Evolution of Soar

based on "The Evolution of the Soar Architecture" Mind Matters, 1992
Laird and Rosenbloom

John E. Laird

University of Michigan

June 26, 2003

23rd Soar Workshop

Outline

- Revisit Soar versions 1-8
- Technical and research ideas
- Representative systems – knowledge
  - Representative applications and research

Research Methodology

1. Pick basic principles to guide development
2. Pick desired behavioral capabilities
3. Make design decisions consistent with above
4. Build/modify architecture
5. Implement tasks
6. Evaluate performance

Principles

- Draw from “intelligence” in humans and machines
- Start with architecture
  - The fixed mechanisms that support general intelligence
- Attempt to create a uniform, universal architecture
  - Use task independent representations and processes.
  - Architectural mechanisms shouldn’t constrain system to specific approaches for all problems
- Minimal architectural mechanisms

Desired Behavioral Capabilities

- Interact with a complex world - limited uncertain sensing
- Respond quickly to changes in the world
- Use extensive knowledge
- Use methods appropriate for tasks
- Goal-driven
- Meta-level reasoning and planning
- Coordinate behavior and communicate with others
- Learn from experience
- Integrate above capabilities across tasks
- Behavior generated with low computational expense

Pre-Soar

- Logic Theorist (LT) (1955)
  - First symbolic problem solver with heuristics
- GPS (1958)
  - Means-Ends Analysis and Recursive Goals
- Problem Spaces (1965)
  - Uniform Task Structure
- Production Systems (1967)
  - Uniform, Incremental, Context-sensitive Knowledge Representation
- Weak Methods (1969)
  - Organization of General Control Knowledge

Soar 1 – 1982
Goals

- Develop an architecture that supports multiple methods
  - Use different methods for different tasks
  - Use different methods for different subtasks
  - Methods determined by available knowledge

Soar 1: Approach

- Develop an AI architecture that supports problem spaces
  - Explicit representation of goals, problem spaces, states, operators
  - All tasks cast as decision making in a problem space
  - All knowledge supports problem space functions
- Use a parallel production system for long-term memory
  - Working memory contains current and proposed
    - goals,
    - problem spaces,
    - states,
    - operators.
  - Productions encode problem space knowledge
    - Propose object, vote on objects, apply operators.
  - Decision procedure tallies votes for selecting objects

Soar 1

Production Memory

Conditions → Actions

Conditions → Actions

Conditions → Actions

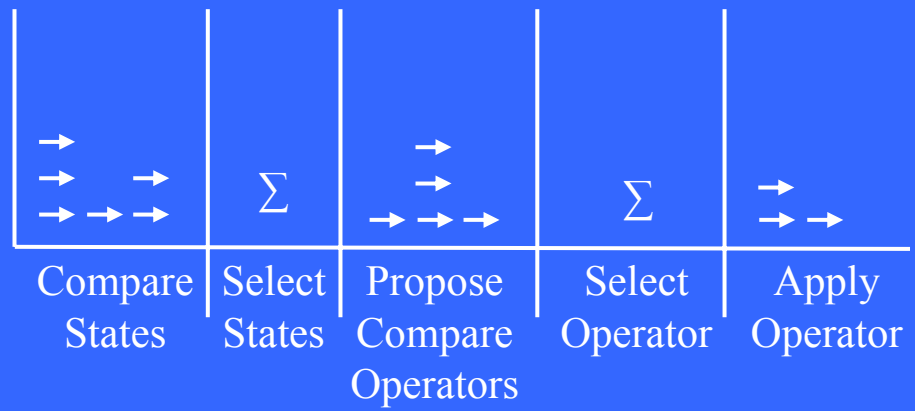
Conditions → Actions

Working Memory

Type:	Slot	Stock
Goal	[G1]	G2, G3
Problem Space	[P2]	P4
State	[S13]	S10, S11, S1
Operator	[ ]	O32, O33

Process

Elaboration, Decision, Application Cycle



Working memory is a set, not a graph

Special slot for selected goal, problem space, state, operator

Votes are tallied to select current object

Contrast with Other Approaches

- STRIPS – for each operator, single representation of
  - operator preconditions and actions
- GPS – for each operator, single representation of
  - operator preconditions, actions
  - table of connections
- Soar
  - Operators are first-class objects
  - Independent knowledge for
    - Proposal
    - Selection
    - Application
  - Can be disjunctive, conditional

Weak Methods in Soar 1

- By adding rules about task, could change methods
  - Blind search
  - Avoid duplicate states
  - Heuristic search – task specific heuristics
  - Means-ends analysis
  - Hill climbing
  - Steepest ascent hill climbing
  - Breadth-first search
  - Depth-first search
  - Best-first search
- Not a big switch where method is selected explicitly

Soar 1 Achievements

- Unified problem spaces and production systems
- Basis for Universal Weak Method
  - Separate representation of control knowledge
  - Decisions based on dynamic integration of knowledge
- Systems built: Variety of simple puzzles and toy tasks
- Users: 1
- Implemented in XAPS 2
  - Rosenbloom's eXperimental Activation Production System

Soar 1 Shortcomings

- No subgoals
- No meta-level reasoning
- Voting results may not reflect knowledge
  - More votes ≠ more evidence, more knowledge
  - Limited expressability of knowledge – no partial orders.

Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation

Weak Methods

Production Systems

Symbol Systems

Heuristic Search

Problem Spaces

Soar1 - 1982

Universal Weak Method

Toy Tasks

Xaps Lisp

Soar 2 – 1983

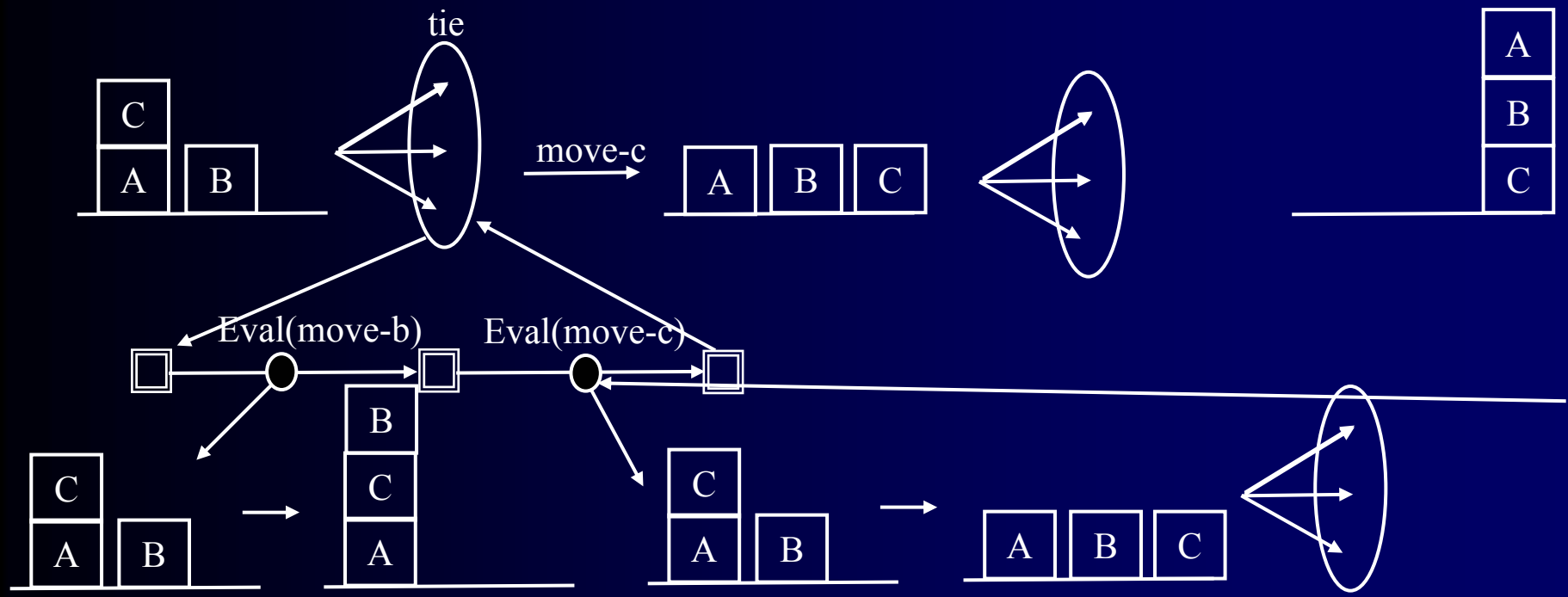
- Any goal principle
  - System can generate any and all types of goals
- Universal subgoaling
  - Single mechanism for creating all types of goals
- Automatic subgoaling
  - All subgoals generated automatically
- Preference-based decision procedure

Soar 2 Approach

- Symbolic preference scheme replaces voting
  - Acceptable, reject, better, worse
- Detect inability to make decisions automatically:
  - Impasses: tie, conflict, no-change in decision
- Create subgoal automatically to resolve impasse
  - Recursively use problem spaces in subgoal
  - Subgoal parameters and results are determined dynamically by problem solving
- Working memory becomes graph structure rooted in goals

Soar 2: Selection Space

- Meta-level reasoning with evaluation
- Basis of many weak methods
- Depth-first search, mini-max, alpha-beta, iterative deepening, progressive deepening, …



Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation
Preferences	Subgoals	Soar2 - 1983	Universal Subgoaling	R1-Soar Dypar-Soar	OPS5 Lisp
Weak Methods	Production Systems	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Symbol Systems	Heuristic Search	Problem Spaces			

Soar 2 Shortcomings

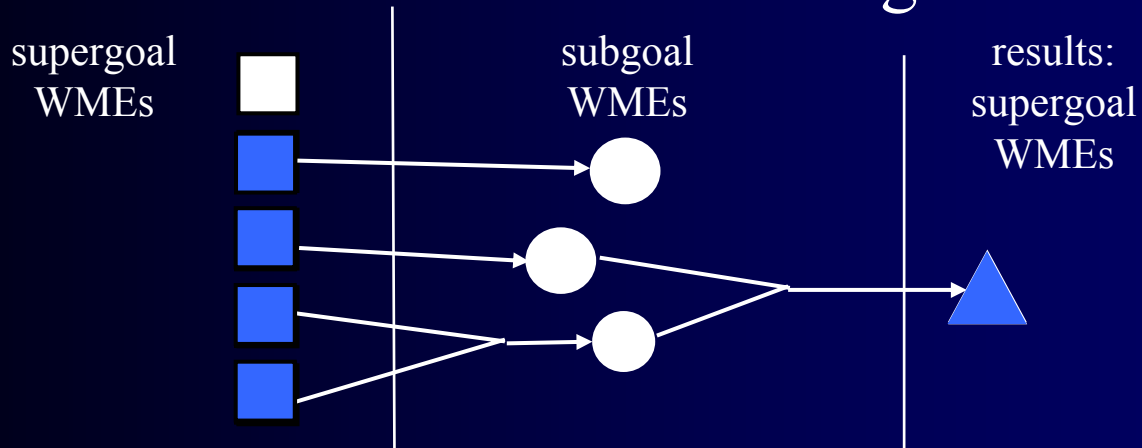
- No learning
- Must resolve same impasse multiple times during a task

Soar 3 – 1984

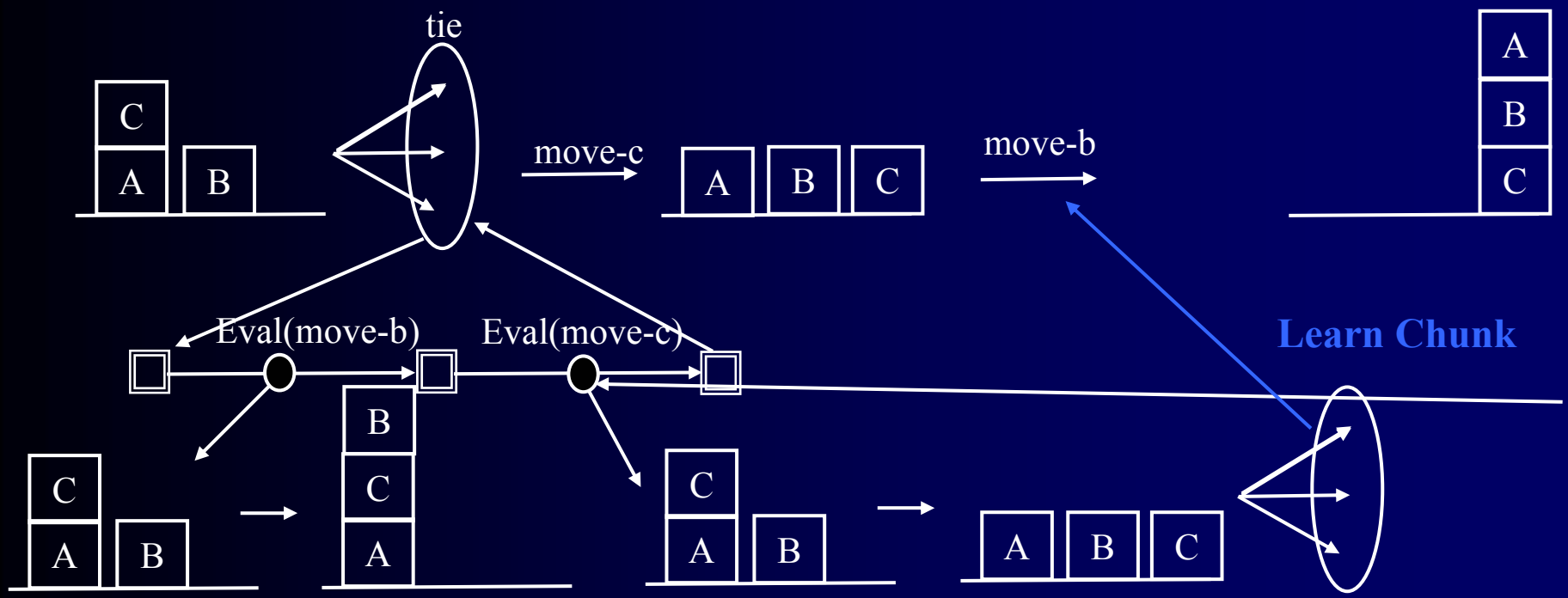
- How can learning be integrated with problem solving?
  - Learn when knowledge is incomplete: impasses
  - Learn when knowledge becomes available: results
  - Connect situation where learning is needed to when it is available
  - Knowledge transfer depends on similarity of situation

Soar 3 Approach

- Inspired by "chunking" as developed in XAPS.
- A rule is learned that summarizes processing in a subgoal.
- Conditions based on working memory elements tested by productions fired in subgoals.
  - Later restricted to just those tested on path to results.
  - Also inspired by goal-regression learning – Mitchell, DeJong
- Actions are based on results of subgoal.



Soar 3: Chunking Example



Within Task Transfer: chunk learned in subgoal applies to avoid future search

Across Task Transfer: chunk learned in one problem transfers to another

Soar 3 Results

- General learning method integrated with problem solving.
  - Learns incrementally and continually on all tasks.
  - Learns from both success and failure.
  - Learns a variety of types of knowledge:
    - Selection knowledge, operator application, operator creation, state refinement, problem formulation
  - Automatically converts deep knowledge to shallow knowledge
    - R1-Soar
- Users: 8

Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation
Chunking	Soar3 - 1984	General Learning	R1-Soar	
Preferences, Subgoals	Soar2 - 1983	Universal Subgoaling	R1-Soar, Dypar-Soar	OPS5 Lisp
Weak Methods, Production Systems	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Symbol Systems, Heuristic Search, Problem Spaces				

Soar 4 – 1986

- First version prepared for external release
  - Manual published, ported to Common Lisp, first Soar Workshop
- Explosion in applications and users
  - > 50 by 1988
- Explosion in types of learning and problem solving
- Newell proposes Unified Theory of Cognition

Soar 4 Results
Knowledge-based Systems

- Algorithm design (Designer-Soar, Cypress-Soar)
- Medical Diagnosis (NeoMycin-Soar, Red-Soar)
- Production Line Scheduling (Merl-Soar)
- Chemical Process Modeling
- Natural Language Understanding (NL-Soar)
- Intelligent Tutoring (ET-Soar)

Unified Theories of Cognition

Allen Newell, 1989:

"I mean a single set of mechanisms for all of cognitive behavior."

- Problem solving, decision making, routine action
- Memory, learning, skill
- Perception, motor behavior
- Language
- Motivation, emotion
- Imagining, dreaming, daydreaming, …"

"Our ultimate goal is a unified theory of human cognition. This will be expressed, I have maintained, as a theory of the architecture of human cognition–that is, of the fixed (or slowly varying) structure that forms the framework for the immediate processes of cognitive performance and learning."

Soar 4 Results: Unified Theory of Cognition (1987)

- Immediate Reasoning Tasks
- Syllogisms
- Balance Beam
- Cryptarthmetic
- Towers of Hanoi
- Transcription Typing
- Instruction for simple tasks
- Verbal learning
- Series completion

Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation
	Soar4 - 1986	UTC	ET-Soar NL-Soar	External Release
Chunking	Soar3 - 1984	General Learning	R1-Soar	
Preferences, Subgoals	Soar2 - 1983	Universal Subgoaling	R1-Soar Dypar-Soar	OPS5 Lisp
Weak Methods, Production Systems	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Symbol Systems, Heuristic Search, Problem Spaces				

Soar 4 Shortcomings

- Limited interaction with external environments
- Excessive copying of state structures during operator application

Soar 5 – 1989

- Add interaction with external environments
- Improve efficiency by eliminating state copying

Soar 5 Approach

- Modify problem space computational model
- Only a single state per goal
  - State changes through operator application or perception
- Operator productions destructively modify current state
  - Retractable entailments: i-support
  - Automatic classification of rules into entailments/destructive
    - i-support vs. o-support
- Operators stay selected until terminated with a reconsider preference: @
- Justifications built for results
  - Not required in Soar 4 because everything o-supported

Soar 5 Results

- Integrates interaction, reaction, planning, learning
- Four level architecture
  - Motor programs: innate – not subject to cognition
  - Parallel Production System: Reflexes
  - Operators: Deliberation
  - Subgoals: Planning and Hierarchical Reasoning
- Used worldwide at over 15 sites, 100 researchers

Soar 5 UTC Applications

- Number conservation (Q-Soar)
- Concept learning (SCA)
- Visual Attention (NOVA)
- Sentence parsing (NL-Soar)
- Learning Physics (Dyna-Soar)
- Browsing (Browser-Soar)
- Highly interactive tasks (HI-Soar)
- Decision modeling (NTD-Soar)
- Learning task-action mappings (TA-Soar)

Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation
Single State	Soar5 - 1989	External Tasks	HI-Soar Hero-Soar	
	Soar4 - 1986	UTC	ET-Soar NL-Soar	External Release
Destructive Operators				
Chunking	Soar3 - 1984	General Learning	R1-Soar	
Preferences	Soar2 - 1983	Universal Subgoaling	R1-Soar Dypar-Soar	OPS5 Lisp
Subgoals				
Weak Methods	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Production Systems				
Symbol Systems				
Heuristic Search				
Problem Spaces				

Soar 5 Shortcomings

- Software rot
  - Difficult to maintain and extend
  - Code is > 10 years old
- Significant efficiency problems with big systems and long runs
  - Having impact on initial versions of Air-Soar

Soar 6 – 1992

- Significantly improve efficiency, portability, correctness maintainability

Soar 6 Approach

- Port to C
  - Rewrote from scratch by Bob Doorenbos
- Provide efficiency for large highly interactive systems.
  - 8-10 times faster than Soar 5 for medium size tasks (1000 rules)
  - 20-80 times faster for large tasks and long runs.
  - Dispatcher-Soar learning > 100,000 chunks without significant slowdown

Soar 6 Applications

- Cognitive Modeling, many with learning
  - Play Super Marios (HI-Soar)
  - Solve Electric/Magnetic Physics Problems (EFH-Soar)
  - Natural Language Processing (NL-Soar)
  - Medical Diagnosis using abduction (Red-Soar)
  - Air Traffic Controller (ATC)
- Learning Systems
  - Learning by Instruction (Instructo-Soar)
  - Learning from interaction with environment (IMPROV)
  - Symbolic Category Learning (SCA-2)
- Performance Systems
  - Fly SGI flight simulator (Air-Soar)
  - First versions of military simulation agents (TacAir-Soar, RWA-Soar, Debrief)
  - Teamwork (STEAM)
  - Training and Instruction (STEVE)

Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation
	Soar6 - 1992	High Efficiency	Air-Soar Instructo-Soar	C
Single State Destructive Operators	Soar5 - 1989	External Tasks	HI-Soar Hero-Soar	
	Soar4 - 1986	UTC	ET-Soar NL-Soar	External Release
Chunking	Soar3 - 1984	General Learning	R1-Soar	
Preferences Subgoals	Soar2 - 1983	Universal Subgoaling	R1-Soar Dypar-Soar	OPS5 Lisp
Weak Methods Production Systems	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Symbol Systems Heuristic Search Problem Spaces				

Soar 6 Shortcomings

- No runtime environment in C like in Lisp
- Difficult to integrate Soar with other applications

Soar 7 - 1996

- Make easier to:
  - create runtime tools
  - connect to external environments
  - run multiple agents
- Integrated Soar with Tcl/Tk

- Two companies use Soar
  - ERS: ExpLore Reasoning Systems
    - KB Agent
  - Soar Technology, Inc.

Soar7 Applications

- Final TacAir-Soar, RWA-Soar
  - Attention modeling
- Teamwork applications using STEAM:
  - RWA-Soar, Robo-cup, Electric Elves (organization management and teamwork)
- Learning multi-tasking (EPIC-Soar)
- Modeling air-traffic controllers
  - Working memory decay model
- Vision-processing (Vision-Soar)
- Language Generation: (LG-Soar & continued work on NL-Soar)
- More games:
  - Descent-Soar, Quake-Soar
  - PACMAN
- Mission rehearsal bots
  - HTNs
  - Emotion Modeling
  - Social Interactions
- Organization modeling (Country Chaos)

Contributing Ideas	Soar Version	Major Results	Example Systems	Implementation
	Soar7 - 1996	Improved Interfaces	TacAir-Soar EPIC-Soar	TCL/Tk Wrapper
	Soar6 - 1992	High Efficiency	Air-Soar Instructo-Soar	C
Single State, Destructive Operators	Soar5 - 1989	External Tasks	Air-Soar Hero-Soar	
	Soar4 - 1986	UTC	ET-Soar NL-Soar	External Release
Chunking	Soar3 - 1984	General Learning	R1-Soar	
Preferences, Subgoals	Soar2 - 1983	Universal Subgoaling	R1-Soar Dypar-Soar	OPS5 Lisp
Weak Methods, Production Systems	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Symbol Systems, Heuristic Search, Problem Spaces				

Soar 7 Shortcomings

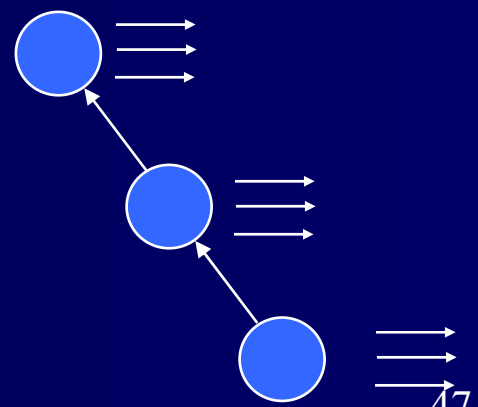
- Race conditions between rules and across subgoals
- Inconsistency in reasoning across subgoals possible
- Learning could create rules that would never fire
- Output could be inconsistent with selected operators

Soar 8 - 1999

- Fix problems with interactions with external environments
  - Across goal consistency
    - Goal dependency sets
    - Iterative instead of parallel processing of goals
    - Non-contemporaneous chunks
  - Change operator selection to be i-supported
    - Eliminate reconsider preference
  - Change Decision Cycle

Processing Across Substates

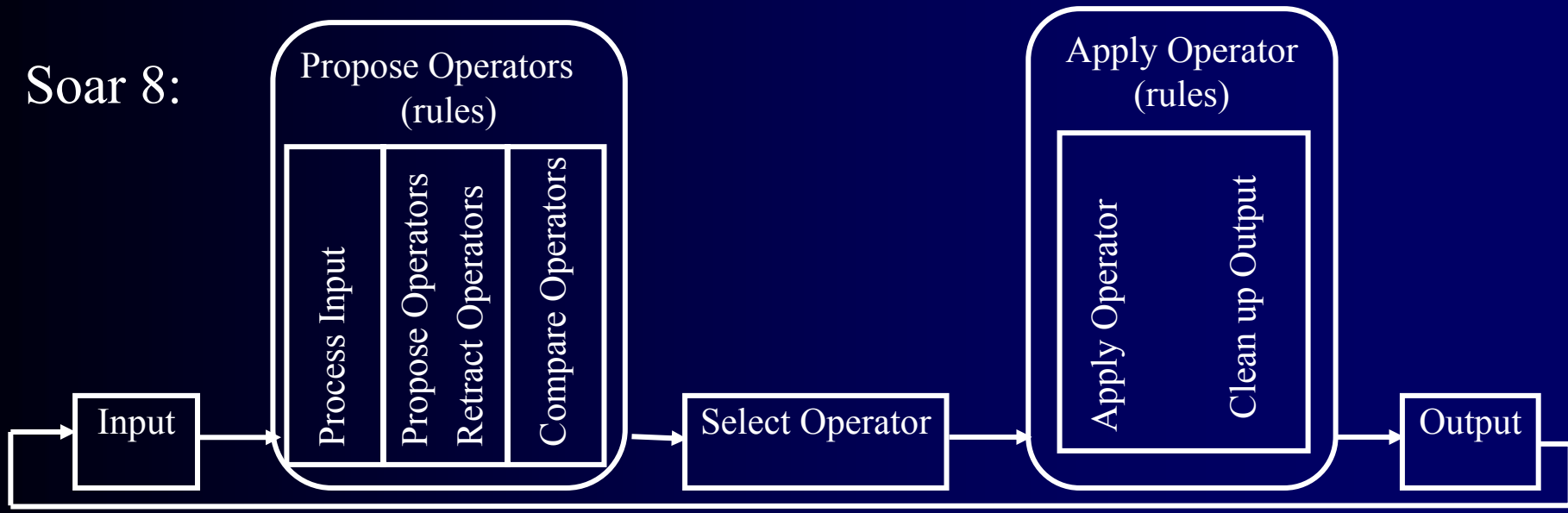
- Problem:
  - Rules can fire in substates even though impasse is about to be resolved
  - Run-away substate can generate results that are invalid
- Approach
  - Cascade rules from oldest to newest substate
  - Remove substates if impasse is resolved
  - Recompute match set after each substate processed
- Implications
  - Avoids firing rules that will be irrelevant
  - Avoids some race conditions



Soar's Decision Cycle

Soar 7

Soar 8:



Soar 8 Applications

- External Interaction
  - Play Quake (Quakebot)
  - Computer game non-player characters (Haunt II)
  - Adversaries for military simulations (MOUTBot)
  - Unmanned Air Vehicle (UAV) control
  - …
- Social Interaction Agents
- …

Contributing Ideas		Soar Version	Major Results	Example Systems	Implementation
Goal Dependency	Decision Cycle	Soar8 - 1999	Substate Coherence	MOUTBOT QuakeBot	SGIO
		Soar7 - 1996	Improved Interfaces	TacAir-Soar RWA-Soar	TCL/Tk Wrapper
		Soar6 - 1992	High Efficiency	Air-Soar Instructo-Soar	C
Single State	Destructive Operators	Soar5 - 1989	External Tasks	Air-Soar Hero-Soar	
		Soar4 - 1986	UTC	ET-Soar NL-Soar	External Release
	Chunking	Soar3 - 1984	General Learning	R1-Soar	
Preferences	Subgoals	Soar2 - 1983	Universal Subgoaling	R1-Soar Dypar-Soar	OPS5 Lisp
Weak Methods	Production Systems	Soar1 - 1982	Universal Weak Method	Toy Tasks	XAPS 2 Lisp
Symbol Systems	Heuristic Search	Problem Spaces			

What's Next?

- Soar 8.5
  - Bug fixes
  - Numeric indifferent preferences
- Soar 9
  - Working memory activation
  - New architectural learning mechanisms
    - Reinforcement learning
    - Episodic learning?
  - Rule decay?

Biggest Surprises

- Lack of buy in AI to Soar's best ideas
  - Open decision making for operator selection
- Success of niche AI
  - Specialized techniques for specific problems
- Difficulty of finding applications that require human-level AI
  - Military Simulations
  - Computer Games

Where does Soar sit today?

- Best symbolic architecture for building complex, knowledge-rich performance systems.