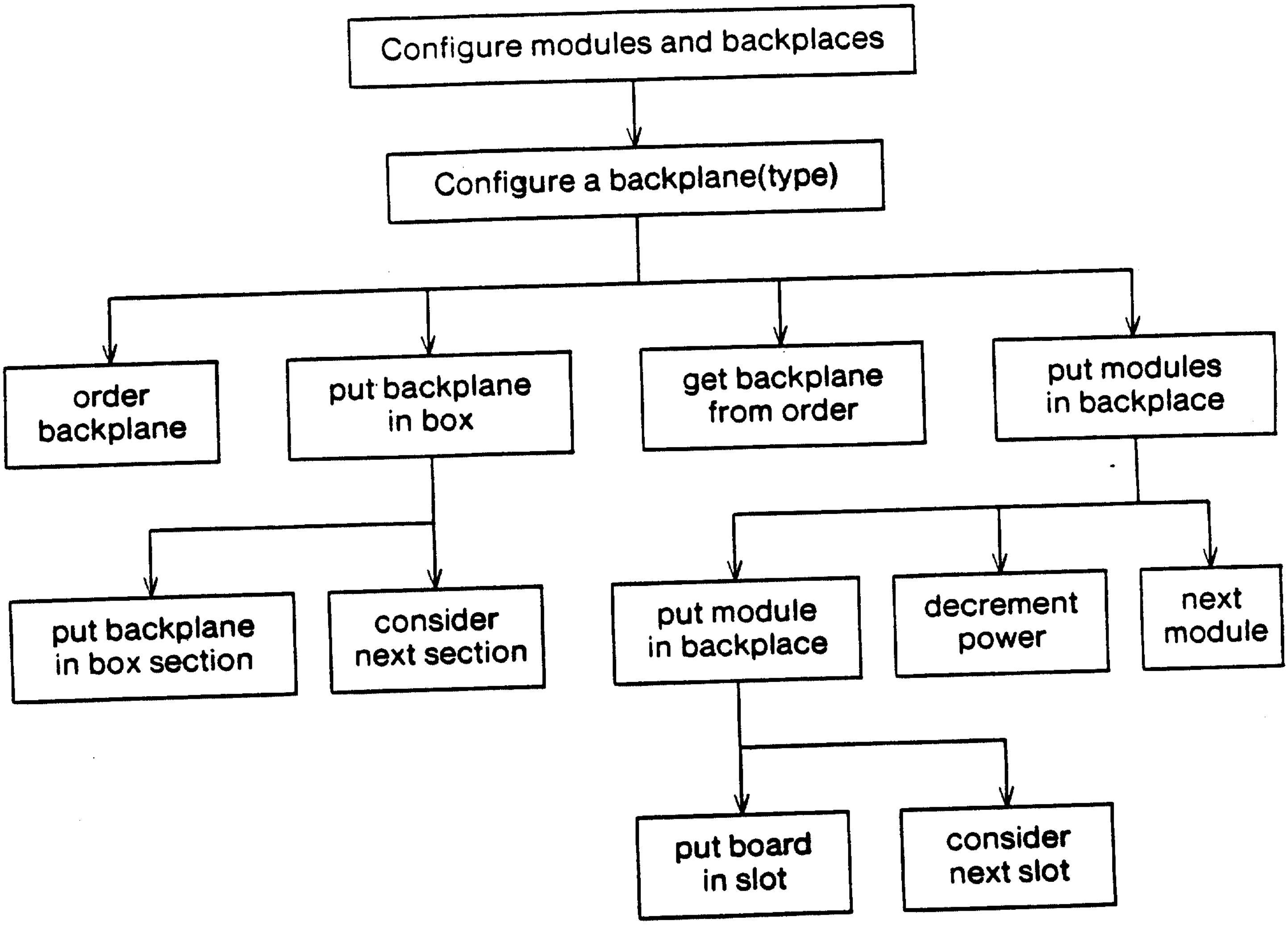
- Cost of components, power demands, cable lengths, ordering on bus, component compatibilities, ...

R1 characteristics (1984)

- 3300 Ops5 productions
- 10000 components (data base)
- About 1000 production cycles for a typical task

R1-SOAR: PROBLEM SPACES

Second version of R1-Soar



- Configure modules and backplaces
  - Configure a backplane(type)
    - order backplane
    - put backplane in box
      - put backplane in box section
      - consider next section
    - get backplane from order
    - put modules in backplace
      - put module in backplace
        - put board in slot
        - consider next slot
      - decrement power
      - next module

PERFORMANCE AND LEARNING ON R1-SOAR

	No Learning	During Learning	After Learning
Base [232]	1731	485 [+59]	7 [291]
Partial [234]	243	111 [+14]	7 [248]
Full [242]	150	90 [+12]	7 [278]

Tasks

Base: No search-control knowledge
Partial: Two key search-control rules
Full: Search control equivalent to R1's

Units

Decision cycles (e.g., select operator)
[numbers of rules]

DESIGNER-SOAR: ALGORITHM DESIGN

Steier (1986)

Original system: Designer (Kant, Newell, Steier)

- Designer-Soar is to complete and extend Designer
- Target is design of convex hull

Major problem spaces:

- Algorithm Design (top level)
  - States: data flow descriptions of algorithms
  - Operators: modify descriptions, focus attention
- Developmental Evaluation
  - States: algorithm descriptions with data
  - Operators: execute descriptions on data
- Application Domain
  - States: domain objects (sets, figures)
  - Operators: modify domain objects

DESIGNER-SOAR: SIMPLE EXAMPLE

Intersection

Given two sets, produce set of common elements

Memory to hold one input set

→{M1}→

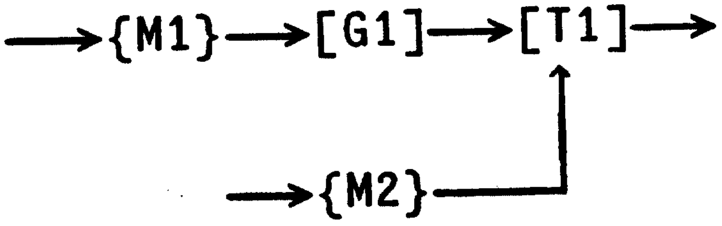
Generator to generate set elements

→{M1}→[G1]→

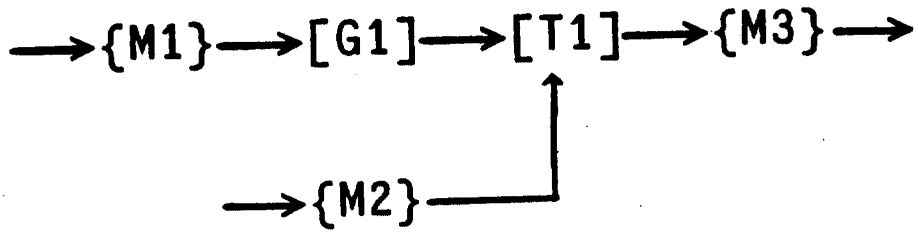
Test to check if element is in other set

→{M1}→[G1]→[T1]→

Memory to hold second input set for test



Memory to build output set



CYPRESS-SOAR: LEARNING ALGORITHM DESIGN

Steier (1986)

Design-level of Cypress (D. Smith, 1986)

Algorithm design space (partial algorithms)

Logical-inference space (assertions) — Not incorporated

	Search control	Insertion sort	Merge sort	Quicksort full spec	Quicksort bad spec
NO LEARN	Minimal	303	342	476	1132
	Full	140	140	140	266
WHILE LEARN	Minimal	222	236	226	238
	Full	135	140	130	188

Across task transfer (minimal search control)

Prior learning			
Insertion-sort	20 7%	296 86%	421 88%
Mergesort	269 89%	20 6%	417 87%
Quicksort	273 90%	292 85%	20 4%

SUMMARY OF TASKS

Many small and modest tasks (21), many methods (19)
- Eight puzzle, Tower of Hanoi, Waltz labeling
- Dypar (NL parsing), Version spaces, Resolution TP
- Generate & test, Hill climbing, Means-ends analysis
- Constraint propagation

Larger tasks (some in progress)
- R1-Soar: 3300 rule industrial expert system (25%)
- Neomycin-Soar: Revision of Mycin
- Designer-Soar: Algorithm discovery
- Cypress-Soar: Divide-&-conquer algorithm designer
- (Coder-Soar: Algorithms to code)
- (Weaver-Soar: VLSI router)

Learning (chunking)
- Learns on all tasks it performs
  - Learns search control, operators, spaces
  - Improves with practice, transfers to other tasks
- Explanation-based generalization
- Outside guidance (by chunking)
- Abstraction planning (by chunking)
- Constraint compilation (by chunking)

Task acquisition
- Builds spaces from external specs (by chunking)

HOW DOES SOAR APPROXIMATE A KL SYSTEM

1. Computationally universal

   Necessary, but not deal with real time constraint

2. Production systems

   Real time via recognition

   Abandon fixed conflict resolution

3. Decision process

   Open to quiescence to get all that is available

3. Impasses

   Seek knowledge whenever it is not available

   Never rest on apriori fixed finite mechanism

   Errors are due to knowing the wrong thing

4. Chunking

   Continually convert slow processes to fast ones

Issues — Sharing knowledge, scope of chunking

MAPPING SOAR INTO HUMAN COGNITION

- Productions ~10 ms
  - Symbol system (access and retrieve)
  - Recognition system (content addressed)
  - Parallel operation
  - Involuntary
  - Unaware of individual firings
  - Duration: Match a function of complexity
    - (Should be simpler match than Ops5, possibly)

- Decision cycle ~100 ms
  - The smallest deliberate act
  - Accumulates knowledge for an act and decides
  - The smallest unit of serial operation
  - Involuntary (exhaustive)
  - Awareness attends decision (products, not process)
  - Duration: Longest production chain (to quiescence)

- Primitive operators ~1 s
  - Serial operation
  - Primitive observable thinking acts
  - Duration: Sequence of decision cycles (2 minimum)
  - Goal-oriented

- Goal attainments ~10 s
  - Smallest unit of goal attainment
  - Smallest non-primitive operators
  - Smallest unit of learning (chunking)

SOAR AND THE SHAPE OF HUMAN COGNITION #1

Does Soar have the right qualitative shape?

1. Has general features derived from real-time constraint

   Symbol system, automatic/controlled behavior,

   recognition-based, fast-read/slow-write,

   continual shift to recognition (learns from experience)

2. Behaves intelligently

   But is not completely rational (only approximates KL)

3. Goal oriented

   But not just because it has learned goals

   Goals arise out of its interaction with environment

4. Interrupt driven

   Depth-first local behavior, progressive deepening

5. Default behavior is fundamentally adaptive

   Does not have to be programmed to behave

SOAR AND THE SHAPE OF HUMAN COGNITION #2

6. Serial in midst of parallelism

    Autonomous behavior (hence an unconscious)

7. Recognition is strongly associative

    Does not have access to all that it knows

    Remembering can be a problem — can work at it

8. Not know how it does things

    Learned procedures are non-articulatable

        Chunking accesses WM trace, not productions

    Can work interpretively from declarative procedures

9. There is meta-awareness or reflection

    Can step back and examine what it is doing

10. Indefinitely large knowledge

11. Aware of large amounts of immediate detail

    But focused, with a penumbra

ISSUES AND LACUNA IN SOAR

1. Things not in Soar 4.4, but coming in Soar 5
   - P-E-C-D-M
     - Full development of perceptual mechanisms
     - Full development of motor system
   - Single state principle
   - Less powerful match (no equality testing)
2. Default behavior is not quite all in the architecture
   - Currently default productions avoid impasse pits
3. Not demonstrated yet — although consonant
   - Flexibility to the point of non-brittleness
   - Full scope of learning
4. Missing major aspects (Require structural additions?)
   - Emotion, dreaming, imagery, ...

SUMMARY:
SYMBOLIC PROCESSING FOR INTELLIGENCE

A specific architecture for cognition — Soar

The central construct of a unified theory of cognition

Focus is on functionality — being intelligent

1. Architecture for central cognition

   Problem spaces, productions, goals

   Decision cycle, impasses

2. Learning from experience

   Chunking, at the production level

3. The total cognitive system (P-E-C-D-M)

   Encoding & decoding — Uncontrolled productions

4. Functionality and ability

   Incorporates most mechanisms of intelligence

5. Qualitative aspects of human cognition

SOAR DESIGN AND THE MULTIPLE CONSTRAINTS

1. Behave flexibly — Yes
2. Adaptive (rational, goal-oriented) behavior — Yes
3. Operate in real time — Yes
4. Rich, complex, detailed environment
   Perceptual detail — Interface only
   Use vast amounts of knowledge — Yes
   Motor control — Interface only
5. Symbols and abstractions — Yes
6. Language, both natural and artificial — No
7. Learn from environment and experience — Yes
8. Acquire capabilities through development — No
9. Live autonomously within a social community — No
10. Self awareness and a sense of self — No
11. Be realizable as a neural system — No
12. Arise through evolution — No

REFERENCES, LECTURE 4

On Soar

J. Laird, P. Rosenbloom & A. Newell, "Soar: An architecture for general intelligence", *Artificial Intelligence*, 1987 (in press).
Copies of tech report in Harvard Psychology Library

J. Laird, P. Rosenbloom & A. Newell, "Chunking in Soar: The anatomy of a general learning mechanism", *Machine Learning*, vol. 1, 1986, pp. 11-46.

General references for Lecture 4

J. R. Anderson, *The Architecture of Cognition*, Cambridge MA: Harvard University, 1983.

J. Laird & P. Rosenbloom, "Mapping explanation-based generalization onto Soar", *Proceedings of AAAI-86*, National Conference on Artificial Intelligence, Menlo Park CA: AAAI, 1986.

SOAR AND ACTIVATION

1. Multiple roles of activation
    - Determines access path (Quillian, many others)
    - The representational media (connectionism)
    - Determines processing rate (Anderson)
2. General concern about doing real tasks
    - Activation systems still good only for analysis
3. Specific theoretical concern about learning
    - Crypto-knowledge constraint
        - Tried activation-based productions in Xaps2
4. Yielding interesting new forms of representation media
    - Properties of continuity, coarse coding
5. The issue of approximation
    - Activation cannot be critical as duration increases
    - Soar as approximation to an activation-based system

SOAR AND SCHEMAS

1. Knowledge is organized — A basic truth

2. Schemas are a data-structure solution to this

   "Real schemas" — The kind we program

   They are rigid and unadaptive

   Large-grain-size argument is misplaced

      Because it confuses structure with behavior

3. Knowledge organization in Soar

   The declarative representation

      Attributes and values (as opposed to lists)

      No inheritance, defaults, attached procedures

   Productions provide dynamic, complex semantic net

   Inheritance comes in the elaboration phase

   Attached procedures realized by impasses

Key: Problem spaces are an action-oriented encoding

SOAR AND ACT*

Differences

	ACT*	Soar
Memory	Declarative procedural	Procedural
Higher organization	None	Problem spaces
Goals	Deliberate learned	Impasse created
Control	Activation variable rate	All-or-none cycles
Learning	Compilation composition proceduralization Tuning strengthening generalization discrimination	Chunking

USING KNOWLEDGE FOR CONTROL

Standard AI scheme: Methods + Selection

- Method = Procedure + Deliberate-subgoals

Weak methods are basic to intelligent action

- Generate-test, hill climbing, progressive deepening, means-ends analysis, minimax, constraint propagation

Soar uses <u>implicit methods</u> for the weak methods

- Implicit method = Conjoining independent heuristics

Major implications

- Knowing leads directly to doing
- No need to learn method control structure — Emerges

Permitted by two conditions

- Soar — Problem spaces and production systems
- Weak method — Search related and simple