ADAPT: A Cognitive Architecture for Robots

D. Paul Benjamin

Pace University Robotics Laboratory

Deryle Lonsdale

Brigham Young University Department of Linguistics and English Language

Damian Lyons

Fordham University Robotics Laboratory

ADAPT

Adaptive Dynamics and Active Perception for Thought

The goal of the ADAPT project is to create a robot that can model its environment accurately in real time, and use that model to perform tasks and interact with people using natural language.

We are not interested in robot programming for particular tasks but in investigating embodying cognition.

The emphasis in ADAPT is on solving problems by reformulation (reperception).

The structure of the ADAPT architecture is based on linguistics.

We are building ADAPT by implementing the RS (Robot Schemas) language in the Soar cognitive architecture.

ADAPT uses a sophisticated, multimedia internal world model. Comprehension is modeled as search to reconstruct the environment within this world model.

Overview

Overall structure and philosophy

RS/Soar

Coherence Theory

Virtual World

Visualization and Semantics

Lessons Learned the Hard Way:

Everything is sensory-motor.

Perception is an active, goal-directed process.

Robotics requires a high degree of true concurrency.

Analyzing interactions is just too hard.

Active Perception

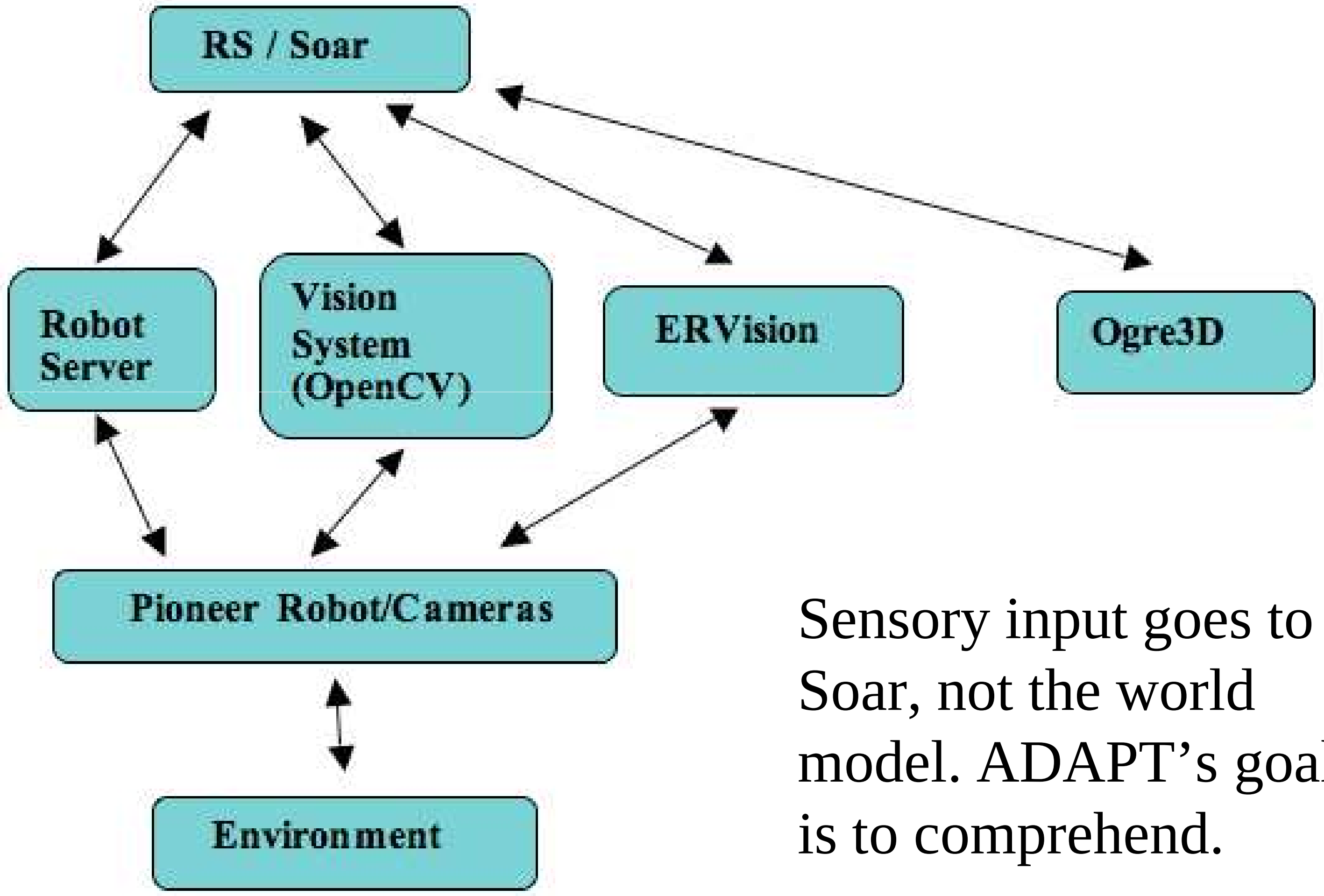
Active perception is top-down and goal-directed, so that perception becomes a problem solving process. This is in contrast to the way perception is usually approached in AI and cognitive science. Cognitive robotics means more than just using a cognitive architecture on a robot; it means treating robotics as a cognitive domain, which includes treating perception as problem solving.

Perception is the hard problem in robotics. This distinguishes robotics from tasks like chess.

We view perception as the representation problem. Perception is the construction and modification of problem-solving representations.

The representation problem is at least hyperexponential ($O^{2^{2^n}}$).

ADAPT's Structure



Sensory input goes to Soar, not the world model. ADAPT's goal is to comprehend.

ADAPT integrates distributed, concurrent control with cognitive structures

The RS (Robot Schemas) language is the basis of the robotics capabilities of ADAPT. RS is precise and mature.

RS is a CSP-type programming language for robotics, that controls a hierarchy of concurrently executing schemas.

$\mathrm{Joint}_i(s)() = [\mathrm{Jpos}_i()(x), \mathrm{Jset}_i(s, x)(u), \mathrm{Jmot}_i(u)()\ ]^{c0}$

c0: $(\mathrm{Jpos}_i, x)\ (\mathrm{Jset}, x)$ $(\mathrm{Jset}, u)\ (\mathrm{Jmot}_i, u)$

$\mathrm{Jpos}_i()(x)$ continuously reports the position of joint i on port x
$\mathrm{Jmot}_i(u)()$ accepts a signal on port u and applies it to the actuator of joint i
$\mathrm{Jset}_i(s, x)(u)$ accepts a setpoint on port s and iteratively inputs a joint position on port x and outputs a motor signal on port u to drive the joint position to the setpoint

A Sensory-motor Schema Hierarchy

$$\text{Touch}_i = [\text{Tact}_i()(v), \text{Gmove}_i(v)(y), \text{Joint}_i(y)()\ ]^{c1}$$

c1: $(\text{Tact}_i, v)\ (\text{Gmove}, v)$ $(\text{Gmove}, y)\ (\text{Joint}_i, y)$

Tact_i reports on tactile contact on the i-th join on its port v.

Gmove increments the setpoint of the joint actuator as long as it gets a no-contact signal on its port v.

Touch_i implements a guarded move of the i-th link.

RS has a Formal Semantics

$P = (Q, L, X\ \delta, \beta, \tau)$ where

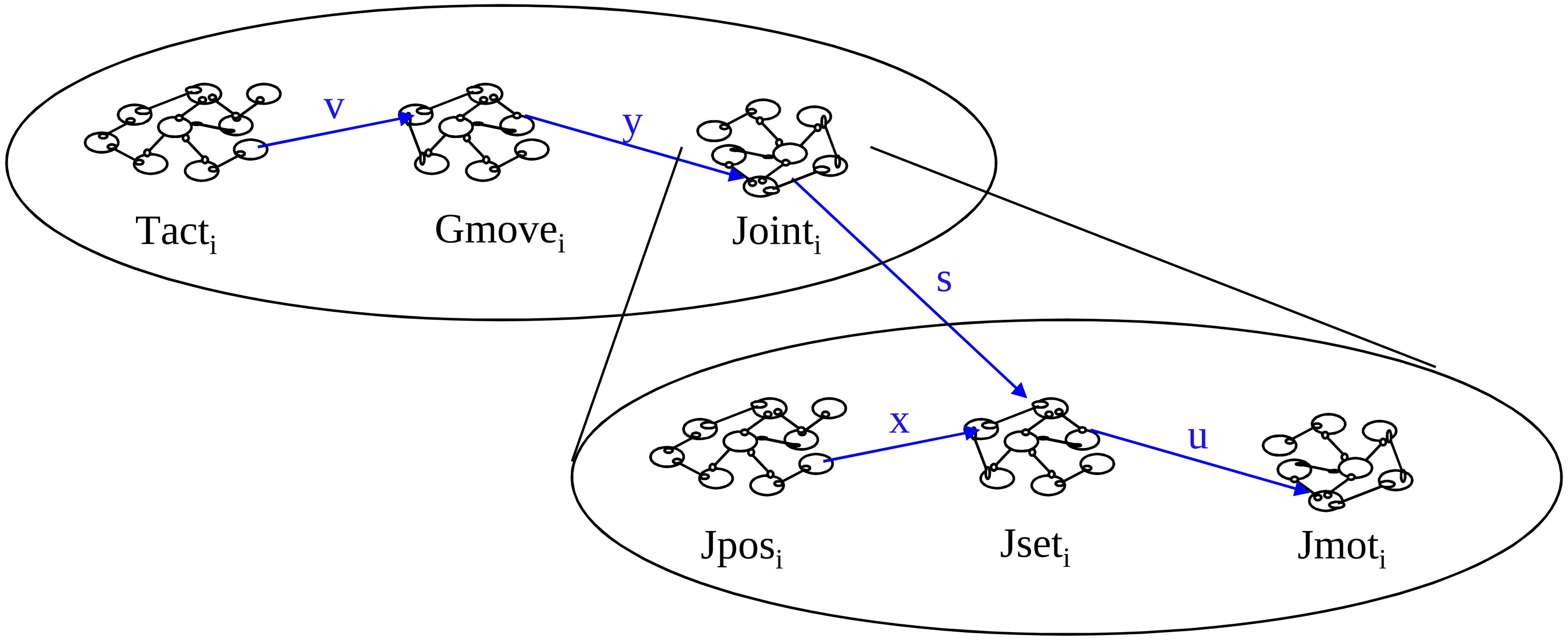
Q is the set of states
L is the set of ports
$X = (X_i \mid i \in L)$ is the event alphabet for each port

$XL = \{ (i, X_i) \mid i \in L \}$ i.e., a disjoint union of L and X

$\delta : Q \times XL \rightarrow 2^Q$ is the transition function
$\beta = (\beta_i \mid i \in L)\ \beta_i : Q \rightarrow X_i$ is the output map for port i
$\tau \in 2^Q$ is the set of start states

RS has a Formal Semantics

The behavior of every RS schema is defined using port automata. This provides precision to the semantics and also a constructive means of reasoning about the behavior and meaning of schemas.



Process Composition in RS

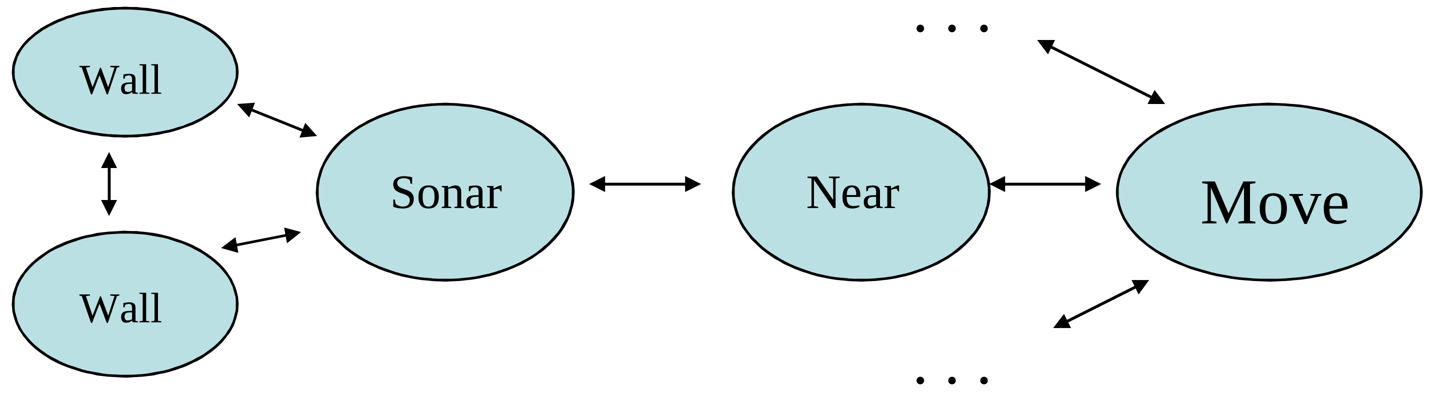
1. Sequential Composition: $\mathtt{T} = \mathtt{P};\mathtt{Q}$. The process $\mathtt{T}$ behaves like the process $\mathtt{P}$ until that terminates, and then behaves like the process $\mathtt{Q}$ (regardless of $\mathtt{P}$'s termination status).
2. Concurrent Composition: $\mathtt{T} = (\mathtt{P} \mid \mathtt{Q})^c$. The process $\mathtt{T}$ behaves like $\mathtt{P}$ and $\mathtt{Q}$ running in parallel and with the input ports of one connected to the output ports of the other as indicated by the port-to-port connection map c. This can also be written as $\mathtt{T} = (\bigm|_{i \in I} \mathtt{P}i\ )^c$ for a set of processes indexed by I.
3. Conditional Composition: $\mathtt{T} = \mathtt{P}\langle v \rangle : \mathtt{Q}_v$. The process $\mathtt{T}$ behaves like the process $\mathtt{P}$ until that terminates. If $\mathtt{P}$ aborts, then $\mathtt{T}$ aborts. If $\mathtt{P}$ terminates normally, then the value v calculated by $\mathtt{P}$ is used to intialize the process $\mathtt{Q}$, and $\mathtt{T}$ then behaves like $\mathtt{Q}_v$.
4. Disabling Composition: $\mathtt{T} = \mathtt{P}\#\mathtt{Q}$. The process $\mathtt{T}$ behaves like the concurrent composition of $\mathtt{P}$ and $\mathtt{Q}$ until either terminates, then the other is aborted and $\mathtt{T}$ terminates. At most one process can stop; the remainder are aborted.
5. Synchronous Recurrent Composition: $\mathtt{T} = \mathtt{P}\langle v \rangle :; \mathtt{Q}_v$. This is a recursively defined as follows: $\mathtt{P} :; \mathtt{Q} = \mathtt{P} : (\mathtt{Q}; \mathtt{P} :; \mathtt{Q})$.
6. Asynchronous Recurrent Composition: $\mathtt{T} = \mathtt{P}\langle v \rangle :: \mathtt{Q}_v$. This is recursively defined as follows: $\mathtt{P} :: \mathtt{Q} = \mathtt{P} : (\mathtt{Q} \mid (\mathtt{P} :: \mathtt{Q}))$.

Operator Precedence: The operator precedence from loosest to tightest is as follows: Concurrent; Disabling; Sequential; Conditional; Synchronous Recurrent; Asynchronous Recurrent.

Implementation of RS in Soar

Schemas, facts, and hypotheses are nodes in a graph.
Links implement the composition operations, as well as other relations, including deductive and evidential inference.

Automata that implement a schema are built as needed.



Advantages of RS

Formal Semantics
Complete Representation of Distributed Control
Maturity
Invariants to monitor tasks/environment

Disadvantages of RS

No synthesis method
No cognitive plausibility
No learning

Goal: to use chunking to learn RS schemas

RS has been successfully used in factory automation.

QuickTime™ and a
Microsoft Video 1 decompressor
are needed to see this picture.

However, this was a hand coded planner using Allen’s interval logic.

Coherence Theory

A Method of abductive inference for evidential reasoning with a claim to psychological plausibility

Represents evidence, facts and hypotheses in a constraint network. Each of these can be incomplete and/or inaccurate.

Searches for a set of accepted (believed) hypotheses that satisfies the maximal weighted sum of constraints

Used by Johnson et al.: (unfinished)

- Thagard: scientific discovery, jury decisions, problem solving
- Ranney & Thagard: students solving physics problems
- Mead & Miller: perception of social relationships

Implementation of Coherence Theory

Rather than use a connectionist approach, ADAPT implements coherence theory in Soar.

> Evidence, facts and hypotheses are connected by constraints of varying types (deductive or evidential).
>
> A model checker proves some of the hypotheses to be true/false. True hypotheses are *Accepted*; false ones *Rejected*. Other hypotheses are randomly assigned initially.
>
> Operators compute the amount of constraint satisfaction to be gained by changing each hypothesis to the other set. The hypothesis with the highest gain is switched. Repeat until local maximum reached.

Fast: typically under 250 ms.

Experience in Cybersecurity Domain

DARPA-funded project with BBN Technologies and Adventium Labs.

Goal: To defend a network of > 30 hosts from insider threats, catching at least 50% of attacks with no more than 10% false positives, and response in less than 250 ms.

Computer network is simulated in JESS.

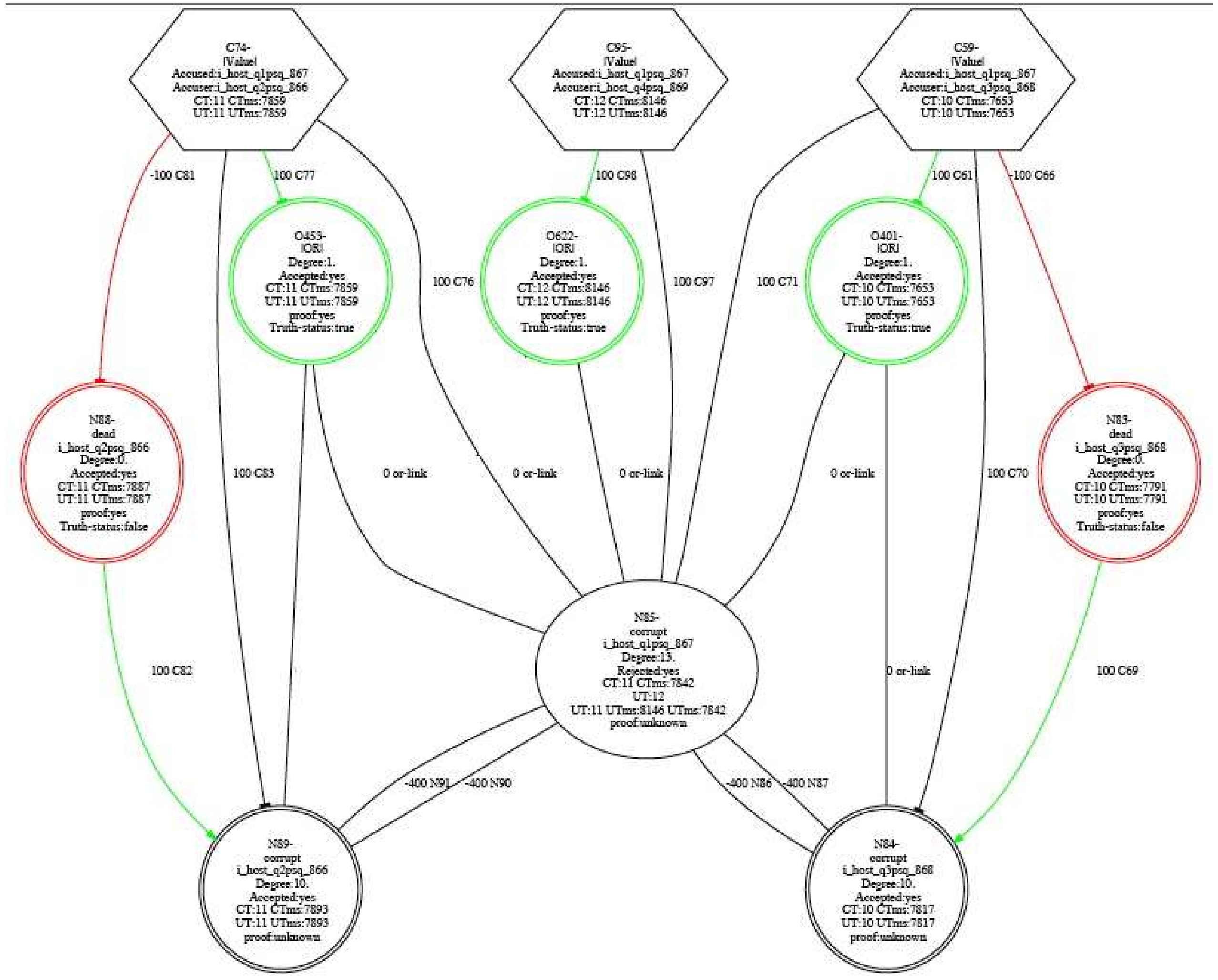
Successfully handles all scenarios from 2005 tests, as well as tests from a problem generator.

Fast and accurate. Hypothesis networks often > 100 nodes, sometimes > 500 nodes.

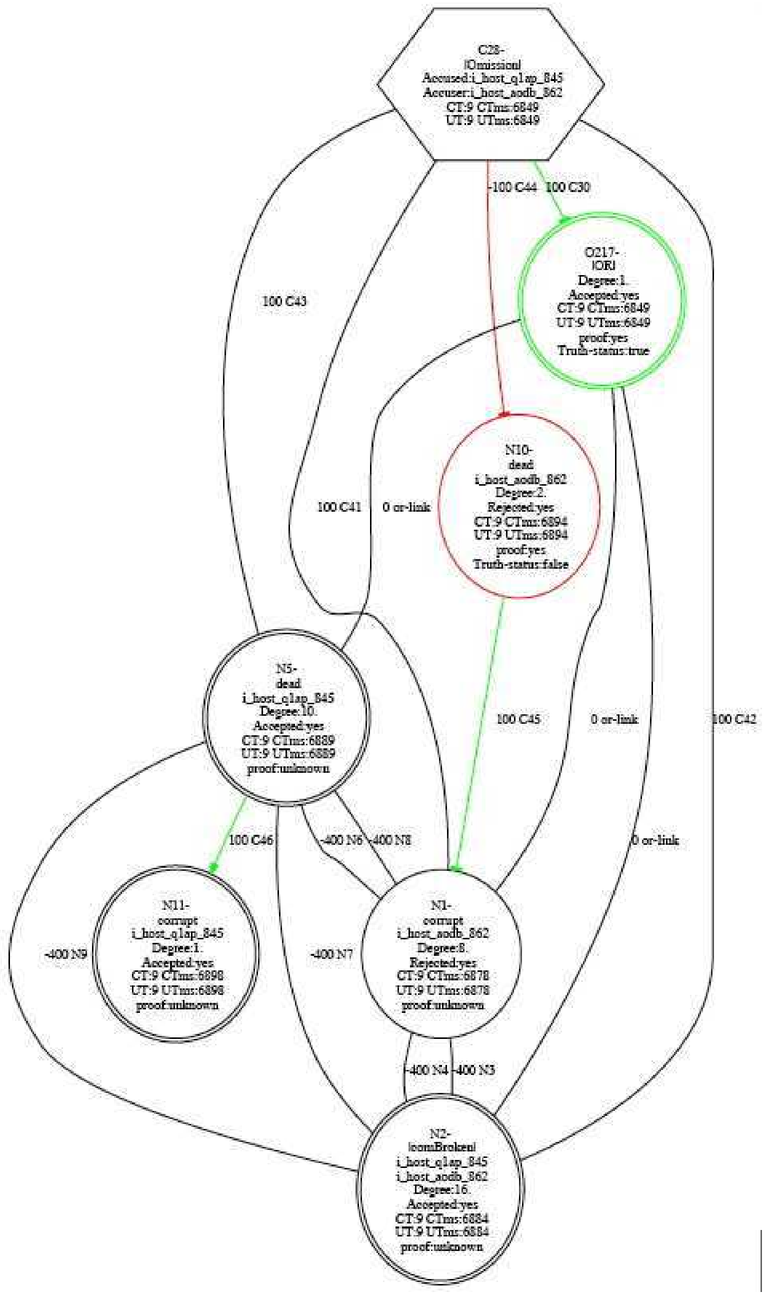
Red Team test to occur in late May, 2008.

Cybersecurity Example Hypothesis Network

Run #11 from November, 2005. The hypothesis network has two components:



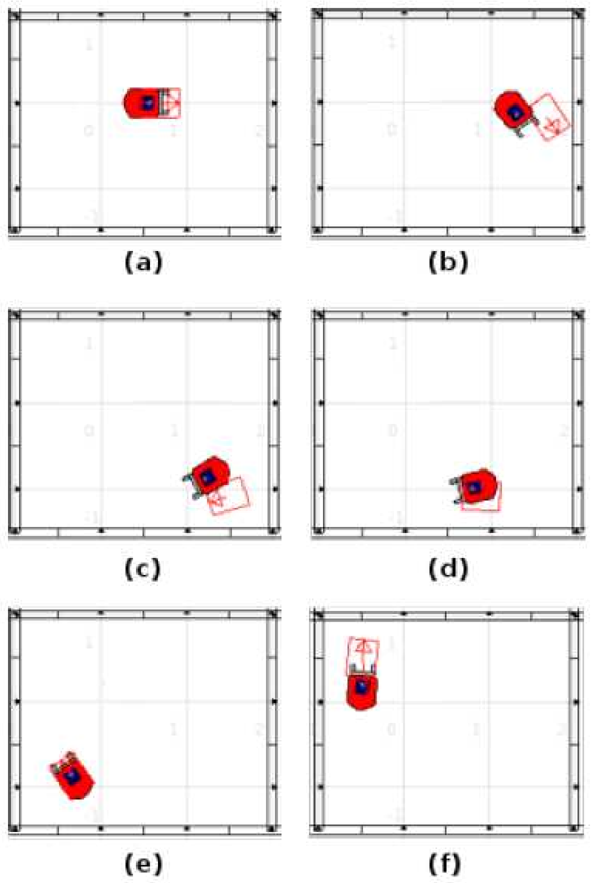
C74- |Value| Accused:i_host_q1psq_867 Accuser:i_host_q2psq_866 CT:11 CTms:7859 UT:11 UTms:7859
C95- |Value| Accused:i_host_q1psq_867 Accuser:i_host_q4psq_869 CT:12 CTms:8146 UT:12 UTms:8146
C59- |Value| Accused:i_host_q1psq_867 Accuser:i_host_q3psq_868 CT:10 CTms:7653 UT:10 UTms:7653
-100 C81
100 C77
100 C98
100 C61
-100 C66
O453- |OR| Degree:1. Accepted:yes CT:11 CTms:7859 UT:11 UTms:7859 proof:yes Truth-status:true
100 C76
O622- |OR| Degree:1. Accepted:yes CT:12 CTms:8146 UT:12 UTms:8146 proof:yes Truth-status:true
100 C97
100 C71
O401- |OR| Degree:1. Accepted:yes CT:10 CTms:7653 UT:10 UTms:7653 proof:yes Truth-status:true
N88- dead i_host_q2psq_866 Degree:0. Accepted:yes CT:11 CTms:7887 UT:11 UTms:7887 proof:yes Truth-status:false
100 C83
0 or-link
0 or-link
0 or-link
0 or-link
100 C70
N83- dead i_host_q3psq_868 Degree:0. Accepted:yes CT:10 CTms:7791 UT:10 UTms:7791 proof:yes Truth-status:false
N85- corrupt i_host_q1psq_867 Degree:13. Rejected:yes CT:11 CTms:7842 UT:12 UT:11 UTms:8146 UTms:7842 proof:unknown
100 C82
0 or-link
100 C69
-400 N91
-400 N90
-400 N86
-400 N87
N89- corrupt i_host_q2psq_866 Degree:10. Accepted:yes CT:11 CTms:7893 UT:11 UTms:7893 proof:unknown
N84- corrupt i_host_q3psq_868 Degree:10. Accepted:yes CT:10 CTms:7817 UT:10 UTms:7817 proof:unknown



C28-
|Omission|
Accused:i_host_q1ap_845
Accuser:i_host_aodb_862
CT:9 CTms:6849
UT:9 UTms:6849
-100 C44
100 C30
O217-
|OR|
Degree:1.
Accepted:yes
CT:9 CTms:6849
UT:9 UTms:6849
proof:yes
Truth-status:true
100 C43
100 C41
0 or-link
N10-
dead
i_host_aodb_862
Degree:2.
Rejected:yes
CT:9 CTms:6894
UT:9 UTms:6894
proof:yes
Truth-status:false
N5-
dead
i_host_q1ap_845
Degree:10.
Accepted:yes
CT:9 CTms:6889
UT:9 UTms:6889
proof:unknown
100 C45
0 or-link
100 C42
100 C46
-400 N6
-400 N8
0 or-link
N11-
corrupt
i_host_q1ap_845
Degree:1.
Accepted:yes
CT:9 CTms:6898
UT:9 UTms:6898
proof:unknown
-400 N9
-400 N7
N1-
corrupt
i_host_aodb_862
Degree:8.
Rejected:yes
CT:9 CTms:6878
UT:9 UTms:6878
proof:unknown
-400 N4
-400 N3
N2-
|commBroken|
i_host_q1ap_845
i_host_aodb_862
Degree:16.
Accepted:yes
CT:9 CTms:6884
UT:9 UTms:6884
proof:unknown

SoarSLAM Example

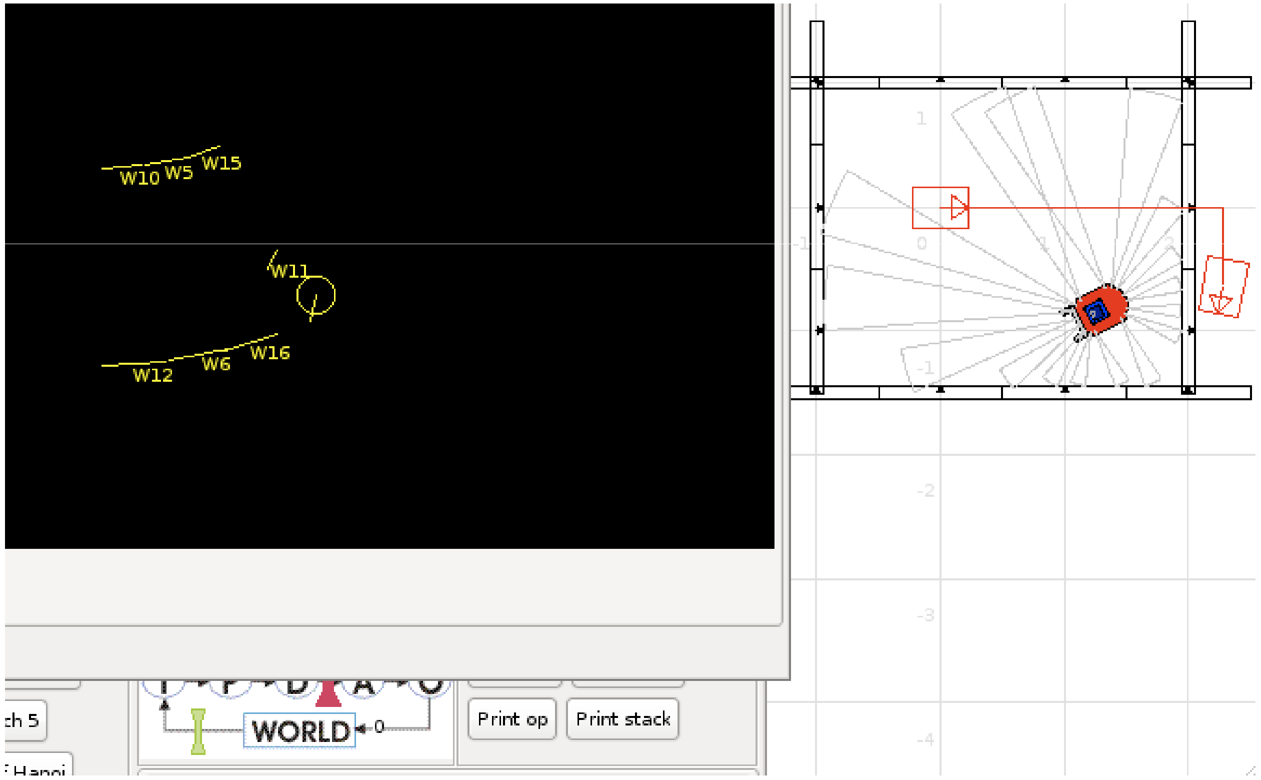
Sonar SLAM written in Soar. Runs on the Pioneer robots and simulator. Successfully maps floors of office buildings.



Dead reckoning errors accumulate. The empty red rectangle shows where the robot thinks it is.

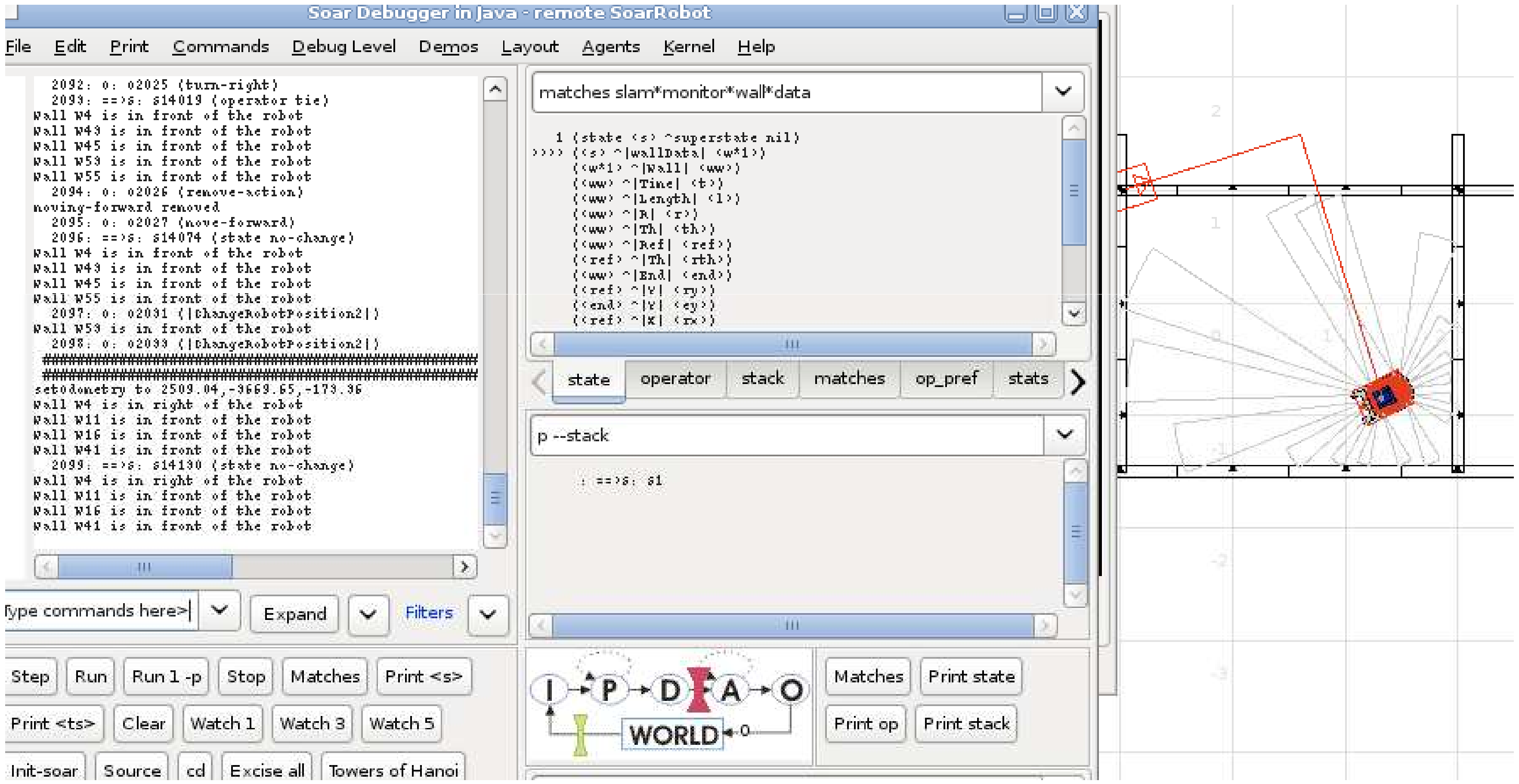
SoarSLAM Example

Sonar SLAM creates local maps consisting of wall segments.
Hypotheses are the robot’s position relative to local maps.



SoarSLAM Example

SoarSLAM switches local maps and resets the odometry.


Soar Debugger in Java - remote SoarRobot
File Edit Print Commands Debug Level Demos Layout Agents Kernel Help
Type commands here>
Expand
Filters
state operator stack matches op_pref stats
Step Run Run 1 -p Stop Matches Print <s>
Print <ts> Clear Watch 1 Watch 3 Watch 5
Init-soar Source cd Excise all Towers of Hanoi
I P D A O
WORLD
Matches Print state
Print op Print stack
2
1
0
-1
-2
-3

```
  2092: O: O2025 (turn-right)
  2093: ==>S: S14019 (operator tie)
Wall W4 is in front of the robot
Wall W43 is in front of the robot
Wall W45 is in front of the robot
Wall W53 is in front of the robot
Wall W55 is in front of the robot
  2094: O: O2026 (remove-action)
moving-forward removed
  2095: O: O2027 (move-forward)
  2096: ==>S: S14074 (state no-change)
Wall W4 is in front of the robot
Wall W43 is in front of the robot
Wall W45 is in front of the robot
Wall W55 is in front of the robot
  2097: O: O2031 (|ChangeRobotPosition2|)
Wall W53 is in front of the robot
  2098: O: O2033 (|ChangeRobotPosition2|)
 ############################################################
 ############################################################
setOdometry to 2509.04,-3669.65,-173.36
Wall W4 is in right of the robot
Wall W11 is in front of the robot
Wall W16 is in front of the robot
Wall W41 is in front of the robot
  2099: ==>S: S14130 (state no-change)
Wall W4 is in right of the robot
Wall W11 is in front of the robot
Wall W16 is in front of the robot
Wall W41 is in front of the robot
```

```
matches slam*monitor*wall*data

    1 (state <s> ^superstate nil)
>>>> (<s> ^|wallData| <w*1>)
     (<w*1> ^|Wall| <ww>)
     (<ww> ^|Time| <t>)
     (<ww> ^|Length| <l>)
     (<ww> ^|R| <r>)
     (<ww> ^|Th| <th>)
     (<ww> ^|Ref| <ref>)
     (<ref> ^|Th| <rth>)
     (<ww> ^|End| <end>)
     (<ref> ^|Y| <ry>)
     (<end> ^|Y| <ey>)
     (<ref> ^|X| <rx>)
```

```
p --stack

     : ==>S: S1
```

Lesson and Question

This simple example shows that robotics algorithms can be realized in a cognitively plausible manner within a cognitive architecture.

Is this cognitive science?

The basic loop of ADAPT is:

1 - check Soar's output link to see if there are any commands, which may be either motion commands for the robot or modeling commands for the World Model,

2 - blend the motion commands that are to be sent to the robot,

3 - send all robot commands both to the robot and to the virtual robot in the World Model,

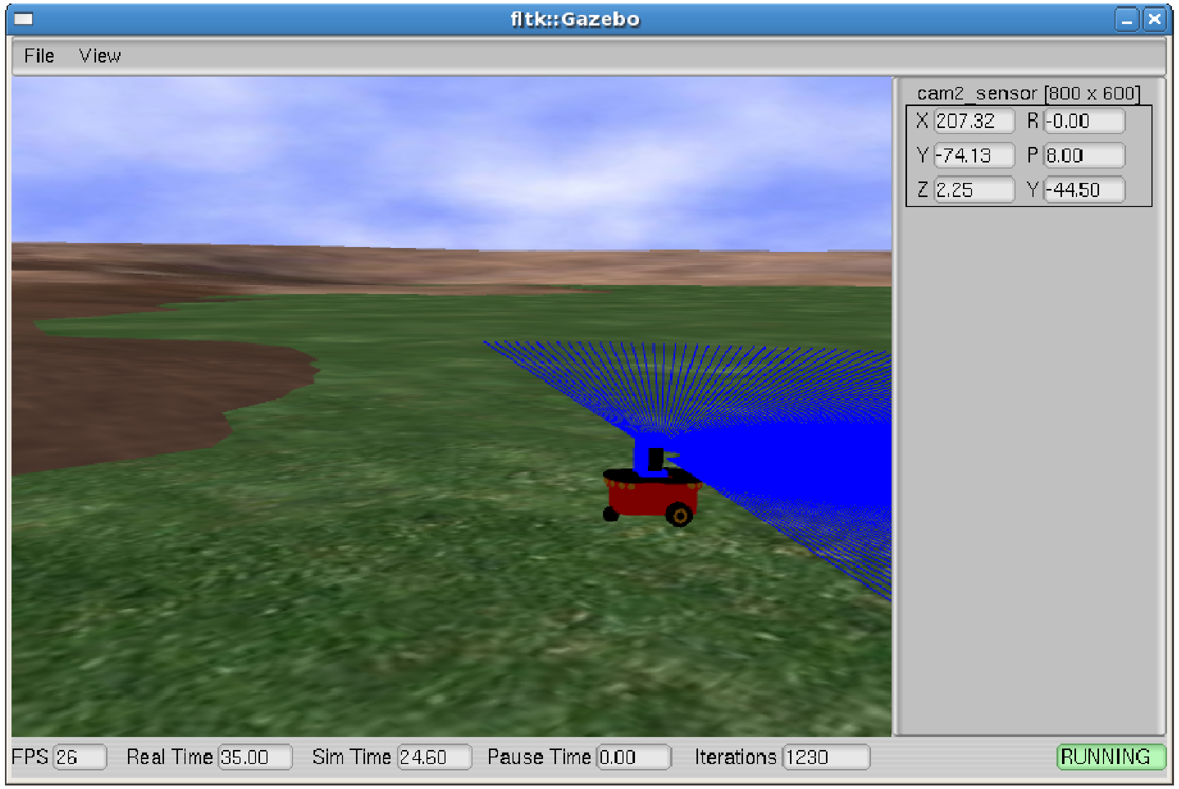
4 - send all other commands to the World Model,

5 – periodically (every tenth of a second) fetch data from the robot to be put into Soar's working memory,

6 - periodically fetch data from the Vision System, compare it to visual data from the World Model, and put any significant differences into Soar's working memory.

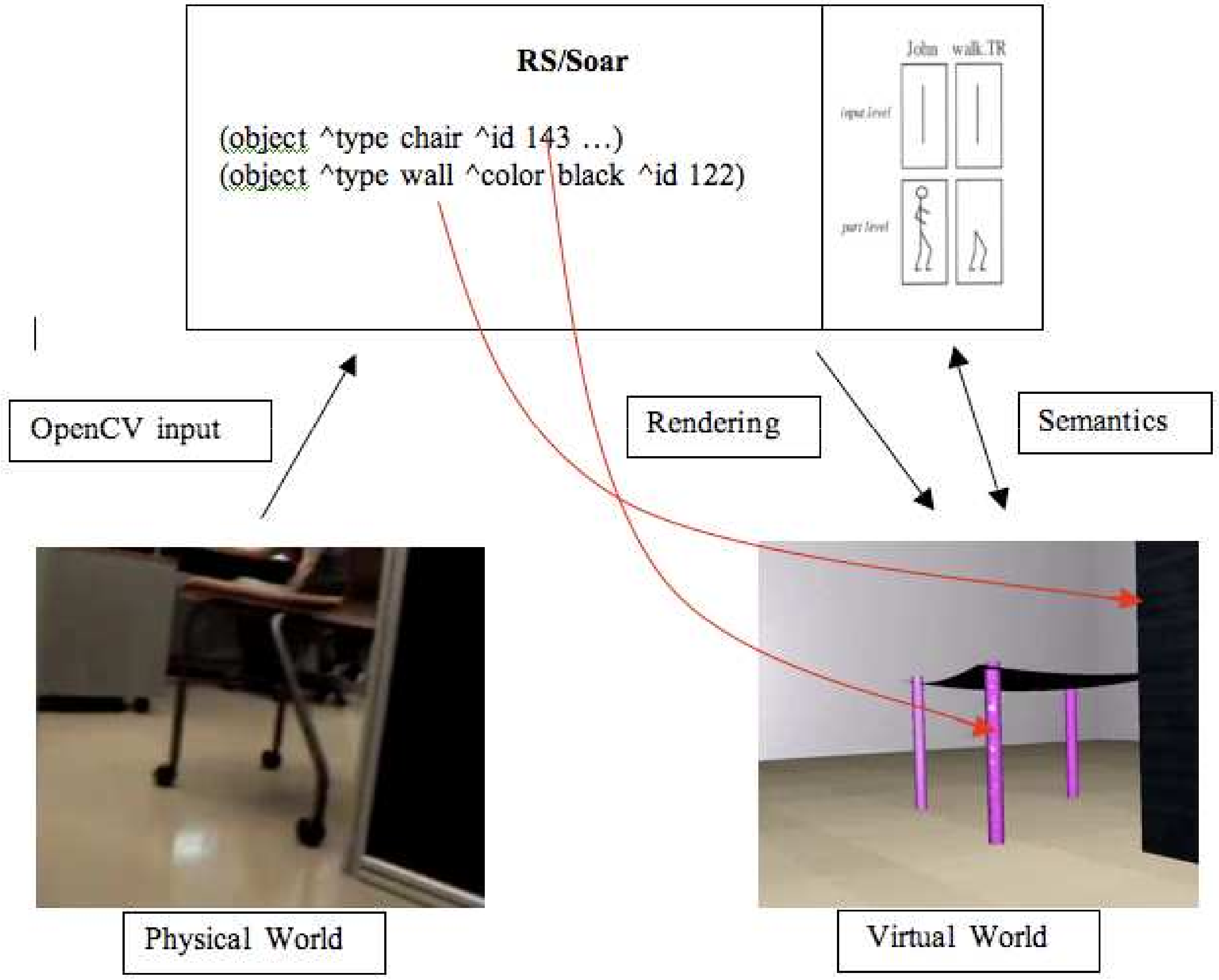
ADAPT's Mental Model

Ogre3D Video game platform: physics and graphics



Goal: to create a working copy of the environment

ADAPT’s Mental Model



```
(object ^type chair ^id 143 …)
(object ^type wall ^color black ^id 122)
```

Videos showing the Modeling Process

QuickTime™ and a
H.263 decompressor
are needed to see this picture.

QuickTime™ and a
YUV420 codec decompressor
are needed to see this picture.

Virtual World

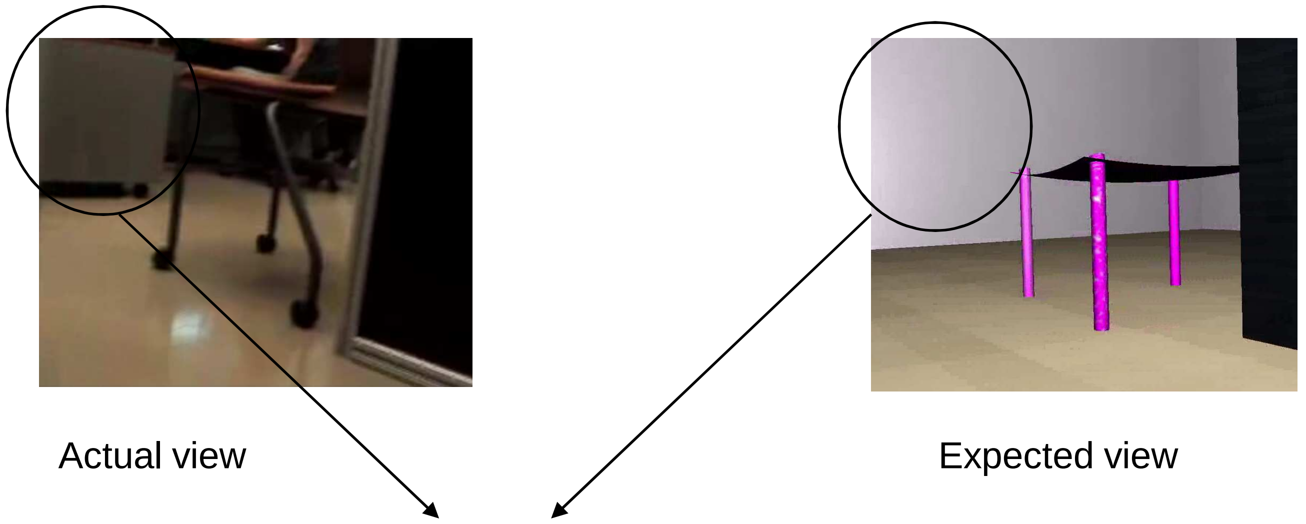
Reconstruction currently uses a hand-built library of objects and schemas.

Ogre is used for:

- motion planning in dynamic environments
- predictive vision
- language comprehension.

ADAPT searches to reconstruct the environment. This approach is used for recall, instead of data chunking.

Predictive Vision

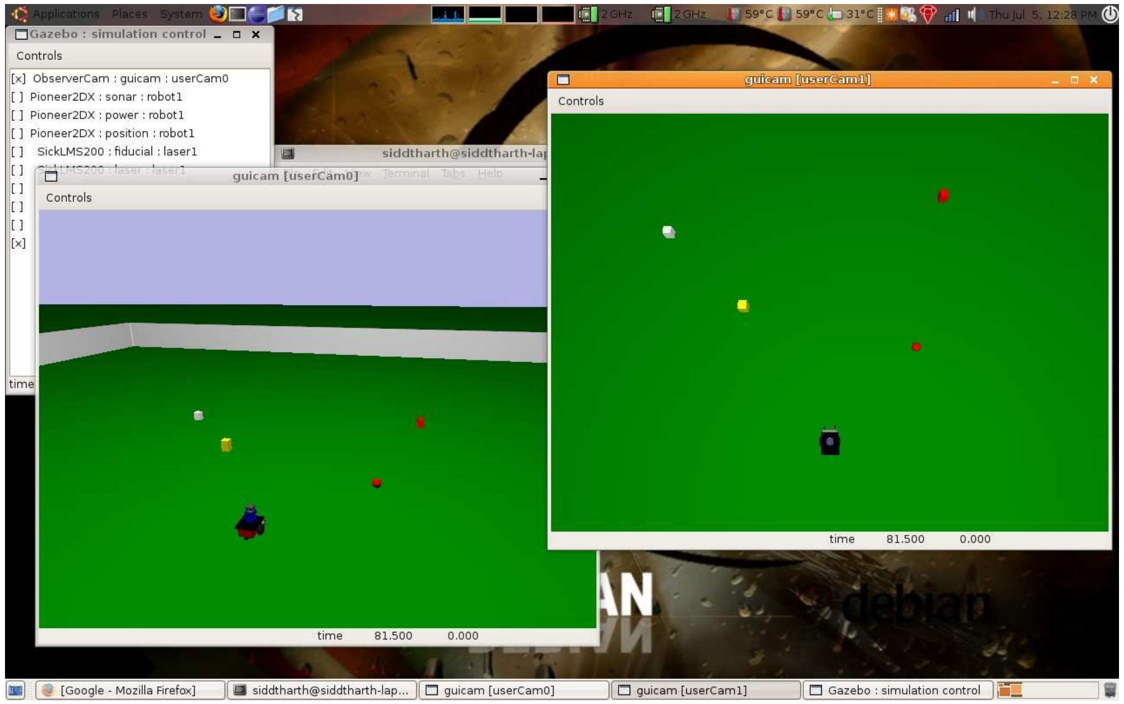


Sum-of-squares difference exceeds threshhold.
Soar operator proposed to focus on difference.

Expensive vision operations are goal-dependent,
greatly reducing their frequency of use.

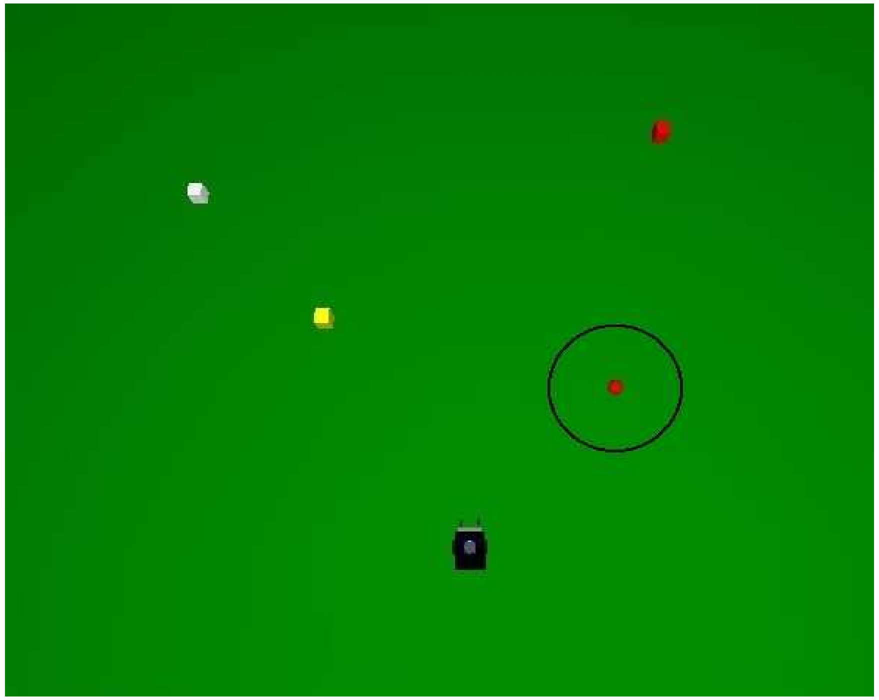
Visualization for Semantics

Comprehension requires visualization.


Gazebo : simulation control
Controls
[x] ObserverCam : guicam : userCam0
[ ] Pioneer2DX : sonar : robot1
[ ] Pioneer2DX : power : robot1
[ ] Pioneer2DX : position : robot1
[ ] SickLMS200 : fiducial : laser1
guicam [userCam0]
guicam [userCam1]
time 81.500 0.000
[Google - Mozilla Firefox]
guicam [userCam0]
guicam [userCam1]
Gazebo : simulation control

Observer camera   Neighborhood camera

Visual Context

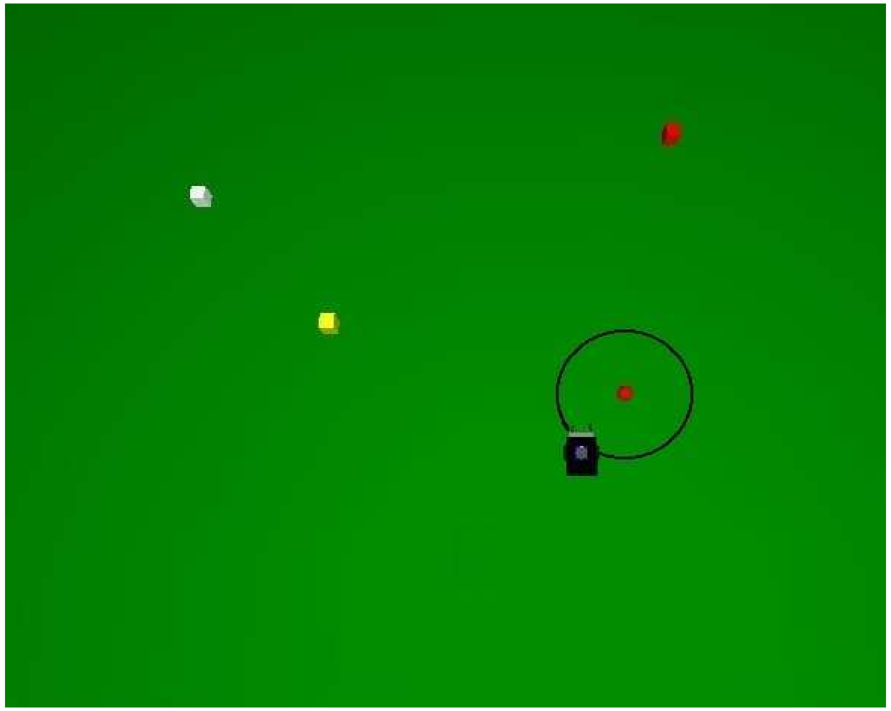


The Neighborhood camera moves with the robot and defines the visual context.

The circle defines “near”; it is movable But of a fixed size.

If the robot is told to move near the small red block, it will plan motions to take it inside the black circle.

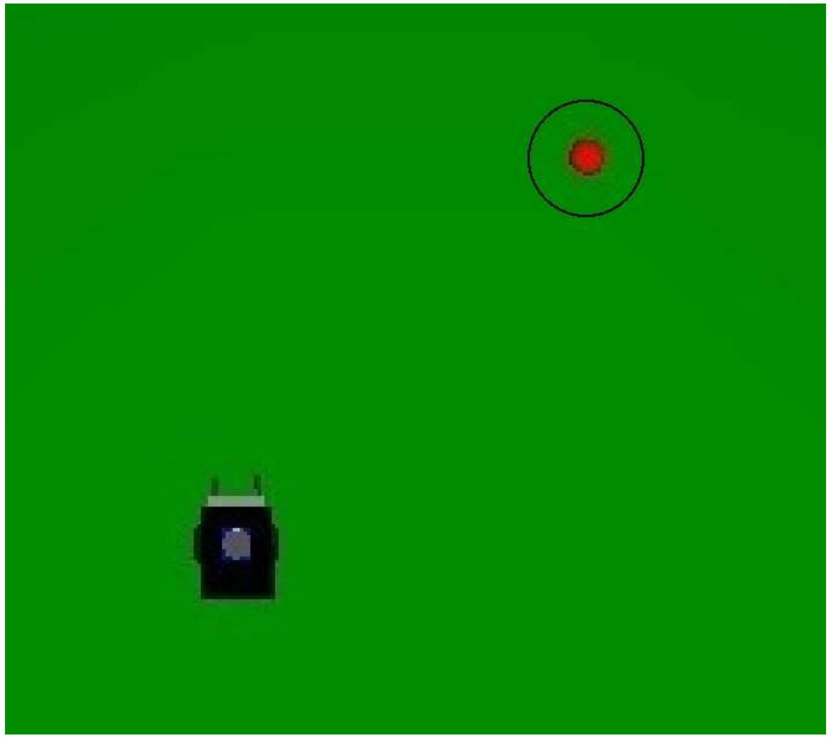
The Visual Context is Associated with the Task



If the robot is then told to pick up the small red block, the new task changes the context.

The new context is smaller, causing the camera to zoom in.

Changing the Visual Context

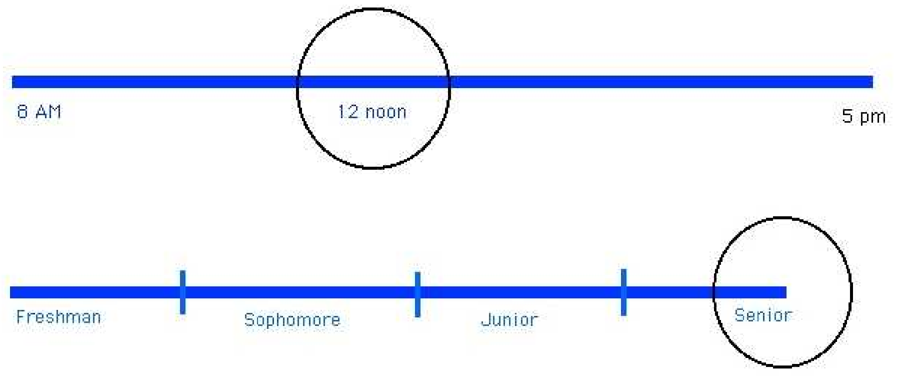


In the new context, the robot is no longer near the small red block, because although "near" is visually the same neighborhood, it denotes a much smaller region.

If the robot is told to pick up the small red block, it must move inside the black circle, because picking up something requires being closer than moving near it.

Tasks Determine Contexts

Visualization uses the virtual camera to define a context within which terms have a single meaning. This is intended to fit NL-Soar's comprehension mechanism.

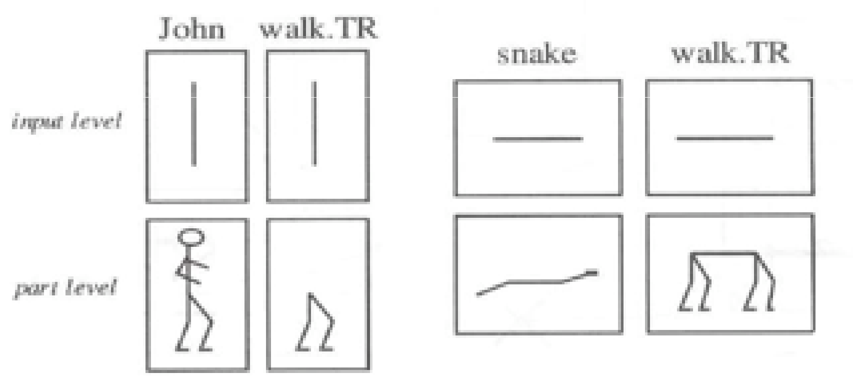


ADAPT must search among contexts instead of among meanings for terms.

This approach could conceivably be used for terms that are not inherently physically grounded.

Cognitive Semantics

The idea of using visualization for linguistic semantics is not new. Langacker's Cognitive Grammar described a way to do this.



Holmqvist partially implemented this grammar, but didn't finish.

Our approach improves the methodology because the virtual world is 3D and dynamic.

Cognitive Plausibility and Soar Theory

Duncker: “Problem solving consists of a sequence of phases; each phase is a reformulation of the problem (Newell).”

Polk & Newell: “We propose that the central processes in deductive reasoning are linguistic (encoding, reencoding, and generation) rather than reasoning-specific skills.”

“… for deduction tasks for which the necessary information is provided verbally, the heart of deduction involves repeatedly reencoding the problem, …”

Summary

ADAPT addresses necessary choices for using Soar in robotics:

Providing computational facilities for concurrency and distributed control: RS

An methodology for perception: Reformulation based on linguistic reencoding

A fast method of reasoning and sensory fusion: Coherence theory

Representing comprehension as search: Gaming platform as virtual world

Status

Individual capabilities have been demonstrated: NUGGETS!

- RS in Soar for basic navigation
- Coherence theory in navigation, cybersecurity
- Modeling vision data in Ogre in real time
- Use of visual contexts for semantics in navigation
- Predictive vision attends to and models changes

But: ADAPT is still in three big pieces COAL

- Everything is hand built
- RS/Soar, vision, NL-Soar: what are the constraints?
- A big problem is the virtual world software

Thanks!

References

ADAPT:

Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, “Embodying a Cognitive Model in a Mobile Robot”, Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, Boston, October, 2006.

Benjamin, D. Paul, Damian Lyons and Thomas Achtemichuk, “Obstacle Avoidance using Predictive Vision based on a Dynamic 3D World Model”, Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, Boston, October, 2006.

Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, "Designing a Robot Cognitive Architecture with Concurrency and Active Perception", Proceedings of the AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics, Washington, D.C., October, 2004.

RS:

Lyons, D.M. and Hendriks, A., “Exploiting Patterns of Interaction to Select Reactions”, Special Issue on Computational Theories of Interaction, Artificial Intelligence **73**, 1995, pp.117-148.

Lyons, D.M. and Arbib, M.A., “A Formal Model of Computation for Sensory-based Robotics”, IEEE Transactions on Robotics and Automation **5**(3), Jun. 1989.

Lyons, D., and Arkin, R.C., “Towards Performance Guarantees for Emergent Behavior”, (Submitted) IEEE Int. Conf. on Robotics and Automation, New Orleans LA, April 2004.

References

Port Automata:

Martha Steenstrup, Michael A. Arbib, Ernest G. Manes, *Port Automata and the Algebra of Concurrent Processes*. JCSS 27(1): 29-50, 1983.

Cognitive Semantics:

Holmqvist, K., “Implementing Cognitive Semantics”, Lund: Department of Cognitive Science, 1993.

Holmqvist, K., “Conceptual Engineering”, in Cognitive Semantics: Meaning and Cognition, Allwood and Gardenfors (Eds.), John Benjamins, pp.153-171, 1999.

Langacker, R., “Foundations of Cognitive Grammar, Vol. I”, Stanford University Press, 1987.

Langacker, R., “Foundations of Cognitive Grammar, Vol. II”, Stanford University Press, 1991.

Langacker, R., “Concept, Image, and Symbol”, Berlin, New York, Mouton de Gruyter, 1991.

References

Reformulation:

Benjamin, D. Paul, “On the Emergence of Intelligent Global Behaviors from Simple Local Actions”, Journal of Systems Science, special issue: Emergent Properties of Complex Systems, Vol. 31, No. 7, 2000, 861-872.

Benjamin, D. Paul, “Formulating Patterns in Problem Solving”, *Annals of Mathematics and AI*, **10**, pp.1-23, 1994.

Benjamin, D. Paul, “Reformulating Path Planning Problems by Task-preserving Abstraction”, *Journal of Robotics and Autonomous Systems*, **9**, pp.1-9, 1992.