TABLE OF CONTENTS

AUTHOR INDEX

26TH SOAR WORKSHOP
ANN ARBOR, MICHIGAN
MAY 22-26, 2006

WEDNESDAY, MAY 24, 2006

8:00-9:00	60		*Registration and Bagels*
9:00-10:15	75		*Cognitive Modeling*
	15	John Laird	Introductions
	30	Bob Marinier	Unifying Cognitive Functions and Emotional Appraisal
	15	Deryle Lonsdale	Update on Soar-based language processing
10:15-10:45	30		*Break*
10:45-12:00	75		*Learning and Memory*
	15	Andrew Nuxoll	Steps toward a Domain Independent Episodic Memory
	15	Yongjia Wang	Integrating Semantic Memory in Soar
	15	Shelley Nason	Hierarchical Reinforcement Learning in Soar
	15	Nick Gorski	Methods for Transfer Learning Using Soar
	15		Discussion
12:00-1:15	75		*Lunch*
1:15-2:45	90		*Learning and Advanced Capabilities*
	5	David Ray	Soar Technology Community Liaison
	5	Alan Vayda	Self Introduction
	5	Jong Kim	Soar FAQ
	15	Elyon DeKoven	Participatory scenario design and simulation on the ICF testbed
	30	Joseph Xu	ORTS: A Case Study of Multi-Tasking in Soar
	15	Sam Wintermute	Visual Attention for a Real-Time Strategy Game Game
	15		Discussion
2:45-3:15	30		*Break*
3:15-5:00	105		*Advanced Capabilities*
	30	Scott Lathrop	Incorporating Visual Imagery into a Cognitive Architecture
	30	Jonathan Beard	Spatial & Temporal Reasoning (SPAT-R)
	25	Brian Magerko	Player Modeling in IDA
	20		Discussion

THURSDAY, MAY 25, 2006

9:00-10:15	75		*Applications*
	20	Brian Magerko	Interactive Storytelling Architecture for Training (ISAT)
	15	Jim Rosbe	Soar Technology Update
	25	Brian Stensrud	IF-Soar: An Autonomous Fire Direction Center for Indirect Fire Training
	15	Jonathan T. Beard	Joint Fires & Effects Training System (JFETS)
10:15-10:45	30		*Break*
10:45-12:00	75		*Soar Architecture*
	15	John Laird	A Proposal For Changing Soar's Decision Procedure
	45	Douglas Pearson	What's New in Soar 8.6.2
	15		Discussion
12:00-1:15	75		*Lunch*
1:15-2:45	90		*Soar from an AI perspective*
	15	Scott Wallace	S-Assess: Self-Assessment with Soar
	15	Robert Wray	Relevance Estimation for Evidence Marshalling
	30	Randolph Jones	HLSR: Compiling to Soar
	15	Robert Wray	Modularity in Soar-based Applications: Practical Issues
	15	Jacob Crossman	SoarML: A Graphical Modeling Language for Agents
2:45-3:15	30		*Break*
3:15-4:30	75		*Control and Architecture*
	15	John Laird	Pyramid Problems in Soar & ACT-R
	15	Rick Lewis	Control and Metacognition: A View from EPIC, Soar and ACT-R
	30	Rick Lewis	Cognitive Constraint Modeling: An Alternative to Traditional Architectures
	15		Discussion

FRIDAY MAY 26, 2006

9:00-10:15	75		*Brain inspired computation*
	15	John Laird	BICA: Biologically-Inspired Cognitive Architecture
	60	Rick Granger	Principles of Brain Computation
10:15-10:45	30		*Break*
10:45-12:00	75		*Brain inspired computation*
	15	Lee Newman	Clusters, Symbols and Cortical Topography
	15	Yongjia Wang	Integrating Clustering and Semantic Memory in Soar
	30	Thad Polk	Biologically-Inspired Control in Problem Solving
	15	Discussion	

ATTENDEE INFORMATION

First Name	Last Name	Affiliation	Email Address
Thom	Bartold	Soar Technology	thom.bartold@soartech.com
Satinder	Baveja	University of Michigan	baveja@umich.edu
Jonathan T.	Beard	Soar Technology	beard@soartech.com
Vladimir	Bouchev	Soar Technology	vlad.bouchev@soartech.com
Alina	Chu	University of Michigan	achu@umich.edu
Scott	Colcord	Soar Technology	sacolcor@soartech.com
Steve	Cooke	The Boeing Company	steven.w.cooke@boeing.com
Karen	Coulter	University of Michigan	kcoulter@umich.edu
Dave	Crepps	The Boeing Company	david.crepps@boeing.com
Jacob	Crossman	Soar Technology	jcrossman@soartech.com
Andy	Dallas	Soar Technology	adallas@soartech.com
Grant	Degenhardt	The Boeing Company	grant.d.degenhardt@boeing.com
Elyon	DeKoven	Soar Technology	dekoven@soartech.com
Alex	Dow	HRL	
Cory	Dunham	Soar Technology	dunham@soartech.com
Susan	Eitelman	Soar Technology	seitelman@gmail.com
Robert	Gaimari	The MITRE Corporation	rgaimari@mitre.org
Anya	Getman	Harem Hills	agetman@haremhills.com
Nicholas	Gorski	University of Michigan	ngorski@umich.edu
Rick	Granger	University of California, Irvine	richard.granger@gmail.com
Scott	Hanford	Penn State University	sdh187@psu.edu
Katherine	Harding	Soar Technology	kate.harding@soartech.com
Echo	Harger	Soar Technology	echo.harger@soartech.com
Randy	Jones	Soar Technology	rjones@soartech.com
Jong W.	Kim	Penn State University	jongkim@psu.edu
Aaron	Kluck	Soar Technology	aaron.kluck@soartech.com
Frank	Koss	Soar Technology	koss@soartech.com
Adam	Krawitz	University of Michigan	akrawitz@umich.edu
Unmesh	Kurup	Ohio State University	kurup@cse.ohio-state.edu
John	Laird	University of Michigan	laird@umich.edu
Scott	Lathrop	University of Michigan	slathrop@umich.edu
Rick	Lewis	University of Michigan	rickl@umich.edu
Sean	Lisse	Soar Technology	lisse@soartech.com
Lyle N.	Long	Penn State University	lnl@psu.edu
Deryle	Lonsdale	BYU Linguistics	lonz@byu.edu
Brian	Magerko	Michigan State University	magerko@msu.edu
Bob	Marinier	University of Michigan	rmarinie@umich.edu
Ben	Medler	Michigan State University	medlerbe@msu.edu
Waseem	Naqvi	Raytheon	Waseem_Naqvi@raytheon.com
Shelley	Nason	University of Michigan	snason@umich.edu
Lee	Newman	University of Michigan	leenewm@umich.edu
William	Niebruegge	The Boeing Company -	william.g.niebruegge@boeing.com
Andrew	Nuxoll	University of Michigan	anuxoll@eecs.umich.edu
Douglas	Pearson	ThreePenny Software	doug@threepenny.net
Nicholas	Piegdon	Soar Technology	piegdon@soartech.com

Thad	Polk	University of Michigan	tpolk@umich.edu
Anne	Porbadnigk	University of Michigan	akporbad@umich.edu
David	Ray	Soar Technology	ray@soartech.com
Douglas	Reece	Soar Technology	douglas.reece@soartech.com
Jim	Rosbe	Soar Technology	rosbe@soartech.com
Thuy	Schnur	The Boeing Company	thuy.c.schnur@boeing.com
Bethany	Soule	Soar Technology	bethany.soule@soartech.com
Ann Marie	Steichmann	Soar Technology	annmarie.steichmann@soartech.com
Brian	Stensrud	Soar Technology	stensrud@soartech.com
Mike	Sullivan	The Boeing Company	mike.sullivan5@boeing.com
Glenn	Taylor	Soar Technology	glenn@soartech.com
Toby	Tripp	Soar Technology	ttripp@soartech.com
Alan	Vayda	Soar Technology	alan.vayda@soartech.com
Jonathan	Voigt	University of Michigan	voigtjr@gmail.com
Ryan	Wagoner	Michigan State University	wagonerr@msu.edu
Scott	Wallace	Washington State University Vancouver	swallace@vancouver.wsu.edu
Yongjia	Wang	University of Michigan	yongjiaw@umich.edu
Sam	Wintermute	University of Michigan	swinterm@umich.edu
Robert	Wray	Soar Technology	wrayre@acm.org
Joseph Z.	Xu	University of Michigan	jzxu@umich.edu

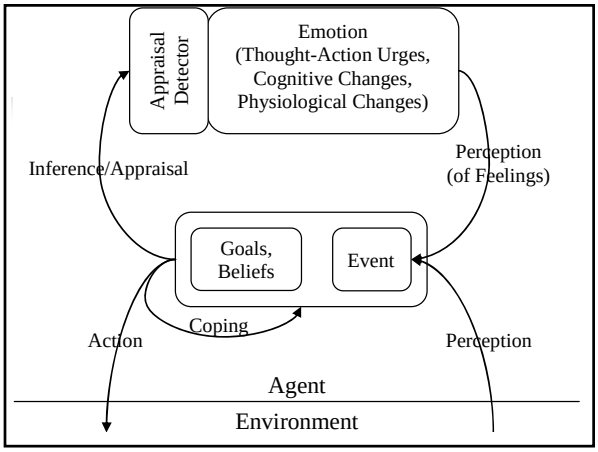
Unifying Cognitive Functions and Emotional Appraisal

Bob Marinier
John Laird
University of Michigan
26th Soar Workshop: May 24, 2006

Introduction

- Have independent theories of emotion and cognitive functions
  - Emotion: Appraisal Theory
    - Data without process
  - Cognitive Functions: Allen Newell's PEACTIDM
    - Process without data
- Each of these is incomplete
- Emotion and cognition are tightly integrated in humans
- How can we unify cognitive functions with appraisal?
  - Claim: Both are concerned with *event processing*

2



Appraisal Theory of Emotion

- Suppose a person has some goals, beliefs, etc. (knowledge)
- An event occurs (internal or external)
- The person *appraises* the *relationship* between his goals and the event along a number of dimensions (e.g. unexpectedness, conduciveness, agency, etc).
- The appraisal automatically leads to *emotion* (e.g. physiological/cognitive changes, thought-action urges, etc)
- The person perceives emotion as *feelings* (internal event)
- The person *copes* with feelings by taking internal or external actions to improve/maintain the relationship between his goals and the environment

4

Proposed Appraisals Dimensions

Scherer 2001	Roseman 2001	Smith & Lazarus 1990; Smith & Kirby 2001	Lazarus 1991/2001	Gratch & Marsella (2004)
Novelty: Suddenness				
Novelty: Familiarity				
Novelty: Predictability				
Intrinsic pleasantness				
Goal/need relevance		Motivational relevance	Goal relevance	Relevance
Cause: agent	Agency	Self/Other accountability	Blame and credit	Causal attribution
Cause: motive				
Outcome probability	Probability	Future expectancy	Future expectations	Likelihood
Urgency				
Discrepancy from expectation	Unexpectedness			
Conduciveness	Situational state	Motivational congruence	Goal congruence	Desirability
Control	Control potential	Problem-focused coping potential	Coping potential	Changeability
Power				Controllability
Adjustment		Emotion-focused coping potential		
Internal standards compatibility			Type of ego involvement	Perspective
External standards compatibility				
	Motivational state			
	Problem type			

5

Appraisals to Emotions

	Scherer 2001	Elation/Joy	Fear	Rage/Hot Anger
Relevance	Suddenness	High/medium	High	High
	Familiarity		Low	Low
	Predictability	Low	Low	Low
	Intrinsic pleasantness		Low	
	Goal/need Relevance	High	High	High
Implication	Cause: agent		Other/nature	Other
	Cause: motive	Chance/intentional		Intentional
	Outcome probability	Very high	High	Very high
	Discrepancy from Expectation		Dissonant	Dissonant
	Conduciveness	Very high	Obstruct	Obstruct
	Urgency	Low	Very high	High
Coping potential	Control			High
	Power		Very low	High
	Adjustment	Medium	Low	High
Normative Significance	Internal standards compatibility			
	External standards compatibility			Low

6

What's Missing?

- When are appraisals generated?
- Why are the appraisals generated then?
- How are appraisals generated?
- How do appraisal and emotion impact behavior?

7

Cognitive Functions: Allen Newell's PEACTIDM

An agent must be able to perform the following functions

Event Processing	Perceive	Raw perception
	Encode	Create domain-independent representation
	Attend	Chose stimulus to process
	Comprehend	Generate structures that relate stimulus to goals and can be used to inform behavior
Response Processing	Tasking	Perform goal maintenance
	Intend	Chose an action
	Decode	Decompose action into motor commands
	Motor	Execute motor commands

What's Missing?

Example: Bob steps down from the curb.

Perceive	What information is generated?
Encode	What information is generated?
Attend	What information is required?
Comprehend	What information is generated?
Tasking	What information is required?
Intend	What information is required?

9

Unifying Cognitive Functions and Appraisal

Appraisal Generators

Appraisal Consumers

Event Processing	Perceive	Raw perception
	Encode	Domain-independent representation
	Attend	Chose stimulus to process
	Comprehend	Generate structures that relate stimulus to goals and can be used to inform behavior
	Tasking	Perform goal maintenance
	Intend	Chose an action

10

Encode and Event Structure

- Encode generates domain-independent *event* structures from the raw Perceptual information
  - *Events are the foundational data structure that unify appraisal and PEACTIDM*
- Simplification of Talmy (1975)
  - Actor Bob
  - Action Walking across street
- Also includes metadata about the event

11

Attend

- Most events are probably not worth paying attention to
- Attend uses metadata from Encoded structure determine if an event should be processed further
- What metadata?
  - Suddenness
  - Familiarity
  - Predictability

12

Comprehension Process

- Goal: To create data structures that inform behavior
- Key: Process *sequences* of events
- Process
  - Observe partial sequence of events
  - Match partial sequence to known complete sequence
  - Use complete sequence to predict next event
- Only work on one event or sequence at a time (i.e. processing is local)
- Since the event structures are domain independent, this process is also domain independent

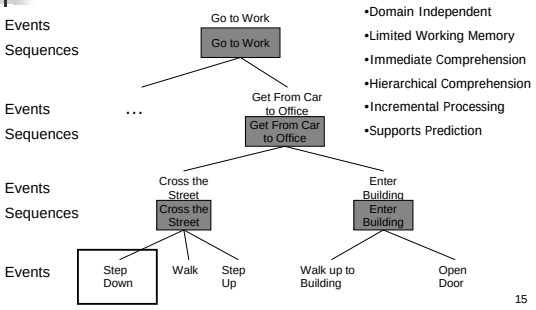
13

Abstract Events, Sequences and Subgoals

- An event sequence can be abstracted to represent a single event in a more abstract sequence
- Example:
  - Step down from curb
  - Take a few steps
  - Step up onto curb
  - ...this is just the "Cross the Street" event, which may be just one event in the "Get from Car to Office" sequence, which may be one event in the "Go to Work" sequence...which may be just one event in the "Living My Life" sequence.
- Abstract events can be thought of as subgoals

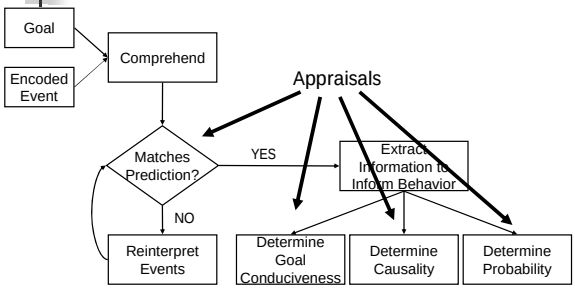
14

Event Knowledge Hierarchy



15

Comprehension Process Details



16

Unifying Cognitive Functions and Appraisal Revisited

Response Processing	Perceive	Raw perception
	Encode	Domain-independent representation
	Attend	Chose stimulus to process
	Comprehend	Generate structures that relate stimulus to goals and can be used to inform behavior
	Tasking	Perform goal maintenance
	Intend	Chose an action

Appraisal Generators

Appraisal Consumers

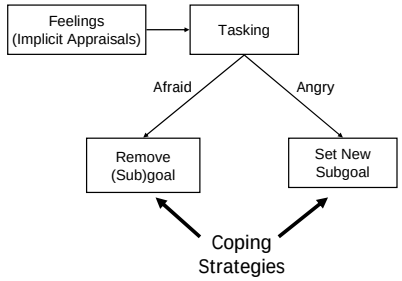
17

Tasking Process

- Goal: Update current (sub)goals as necessary
- Key: Emotion automatically signals with status (goal threatened, situation alterable) and how to fix it (e.g. whose fault is it, etc)
- Process:
  - Determine how to proceed based on implications of emotion

18

Tasking Process Details



19

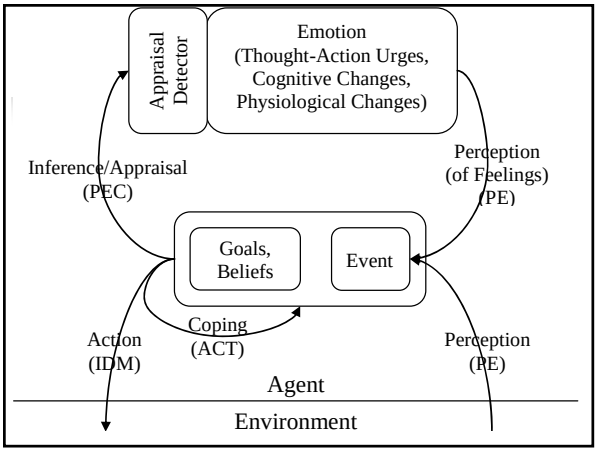
Intend Process

- Goal: Determine next action to execute
- Key: In general, there may be many paths from the current situation to the goal, so Intend must pick one
  - Also has to compete with action tendencies (e.g. automatic responses)
- Process: Appraisal
  - If urgency is high, "automatic" responses win
  - Otherwise, walk event hierarchy to find path to goal

20

Unification

Scherer 2001	Generated By	Required By
Novelty: Suddenness	Perception	Attend
Novelty: Familiarity	Encoding	
Novelty: Predictability		
Intrinsic pleasantness		Tasking (via Feelings)
Goal/need relevance	Comprehension	
Cause: agent		
Cause: motive		
Outcome probability		
Urgency		Intend (via Feelings)
Discrepancy from expectation		Comprehension
Conduciveness		Tasking (via Feelings)
Control		
Power		
Adjustment		
Internal standards compatibility		
External standards compatibility		

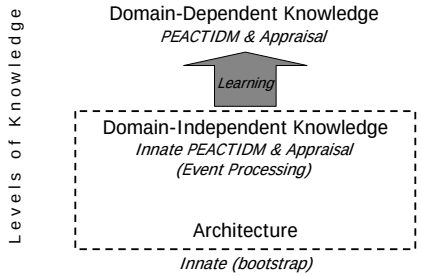


Predictions

- Agent will be interruptible
- Partial ordering constraint on appraisal generation
- Different emotions may require different amounts of processing
- Time constraints may lead to errors in Comprehension (and thus emotion)

23

Impact on Soar: Innate Knowledge



24

Summary

- Nuggets
  - Appraisal processing and PEACTIDM both fill in missing pieces of each other
  - The story satisfies multiple psychological constraints
  - May give some insight into innate knowledge
  - **Appraisal generation isn't special – it results from normal processing**
- Coal
  - Unifying these does not solve everything: theoretically and implementationally, there are still a lot of hard, unanswered questions

25

Update on Soar-based language processing

Deryle Lonsdale
(and the rest of the BYU NL-Soar Research Group)
BYU Linguistics
lonz@byu.edu

NL-Soar

NL-Soar developments

- Discourse/robotic dialogue
  - ICSLP DoD
  - BRIMS poster
- Running on Soar 8.5.2
  - Some NLG chunking issues remain
- Having trouble getting to 8.6.1
  - Fresh start with 8.6.2...

LG-Soar

Overview

- Link-Grammar Soar
  - Implements syntactic, shallow semantic processing
  - Used for information extraction
  - Components
    - Soar architecture
    - Link Grammar parser
    - Discourse Representation Theory for discourse modeling
  - Discussed in Soar 21, Soar 22

LG-Soar developments

- Predicate extraction in biomedical texts domain (www.clinicaltrials.gov)
- NLDB 2006
- Two stages:
  - Identify and extract predicate logic forms from medical clinical trial (in)eligibility criteria
  - Match up the information with other data, e.g. patients' medical records
- Result: tool for helping match patients with clinical trials

XNL-Soar

Our goals

- Integrate the MP into a cognitive modeling engine
- Explore language/task-integrations using the MP
- Test cross-linguistic implementation possibilities with the MP
- Ultimately, determine whether the MP supports incremental, operator-based processing

Our approach

- Map the syntactic parsing onto operators
- Integrate external knowledge sources
- Strengths:
  - We have already done this for NL-Soar
  - The MP has an operator-like feel to it
- Weaknesses:
  - MP lit sketchy on incremental parsing
  - External knowledge sources incommensurable
  - Our scant knowledge of human performance data

Derivational principles

- Minimalist Principles (Chomsky 1995)
  - Merge
  - Move
- Hierarchy of Projections (Adger 2003)
  - Nominal: D > (Poss) > n > N
  - Clausal: C > T > (Neg) > (Perf) > (Prog) > (Pass) > v > V
- Governed by features
  - Strong and weak features
- NP, VP symmetry including shells

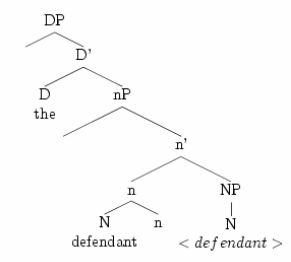
Operators and operator types

- XNL-Soar op types and functions:
  - Lexical access: retrieve & store lexically-related information
  - Merge: construct syntax via MP-specified merge operations
  - Movehead: perform head-to-head movement (via adjunction)
  - HoP: consult hierarchy of projections, return next possible target level
  - Project: create bare-structure maximal projection from lexical item

Nominal projections

- DP's
- NP Shells
- Feature checking
  - HoP for nominals
  - Projection of N to NP
    - Bare phrase structure for lexical heads
- Operators: Project, Merge1, Merge2, Check-Root, and HoP

Projection of a DP



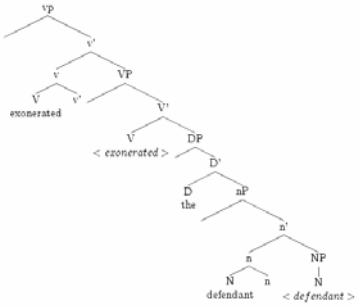
Verbal projections

- VP Shells
- Theta roles & the LCS lexicon
- ucat grids
- The HoP for the clause
- Operators: Merge1, Merge2, Check-root, HoP.

Merge

- 1st Merge
  - Complement merge based on ucat features
- 2nd Merge
  - Specifier merge based on a second ucat feature

Projection of a VP: HoP & Merge



Move

- Governed by strong ucat features
- Two types of movement
  - Head-head movement
    - Creates new structure at the head level
  - Phrasal movement
- Operators: Copy, Hadjunction

External knowledge sources

- WordNet 2.0 (wordnet.princeton.edu)
  - Lexical semantics: part-of-speech, word senses, subcategorization
  - Inflectional and derivational morphology
- English LCS lexicon (www.umiacs.umd.edu/~bonnie/verbs-English.lcs)
  - Thematic information: θ-grids, θ-roles
  - Used to derive uninterpretable features
  - Triggers syntactic construction
  - Aligned with WordNet information

English LCS lexicon data

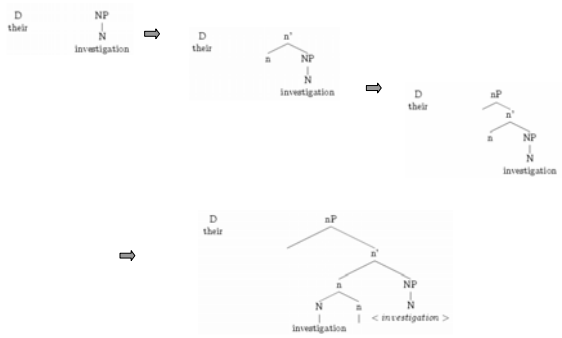
```
10.6.a#1#_ag_th,mod-
poss(of)#exonerate#exonerate#exonerate#exonerate+ed#
(2.0,00874318_exonerate%2:32:00::)

10.6.a Verbs of Possessional Deprivation: Cheat Verbs/-of
WORDS (absolve acquit balk bereave bilk bleed burgle cheat
cleanse con cull cure defraud denude deplete depopulate
deprive despoil disabuse disarm disencumber dispossess divest
drain ease exonerate fleece free gull milk mulct pardon
plunder purge purify ransack relieve render rid rifle rob sap
strip swindle unburden void wean)

((1 "_ag_th,mod-poss()")
 (1 "_ag_th,mod-poss(from)")
 (1 "_ag_th,mod-poss(of)"))

"He !!+ed the people (of their rights); He !!+ed him of his
sins"
```

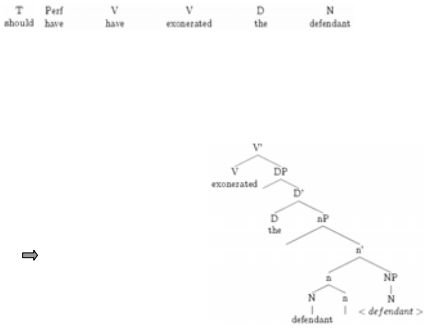

Similar Work

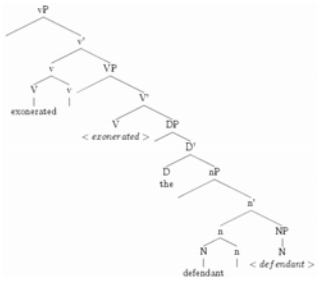
- Incremental parsing in general (Phillips '03)
- Other linguistic theories for incremental parsing
  - GB (Kolb 1991)
  - Dependency grammar (Milward 1994, Ait-Mokhtar et al. 2002)
  - Categorial Grammar (Izuo 2004)
- Finite-state methods (Ait-Mokhtar & Chanod 1997)

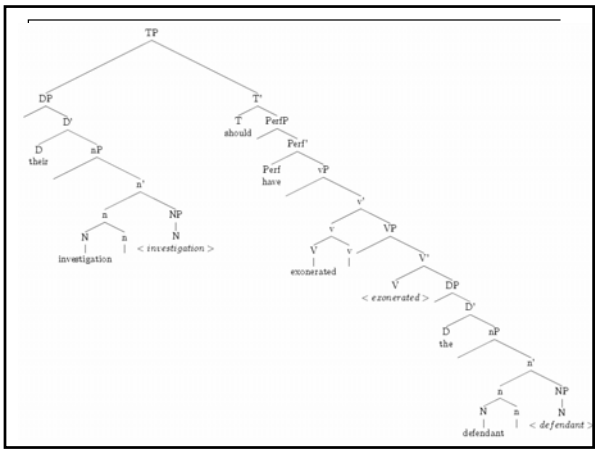
Similar Work

- Minimalist parsing in other frameworks (Stabler 1997, Harkema 2001)
- Thematic information and parsing (Schlesewsky & Bornkessel 2004)
- Crosslinguistic considerations in incremental parsing (Schneider 2000)
- Human studies on ambiguity, reanalysis
  - Eye tracking (Kamide, Altmann, & Haywood '03)
  - ERP (Bornkessel, Schlesewsky, & Friederici '03)









Building a full sentence

```
agent> init-soar
agent> r
     0: ==>S: S1

     2:    O: O1 (getword)
Input a word: exonerated

     4:    O: O2 (getword)
Input a word: defendants
     5:    O: O3 (project) --> NP

     7:    O: O5 (hop)
     8:    O: O8 (merge1)
    10:    O: O9 (merge2) --> nP

    12:    O: O14 (hadjoin) --> move N

    13:    O: O13 (hop)
    14:    O: O16 (merge1)
    16:    O: O17 (merge2) --> DP

    18:    O: O21 (merge1)
    20:    O: O23 (merge2) --> VP

    22:    O: O27 (hop)
    23:    O: O30 (merge1)
    25:    O: O31 (merge2) --> vP

    27:    O: O36 (hadjoin) --> move V

    28:    O: O35 (hop)
    29:    O: O38 (merge1)
    31:    O: O39 (merge2) --> TP
```


Current status

- POC for fundamental syntactic structures
- Basic sentence types (transitives, unergatives, unaccusatives)
- All functional and lexical projections in syntactic structure
- Most feature percolation, feature checking
- Current system: about 60 productions (cf. 3500 NL-Soar)
- External knowledge sources: interfaced via 1000+ lines of Tcl/Perl

Issue

- Find a balance between generation and parsing
  - Most MP descriptions are generative, not recognitional in focus
  - Is it advisable and well motivated to "undo" or "reverse" movements?
  - If not, is generate-and-test the right mechanism for parsing input?
  - What are the implications for learning and bootstrapping language capabilities (e.g. parsing in the service of generation)?

Future functionality

- XP adjunction
- Assigners/receivers set?
- Wider coverage of complex constructions
  - Ditransitives, resultatives, causatives, etc.
- More semantics/deeper semantics.
  - Quantifier raising
  - Scopal relationships
  - C-command and other interpretive mechanisms
  - More detailed LCS structures
- Web-based Minimalist Parser grapher

Future applications: cf. NL-Soar

- Explore human parser robustness, processing of ambiguity, learning
- Integrate syntax/semantics into discourse/conversation component
- Bootstrapping: parsing and generation
- Develop human-agent & agent-agent comm
- Parameterize XNL-Soar for processing of other languages besides English
- Model cognition in reading
- Model real-time language/task integrations

Conclusion

- Coals
  - Performance?
  - MP not fully explored
  - Reconciling disparate lexical resources is non-trivial (WordNet + LCS)
  - Redoing learning in Soar8
  - Graphing is more complicated
- Nuggets
  - Better coverage (Engl. & crosslinguistically)
  - New start in Soar8
  - State-of-the-art syntax
  - Interest: CUNY Sentence Processing, CogSci

Soar 2006 31

Steps toward a Domain Independent Episodic Memory

Andrew Nuxoll
24 May 2006

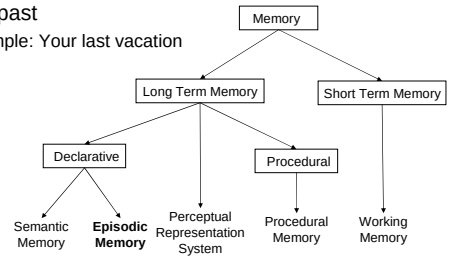
Outline

- Review
  - Definitions
  - Improving agent behavior
- Improving Domain Independence
  - Improving Match
  - Chunking (with confidence)

2

What is Episodic Memory?

- Memories of specific events in our past
  - Example: Your last vacation



3

Research Goals

- Explore the cognitive capabilities granted to an agent with an episodic memory
- Explore what's necessary to build an effective episodic memory for a general cognitive architecture
  - Domain independence
  - Performance
- Take inspiration from cognitive psychology

4

Previous Work

- Psychology
  - Observations of Humans - Endel Tulving
- Cognitive Modeling
  - Soar Model (non-architectural) - Erik Altmann
- Artificial Intelligence
  - Continuous CBR - Ram and Santamaría
  - Comprehensive Agents - Vere and Bickmore

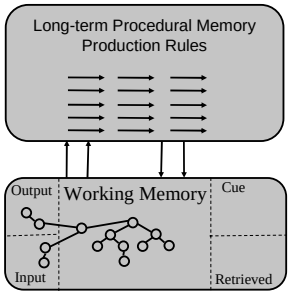
5

Current Implementation

Encoding
Initiation?

Storage

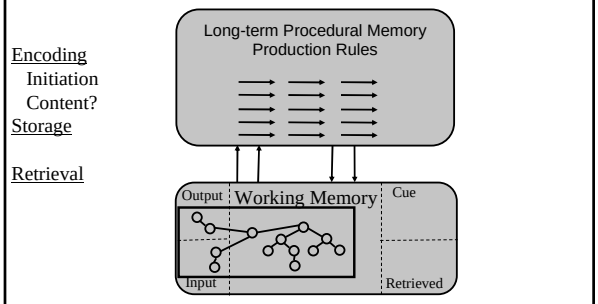
Retrieval



When the agent takes an action.

6

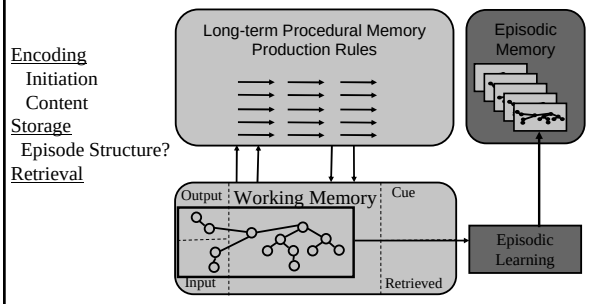
Current Implementation



A portion of working memory is stored in the episode

7

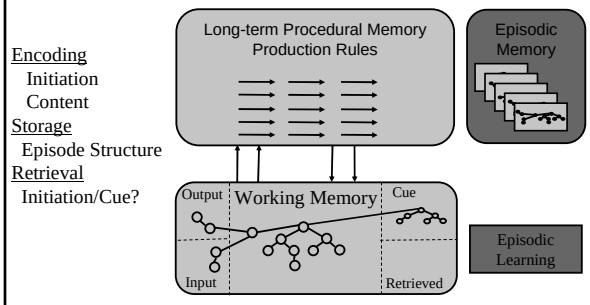
Current Implementation



Episodes are stored in a separate memory

8

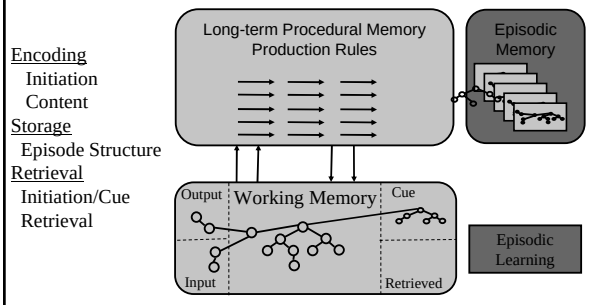
Current Implementation



Cue is placed in an architecture specific buffer.

9

Current Implementation

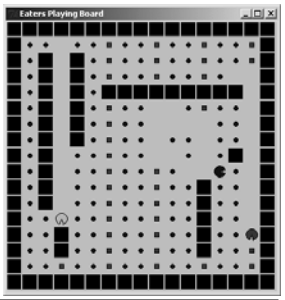


The closest partial match is retrieved.

10

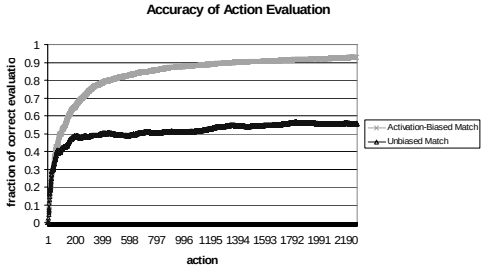
First Domain: Eaters

- Pacman-like environment
- Decisions based upon results of actions taken in previous, similar situations



11

Effects of Memory Activation Bias



12

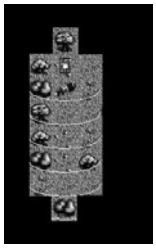
Eaters Results

- Episodic memory improves agent behavior
  - Cognitive Capability: Using past experiences to improve future decisions
- Working memory activation is an effective bias for partial match

13

Improving Domain Independence

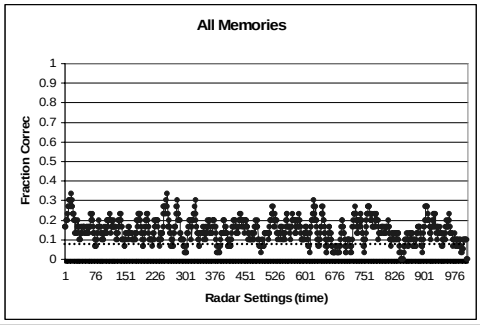
Second Domain: Tank Soar



- Environment: TankSoar
  - "Two-dimensional Quake"
- Task: conserve energy
  - Selecting proper radar setting to minimize energy consumption
- Key Differences (vs. Eaters)
  - Selective Sensing
  - Small cue
  - Limited feedback

15

Initial Performance



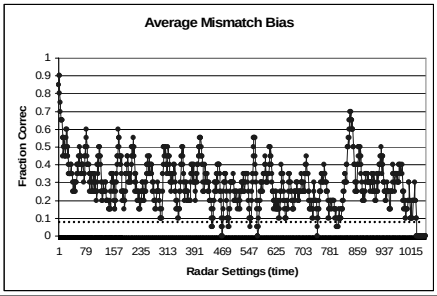
16

Analysis of Initial Performance

- Incorrect Retrieval
  - Small cue means memory activation bias overrides exact match
- Poor decisions beget poor memories
  - Without feedback, agent uses memories of poor decisions to make future decisions

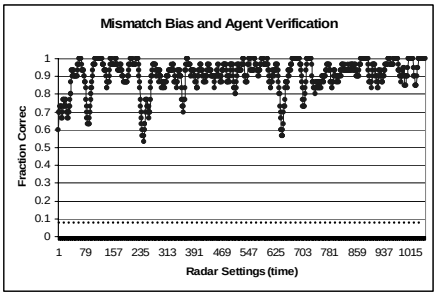
17

Bias Against Mismatch



18

With Agent Verification



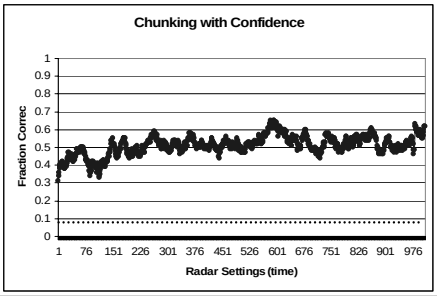
19

Adding Chunking (with confidence)

- Chunking allows the agent to "save" behavior resulting from a good retrieval
  - Allow the chunker to backtrace through the retrieval
- Initial data shows match score is a reliable predictor of episode "correctness" in the radar tank domain
  - Therefore, we can use a match score as a measure of agent confidence
  - First experiment with a domain specific (hard-coded) confidence threshold

20

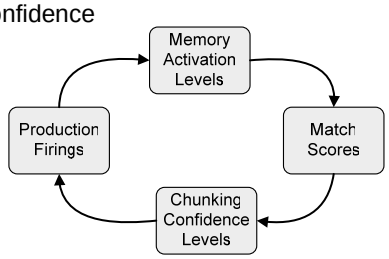
Chunking Results



21

Analysis of Chunking Results

- Interdependency between Activation and Confidence



22

Lessons

- Activation level is a helpful but not reliable predictor of "correctness"
  - Poor memories beget more poor memories
  - Forgetting mechanism?
- Agent ↔ Episodic Memory System communication is essential
  - Agent cue selection
  - Episode includes meta data
  - Agent episode evaluation

23

Nuggets

- Demonstrated effectiveness in two domains
- Improved match
- Activation bias is not enough

Coal

- Activation bias is not enough
- Episodic memory meta-data is needed to improve agent behavior

24

Integrating Semantic Memory in Soar

Yongjia Wang
John E. Laird

1

Outline

- Background & Motivation
- Implementations and Experiments

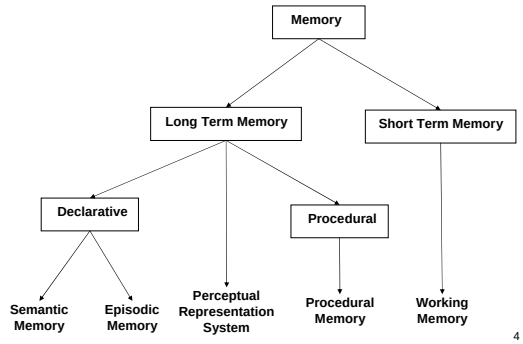
2

What is Semantic Memory

- Definition
  - 'Your memory for meanings and general (impersonal) facts.' [from WordNet]
- Episodic memory and Semantic memory distinction
  - Episodic memory
    - Tied to a specific learning episode or experience
    - What you remember
  - Semantic memory
    - General knowledge not tied to a learning experience
    - What you know

3

Memory Systems



4

Related Fields and Motivation

Architectures	Focus	Feature	Limitations
Cognitive Psychology (ACT-R)	To model human behavior	Long-term declarative memory and learning	Haven't been used to build functional agents
AI Agent Architectures (Soar)	To build intelligent agents	Efficient domain knowledge engineering	No long-term semantic memory, limited learning
Knowledge Representation Systems (Cyc)	To represent common sense semantic knowledge	Declarative knowledge representation	Representational model, not learning model
Our Approach (Soar + semantic memory)	To build intelligent agents	Efficient domain knowledge engineering and more learning capabilities	Constrained by Soar

5

Research Goals

- To improve general functionality of Soar by semantic memory
  - Explore new cognitive capabilities
  - Characterize computational functionalities
- To understand semantic memory in the context of a general cognitive architecture
  - How to use semantic memory in specific tasks?
  - How semantic memory interacts with other mechanisms in Soar?
  - What are the computational implications of semantic memory and episodic memory distinction?

6

Distinction Between Semantic Memory and Episodic Memory in Soar

	Semantic Memory	Episodic Memory
Storage & retrieval unit	Single level objects in working memory (declarative chunks)	Entire working memory snapshot (episode)
Temporal information	No architectural temporal information	Architectural temporal information (ex: next episode)
Main purpose	Store general knowledge Category learning	Store specific events Case-based reasoning

7

Outline

- Background & Motivation
- Implementations and Experiments
  Task: Cognitive Arithmetic

8

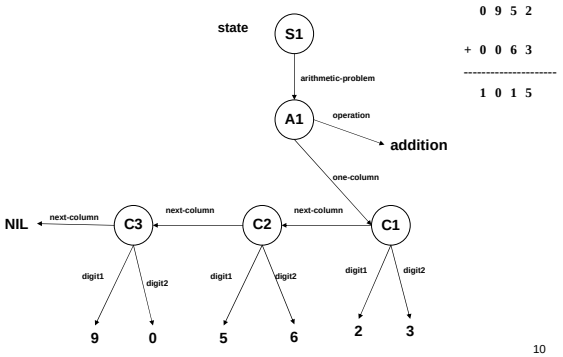
Overview of Experiment

- Purpose:
  - Integrate a declarative semantic memory component
  - Demonstrate related functional advantage of declarative representation
- Implementation:
  - Semantic memory with declarative representation
  - Deliberate and automatic semantic learning
- Task: Cognitive arithmetic
  - Easy to understand
  - Universally performed
  - Multiple types of learning

```
  0 9 5 2
+ 0 0 6 3
---------
  1 0 1 5
```

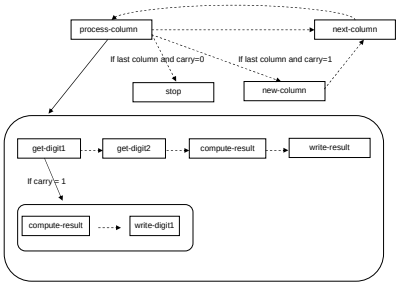
9

Working Memory Representation of an Arithmetic Problem



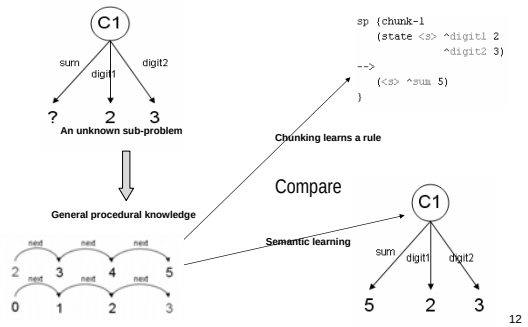
10

Problem Space

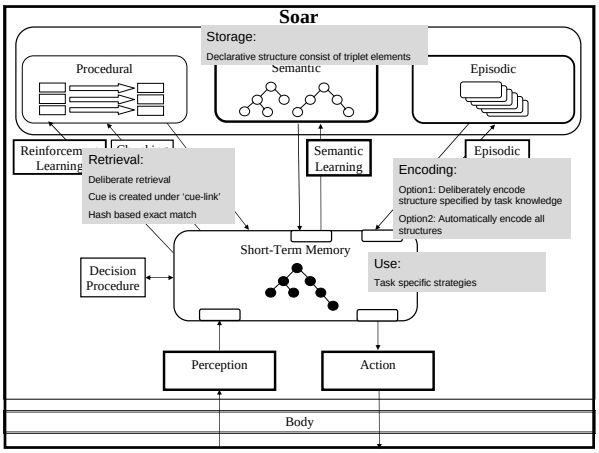


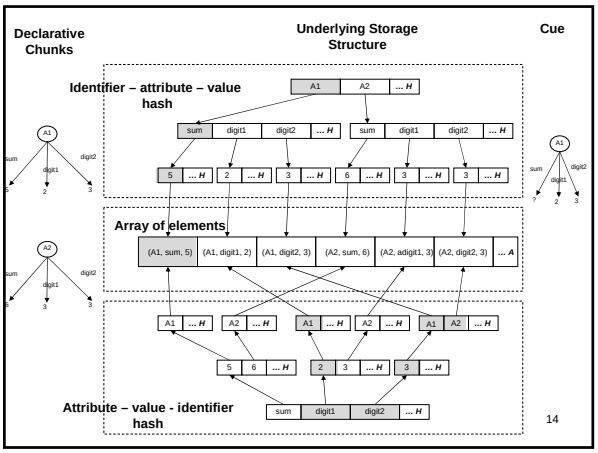
11

Solving One Column

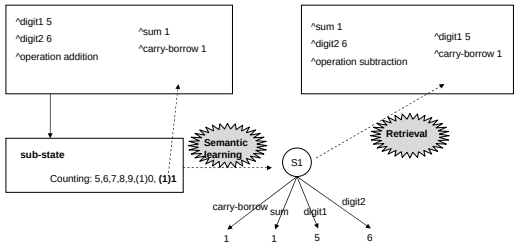


12





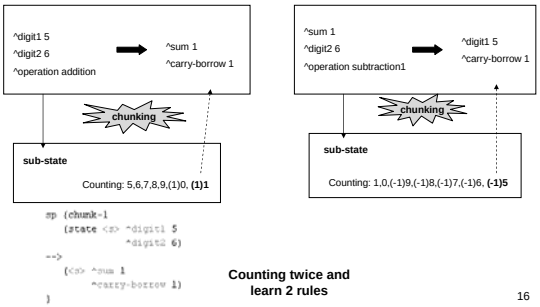
Transfer Learning Effect from Semantic Learning



Counting once and learn 1 declarative chunk

15

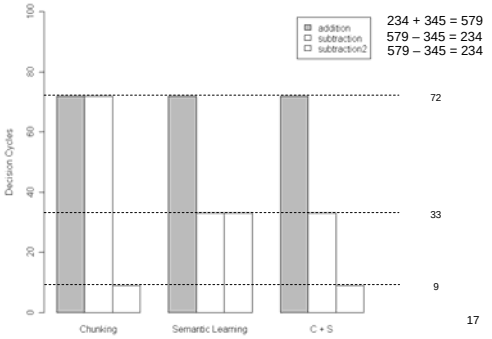
Procedural Representation Cannot be Transferred



Counting twice and learn 2 rules

16

Comparison of Different learning Configurations



17

Decision Cycles Breakdown

Situations / Decision Cycles / Operators	All computations	With arithmetic facts	After chunking
operators in top-state	9	9	9
get-digits (from top-state)	3*3=9	9	0
write-result (to top-state)	3*1=3	3	0
retrieve	3*4=12	12	0
counting	39	0	0
Total	72	33	9

18

Summary

- Nuggets
  - Implemented a semantic memory with declarative representation
  - Demonstrated the functional advantage of declarative representation over procedural representation
  - Demonstrated transfer learning effect by semantic learning
  - Demonstrated the functional Interaction between semantic learning and chunking
- Coals
  - Cognitive arithmetic is an internal mental task
  - The task is completely deterministic

19

Thank You

20

Hierarchical Reinforcement Learning in Soar

26th Soar Workshop
May 24, 2006

Shelley Nason
University of Michigan

Methods for Transfer Learning Using Soar

26th Soar Workshop
May 24, 2006

Nicholas Gorski, John Laird, Taylor Lafrinere

What is Transfer Learning?

- Transfer Learning: using previously learned knowledge to improve performance on later, related tasks
- Important because traditional machine learning techniques learn over narrow problem-spaces
- TL approaches should be general enough to apply to a wide variety of domains

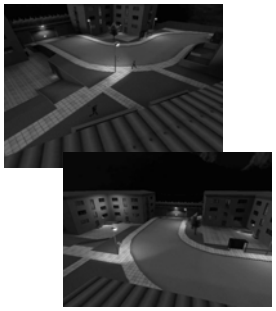
2

Developing TL Approaches

- Our strategy is to develop multiple approaches and compare their performance on various tasks
- Combining approaches to leverage strengths of each will achieve good transfer
- Soar is a good platform for this comparison study: it provides multiple learning mechanisms in a single architecture

3

Urban Combat Testbed



- Multi-agent FPS real-time video game
- Built on Quake 3 engine
- UCT exposes shared memory interface, interfaced to Soar kernel via SML

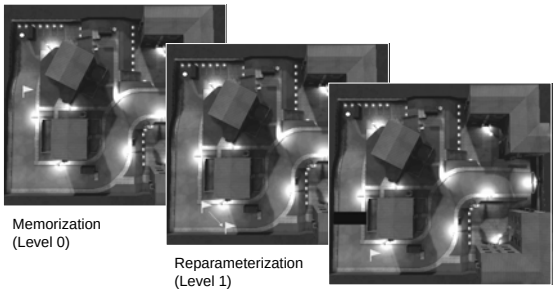
4

Transfer Learning in UCT

- Many different types of transfer, divided into 10+1 levels
- First-to-flag scenarios created for UCT to test each level of transfer
  - Each scenario consists of a source and target problem
  - Knowledge transferred from the source improves performance on the target
- Declarative knowledge that can be transferred includes the location of a flag, a map, and routes

5

Transfer Learning in UCT



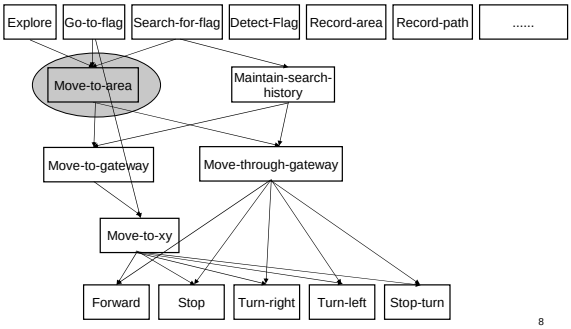
6

Approaches

- Multiple approaches developed that take advantage of multiple learning mechanisms in Soar
  - Memory-based
  - Search-based
  - Reinforcement Learning

7

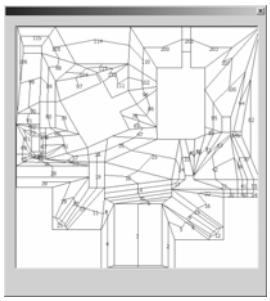
Urban Combat Agent - UCBot



8

Memory-based

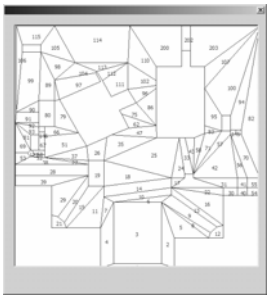
- Trades memory for fast execution
- Stores *O(N)* areas & *O(N²)* paths in working memory
- When the agent needs a path, no computation is required
- Abuses working memory and affects the Rete matcher



9

Search-based

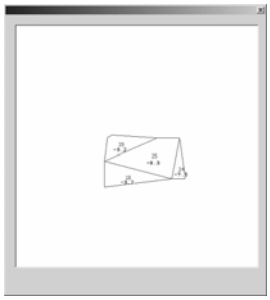
- Still maintains N areas, but no path information in working memory
- When the agent needs a path, it performs a model-based search using a nearest-gateway heuristic
- Chunks results of search: agent stores path information as procedural knowledge



10

Reinforcement Learning

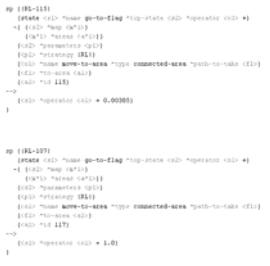
- Stores no spatial knowledge in working memory
- Learns value of moving to an area, stored as numeric preferences on operator proposals
- Significantly longer training time required
- However, route finding UCT is essentially deterministic



11

Reinforcement Learning

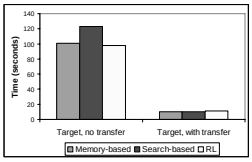
- Location of flag and paths are stored implicitly as expected values
- Captured as RL rules, where numeric preferences specify the expected value of moving to an area



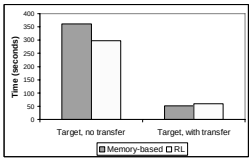
12

Evaluation

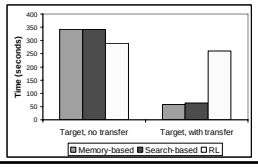
Memorization (Level 0)



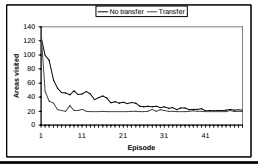
Reparameterization (Level 1)



Extrapolation (Level 2)



RL agent's level 2 learning curve



Generalizing the Approaches

- How do these approaches generalize across domains?
  - Memory-based: pre-compute all solutions and store in working memory
  - Search-based: store domain knowledge in working memory, but search for solutions rather than pre-computing all possible solutions
  - Reinforcement Learning: transfer of statistical knowledge
- Strongest transfer will be achieved when all approaches are combined, leveraging the strengths of each

14

Nuggets & Coal

- Existing UCBot should easily extend to higher levels of transfer
- Comparison of memory/search based methods to RL highlight strengths and weaknesses of each
- Results for only one domain, and UCT is not an ideal TL testbed (yet)
- Haven't investigated transferring more complicated semantic knowledge

15

Soar Technology
Thinking inside the box.

Soar Technology Community Liaison

Dave Ray
Soar Workshop
May 24, 2006

Role

- Primary developer/maintainer of Soar products at Soar Technology
  - Kernel
  - gSKI
  - etc
- Keep SoarTech informed of developments in the community
- Share SoarTech developments, bug fixes, etc with community
- Provide feedback to community on SoarTech usage of community tools

Near-term Tasks

- Integrate 8.6.2 kernel and gSKI into SoarTech tool-chain
- Integrate rewritten gSKI working memory (input/output) into 8.6.2 code base for 8.6.3 release
  - Significantly reduced code duplication
  - Easier to debug
  - Consistent reference counting
  - Eliminated several memory leaks

Contact Information

- Email: ray@soartech.com
- Phone: 734 327 8000 x223
- SML mailing list

Soar Technology
Thinking inside the box.

Self Introduction

Alan J. Vayda, Ph.D.
Senior Scientist

alan.vayda@soartech.com
734-327-8000 x355

Soar Workshop
May 24, 2006

Education

- B.S. Engineering Science - Penn State
  - Thesis: An Analysis of Some Factors Affecting the Computation of Far Field Acoustic Response from Near Field Data
- M.S. Electrical Engineering - Penn State
  - Thesis: Development of an Interactive Computer Assisted Instruction System
- Ph.D. Electrical Engineering - Purdue
  - Thesis: Reasoning with Geometric Constraints for Generic 3-D Object Recognition in Occluded Environments

Employment

- ERIM (now Altarum)
  - Research institute
  - 7 years
- Nonlinear Dynamics/NovoDynamics
  - Start-up
  - 8 years
- Soar Technology
  - 7 months

Pre-SoarTech Experience

- Pattern Recognition (Recursive Partitioning)
- Data Analysis and Visualization
- Information Retrieval
- Optical Character Recognition (OCR)
- Video Tracking
- Robot Vision
- Contextual Reasoning
- Geometric Reasoning
- Large Scale Heterogeneous Data Management
- Web-Based Systems
- Materials Discovery (Combinatorial Chemistry)

Soar Tech Projects

- Real-time Adversarial Intelligence and Decision-making (RAID)
  - Predicting opponent course of action in an urban environment
- Intelligence Analysis Support (Tangram)
  - Soar agent composes sequences of graph algorithms to achieve desired outcome based on graph and algorithm features specified in an Ontology
- Biologically-Inspired Cognitive Architectures (BICA)
  - Integration of biologically-based algorithms into Soar and considering changes to Soar architecture

Interests

- Predictive Modeling
- Sensing and Perception
- Robotics and Unmanned Vehicles
- Uncertainty and Evidential Reasoning
- Contextual Reasoning
- Multi-Agent Teams
- Soar (in concert with all of the above)

Soar FAQ

JongKim@psu.edu
Frank.Ritter@psu.edu
The Pennsylvania State University

- It provides answers to frequent Soar questions
- It is also a guide for learning more about Soar
  - Psychological theories underlying Soar
  - Programming tips for building AI agents
- It is intended to be used by all levels of users from novices to experts
- The initial versions were supported by:
  - DERA (Defense Evaluation and Research Agency)
  - ESRC Centre for Research in Development, Instruction and Training, and now
  - ONR (Office of Naval Research)
- http://acs.ist.psu.edu/projects/soar-faq/soar-faq.html
- http://sitemaker.umich.edu/soar/documentation_and_links

Maintenance of Soar FAQ

- The FAQ is maintained by ACS Lab at Penn State University under the supervision of Dr. Frank Ritter
- We scan Soar Workshop Proceedings annually and keep watching Soar-group emails with the FAQ in mind
- We post answers where we can see common and important questions
- The Soar FAQ is not just our work but includes numerous answers from members of the Soar community

Future Directions

- The FAQ accumulates valuable knowledge of Soar
- The role of Soar FAQ is to provide a rapid access to Soar knowledge resources for developers
- May wish to consider a Wiki-based system to promote easier, broader support
- Suggestions or Comments:
  - Frank E. Ritter: frank.ritter@psu.edu
  - Jong W. Kim: jongkim@psu.edu

Soar Technology
Thinking *inside* the box.

Participatory scenario design and simulation on the ICF testbed

Elyon DeKoven, Ph.D.

dekoven@soartech.com
Soar Technology, Inc.

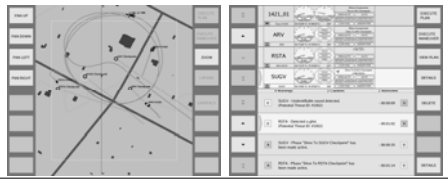
Future Battlefield Teamwork

- U.S. Army is undecided on future UV capabilities and UV-human team structure
  - C2 of UVs requires new tactics, techniques and procedures
- There is little known about how to support effective C2 of multiple autonomous systems
  - Current robot control primarily teleoperation
  - Autonomous is not the same as effective

Intelligent Control Framework (ICF)

- Research operator-oriented issues in C2 of mixed human-robot teams
- Develop multi-agent control framework and components for operator C2 of robot teams
- Build a test bed, method and techniques for scenario-based simulation and evaluation

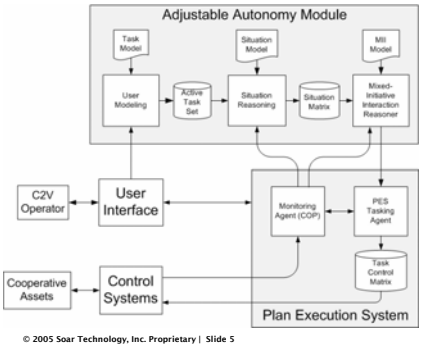
Issues in human-UV teamwork



- Collaborative replanning
  - Option generation and selection
  - Autonomy allocation and real-time adaptations
- Maintaining team SA
  - Fog of war, imperfect sensors and soda straws
  - Multiple simultaneous tasks
- Operator attitude
  - Confidence, over-reliance, etc.

There is no perfect solution for every situation, user or robotic team

ICF Architecture (Year 1)

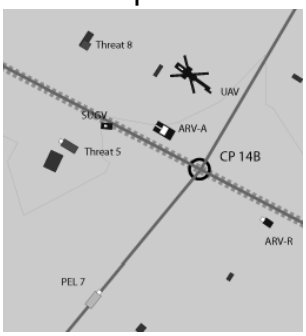


System knowledge and behaviors are driven by data files and heuristics

ICF testbed development efforts

- *Tools* for simulation of C2 of multi-UV teams and mixed human-robot teams
- *Techniques* for SME knowledge elicitation based on wargaming and participatory design
- *Methods* for scenario based design and assessment of TTPs, policies, and UVs

Example Scenario: IED Ambush

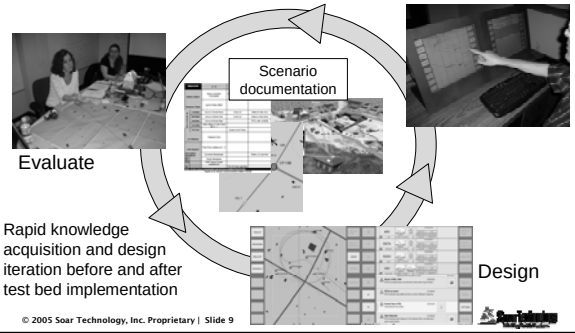


- IED discovered, small arms fire, multiple moving contacts (hostile & unknown)
- What happens in the first few seconds will likely determine the survivability of the assets
- RNCO has 4 UVs to think of and 3 possible threats.
- Automation has possible advantages here, but automation not always appropriate or effective

Next steps

- Explore possible variations and endings
  - Work with SMEs to identify tactics, useful automations, and policies (ROEs)
- Translate findings into ICF prototype
  - Encode SME knowledge into domain ontologies and behavior models
- Evaluate and redesign the ICF UI
  - Identify key operator issues in management of robot teams and develop solutions

Approach: Iterative scenario development and simulation



Rapid knowledge acquisition and design iteration before and after test bed implementation

What's inside this box?

Nuggets	Coals
• Military compatibility – U.S. military already uses similar wargaming and simulation techniques	• Good men = hard to find – There are few soldiers with significant amount of robot experience
• Reduce engineering time – Iterative wargaming can reduce development time in the face of unknowns	• Good men = expensive – SMEs are expensive and it can be difficult to get them to think outside their box
• Iterative refinement – This process will help refine the ICF test bed, and the ICF test bed will be a valuable piece in this process	• Need more tools – There is currently little tool support to ease translation from envisioned capabilities to encoded behaviors

Discussion

Elyon DeKoven, Ph.D.
dekoven@soartech.com

ORTS: A Case Study of Multi-Tasking in Soar

Joseph Xu
University of Michigan
Soar Workshop 2006

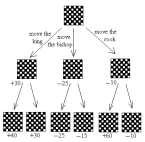
1

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

2

Traditional Games in AI



- Discrete time/actions
- Perfect information
  - Enumerable states
- Low perceptual/motor load
- Examples:
  - Chess
  - Towers of Hanoi
  - Water Jug

3

Real-Time Strategy Games

- Played in real time
- Maintain an economy
- Develop production capabilities via cities/bases
- Defend against enemy attacks
- Launch attacks against enemies
- Examples:
  - Starcraft
  - Command & Conquer

4

Real-Time Strategy Games

- Continuous time/space/actions
- Imperfect information
  - State space not practically enumerable
- High perceptual/motor load

5

Challenges

	Which game stresses this more?	What's better at this (presently)?
Look-ahead	Chess	Computer
Opponent modeling	Chess	Human
State abstraction	RTS	Human
Spatial/Temporal reasoning	RTS	Human
Manage perceptual overloading	RTS	Human
Multi-faceted gameplay	RTS	Computer
Divided Attention	RTS	Computer

6

Challenges

	Which game stresses this more?	What's better at this (presently)?
Look-ahead	Chess	Computer
Opponent modeling	Chess	Human
State abstraction	RTS	Human
Spatial/Temporal reasoning	RTS	Human
Manage perceptual overloading	RTS	Human
Multi-faceted gameplay	RTS	Computer
Divided Attention	RTS	Computer

7

Challenges

	Which game stresses this more?	What's better at this (presently)?
Look-ahead	Chess	Computer
Opponent modeling	Chess	Human
State abstraction	RTS	Human
Spatial/Temporal reasoning	RTS	Human
Manage perceptual overloading	RTS	Human
Multi-faceted gameplay	RTS	Computer
Divided Attention	RTS	Computer

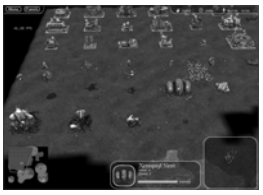
8

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

9

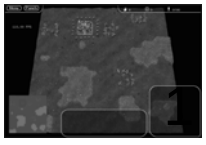
Open Real Time Strategy

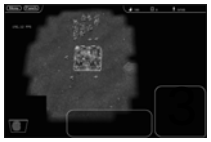


- Open source RTS implementation
- Designed specifically for AI research
- Completely customizable via scripts
- C++ API
  - receive information about state of the world from server
  - Send commands to server
- Under active development at University of Alberta

10

AI Competition at AIIDE 06





- Game 1 – Resource gathering
- Game 2 – Offense and Defense
- Game 3 – Full RTS game

11

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

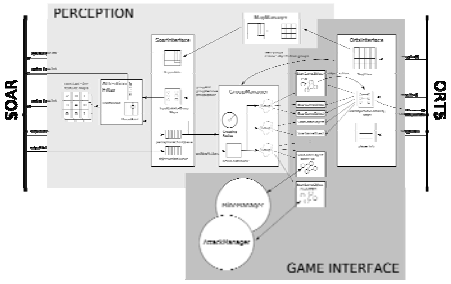
12

Our Approach: SORTS

- Create a Soar agent to play ORTS
- Middleware serves as both Soar's perceptual system as well as a gaming interface
  - Like a real game interface, the middleware handles micromanagement such as pathfinding and default unit behaviors

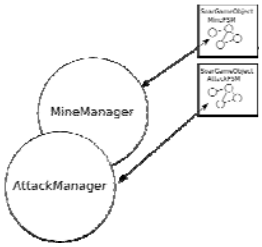
13

SORTS Architecture



14

Low Level Control

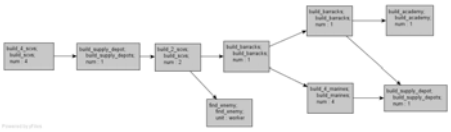


- Each unit controlled by finite state machines
- Soar agent doesn't have to micromanage
- MineManager and AttackManager necessary for finer control in competition

15

Soar Agent Preliminary Design

- Three ways of acting
  - Static plans
    - Defined ahead of time, like the opening book in chess
    - Partial Order Plans



16

Soar Agent Preliminary Design

- Three ways of acting
  - Opportunistic plans
    - Plans that the agent comes up with while playing the game
    - Backward chaining
  - Example: I need to build anti-air defenses to counter enemy fighters, but to do that I need to build a factory first

17

Soar Agent Preliminary Design

- Three ways of acting
  - Reactions
    - Reactions to the current state that can occur at any time
  - Example: I can't win this battle, retreat with remaining forces

18

Soar Agent Preliminary Design

- Situation Awareness
  - Soar agent can only "see" a limited area of game field at any time
  - Agent must make decisions that account for unseen parts of the game field too
  - Must maintain situation awareness by memorizing important things going on at different parts of the map

19

Soar Agent Preliminary Design

20

Division of Responsibilities

- Soar handles
  - State abstraction
  - Planning
  - High level commands
  - Multi-tasking
- MW handles
  - Visual abstraction
  - Command implementation
  - Default unit behaviors
  - "Uninteresting" strategies
    - Mining
    - Micromanage attacks

21

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

22

Multi-tasking

- RTS games typically require player to manage many simultaneous tasks
  - Tasks are attentionally and cognitively far apart
    - Hence there is a cost in switching between tasks
  - Attention tunneling is usually detrimental

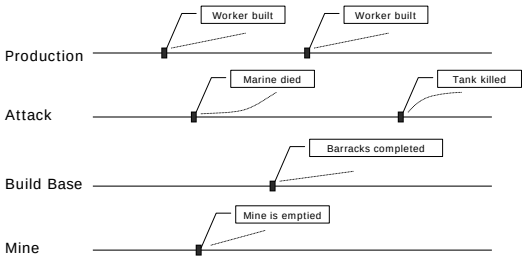
23

Channel Model Approach

- Group actions taken over the entire game into tasks
  - How to group is not yet finalized
  - Follow human tendency
- Each task is a *channel*
  - The agent performs some action on the channel then waits for feedback in the form of *Events*
  - Arrival times of events are not known
  - There is a cost incurred that is a function of the amount of time an event remains on a channel unprocessed

24

Channel Model



25

Task Switching

- The agent builds an internal model of expected event arrival times on each channel
- Accuracy of the model distinguishes novice and expert players
- Environmental cues help to determine when to switch
- Top-down and bottom-up control

26

Outline

1. Motivation for studying Real Time Strategy games
2. ORTS and AIIDE competition
3. SORTS Design
4. Multi-tasking in SORTS
5. Progress & Conclusions

27

Progress

- What is implemented
  - Perceptual system
  - Command system (Game Interface)
  - Low level FSMs
  - Planning
- What has to be implemented
  - Real Soar agents
  - Situation awareness

28

Progress

- Competition
  - Game 1 – 80%
  - Game 2 – 30%
  - Game 3 – 30%

29

Conclusions

- RTS games present a set of challenges to AI research that chess does not
- We are building SORTS to try to meet some of these challenges
- SORTS will make a good platform on which to test the new Soar architecture

30

Conclusions

- Nuggets
  - Provides a rich environment to test many of Soar's new capabilities
  - Forced us to confront issues that would not have come up in other environments / architectures
- Coals
  - Still in pre-alpha stage
  - Some decisions were made in the interest of competition performance rather than psychological plausibility
  - Abuses working memory

31

Visual Attention for a Real-Time Strategy Game

Sam Wintermute
University of Michigan

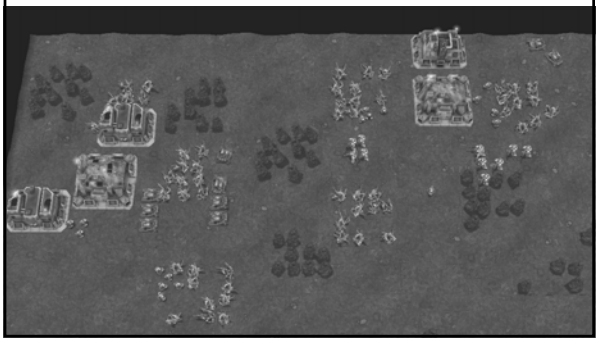
1

Introduction

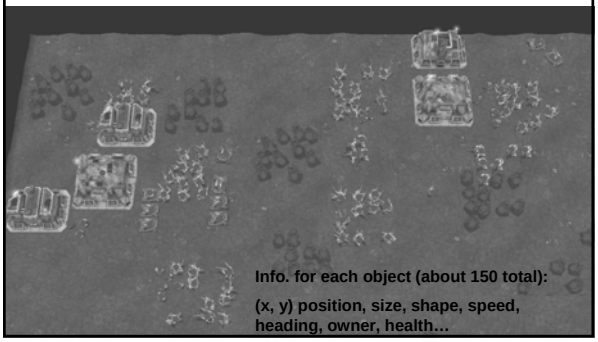
- RTS Games present lots of information to the player
- Fundamental problem: Which of this information should be selected, and how should it be presented to the Soar agent?

2

Typical (Small) RTS Game State



Objects Present



Desired Vision System Properties

- Grouping of similar objects
- Levels of abstraction (varying what "similar" means)
- Spatially-local selection
- General information about un-selected regions
- Efficient searching
- "Pop out" effects
- Constant-bounded input size

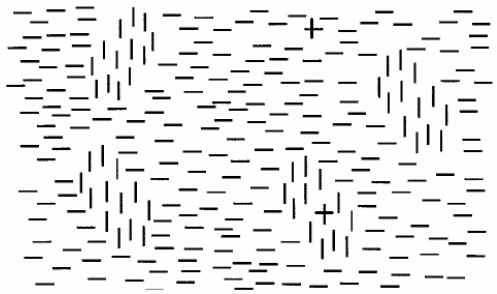
5

Human Visual Attention

- Gestalt grouping
- Object (group) based selection
- Feature Integration Theory
  - "Feature maps"- information about presence of object features in unattended regions
  - Allow for fast searches for unique features- for example, finding a red object amongst gray objects
- Top-down and bottom-up control

6

Example- Grouping, Features, Features of Groups

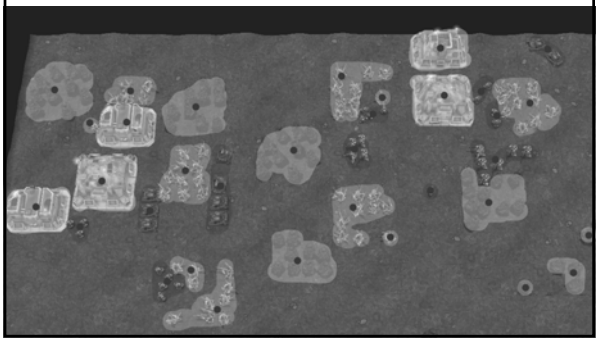


Implementation for Soar/ORTS

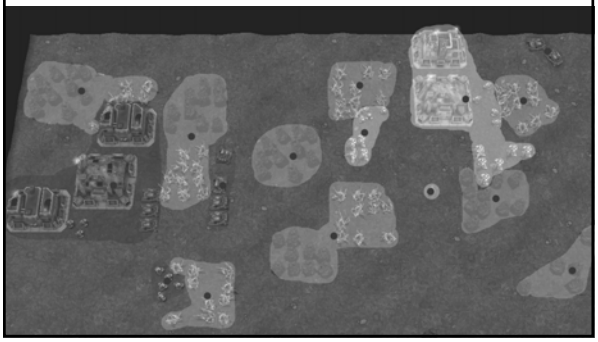
- Middleware -> Soar
  - Groups
  - Feature maps
- Soar -> Middleware
  - (x,y) position of focus center
  - Grouping parameters
  - “look at feature” commands
  - Visual range parameters

8

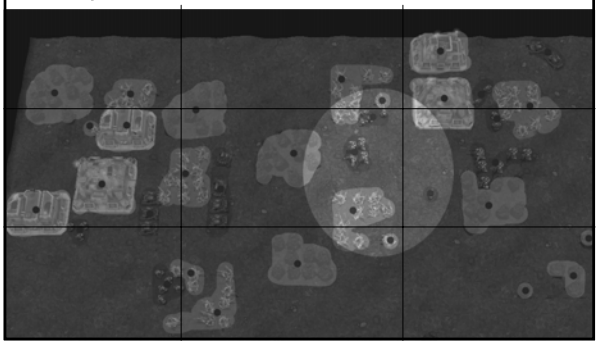
Grouping of Game Objects, by Type



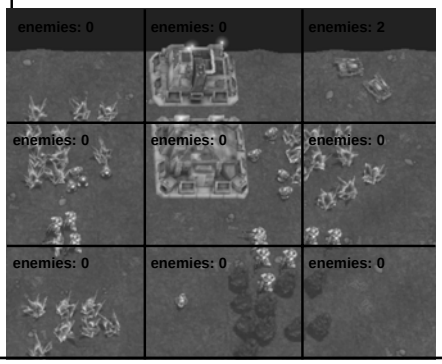
Grouping of Game Objects, by Owner



Spotlight of Attention



Feature Map Example



Realism vs. Optimality tradeoffs

- Soar can choose how many groups it sees
- Feature maps based solely on usefulness
- Manual / task-based grouping possible
- Split attention may be added
- Uniform resolution

13

Progress

- All vision commands have been implemented in the middleware
- Basic usage of commands has been tested
- Soar agent to do more complicated visual tasks under development

14

Nuggets and Coal

- Nuggets
  - Feature maps have proven straightforward and efficient for finding objects
  - RTS game domain seems a good fit for the system
  - Most of the non-Soar development is done
- Coal
  - Are we overly constraining our system?
    - AIIDE competition..
  - Non-simplistic agent development still has a long way to go

15

Incorporating Visual Imagery into a Cognitive Architecture

Theory, Design, and Implementation

Scott Lathrop
John Laird

26th SOAR WORKSHOP

OUTLINE

- BACKGROUND & MOTIVATION
- ARCHITECTURE
- TABLE SETTING DOMAIN
- GEOMETRY PROBLEM DOMAIN
- NUGGETS & COAL

2

WHAT IS VISUAL IMAGERY?

- What is larger, a softball or a baseball?

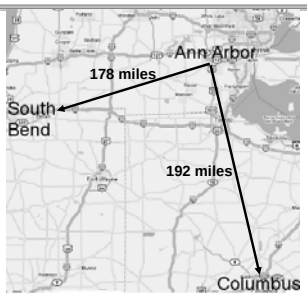
3

WHAT IS VISUAL IMAGERY?

- What hand does the Statue of Liberty hold the torch?

4

WHAT IS VISUAL IMAGERY?



- What city is closer to Ann Arbor: South Bend, Indiana or Columbus, Ohio?

5

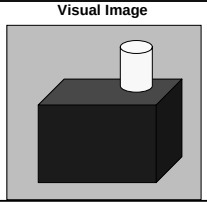
WHAT IS VISUAL IMAGERY?

Symbolic (Descriptive)	Visual Image (Depictive)
Explicit, abstract objects	**Implicit, concrete, objects**
Explicit relations	**Implicit relations**
Location, size, shape, features, orientation optional	**Location, size, shape, features orientation inherent**
Computationally efficient for maintaining semantic interpretations	**Computationally efficient for maintaining visual and spatial relationships**

Symbolic

Object (can) Yellow(Can)
Object(box) Red(box)
On(can, box)

Visual Image



*Adapted From Kosslyn Image and Mind

6

WHY STUDY VISUAL IMAGERY?

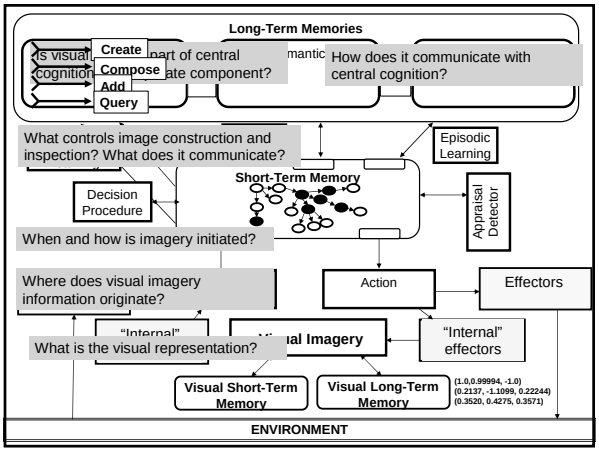
- "Best of both worlds" multi-modal approach
- General AI Agent/Cognitive Architectures
  - Symbolic representations & computations
  - Fewer efforts to integrate sensory modalities
- Depictive Representations
  - Visual and spatial format
  - Computationally more efficient for visual-spatial tasks.
  - Requires less domain knowledge for visual-spatial tasks
- Applicable to visual-spatial domains

7

RESEARCH GOALS

- To incorporate visual imagery within the context of a cognitive architecture constrained by psychological and biological evidence
- To improve the spatial reasoning capability of Soar
- To understand visual imagery's functional capabilities and limitations
  - What are the environment and task conditions where visual imagery provides additional capabilities to a general cognitive architecture?
  - Under what environment and task conditions is it computationally more efficient than a symbolic representation?
  - When does it require less task knowledge?

8

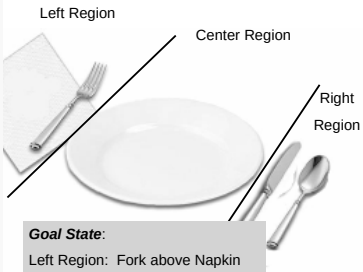


THEORY & ARCHITECTURE PROBLEM DOMAIN

Initial Knowledge

- Fork, Napkin, Plate, Knife, Spoon
- Place Setting made up of above objects
- Local Relationship
  - On-top(Fork, Napkin)
  - Left-Of (Fork, Plate)
  - Right-Of (Knife, Plate)
  - Right-of (Spoon, Knife)

• *No knowledge of global relationships*



Goal State:
Left Region: Fork above Napkin
Center Region: Plate
Right Region: Knife left of spoon

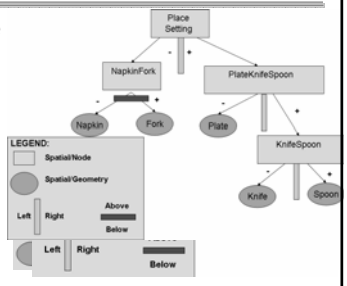
10

SYMBOLIC REPRESENTATION

11

VISUAL REPRESENTATION

- Scene Graph / BSP Tree
  - DAG
  - Represents a region of space
- Desired Properties
  - Hierarchical
  - Combines Geometric and Spatial data
  - Logical and Spatial Groupings (ideally)
  - Renderable to 2D image



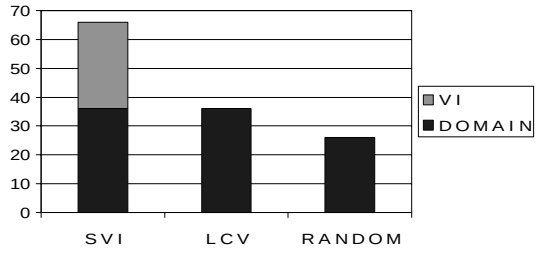
12

EVALUATION CRITERIA

- **Knowledge – Number of productions (Long term knowledge)**
- **Problem Solving – Number of decision cycles (count of reasoning steps in Soar)**
- **Computational Efficiency**
  - Time in Soar kernel (central cognition)
  - Time in Visual Imagery
  - Total Time
- **Functional Capability – What additional capability does visual imagery provide?**
- **Constraints – Must work with given behavioral, structural, functional, and computational constraints**

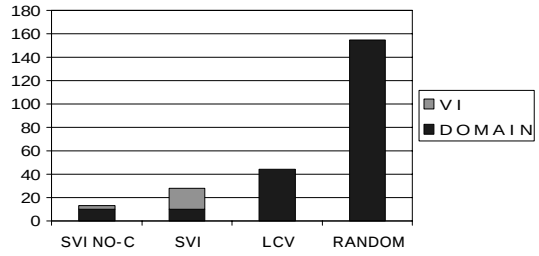
13

PRODUCTION COUNT



14

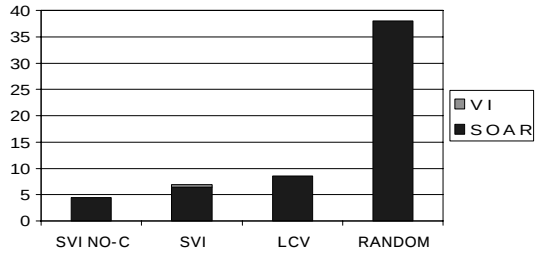
DECISION CYCLES



* Median over 30 trials

15

TIME (ms)



* Median over 30 trials

16

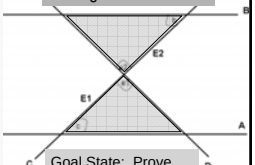
EXPERIMENT #2: GEOMETRY PROBLEM

Initial Knowledge

- 4 lines (A, B, C, D)
- A is parallel to B
- C intersects A
- D bisects the line segment formed by the intersection of C and A and C and B
- If something is a bisector, then it divides line segment into two congruent segments
- If two angles are alternate interior angles, then they are congruent
- If two angles are vertical angles, then they are congruent
- (ASA Rule). If two angles and the included side of one triangle are congruent to the corresponding two angles and included side of another triangle, then the triangles are congruent

What's Missing

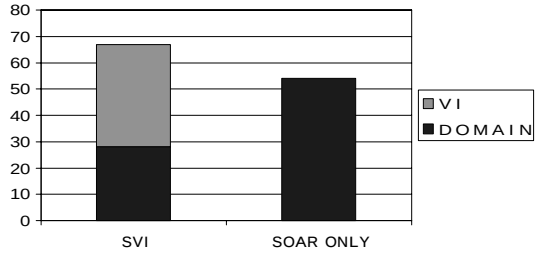
- Angles
- Points
- Line Segments
- Triangles



Goal State: Prove the two triangles congruent

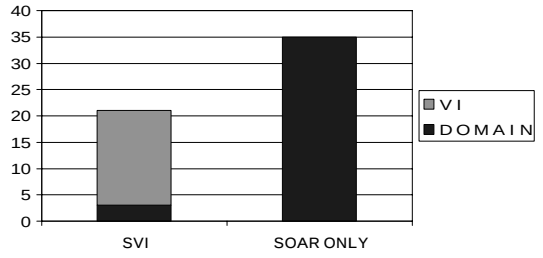
17

PRODUCTION COUNT



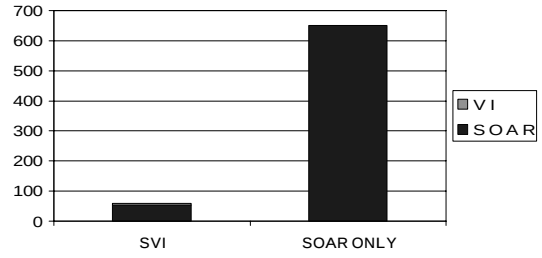
18

DECISION CYCLES



19

TIME (ms)

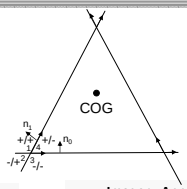


* Median over 30 trials

20

FUNCTIONAL CAPABILITY

- "Sense of direction"
  - Required to find triangle's interior angles
  - Symbolic approach alone not capable without providing more knowledge



Symbolic Approach

- One angle per region (1,2,3,4)
- Which angle belongs to the triangle?
- Which side of the line does each angle belong to?

Imagery Approach

- Find triangle center of gravity (COG)
- For each triangle line segment, determine side COG is on (+/-).
- Triangle angle at each vertex corresponds to region (+/+, -/+, -/-, +/-)

21

KEY POINTS

- Central Cognition
  - Domain knowledge (problem space, goals, states)
  - Controls visual image construction and inspection
- Visual Imagery
  - Constructs and searches what it is "told"
  - Provides perceptions based on what it "sees"
    - Visual properties
    - Spatial relationships
  - Is not solving problem (it has no notion of problem space, goals, states, etc.)
  - Performs efficient computations and provides new knowledge
  - Reacquires concrete knowledge "abstracted away" during initial perception

22

NUGGETS & COAL

- NUGGETS
  - Depictive representations show promise for achieving computational efficiency in visual-spatial domains
  - Additional functional capabilities for spatial reasoning
- COAL
  - Still looming questions as to what is the image's internal representation? Is there more than one format/data structure?
  - Lot of unknowns about the relationships between vision and visual imagery
  - Truth maintenance between central cognition, imagery, and vision

23

QUESTIONS

24

Soar Technology
Thinking *inside* the box.

SOAR26:
SPatial And Temporal Reasoning (SPAT-R)

Randy Jones – Scientific Advisor
Jack Zaientz – Scientific Advisor
Jens Wessling – Project Manager
Brian Stensrud – Research Scientist
Jonathan Beard, Sean Lisse, David Ray – Research Engineers

What is Spatial/Temporal Reasoning?

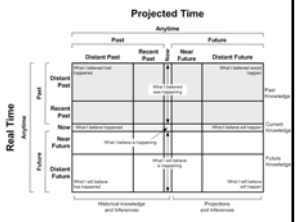
- Humans reason about space and time through both *quantitative* and *qualitative* assertions and relationships
  - qualitative
    - *"that object is closer to me than this object"*
    - *"I am inside this room"*
    - *"that event occurred a long time ago"*
  - quantitative
    - *"that event happened 6 minutes and 30 seconds ago"*
    - *"My current position is (23.2, 100.4)*
    - *"object B lies 6 meters closer to me than object C"*

What is Spatial/Temporal Reasoning? (cont'd)

- Humans use these assertions to make decisions in their environment
- Spatial and Temporal Reasoning are the **processes by which** these assertions (or beliefs, in some cases) are derived or calculated
- **To be effective in real-time environments, cognitive agents must also have the capability to generate and use assertions related to space and time**

What is the SPAT-R project?

- SPAT-R is the SPatial And Temporal Reasoning project
- Created as a spinoff of the BINAH project
- Focused on developing re-usable spatial & temporal reasoning capability (as previously defined) for Soar (and other) agents



How can SPAT-R help agents?

- Provides a general spatial, temporal, and spatio-temporal reasoning representation and inference capability
- Permits the leveraging of spatio-temporal reasoning knowledge from one domain application to another
- Forms a common company metaphor for spatio-temporal reasoning, lowering the barrier for behavior-developer ramp-up

Why are we working on SPAT-R right now? (Scientific Motivations)

- Spatial and temporal reasoning are current, relevant, and largely unexplored research areas
- Plenty of room for scientific exposure in this area
- Plenty of opportunities for funded research efforts to extend the SPAT-R appliance

Why is SPAT-R better than the alternative?

- Cognitive rule-based systems are not generally efficient at mathematical calculation
- Majority of mathematical calculation in Soar agents often supports spatial and/or temporal reasoning
- Extensive mathematical calculations can make cognitive agents
  - **Difficult** to develop *(hard to do in Soar)*
  - **Brittle** in execution *(more opportunity for failure)*
  - **Expensive** in computation *(many steps required)*
- SPAT-R appliance moves most calculation out of the Soar agent but makes it easy for a Soar agent to use
  - **Easier** to develop *(it is a re-usable appliance)*
  - **Robust** in execution *(simple representation)*
  - **Efficient** in computation *(complex calculation offloaded)*

What can SPAT-R do for applications?

- Powerful tool for agent development
  - Agents frequently have to reason about space and time as preconditions to actions and decisions
  - The SPAT-R tool will provide developers with a tool to generate these preconditions, so that they can spend more time on domain-specific development
- Isolating spatial reasoning from the rest of the agent has several benefits
  - Supports rigorous engineering while avoiding the Soar kernel's data overhead
  - Enhances code reuse & portability
- We often confront the question of how to interface symbolic to non-symbolic reasoning & memory
- Our emerging customer base for C^3 agents and systems deals a lot with strategy and tactics

What is the current 2006 scope of SPAT-R?

- Spatial Reasoning (IN SCOPE)
  - General spatial reasoning capability not available
  - Almost every project needs, but rebuilds from scratch
  - A reusable general purpose
- Temporal Reasoning (**NOT** IN SCOPE)
  - Agent-based temporal reasoning module already investigated on BINAH project
- Spatio-Temporal Reasoning (**NOT** IN SCOPE)
  - Future Work

How does SPAT-R appliance work?

- Agent architecture agnostic spatial operation toolset
- Built on top of SoarTech's "information management system"
- Accessible to Soar agents as an ATE plugin
- Input/Output-link interface
  - Region definition, operator definition, queries
- Snapshot Query/Response and Persistent Query capabilities

SPAT-R Interface Elements

- Region Definition
  - qualitative spatial region definitions with frame-of-reference, dimensionality, and dimensional projection
- Comparison Operator Definition
  - most frequent operations on regions will be intersection testing "is region X enclosed by region Y?"
- Relationship queries
  - inside/outside, above/below

SPAT-R Region Definition

- **0-D/1-D 'Region' Types**
  - **Points, Route Segments, Routes**
- **2-D Region Types**
  - **Circular regions, polygonal regions, composite 2-D regions**
- **3-D Region Types**
  - **Spherical regions, cylindrical regions, N-gon regions, hyper-polygonal regions, generic 3-D regions, composite 3-D regions**

Comparison Operator Definition

- Comparison Operator Definition
  - most frequent operations on regions will be intersection testing "is region X enclosed by region Y?"
  - intersection tests include region transformation by dimensionality
    - i.e., asking a 2-D question from the reference of a plane whether a point is within a region which was originally defined as a 3-D sphere but is projected as a circle for the purposes of generating the result set

X in Y?	X is 0-d	X is 1-d	X is 2-d	X is 3-d
Y is 0-d	X == Y	N/A	N/A	N/A
Y is 1-d	X < Y	X < Y	N/A	N/A
Y is 2-d	X < Y	X < Y	X < Y	N/A
Y is 3-d	X < Y	X < Y	X < Y	X < Y

Relationship Queries

- Domain-specific
  - "is the **aircraft** in my **flight corridor**?"
  - "are any **enemies** on my **left flank**?"
- Composed from operator definitions
- Return those data elements which resolve as "true" (matching the query tests) in a result set

What kinds of projects could benefit from SPAT-R?

- Airspace control:
  - dependent on spatial & temporal assertions
- Aviator behaviors:
  - Spatial understanding required in formation or route flying
- Intelligence analysis:
  - Spatial & temporal correlation of intelligence hypotheses
  - Spatial and temporal assertions made by the agent can be converted to graphical or textual notations displayed on the GUI
- C4ISR:
  - external planning & navigation systems require spatial/temporal assertions
- Pedagogic training systems:
  - Director needs real-time awareness of trainee's position and orientation relative to other objects and NPCs
  - Director is responsible for the proper timing of events in each training scenario

How could SPAT-R appliance help future work?

- Prevents re-invention of the wheel
  - Availability of a re-usable spatial & temporal reasoning capability (as previously defined) for Soar (and other) agents
- Additional feature at low cost
  - Permits the leveraging of spatio-temporal reasoning knowledge from one domain application to another
- Leverages well-considered representations and interfaces
  - Provides a general spatial, temporal, and spatio-temporal reasoning representation and inference capability
- Reduces resource mismatch risk
  - Forms a common metaphor for spatio-temporal reasoning, lowering the barrier for behavior-developer ramp-up

What have we accomplished?

- IR&D project initially approved - late November 2005
- Outreach requirement gathering from stakeholders - December 2005
- Project planning & "Internal 1st Stage Customer" (ISAT) defined –December 2005/January 2006
- Additional scientific & engineering resources brought on board - February 2006
- Created socialtext repository for team work products – February 2006
- Gathering results of initial scientific investigations - late February / early March 2006
- Appliance functionality & interface design – April/May 2006
- Initial Draft Implementation – May/June 2006

Backup Slides

Player Modeling in IDA

Dr. Brian Magerko
University of Michigan

Interactive Drama

2

The Boundary Problem

- Author defines a *story space*
  - The space of intentionally dramatic stories possible
  - Consonance between player actions and authored content
- Player actions may lead to a world state outside the boundary of the story space

The dramatic experience may stall or even halt…

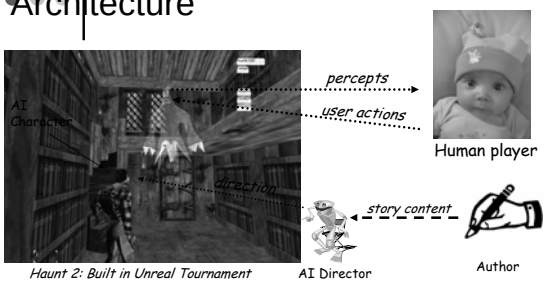
3

Contributions

1. Incorporation of semi-autonomous characters
2. A *complete* story director with..
   a. Player knowledge modeling
   b. Player prediction
   c. Both reactive and preemptive direction for addressing boundary problems
   d. Two-tiered selection of content
3. A story representation that supports:
   a. Reactive & preemptive direction
   b. Pacing
4. Evaluation via archetypes

5

IDA: An Interactive Drama Architecture



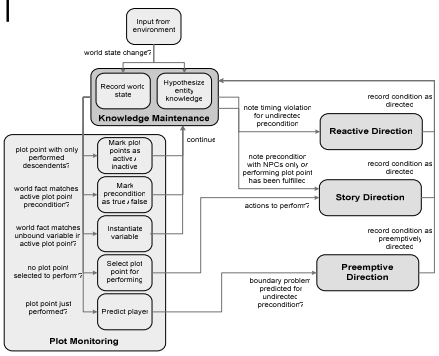
6

Director

- Omniscient & omnipotent intelligent agent
- Executes “direction”
  - Enacts story content
  - Mediates between player actions & plot
  - Influences player behavior to stay within story space
    - Based on current actions
    - Based on predicted actions

7

Direction Execution Cycle



8

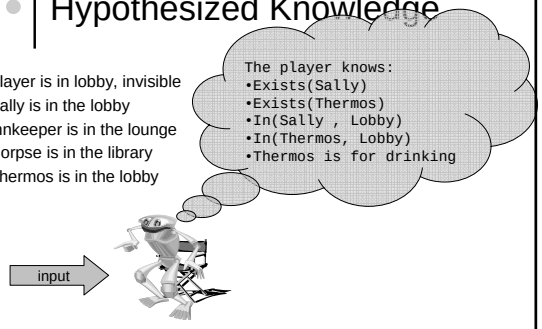
Knowledge Maintenance

- Records when world changes significantly
- Observes story world omnisciently
- Creates new WMEs to reflect changes in the state of the world
- Hypothesizes basic situational awareness of entities in the story world
- Applies ontology of everyday world knowledge

9

Hypothesized Knowledge

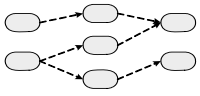
- Player is in lobby, invisible
- Sally is in the lobby
- Innkeeper is in the lounge
- Corpse is in the library
- Thermos is in the lobby



10

Plot Monitoring

- Occurs when plot content matches with current knowledge base
  - State of the world
  - Mental states
- Plot:
  - Content: plot points
    - Preconditions
    - Actions
  - Structure:
    - Ordering constraints
    - Timing constraints



11

Plot selection

- What if there is more than one candidate for performance?
- Two-tiered process
  - Player-motivated
  - Director-selected
- Selected plot point marked

12

Reactive Direction

- Immediate response to boundary problems
- Relies on more *effective* director strategies
- Boundary problems signaled by an *occurrence of timing violations*

13

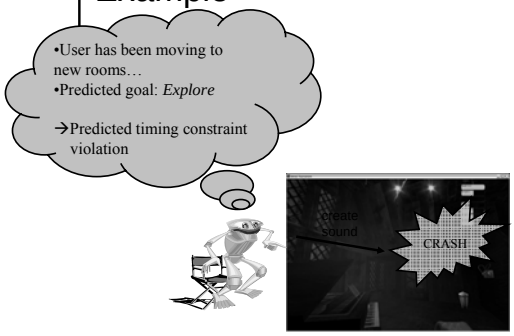
Preemptive Direction

- Preemptive response to predicted boundary problems
- Relies on more *subtle* director strategies
- Queries predictive model of player behavior

More believable than relying only on reactive measures or directing continuously

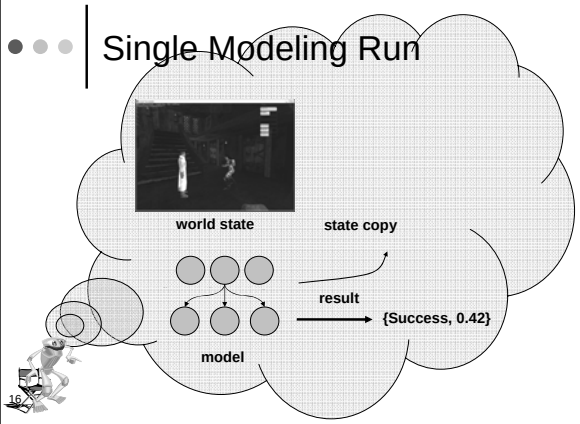
14

Example



15

Single Modeling Run



16

Player Prediction

- Success
  - Precondition is fulfilled before a timing constraint is violated
- Failure
  - Timing constraint is violated
- Result determines:
  - *if* the director preemptively directs
  - *how* it directs preemptively

17

Probabilistic Sampling

- Player model
  - Returns a tuple, M → (R, P)
    - R: success / failure
    - P: probability of particular sequence of actions chosen
  - Runs created iteratively until an author-defined limit ρ is reached
- Director computes confidence in the user fulfilling plot content (C_m)
  - Function of likelihood of each run & its result
  - $C_m > \alpha$ → success
  - $C_m < \alpha$ → insignificant success *or* failure

18

Connecting Modeling to Direction

- Direction chosen by best *score*
- Scores dynamically assigned as function of modeling result, C_m
- Each director action rated by a scoring function:
  $Score_{action} = (Sub * S_{action} + Eff * E_{action}) / 2$
  - S_{action} / E_{action}: authored rating for a particular director action
  - Sub / Eff: weights assigned at run-time
- Example: *transport-player* {S = 0.05, E=.9}

19

Quantitative Evaluation

- Play as archetypes
  - Explorer
  - Chaser
  - General
- Experimental groups
  - Modeling on
  - No modeling
    - Model still run
    - Preemptive direction "turned off"
- Measures
  - Boundary problems
  - Frequency of direction
  - Average subtlety of direction

20

Results

- No difference for avg subtlety and # of directions
  - Director should direct the same #
  - Small variability in ratings or problem in authorship

21

Results (cont.)

- Boundary problems
  - modeling < no_modeling ($p < 0.01$)
  - Within-groups:
    - General: no effect
    - Chaser: no effect
    - Explorer: $p < 0.01$
- Why?
  - Lack of general coverage of predictive model
  - Explorer easily "most accurate" hand-encoded model
  - Points to the need for adaptive modeling

22

Future Work

- Authoring tools
- Categorization of director strategies
- Player modeling
  - Unified model
    - Knowledge model
    - Behavioral model
    - Skill model
  - Adaptive models
- Adaptive story for education
  - Story as engagement
  - Experience tailored for dramatic and learning value

23

Nuggets

- A *complete* story director with..
  - Player knowledge modeling
  - Player prediction
  - Both reactive and preemptive direction
  - Two-tiered selection of content
- A story representation that supports
  - Reactive & preemptive direction
  - Pacing
- Evaluation via archetypes
- Successful defense (!)

24

Coal

- *Haunt 2* did not wind up being a complete or robust experience
- Starting over with a new domain
- No rigorous user study done in evaluation
- Possible bias in evaluation
  - Parameter values
  - Archetype & individual sensitivity to direction

25

Interactive Storytelling Architecture for Training (ISAT)

Brian Magerko, Lisa Holt, & Brian Stensrud

RDECOM

ECS

games for entertainment & learning lab

Soar Technology
Thinking *inside* the box.

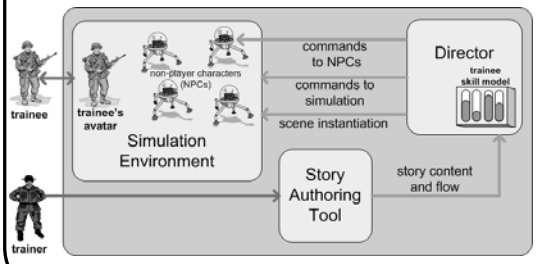
Project Team

- Michigan State GEL Lab
  - Brian Magerko, Ph.D. (co-PI)
  - Ben Medler (RA)
- Soar Technology
  - Lisa Holt, Ph.D. (co-PI)
  - Brian Stensrud, Ph.D. (co-PI)
  - Al Wallace (PM)
  - Ann Marie Steichmann (systems integration lead)
  - Robert E. Wray, Ph.D. (scientific consultant)
- ECS
  - Larry Kayne (PM)
  - Howard Mall (technical lead)
  - Seth Frolich (art & modeling)
  - Ben Quintaro (software engineer)

ISAT Overview

- Combines interactive storytelling and intelligent tutoring for effective training
- Benefits of interactive storytelling via simulation & game-based training
  - Distributable (any time, any duty station)
  - Readily available (as opposed to human instructor)
  - Engaging & realistic
- Benefits of intelligent tutoring
  - Direct connection to training goals
  - Individualized training
  - Guidance and feedback based on assessment of performance

High-Level ISAT Architecture



ISAT Progress to Date

- Director
  - Implementation of various types of Director actions
  - Implementation of skill model
- Enhancements to TC3 Simulation
  - Character spawning
  - Lua scripting
  - Navigation mesh
- Authoring tool prototype
  - Development & evaluation
  - Defined XML format for map input

ISAT Progress to Date (cont.)

- Integration
  - Director agent & TC3 simulation
  - Director agent & authoring tool
- General
  - Documentation of correct treatment procedures
  - Mapping of simulation actions to skills

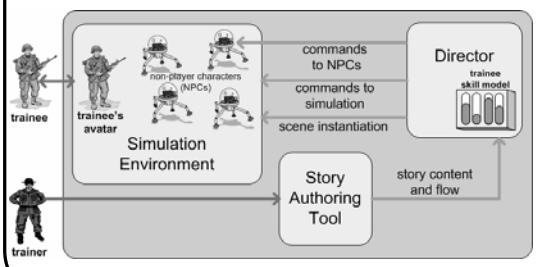
Tactical Combat Casualty Care (TC3)

- Baseline simulation developed by ECS, Inc. through RDECOM-STTC funding
- Combat medic simulation using computer game technology
- First person 3-D environment

- Training for triage and treatment procedures
- Focuses on casualty care

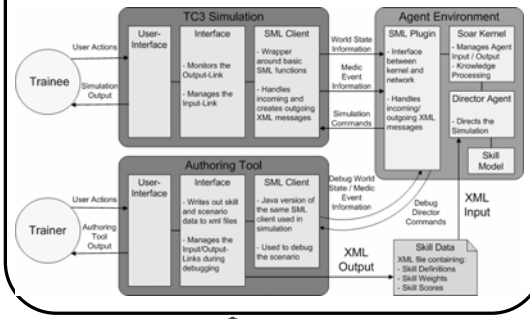
4 Oct 2005 Slide 7 7

High-Level ISAT Architecture



4 Oct 2005 Slide 8 8

ISAT Architecture Details



4 Oct 2005 Slide 9 9

The ISAT Director

- Directs content and flow of training scenario
  - Selects and instantiates each scene
  - Generates required events and objects within each scene
  - Manages non-trainee characters and their actions
- Identifies trainee skill-proficiency
  - Maintains skill model to actively measures trainee's proficiency at each skill
  - Skill model values can be imported to or exported from the Director agent

4 Oct 2005 Slide 10 10

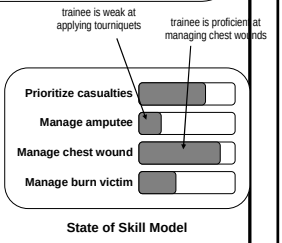
The ISAT Director (cont.)

- Actively responds to trainee errors within the scenario
  - Calls attention to the error or strongly guide trainee to correct behavior
  - Highlights or corrects errors when trainee take actions that move him outside the training experience (e.g. wanders off the map)
- Director actions are often dependent on state of trainee skill model

4 Oct 2005 Slide 11 11

The Skill Model

- Real-time scoring system for individual skills
- Continuously updated by the Director
- Used to evaluate trainee performance and adapt Director actions to trainee needs
- Not visible to trainee
- Individual trainee scores can be maintained and used as input to Director at execution-time



State of Skill Model

Updating algorithms not yet fully developed. Strawman implementation of skill model for purposes of research

4 Oct 2005 Slide 12 12

Error Types

- Different error types will affect the skill model in various ways
  - Omission
  - Commission
  - Out-of-order
  - Inappropriate action
- Director considers the type of error when assigning scores for each step
  - Omitting a step, for instance, may be more harmful than simply executing it out of order

4 Oct 2005 Slide 13  13

Direction Types

- Reactive Direction (from IDA)
- Story Direction (from IDA)
- Skill-based Direction (new)
  - Responds to trainee skill errors by executing actions within the environment
  - Direction selection based on both the nature of the error and state of the skill model
  - Scaffolding & fading
    - Can be indirect, e.g., changing the state of an object/NPC to affect future events
    - Can be direct, e.g., having the squad leader yell at the trainee that he has made a mistake

4 Oct 2005 Slide 14  14

Demo Preview

- Setting
  - a courtyard after a suicide bomb attempt
  - 4 casualties: one motionless (dead), one amputee, one burn victim (screaming) and one chest wound
- Primitive reactive direction
  - If trainee is inactive for a period of time, squad leader prompts trainee to “wake up & start treating casualties”

4 Oct 2005 Slide 15  15

Demo Preview

- Primitive scaffolding
  - Direction will vary depending on trainee skill level
  - If trainee is relatively proficient at prioritization of casualties and makes an error, cue will be subtle
    e.g., Amputee will begin screaming in agony “Aahhh! My arm!”
  - If trainee is not proficient at prioritization of casualties, cue will be very direct
    e.g., squad leader yells “There’s a man over there who’s lost his arm. He’ll die if you don’t tend to him soon.”

4 Oct 2005 Slide 16  16

Demo Preview

- Skill-based direction
  - Treating casualties out of order
    e.g., “I know that soldier is hurt but there are more serious casualties you need to deal with.”
  - Implementing tourniquet treatment steps out of order
    e.g., “Cut away that man’s sleeve before you apply the tourniquet.”
- Ultimately the Director will be able to take action in ways other than verbal cues

4 Oct 2005 Slide 17  17

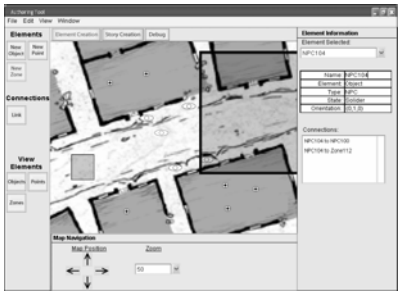
ISAT Demo

4 Oct 2005 Slide 18  18

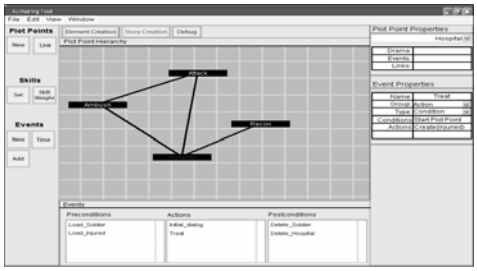
Story Authoring tool

- Difficult for non-programmer to encode & edit training content
- Graphical story editor & debugger
- Use: Non-programmer SME or Trainer
- Modes
  - Element placement
  - Story creation
  - Debugging

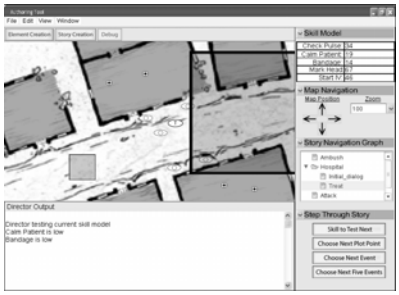
Element Placement (Prototype)



Story Creation (Prototype)



Debugging (Prototype)



Next Steps

- Director
  - Story Direction & Scene instantiation
  - More complex director actions
  - Refinement of skill model updating functions
    - Recency
    - Decay
- Authoring Tool
  - Integration with director
  - Import XML maps
  - Java implementation
- Evaluation

Nuggets

- Expands interactive drama concepts from IDA for education
- Significant progress in development
- Authoring tool prototype developed and informally evaluated
- Skills more rigorously defined
- New TC3 environment

Coal

- Evaluation subject pool still unclear
- Authoring tool implementation just beginning
- Strawman model updating
- No access to SME
- “S” in ISAT not yet visible
- Instantiation of plot content not yet clearly defined

4 Oct 2005 Slide 25

25

Soar Technology
Thinking *inside* the box.

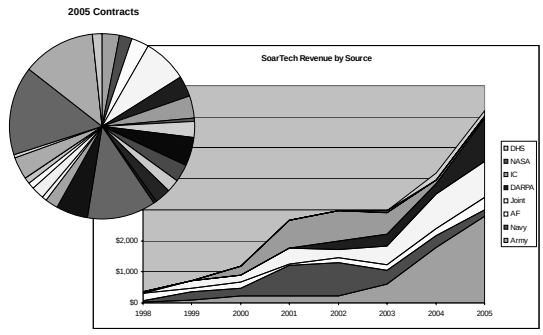
SoarTech Update

Jim Rosbe
25 May 2006

A busy year

- Revenue up 50% again in 2005
- ~30 projects …for 19 federal agencies & prime contractors & with 12 subcontractors
- At DARPA, projects for IXO, IPTO, ATO, DSO
- Notable new starts since Soar 25:
  - Integrated Battle Command (IBC) (DARPA)
  - BICA (DARPA)
  - ISAT II (PEO STRI)
  - ROCCIE II (CERDEC)
  - AATC II (AMRDEC)
  - SYNC (AFRL)
  - TANGRAM (DTO)
  - Boeing Virtual Warfare Center
- New starts pending
  - CASE (DTO)
  - OneSAF Objective Systems Models & Tools (PEO STRI)
  - HEAT/ROCCIE III (CERDEC)
  - Computational Narratives (DARPA)

Revenue Sources



Growth…

- 17 new staff …& at least 3 more to start this summer … scientists, researchers, engineers, managers, admin staff:

Thom Bartold	Kate Harding	Kathy Putt
Bob Bechtel, PhD	Echo Harger	Doug Reece, PhD
Vladimir Bouchev, PhD	Geoff Morgan	Toby Tripp
Phil Donate	Kate Pierro	Deb Webster
Cory Dunham	Joyce Potter	Alan Vayda, PhD
Steve Furtwangler	Ann Marie Steichmann	

  - New staff in Orlando in next months: Susan Eitelman, Angela Woods, Gil Barrett
  - & two Ann Arbor interns: Bethany Soule and Aaron Kluck
- Expanded our offices twice last year in Ann Arbor; probably more this year …and increasing our space in Orlando

A busy year (cont.)

- New teaming relationships
- Additional management & admin staff
- Recruiting
- New systems
- More formal planning and structure for IR&D investment, engineering processes and preparation for productization
- 20 Conferences…& sponsored AAMAS, AIIDE, Soar Workshop
- 5 kids born…2 more on the way…

Nuggets… & Coal

- The quality of the staff we were able to recruit, and how many people in our field contacted us expressing interest in working for us
- The number of organizations who approached us to team with them
- Availability of talent
- More opportunities than we can respond to
- Maturing & packaging our technology both for ourselves and for future markets is a continuing challenge

Soar Technology
Thinking inside the box.

IF-Soar: An Autonomous Fire Direction Center for Indirect Fire Training

Brian Stensrud
Glenn Taylor
Jacob Crossman

Soar Technology, Inc.
May 25, 2006

Agenda

- Overview
  - Description of Indirect Fire
  - The Call for Fire
  - IF-Soar's Job Description
- Connected Components and Architecture
  - JSAF
  - SoarSpeak
  - System Architecture
- Adjust Fire: An Extended Example

Agenda

- IF-Soar Design
  - New Goal System (NGS) agent design paradigm
  - Dialog Management
- Nuggets/Coal

What is Indirect Fire?

- Any fire on a target where the firing entity does not take direct visual aim on the target
- Typically artillery fire
  - Howitzers
  - Mortars
  - Naval cannons
- Executing indirect fire missions involves three functional elements
  - Forward Observer (FO) to identify and describe the target
  - Fire Direction Center (FDC) to process the mission and provide appropriate parameters to the firing entities
  - The gun(s) to do the shooting

The Call for Fire (CFF)

- Structured sequence of messages from the FO to the FDC
- Mechanism for delivering all necessary fire mission parameters and requests to the FDC
- Organic CFFs consist of three lines
  1. Observer identification and warning order
  2. Target location
  3. Target description, method of engagement, method of fire and control

An Autonomous Fire Direction Center

- IF-Soar is a model of an autonomous FDC
- Processes incoming fire requests from human FO
- Executes fire requests within JSAF simulation environment
- Communicates with FO through
  - Acknowledgments to the CFF
  - Message to Observer (MTO)
  - Fire/detonation confirmations

Agenda

- Overview
  - Description of Indirect Fire
  - The Call for Fire
  - IF-Soar's Job Description
- Connected Components and Architecture
  - JSAF
  - SoarSpeak
  - System Architecture
- Adjust Fire: An Extended Example

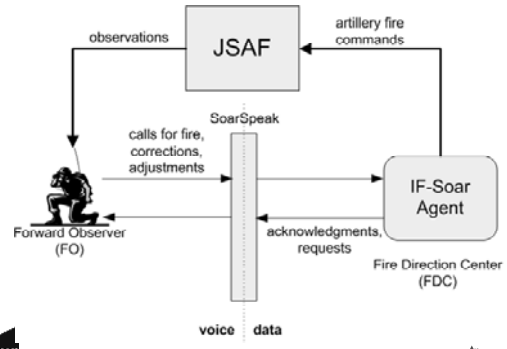
JSAF

- Simulation environment
- 2D planned-view display (PVD) for observing entities and fires effects
- Provides capability for other computer-generated forces (CGFs) to populate the environment

SoarSpeak

- Collection of applications for providing text-to-speech (TTS) and speech-to-text services for communication with Soar agents
- XML-based configuration
- Support for several network protocols including TCP/IP, HLA and DIS
- Support for multiple speech-to-text engines including Nuance and IBM's ViaVoice
- AT&T's NaturalVoices text-to-speech engine
- Semantic language parser ANGST converts recognized text to readable form for IF-Soar agent

System Architecture



Agenda

- Overview
  - Description of Indirect Fire
  - The Call for Fire
  - IF-Soar's Job Description
- Connected Components and Architecture
  - JSAF
  - SoarSpeak
  - System Architecture
- Adjust Fire: An Extended Example

Adjust Fire/Fire for Effect

- Adjust Fire missions allow FO to fire single rounds of munitions at targets
- Typically used when FO needs to iteratively determine coordinates for an effective artillery strike
- After observing effects of an adjust fire mission, FO can call an 'adjustment' to generate an identical shot at a slightly altered location
- When FO has determined the appropriate coordinates, a Fire for Effect mission can then be called to deliver multiple rounds on target

An Adjust Fire Mission

- FO (human, jaguar01):
  "A3R51, this is jaguar01, adjust fire, over"
- FDC (IF-Soar, A3R51):
  "jaguar01, this is A3R51, adjust fire, out"
- *"grid 6 3 5 9 2 7, over"*
- **"grid 6 3 5 9 2 7, out"**
- *"correction, grid 6 4 5 9 2 7, over"*
- **"grid 6 4 5 9 2 7, out"**

An Adjust Fire Mission (cont.)

- *"seven tanks in the open, over"*
- **"seven tanks in the open, authenticate BRAVO JULIET FOXTROT, over"**
- *"I authenticate ECHO, out"*
- **"good authentication, out"**
- **"hotel, D P I C M, 1 round, target AA0001, over"**
- *"hotel, D P I C M, 1 round, target AA0001, out"*

Agenda

- IF-Soar Design
  - New Goal System (NGS) agent design paradigm
  - Dialog Management
- Nuggets/Coal

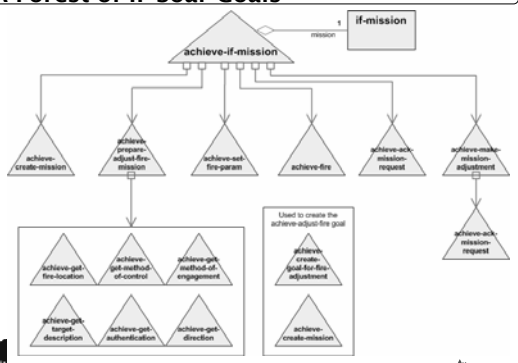
New Goal System (NGS)

- An implementation of the 'Forest of Goals' behavior design paradigm
  - Declarative goals
  - No operator hierarchy or persistence
- Exists as a library for Soar 8
- Includes assortment of Tcl macros that reproduce common LHS and RHS production segments
  - Operator proposal
  - Operator application
  - Goal creation

Overview of IF-Soar Design

- General
  - NGS v1
  - 1954 productions
  - *o*-supported goals (mostly)
- Goals used to
  - Logically filter incoming information from FO
  - Handle incoming and outgoing communication
  - Generate outgoing fire parameters
- XML Mission configurations
  - Callsign of IF-Soar agent (e.g. A3R51)
  - Locations of FO and available batteries
  - Available munition types and quantities
  - Reference IDs for known points

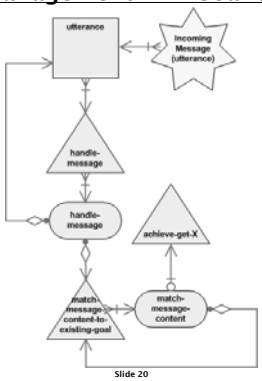
A Forest of IF-Soar Goals



Dialog Management in IF-Soar

- Standard messages and corrections
  - Format of lines defined in SoarSpeak grammar
  - **correction** modifier tells agent to overwrite existing data
  - Agent uses series of goals to fetch the incoming message and append it to appropriate retrieval goal
    - *handle-message* fetches message from input-link, hands it off to…
    - *match-message-content-to-existing-goal* parses information from message, appends arguments to…
    - *achieve-get-**X*** goals
      - *achieve-get-fire-location*
      - *achieve-get-target-description*
      - …

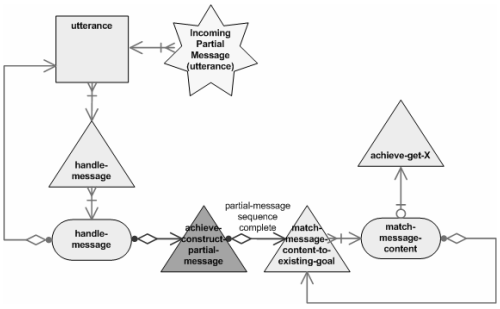
Dialog Management in IF-Soar (cont.)



Dialog Management in IF-Soar (cont.)

- Partial messages
  - Helped to mitigate the problem of poor speech-recognition performance
  - Message fragments defined in SoarSpeak grammar, identified as a **partial-message**
  - Agent deciphers message type, creates new goal to collect incoming message fragments
    - Example from target description message:
      - *"three tanks in the open,"*
      - *"willy pete point detonating,"*
      - *"at my command, over"*
  - 'over' denotes end-of-message, agent collects the fragments and hands over to *match-message-content-to-existing-goal* as it would otherwise

Dialog Management in IF-Soar (cont.)



Agenda

- IF-Soar Design
  - New Goal System (NGS) agent design paradigm
  - Dialog Management
- Nuggets/Coal

Nuggets

- Robust system, capable of handling a variety of
  - warning-order types
  - munition types
  - target descriptions
- Works with JSAF, communicates with speech-recognition federate using HLA
- Supports corrections and partial messages, alleviating limitations of speech recognition

Coal

- Agent can get confused by utterances recognized by the grammar but not in context of the mission
- Agent does not consider munition ballistics or ranges when computing a fire solution
- Partial message capability
  - requires artificial expansion of SoarSpeak grammar (~15%)
  - message fragments only recognized in logical blocks

Demo of System

- If anybody is interested, come see me for a demo

Soar Technology
Thinking inside the box.

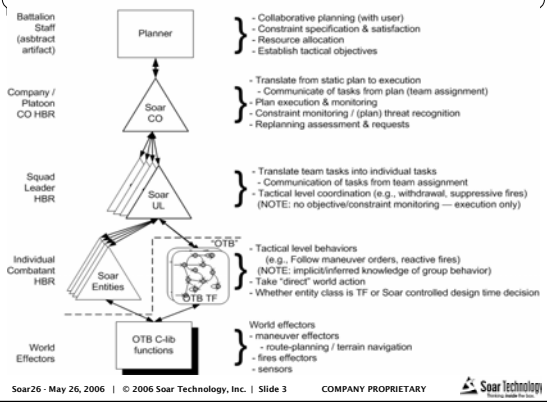
SOAR26:
Joint Fires & Effects Training System (JFETS)

Robert Wray, Randy Jones – Principal Investigator
Scott Sorel, Al Wallace – Project Manager
Jonathan Beard – Tech Lead
Geoff Morgan, Steve Furtwangler – Research Engineers

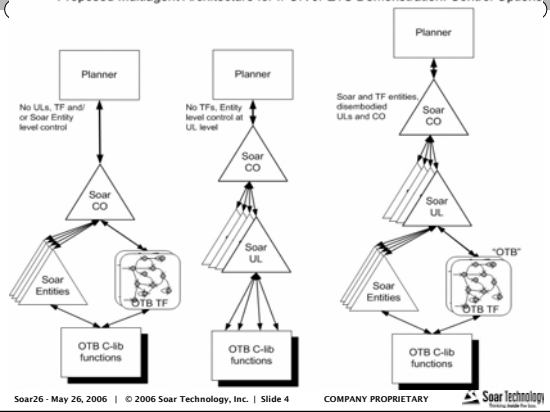
Project Details

- SoarTech working as a subcontractor to the Institute for Creative Technologies (ICT)
- Overall Objective
  - Build an immersive training environment for the Ft. Sill artillery school to train forward observers
- SoarTech Objective
  - Create believable and evocative insurgent behaviors for observation & interaction

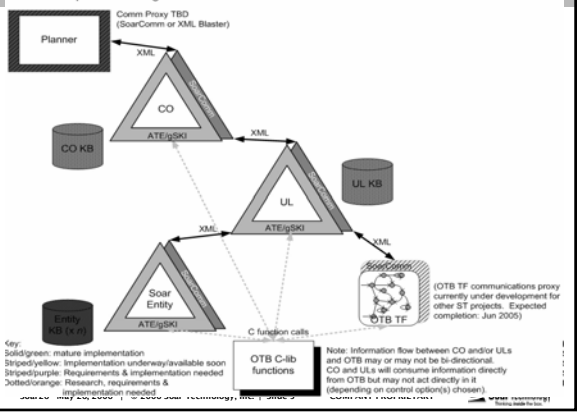
Proposed Multiagent Architecture for IFOR JFETS Demonstration: Component Descriptions



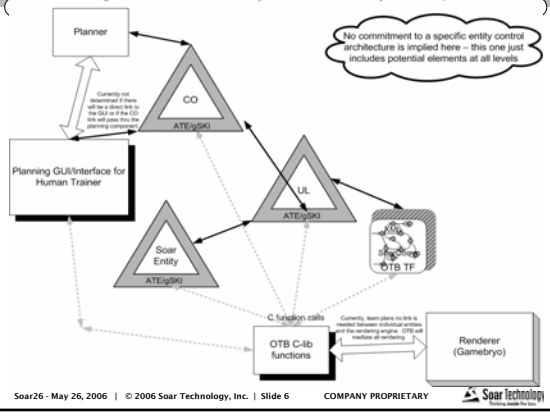
Proposed Multiagent Architecture for IFOR JFETS Demonstration: Control Options



Proposed Multiagent Architecture for IFOR JFETS Demonstration: Middleware

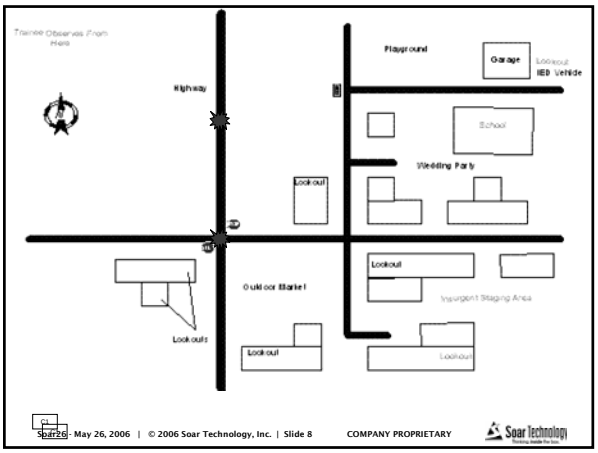


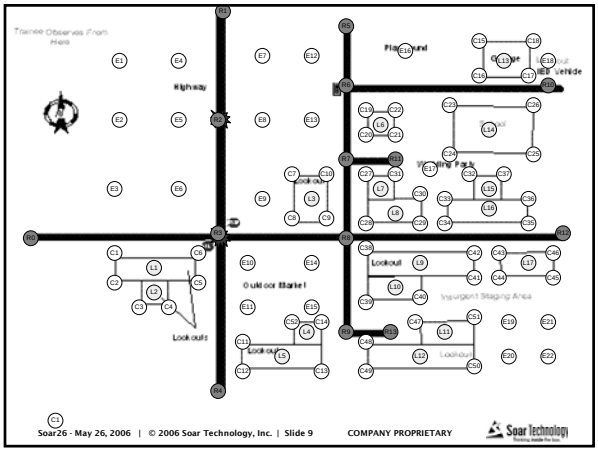
Defining Interactions between Entity Control and Other System Components

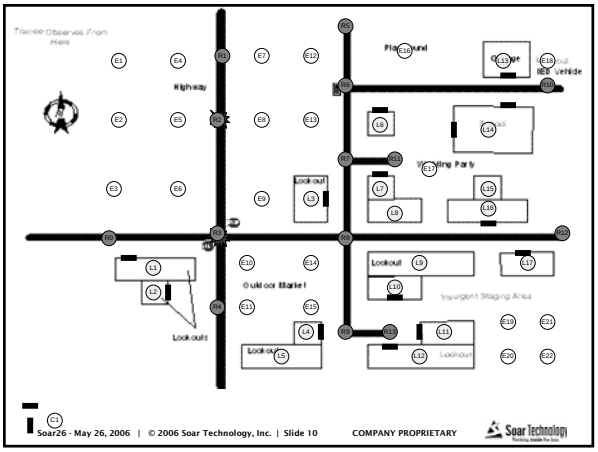


Soar Technology
Thinking *inside* the box.

Scenario







A Proposal For Changing Soar's Decision Procedure

JOHN E. LAIRD
26TH SOAR WORKSHOP

1

Soar Basics

- Activity is a series of operators controlled by knowledge:
  1. Input from environment
  2. Elaborate current situation
  3. Propose and compare operators via preferences
  4. Select operator
  5. Apply operator: modify internal data structures
  6. Output to motor system

Monotonic inferences

Non-monotonic inferences

2

Soar Basics

1. Input from environment
2. Elaborate current situation
3. Propose and compare operators via preferences
4. Select operator
5. Apply operator: modify internal data structures
   - *Non-monotonic change*
   - *Non-monotonic change*
   - *Non-monotonic change*
   - *Non-monotonic change*
   - *Non-monotonic change*
   - …

Long chain of operator application rule firings

6. Output to motor system

3

What's Wrong with Soar 8.6.2?

1. No bound on the number of rule firings during operator application
2. Prevents Soar from entering output/input
3. Soar can do many internal actions with "eyes closed"
4. Post-chunking can be less reactive than pre-chunking
5. Timing predictions can become very unreasonable

4

Proposal

- 1 PE phase/decision cycle: Limit operator application to one wave of parallel o-supported rule firing
- Forces architecture to go back through input-output
  - Making it more reactive
- Has no impact on operator selection
  - except from input
  - [Limiting elaborations to 1/decision would impact selection]
- Number of decisions doesn't collapse after chunking
  - Chunking will eliminate only unnecessary reasoning

5

Proposed Change

1. Input from environment
2. Elaborate current situation
3. Propose and compare operators
4. Select operator → Decision procedure: If current operator wins based on same acceptable preference, it stays selected (no blinking).
5. Apply operator
   - *One non-monotonic change*
6. Output to motor system

6

Example

Task: Count to 10 in subgoal: intermediate results in the super state.

Before Chunking

```
24: O: O24 (count-test4)
25: ==>S: S2 (operator no-change)
26:    O: O25 (substate-count)
27:    O: O26 (substate-count)
28:    O: O27 (substate-count)
29:    O: O28 (substate-count)
30:    O: O29 (substate-count)
31:    O: O30 (substate-count)
32:    O: O31 (substate-count)
33:    O: O32 (substate-count)
34:    O: O33 (substate-count)
35:    O: O34 (substate-count)
36: O: O35 (test-complete)
```

After Chunking

Soar 8

```
24: O: O24 (count-test4)
25: O: O25 (test-complete)
```

Ten rule firings in sequence

1 PE/Operator

```
42: O: O24 (count-test4)
43:    O24 (count-test4)
44:    O24 (count-test4)
45:    O24 (count-test4)
46:    O24 (count-test4)
47:    O24 (count-test4)
48:    O24 (count-test4)
49:    O24 (count-test4)
50:    O24 (count-test4)
51:    O24 (count-test4)
52: O: O25 (test-complete)
```

7

Nuggets and Coal

- Nuggets:
  - Make Soar more reactive
  - Makes post-chunking behavior more *reasonable*
- Coal:
  - Behavior might be less intuitive

8

What's New in Soar 8.6.2

Douglas Pearson
+ Bob, Jon, Karen, Taylor, John

May 25, 2006

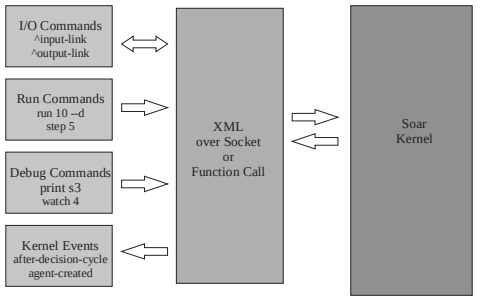
douglas.pearson@threepenny.net
soar-sml-list@lists.sourceforge.net

ThreePenny SOFTWARE

Quick Review of 8.6.0/8.6.1

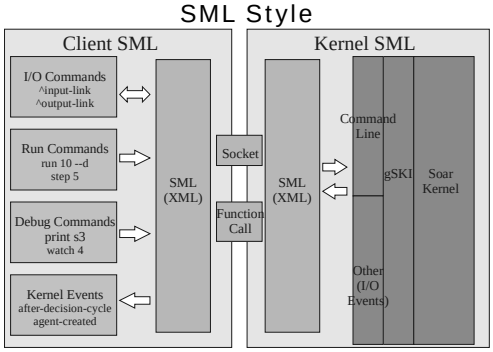
- Introduction of XML based interface to Soar
  - SML (Soar Markup Language)
- Opened the door to other languages
  - Java, C++, Tcl
  - New debugger in Java
- More flexible debugging
  - Embedding kernel into environment and debug remotely
  - Faster performance
- No kernel level changes
  - Just new way to connect environments & tools to Soar

Connecting to Soar SML Style



Uniform for entire interface

Connecting to Soar SML Style



Setting a "stop-point"

- In 8.5 and 8.6.1 "run n decisions" always stops after output
  - Hard to use "matches"
  - Want an easy way to control when to stop

- Green marker shows current phase
  - But only updates at end of a run
- Red marker shows where to stop (for "run n")
  - Click in GUI in debugger to change
  - Or "set-stop-phase --before –apply" etc.

"Run 0" and "Run n"

- "run 0"
  - Runs all agents up to the current stop-point
  - Quick way to synchronize agents to a phase
- Decisions vs Decision Cycles
  - Decision cycle ends after output
  - Decision occurs when select operator / impasse
  - "run n" ?
  - Run for n ***decisions*** and then to the stop-point
  - 8.6.2. not 100% compliant to this yet but close

Flexibly Interleaving Agents

- 8.6.1
  - Agents always interleaved by phase
  - Java TankSoar requires by output generation
- 8.6.2
  - Can interleave by elaboration, phase, decision, output
  - E.g. “run -o 3 -interleave d”
  - Combined with other changes => total rewrite of scheduler

ThreePenny Software 7

Different I/O Models

- Soar agent *always*
  - Receives input in the input phase
  - Generates output in the output phase
- Environments can vary
  - Asynchronous: Environment updates when each agent acts
    - Real world
    - Not all actors are Soar agents (or necessarily intelligent)
  - Synchronous: Environment updates after all agents act
    - Easier to debug
    - May be better for some research
    - Probably less interested in the environment / task and more in the agent

http://winter.eecs.umich.edu/soarwiki/Main_Page

ThreePenny Software 8

Output: Updating the World

- Option 1: Agent::Register(smlEVENT_AFTER_OUTPUT_PHASE)
  - Check for changes to the output link and change world
  - Good for asynchronous environment
  - Difficult for synchronous because agentA acts before agentB unless buffer actions in environment
  - Low performance – one event per agent per decision cycle. May not have acted.
- Option 2: Agent::AddOutputHandler(attribute, handler)
  - Called immediately after attribute is added to output link
  - Better performance than option 1 but must know attribute names
  - Similar strengths and weaknesses to option 1
- Option 3: Kernel::Register(smlEVENT_AFTER_ALL_OUTPUT_PHASES)
  - Called after all agents have completed output phase
  - Easier to produce synchronous interaction
  - Better performance – one event per execution cycle (for any number of agents)
- Option 4: Kernel::Register(smlEVENT_AFTER_ALL_GENERATED_OUTPUT)
  - Called after all agents have generated output
  - Turn based environments (e.g. Tank Soar)
  - Enforces completely synchronous behavior
  - An unusual choice

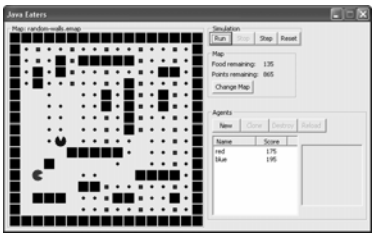
ThreePenny Software 9

Push vs Pull for Input

- Option 1: Push
  - When environment changes send new state to kernel
  - Ignores agent’s phases
  - Requires SML/gSKI to buffer until each agent’s next input phase
  - If environment changes faster than agent checks input, this option is lower performance
  - Implementation: Driven by output events (or external actors)
    - (Output) -> Change World -> Send Input
- Option 2: Pull
  - Agent calls over to environment each input phase to get current state
  - No buffering required
  - If environment changes slower than agent checks input, this option is lower performance
  - Implementation: Register(smlEVENT_BEFORE_INPUT_PHASE)
    - Send current state in event handler.
    - (Output) -> Change World. Don’t send new input.
- Soar 8.6.2 supports all of these different input/output options
  - Please consider your task in selecting implementation
  - Pretty easy to switch back and forth

ThreePenny Software 10

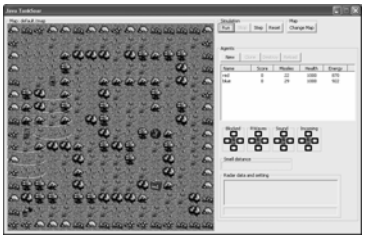
Java Eaters



- All new implementation in 8.6.2
- Higher performance.
- Output – smlEVENT_AFTER_ALL_OUTPUT_PHASES
- Input – push model (output -> update -> send input)
- Run – RunAllAgentsForever()

ThreePenny Software 11

Java Tank Soar

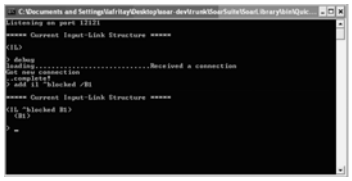


- All new implementation in 8.6.2. Shares some code with Java Eaters.
- Higher performance.
- Output – smlEVENT_AFTER_ALL_GENERATED_OUTPUT
- Input – push model (output -> update -> send input)
- Run – RunAllAgentsForever(sml_INTERLEAVE_UNTIL_OUTPUT)

ThreePenny Software 12

New Tool: Quick Link

- Manually control the input link
- "Fake" an environment
  - Test specific situations



- Examine current input and output links
- Add input wmes
- Modify or delete existing input wmes
- Run Soar
- Store and load scripts of commands

New Tool: Soar Text IO

- Easy way to place text (individual words) onto the input link in a standard way
  - Providing problem sets to an agent
  - Providing guidance or instruction

```
^input-link
   ^text
      ^text-input-number <num>
      ^length <num-words>
      ^next
         ^value <word-one>
         ^next
            ^value <word-two>
            ^next
               ^value <word-three>
               ^next
                  ^value nil
```

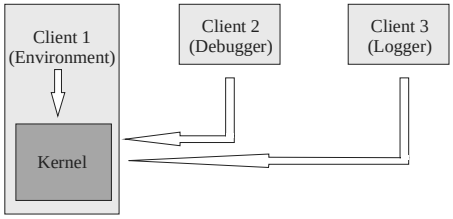

Better Logging

- How to log what Soar is doing?
  - Record trace as text file and parse it
  - Augment productions to output log information
  - Modify kernel to generate logging data
- Alternative is a logging application (client)
  - Connects to Soar while it's running (no overhead when not logging)
  - Register for events you are interested in
  - Output log information in any format desired
  - Examples in C++ and Java included in 8.6.2
- E.g. To create a behavior trace in your format

```
MyXMLEventHandler(ClientXML* pTraceXML) {

if (pTraceXML->IsTagState()) {
        std::string count = pTraceXML->GetDecisionCycleCount() ;
        std::string stateID = pTraceXML->GetStateID() ;
        std::string impasseObject = pTraceXML->GetImpasseObject() ;
        std::string impasseType = pTraceXML->GetImpasseType() ;

        // Write this out any way you want
        fprintf(gOutputFile, "%s %s (%s %s)\n", count.c_str(), stateID.c_str(), impasseObject.c_str(),
impasseType.c_str()) ;
}
```

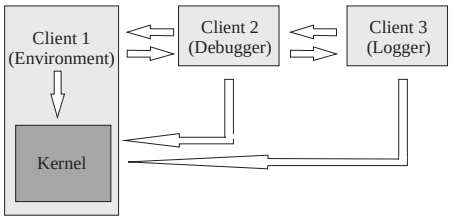

Client-to-Client Communication

- 8.6.1 – Sets of clients talking to kernel



Client-to-Client Communication

- 8.6.2 – Clients can talk to each other
  - Actually goes through kernel (synchs with kernel actions)
  - Kernel::RegisterForClientMessageEvent("debugger-status", handler)
  - Kernel::SendClientMessage("debugger-status", "ready") ;
  - Messages are strings; can be XML



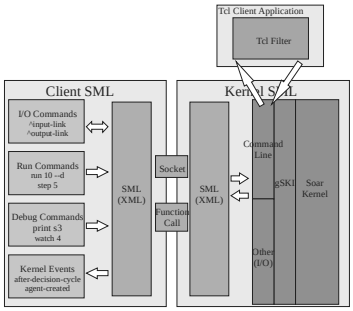
Better Tcl Support

- Problem
  - 8.5 Tcl was part of the kernel
  - 8.6 Tcl removed from kernel, available for environments
  - But in 8.5 could use Tcl to generate/expand productions:

```
proc NGS_prefer-operator-x-over-y { op1 op2 {prefix
   "operator-preference*"}} {
      sp "$prefix-$op1-over-$op2
         (state <s> ^operator <o1> + <o2> +)
         (<o1> ^name $op1)
         (<o2> ^name $op2)
        -->
         (<s> ^operator <o1> > <o2>)" }

NGS_prefer-operator-x-over-y generate-goal-subgoal-summary \
                             generate-goal-alternatives-summary
```

  - Soar Tech uses this extensively
  - How to support this in 8.6.2 when debugger in Java, kernel in C++ and Tcl not required?

Better Tcl Support



Better Tcl Support

- Filter main loop:

```
proc MyFilter {id userData agent filterName commandXML} {
    set interpreter [getInterpreter [$agent GetAgentName] $agent]

    set xml [ElementXML_ParseXMLFromString $commandXML]
    set commandLine [$xml GetAttribute $sml_Names_kFilterCommand]

    # Evaluates the command within the child interpreter for this agent
    set error [catch {$interpreter eval $commandLine} result]

    # Return the result of the command as the output string for the command
    # which in turn will appear in the debugger
    $xml AddAttribute $sml_Names_kFilterOutput "$result"
}
```

- Soar commands routed back from Tcl to Soar

```
proc add-wme {args} {return [soar_agent ExecuteCommandLine "add-wme $args"]}
proc excise {args} {return [soar_agent ExecuteCommandLine "excise $args"]}
```


Properties of Filter Solution

- Full Tcl interpreter(s) in own process
  - No limitations on Tcl code
- Works for all clients not just Java debugger
  - E.g. Can load “Tcl productions” from standalone environment
- No modifications to clients
  - Debugger is completely unaware filtering is happening
- No impact on people not using Tcl
  - Debugger doesn’t include Tcl interpreter support
- Performance should be good
  - Only affects time to parse commands
  - Once Soar is running filter is never called
- Filter is modular and separate from rest of code
- Filter can be in any supported language
- Other filters are possible
  - E.g. Just listen for “source x.soar” and run a precompiler
- But it’s not finished yet in 8.6.2

Commits are easier

- 8.6.1

```
Identifier* pSentence = pAgent->CreateIdWME(pAgent->GetInputLink(), "sentence") ;
pAgent->CreateStringWME(pSentence, "newest", "yes") ;
pAgent->CreateIntWME(pSentence, "num-words", 3) ;
pAgent->Commit() ;
```

- Commit()
  - Collects all input changes and sends them to kernel in one go
  - Higher performance but error prone
- 8.6.2

```
Identifier* pSentence = pAgent->CreateIdWME(pAgent->GetInputLink(), "sentence") ;
pAgent->CreateStringWME(pSentence, "newest", "yes") ;
pAgent->CreateIntWME(pSentence, "num-words", 3) ;
// Not needed: pAgent->Commit() ;
```

  - Kernel::SetAutoCommit(false) to revert to 8.6.1 behavior
  - AutoCommit on => slightly lower performance
  - But kernel often inside environment now so less impact
- Init-soar
  - Works (8.6.1 too) and resends input link to agents automatically

Wide Range of Events

- Run Events
  - [Before | After] Each Phase // As agent runs a phase
  - [Before | After] Decision Cycle // Completes decision cycle
  - [Before | After] Run Starts | Stops // Agent run / stop
  - [Before | After] Running // Each step of a run (decision, phase etc.)
  - Interrupt Check // Low bandwidth chance to stop
  - After Interrupt // Run was interrupted
- Update Events
  - After all output phases // Synchronous world
  - After all generated output // Turn based world
- Production Events
  - After Production Added // During loading
  - Before Production Removed // Excise
  - After Production Fired // Production firings
- System Events
  - After agent created // New agent created
  - Before agent destroyed // Agent about to be destroyed
  - [Before | After] agent reinitialized // init-soar
  - System Property changed // set [x] [y]
  - System Start | Stop // Agents are running / all stopped
- Trace Events
  - Print // Text output
  - Echo // Echo’d commands
  - Trace Output // XML output
  - Input Received // Listen for what environment is doing
- RHS and User Events
  - RHS Function handler // Implement RHS function in Java, C++, Tcl or C#
  - Command Line filter // Filter commands before kernel processes them
  - Client messages // From one client to another (not to/from kernel)
  - Edit production // Ask editor to locate production

TestSMLEvents & Wiki

Other 8.6.2 additions

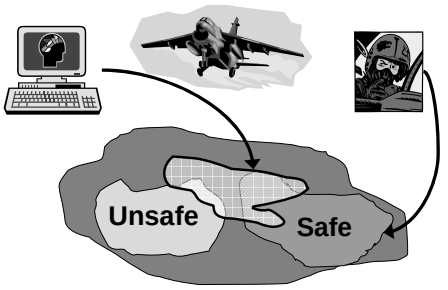
- Added Java TOH as an SML tutorial explaining all steps
- Added new phase specific events: before_input_phase etc.
- Added log output in the debugger on a window by window basis
- Added Visual Studio 2005 support (as well as VS 2003)
- Added Java 5.0 support (as well as Java 1.4.2)
- Added C# as a supported language
  - C++, Java, Tcl, C#
  - Identifier pInputLink = agent.GetInputLink() ;
- Added new random number generator and srand() command
- Improved Linux performance vastly (20x in some cases)
- Improved Windows performance further (> 30% faster in TOH)
- Loose coupling working
  - 8.6.2 debugger will load and run 8.6.1 kernel w/o modification
  - “New bits” (e.g. phase diagram) just do nothing
- Lots of bugs fixed
  - http://winter.eecs.umich.edu/soarwiki/Items_for_Consortium_Review

WASHINGTON STATE UNIVERSITY VANCOUVER

S-Assess:
Self-Assessment with Soar

Scott Wallace
WSU Vancouver

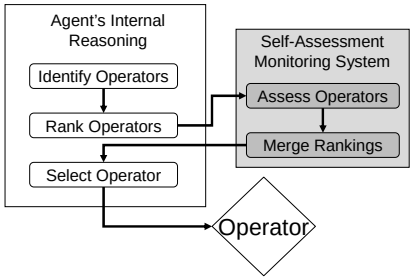
Building Human-Level Agents



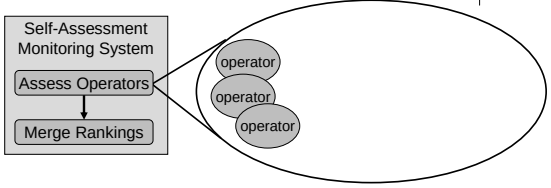
Detecting Errors in the Lab

- Abstract Behavior Representations
  - Concisely represent patterns in behavior
  - Provide a basis for automated comparison
  - Allow aggregate behavior to be examined efficiently
- Yet, in lab testing is *incomplete by nature*
  - Creates trust problems
  - Need to bring validation into the field

Self Assessment Framework

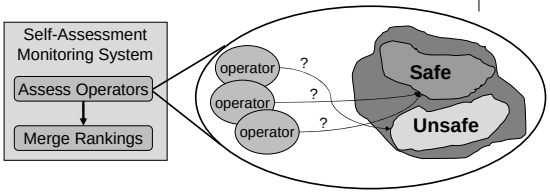


Assessment



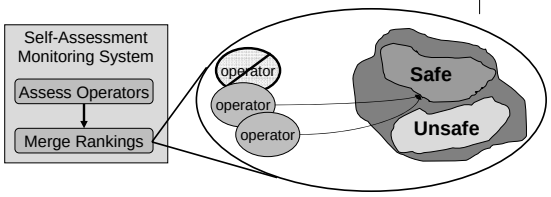
- The Agent is allowed to suggest a set of operators and their relative rankings

Assessment



- Framework evaluates potential operators by comparing against domain constraints

Assessment



- Unsafe operators are rejected
- Partial ordering remains otherwise intact

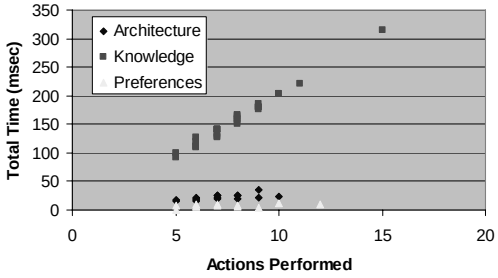
S-Assess via Knowledge

- Domain independent
- Set of 62 Soar rules
- Minor adjustments required to existing agents
  - Preferences encoded as operator augmentations
- Impasses trigger assessment operation
  - Assessment creates new symbolic preferences

S-Assess via Architecture

- Domain independent
- Architectural Modification
- Intercepts Preference Calculations
  - For each preference, S-Assess returns either:
    - ALLOW (preference consistent with rule book)
    - DENY (preference inconsistent with rule book)
  - Based on assessment result Soar's own preference calculations may be short circuited

S-Assess Overhead



Beyond Safety

- S-Assess acts as a high-level control
- Potential uses for such a control:
  - Safety/Correctness (specified by designers)
  - Environmental Policy (specified by environment)
  - Social Policy (specified by other agents)
  - As an exception mechanism
    - Adjustable Autonomy

S-Assess and Soar-RL

- S-Assess
  - Targets offline learning
  - Provides a mechanism for "universal preference computers"
- Soar-RL
  - Targets online learning
  - Provides a clean integration of reinforcement learning with Soar's knowledge representation

Nuggets & Coal

- Nuggets:
  - An interesting counterpart to Soar RL
  - Ready for real experimentation
    - Soar + Bayes Nets?
- Coal:
  - Some efficiency is sacrificed

Soar Technology
Thinking inside the box.

Relevance Estimation for Evidence Marshalling

23 May 2006

Robert E. Wray
Soar Technology, Inc.

Ron Chong
Cognitive Composites, LLC

Outline

- Goal:
  - Support evidence marshalling (EM) with automated reasoning
- Problem:
  - Infinite search spaces
  - Difficult to know if/when progress is being made
    - One step from "Eureka!"
- Proposed Solution: Relevance Estimation
  - Algorithms that guide EM searches and get better with experience
- Methods:
  - Test bed for evaluating relevance estimation and evidence marshalling
- Warning:
  - Not a lot of Soar in this story; mostly laying groundwork…

Soar Technology

Evidence Marshalling

- Process of "connecting the dots" to form a coherent narrative about a collection of evidences
  - Pop-culture Examples:
    - Sherlock Holmes
    - Procedural TV shows (Perry Mason, Law & Order)
  - Specialized human techniques for EM
    - Wigmorean analysis
- Many potential applications for automated methods:
  - Police investigations
  - Insurance fraud
  - Intelligence analysis

Soar Technology

Automating EM

- Automated reasoning techniques have been demonstrated as effective tools for EM in very restricted problem domains
  - cause-of-death conclusions from evidence
  - insurance fraud detection
- Limits to automated EM in open-ended domains
  - Hypothesis generation
  - Knowledge sources
  - Efficient control of search process

Soar Technology

Automating EM: Near-term solutions

- Hypothesis generation:
  - In the near- to mid-term, no automated approach is likely to challenge human creativity
  - Our approach leverages user insight and experience by capturing their notions and "hunches" and then searching for confirming or disconfirming evidence
- Sources of knowledge
  - Trifles
    - Many existing automated sources: data extraction tools
  - Domain knowledge
    - Facts and inferences from applicable domain(s)
    - Increasing stores of general knowledge resources: Cyc, Semantic Web
- Efficient search
  - Must find intermediate hypotheses/assertions that can connect trifles to user hypotheses

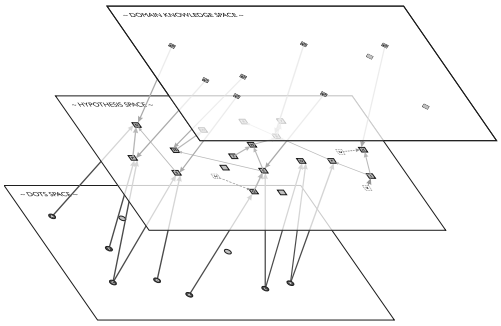
Soar Technology

Challenge: Search Space is Infinitely Large

- Domain knowledge and trifles combine in infinitely many possible ways
  - Bill Clinton was POTUS in 1996
  - Hillary Clinton is Bill Clinton's spouse
  - …
- Difficult to judge if any individual intermediate hypothesis is getting closer to the goal
  - Hillary Clinton lived in the White House in 1996
- Multiple, simultaneous searches
  - Trifles: Which "trifles" are relevant
  - Domain knowledge: Which facts about the world are relevant?
  - Hypotheses: Which intermediate hypotheses are on the right track?
  - Confidence: Which assertions are more likely/probable?

Soar Technology

Simultaneous searches



Hypothesis:
Efficient search for EM is a solvable challenge

- Evidence:
  - Although the search space is infinitely large, people perform evidence marshalling adequately
  - How?
    - Training & experience, tools (Wigmorean analysis)
    - "Natural" pruning of the search space
      - Humans generate heuristic choices ("hunches") that can guide them thru very large search spaces
- Goal: explore computational methods for:
  - Generating and assessing "hunches" (relevance estimation)
  - Learning from experience to improve the relevance estimation process

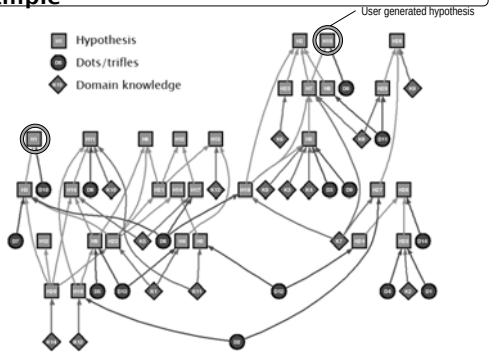
Proposal: Relevance Estimation

- Goal: Evaluate the relevance/importance of each new assertion
  - Cannot assume system already "knows" the answer or knows that some new inference is an important "dot"
    - Many potential assertions can be generated
    - May not be obvious which assertions will actually be valuable or "relevant" to the problem (Avoid "wishful thinking")
  - Relevance estimation is the "strategy" for prioritizing direction(s) of reasoning and establishing "sufficiency" of a potential direction of reasoning
  - Context is important: Not evaluating in isolation
- Contributing methods:
  - Domain neutral: analogy, deduction, classification, etc.
  - Domain specific: social network analysis, model of user's interest
- Metrics for relevance determination:
  - complexity-based (Occam's razor)
  - graph-based (path length)
  - knowledge-based
    - information scent
    - ontological relationships

Test bed for RE/EM Research

- Problem:
  - Need large bodies of encoded knowledge to evaluate approaches to RE/EM
  - Access to "real" data is difficult for research purposes
    - security, sensitivity, size, etc.
- Proposed Approach:
  - Test bed for RE/EM
    - Automatically generate potential search spaces
    - Capture size and complexity of the space
    - Systematically vary parameters that represent the "connectedness" of possible hypotheses in the example
    - Simulate relevance across a number of orthogonal and over-lapping dimensions

Example



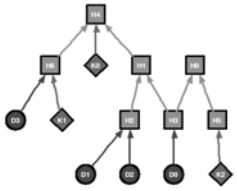
Simulating relevance in the test bed

- Relevance:
  - Predicts the likelihood of a relationship between m and n
  - Estimated by aggregating multiple knowledge sources
  - RE $= f(re_0, re_1, re_2, \ldots re_n)$, where re_i is a source of knowledge about the relationship between m and n
- Contributing knowledge sources, re_i, can include:
  - ontological relationships between m and n
  - semantic and syntactic similarity between m and n
  - completeness of analogical mapping between m and n
  - "cognitive" distance between m and n
  - expectation of success of getting to m and n
  - recognition of m in the context of n

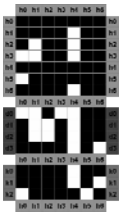
Simulating relevance in the test bed, cont'd

- Some sources can be simulated using graph theoretic operations, e.g., transitive closure in graph implies perfect knowledge of relations between dots, domain facts, and hypotheses.

pEM graph



TC matrix

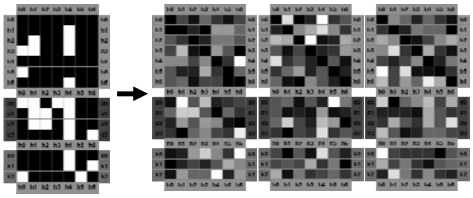


Soar Technology

Simulating relevance in the test bed, cont'd

- Real knowledge is imperfect and from multiple, potentially contradicting, sources
- In the testbed, the perfect TC relationships are distributed (with noise) to multiple re_i matrices

TC matrix re_0 re_1 re_2

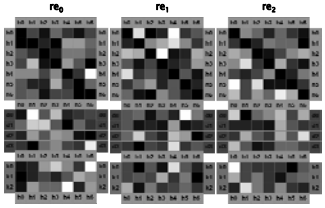


Soar Technology

Simulating relevance in the test bed, cont'd

- Test bed facilitates:
  - Easy use of knowledge sources represented as a matrices
  - Explorations of how to aggregate noisy and contradictory knowledge

re_0 re_1 re_2



Aggregation options:

$RE(m,n) = \Sigma c_j * re_i(m,n)$

$RE(m,n) = \max(re_i(m,n))$

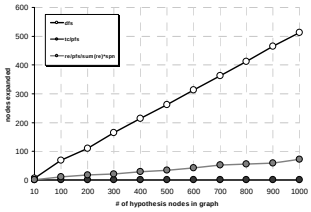
$RE(m,n) = \text{avg}(re_i(m,n))$

$RE(m,n) = \ldots$

Soar Technology

Initial Results (1)

- Search for all nodes in pEM graphs
  - DFS: exhaustive knowledge-free search; upper-bound
  - TC: perfect knowledge give direct path to goal; lower bound
  - RE: RE(m,n) guides priority-first search;
    - 90% fewer expanded nodes in comparison to DFS



Soar Technology

Conclusions

- Humans provide evidence tractable search for EM is feasible and clues for how to do it
- Test beds can be developed for exploring search challenge without large, a priori investments in knowledge encoding
  - Proof-of-concept test bed can generate problems automatically for exploring the consequences of RE design options
  - Plan: Complement with open-source, real-world datasets
- Nuggets:
  - Interesting ideas about how to attack this problem
  - Developed simple testbed for exploring computational approaches
  - Rich literature from which to draw inspiration
- Coal:
  - No feasibility implementation/exploration of solution ideas

Soar Technology

Thinking…

…*inside* the box

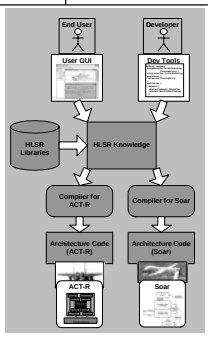
HLSR: Compiling to Soar

Randolph M. Jones
Jacob A. Crossman
Christian Lebiere
Bradley J. Best

Soar Technology
Thinking *inside* the box.
Micro Analysis & Design

What is HLSR?

- High Level Symbolic Representation
- A language for encoding knowledge
- The language is:
  - Architecture-neutral
  - Domain-independent
  - High-level
  - Designed to support reuse
- Target users:
  - Cognitive modelers
  - End user tool developers



What HLSR Contributes

- Design at the representation level, hide implementation details
  - Free modeler from architecture-level details
  - Emphasize understandability, maintainability, and reuse
- Why an abstract language?
  - Better tools are necessary but not sufficient
  - Cognitive architectures are necessary but not sufficient
- A language allows merging of different architectural concepts while abstracting low-level details

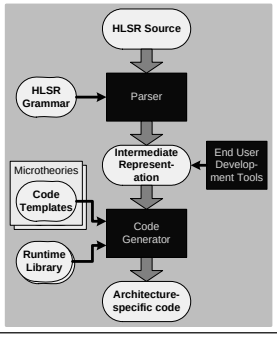
Common Mechanisms Captured by HLSR

- Goals
- Declarative memory
  - Structure and retrieval
- Timely reaction to external events
- Decision processes
  - Goal selection
  - Action selection
- HLSR creates higher-level constructs that map onto different lower-level constructs in different cognitive architectures
  - Compiler should be able to translate HLSR to ACT-R or Soar

Micro-Theories

- Description of structures, templates, and execution strategies used to execute HLSR constructs
  - Architecture-specific
  - Invisible to HLSR developer
- Micro-theories are modular
  - One micro-theory for each HLSR construct

The HLSR Compiler



Abstracting Low-Level Details

- Process tagging
- Integrating knowledge from different models
- Computing answers vs. retrieving stored answers
- Iteration
- Copying
- Complex logic
- Representing sensory-motor interactions

HLSR Building Blocks (Primitive Constructs)

- **Relations**
  - Declarative memory, goals
  - Form: production or rule
- **Transforms**
  - Procedural knowledge
  - Form: body of execution
- **Activation Tables**
  - Pattern recognition for response selection
  - Form: decision matrix

Relation

- A relationship between symbols in declarative memory
- Defined by:
  - Name
  - Attributes
  - Met condition (optional)
- Can be:
  - A fact
  - A goal
  - A request to retrieve something from declarative memory

```
relation Square
   name is a string
   size is a integer

relation SmallerThan
   a is a Square
   b is a Square
   met condition
      a.size < b.size
```


Transform

- A conditionally executed procedure
- Defined by:
  - Name
  - Trigger conditions
  - Body (set of actions)
- Actions execute serially
- Multiple transforms may execute in parallel
- Failure to execute → transform suspended and subgoal created

```
transform MoveSquareLeft
   a is a square
   consider if
      goal is to change location
      best place is to the left
   body
      pick up square
      move left
      put down square
```


Activation Table

- Specifies conditions and actions
  - Like truth tables or production rules
- Defined by:
  - Condition block
  - Action block

    Actions are labeled:
    - T (true)
    - F (false)
    - * (don't care)

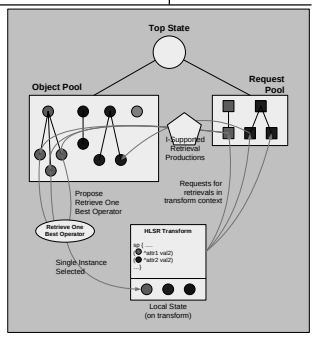
```
activation table WeatherGear
conditions
1: It is raining
2: It is windy

TT  Wear raincoat, no umbrella
TF  Wear raincoat, bring umbrella
F*  No raincoat, no umbrella
```


Compiling Relations to Soar

- **Key Requirements**
  - Blend asserted facts with computed facts (retrieve v. compute problem)
  - Map to HLSR global memory pool (no state references)
  - Retrieve one best (eliminate multiple retrievals)
- **Constraints**
  - Partial matches can cause significant slow down
- **Observation**: compiler lacks some of the semantic information humans use to do this efficiently
  - Cardinality constraints
  - Data lifetime: how long data is valid

A Soar Microtheory for Relations

- Objects placed in pools based on type
- Retrievals on demand
  - Transform assert requests in pool
  - I-supported productions assert value based on met condition
- Operator used to select one best
- Directly asserted and retrieved facts represented in object pool the same way



Compiling Goals to Soar

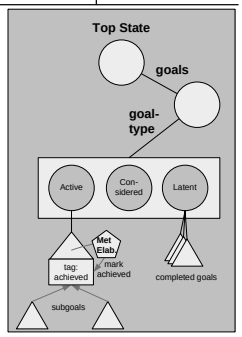
- **Key Requirements**
  - Represent goal forest
  - Auto-reconsideration via met condition
- **Constraints**
  - Soar "state stack" can only represent single thread of goals
- **Observation**: we have to decide how to leverage goal stack

Our Current Approach

- Use FOG approach similar to "Radical Randy" OR
- Use FOG declarative representation but use state stack to have single thread of **active** goals

A Soar Microtheory for Goals

- Goals pooled in similar way to objects
  - Extra layer for active state of goal
  - Active goal pool should be small for performance
- "Met" condition elaborations mark goal achieved
- Operator used to move goal to the "Latent" bin after achievement

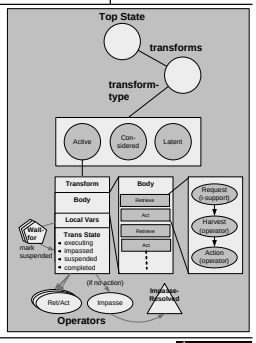


Compiling Transforms to Soar

- **Key Requirements**
  - Hold consistent variable bindings
  - Execute sequences including waitfor statements
  - Provide an automatic subgoal mechanism for transform failures
- **Constraints**
  - Soar is inherently parallel
- **Observation**: with transforms the compiler does most bookkeeping that developers usually do
  - Process tags and temporary variables
  - Sequence tags and conditions

A Soar Microtheory for Goals

- Transform objects store state and tags for multiple operators
  - Could be a Soar state
- Code generation decomposes body to retrieve/act pairs
- Each retrieve/act pair executed sequentially with tags used to control sequence
- Waitfors using i-support
- Impasses generated when no operator (i.e. state no change)



Compiling ACT-R

- Challenges
  - ACT-R much more sequential: few, narrowly defined points of parallelism
  - ACT-R has no support for predicate logic
  - ACT-R can be non-deterministic: what is the acceptable number of times it should get the "right? answer?
- ACT-R microtheories
  - Map complex retrievals to low level retrieval sequences
  - Leverage the goal (or context) buffer to represent processing state for all HLSR constructs
  - Provide less parallelism: generally the ACT-R program has to decide explicitly *when* to check conditions (e.g. met conditions, activation table conditions, etc)

Conclusions

- Status of the HLSR project
  - Initial implementation nearly complete
  - Evaluation on the way
  - Building abstractions and micro-theories has revealed interesting and subtle differences between architectures
- HLSR will:
  - **Abstract** away from details of a particular cognitive architecture
  - **Encapsulate** knowledge and behaviors
  - Improve **efficiency** of creating new models
  - Allow easier **comparisons** of models and architectures
  - Make cognitive modeling more accessible

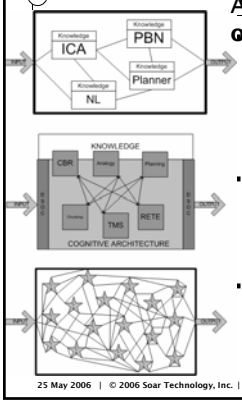
Soar Technology
Thinking inside the box.

Modularity in Soar-based Applications: Practical Issues

25 May 2006

Robert E. Wray
www.soartech.com

AI system engineering options

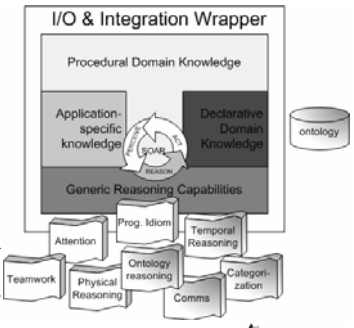


- Software Engineering (GOFAI):
  - Modular decomposition derived from functional decomposition
  - Fixed, engineered interaction & control
  - Knowledge matched to module
  - Black-box module operation
  - Includes "Engineered MAS" approaches
- Cognitive architecture approach
  - Uniformly encoded knowledge
  - White-box knowledge modules
  - Least-commitment control and knowledge integration
- DAI/Multiagent System approach
  - Opportunistic, unscripted interaction
  - Distributed ("no executive") control
  - System behavior is "emergent"

Soar Technology

Typical Soar-based Application

- Agent maps to human actor in a physical environment
- Agent should exhibit capabilities roughly comparable to the human agent in the environment
- Typical HBR/CM implementation is consistent with Soar theory:
  - Soar is the sole "intelligence" platform
  - All knowledge is dynamically integrated at run-time within Soar
  - Examples:
    - NASA-TD, TAS, RWA (STEAM), AMBR (SCA), etc.



Soar Technology

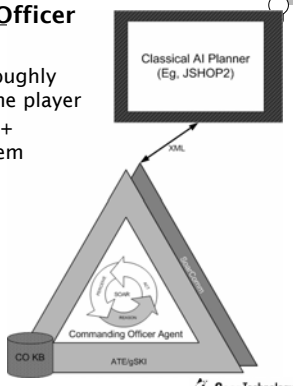
Recent Soar Technology Application Architectures

- Current application development at Soar Tech demonstrates strong reaction to practical constraints of Soar
  - Result: System architectures beginning to look more like GOFAI systems than systems constrained by Soar theory

Soar Technology

JFETS Commanding Officer

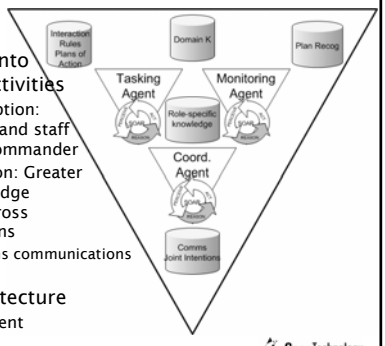
- "Command" function roughly comparable to RTS game player
- Realized as Soar agent + separate planning system



Soar Technology

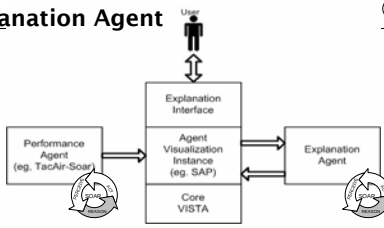
Commanding Officer Decomposition

- Decomposes command role into three distinct activities
  - Original assumption: model of command staff vs. individual commander
  - Partial motivation: Greater reuse of knowledge components across different domains
    - joint-intentions communications knowledge
- Extensible architecture
  - visualization agent



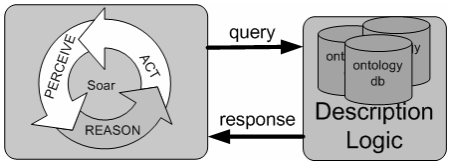
Soar Technology

VISTA Explanation Agent



- Generation of explanations realized in a separate agent
- Architecture facilitates explanations from non-Soar agents

Onto2Soar (proposed 2nd-generation system arch)



- Move formal ontologies outside representation within Soar and use description logic tools to resolve ontology queries

Practical Issues in Soar Application Development

What is driving divergence from theoretical assumptions?

- Soar theory:
  - Completeness in functionality
  - Knowledge reuse across applications (especially knowledge for general capabilities)
  - Architectural assumptions enable run-time knowledge integration (interleaving and open, not encapsulated, knowledge dependencies)
- Application-development constraints:
  - Current release of Soar is not complete (research in progress)
  - Special-purpose mechanisms (e.g., planners) offer significant performance improvements
  - Knowledge reuse is the exception, not the rule
  - Knowledge development cost tends to scale super-linearly
  - Large knowledge bases do not necessarily provide adequate performance
  - No knowledge packaging methodology/tools
  - Soar is used for many non-HBR/CM applications
    - Are "Soar claims" specific only to human-inspired models, or to intelligent systems generally?

Directions for application-development tools

- Research in knowledge packaging
  - Initial explorations/lessons: SCA (impasses), STEAM (annotations)
  - High utility for both traditional and GOFAI approaches
  - Preserves/enables "white box" modularity?
- Research/engineer "semantic interfaces"
  - Define good abstractions for Soar-Module information exchange
  - Example: Generic Soar-planner interface
- Research/engineer enabling technology for interfaces
  - Blackboards
    - Agent memory as blackboard (ala JESS)
    - Distinct shared memory component
      - Soar agent as blackboard (ala AIS)
  - Communication infrastructure
  - Understand and document trade offs!

Conclusions

- Nugget
  - Soar theory offers compelling story for least-commitment control and dynamic knowledge integration
- Coal
  - Many practical limitations impede realization of theoretical benefits
- Unresolved questions
  - Which system engineering approaches are most appropriate for what kinds of applications?
  - What are "natural" units of agency for different kinds of applications?
  - Are Soar constraints in non-HBR systems useful?
  - What research and tools are needed to support the different directions for supporting application development?
  - How could Soar best be applied in DAI systems (if at all)?

Soar Technology
Thinking inside the box.

SoarML: A Graphical Modeling Language for Agents

Glenn Taylor, Jacob Crossman
glenn@soartech.com, jcrossman@soartech.com

25 May 2006

www.soartech.com

What is SoarML?

- A **visual language** for representing single agent designs
  - Based roughly on the Prometheus agent design methodology
  - Customized for human behavior modeling
- Initially developed as part of HLSR effort
  - Iteratively improved over two years
- Allows **code/architecture independent** descriptions of an agent's behavior
  - In general, is NOT specific to Soar
  - Was used initially to document HLSR designs
- Used for a several Soar Technology systems
  - Adversarial reasoning module
  - Indirect Fire (IF)-Soar
  - Deontics additions to Command and Control
  - AutoATC

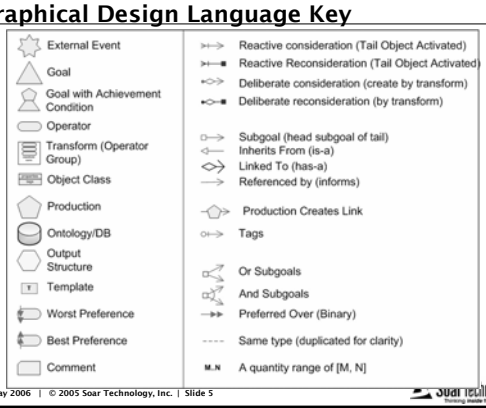
Motivation for a Modeling Language for Agents

- **Promotes High-level design**
  - Almost always better to think through design before coding
  - Text and code are not always best ways to encode designs
  - A modeling provides constructs that map to design concepts and ignore low-level details
- **Communication to Management**
  - PI needs a way to express/understand what is going on in an agent without looking at code
  - Customers sometimes need design documentation or key algorithms/processes explained
- **Communication within a development team**
  - Understanding what is happening in a Soar program is hard
  - Understanding is easier when the high-level concept is clear before looking at the code

Another Modeling Language?

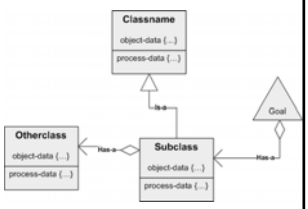
- **Others exist: Why invent our own?**
  - Existing methods: OO UML, AUML, Prometheus
  - **But:**
  - Most agent MLs focus on multi-agent aspects, little detail at the individual agent level
  - None capture *cognitive architecture* aspects (goals, truth maintenance, deliberate consideration, preferences, etc.)
  - In many cases UML is helpful to cover other areas; SoarML focus is on areas UML doesn't cover well for agents

Graphical Design Language Key

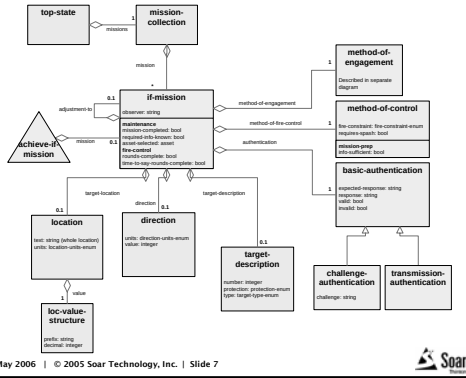


Static Structure Diagram Examples

- Description
  - Representation of declarative memory's structure
  - Consistent with OO UML specifications
  - Tagging separates process-centric data (usually only shown in process diagrams)
- Notes
  - Can be used standalone or as part of process diagrams
  - Soar doesn't really directly support structures or inheritance

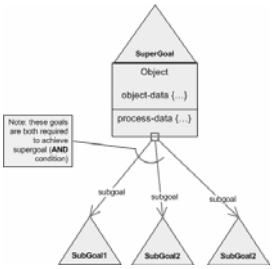


Knowledge Structure for Indirect Fire

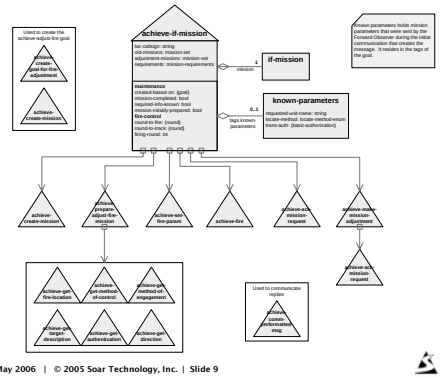


Goals and Goal Hierarchies

- Description
  - Represent goal hierarchies
  - Supports forests or stacks
  - Does not require any specific Soar implementation (e.g. using impasses or top-state goals)
  - Can augment with "met" condition: a production that marks the goal "achieved"
- Notes
  - Can be used to show goal forest or connection to goals in process diagrams

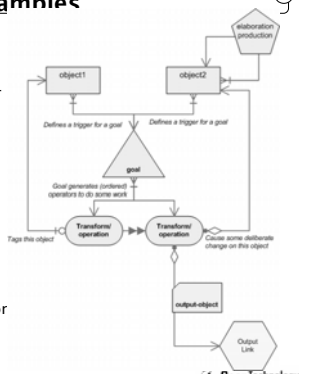


Goal Forest for Indirect Fire

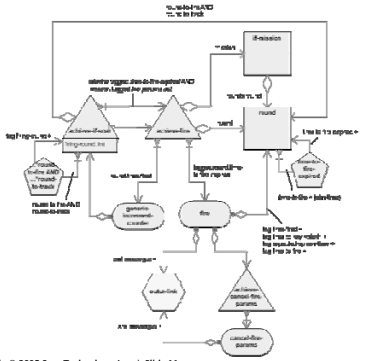


Process Diagram Examples

- Description
  - Represent processes: sets of related operators in Soar
  - Integrates static structure, goals, operators, and preferences
  - Key productions can be highlighted
  - Includes key memory changes and trigger conditions
- Notes
  - Typically processed are documented to the operator level



The Firing Process for Indirect Fire



When/How to Use

- **Documenting Design Concepts**
  - Purpose:
    - construct and analyze the framework for an agent's behavior
    - understand major behavior interactions and knowledge structures
  - Guidelines
    - Most effective early in a project
    - Focus on major objects, processes, and relationships
    - Keep abstract: implementation will refine and suggest changes
- **Document Existing Systems**
  - Purpose:
    - Provide an overview of system for maintenance team
    - Provide customer/management with technical details
  - Guidelines
    - Most effective late in development cycle after details solidify
    - Focus on key patterns of behavior and concepts to understand how the agent behaves and how it can be modified
    - Drill down to moderate levels of detail (e.g. provide more firing conditions and knowledge structure details)

Nuggets/Coal

Nuggets

- Useful for design documentation and presentations
- Being used on several projects
- A good way to visually inspect design for flaws/commonalities

Coal

- Hard to get some engineers to design and document
- Only a few people using it regularly
- Doesn't address multi-agent processes (other MLs might cover this sufficiently)

Soar Technology

References

- For Visio Stencil, email Glenn or Jacob
- Prometheus

Padgham, L. and Winikoff, M., Prometheus: A Methodology for Developing Intelligent Agents, Proceedings of the Third International Workshop on AgentOriented Software Engineering, at AAMAS 2002. July, 2002, Bologna, Italy

http://www.cs.rmit.edu.au/agents/prometheus/

Soar Technology

Soar Technology
Thinking *inside* the box.

Questions?

Pyramid Problems in Soar & ACT-R

John Laird
26th Soar Workshop

Big Picture Goals

- Take instruction (not using NL)
  - Task instructions
  - Problem structure
- Execute task using domain-independent interpretation
  - No task-specific knowledge in rules
  - Except basic mathematics (7+6 = 13)
  - A few bits of special knowledge for meta-reasoning
- Match human data and compare to ACT-R
  - Chunking?
- Meta-Cognition

Mastering an Algebraic Concept

Pyramids:
There is a notation for writing repeated addition where each term added is one less than the previous:

For instance, 5 + 4 + 3 is written as 5 $ 2

Since 5 + 4 + 3 = 12 we would evaluate 5$2 as 12 and write 5$2 = 12

The parts of 5 $ 2 are given names:

5 is the base and reflects the number you start with
2 is the height and reflects the number of items you add to the base
5 $ 2 is called a pyramid

Instructions

Soar:

```
(<s1> ^action <a10> <a11> <a12>
      ^next <s2>)
(<a10> ^command set ^variable sum
       ^value 0 ^value-type constant)
(<a11> ^command set ^variable term
       ^value base ^value-type variable)
(<a12> ^command set ^variable count
       ^value 0 ^value-type constant)
(<s2> ^action <a30>
      ^next <s3>)
(<a30> ^command add ^variable sum
       ^value term ^value-type variable)
(<s3> ^action <a6>
      ^next <s4>)
(<a6> ^command goal-test ^relation equal
      ^variable count
      ^value height ^value-type variable
      ^type finished)
(<s4> ^action <a4> <a5>
      ^next <s2>)
(<a4> ^command decrement ^variable term)
(<a5> ^command increment ^variable count)
```

English

1. Set sum to 0
   Set term to base
   Set count to 0
2. Add term to sum
3. Test if count = height
4. Decrement Term
   Decrement Count
   Goto 2

Problem Structure and Example Problem

```
(<ps1> ^name base ^type variable ^next <ps2>)
(<ps2> ^name |$| ^type symbol ^next <ps3>)
(<ps3> ^name height ^type variable  ^next <ps4>)
(<ps4> ^name |=| ^type symbol ^next <ps5>)
(<ps5> ^name answer ^type variable ^next nil)
```

```
(<p1> ^value 5 ^type constant ^next <p2>)
(<p2> ^value |$| ^type symbol ^next <p3>)
(<p3> ^value 3 ^type constant ^next <p4>)
(<p4> ^value |=| ^type symbol ^next <p5>)
(<p5> ^value |?| ^type unknown ^next nil)
```

Basic Flow

- Initialize-instruction
- Initialize-problem
- Encode [Map problem onto problem structure]
  - Process-symbol, Process-variable, Process-unknown
- Execute-solve-procedure [Interpret procedure to solve problem]
  - Execute-steps
    - Set, Add, Subtract, Increment, Decrement, Goal-test
  - Next-step
- Write-answer [Write out the answer]
- Reflect - [Looks for patterns in problems]
  - Detect first-term - height = last-term
  - Detect balanced problems around 0
- Next-problem

Evaluation Problems

1. 5 $ 3
   5 + 4 + 3 + 2 = 14
2. 10 $ 4
   10 + 9 + 8 + 7 + 6 = 40
3. 8 $ 1
   8 + 7 = 15
4. 3 $ 4
   3 + 2 + 1 + 0 + -1 = 5
5. 5 $ 7
   5 + 4 + 3 + 2 + 1 + 0 + -1 + -2 = 12
6. 0 $ 4
   0 + -1 + -2 + -3 + -4 = -10
7. 13 $ 0
   13
8. 1000 $ 2000
   1000 + … + 1 + 0 + -1 + … + -1000 = 0
   2000

Expression Writing Problems

9. 6 + 5 + 4 + 3
   6$3
10. 9 + 8 + 7
    9$2
11. 1 + 0 + (-1) + (-2)
    1$3
12. x + (x – 1) + (x – 2) + (x – 3) + (x – 4)
    x$4
13. 20 + (20 – 1) + …. + (20 – 11)
    20$11
14. 15 + (15 – 1) + …. + (15 – x)
    15$x
15. z + (z – 1) + …. + (z – y)
    z$y

Find the Height Problems

16. 6 $ x =15
    6 + 5 + 4 = 15 --> x = 2
17. 10 $ x =55
    10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 = 55 --> x = 10
18. 912 $ x =912
    x = 0
19. 3 $ x = -9
    3 + 2 + 1 + 0 + -1 + -2 + -3 + -4 + -5 = -9 --> x = 8
20. 100 $ x = -101
    100 + … + 1 + 0 + -1 + …-100 + -101 = -101 --> x = 201
    201

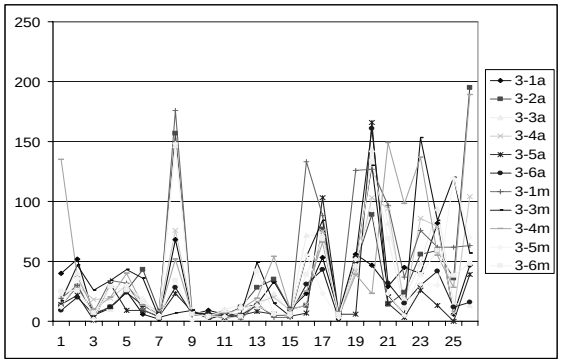
Find the Base Problems

21. x $ 2 =15
    guess and check: 7+6+5 = 18; 6+5+4 = 15 or
    x + (x - 1) + (x -2) = 15 --> 3x - 3 = 15 --> x = 6
22. x $ 1 = 15
    x = 8
23. x $ 4 = 35
    x = 9
24. x $ 6 = 35
    x = 8
25. x $ 6 = 0
    x = 3
26. x $ 6 = -7
    x = 2

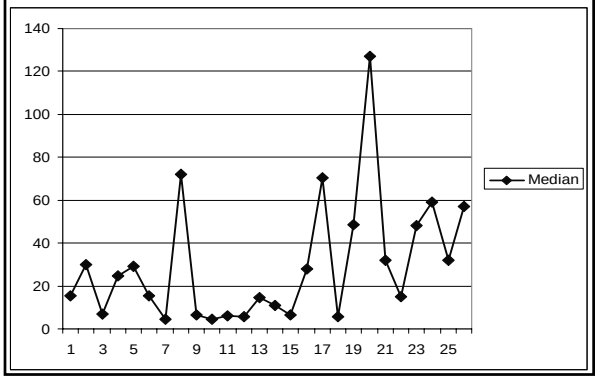
Soar Approach to Problem Types

- Solve: 5$3 =
  - Uses execution procedure
- Describe: 6+5+4
  - Uses describe procedure (what ACT-R does too)
- Solve: 6$x=15
  - Uses execution procedure - stops when answer achieved: Learned stop by doing first set of problems
- Solve: X$2=15
  - Impasses on setting Base = X
    - Generate and tests values of X and then solves
      - Must create hypothetical problems
    - If fails, then must generate a new guess
    - Smart generator (based on prior problem, prior guesses)

Individual Human Data



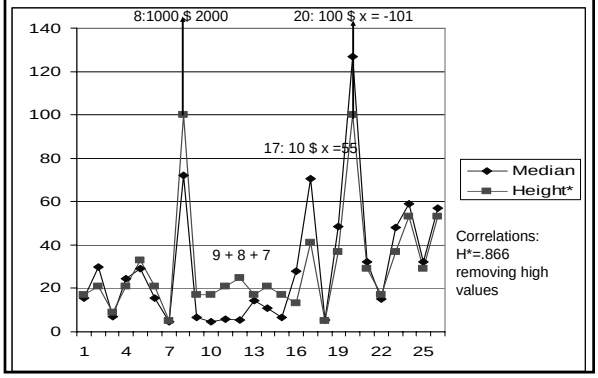
Median Human Data



Simple Model: Height*

- Time is proportional to Height
  - Base $ Height = ?
  - This is clearly the most important part of the procedure
- Extend to take into account finding base problem
  - X $ 2 = 15
  - Simple model of guessing X, modifying guess if wrong.

Median, Height*



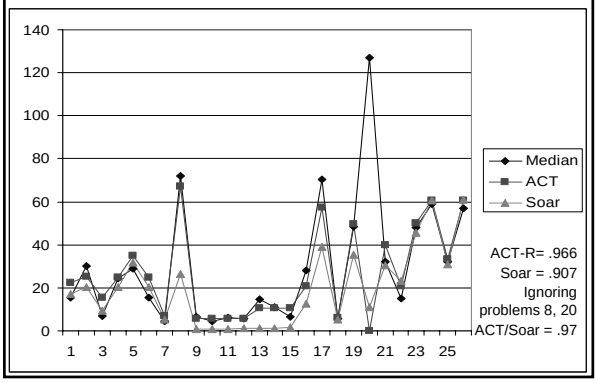
Comments on 1000$2000: John Anderson

1. Students averaged about half of their time in unproductive attempts before they tried a method that work.
2. An unproductive path tried by many was to find an analogy to what they knew about factorial.
3. Five students reasoned about simpler problems like 2$4.
4. Others reasoned more abstractly.
5. A number of students confirmed the answer (0) by a second method before giving it as their final answer.
6. The final ACT-R model tried factorial, then abstract reasoning, and finally confirmed by solving 2$4.
7. Two significant issues for modeling are interrupting regular processing and accumulating needed knowledge.
8. Both are metacognitive in that they require parallel reflection on the ongoing problem solving

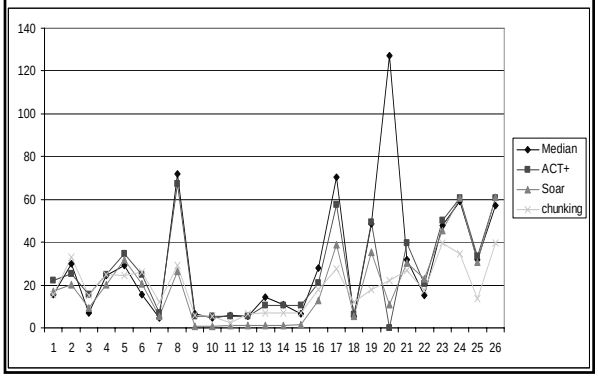
Soar Approach to 1000$2000

- *Detects "large" height*
- Attempts "abstract" solution
  - What can it compute?
  - First-term: 1000, Last-term: -1000 (derived from observed relation)
  - *Notice "balanced": 1000, -1000 => 0*
- Create simple problem to check
  - *Creates 2$4=*
  - Solve simple problem => 0
- Assumes that is the answer
- Special prior knowledge:
  - *Detect large height*
  - *Note balanced*
  - *Simple problem generator*
- Soar doesn't mess around with factorial, etc. like ACT model and humans do but clearly could.

Median, ACT, Soar (scaled)

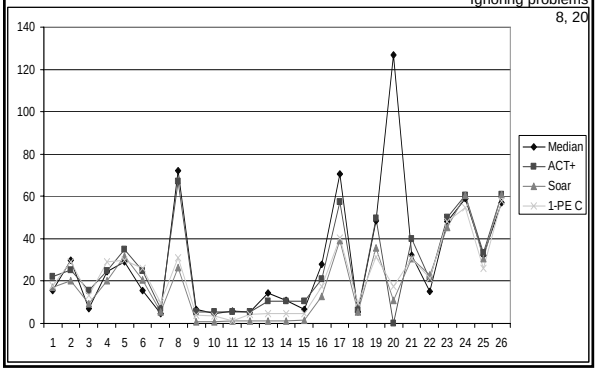


Median, ACT+, Soar, Soar w/ Chunking

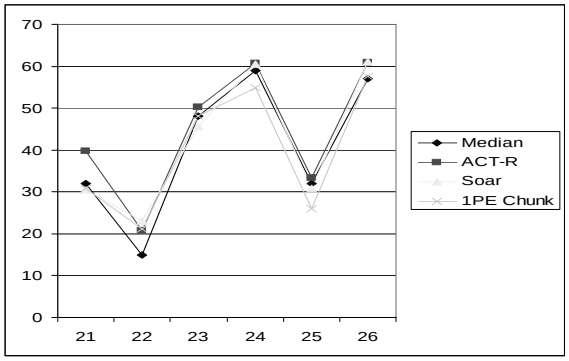


Chunking with 1PE/Decision

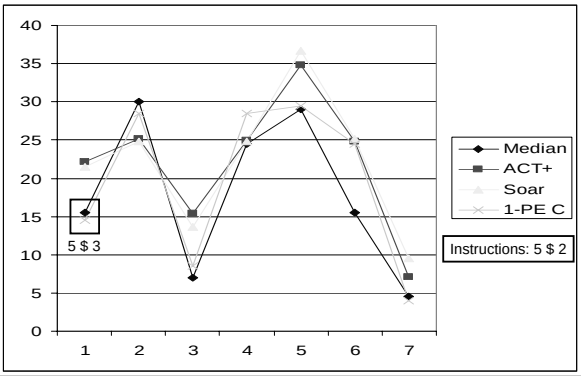
ACT-R= .966
Soar = .907
1PE = .906
Ignoring problems
8, 20



Last 6 problems



First 7 problems



Conclusions

- Nuggets:
  - Can do instruction taking (again)
  - Leads to surprisingly good results
    - It is (almost) all about doing the task (following instructions)
    - Results hold up with chunking 1PE/Decision
  - Soar is natural for metacognition
    - Impasses
    - Creating test problems in subgoals
    - Reasoning about structures complex structures (variable attributes)
- Coal:
  - More work to do on detailed comparison with ACT-R
  - More work on where some extra knowledge comes from
  - Soar model is scaled
    - Not 50msec/decision
    - No model of perception, …

Control and Metacognition: A View from EPIC, Soar, and ACT-R

Richard L. Lewis

Department of Psychology
University of Michigan

May 22, 2006

Overview

1. Goals
2. Analysis and critique of current architectures
   - EPIC
   - ACT-R
3. A Three Component Framework for Control
4. Specific proposal for architectural changes
   - Architectural changes
   - Evaluation domains
   - Nuggets de Oro

Present Goals

Two main goals:

1. Develop a (generative) **framework for control** that will lead to better understand of existing architectures and point to promising new points in the architectural space
   - Not review evidence from psychology, neuroscience
   - Rather provide the computational backdrop against which to interpret such evidence
2. Understand the **architectural requirements for metacognition**

EPIC Basics: What makes EPIC distinctive

1. **Maximum strategic flexibility** in control
   - All control functionality (with exception of synchronous clocking) off-loaded to knowledge
   - Via *task-specific* productions
   - Via *general* productions
2. **Unlimited cognitive parallelism** (directly supports multiple parallel threads of control)
3. Performance is limited by **P/M subsystems, learning**, and **similarity-based interference**
   - P/M limits are familiar
   - SBI in working memory limits performance in novel task composition
   - May be theoretical limits on learning *general executive* for multi-tasking for anything beyond about two independent tasks (Kieras, 2006)

ACT-R basics: What makes ACT-R distinctive

1. **Serial production rule firing**
2. **Conflict resolution (choice mechanism) determined by learned utility estimates**
   - $PG - C$ (*note:* utility estimation being changed to a reinforcement-learning scheme)
   - Choice is function of the rule itself, *not* the rule + context
     - See, for example, Marsha Lovett's stick-building task model—she was forced to split up two simple strategies ("under-shoot" and "over-shoot") into multiple productions in order to learn the appropriate contextual utilities
   - Falsifying this seems like a tractable empirical problem in reinforcement learning

Architectural Support for Choice

	ACT-R	Soar	EPIC
Choice mechanism	Mellow do-er	Semi-worried thinker	None (clever do-er)
Knowledge source	Statistical, with limited context	EBL/analytic, and statistical with arbitrary context	Dave Kieras
Available information about the choice	None (or result)	Proposals, preferences, and impasses	None (or result)

Key differences

1. Differences in **default speed-accuracy tradeoff**
   - Soar *impasses* (and takes no action) at the first sign of trouble
   - If ACT-R *can* do something, it *will*
2. Differences in **automatic detection and representation of meta-information about the choice**
   - Soar can detect and represent (architecturally) not just *response conflict* but *cognitive conflict*, and more generally, lack of knowledge
   - ACT-R and EPIC are blissfully oblivious
   - Functional implication of this is a difference in the support for deliberation and knowledge composition in *novel* contexts[1]

[1] Though nontrivial to convincingly demonstrate this—perhaps a *Control Pentathlon* is needed

7 / 18

A Three Component Framework for Control

Productions
Choice mechanisms
Monitors
(Controlled subsystems)

Productions provide:
- Computational completeness
- Arbitrary, fine-grained contingencies for behavior

Choice mechanisms provide:
- Locus of learning of control; support for deliberation

Monitors provide
- Immediate representation of information about *internal processing state* in a form that control can be made contingent upon
- Integration over different time scales, modules

Controlled subsystems provide
- Internal agent resources (LTM, WM, perception, motor, etc.)
- "Control knobs" (commands, inhibition, etc.)

8 / 18

Examples of existing (and proposed) monitors

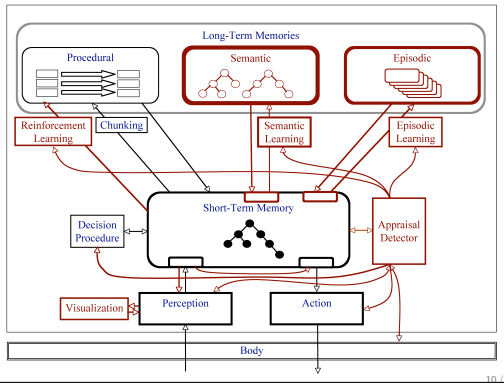
ACT-R
- Motor module state flags
- Retrieval module state flags (failure)
- Temporal module (?)

EPIC
- P/M module state flags

Soar
- Impasse detection (state no-change, tie)
- Appraisal system for emotion

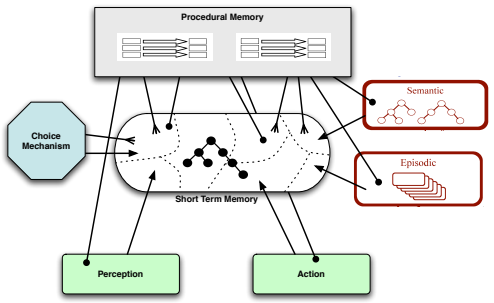
9 / 18

The Old New Soar



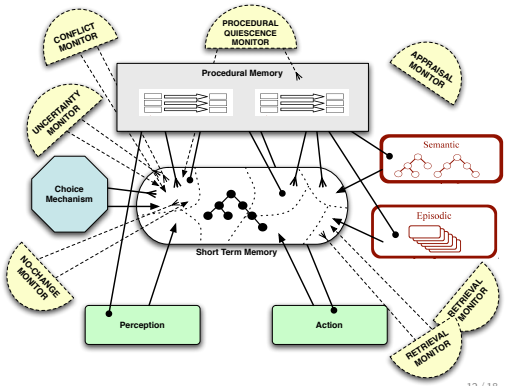
10 / 18

The New New Soar



11 / 18

The New New Soar



12 / 18

Specific proposal: Part 1

Monitors Should Be First-class Architectural Objects

- Could be implemented as separate module/buffer
- Should not be strictly associated with single module; could in principle monitor multiple modules

Specific proposal: Part 2

Choice Mechanisms Should Be First-class Architectural Objects

- Decision procedure should be independent of quiescence detection and impasse detection
- Should be possible to create multiple (asynchronous) decision/control streams

Specific proposal: Part 3

Add/Reimplement and Explore the Following Monitors

1. **Cognitive Slack Monitor** (aka "State No Change Impasse Detector")
   - Input from *procedural* and *temporal modules*
   - Output to buffer a representation of time passed since last production fire; analogous to failure in retrieval buffer
   - *(Perhaps a thalamic-frontal circuit under current mapping)*
2. **Utility Uncertainty Monitor** (aka "Tie Impasse Detector")
   - Input from *procedural module*
   - Output to buffer representation of how "close" the race is
   - Perhaps modulated by confidence (see Belavkin & Ritter (2004) proposal)
   - And perhaps the procedural system itself could be modulated in continuous way to achieve various SAT's (purely under knowledge-driven control)
   - *(Perhaps a caudate/palladal-frontal circuit under current mapping)*
3. **Retrieval Uncertainty Monitor**
4. **Appraisal Monitor** (Emotion)

Proposed evaluation domains

Appropriate evaluation domains include:

1. **Meta-cognitive judgments**
   - Including the large and body of empirical work on memory retrieval and meta-cognitive judgements modeled by SAC and related models
2. Paradigms investigating **speed-accuracy tradeoffs** at multiple time scales
   - At level of seconds–10s of seconds–minutes
   - At level of hundreds of milliseconds—including the very large and detailed body of empirical work on choice-reactions modeled by mathematical choice models such as the diffusion model, and SAT deadline paradigms

Possible Golden Nuggets

1. Provide new class of key functional modules to **guide mapping from functional architecture to brain**
2. New Soar or ACT-R could **detect (the degree to which) its current knowledge is insufficient**, even more flexibly than Soar can now
   - But could (based on the immediate speed-accuracy trade-off considerations) choose to use (or ignore) that information as it sees fit
3. The resulting architecture should be **more flexible and "temporally situated" than either current Soar or ACT-R**, and **better able to deal with novel contexts than EPIC**

Possible Coal

- Soar and ACT-R lose their distinctiveness and relative competitive theoretical advantages
- ???
- (It's easy to have a tiny haul of coal when no systems have been built. . .)

Cognitive Constraint Modeling: An Alternative to Traditional Architectures

Andrew Howes[1] Richard Lewis[2] Alonso Vera[3]

[1]School of Informatics
University of Manchester

[2]Department of Psychology
University of Michigan

[3]HCI Group
NASA Ames and Carnegie Mellon

May 23, 2006

Outline

Key Claims

1. Human Task Performance can be predicted by **formally reasoning about the implications of a theory** rather than running a simulation.
2. A cognitive architecture theory **explains** empirically observed asymptotic bounds on adaptation of performance if there is **substantial correspondence between the asymptote and the optimal performance implied by the theory.**
3. The ability to automatically derive optimal predictions from cognitive theory has **significant theoretical and applied benefits**.

How Architectures Make Predictions

ARCHITECTURE + KNOWLEDGE (STRATEGY) = BEHAVIOR

A Conundrum for Cognitive Theory

Complete cognitive theories must take the form architectures that admit of arbitrary knowledge/strategic variation

BUT: knowledge, strategy can become theoretical degrees of freedom in modeling data

- Explanation may reside primarily in strategy, not architecture
- And strategy may have been selected to fit the data at hand
- *(But that never happens, right?)*

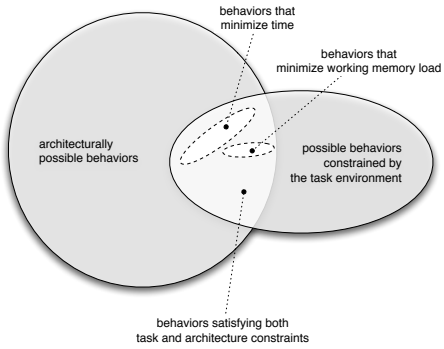
Two Possible Solutions

1. Focus on **"immediate behavior"** (Newell 1990)
   - Behavior < 1 s
   - Problem: Even < 1 s behavior shows surprising amount of strategic modulation (Meyer & Kieras, 1997)
2. Theory of **learning/instruction taking**
   - "Close the loop", so strategy not under theorist's control
   - Problem: complexity; testing many aspects of theory simultaneously

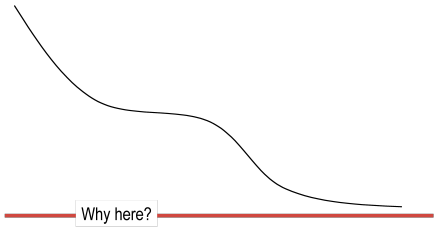
Constraint Analysis Overview

- Adaptive behavior is bounded by **Objective + Environment + Knowledge + Architecture** (Simon 1992)
- **Constraint satisfaction techniques can be used to calculate the optimal behavior given a set of heterogeneous constraints**
- In short, combining **Formal Rational Analysis** with **Bounded Rationality**

Constraint Analysis Overview

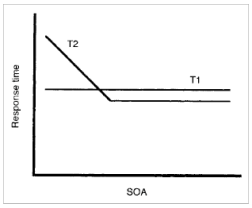


Explaining the Bounds on Adaptation



Typical PRP (psychological refractory period) Experiment

- Choice response to a tone (T1) and a pattern (T2).
- Give priority to the tone response.
- Tone presented first, pattern stimulus is presented after an SOA.
- According to Meyer and Kieras, elevated RT2 is because participants ensure T2 response is after T1 response.



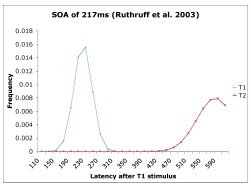
Simple Dual-Tasking PRP Study

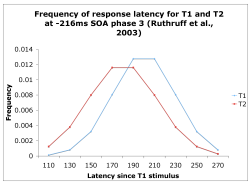
Ruthruff et al., 2003 report a PRP experiment with:

- Single participant.
- Unordered responses.

Now imagine if subject must produce **ordered** responses:

- At long SOAs no SRD is required to avoid response reversal.
- At short SOAs more than 50% response reversal when objective not sensitive to reversal.



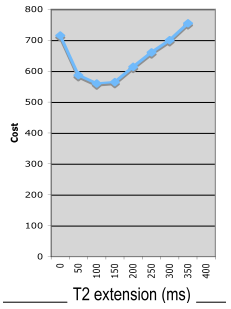


A Very Simple Constraint Model

- Constraints consist of the mean overall RTs and SDs.
- Space of strategies defined by a single variable: Extension of T2 response (E).
  - Cf. Meyer and Kieras' SRD (Strategic Response Deferment).
- Objective is to minimize total duration and total response reversals.
  - Note the **trade-off:** Reduced reversals vs. total duration.

Combining Task + Architecture to Compute Optimal Behavior

- Now we can compute cost function from Monte-carlo simulations given this subject's standard deviation of RTs.
- Note that this combines two features:
  1. Constraints on the **TASK** (ordering and speed constraints, as expressed through explicit payoff).
  2. Constraints on the **ARCHITECTURE** (noise in the performance system).



Simple Ordered Responses

New experiments (Kopecky):

1. Visual cue appears.
2. Subject must quickly press two keys in order:
   - Left index, right middle.
   - Left middle, left index.
   - Right ring, left middle.
   - etc.
3. Subject rewarded for speed and accuracy.

SIMPLE MODEL

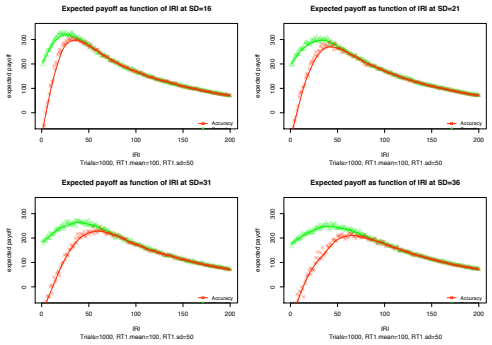
- Subjects defer R2 for *IRI* milliseconds after R1, where IRI maximizes payoff given their indiosyncractic variance;

Explicit Payoff Schemes

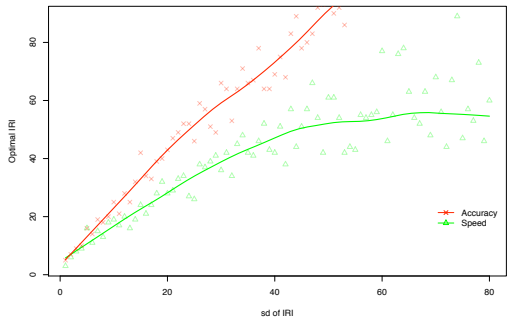
Subjects were rewarded with CASH with explicit payoff schemes. Example:

- If correct and Total RT $< 500ms$, then award $100 - RT/5$ points.
- If correct and Total RT $>= 500ms$, then award zero points.
- If incorrect, then lose 100 points.

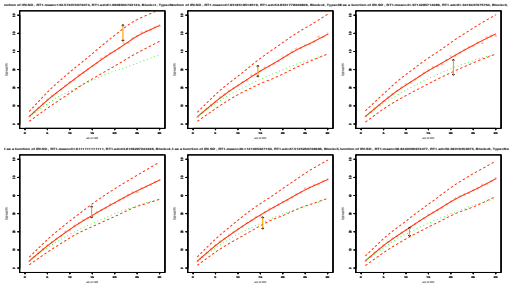
Sample Payoff Curves at 4 Different Standard Deviations (SD) of IRI



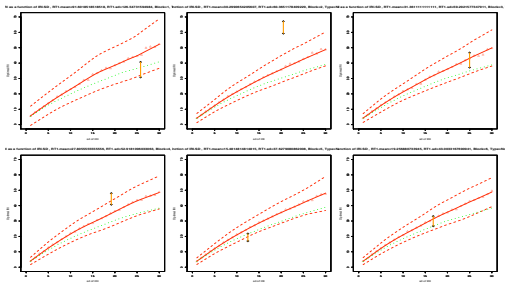
Predicted optimal IRI as function of SD of IRI



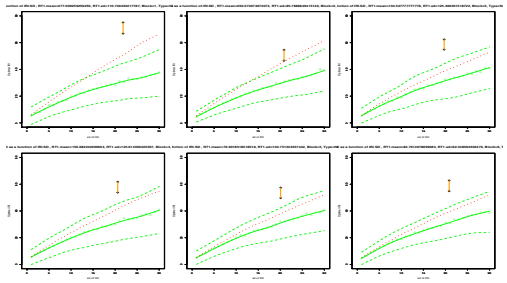
A Good Subject



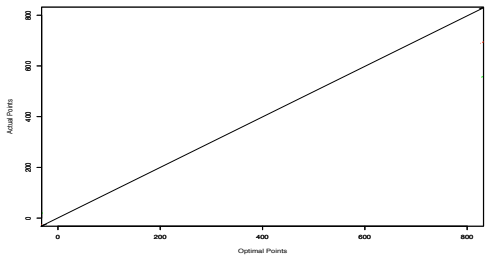
Another Good Subject



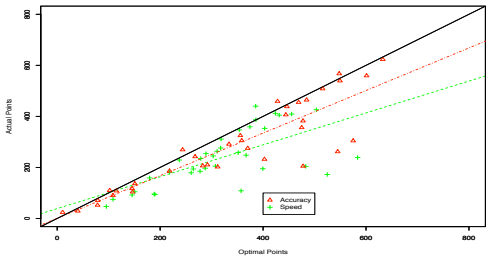
A So-So Subject



All Subjects, all Finger-Pairs: Actual vs. Predicted Optimal Points

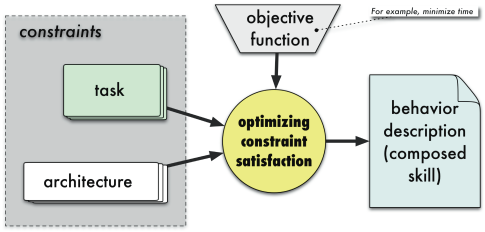


All Subjects, all Finger-Pairs: Actual vs. Predicted Optimal Points



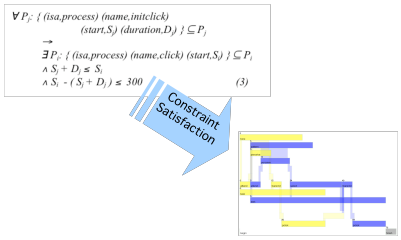
But analysis of more sophisticated strategies needs a general purpose solution. . .

Cognitive Constraint Modeling



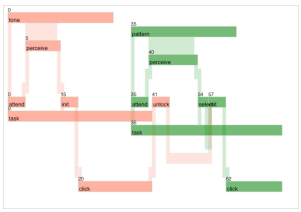
Cognitive Constraint Modeling

The tool is called CORE: Constraint-based Optimizing Reasoning Engine



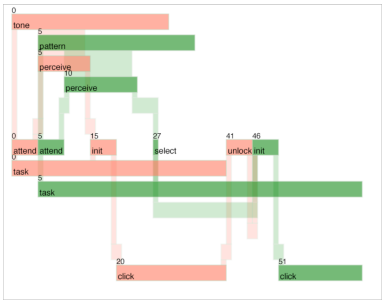
Constraints are logical relations between variables. They may specify partial values (e.g., duration, $D_i > 24$ ms), are non-directional (E.g. $S_j <= E_i + 300$ ms), and declarative.

Behavior Graphs



- Boxes represent **cascaded processes**.
- Rows of processes represent **resources** (cognition, perceptual, motor).
- Time is represented from left to right.
- Horizontal position represents onset.
- Spatial extent represents duration.

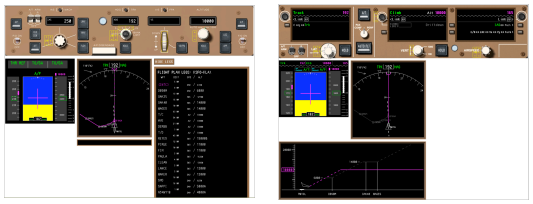
Dual-task (PRP) with simple set of process & information-flow constraints (50ms SOA)



But does the approach scale to more complex tasks?

Comparing two cockpit designs

777 **FDF**



Goals:

- New design should reduce errors
- New design should be no slower than old

777 Interface: Task 1

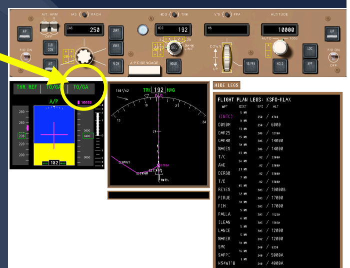
"You are following the altitude restrictions of the Moorpark 3 arrival; your last altitude clearance was 1-7 thousand. Descend via the Moorpark 3 arrival; maintain 1-2 thousand"

1. Verify current vertical mode
2. Dial Altitude Selector down to 12,000
3. Hit Altitude Selector
4. Verify new altitude
5. Verify new vertical mode

777 Interface: Task 1

"You are following the altitude restrictions of the Moorpark 3 arrival; your last altitude clearance was 1-7 thousand. Descend via the Moorpark 3 arrival; maintain 1-2 thousand"

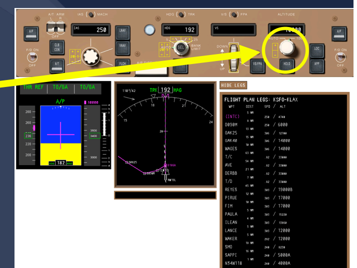
1. Verify current vertical mode
2. Dial Altitude Selector down to 12,000
3. Hit Altitude Selector
4. Verify new altitude
5. Verify new vertical mode



777 Interface: Task 1

"You are following the altitude restrictions of the Moorpark 3 arrival; your last altitude clearance was 1-7 thousand. Descend via the Moorpark 3 arrival; maintain 1-2 thousand"

1. Verify current vertical mode
2. Dial Altitude Selector down to 12,000
3. Hit Altitude Selector
4. Verify new altitude
5. Verify new vertical mode



777 Interface: Task 1

"You are following the altitude restrictions of the Moorpark 3 arrival; your last altitude clearance was 1-7 thousand. Descend via the Moorpark 3 arrival; maintain 1-2 thousand"

1. Verify current vertical mode
2. Dial Altitude Selector down to 12,000
3. Hit Altitude Selector
4. Verify new altitude
5. Verify new vertical mode

777 Interface: Task 1

"You are following the altitude restrictions of the Moorpark 3 arrival; your last altitude clearance was 1-7 thousand. Descend via the Moorpark 3 arrival; maintain 1-2 thousand"

1. Verify current vertical mode
2. Dial Altitude Selector down to 12,000
3. Hit Altitude Selector
4. Verify new altitude
5. Verify new vertical mode

777 Interface: Task 1

"You are following the altitude restrictions of the Moorpark 3 arrival; your last altitude clearance was 1-7 thousand. Descend via the Moorpark 3 arrival; maintain 1-2 thousand"

1. Verify current vertical mode
2. Dial Altitude Selector down to 12,000
3. Hit Altitude Selector
4. Verify new altitude
5. Verify new vertical mode

The Demands of Applied Modeling

- Not just interested in *time*, but **memory load** and **ability to handle interruption**
- Tracking memory load requires specifying what must be held in memory and when
- Our task specification language and models capture this in the form of **information flow constraints**. . .

36 / 56

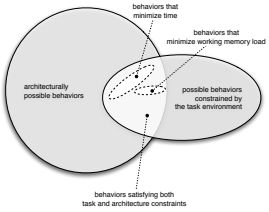
A Natural Task Specification

Captures the task's **information flow** rather than a fixed sequence of steps.

boeing FDF tt1
–>
comprehend situation : *FLIGHT_PLAN LAST_CLEARANCE*,
comprehend clearance : *INSTRUCTION* ***ALTITUDE,***
get vertical_mode after INSTRUCTION ALTITUDE : *VMODE*,
set altitude to **ALTITUDE** given INSTRUCTION VMODE : *DIALED PUSHED*,
check limit against ALTITUDE after DIALED PUSHED : *LMT_CHECKED*,
check ap_status against INSTRUCTION after LMT_CHECKED : *AP_CHECKED*.

Emergent Strategies

- Fully specified task constraints may still leave many details of behavior unspecified
- These details are automatically worked out by CORE to satisfy the architectural constraints
- *Example:* Precise timing of the perception of the mode information

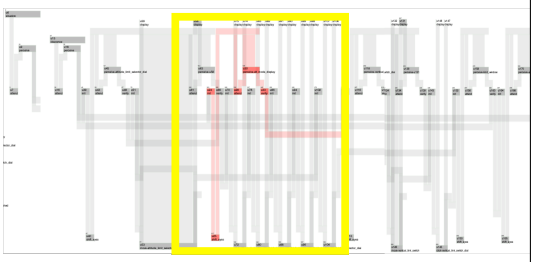


Emergent Strategies

Early look to the mode display, in series with the rest of the task:

Emergent Strategies

Later look to the mode display, in parallel with dialing the altitude:



A PRP Emergency!!

WHAT IF . . . the pilot had to handle an auditory interruption that required a manual button press response?

boeing FDF tt1
→
comprehend situation
comprehend clearance
get vertical_mode
set altitude
check limit
check ap_status

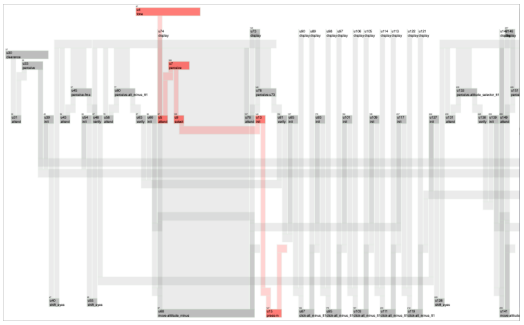
auditory interruption
→
auditory tone,
attend auditory
perceive auditory tone,
choose_response
press key.

A PRP Emergency!!

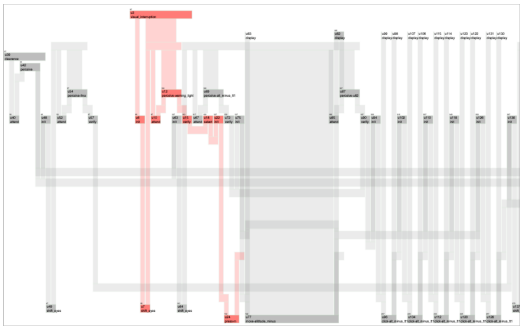
WHAT IF . . . the pilot had to handle an auditory interruption that required a manual button press response?

task
→
auditory interruption,
boeing FDF tt1.

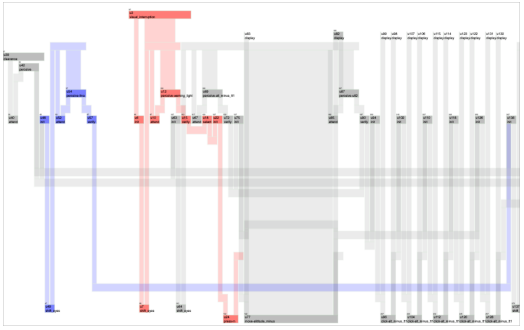
Auditory Interruption



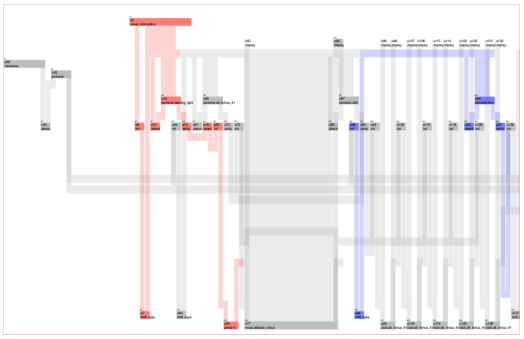
Visual Interruption



Visual Interruption



Same Task Spec, Different Objective: Reduce Memory Load



24 Models

2 interfaces {FDF, 777} × 2 tasks × 3 interruption conditions × 2 optimizations {time, WM}

Interesting predictions:

1. FDF faster than 777
2. Little difference in WM load
3. Simple auditory interruption slightly increases time and WM load
4. Simple visual interruption increases time more, and effect is greater for 777

But is this just a different way to do architectural modeling, or does it really change the way we should build and test cognitive models?

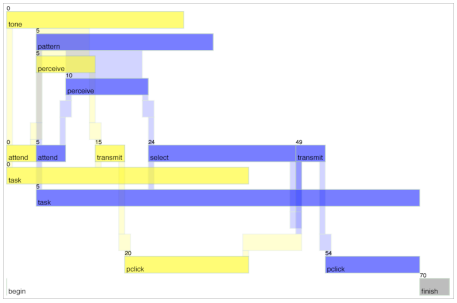
ACT-R vs. EPIC, in Psych Review (2001)

- In ACT-R, retrieval time from memory is sensitive to a limit on total source activation.
  - The more features associated with the goal, the more the source activation is spread and the less each to-be-retrieved chunk receives.
- In the Byrne & Anderson 2001 ACT-R model, this created a *dual-task interference effect* because the source activation was less when tasks overlapped because the goal contained features from both tasks.

We can model this as a constraint.

An ACT-R model of the PRP task

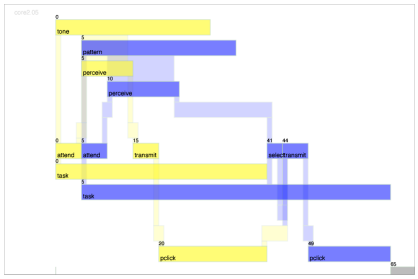
Our reconstruction of one of the models in Byrne & Anderson (2001), *Psychological Review*:



What did the original ACT-R model explain?

- Byrne and Anderson created a ACT-R model (based on a particular strategy) that fit the data
- **But if a better strategy is available, given ACT-R's constraints, has skilled PRP performance been explained?**

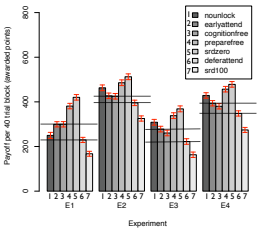
ACT-R model (optimal)



The optimal model not only deferred response, **but deferred retrieval too.** Byrne and Anderson didn't think of this—and neither did we.

An Astonishing Result

- Using CORE, we performed a systematic analysis of possible strategies for ACT-R models on all four PRP experiments modeled in Bryne & Anderson (2001), computing the expected payoff based on 40,000 runs.
- **In each experiment, the Byrne & Anderson models consistently underperform—sometimes by substantial amounts—the best strategy.**



Nuggets de Carbón

1. **Efficiency.** Some models take 2 seconds, some take 24 hours, some never return.
2. **Interaction with task simulation.** Presently, can't be done.
3. **Difficulty formalizing learning constraints.** Presently, can't be done (though we haven't really tried).

Summary/Nuggets de Oro

1. Adaptation is bounded by the task environment and architecture.
2. An architecural theory explains behavior, with no further assumptions, if the optimal performance predicted by the theory corresponds to the observed asymptotic bound.
3. **Constraint satisfaction can be used to predict the asymptotic bound on adaptation, formally deriving the predictions of an architectural theory while minimizing assumptions about strategy.**
4. **Significant theoretical and applied benefits may accrue from this approach and its associated tools.**

Acknowledgements

People:

- Alina Chu (Michigan)
- Katherine Eng (NASA Ames)
- Jonathon Kopecky (Michigan)
- Mason Smith (Michigan)
- Irene Tolinger (NASA Ames)

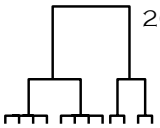
Agencies/companies:

- Office of Naval Research
- NASA (Ames Research Center)
- Boeing

BICA: Biologically-Inspired Cognitive Architecture

JOHN LAIRD

26TH SOAR WORKSHOP



1

Major New DARPA Project

- Goal: Capture the Magic of the Human Mind
  - Look beyond traditional AI
  - Computation and Psychology: Cognitive Architecture
  - Biology: Brain Computation
- Programmatics
  - Phase I: 13 months, started in October 2005
    - A. Cognitive Architectures (4)
    - B. Biologically-based Computation (8)
    - C. Evaluation (1)
    - D. Basic Research (3-4)
  - Phase II: 4-5 years, starting in ~March 2007
    - 2-3 (Big) Groups

2

Evaluation (Yearly)

- Cognitive Decathlon (details still up in the air)
  - Psychology Experiments Across the Frontiers of Cognition
    - Functional Integration, Flexibility, Metacognition, Social Embeddedness
  - Embedded within same environment as challenge problems
- Challenge Problems
  - Simulated robot embedded complex environment
  - Find IEDs in a (hot, dry, dusty) urban environment
- Emphasizing
  - Learning, learning, learning,
  - Vision, vision, vision,
  - Language, language

3

Four Possible Approaches

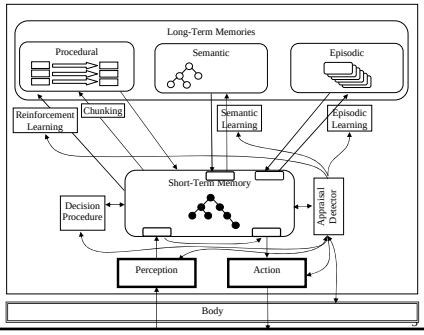
1. Cognitive architectures with
   a. added abstract biologically-inspired functions
   b. added perceptual-motor inputs and outputs
   c. biologically-inspired module implementation
2. Biological architectures, cognitively constrained

4

1a. CA with added abstract biologically-inspired functions

- Abstract functionality e.g., reinforcement learning, emotion
- But no direct connection to underlying computational process

Cognitive Architecture
+
Episodic, Semantic, Reinforcement Learning
+
Emotion



5

The Future for Soar-style Cognitive Architectures

- Generate coherent, purposeful behavior across a wide variety of tasks
- Use many sources and types of knowledge
  - Perception, personal history, facts, abstractions, skills, models, …
  - Gained by experience, observation, communication, and programming
  - Are immediately taskable
- Combine knowledge in novel ways to generate novel behavior
  - Not a slave to experience or current situation
  - Can use abstraction, hypothetical situations, internal simulation
- In real time using conventional computational hardware

But unable to achieve *wild learning:*

- ubiquitous automatic learning of unexpected types of regularities
- in noisy, feature-rich environments
- that combine with previously learned concepts and relations
- that give rise to hierarchies of new symbols, concepts, and relations
- that lead to prediction, anticipation, comprehension, …

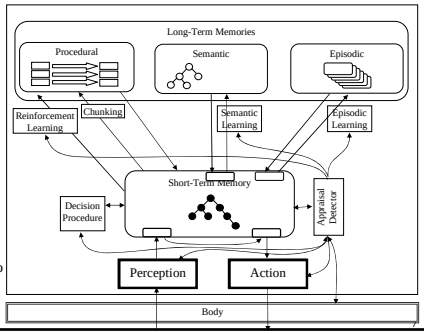
6

1b. CA with added perceptual-motor I/O

- Brain-based computation for perception & motor control
- Cognitive architecture remains essentially unchanged

Cognitive Architecture + Biologically Based Sensing and Action

Transduction of continuous features to symbol



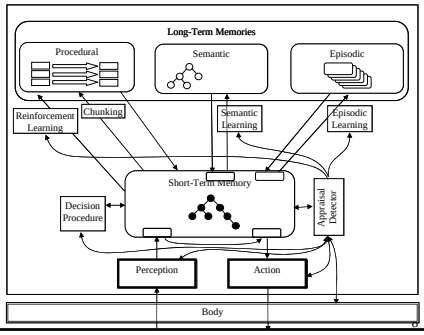
7

1c. CA with biologically-inspired module implementation

- Brain-based computation for existing CA components
- Retains overall structure of cognitive architecture

Cognitive Architecture Replace with Biologically Based Components

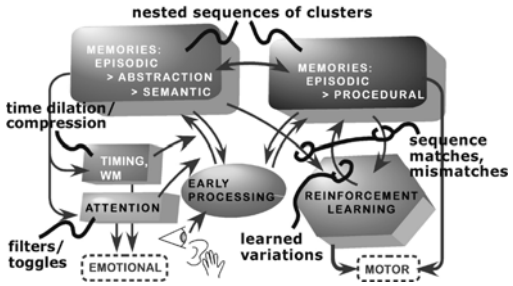
Component interaction: Symbolic & non-symbolic



8

2. Biological architecture, cognitively constrained

- New architecture; biologically derived
- Cognitive functions emerge from component interactions



9

Nuggets and Coal

- Nuggets:
  - *Chance of a lifetime* to explore integration of brain-based computation and cognitive architecture.
- Coal
  - Incredible amount of work
    - Synthesizing different computational approaches
    - Managing large group across many institutions
    - Developing/Engineering tasks and agents

10

Principles of Brain Computation

26th Soar Workshop
May 26, 2006

Rick Granger
University of California, Irvine

Clusters, Symbols and Cortical Topography

Lee Newman
Thad Polk

Dept. of Psychology
Dept. Electrical Engineering & Computer Science
University of Michigan

26th Soar Workshop
May 26, 2006
Ann Arbor, MI

objective

talk briefly about work on cortical maps and their possible relevance to Soar:

- mapping from cortex to Soar
- biologically inspired clustering
  - competitive learning algorithm
  - sensory transduction
  - higher-order symbolic representations

2

agenda…

① **mapping** symbols with similarity

② **model** self-organizing maps (SOMs)

③ **demo task** object categorization

④ **wrap-up** a useful mapping?

3

① motivation

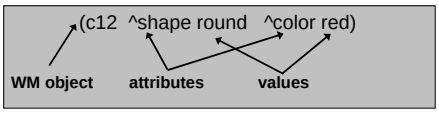
mapping the cortex to *Soar*

attributes

- cortical areas correspond to attributes (color, shape, etc.)
- connectivity between attributes is architectural (fixed by "nature", roughly)
- semantics of attributes are experiential ("nurture")

values

- active representation in a cortical area, *winning cell*



4

① motivation

a complication in SOAR

- similarity relations are problematic

(b32 ^shape round ^color red)

red apple

- symbols "red" and "pink" have no inherent similarity

pink apple

- no guarantee that if "red and round" tends to activate "apple", "pink and round" will do the same

5

agenda…

① **motivation** symbols with similarity

② **model** self-organizing maps (SOMs)

③ **demo task** object categorization

6

model

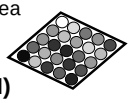
overview: self-organizing maps (SOM)

general features

- inspired by properties of cortical representations
- in class of competitive learning algorithms
- synaptic connectivity via "codebook vectors"
- single winning cell (attribute-value) via competition
- learn by moving winner's vector closer to input

unique feature of SOMs

- winner ***and*** spatial neighbors moved towards input
- similarity via 2D location in cortical area

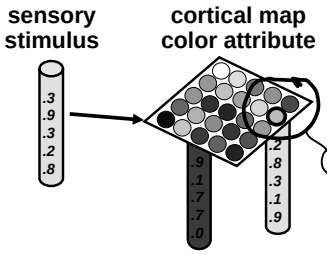


cortical map (SOM)

7

model

SOM learning algorithm (in a nutshell)



1. winner's codebook vector moved closer to input vector
2. neighbors' codebook vectors moved closer to input vector (by less)

→ with experience, regions of similarity develop.

→ spatially proximal cells have similar receptive fields; winning cell is value for attribute.

8

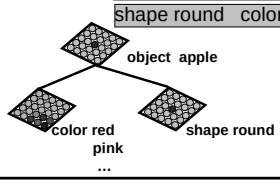
model

SOMs and symbols

- **SOMs**
  - can transduce continuous sensory inputs
  - and provide similarity, via topography

i.e. similar cells in one cortical map tend to excite the same cells in other areas of cortex.

shape round color red excites fruit apple
shape round color pink excites fruit apple



9

model

topography: important principle in brain

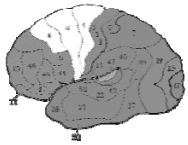
- **sensory cortex is topographically organized**
  - **visual**: *retinotopic* (based on retina, visual field)
  - **auditory**: *tonotopic* (based on auditory nerve, frequency)
  - **somatosensory**: *somatotopic* (based on location on the human body)
- **sensory topography not just in primary cortex**
  - starts in primary sensory cortex
  - continues to later stages of processing stream
- **when sensory-based topography ends, what comes next?**
  - no topography?
  - "semantic topography"?

10

model

taking SOMs to the next level…

- **situation:** SOMs work for sensory transduction, i.e. converting continuous valued inputs to symbols.
- **complication**: most cortical areas are **not** directly connected to sensory inputs, but to other cortical areas.



Direct Connections to Sensation
- Primary Visual
- Primary Auditory
- Primary Somatosensory

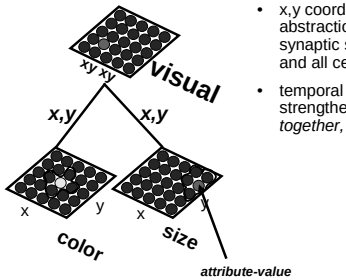
CorticoCortical Connections
- Uni-, multi-modal association

- **question:** how are representations learned in higher-order maps receiving symbolic inputs from other maps…while preserving similarity relations?

11

model

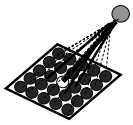
idea: encoding via 2D "cortical coordinates"



attribute-value
"large, horizontal orientation"

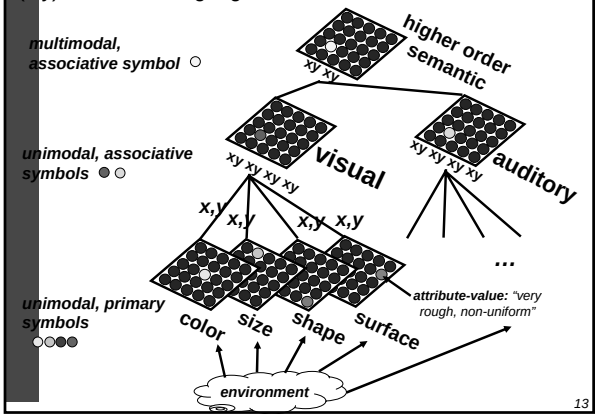
cell's receptive field in an afferent map

- x,y coordinate is a computational abstraction of the ***pattern*** of synaptic strength between a cell and all cells in an afferent map
- temporal coincidence of firing strengthens connections, "*fire together, wire together*"



12

(x,y): common language within a semantic network? ② model



13

agenda…

① **motivation** symbols with similarity

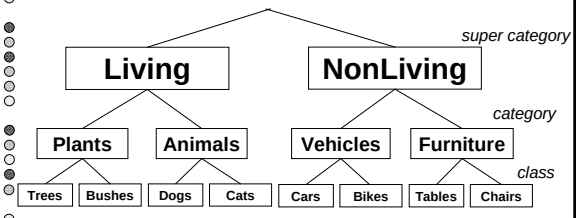
② **model** self-organizing maps (SOMs)

③ **demo task** object categorization

14

③ task

example task: object categorization



- **96 exemplars, 12 per object class**
  e.g. 12 trees: 3 colors x 4 sizes
- **overlapping attribute-values**
  e.g. size: small dog ~ size large cat
  surface: car ~ table

15

③ task

stimulus attributes (assumed continuous valued)

Visual Perception		
color	(hue, saturation, brightness)	[0..1]
size	($size_x$, $size_y$ $size_z$)	[feet]
shape	(roundness, complexity)	[0..1]
surface	(smoothness, uniformity)	[0..1]
Auditory Perception		
sound	(loudness, char. freq)	[0..1,Hz]

16

③ task

attribute coding: independently motivated

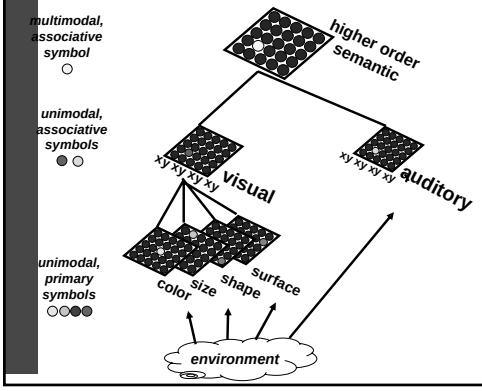
dog shape: (***roundness***: 0.85, ***complexity***: 0.15)

dog colors: (***H/S/B:*** (0.1,0.6,0.6), (0,0,0.1) , (0,0,1), (0,0,0.5))

17

③ task

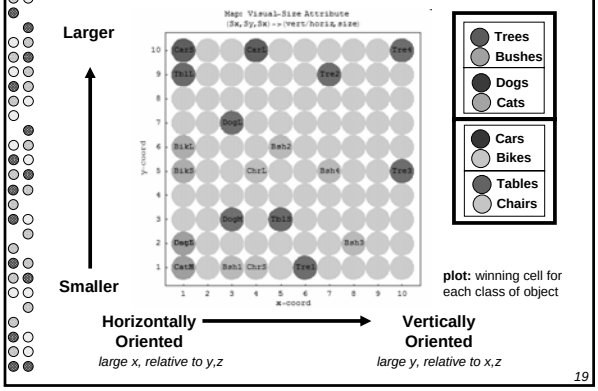
model architecture



18

❸ task

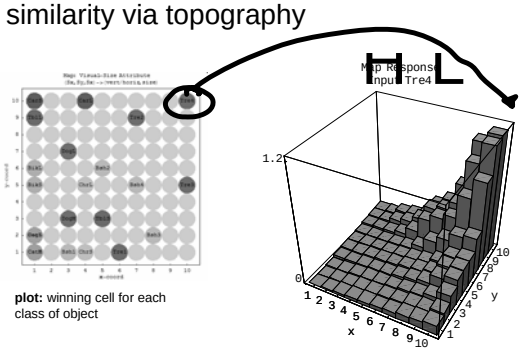
transduction example: size(x,y,z) to 2D-SOM



19

❸ task

similarity via topography



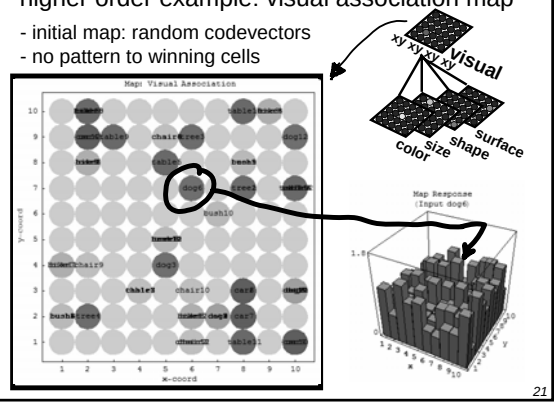
- winning cell, has largest response to Tree4 (tall, skinny tree)
- spatial neighbors have similar receptive field

20

❸ task

higher order example: visual association map

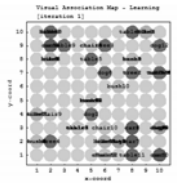
- initial map: random codevectors
- no pattern to winning cells



21

❸ task

Learning - Movie



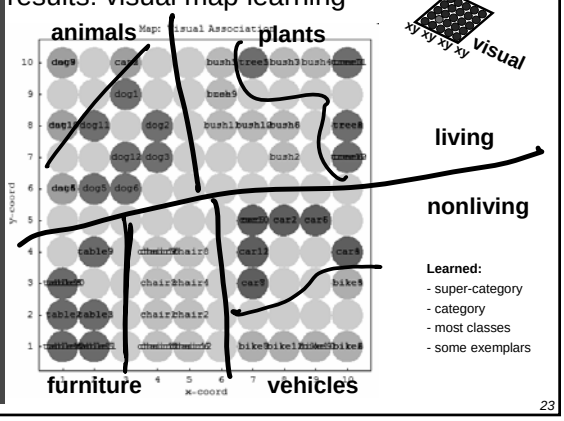
- 50 iterations (“iteration”, one pass through 96 training exemplars)
- Supercategory (living/nonliving) and category (plant, animal, vehicle, furniture) learned rapidly (a few iterations)
- Object classes (tree/bush, car/bike, etc) and individual exemplars learned later

..\Models\Object Demo\SOM Learning Movie.avi

22

❸ task

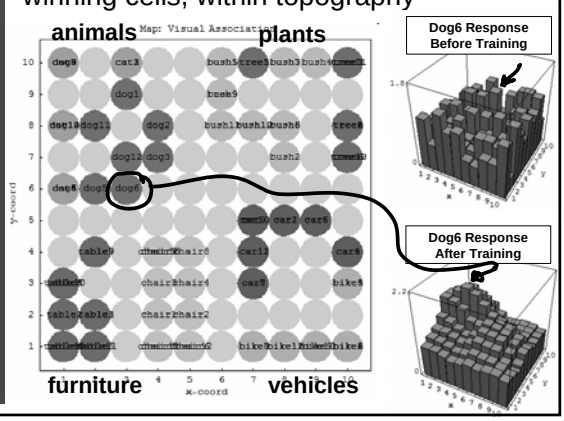
results: visual map learning



23

❸ task

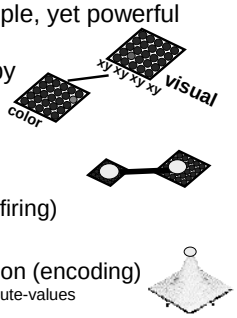
winning cells, within topography



wrap-up

"cortical coordinates": simple, yet powerful

- semantics determined by four key properties:
  - architecture (wiring)
    evolutionary experience
  - temporal coincidence (firing)
    individual experience
  - topography + conjunction (encoding)
    relations between stimuli / attribute-values
  - competition (excitation + inhibition)
    discretization at the level of cells/columns/patches



25

wrap-up

^nuggets golden

- **clustering & similarity** via neurally-inspired competitive learning
- **sensory transduction** to symbols
- **higher-order semantics** at increasing levels of abstraction, via cortical coordinates

26

wrap-up

^nuggets coal

- **top-down effects:** require additional extensions of SOM model (in progress)
- **attentional modulation:** allow relative weighting of attributes based on goals, context (in progress)
- **practical considerations:** viability of semantic network in Soar based on SOMs? training? exploitation of knowledge?

27

Integrating Clustering and Semantic Memory in Soar

Yongjia Wang
John E. Laird

1

Research Goals

- To improve general functionality of Soar by semantic memory
  - Explore new cognitive capabilities
    - Category learning
- To understand semantic memory in the context of a general cognitive architecture
  - How to use semantic memory in specific tasks?
    - Hierarchical structure

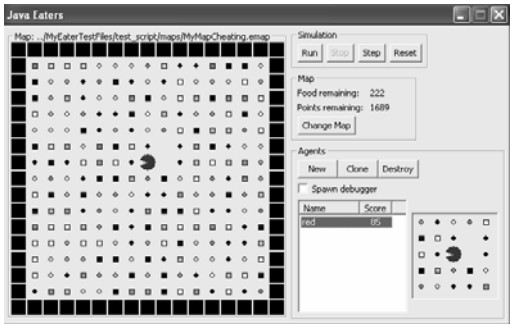
2

Overview of Experiment

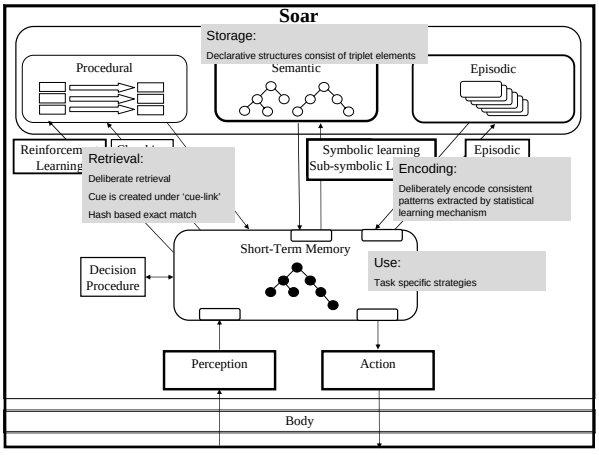
- Purpose:
  - Hook up external environment
  - Need more challenging task with stochastic environment
- Implementation:
  - Integrated statistical learning component
  - Semantic memory provides confidence of retrieval
- Task: Eater's domain
  - Interactive simulated environment
  - The environment is readily available
  - Enrich the domain: inject noise, hierarchical structure

3

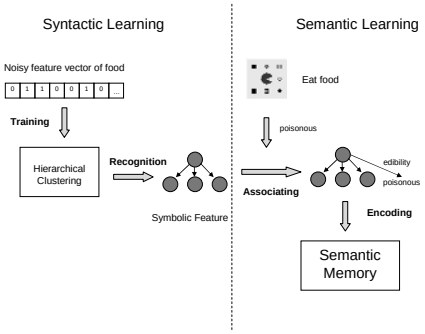
The Eater's Domain



4



Overview of Task and Implementation



6

Comparison between Alternative Approaches

- Semantic learning is based on saving and retrieving instances
  1. Save original instances without clustering
     - Number of unique instances increases linearly
     - Exact match based memory retrieval will not find matches
     - Partial match based memory retrieval is computational expensive
  2. Save instances with reduced features after clustering
     - Instances are collapsed into small set of categories
     - Representation has reduced dimension
     - Underlying structure is still preserved

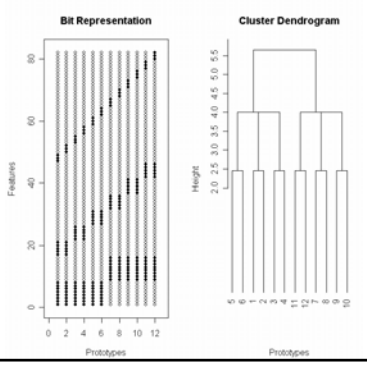
7

The Hierarchical Clustering Algorithm used in our Implementation

- Unsupervised learning
- Online learning algorithm
- Hierarchically refined classification

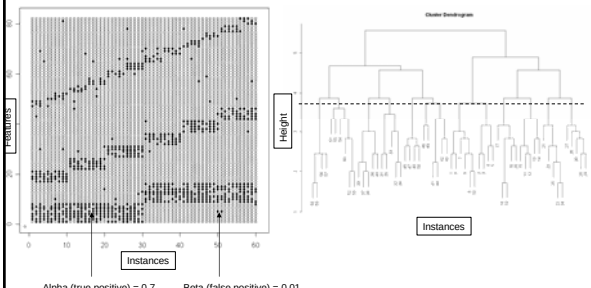
8

Food Prototypes



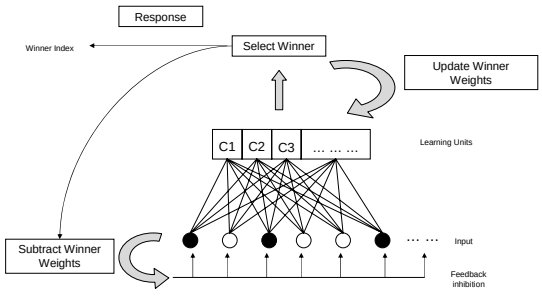
9

Food Instances with Noise



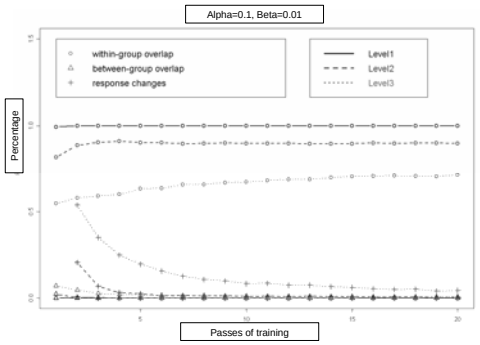
10

Hierarchical Clustering



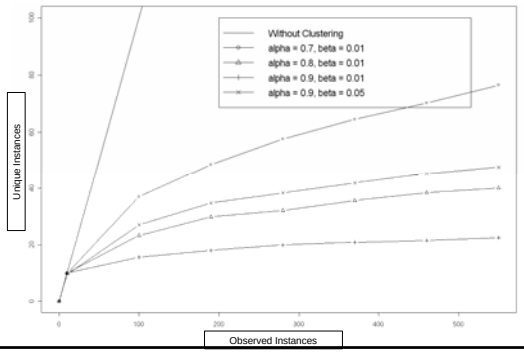
11

Noise Tolerance of the Hierarchical Clustering Algorithm



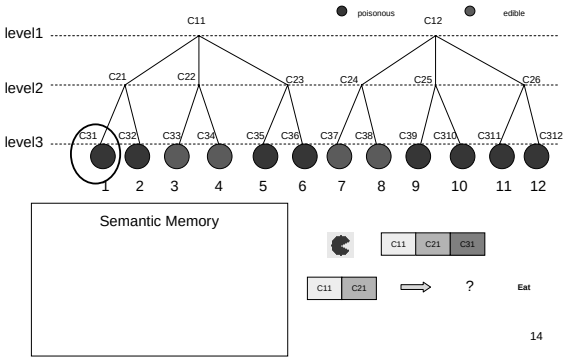
12

Clustering Reduces the Number of Unique Instances



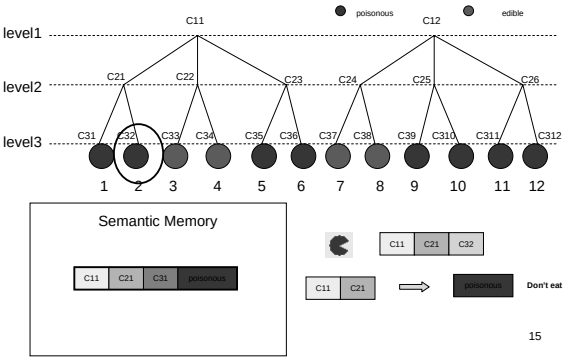
13

The Complete Task



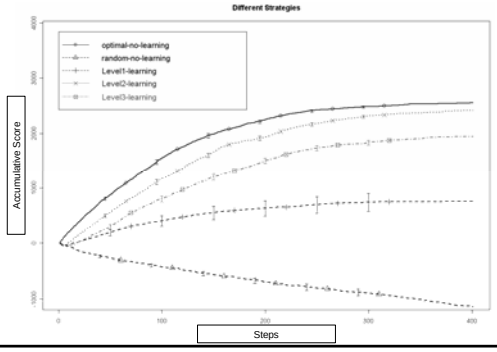
14

The Complete Task



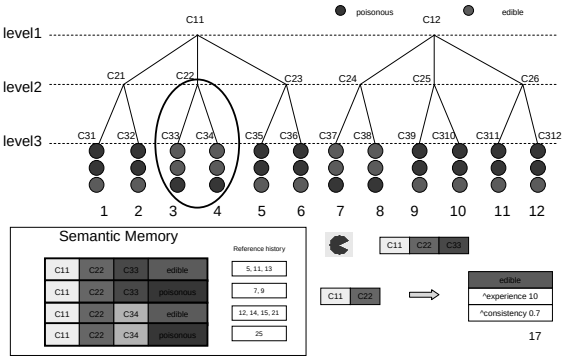
15

Compare Different Strategies



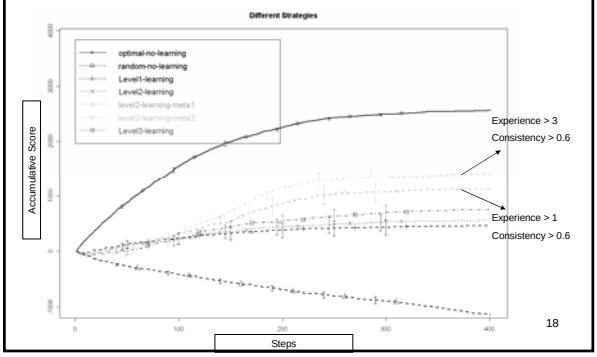
16

The Situation with More Noise



17

Retrieval Confidence Helps Decision Making



18

Summary

- Nuggets
  - Tested semantic memory in stochastic external environment
  - Integrated hierarchical clustering
  - New capability of learning abstract categories from instances (distinctive capability from episodic memory)
  - Semantic memory provides retrieval confidence useful for decision making
- Coals
  - The input in the task is arbitrarily constructed
  - Eater's domain is simple: simple reasoning, simple decision making and limited actions
  - Learning strategies in the experiment are simple
  - Haven't fully explored the benefit of hierarchical structure
  - Integration of hierarchical clustering algorithm is preliminary

19

Thank You

20

Biologically-Inspired Control in Problem Solving

Thad A. Polk, Patrick Simen,
Richard L. Lewis, & Eric Freedman

1

Computational Models of Control

- Challenge: Develop computationally explicit theories of control deficits in complex problem solving
  - Where control deficits are often most apparent
- Symbolic models (production systems)
  - E.g., ACT-R, Soar, Epic, ...
  - Natural model of flexible, goal-driven behavior and therefore easier to apply to complex problem solving
  - But harder to map onto the brain and patients
- Neural networks
  - E.g., Cohen, Levine, Dehaene, Braver, O'Reilly, ...
  - Neural mechanism for control (modulation) and therefore easier to map onto the brain and patients
  - But harder to apply to complex problem solving

2

Major Points

1. Natural & explicit mapping from goal-driven production systems onto neural computation
   - Makes it possible to build plausible neural models of complex problem solving
2. Mapping leads to explicit hypothesis about the role of DLPFC in problem solving:
   - Represents internally generated subgoals that modulate among choices
3. Applying to TOL accurately simulates human behavior
   - Intact model simulates normals, even on hardest problems
   - Lesioning subgoal net simulates prefrontal deficits

3

Plan

- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

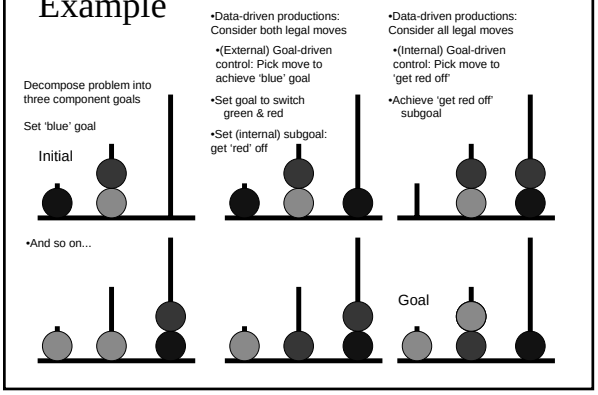
4

Goal-Driven Production Systems

- Almost all models of complex cognition based on production systems (ACT-R, Soar, Epic)
  - Set of symbolic IF-THEN rules that match against memory and take actions and/or change memory:
- Production systems proposed as control theory (Newell, 1973):
  - Allow behavior to be flexible, opportunistic, interruptible
  - Next step chosen dynamically based on what's in memory/world
  - Goals/subgoals modulate/control decisions

5

Example



Key Features of Symbolic Control

- Data-driven production rules
  - Asymmetric associations between symbols
  - Allows flexible behavior that reacts to current state
  - E.g., Recognizing legal moves in current TOL state
- Top-down control from current goal/subgoal
  - Constrains data-driven processing
  - Both external goals and internally generated subgoals can control
  - E.g., preferring moves that satisfy current subgoal over others

7

Plan

- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

8

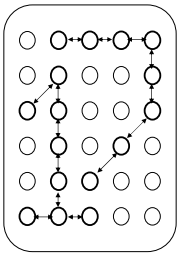
Neural Computation: Assumptions for Present Model

1. Neural processing is *recurrent*
2. Neural representations are *distributed*
3. Neural learning is *correlation-based* (Hebbian)

9

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



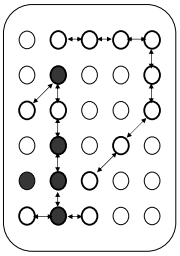
If distributed patterns occur frequently, they become discrete stable states for the network...

…and the network will converge on them given any similar patterns...

10

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



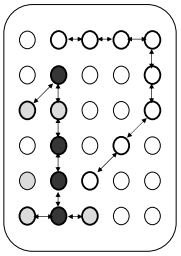
If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

11

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



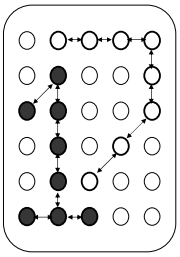
If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

12

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



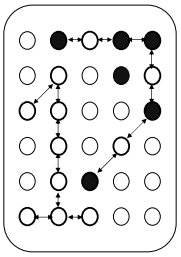
If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

13

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



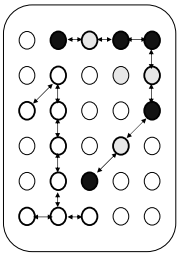
If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

14

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



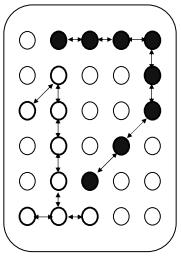
If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

15

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

These "attractors" are discrete & stable like symbols.

16

Plan

- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

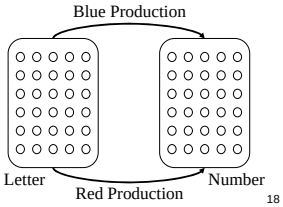
17

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



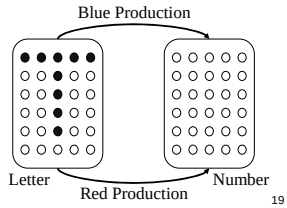
18

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



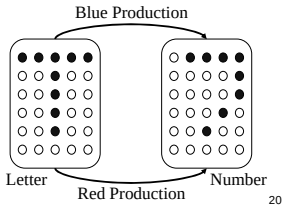
19

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



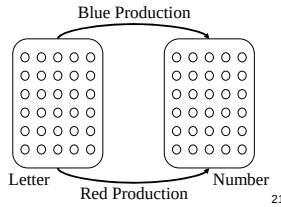
20

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



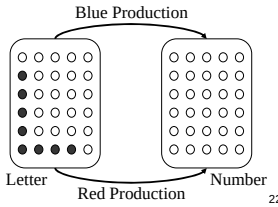
21

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



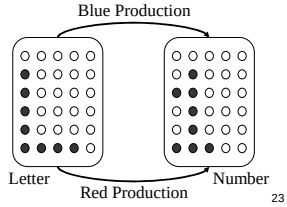
22

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



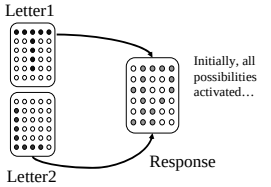
23

Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution

IF
Letter1 T
THEN
Response 7

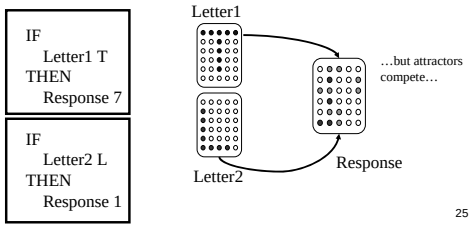
IF
Letter2 L
THEN
Response 1



24

Goal-Driven Control → Attractors

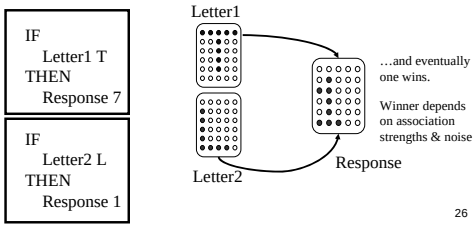
- Goals bias competition among attractors
  - A kind of conflict resolution



25

Goal-Driven Control → Attractors

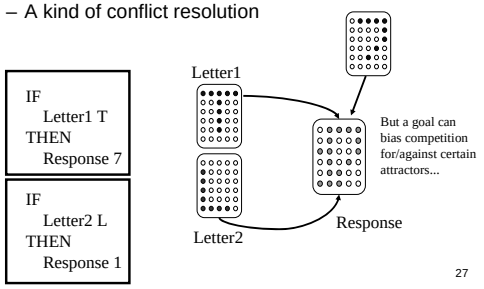
- Goals bias competition among attractors
  - A kind of conflict resolution



26

Goal-Driven Control → Attractors

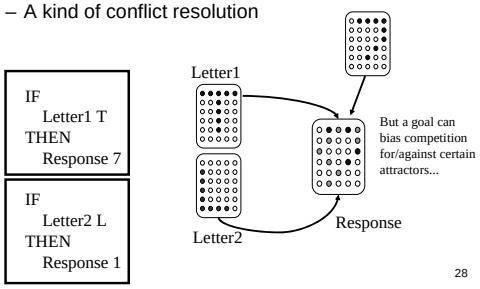
- Goals bias competition among attractors
  - A kind of conflict resolution



27

Goal-Driven Control → Attractors

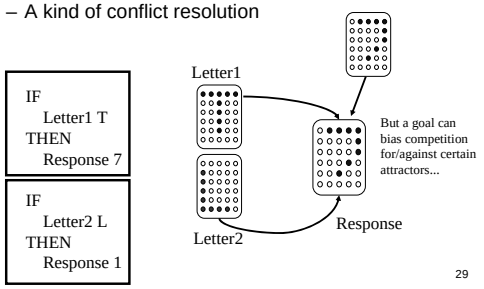
- Goals bias competition among attractors
  - A kind of conflict resolution



28

Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution



29

Mapping Productions To Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

- We've implemented this mapping in Lisp/Perl:
  - Input: Simplified production rules
  - Output: Matlab code that implements attractor nets that (often!) behave like the production system

- Can thus use the attractor architecture for some higher cognitive tasks

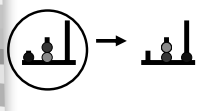
30

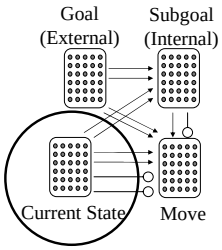
Plan

- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

31

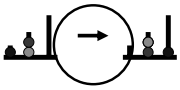
Modeling Tower of London

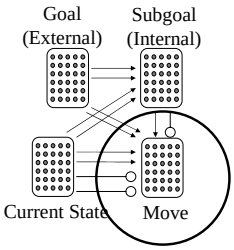




32

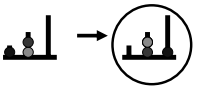
Modeling Tower of London

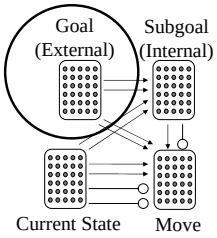




33

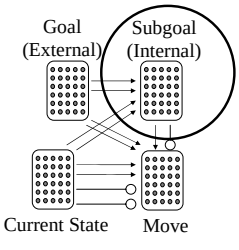
Modeling Tower of London





34

Modeling Tower of London

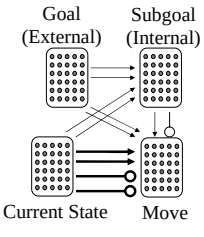


35

Modeling Tower of London

Knowledge

Legal Moves to consider

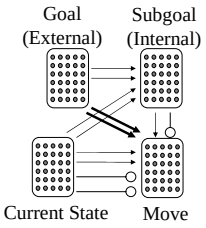


36

Modeling Tower of London

Knowledge

Legal Moves to consider
Try to achieve goals

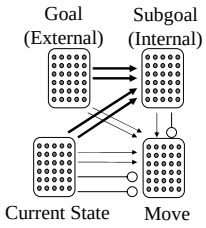


37

Modeling Tower of London

Knowledge

Legal Moves to consider
Try to achieve goals
Set subgoals to remove blockers

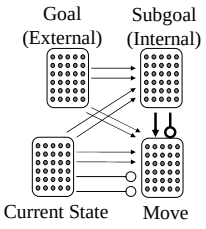


38

Modeling Tower of London

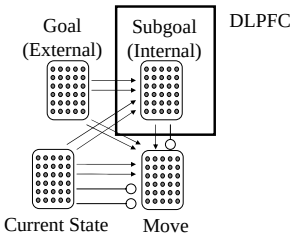
Knowledge

Legal Moves to consider
Try to achieve goals
Set subgoals to remove blockers
Try to achieve subgoals



39

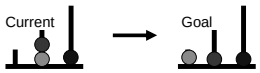
The Role of DLPFC

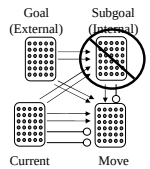


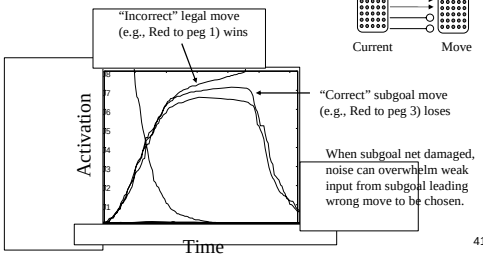
Hypothesis: DLPFC represents internal subgoals
- Modulates competition among posterior attractors
- Note: Not needed for externally provided goals

40

The Function of DLPFC Subgoals



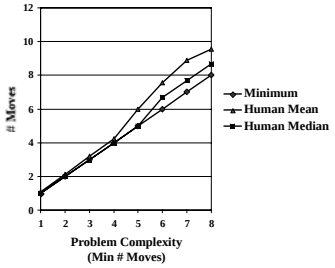




41

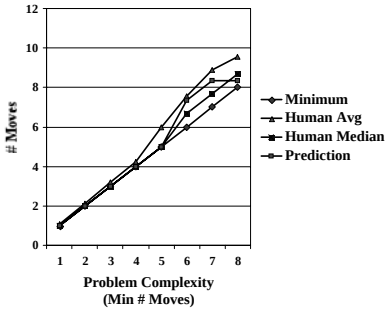
Human Data

42 undergraduates (Intact?)
18 TOL problems varying in minimum number of moves



42

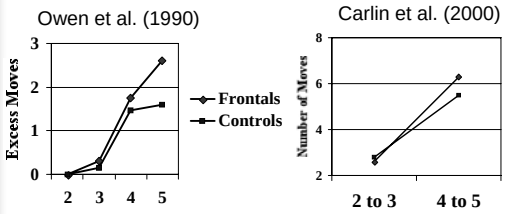
Intact Simulation



43

Prefrontal Data

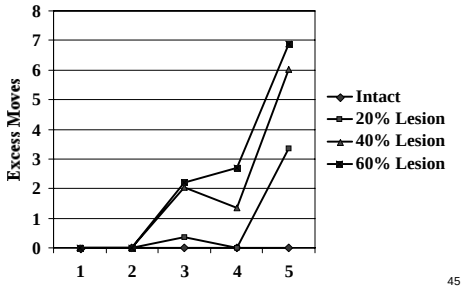
Group x Difficulty interaction:
Prefrontal deficit is more apparent on harder problems



Problem Complexity
(Minimum Number of Moves)

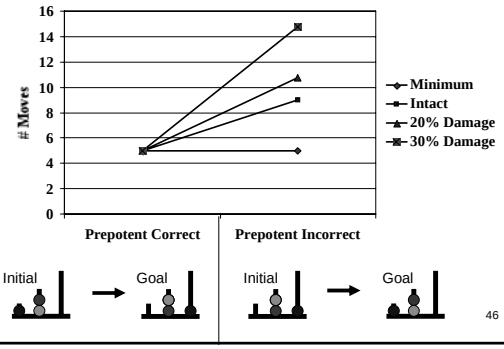
44

Prefrontal Simulations



45

Problems with Prepotent Responses



46

Issues Raised by the Mapping

- A number of features of production systems DON'T map easily:
  - Variables
  - Independence/modularity of different productions
  - All-or-none matching
- Possible responses:
  - Neural nets lack critical functionality for modeling cognition
    - Need to work on increasing their functionality
  - Production systems have too much functionality
    - Might use less powerful production systems that map more naturally
  - Both are probably reasonable

47

Summary

1. Natural & explicit mapping from goal-driven production systems onto neural computation
   - Makes it possible to build plausible neural models of complex problem solving
2. Mapping leads to explicit hypothesis about the role of DLPFC in problem solving:
   - Represents internally generated subgoals that modulate among choices
3. Applying to TOL accurately simulates human behavior
   - Intact model simulates normals, even on hardest problems
   - Lesioning subgoal net simulates prefrontal deficits

48