

# Artificial General Intelligence

„General intelligence doesn't comprise one single, brilliant knock-out invention or design feature; instead, it emerges from the synergetic integration of a number of essential fundamental components.” (Peter Voss)

# Artificial General Intelligence

„Of all the people working in the field called 'AI', 80% don't believe in the concept of General Intelligence (but instead, in a large collection of specific skills and knowledge)

Of those that do, 80% don't believe that artificial, human-level intelligence is possible - either ever, or for a long, long time

Of those that do, 80% work on domain-specific AI projects for commercial or academic-political reasons (results are more immediate)

Of those left, 80% have a poor conceptual framework...” (half-seriously, Peter Voss)

# Artificial General Intelligence

„Only a small community has concentrated on general intelligence. No one has tried to make a thinking machine. The bottom line is that we really haven't progressed too far toward a truly intelligent machine. We have collections of dumb specialists in small domains; the true majesty of general intelligence still awaits our attack. We have got to get back to the deepest questions of AI and general intelligence and quit wasting time on little projects that don't contribute to the main goal.”

(Marvin Minsky, 2000)

# General Intelligence definitions

- „the ability to solve complex goals in complex environments” (Ben Goertzel)
- „the capability to adapt to the environment and to work with insufficient knowledge and resources” (Pei Wang)
- „the ability to acquire (and adapt) the knowledge and skills required for achieving a wide range of goals in a variety of domains”, „a property of an entity that engages in two way interaction with an external environment” (Peter Voss)

# Universal Algorithmic Agent AIXI

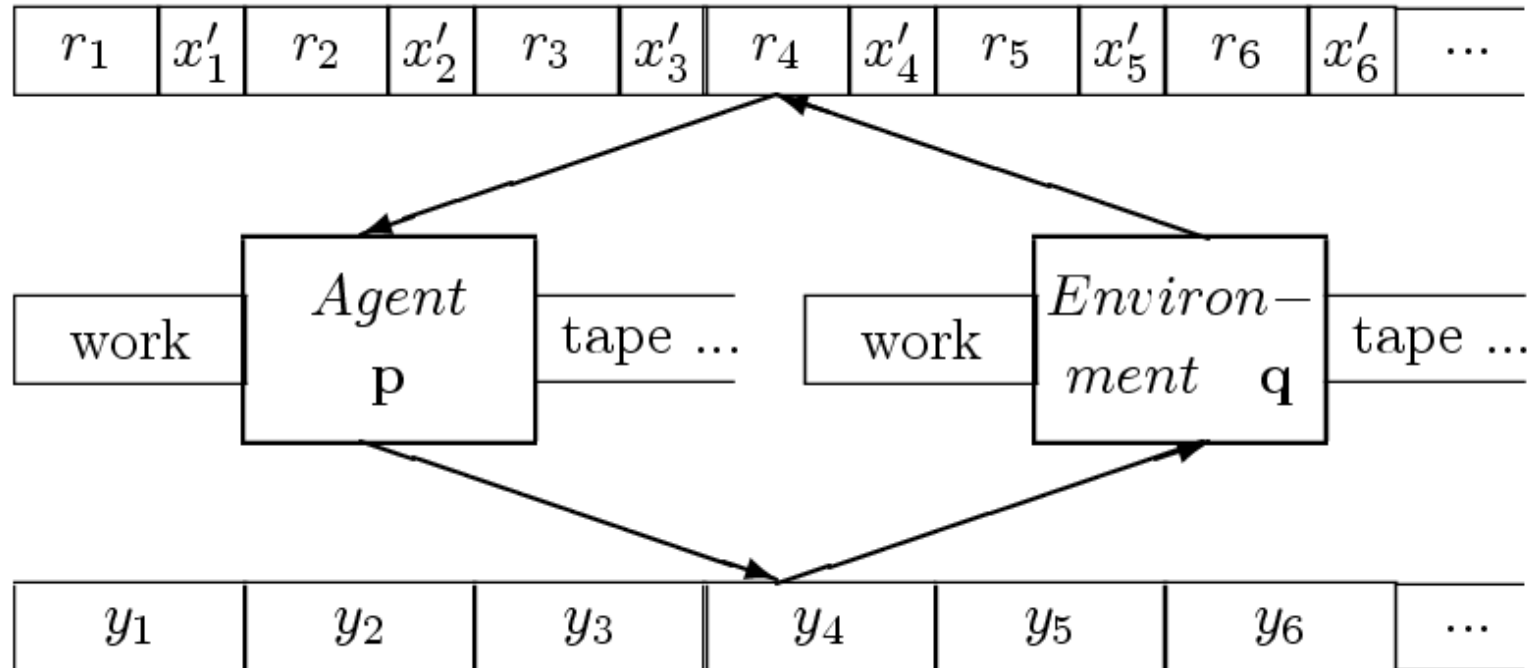
- Parameter-free theory of universal Artificial Intelligence based on ideas from decision theory (known priors) and Solomonoff's universal induction (unknown priors).
- Strong arguments that the AIXI model is the most intelligent unbiased agent possible (but it is uncomputable).
- $AIXI_{t,l}$  is more intelligent than any time  $t$  and space  $l$  bounded agent. Its time complexity is  $t \cdot 2^l$ .

# AIXI

- replace the unknown environmental distribution  $\mu$  in Bellman equations by suitably generalized Solomonoff distribution  $\xi$  (state space is the space of complete histories);  $AIXI = AI\xi$
- (Bellman equations are solved iteratively by integrating current solution wrt. env. distribution)

# Chronological Turing Machines

- Each (agent, environment) pair (p, q) produces a unique I/O sequence  $y_1 x_1 y_2 x_2 \dots x_i = x'_i r_i$ , where  $r_i$  is the reward
- but q is unknown or stochastic, known to agent by distribution  $\mu(q)$



# AIXI

Let  $V_{km}^{pq} := \sum_{i=k}^m r(x_i)$  be the future total reward (called future utility)

The *best* or *most intelligent* agent is now the one which maximizes the *expected* utility (called value function)  $V_\mu^p \equiv V_{1m}^{p\mu} := \sum_q \mu(q) V_{1m}^{pq}$ . This defines the  $AI_\mu$  model.

**Definition 2 (The  $AI_\mu$  model)** *The  $AI_\mu$  model is the agent with policy  $p^\mu$  which maximizes the  $\mu$ -expected total reward  $r_1 + \dots + r_m$ , i.e.  $p^* \equiv p^\mu := \operatorname{argmax}_p V_\mu^p$ .  $V_\mu^* := V_\mu^{p^\mu}$ .*

Define universal prior:

$\xi(\underline{x})$  as the probability that the output of a universal Turing machine  $U$  starts with  $x$  when provided with fair coin flips on the input tape. Formally,  $\xi$  can be defined as

$$\xi(\underline{x}) := \sum_{p: U(p)=x^*} 2^{-l(p)} \geq 2^{-K(x)} \quad (17)$$

$$\sum_{t=1}^{\infty} \sum_{x_{<t}} \mu(\underline{x}_{<t}) \left( \xi(x_{<t}0) - \mu(x_{<t}0) \right)^2 \stackrel{+}{\leq} \frac{1}{2} \ln 2 \cdot K(\mu),$$

$$V_{km}^{p\xi}(\dot{\underline{x}}_{<k}) := \frac{1}{\mathcal{N}} \sum_{\underline{y}_{k:m_k}} 2^{-l(q)} \cdot V_{km}^{\tilde{p}q}$$

Replacing  $\mu$  by  $\xi$  in (12) the iterative  $AI_\xi$  agent outputs

$$\dot{y}_k \equiv \dot{y}_k^\xi := \operatorname{argmax}_{y_k} \sum_{x_k} \max_{y_{k+1}} \sum_{x_{k+1}} \dots \max_{y_{m_k}} \sum_{x_{m_k}} (r(x_k) + \dots + r(x_{m_k})) \cdot \xi(\dot{\underline{x}}_{<k} \underline{y}_{k:m_k}) \quad (23)$$

in cycle  $k$  given the history  $\dot{\underline{x}}_{<k}$ .



# SNePS

- Integrates:
  - intensional relevance logics for commonsense reasoning
  - frame-based system (feature-structure subtyping)
  - semantic network: frame slots are labeled directed arcs (recursive path constructors)
- Designed to support natural language competent agents
- The „domain of discourse” is the domain of all mental entities of the agent
- Propositional assertional: only nodes have semantics (arcs don't denote propositions)

# SNePS


**Propositional Semantic Network:** The only well-formed SNePS expressions are nodes.

**Term Logic:** Every well-formed SNePS expression is a term.

**Intensional Representation:** Every SNePS term represents (denotes) an intensional (mental) entity.

**Uniqueness Principle:** No two SNePS terms denote the same entity.

**Paraconsistent Logic:** A contradiction does not imply anything whatsoever.

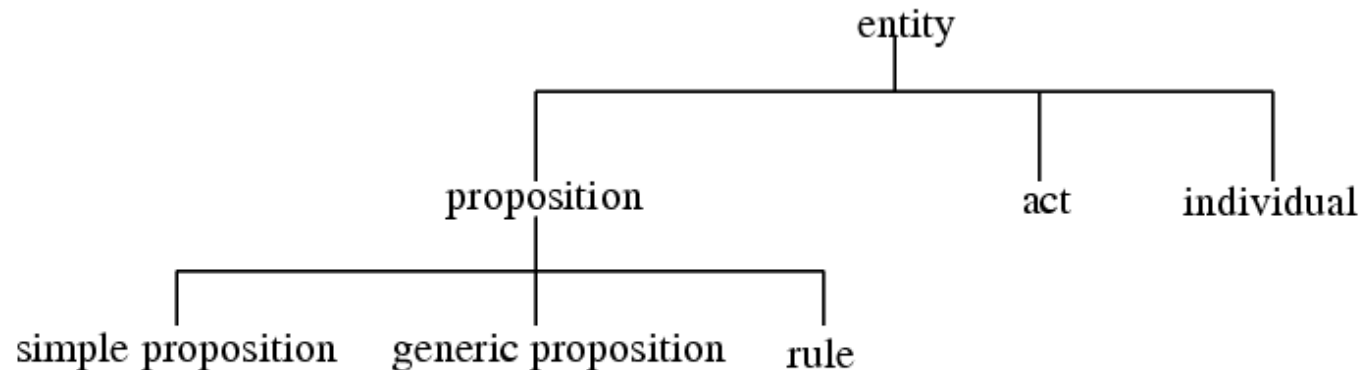


proposition-denoting terms may be arguments of other terms without leaving first order logic

**Monotonic Logic:** belief revision (removing a contradiction) must retract some hypotheses and all beliefs supported by them. (But retracted facts can be reconsidered when new knowledge arrives.)

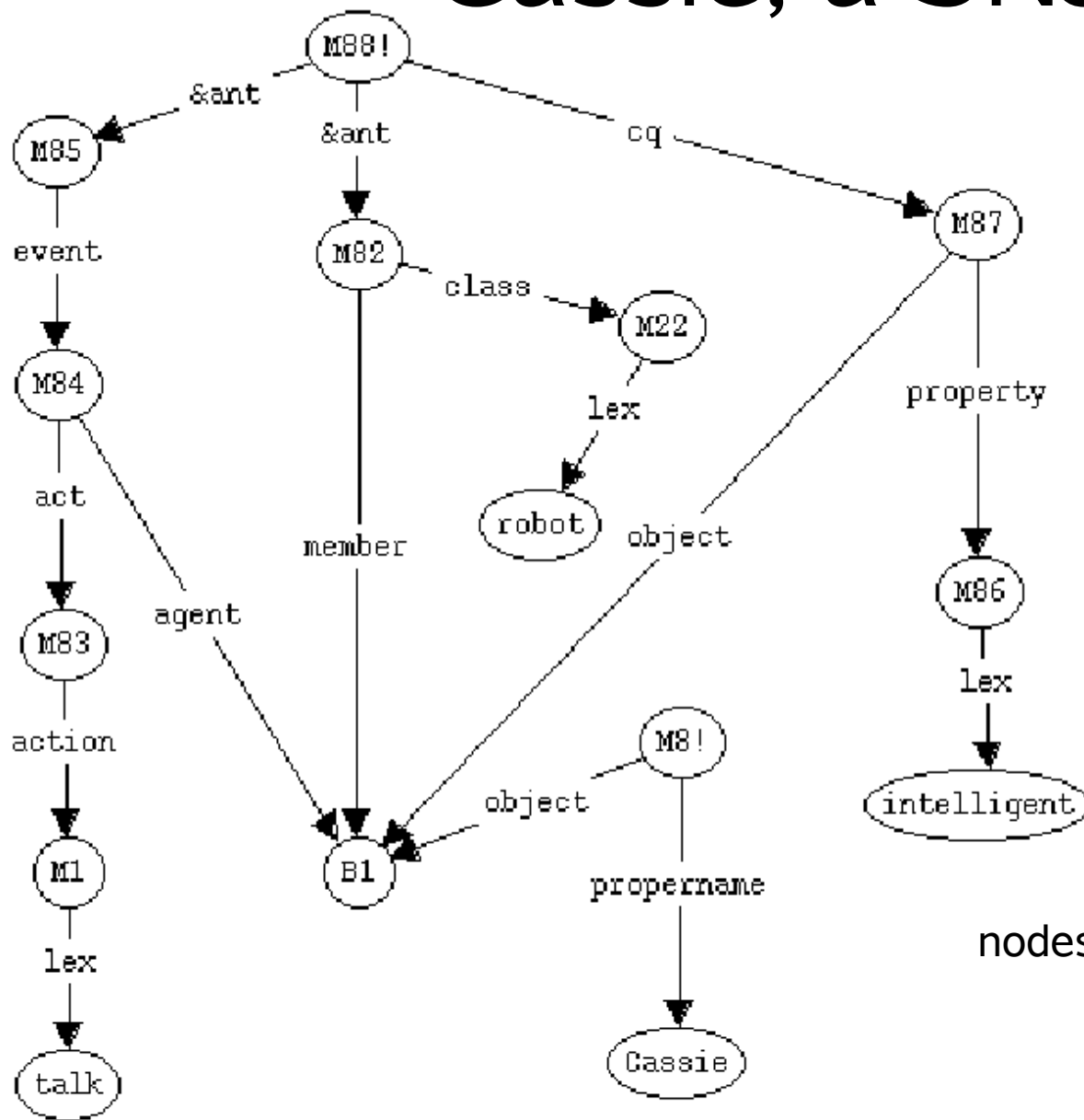
# SNePS

- Not interested in representing the „meaning“, rather the changes to Cassie's mind that result from her understanding.
- Every SNePS term denotes a mental entity. Even „variable nodes“ have compositional semantics.
- *Proposition* nodes have assertional status, *rule* nodes can be used for inference, *act* nodes can be performed, *individual* (or *thing*) nodes are „the rest“.



**Fig. 4.** The initial taxonomy of SNePS semantic classes.

# Cassie, a SNePS Agent



nodes with ! are asserted

**Fig. 5.** M88! is a rule node denoting the proposition, *If Cassie is a robot that talks, then Cassie is intelligent.*

# SNePS Syntax

- Nodes, relations (arcs), case frames (feature-structure-like types for atomic nodes)
- user can specify any case frames for atomic nodes
- Relation:  $\langle name, type, adjust, limit \rangle$ 
  - *name*: symbol identifying relation (given on arcs)
  - *type*: class of nodes pointed by the arc
  - *adjust*: *expand*, *reduce*, *none*, for wire-based inference
  - *limit*: minimal size of a cable containing this relation
- examples:

$\langle member, entity, reduce, 1 \rangle$

$\langle event, event, reduce, 1 \rangle$

$\langle \&ant, proposition, expand, 1 \rangle$

$\langle class, category, reduce, 1 \rangle$

$\langle time, time, reduce, 0 \rangle$

$\langle cq, proposition, reduce, 1 \rangle$

# SNePS Inference

- **Wire-based inference** (reduction inference): introducing a node with a subset (or superset) of arcs of an existing node
- **Path-based inference:** (perhaps undirected) path from  $m_1$  to  $m_2$  implies a proposition  $m_3$  with all  $m_1$ 's arcs plus arc to  $m_2$
- **Node-based inference:** uses nodes representing FOL formulas
- **Subsumption inference:** introduces an instantiation for a variable node, which connects to nodes subsumed by variable node neighbors

# SNePS Logic

- $andor(i, j)\{P1, ..., Pn\}$ : at least  $i$  and at most  $j$  of  $Pk$  are true; e.g.  $andor(1, 1)$  is a disjoint alternative
  - $all(x)(andor(1, 3)\{animal(x), vegetable(x), mineral(x)\})$
- $thresh(i, j)\{P1, ..., Pn\}$ : fewer than  $i$  or more than  $j$  of  $Pk$  are true
  - $all(x)(thresh(1, 2)\{human(x), featherless-biped(x), rational-animal(x)\})$
- $\{P1, ..., Pn\} v=> \{Q1, ..., Qm\}$  (or-entailment): for every  $i, j, Pi => Qj$  (people don't use or-introduction)
  - $\{in(Hilda, Boston), in(Cathy, Las-Vegas)\} v=> \{in(Eve, Providence)\}$
- $\{P1, ..., Pn\} \&=> \{Q1, ..., Qm\} := P1\&...\&Pn => Q1\&...\&Qn$

# SNePS Logic

- Unique variable binding rule: universal instantiation can't replace two variables by the same term in one formula.
 
$$\frac{P(s_1, \dots, s_{i-1}, \tau, s_{i+1}, \dots, s_m), \tau' \subset \tau}{P(s_1, \dots, s_{i-1}, \tau', s_{i+1}, \dots, s_m)}$$
- Set arguments:
 
$$\frac{P(s_1, \dots, s_{i-1}, \tau, s_{i+1}, \dots, s_m), t \in \tau}{P(s_1, \dots, s_{i-1}, t, s_{i+1}, \dots, s_m)}$$
  - Marry, Sue and Sally are sisters.
- Higher-order user language: e.g. *Transitive(bigger)*
  - *bigger (elephant, lion) := Holds (bigger, elephant, lion)*
  - *all(p)(Believes(Bob, p) => p).* (proposition denoting terms)
- *nexists(i, j, k)(x)({P1(x), ..., Pn(x)} : {Q(x)}): k*  
 individuals satisfy *P1&...&Pn* and at least *i*, at most *j*, also satisfy *Q*.
  - „At least two members of the committee are women.”  
*nexists(2,\_,4)(x)({Member(x)} : {Woman(x)})*



# SNePS Logic

- Contexts are sets of assumptions. Assertions only hold in default context (change between contexts).
- Belief revision: if system detects a contradiction, it can ask the user if to keep it or change assumptions.
  - When retracting a hypothesis, the system retracts assertions that no longer hold.
- SNePS can infer relevant implications, handle recursive definitions, etc.
- SNePS performs bidirectional inference: forward and backward chaining.
- Relevance:  $P \Rightarrow Q$  means „if I believe P, I'm justified to believe Q” („I will believe Q when the rule fires”)
  - $P \Rightarrow Q$  is a function from propositions to propositions

# Cassie, GLAIR and SNePS Metacognition

Grounded Layered  
Architecture with  
Integrated Reasoning

- acting subsystem:
  - acts that affect what an agent believes
  - acts that specify knowledge-contingent acts lack-of-knowledge acts
  - policies that serve as “daemons”, triggering acts when certain propositions are believed or wondered about
- a *policy*: a rule that connects propositions and acts
- *action*: act-valued function symbol
- policy-forming function symbols:
  - *ifdo*( $p, a$ ): to determine whether  $p$ , do  $a$
  - *whendo*( $p, a$ ), *wheneverdo*( $p, a$ ): when(ever)  $p$ , perform  $a$

# SNePS Cassie

- external acts either sense or affect the outside world
- mental acts:
  - *believe(p)*: assert  $p$  and do forward inference (and some belief revision); *disbelieve(p)* – just unassert  $p$
  - *adopt(p)*, *unadopt(p)* – whether to follow policy  $p$
- control acts:
  - *achieve(p)*: (when  $p$  unasserted) infer plans (instances of *GoalPlan(p,x)*) to bring about  $p$ , perform *do-one* on them
  - *prdo-one* performs an action selected by roulette-wheel
  - *withall(x, p(x), a(x), [d])* finds entities  $e$  such that  $p(e)$  is believed, and performs  $a$  on them; if no such  $e$  is found,  $d$  is performed.
  - *snif* (switch on condition), *sniterate* (switch and loop)

# SNePS Metacognition

- „Self” is a term like other agents' terms
- Perceptuo-Motor Layer models embodiedness: a source of beliefs about what an agent is doing and percepts
- deictic registers: I, YOU, NOW, ...
- modality registers: current acts and percepts in each effector and affector; used to advance deictic regs
- retracted (unasserted) belief is kept in the system and can still be reasoned about
- metabeliefs can represent credibility (uncertainty etc.); the least credible facts are retracted on revision
- dependency-directed reconsideration, e.g. when we learn that a source is/was not credible

# Cognitive Architectures

## Soar and ACT-R

- Available for download
- Heavily documented
- With long history and many applications
- Based on cognitive psychology insights
- Related psychological research
- *cognitive architecture* = a theory about the fixed computational structure of cognition

# Soar

- Descendant of General Problem Solver
- by Allen Newell
- **...all problem solving activity is formulated as the selection and application of operators to a state, to achieve some goal.**
- Since 1982, initially in Lisp then rewritten to C and Tcl.
- Basic knowledge: state and operators
- Control knowledge: heuristics
- Knowledge can be learned

# Soar knowledge and action

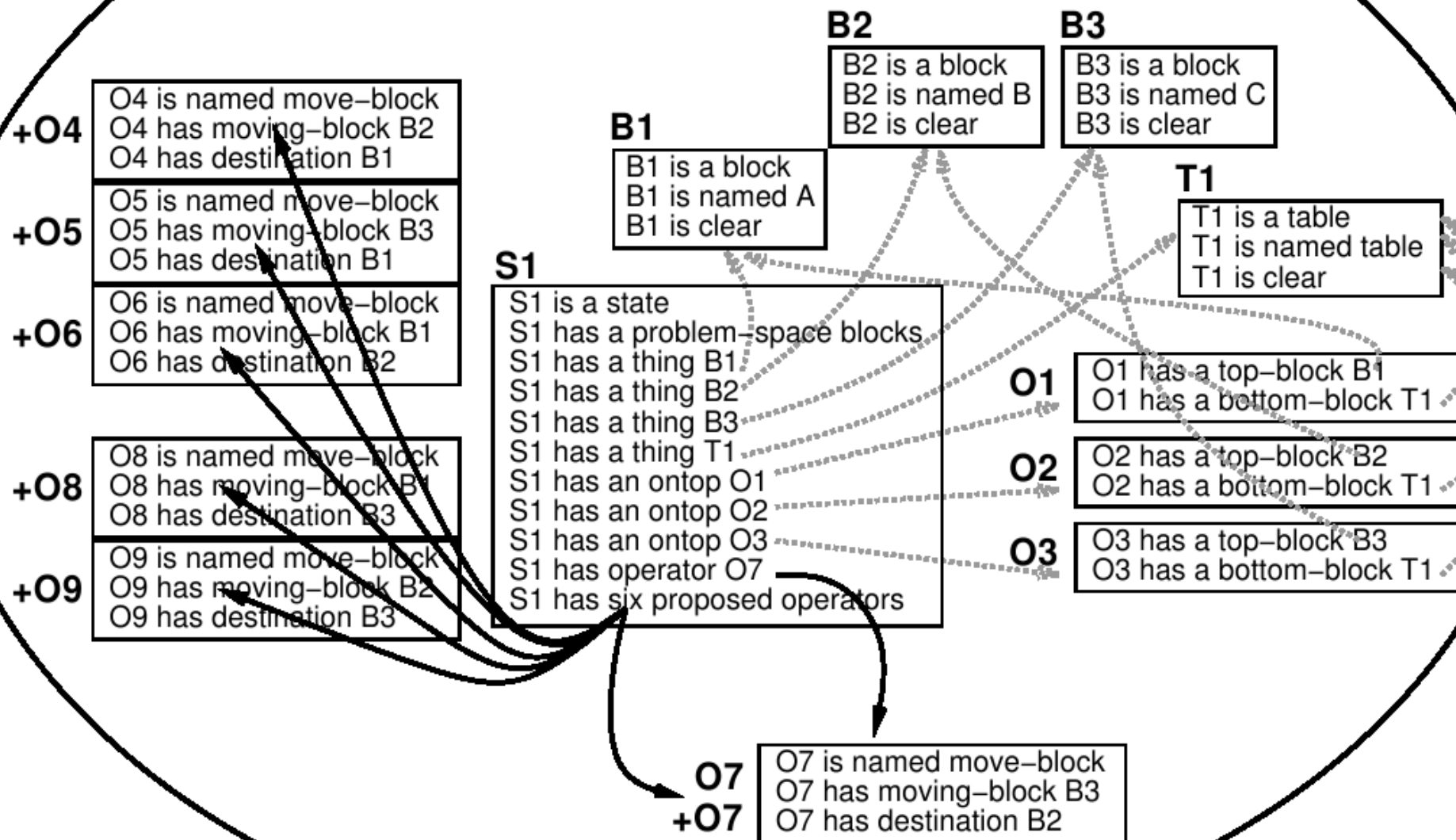
- *production rules*: conditions --> actions. Conditions test for patterns in *working memory*.
- productions = associative long term memory
- cognitive loop of alternating operator *selection* and *application* both done by productions
- can have a stack *contexts* (problem spaces) active at once
- *Impasse*: no operator applies in the active context (no-change) or no unique one can be determined (tie).
- Knowledge: operator proposal, comparison, selection, application; state elaboration

# Soar knowledge and action

- States are objects with feature structures (attribute-value matrices) (working memory = set of objects)
- All productions that match WM, apply in parallel, rewriting the working memory
- Goals are desirable patterns in states
- Rules vote for changes by preferences which are stored in *preference memory*
- *Elaboration rules* monotonically add facts to WM, are backtracked when no longer supported
- Actions (operator application rules) are persistent
- *Decision cycle*: apply elaboration rules until fixpoint, select operator based on preference memory (if not possible: impasse), apply action rules



*(links from operators to blocks  
are omitted for simplicity)*



**An Abstract View of Working Memory**

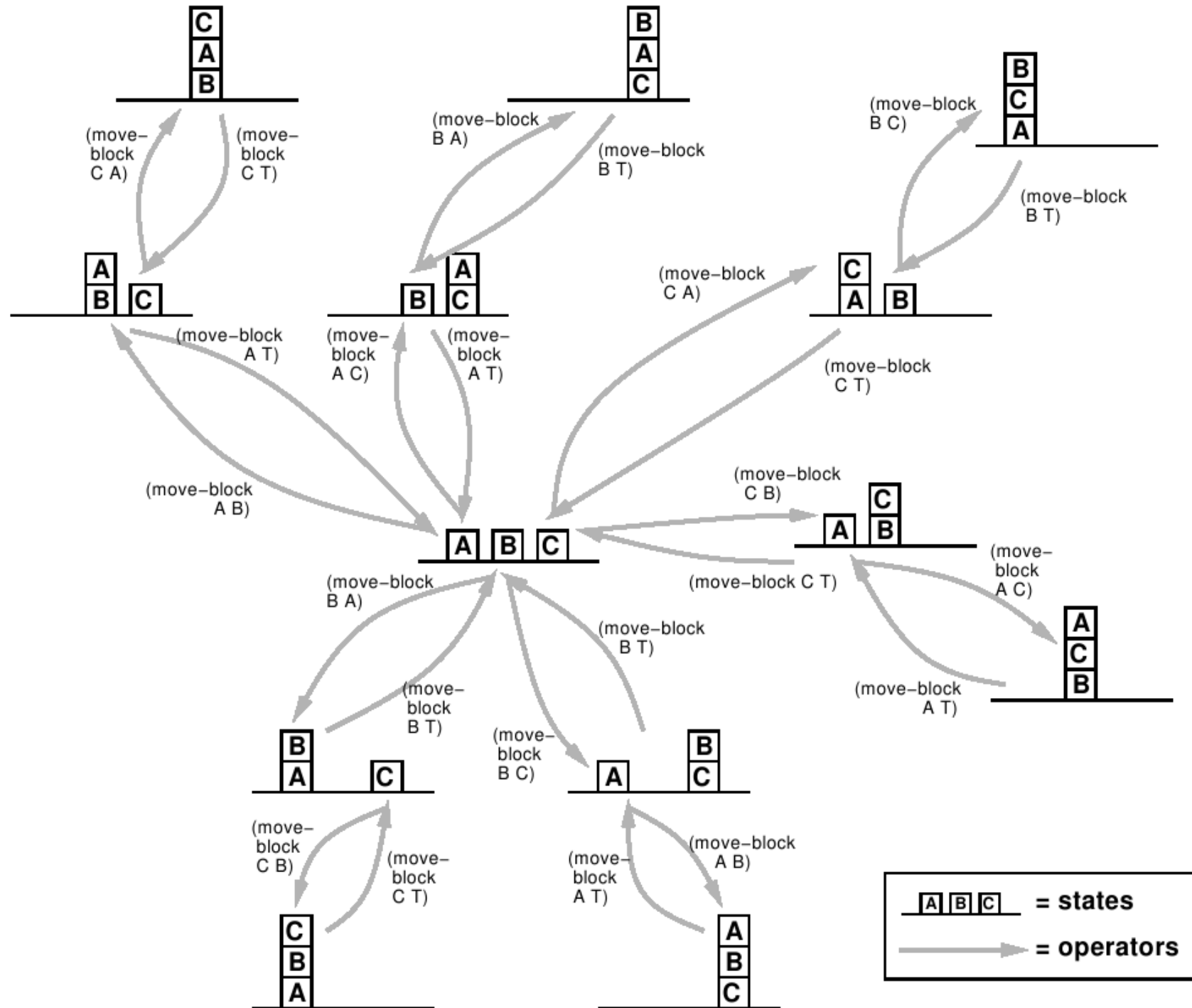


Figure 2.6: The problem space in the blocks-world includes all operators that move blocks from one location to another and all possible configurations of the three blocks.

# Soar learning

- When in impasse, record it in WM, create a new context, which generates a chunk: new production. (This mechanism is recursive.)
- States in new context are called substates.
- The RHS is the result of new context. The LHS are things that have been tested by the linked chain of rule firings leading to the result, the set of things that exist in the higher context (“pre-impasse”) on which the result depends.
- Problems:
  - overgeneralization: e.g. if result dependent on search control knowledge (solution: request condition explicitly)
  - overspecialization: e.g. chunk variable identifies objects realized by the same element in a particular impasse

# Soar Goal Dependency Set

goal is a synonym for state or substate

- to solve symbol-level quirks of Soar „psychology”: problems
  - logical inconsistency in symbol manipulations,
  - non-contemporaneous constraints in chunks,
  - race conditions in rule firings and in the decision process,
  - contention between original task knowledge and learned knowledge
- follow from inconsistency between persistent WM elements and their context (all superstates of a state)

# Soar Goal Dependency Set

- three primary types of persistence (in Soar 7):
  - i-support: feature exists in memory only as long as its creator production remains instantiated; instantiation is retracted when one of production conditions no longer matches
  - o-support: created by action of operator, remains until explicitly removed
  - c-support: (removed in Soar 8) makes an operator persistent (only retracted explicitly)
- solution inspired by chunking: when o-supported WME is created, the superstate dependencies of that feature are added to GDS of that state

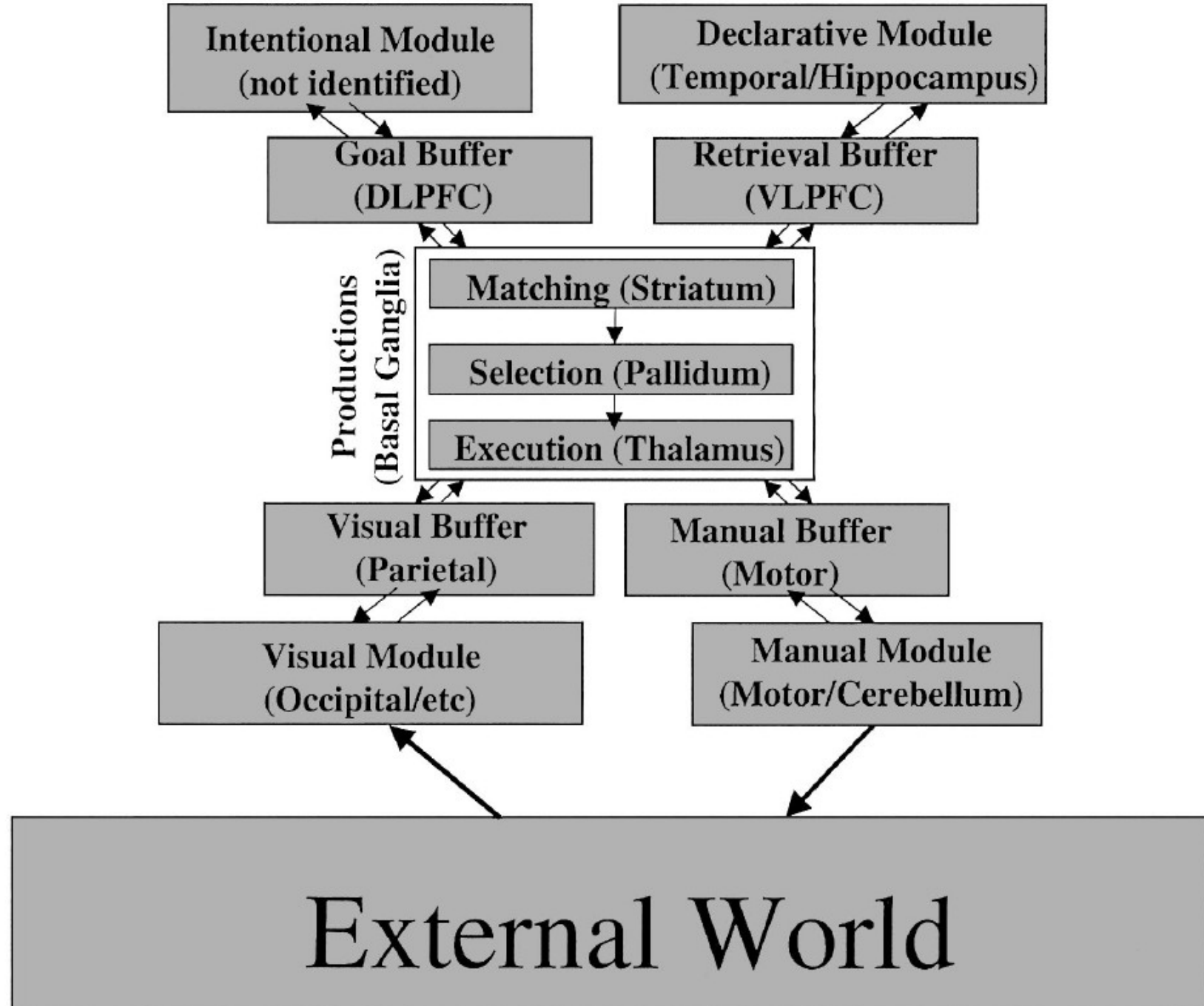
# Soar Goal Dependency Set

- Elements added to GDS for an o-supported feature:
  - elements (WMEs) in a superstate on which it depends
  - WMEs in a superstate supporting i-supported features on which it depends
- In Soar 8, any change to the current dependency set will cause the retraction of all subgoal structure.
- *Remembering* facts should be stored in the top state, *non-monotonic reasoning* about context should be done locally (will be retracted on relevant context change).

# ACT-R

## Adaptive Control of Thought-Rational

- parallel processing local to modules, sequential processing by productions
- modules are interfaced by buffers, productions match and change buffer contents
- a buffer can contain only one chunk (object = a named feature structure) at a time
- subsymbolic processes guide the selection of rules to fire
- the goal buffer keeps state of solving a problem,
- retrieval buffer holds information retrieved from long-term declarative memory, etc.



*Figure 1.* The organization of information in ACT-R 5.0. Information in the buffers associated with modules is responded to and changed by production rules. DLPFC = dorsolateral prefrontal cortex; VLPFC = ventrolateral prefrontal cortex.



# ACT-R

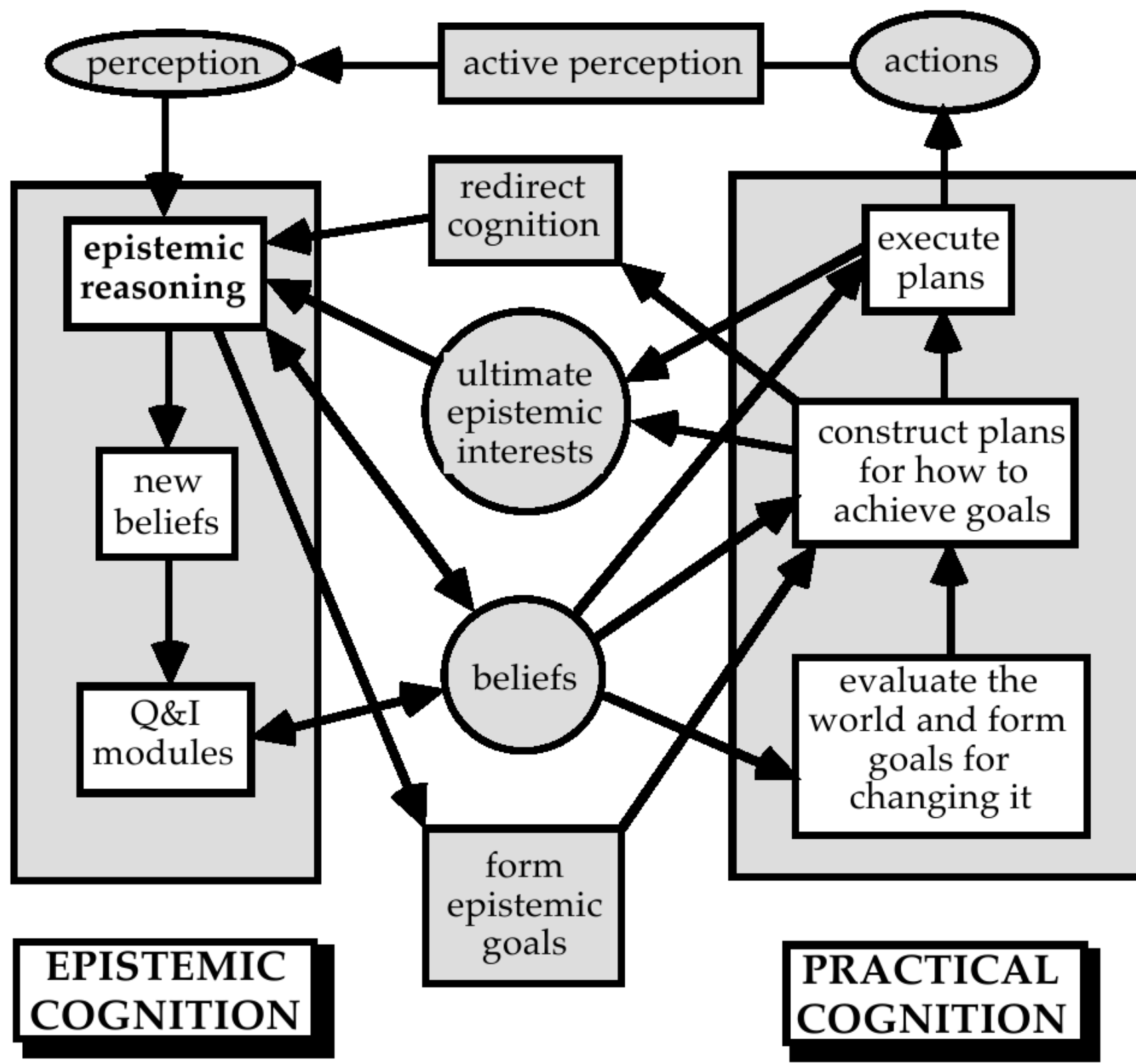
- A module can access other buffers than its own, but usually doesn't
- Slots in a chunk are usually filled by other chunks
- Chunks are typed (with inheritance subtyping)
- Chunks are called *declarative*, productions – *procedural*
- Productions specify a set of conditions to match against buffers and the states of the modules (LHS) and a set of actions that will then modify the contents of the buffers and make requests to the modules.

# ACT-R

- conflict resolution: from the productions that match, the one with highest utility is chosen (can be probabilistically); utility from Q-learning
- production compilation: combination of productions that fire in sequence
- declarative module: adding a chunk when there is an equivalent chunk merges them
- retrieval either deterministic or based on activation:
  - $A = B + S + P + e$ ;
    - **B**: base-level activation reflects the recency and frequency of use of the chunk
    - **S**: spreading activation: weighed sum of activations of buffer-residing chunks which contain this chunk
    - **P**: partial matching – the degree to which the chunk matches the specification requested
    - **e**: noise – specific to chunk, and transient for curr. activation

# OSCAR

- individual acts are rational only in context of plans
- *epistemic cognition* – what to believe, *practical cognition* – what to do
- OSCAR expresses most of the latter in terms of the former: practical cognition evaluates the world and then poses queries concerning how to make it better
- epistemic reasoning vs. quick and inflexible modules
- epistemic query can span plans for empirical investigation
- reflexive cognition – applying practical cognition to reasoning e.g. by altering the priority of cognitive tasks waiting to be performed



# OSCAR Epistemic Reasoning

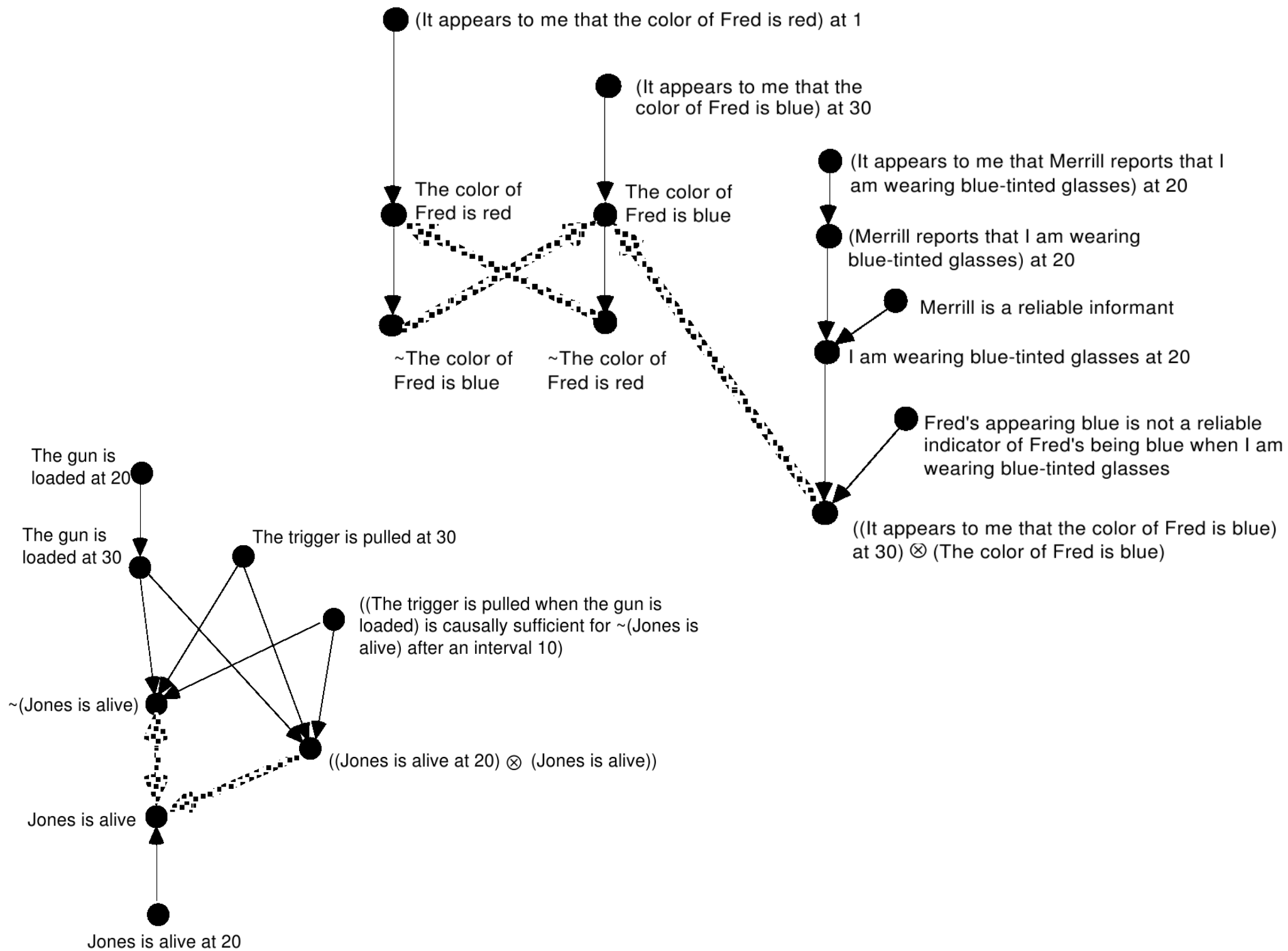
- backwards from epistemic interests to epistemic interests and forward from beliefs to beliefs („natural deduction” theorem prover)
- reductio at absurdum is invalid for defeasible argumentation
- defeasible perception, generalization, time projection, planning
- *Rebutting defeaters* attack the conclusion of the inference. *Undercutting defeaters* attack the connection between the premise and the conclusion.

# OSCAR Epistemic Reasoning

- inference graph records constructed arguments
- status-assignment
  - if a defeating argument for an inference in A is assigned “undefeated”, A is assigned “defeated”;
  - if all defeating arguments for inferences in A are assigned “defeated”, A is assigned “undefeated”.
- an argument is undefeated iff it is “undefeated” in every (maximal) status-assignment; a belief is justified iff it is supported by an undefeated argument (rel. to current epistemological state)
- *Warranted conclusions* are undefeated relative to the set of all possible arguments given the current inputs. Well-behaved reasoner for each (un)warranted proposition P will eventually reach a stage P stays (un)justified thereafter.

# OSCAR Epistemic Reason Schemas

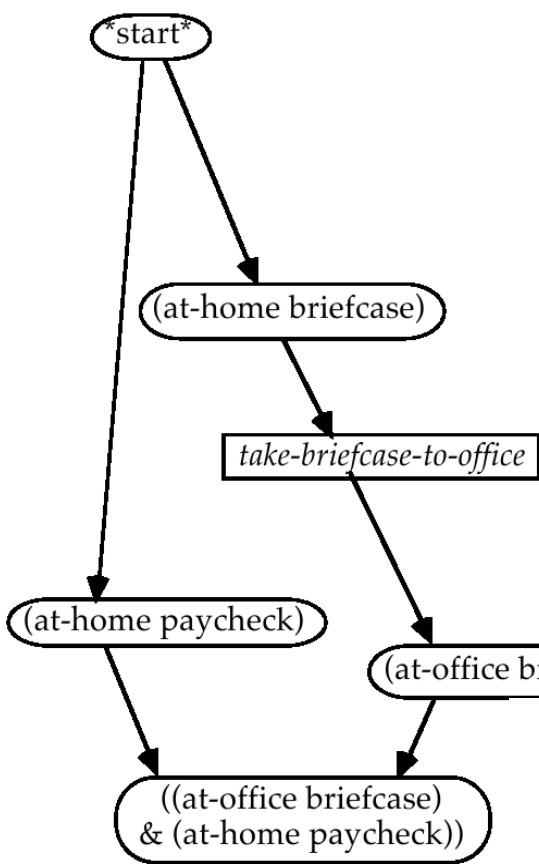
- PERCEPTION: Having a percept at time  $t$  with content  $P$  is a defeasible reason to believe  $P$ -at- $t$ .
- PERCEPTUAL-RELIABILITY: “ $R$  is true and having a percept with content  $P$  is not a reliable indicator of  $P$ ’s being true when  $R$  is true” is an undercutting defeater for PERCEPTION.
- TEMPORAL-PROJECTION: “ $P$ -at- $t$ ” is a defeasible reason for “ $P$ -at- $(t+\Delta t)$ ”, the strength of the reason being a monotonic decreasing function of  $\Delta t$ .
- STATISTICAL-SYLLOGISM: “ $c$  is a  $B$  &  $\text{prob}(A/B)$  is high” is a defeasible reason for “ $c$  is an  $A$ ”.
- CAUSAL-IMPLICATION: If  $t^* > t$ , “ $A$ -at- $t$  and  $P$ -at- $t$  and ( $A$  when  $P$  is causally-sufficient for  $Q$ )” is a defeasible reason for “ $Q$ -at- $t^*$ ”.
- CAUSAL-UNDERCUTTER – causal knowledge precedences temporal projection.



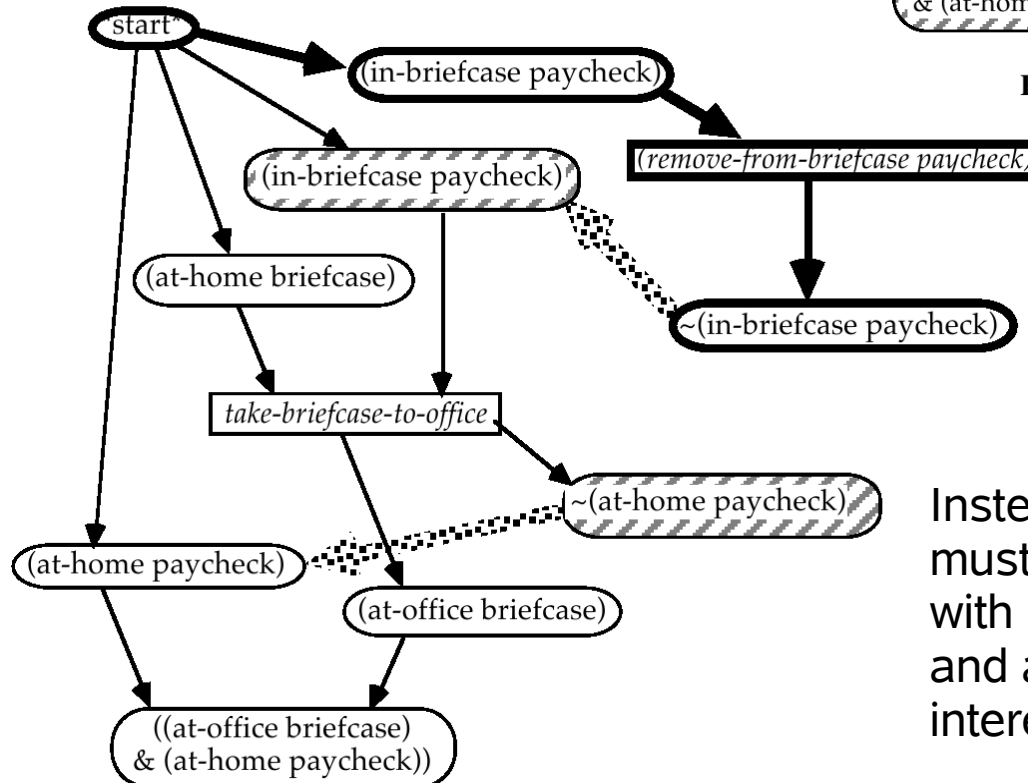


# OSCAR Practical Cognition

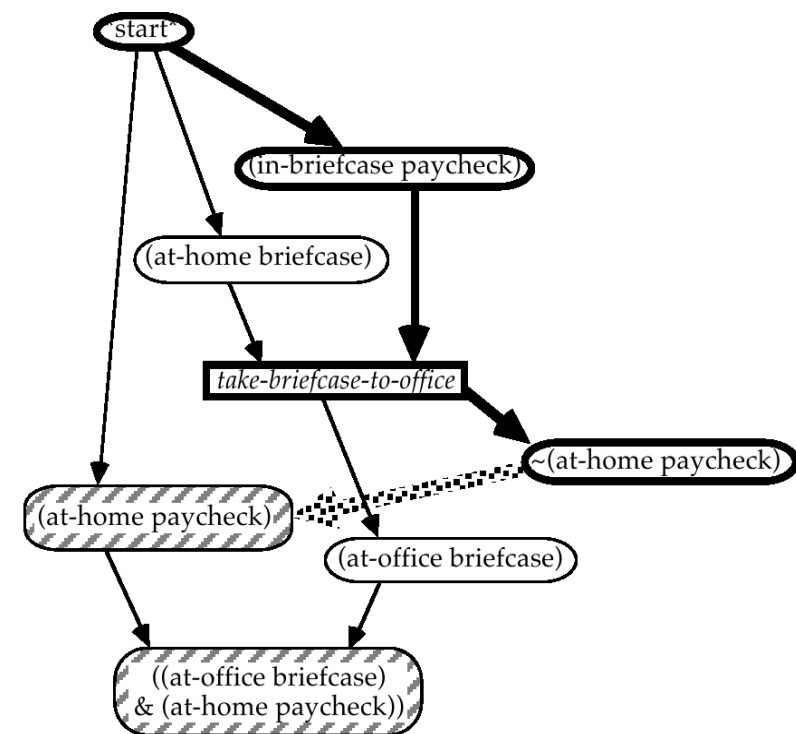
- goal selection, plan-construction, plan-selection, plan-execution; goals have values and plans have expected values
- “performing action  $A$  under circumstances  $C$  is causally sufficient for achieving goal  $G$ ”:  $(A/C) \Rightarrow G$
- impossible to rule out threats before merging plans; whether a plan will achieve a goal is a factual (epistemic) matter
- defeasible reason-schemas for reasoning about plans (each ends with defeasibly inferring that the plan achieves the goal):
  - GOAL-REGRESSION: for  $G\text{-at-}t$ , adopt  $(A/C) \Rightarrow G$ . Then adopt  $C\text{-at-}t^*$ . Then, construct a plan by (1) adding action  $A\text{-at-}t^*$ , (2) adding a constraint  $(t^* < t)$
  - SPLIT-CONJUNCTIVE-GOAL: for  $(G1\text{-at-}t1 \ \& \ G2\text{-at-}t2)$ , adopt  $G1\text{-at-}t1$  and  $G2\text{-at-}t2$  and merge inferred plans



**Figure 8.** Flawed plan



**Figure 10.** Defeating the defeater



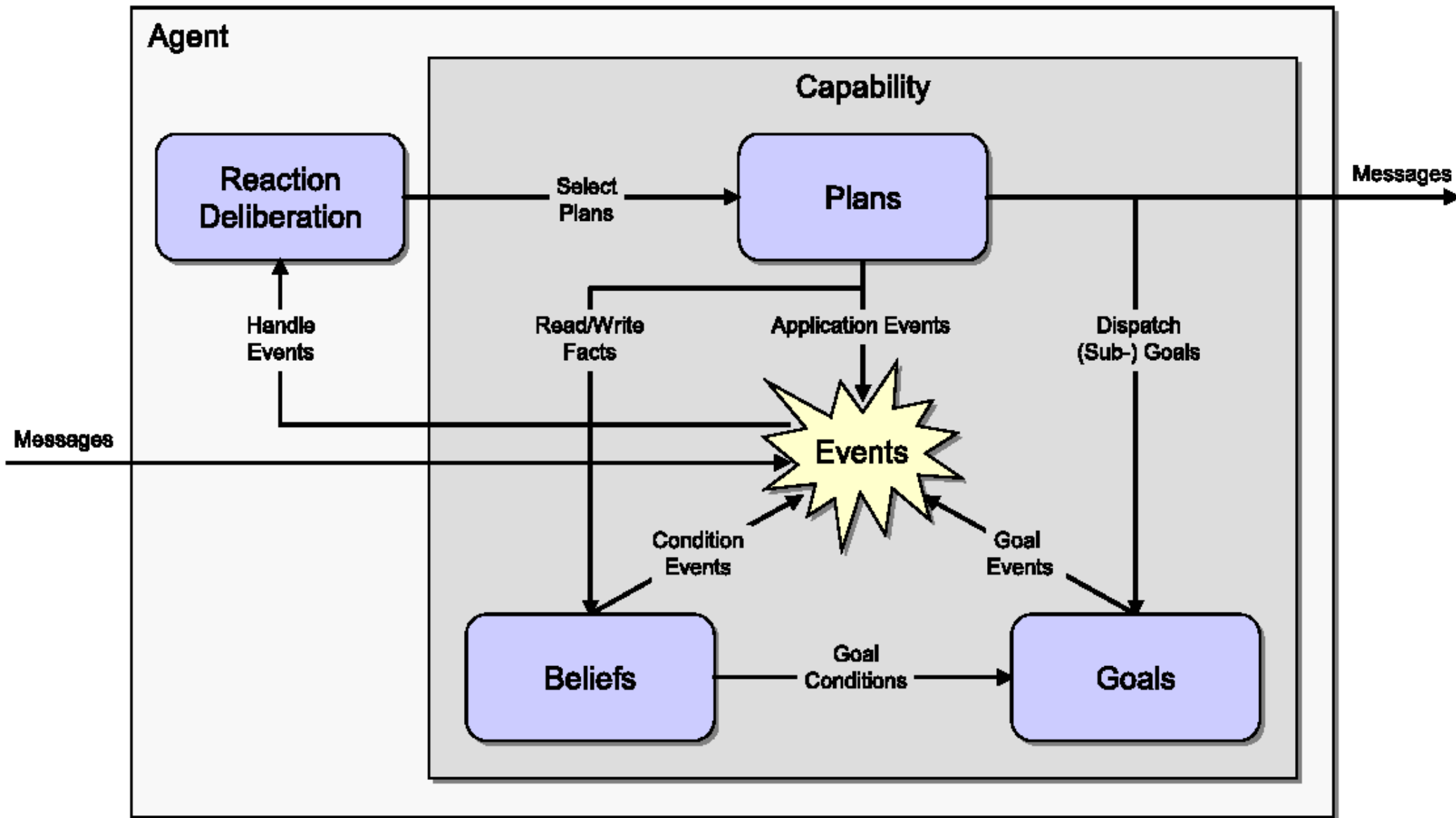
**Figure 9.** Defeating subplan

Instead of maximizing we must *satisfice* — seek plans with positive expected values, and always maintain an interest in finding better plans.

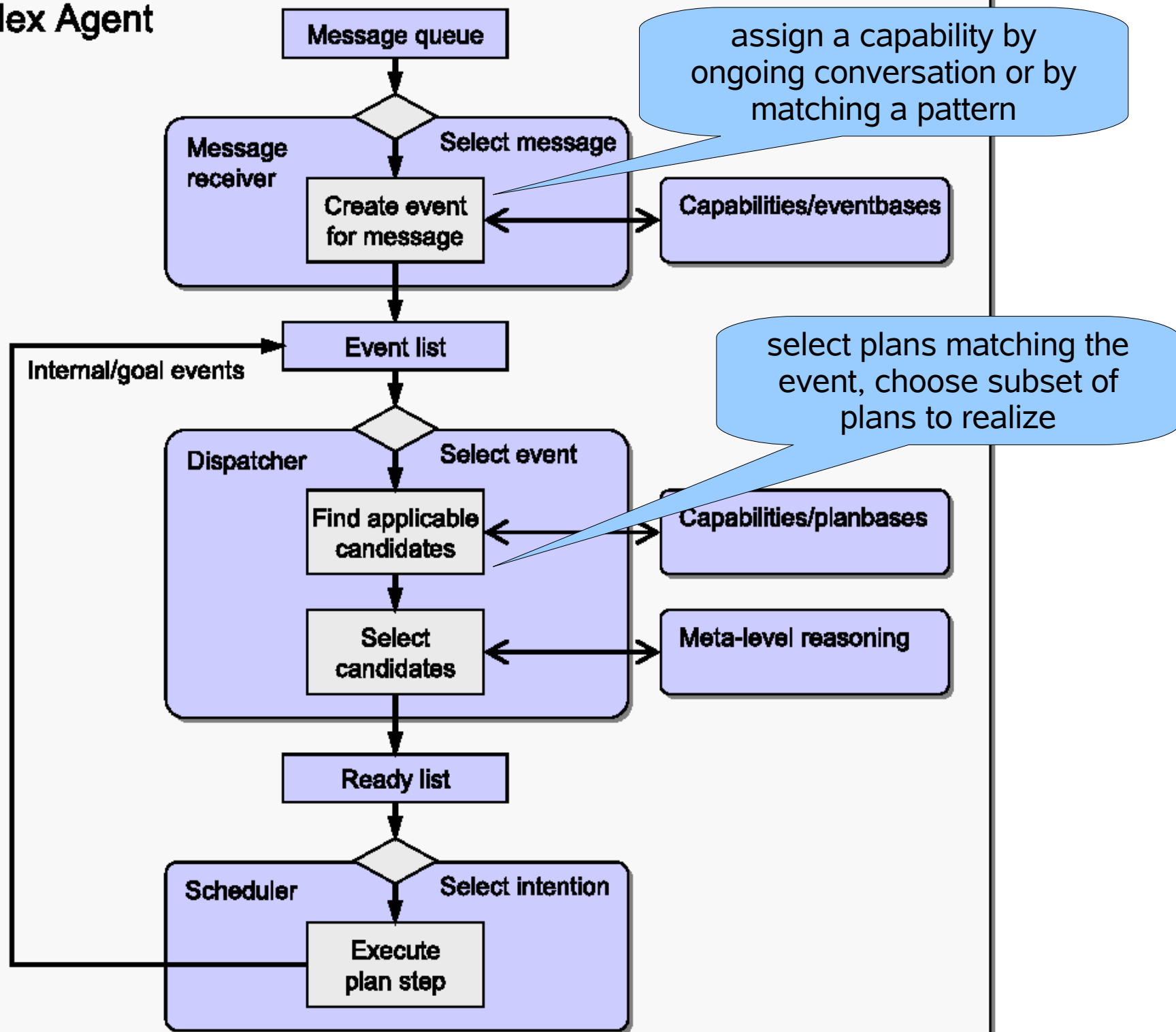
# Jadex: A BDI Reasoning Engine

- Agent acts towards some of the world states it desires to be true and believes to be possible.
- Beliefs are objects – named facts or named sets of facts – stored in beliefbase which monitors belief state conditions and can lead to actions.
- Capabilities: reusable modules of beliefs, goals, plans and events encapsulating a certain functionality.
- Agents defined in XML with procedural parts of plans in Java
- run on top of multi-agent middleware like Jade

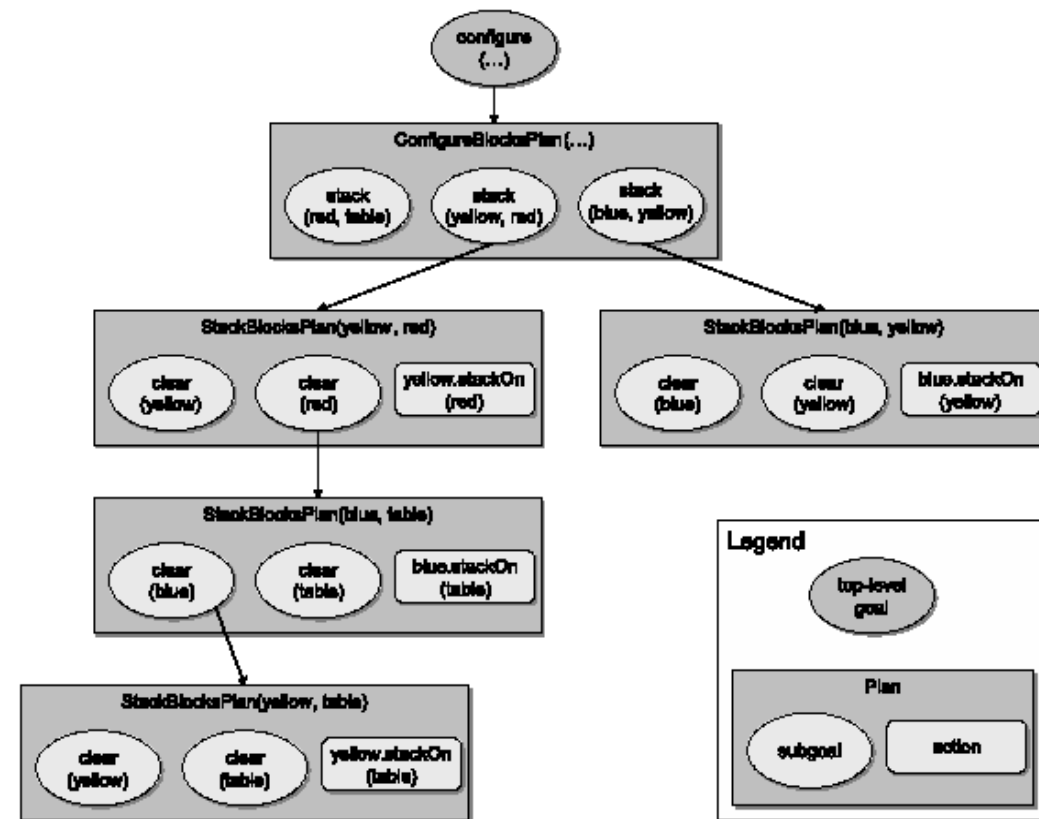
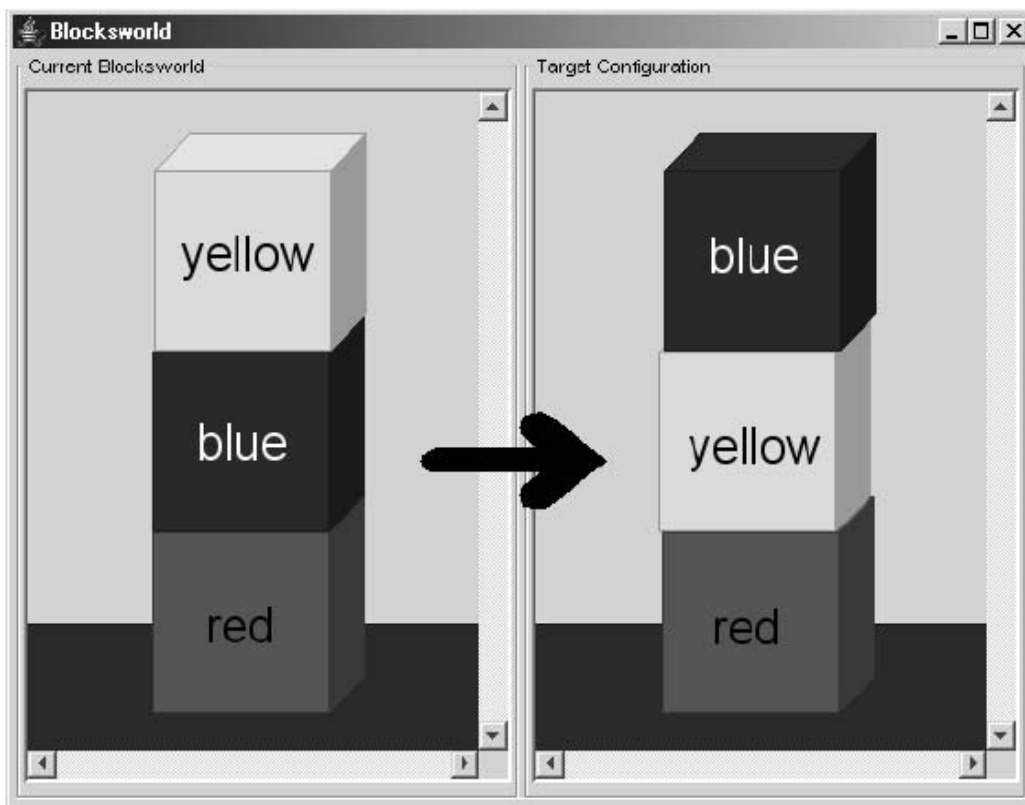
# Jadex: A BDI Reasoning Engine



# Jadex Agent



# Jadex (Blocksworld)



```

01: <agent name="Blocksworld" package="jadex.examples.blocksworld">
02:   <imports>
03:     <import>java.awt.Color</import>
04:   </imports>
05:
06:   <beliefs>
07:     <belief name="table" class="Table">
08:       <fact>new Table()</fact>
09:     </belief>
10:     <beliefset name="blocks" class="Block">
11:       <fact>new Block(new Color(240, 16, 16), $beliefbase.table)</fact>
12:       <fact>new Block(new Color(16, 16, 240), $beliefbase.table.allBlocks[0])</fact>
13:       <fact>new Block(new Color(240, 240, 16), $beliefbase.table.allBlocks[1])</fact>
14:       ...
15:     </beliefset>
16:   </beliefs>
17:
18:   <goals>
19:     <achievegoal name="clear">
20:       <parameter name="block" class="Block" />
21:       <targetcondition>$goal.block.isClear()</targetcondition>
22:     </achievegoal>
23:     <achievegoal name="stack">
24:       <parameter name="block" class="Block" />
25:       <parameter name="target" class="Block" />
26:       <targetcondition>$goal.block.lower==$goal.target</targetcondition>
27:     </achievegoal>
28:     <achievegoal name="configure">
29:       <parameter name="configuration" class="Table" />
30:       <targetcondition>
31:         $beliefbase.table.configurationEquals($goal.configuration)
32:       </targetcondition>
33:     </achievegoal>
34:   </goals>
35:
36:   <plans>
37:     <plan name="stack">
38:       <body>new StackBlocksPlan($event.goal.block, $event.goal.target)</body>
39:       <trigger><goal ref="stack"/></trigger>
40:     </plan>
41:     <plan name="configure">
42:       <body>new ConfigureBlocksPlan($event.goal.configuration)</body>
43:       <trigger><goal ref="configure"/></trigger>
44:     </plan>
45:     <plan name="clear">
46:       <bindings>
47:         <binding name="upper">
48:           select $upper from $beliefbase.blocks where $upper.getLower()==$event.goal.block
49:         </binding>
50:       </bindings>
51:       <body>new StackBlocksPlan($upper, $beliefbase.table)</body>
52:       <trigger><goal ref="clear"/></trigger>
53:     </plan>
54:   </plans>
55: </agent>

```

```

01: package jadex.examples.blocksworld;
02: import jadex.runtime.*;
03:
04: /** Plan to to establish a given configuration of blocks. */
05: public class ConfigureBlocksPlan extends Plan {
06:   protected Table table;
07:
08:   public ConfigureBlocksPaperPlan(Table table) {
09:     this.table = table;
10:   }
11:
12:   public void body() {
13:     Block[][] stacks = table.getStacks();
14:     for(int i=0; i<stacks.length; i++) {
15:       for(int j=0; j<stacks[i].length; j++) {
16:         Block block = (Block)getBeliefbase().getBeliefSet("blocks").getFact(stacks[i][j]);
17:         Block target = stacks[i][j].getLower() == table
18:           ? (Table)getBeliefbase().getBelief("table").getFact()
19:           : (Block)getBeliefbase().getBeliefSet("blocks").getFact(stacks[i][j].getLower());
20:
21:         IGoal stack = createGoal("stack");
22:         stack.getParameter("block").setValue(block);
23:         stack.getI
24:         dispatchS
25:       }
26:     }
27:   }
28: }
01: package jadex.examples.blocksworld;
02: import jadex.runtime.*;
03:
04: /** Plan to stack one block on top of another target block. */
05: public class StackBlocksPlan extends Plan {
06:   protected Block block;
07:   protected Block target;
08:
09:   public StackBlocksPaperPlan(Block block, Block target) {
10:     this.block = block;
11:     this.target = target;
12:   }
13:
14:   public void body() {
15:     IGoal clear = createGoal("clear");
16:     clear.getParameter("block").setValue(block);
17:     dispatchSubgoalAndWait(clear);
18:
19:     clear = createGoal("clear");
20:     clear.getParameter("block").setValue(target);
21:     dispatchSubgoalAndWait(clear);
22:
23:     block.stackOn(target);
24:   }
25: }

```

The explicit specification and strong typing of beliefs, goals, etc. facilitates consistency checks of XML Agent Definition Files to detect errors (e.g. spelling mistakes) as early as possible.

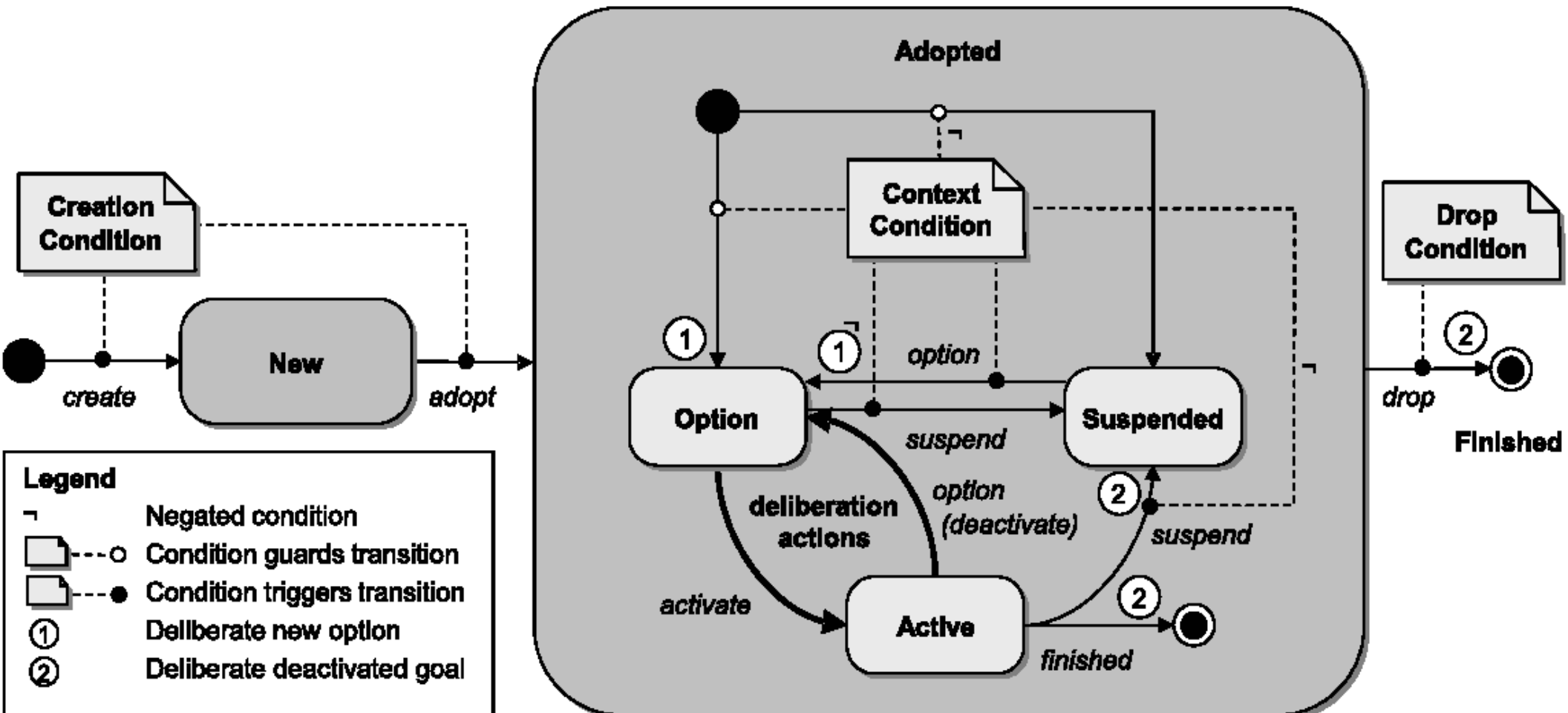
Figure 1.11. Java code for StackBlocksPlan

# Jadex: Goal Deliberation

- BDI system ensures that constraints, set by an agent developer, are respected and only consistent goal sets are pursued at any one time.
- Different goal types:
  - perform: act disregarding of action result,
  - achieve: defines desired world state only,
  - query: like „achieve” wrt. internal state,
  - maintain: monitor state and re-establish it when needed.
- Goal states: active,
  - option: explicitly not pursued, e.g. conflicts with some active goal
  - suspended: its context is invalid
- Other goal properties:
  - creation condition, context condition, drop condition (when a goal instance is removed)



# Jadex: Goal Deliberation



# Jadex: Goal Deliberation

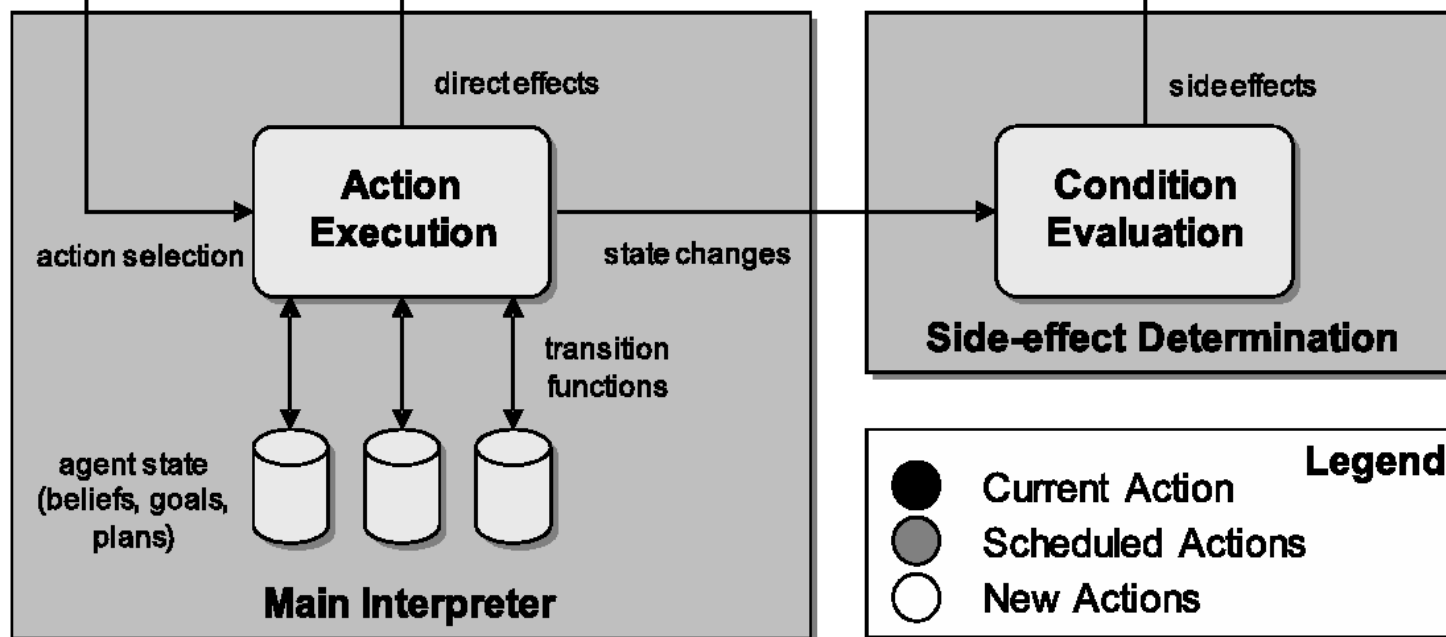
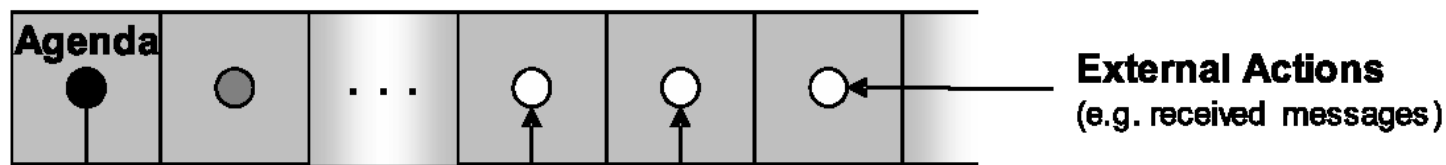
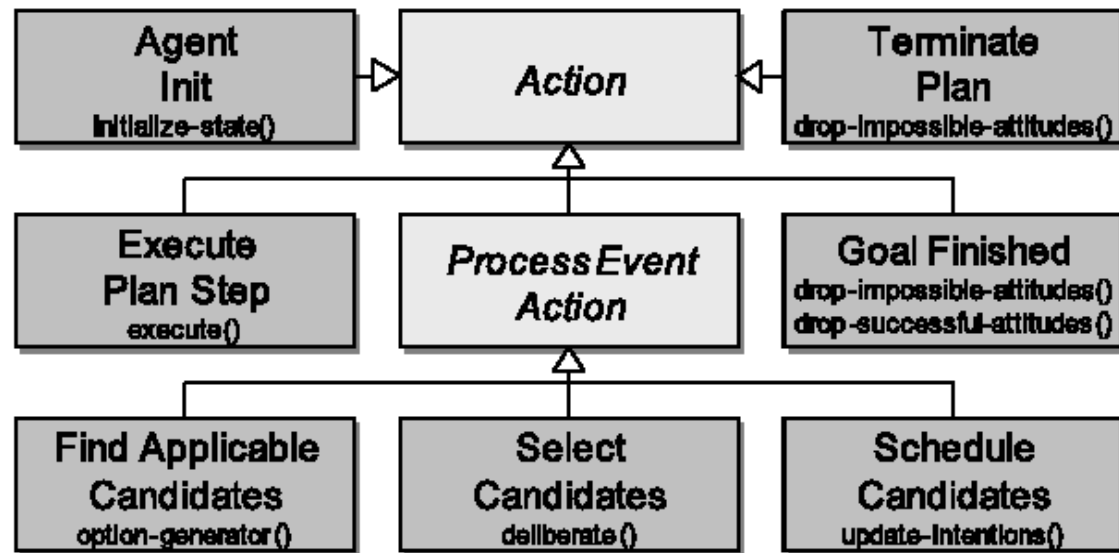
- cardinalities: how many goals of given type at once
- inhibition links: between goal types and between goal instances
- deliberation initiated by:
  - a new goal adopted, or a context of a suspended goal valid again ==> deliberate new option
  - a goal becomes inactive (suspended, finished or dropped) ==> which inhibited options can be reactivated
- consider a local subset of the agent's goals, derived from the goal that triggered deliberation, plus relevant (e.g. linked) others

```

01 initialize-state();
02 repeat
03   options := option-generator(event-queue);
04   selected-options := deliberate(options);
05   update-intentions(selected-options);
06   execute();
07   get-new-external-events();
08   drop-successful-attitudes();
09   drop-impossible-attitudes();
10 end repeat

```

old  
interpreter



new  
interpreter  
deliberates  
only when  
it's needed

# Jadex: Goal Deliberation

- Limitations:
  - The strategy does **only** consider **bilateral relationships**: e.g. not possible that two goals together are more important than another single goal.
  - **Conflicts** between subgoals cannot **always** be **resolved optimally**, e.g. a conflict between subgoals cannot be resolved by replacing one of the subgoals with another non-conflicting subgoal.
  - **Conflicts at plan level** are **not considered**, which means that inconsistencies between plans e.g. because of access to conflicting resources are not detected.
  - **Positive interactions between goals** are **not considered**, which means that the strategy cannot identify and exploit potentially common subgoals.
- The reason for choosing inhibition links instead of using utility values is that it allows to adopt a local view and frees the agent developer from establishing a global ordering between all goals.

# Jadex: Planning BDI Agents

- „Theory of Practical Reasoning” (BDI) to overcome the poor performance of propositional planners by timely reactivity and goal deliberation
- combine their strength with flexible means-ends reasoning of deliberative planners:
  - The planner applied to produce long term plans and to handle single parts to a reactive BDI subsystem:
    - serious performance concerns especially in dynamic environments where continuous changes force the planner to re-plan (algorithms have been devised for one shot planning)
  - Or: augment the BDI system with a relatively simple planner that is invoked from the BDI controller and used for the purpose of creating short-term plans that need a proof of correctness.

# Jadex Planner

- The planner reasons about states of the environment and agent's own inner mental states
  - environment: objects with state-dependent attributes
  - mental states: a stack of goals (measures how far a goal state is from a given state)
- actions are transition functions between states
  - a triple: <precondition; change of goals; change in objects>
- agent desires, in respect to possible solutions, assumed not to change within the short scope of operational planning
- Planning assumed to be a higher cognitive activity than reacting and controlling – granted more computational resources. But Jadex planner is at a level below BDI.

actions

current state

PLAN( $\mathbb{A}, s_c, D, T$ )

desires

1  $best \leftarrow s_c$

2  $e(best) \leftarrow \infty$

3  $agenda \leftarrow \{s_c\}$

4 **while**  $|agenda| > 0$  and  $t_c < T$

5 **do**  $s \leftarrow pop(agenda)$

inverse utility estimate

6 **if**  $e(s) < e(best)$  and  $|G_s| \leq |G_{best}|$

7 **then**  $best \leftarrow s$

8  $Options \leftarrow generateOptions(s, \mathbb{A})$

9 **for each**  $\{a = \langle p_a, \gamma_a, \omega_a \rangle \mid a \in Options\}$

10 **do**  $G' \leftarrow \gamma_a(G)$

precondition, change of goals, change of objects

11  $s' \leftarrow \langle G', \omega_a(\sigma), s, a \rangle$

12  $removeSatisfiedGoals(s')$

13  $e(s') \leftarrow inverseUtility(s', D) + goalsDistance(s', G')$

14  $insert(s', agenda)$

15 **return**  $best$

With an advanced BDI system, one is equipped with reasoning and a strong conceptual framework, so there is no need to duplicate the functionality of both.

# Jadex Planner

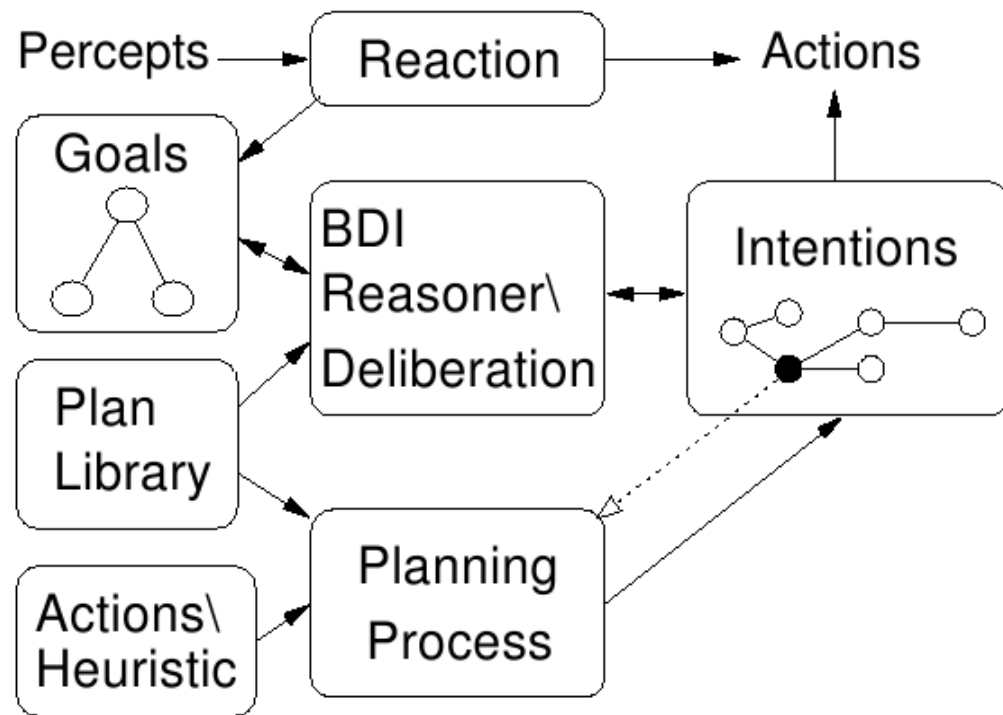
- Under tight timing constraints, state-based planners augmented with domain specific knowledge are superior to partial-order planners.
- State contains reference to parent state allowing for temporal conditions.
- The domain specific knowledge used to guide the planner is hidden in the action applicability predicates, in the goal distance functions, and in the inverse utility functions.
- reasons *about* the goal stack at each step



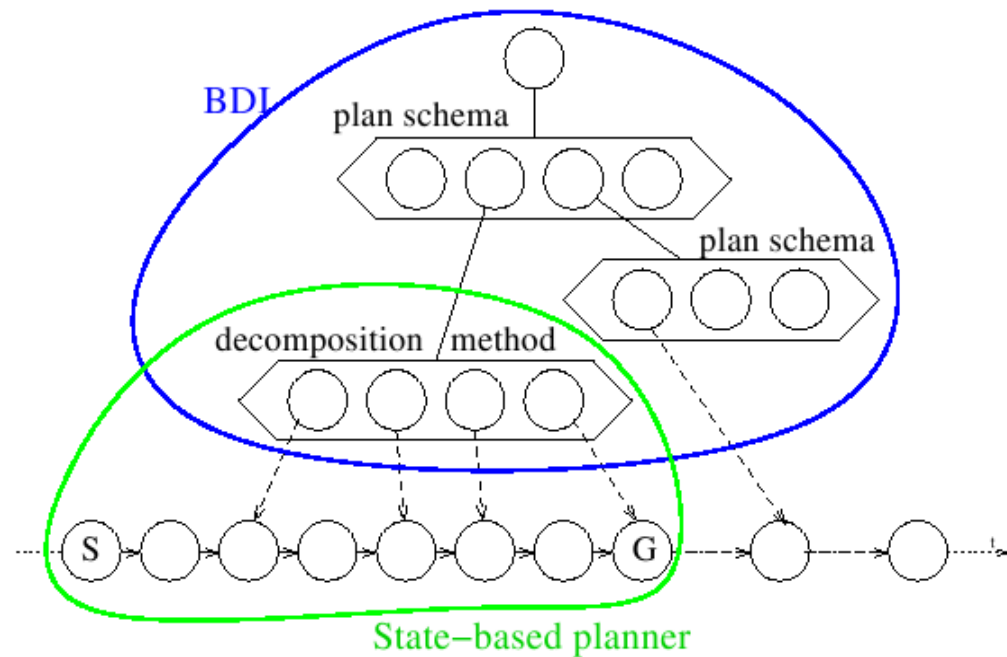
# Jadex Planner-BDI Integration

- Goals are used by the BDI reasoner to choose among plan schemata and create the action structure on the agenda of intentions.
- The intentional structure is given by current plan instances.
- Reactive subsystem, triggered by belief changes or a percept, generates new goals for the reasoner.
- At the meta-level goal deliberation analyzes dependencies and modifies the intentional structure accordingly to agent's preferences.

# Jadex Planner-BDI Integration



**Fig. 1.** Generic BDI architecture with a planner. Planning is an activity of the agent (filled circle) and produces new intentions.



**Fig. 2.** Schematic illustration of a plan in the hybrid system.

# Jadex Planner-BDI Integration

- Metric for state-to-goal distance can be derived automatically from goal schema (but better hand coded with domain knowledge).
- Created plans go directly into the intentions structure (with their BDI goal as parent node) and are not stored in a plan library. Because plans are created at low-level, their parameters are tightly bound and they are applicable only to a particular situation.
- Monitoring correct plan execution: component similar to the planner is used to evaluate the remainder in a simulated environment.

# Jadex Planner-BDI Integration

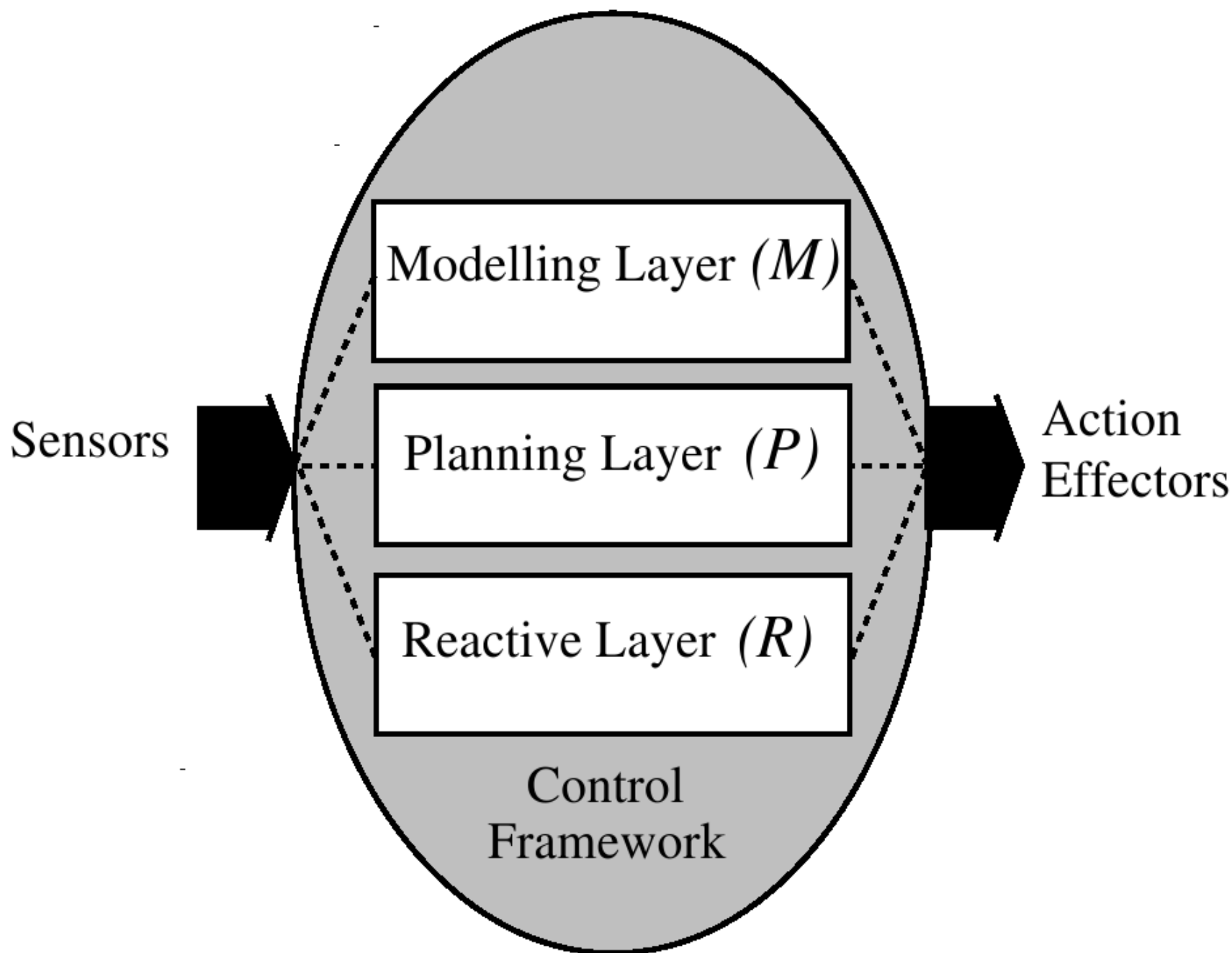
- Planning can fail because:
  - No way to improve the agent's situation. The BDI reasoner may retry finding a plan after some time.
  - No correct plan satisfying all goals and subgoals. The partial plan may be executed with the hope the future planning, starting from a better situation, may find a complete plan.
  - Timeout. This case can be handled as the previous one.
  - A number of correct plan instances is returned in successive trials but they fail to reach the goal. In this case the domain description is too abstract and lacks the knowledge needed by the planner to recognize specific reasons for failure.

# TouringMachines Hybrid Architecture

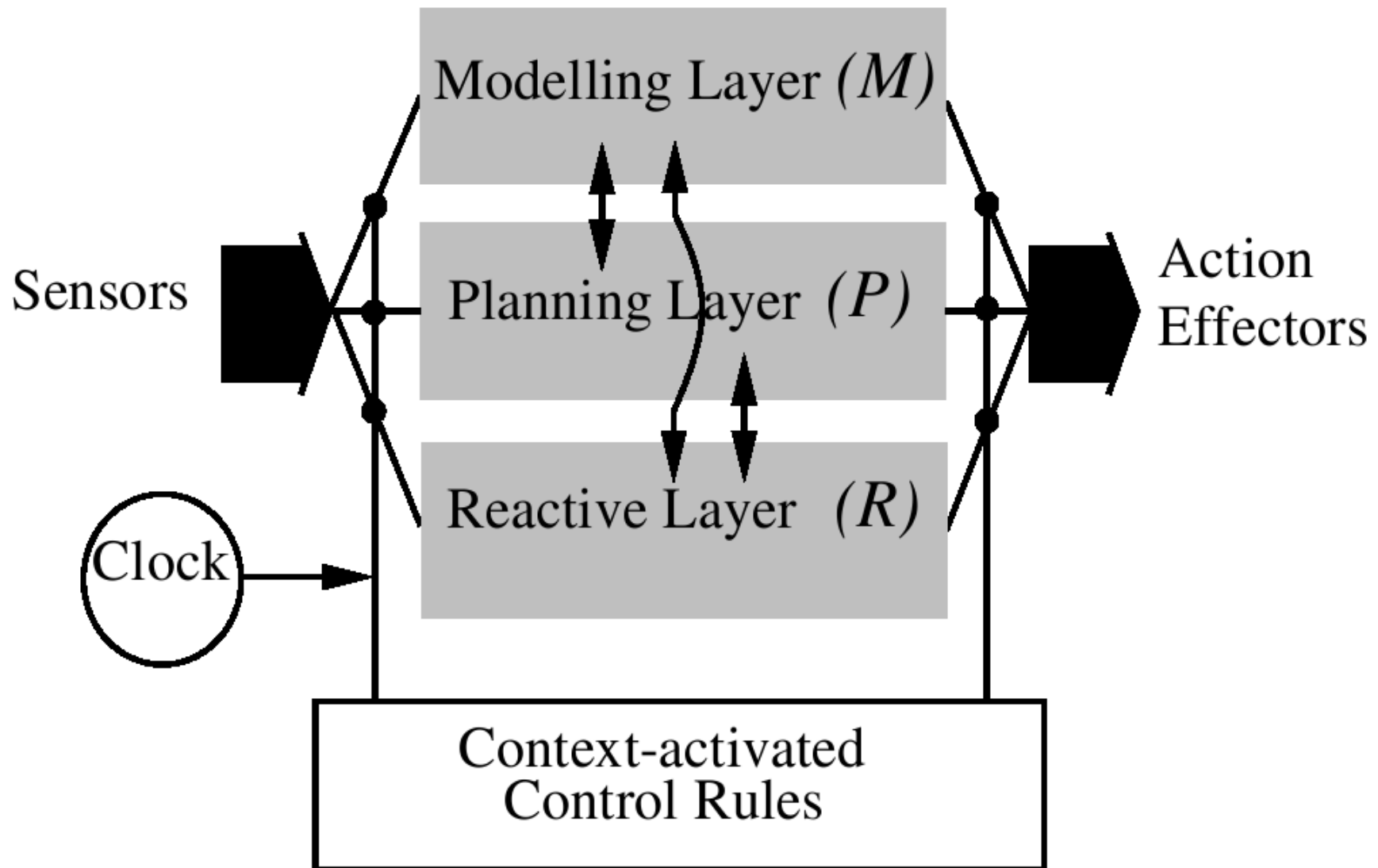
- a resource-bounded, goal-directed agent to react promptly to unexpected changes in its environment;
- at the same time, to reason predictively about potential conflicts by constructing and projecting theories which hypothesise other agents' goals and intentions
- developed to understand the role of different functional capabilities in constraining an agent's behaviour under varying environmental conditions, an experimental testbed comprising a simulated multi-agent world

# Touring Machines Hybrid Architecture

- Agents following a different route from some starting to some goal location within certain time bounds and/or spatial constraints. Agent starts with some geographical knowledge of the world (e.g. locations of paths and path intersections), but no prior knowledge of other agents' locations or goals or static obstacles.
- Three-layered: each connecting perception to action
  - reactive: fast capabilities for unplanned or unmodelled
  - planning: forward route planning
  - modelling: e.g. hypothetical reasoning, attention



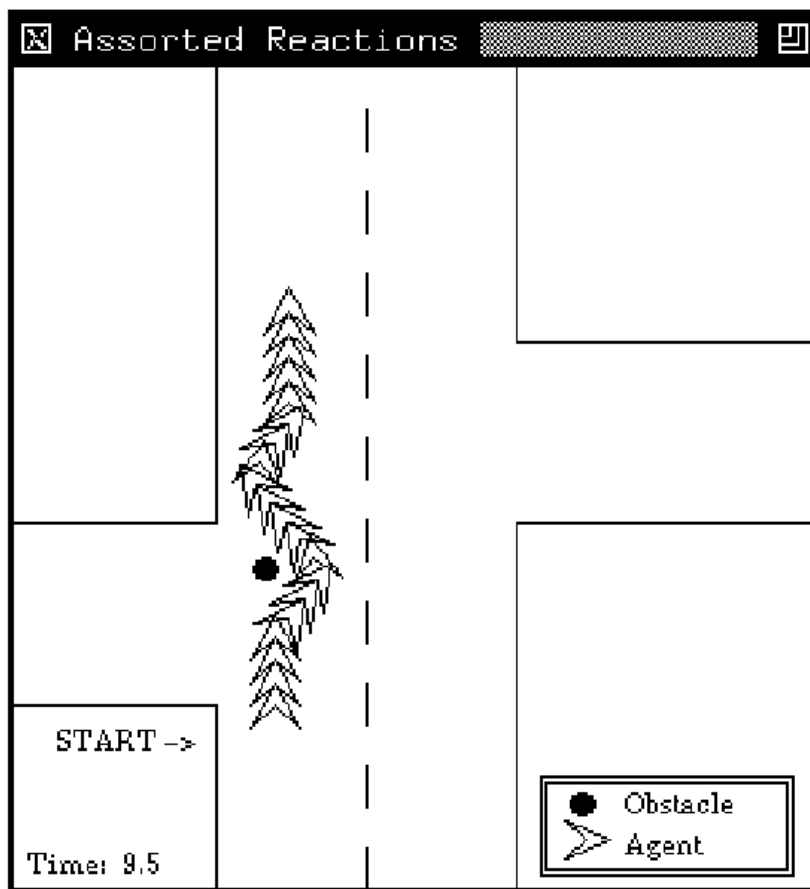
# TouringMachines control framework





# TouringMachines Reactive Layer

- situation-action rules for avoiding obstacles, walls, kerbs or other agents, and for preventing agent from straying over path lane markings
- agent can be made reactive or inert by choosing thresholds for and strength of its reactions
- rules stimulated *solely* by input from sensors; actions sent to effectors if approved by control framework
- actions are not combined like in „boids“, the one for the closest object is selected
- when rule fires, modelling layer is notified



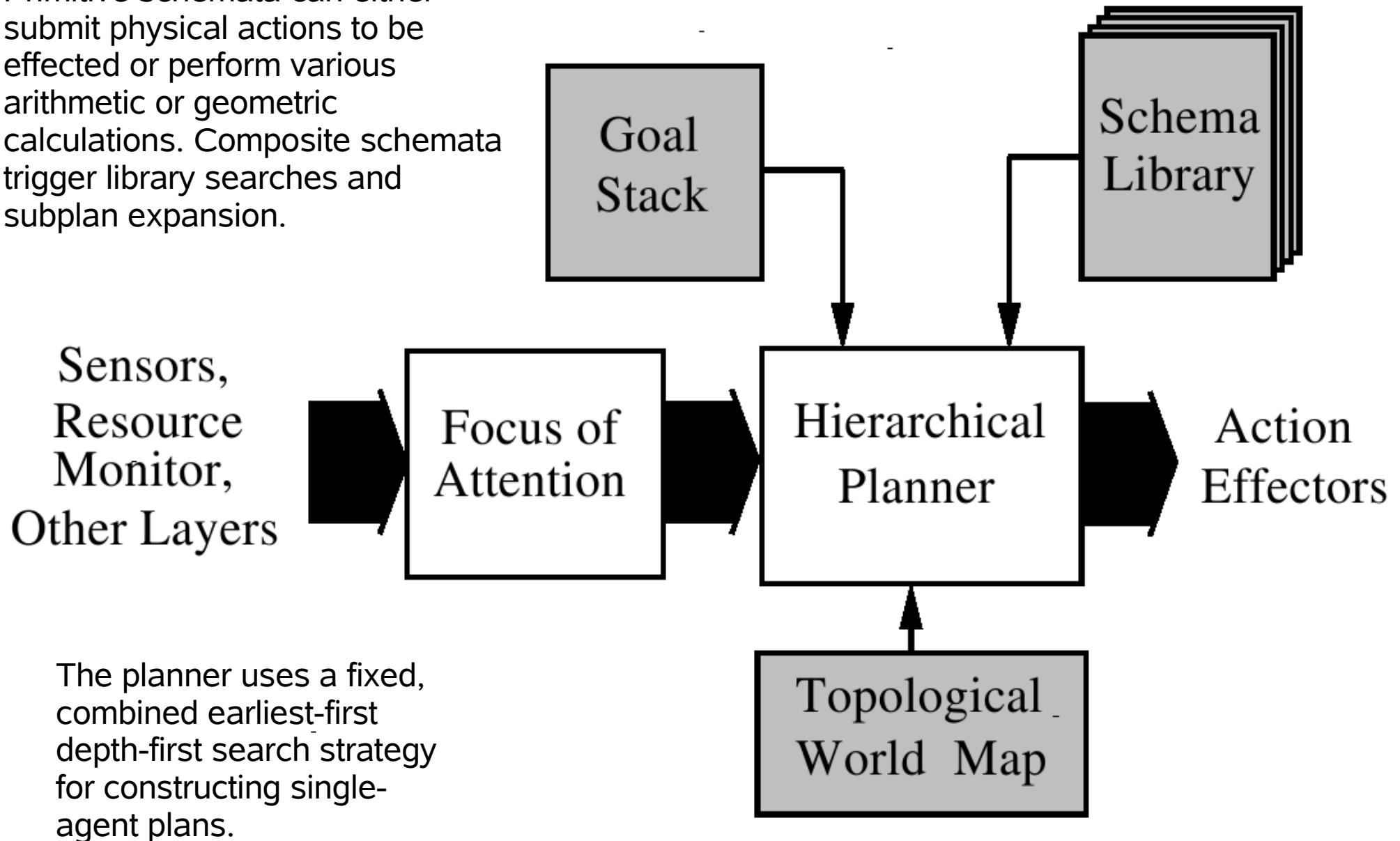
```
rule-3:  if is-in-front(Other, Observer) and  
         velocity(Other) < velocity(Observer) and  
         separation(Other, Observer) < FrontalAgentThreshold  
         then  
           change-velocity(Observer, FrontalAvoidanceVelocity)  
  
rule-4:  if is-behind(Other, Observer) and  
         velocity(Other) > velocity(Observer) and  
         separation(Other, Observer) < RearAgentThreshold  
         then  
           change-velocity(Observer, RearAvoidanceVelocity)
```

# TouringMachines Planning Layer

- hierarchical, partial planner interleaving plan formation and execution, and defer committing to specific subplan execution methods or temporal orderings of subplans until absolutely necessary
- embedded: its operation can regularly be pre-empted and its state suspended for subsequent use
- template plans or schemata: procedural structures
  - consisting of: a body,
  - a set of preconditions,
  - a set of applicability conditions (e.g. temporal ordering constraints),
  - a set of postconditions,
  - cost in terms of computational resources.

# TouringMachines Planning Layer

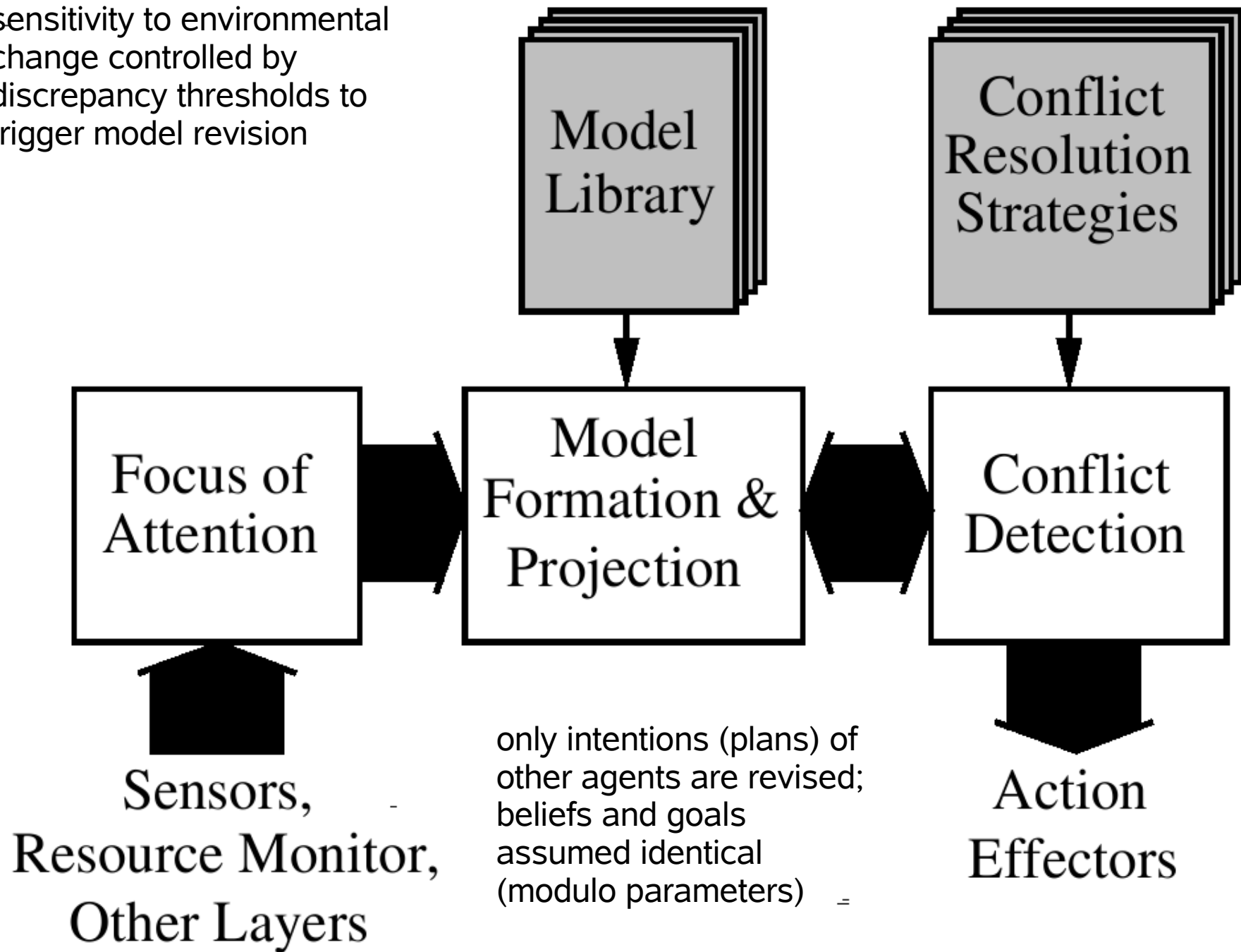
Primitive schemata can either submit physical actions to be effected or perform various arithmetic or geometric calculations. Composite schemata trigger library searches and subplan expansion.



# TouringMachines Modelling Layer

- predictions allow to detect conflicting goals and accomodate by taking correcting action
- deliberation steps are resource (time) bounded
- $\langle C, B, D, I \rangle(t)$  models an entity's behaviour:
  - C is the entity's Configuration:
    - (x,y)-location, speed, acceleration, orientation,
    - signalled communications;
  - B: Beliefs, D: prioritised goals or Desires; I: plan or Intention structure.
- desires can be *achievable* or *homeostatic*
- reasoning is about detecting discrepancies between actual and predicted (them) / desired (me)

- sensitivity to environmental change controlled by discrepancy thresholds to trigger model revision



# TouringWorld Experimental Testbed

- realized by discrete event simulator by action scheduling
- various agent- and environment-level parameters:
  - distribution of computational resources within control layers, amount of forward planning, sensitivity of reactive rules, frequency of sensing or modeling
  - sensing horizon, initial goal deadline, # of other fast-moving agents, ratio of CPU to simulated world time
- behavioral ecology: design-behavior-environment tradeoffs
- the fourth layer: self-tuning (classify operational contexts)

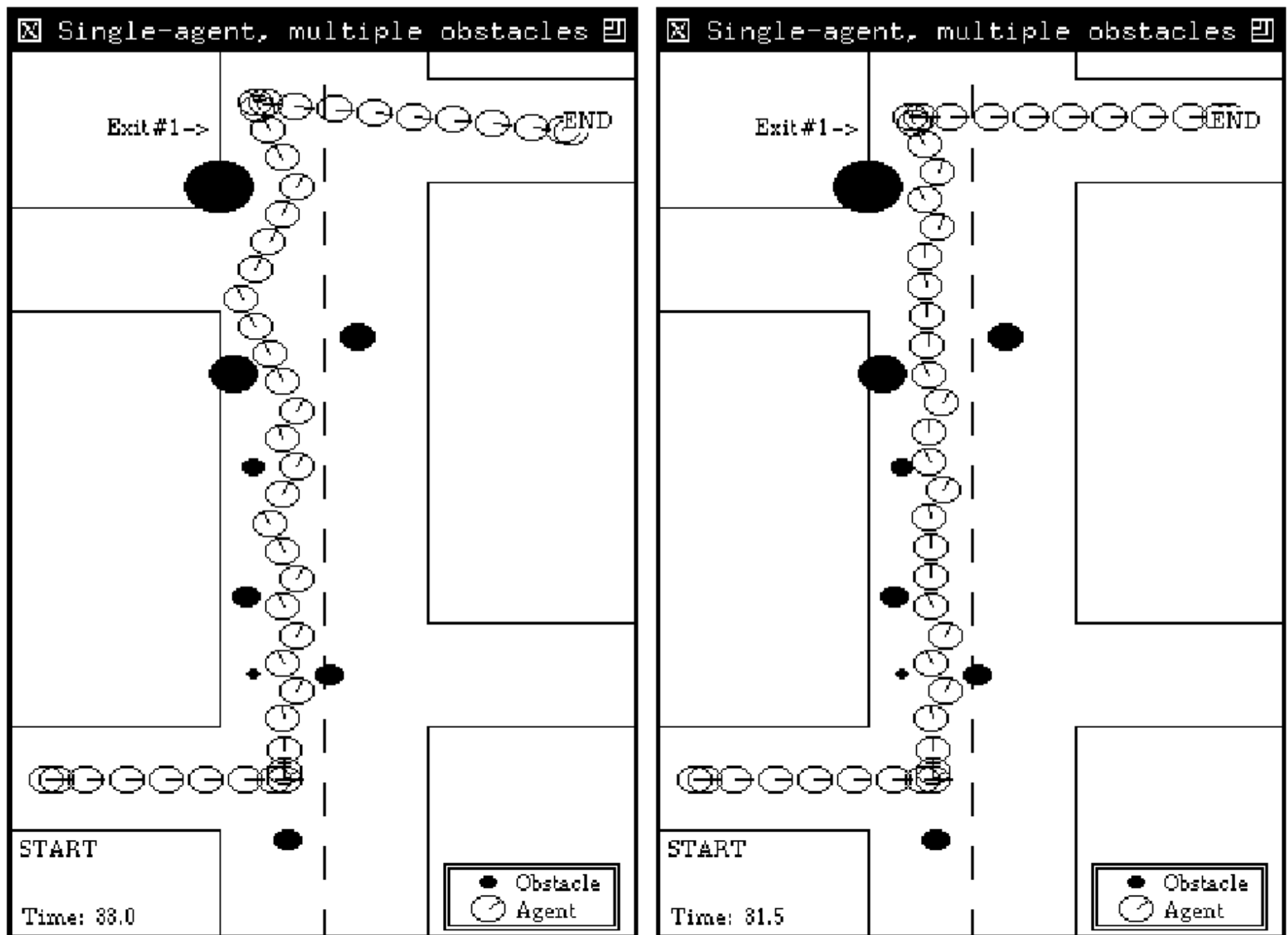


Figure 6: With the bounds around the parameter  $\theta_{desired}$  set to  $\pm 40^\circ$  (left-hand frame), the agent covers more distance and so takes longer (1.5 time units) to arrive at its target than when its bounds are set to  $\pm 0^\circ$  (right-hand frame).



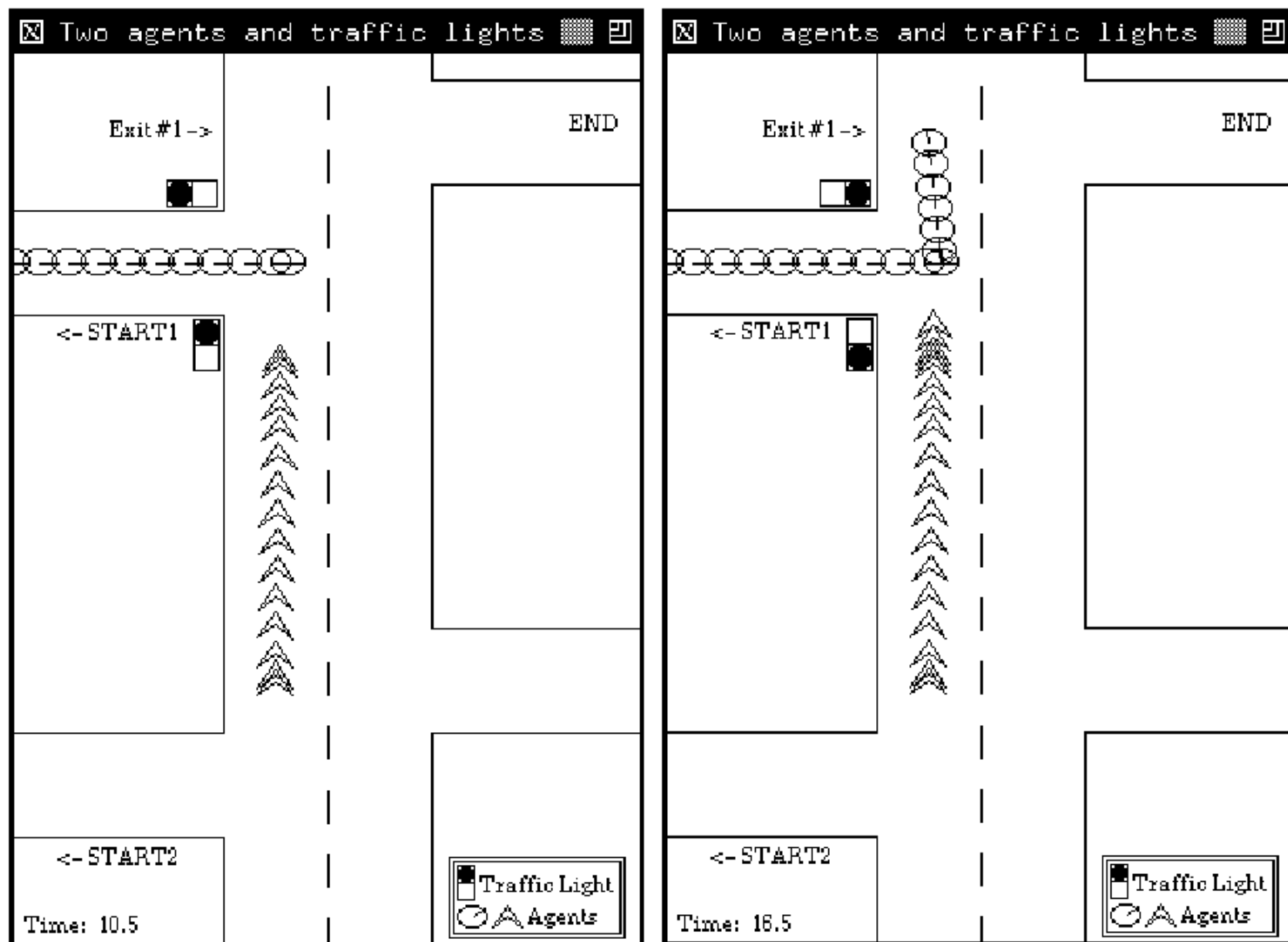


Figure 7: By reasoning about the interactions between themselves and the relevant traffic lights, the two agents coordinate their activities accordingly: in the left-hand frame, the chevron-shaped agent has stopped at its red light and the round agent has proceeded into the intersection; once the chevron-shaped agent's light has turned green (right-hand frame), it sets off to complete its initially intended task.

# Cyc

- Commonsense knowledge is essential to understanding...
- and to provide a specific workable context for each situation.
- Prime the „knowledge pump” with the millions of everyday terms, concepts, facts, and rules of thumb that comprise human consensus reality.
- Represented in a form of second order predicate calculus.

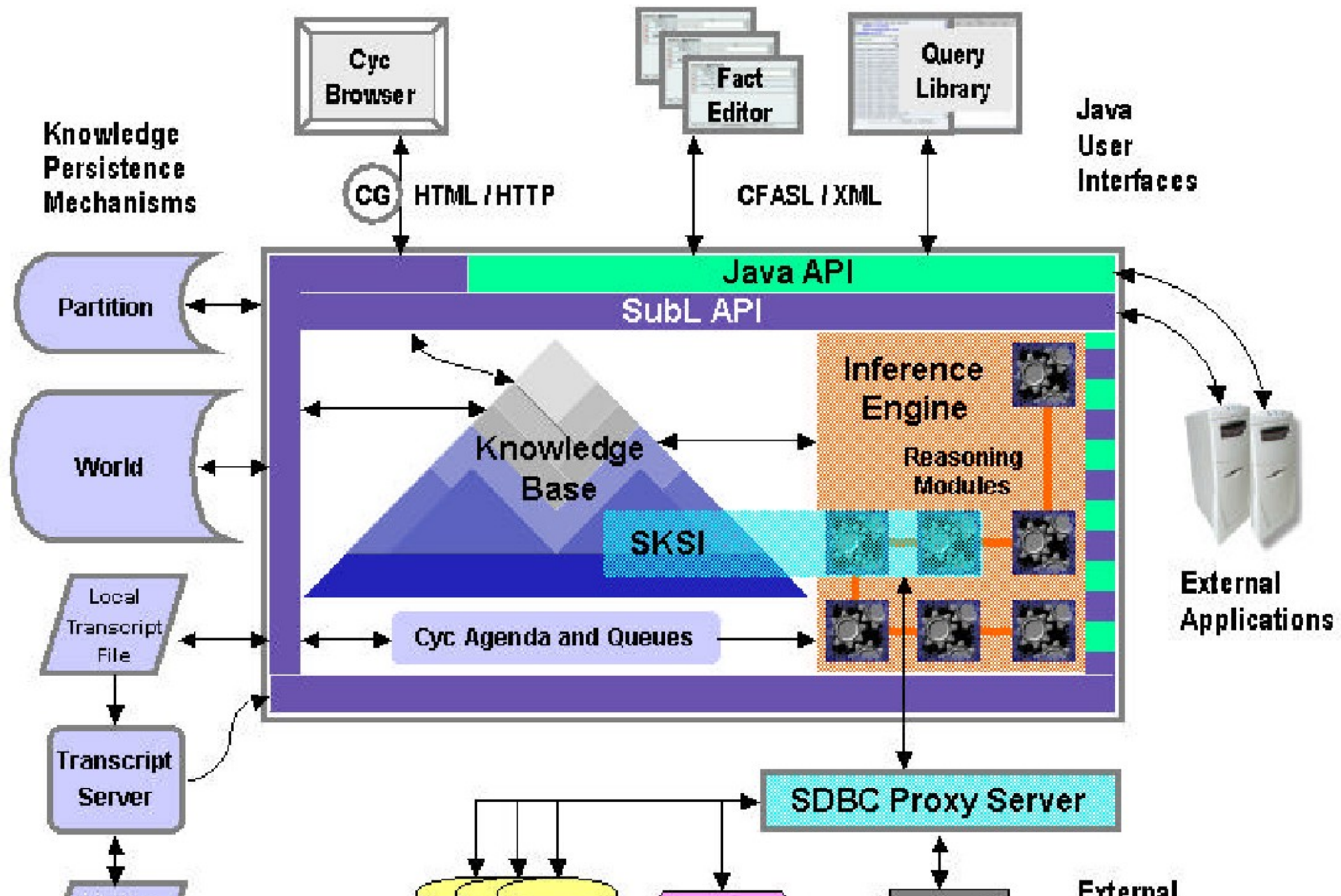
# Cyc lessons

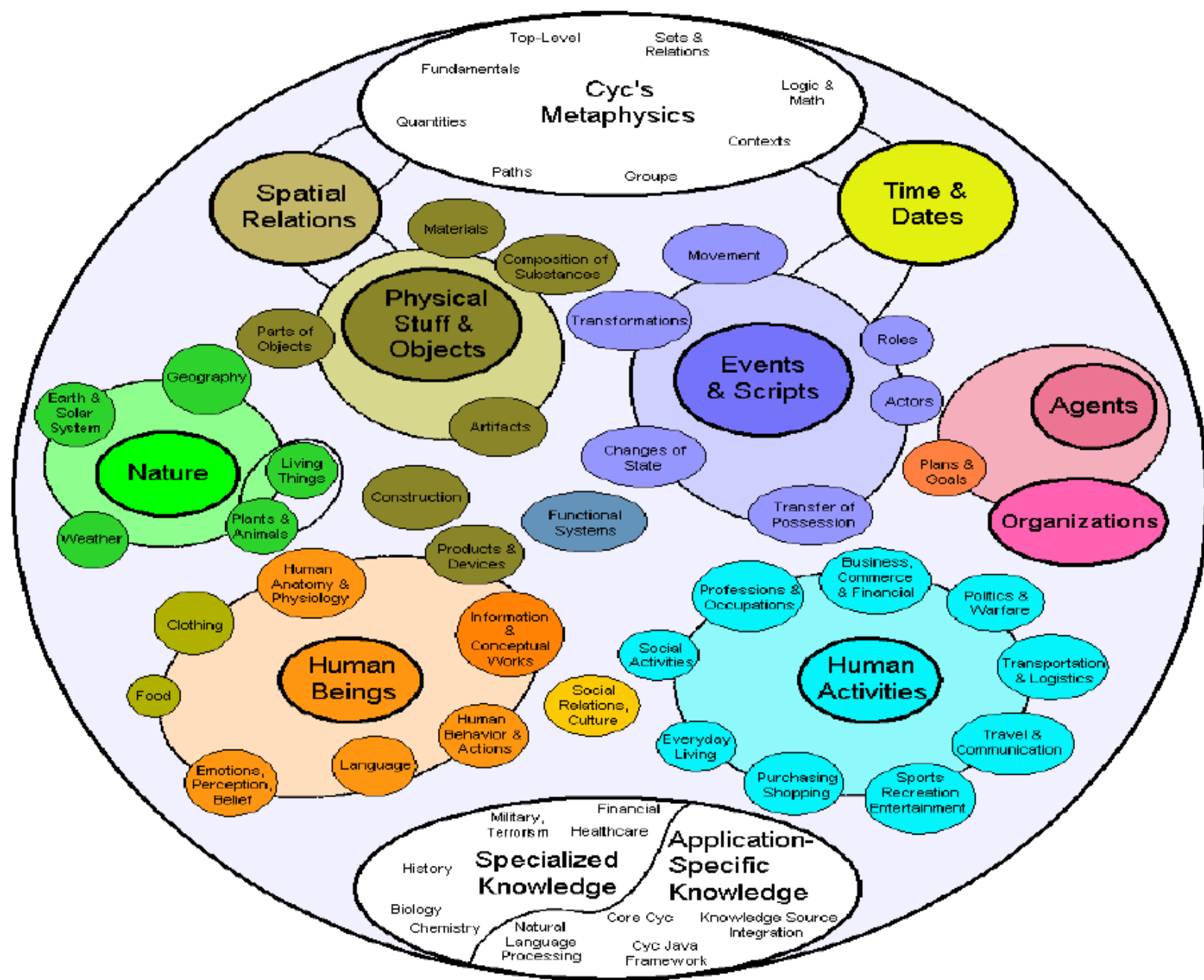
- Holding probabilities for each sentence had bad consequences. To decide whether to believe something, CYC gathers up all the pro and con arguments it can think of, examines them, and then reaches a conclusion.
- *Expressive language* (EL) for knowledge entry and *heuristic language* (HL) for knowledge processing
- Only contexts, or *microtheories*, are hold consistent (coarse-grained paraconsistency)

# Cyc architecture

- Knowledge Base
- Worlds: images of Cyc state
- Inference Engine
- User Interfaces
- Transcripts and the Transcript Server
  - synchronizing multiple Cyc installations
- Partitions
  - knowledge exchange
- Semantic Knowledge Source Integration (SKSI) Facility
  - communication with structured information sources, „outsourcing” knowledge
- Application Programming Interfaces (APIs)

# Cyc architecture





**Fig. 1.** Cyc KB Topic Map. The information in the Cyc KB can be subdivided into loosely grouped, inter-related “blocks” of knowledge at various levels of generality. In this diagram, Cyc’s understanding of metaphysics is the most general block of knowledge, gradating down to the very specific knowledge in tightly defined domains.

# Cyc Some ideas for inference

- Inference engine is composed of approximately a thousand specialized reasoners, called *inference modules*, handling from subsumption to transitivity.
- An inference harness breaks a problem down into sub-problems and selects among the modules that may apply to each and chooses follow-up approaches.
- The behavior of the inference harness is defined by a set of manually coded heuristics.
- *Strategist* keeps track of resource constraints (e.g. memory or time)
- *Tactician* orders proof actions, e.g.:
  - *Balanced Tactician* selects backward inference tactics in best-first manner, where tactics are scored by: tactic type, productivity (the more subgoals the worse), completeness / preference (estimating if leads to all true answers)
- Successful experiments in learning tacticians by reinforcement.

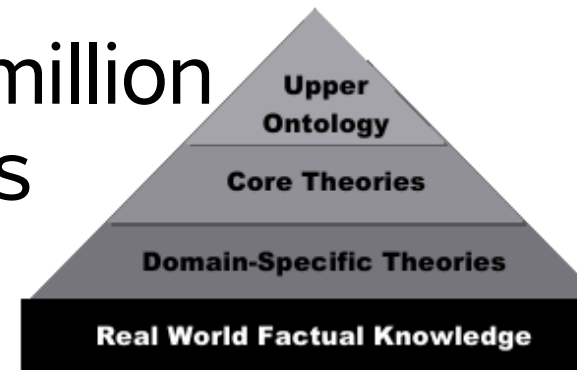
# Cyc logic (representation language)

- Has all logical connectives of FOPC.
- 5 truth values: monotonically false, default false, unknown, default true, and monotonically true.
- Default assertions can be overridden by entered or inferred knowledge.
- Argumentation: different proofs are compared, and one is selected basing on heuristics, e.g. that monotonic values are stronger than default.
- Justification chains of selected proofs are kept.



# Cyc logic (representation language)

- Microtheories (Mt) e.g. NormalPhysicalConditionsMt.
  - provide contexts of reference
  - form a hierarchy (actually, a directed graph); an assertion true in a Mt must be consistent with all Mts above it
  - Cyc currently contains about 4.6 million assertions in 23,627 microtheories
- Mechanisms for learning:
  - induction: rule production by generalization
  - abduction is done as deduction-in-reverse: generates hypotheses from unfinished proofs of known facts



Type : TRANS-PREDICATE-NEGATIONPREDS-NEG

Proven Query :

```
(ist
  (MtUnionFn UniverseDataMt SportsMt)
  (not
    (behaviorCapable PlanetEarth Marathon doneBy)))
```

Rule Assertion :

```
●M(implies
  (and
    (isa ?INS ?TYPE)
    (typeBehaviorIncapable ?TYPE ?SITTYPE ?ROLE))
  (behaviorIncapable ?INS ?SITTYPE ?ROLE)) in CapabilitiesMt
```

Rule Bindings :

```
?TYPE → InanimateObject
?INS → PlanetEarth
?SITTYPE → Marathon
?ROLE → doneBy
```

Additional Local Supports :

```
:NEGATIONPREDS (negationPreds behaviorIncapable behaviorCapable) in (MtUnionFn UniverseDataMt SportsMt)
```

Complete Proof Tree :

[Proof 6432.6] TRANS-PREDICATE-NEGATIONPREDS-NEG

```
●M(implies
  (and
    (isa ?INS ?TYPE)
    (typeBehaviorIncapable ?TYPE ?SITTYPE ?ROLE))
  (behaviorIncapable ?INS ?SITTYPE ?ROLE)) in CapabilitiesMt
:NEGATIONPREDS (negationPreds behaviorIncapable behaviorCapable) in (MtUnionFn UniverseDataMt SportsMt)
```

[Proof 6432.5] Join

[Proof 6432.4] REMOVAL-ALL-ISA

```
:ISA (isa PlanetEarth InanimateObject) in (MtUnionFn UniverseDataMt SportsMt)
```

[Proof 6432.2] REMOVAL-TVA-UNIFY

```
:TVA (typeBehaviorIncapable InanimateObject Marathon doneBy) in (MtUnionFn UniverseDataMt SportsMt)
```

**Fig. 2.** Partial Inference Tree for a proof that the planet Earth is not capable of running a marathon, supported by a proof that, more generally, inanimate objects can't run marathons

# Cyc Learning

- Gathering facts via Web Search:
  - Generating strings for web searches:
    - (occupation Lenat ?WHAT) ==> [“Lenat has been a \_\_\_\_”; “Doug Lenat has been a \_\_\_\_”; “Lenat is a \_\_\_\_”]
  - Identifying and interpreting a match
  - Eliminating bad interpretations: verifying against KB
  - Verifying the correctness by web-searching for nat-lang representation of constructed assertion
- Abduction: The candidate sentences suggested are checked (via inference and specialized well-formedness-checking modules) for consistency with the current knowledge in the KB. They are then evaluated for inferential productivity.
- Rule Induction: now can only help human experts, making their task 3 x faster.

# Cyc Learning

- Where to place new knowledge in an existing ontology?
- Research showed statistical classification (Naive Bayes and SVMs) have high (98% for SVMs) precision and recall success placing whole axioms in Mt hierarchy (but for a selected well-behaved fragment of the hierarchy)
- A bit similar to classification of text documents into a hierarchy, but: sparse data (e.g. *(isa Cat Mammal)* instead of a whole document) into deep hierarchy

# Principles Underlying Polyscheme

- **Procedural Substrate.** Most high-order reasoning and problem solving algorithms can be implemented using the same set of basic computational operation: forward inference, subgoaling, grounding, representing alternate worlds and identity matching.
- **Multiple representations.** Each basic operation can be implemented using multiple representations.
- **Representational Substrate.** Cognition about a basic set of relations (involving times, space, events, identity, causality and belief).

# Polyscheme Architectural Summary

- **Specialists:** modules based on a particular representation. Each executes each of the basic operations of the procedural substrate.
- **Integrative focus of cognitive attention.** All the specialists focus on the same aspect of the world simultaneously.
- **Attention control implements algorithms.** For example, the policy, *when uncertain about A, focus on the world where A and focus on the world where not-A* implements backtracking search.

# Polyscheme Integration

- **Integration of "high-level" and "low-level" cognition, perception and action.** All high-level reasoning and planning algorithms are implemented by a focus of attention that integrates all lower-level representations and perceptual and motor processes.
- **Integration of multiple higher-level cognitive processes with each other.** Very different reasoning algorithms, from truth-maintenance, backtracking search, stochastic simulation and logic theorem proving are each implemented using the same focus of attention.

# Polyscheme Unique Features

- **Higher-level basic services than most cognitive architectures:** reasoning about events, time, space, causality, identity, desire and beliefs (if the cognitive substrate principle is correct, is sufficient for reasoning in most domains)
- **No homunculus:** choose actions as the result of reasoning and problem-solving instead of a priori modeler decisions.
- **Only one model:** accounts of reasoning multiple tasks as part of the same model so that integration is an unavoidable and constant process and so that it is more difficult to gloss over hard problems.
- **Multiple representations.** Polyscheme does not commit users to a single representational formalism.
- **Language** is an important focus of Polyscheme modeling, which has historically (and with a few exceptions) not been the case in most other modeling communities.



# Grammatical Processing using Physical Inference

- construct a cognitive model of syntactic parsing that uses only the mechanisms required for infant physical reasoning
- Category(e, MotionEvent), Agent(e, x), Origin(e, p1), Destination(e, p2), Occurs(e, t), Before(t, t2), Meets(t2, t3), PartOf(e, e2), Subcategory(Fly, MotionEvent)
- Physical and verbal event perception both have a linear order.
- Utterances are events.
- Physical and linguistic events both belong to categories, which exist in hierarchies.

# Grammatical Processing using Physical Inference

- „the dog”: Category(e, CommonNounPhrase), Category(e1, Determiner), Occurs(e1, t1), Category(e2, CommonNoun), Occurs(e2, t2), PartOf(e1,e), PartOf(e2,e), Meets(e1,e2)
- Category(verb, TransitiveVerb) + Occurs(verb, t-verb) ==> Exists(object) + Category(object, NounPhrase) + Occurs(object, t-object) + Before(t-verb, t-object)
- object identity: e.g. looking at sth another time
- event identity: e.g. sth falls from the shelf; there are marks of cat claws on the shelf; pushing event = cat walking event?
- object permanence: reasoning about sth not perceived
- cohesion principle, part inhibition, c-command:  
*[The doctor **[Mary]** met at **[[Bill]'s house]**] likes herself.*

# Grammatical Processing using Physical Inference

<b>Grammatical Structure</b>	<b>Cognitive structure</b>
Word, phrase, sentence	Event
Constituency	Meronomy
Phrase structure constraints	Constraints among (parts of) events
Word/phrase category	Event category
Word/phrase order	Temporal order
Phrase attachment	Event identity
Coreference/binding	Object identity
Traces	Object permanence
Short- and long-distance dependencies	Apparent motion and long paths.

**Table 1.** Dualities between elements of physical and grammatical structure.

# Adaptive AI (a2i2)

- **General** rather than domain-specific cognitive ability
- **Acquired knowledge** and skills, versus loaded databases and coded skills
- **Bi-directional, real-time** interaction, versus batch processing
- **Adaptive attention** (focus & selection), versus human pre-selected data
- Core support for **dynamic patterns**, versus static data
- Unsupervised and **self-supervised**, versus supervised learning

# Adaptive AI (a2i2)

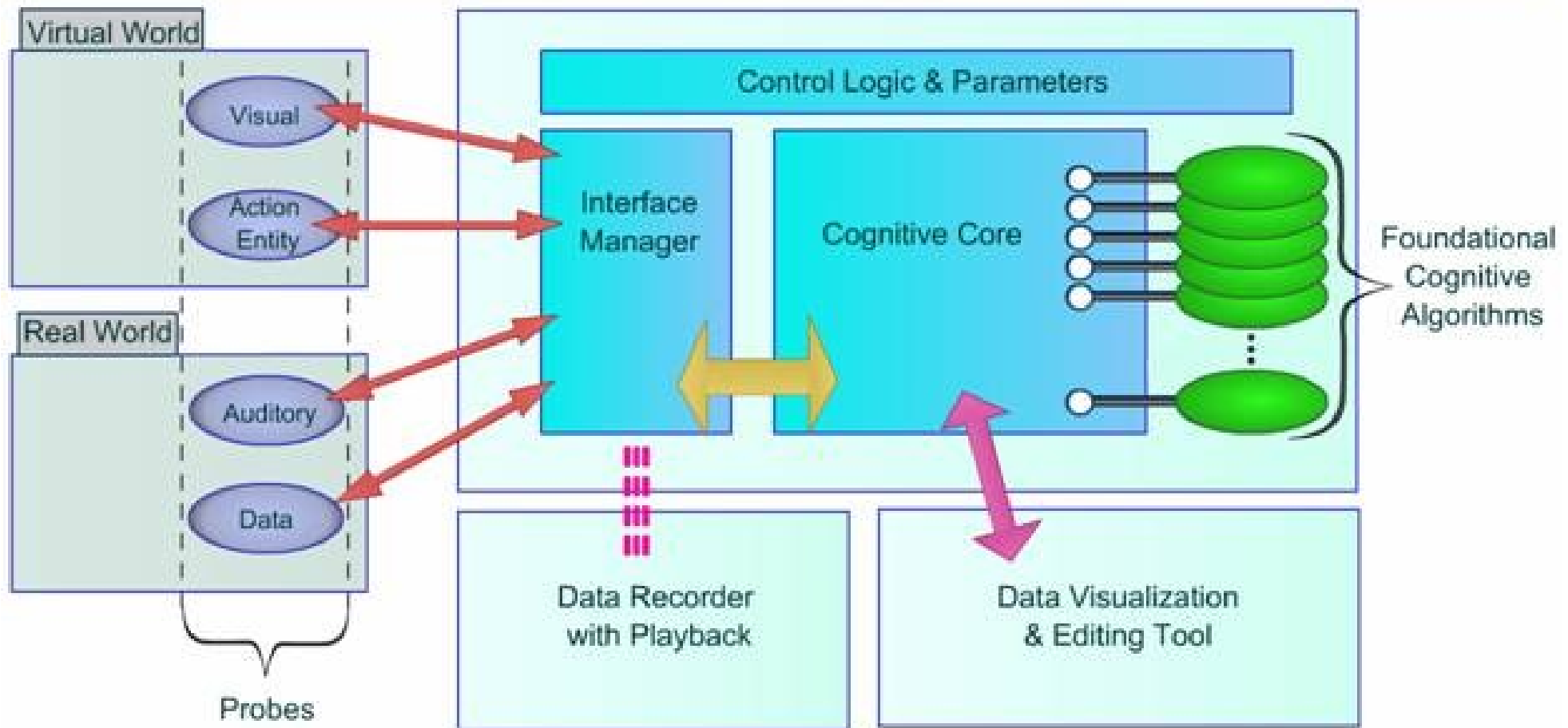
- Adaptive, **self-organizing data structures**, versus fixed neural nets or databases
- **Contextual, grounded concepts**, versus hard-coded, symbolic concepts
- Explicitly **engineering** functionality, versus evolving it
- **Conceptual design**, versus reverse-engineering
- **General proof-of-concept**, versus specific real applications development
- **Animal level cognition**, versus abstract thought, language, and formal logic.

# Adaptive AI (a2i2)

- Pattern learning, matching, completion, and recall.
- Data accumulation and forgetting.
- Categorization and clustering.
- Pattern hierarchies and associations.
- Pattern priming and activation spreading. (helps at disambiguation)
- Action patterns. (feature extractors, actuators, meta-cognition)

# Adaptive AI (a2i2)

## Adaptive A.I.'s AGI Framework



# Adaptive AI (a2i2)

- embodied systems (Brooks 1994),
- vector encoded representation (Churchland 1995),
- adaptive self-organizing neural nets (esp. Growing Neural Gas, Fritzke 1995),
- unsupervised and self-supervised learning,
- perceptual learning (Goldstone 1998),
- fuzzy logic (Kosko 1997)
- lang = C#



# Lazy Learning (a2i2)

- (Aha, 1997) A memory-based technique that postpones all the computation until an explicit request for a prediction is received. The examples considered relevant according to a distance measure are interpolated locally.

Learning of:

- a family of local approximators
- parameters of the local approximator
- a metric to evaluate which examples are more relevant
- bandwidth which indicates the size of the region correctly modeled by local approximator family

# Growing Neural Gas (a2i2)

- Grows a topological map by processing input one at a time
  - select two nodes  $s$  (at  $w_s$ ) closest and  $t$  (at  $w_t$ ) second-closest to the input  $x$
  - $\text{error}_s += |w_s - x|$ ,  $w_s += ew * (x - w_s)$
  - $w_n += en * (x - w_n)$  for all  $n$  joined with  $s$  (neighbors)
  - connect  $s$  and  $t$  with edge (or set edge age to 0)
  - remove too old edges
  - from time to time, add new node between largest error node  $u$  and its largest error neighbor  $v$

# Vector Encoded Representation (a2i2)

- Ideas from P. Churchland: functions of the brain are represented in multidimensional spaces, neural networks should therefore be treated as “geometrical objects”, and “the internal language of the brain is vectorial”,
- thinking is the changing of activation vectors by matrix multiplication and nonlinear transformations.
- In a2i2: all concepts are grounded; no high-level inference mechanisms; no genetic programming, scripts, or other direct symbolic representation.

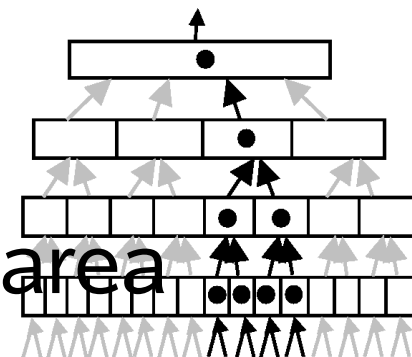
# Hierarchical Temporal Memory

- hierarchical in both time and space
- HTMs are similar to Bayesian Networks; differ by handling time, hierarchy, self-training, discovery of causes, action, attention.
  - Discover causes in the world
  - Infer causes of novel input
  - Make predictions
  - Direct behavior
- inputs are topologically arrayed and must be continuous in time; the input space is tiled
- hierarchical bottom-up Bayes-like net (internal nodes are called „causes”) by conditioning in time

# Hierarchical Temporal Memory

- each node in a layer of a hierarchy learns causes from nearby nodes in the layer below
- each node has a fixed-size Bayes-like net whose variables represent sequences in input
- it quantizes input and assigns a probability of occurring in each variable's sequence
- nodes pass quantization (clustering) info down the hierarchy; (input-output discriminant-like clustering, coalescing and expansion of time-based sequences)
- the nodes-tree structure is fixed
- belief propagation can enter cycles (high fan-in and fan-out of variables reduces reinforcement of false beliefs)
- each node dynamically selects believed sequence based on memory (mapping from spatial to temporal patterns)

# Hierarchical Temporal Memory



- covert attention by switching on only some area
- attentional priming by setting a desired belief at the top of the hierarchy (directed search)
- predictions: node sequences generate distribution of expected patterns and pass it down as a prior
- each node has a single conditional probab. table
- fixed architecture allows for continuous change
- each variable is associated with one quantization point (so there is a fixed number of each input occurrences in learned sequences)
- cannot remember specific events (no one-instance learning) (but expected to be added and based on „emotions”)

# Developmental Robotics

- No Monolithic Internal Models
  - people minimise internal representations
  - and have multiple internal representations not mutually consistent
- No Monolithic Control (e.g. split brain patients)
- Not General Purpose (e.g. emotional content)
- Alternate Essences
  - development, social interaction, physical interaction and integration

# Developmental Robotics

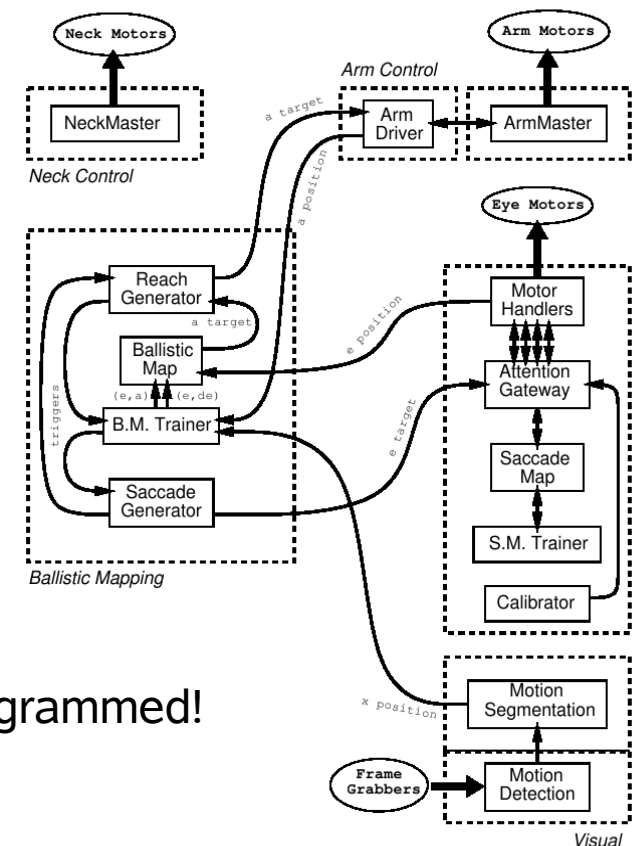
- Developmental Organization
  - A process in which the acuity of both sensory and motor systems are gradually increased significantly reduces the difficulty of the learning problem.
  - The caregiver also acts to gradually increase the task complexity by structuring and controlling the complexity of the environment.
  - Reusing structures and information gained from previously learned behaviors, allows to learn increasingly sophisticated behaviors.
  - Development gives a structured decomposition (situated context of earlier behavior).



# Cog learning to reach target with arm

1. Initialize the maps  $\vec{B}$  and  $\vec{F}$ .
2. Randomly select a visual target.
3. Saccade to that target using the learned saccade map.
4. Attempt to reach for the target using the current ballistic mapping.
5. As the arm moves, track the end of the arm using visual motion detection.
6. Once the arm has finished reaching, compute the visual error signal  $\vec{x}$  between the center of the visual field and the arm position.
7. Transform the visual error from image coordinates  $\vec{x}$  to eye motor coordinates  $\vec{e}$  using the saccade map.
8. Backpropagate  $\vec{e}$  through the forward map  $\vec{F}$  to generate an error signal in terms of the arm coordinates  $\vec{a}$ .
9. Backpropagate the error in arm coordinates  $\vec{a}$  through the Ballistic map.
10. Return the arm to the resting position.
11. Go to step 2.

Hand-programmed!



# Requirements for Autonomous Mental Development

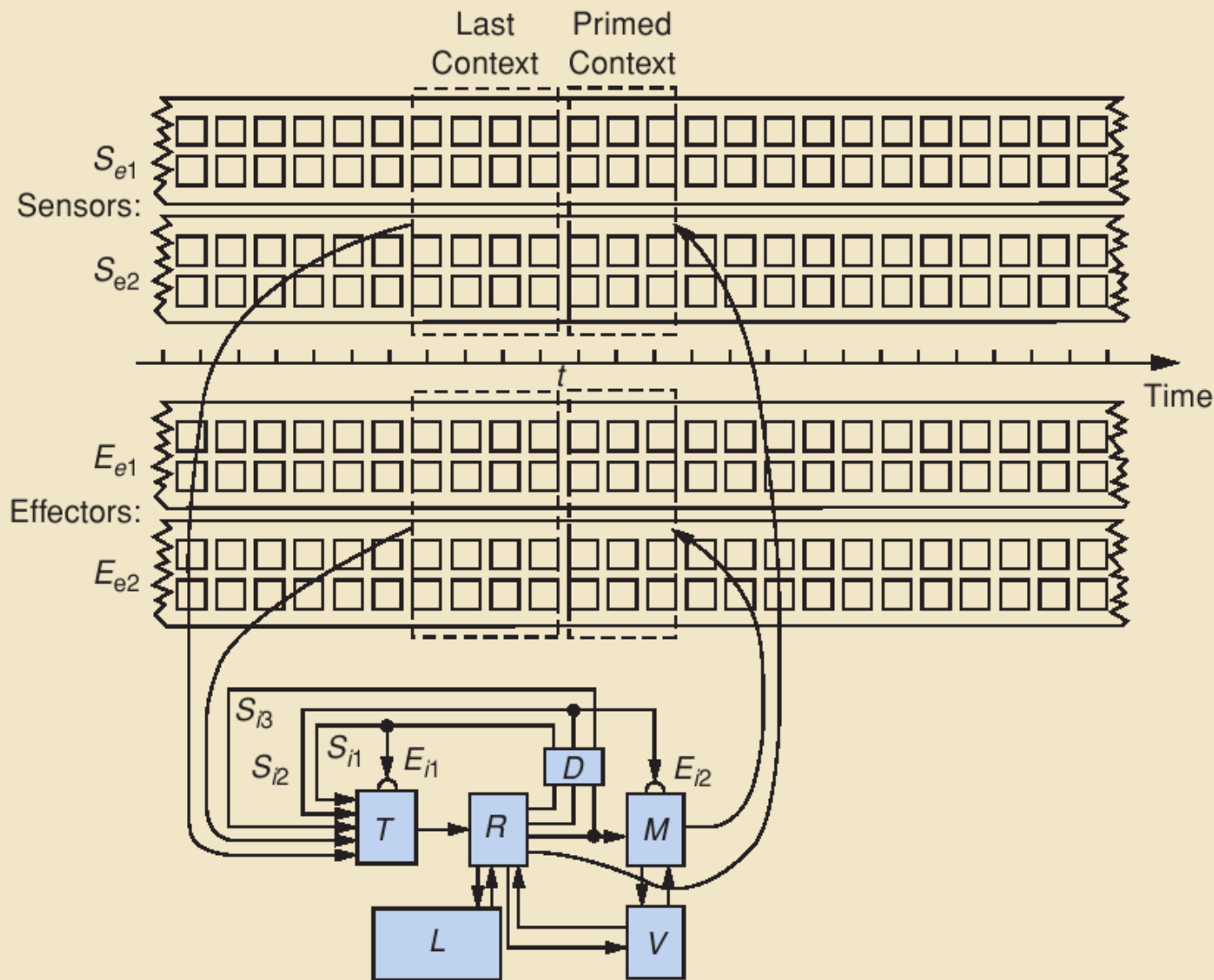
- **Environmental openness:** AMD must deal with unknown and uncontrolled environments.
- **High-dimensional sensors:** AMD must directly deal with continuous raw signals from high-dimensional sensors (e.g., vision, audition and tactition).
- **Completeness in using sensory information:** don't discard, at the program design stage, sensory information that may be useful for some future, unknown tasks.

# Requirements for AMD

- **Online processing:** At each time instant, what the machine will sense next depends on what the machine does now.
- **Real-time speed:** The sensory/memory refreshing rate must be high enough (e.g., about 15Hz for vision). AMD must handle learning from one instance of experience.
- **Incremental processing:** Acquired skills must be used to assist in the acquisition of new skills, as a form of ``scaffolding." Each new observation must update the current complex representation and the raw sensory data must be discarded after it is used for updating.

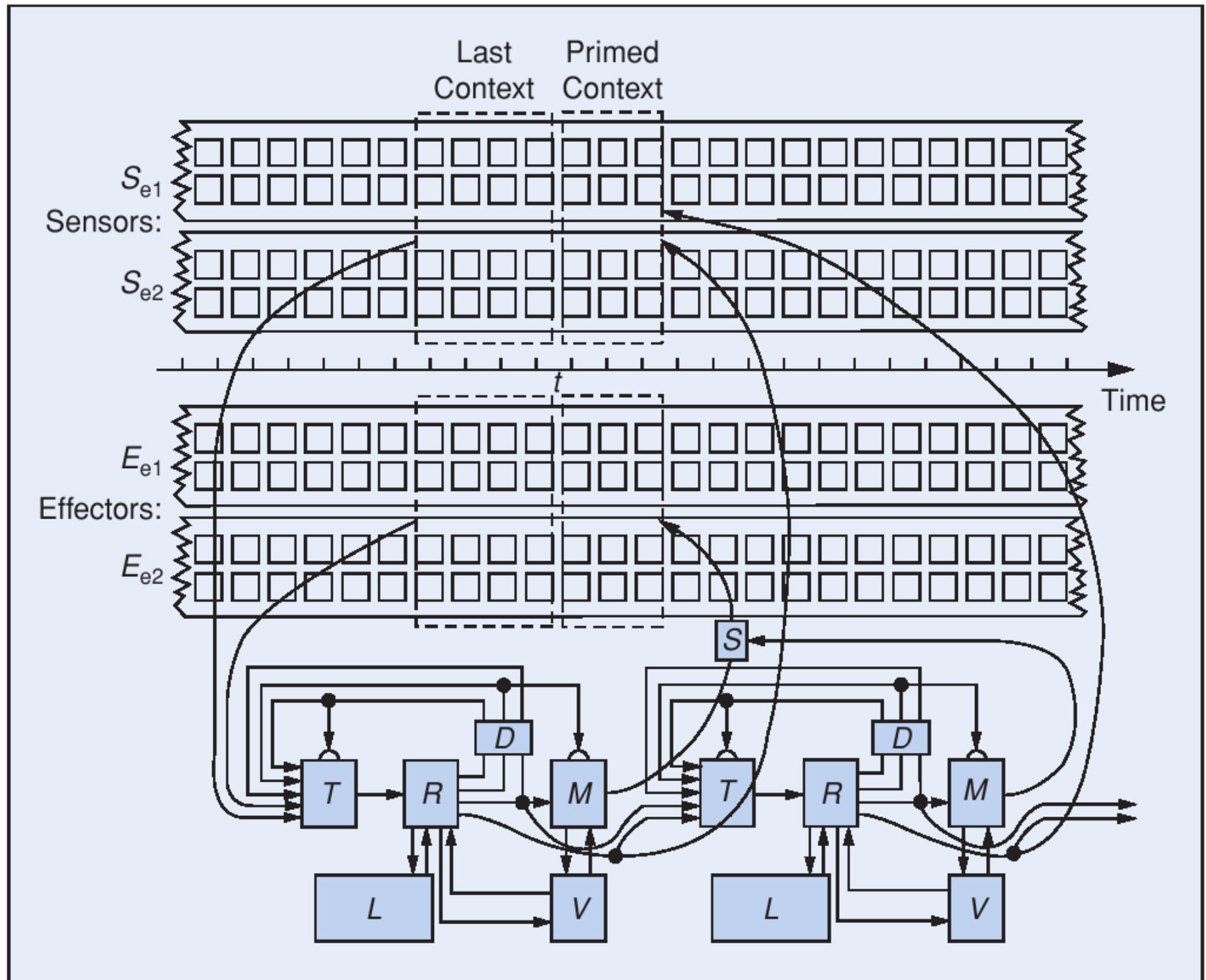
# Requirements for AMD

- **Perform while learning:** An AMD machine must perform while it ``builds" itself "mentally."
- **Scale up to large memory:** For large perceptual and cognitive tasks, an AMD machine must handle multimodal contexts, large long-term memory and generalization, and capabilities for increasing maturity, all in real time speed.
- Mental architecture should be:
  - observation driven, selective, rehearsable, self-ware, self-effecting, multi-level and developmental.



- T – attention selector
- R – learned possible actions (Incremental Hierarchical Discriminant Regression)
- V – action values (motivational system)
- L – context clusterization (prototypes)
- M – motor mapping (allows for rehearsal)
- D – delay module
- S<sub>i1</sub>, S<sub>i2</sub>, S<sub>i3</sub> – internal sensors
- E<sub>i1</sub>, E<sub>i2</sub> – internal effectors

**FIGURE 2** Progressive additions of architecture components from Type-2 to Type-5. Type-2: adding attention selector  $T$  and its (internal) control input  $E_{i1}$ . Type-3: Adding motor mapping  $M$  and its (internal) control  $E_{i2}$ . Type-4: Adding internal controls  $S_{i1}$  and  $S_{i2}$  and the primed sensation  $S_{i3}$  to the entry port of perception  $T$ . The block marked with  $D$  is a delay module, which introduces a unit-time delay for the corresponding vector. Type-5: Developmental  $T$ ,  $R$ ,  $M$  and  $V$ .



**FIGURE 4** The Type-6 architecture: multi-level DOSASE MDP.

# Incremental Hierarchical Discriminant Regression

(in CompSci terms, a search tree)

- IHDR automatically derives the most discriminant features subspaces in every node of the tree.
- One-instance learning is realized by either a new prototype (when sufficiently distinct), or a similar prototype.
- IHDR dynamically grows subtrees to adapt to the increased complexity.
- IHDR uses a data-driven coarse-to-fine search tree to contain the local minima problem.
- The long-term memory stored as tree structure and micro-prototypes prevent catastrophic memory loss.
- Given a input vector  $l(t)$ , the time complexity for IHDR to find a match and update the IHDR tree with  $n$  leaf nodes is  $O(d \log(n))$  where  $d$  is the constant dimension of  $l(t)$ .

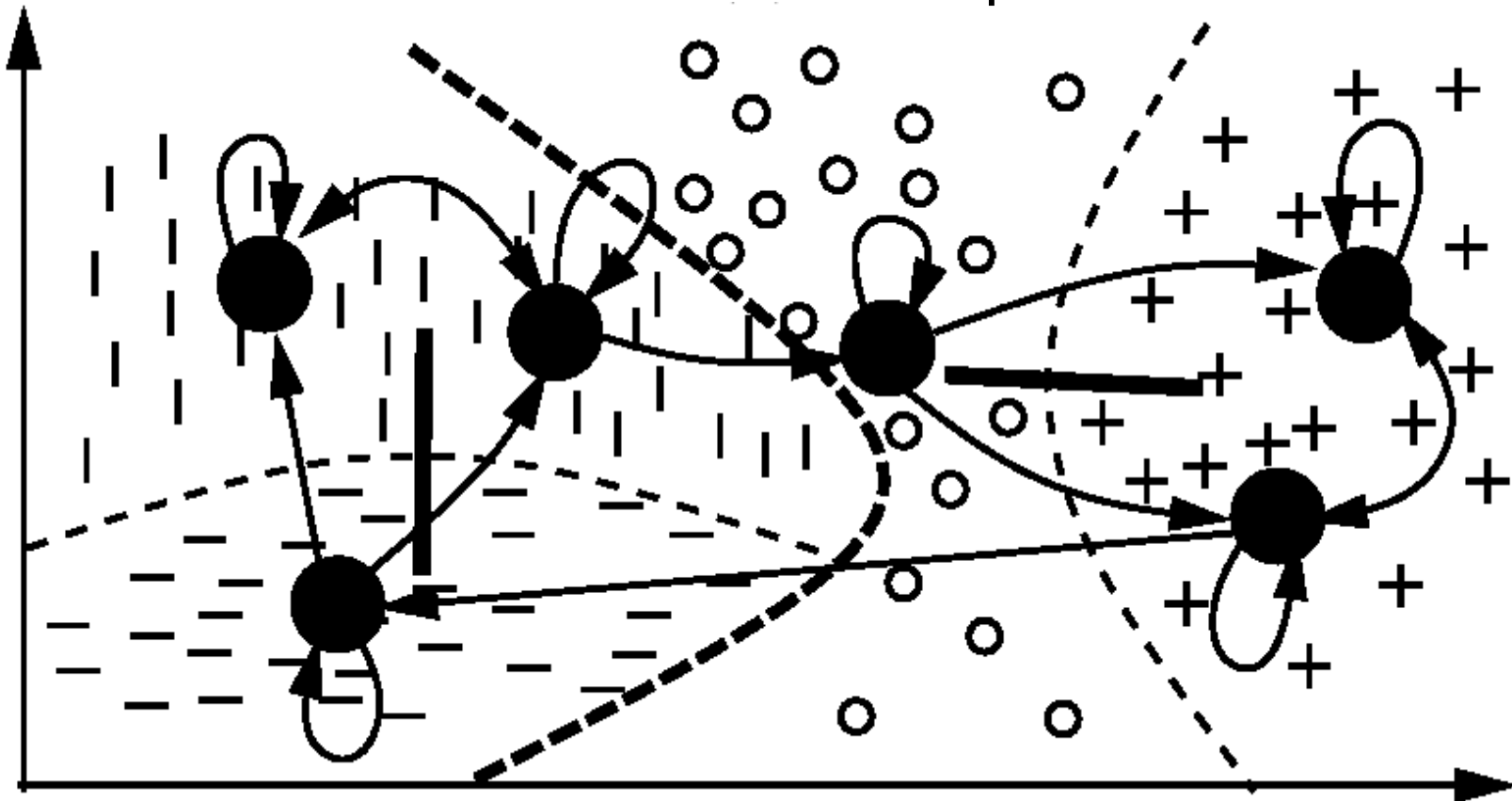
# (SAIL) IHDR Details

- clustering of output space provides labels for discriminant analysis (allows disregard input components irrelevant to output)
- each node of the tree clusters input and output (local-hierarchical probability distribution approximation; clusters represented by f.o. statistics – Gaussian mixture)
- Mahalanobis distance (Euclidean when little of samples) – which cluster to descend
- fully incremental: updated with every input vector which is then discarded
- Observation-driven Markov Decision Process (time invariants)



# (SAIL) IHDR Details

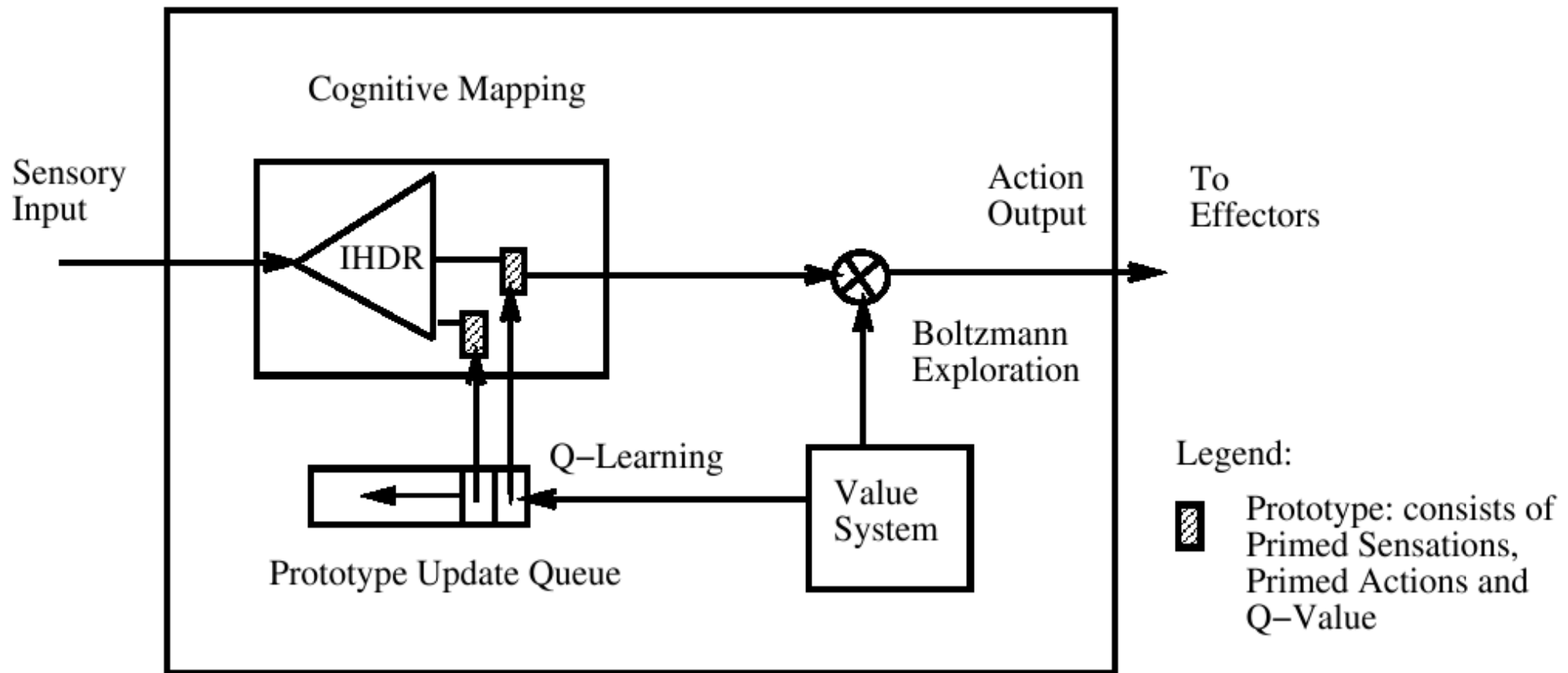
A solid black cycle indicates a primitive prototype (context state) in one of the four leaf nodes. An arrow between two states indicates observed temporal transitions.



# SAIL

- Supervised and reinforcement online learning: pressure detectors let the teacher push the robot in desired directions, „good – bad” buttons.
- With no teacher feedback, the robot acts from context-sensitive memory.
- Has learned real-time vision-guided navigation in complex indoor environment.

# SAIL



Cognitive mapping  $M : S * X \rightarrow X' * A * Q$

- $S$  – state (context)
- $X$  – sensory input (sensation)
- $X'$  – primed sensation (prognostic)

**prototype**: a set of primed contexts  
**primed context**: action  $a$ , expected sensation  $s$ , expected value  $Q(a, s)$

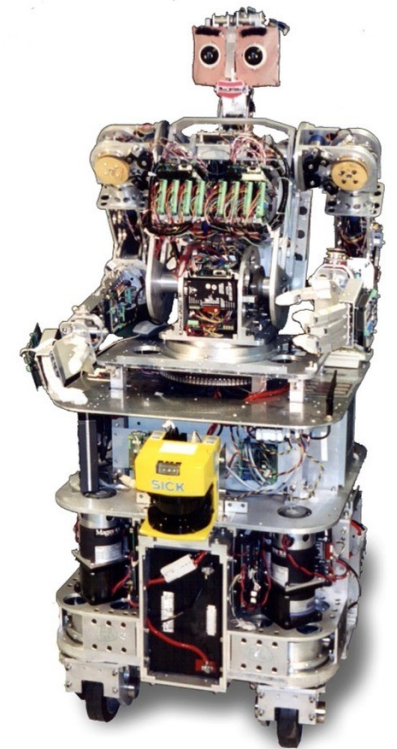
probability of choosing action  $a$  in context  $s$ :

$$p(s, a) = \frac{e^{\frac{Q(s, a)}{\theta}}}{\sum_{a' \in A(s)} e^{\frac{Q(s, a')}{\theta}}}$$

# SAIL

- **novelty**  $n(t)$ : error of prediction normalized wrt. speed of change
- combined reward:  $r(t) = \alpha p(t) + \beta r(t) + (1 - \alpha - \beta)n(t)$
- Q-learning update:
  - $Q(x_p(t), a_p(t)) := (1 - \alpha)Q(x_p(t), a_p(t)) + \alpha(r(t + 1) + \gamma \max_{a'} Q(x_p(t + 1), a_p(t + 1)))$
- prediction update:
  - $x^{(n)}(t) := x^{(n-1)}(t) + \frac{1 + l}{n} \gamma (x(t + 1) - x^{(n-1)}(t))$
- updates are backpropagated through a fixed-length queue of recent contexts (Q-algorithm would update all contexts)

# SAIL (and Dav)



# SAIL Action Chaining

# SAIL Action Chaining

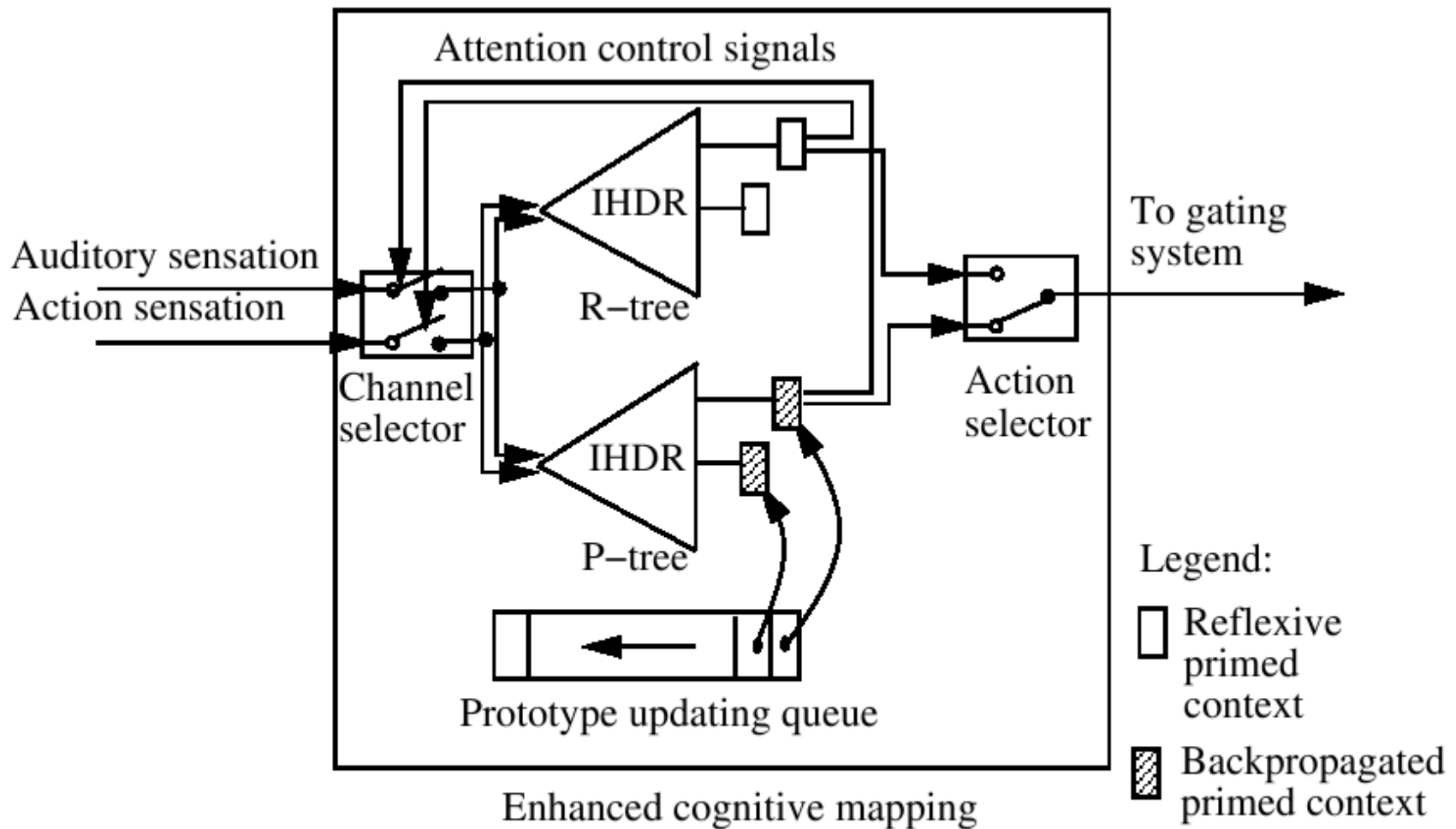


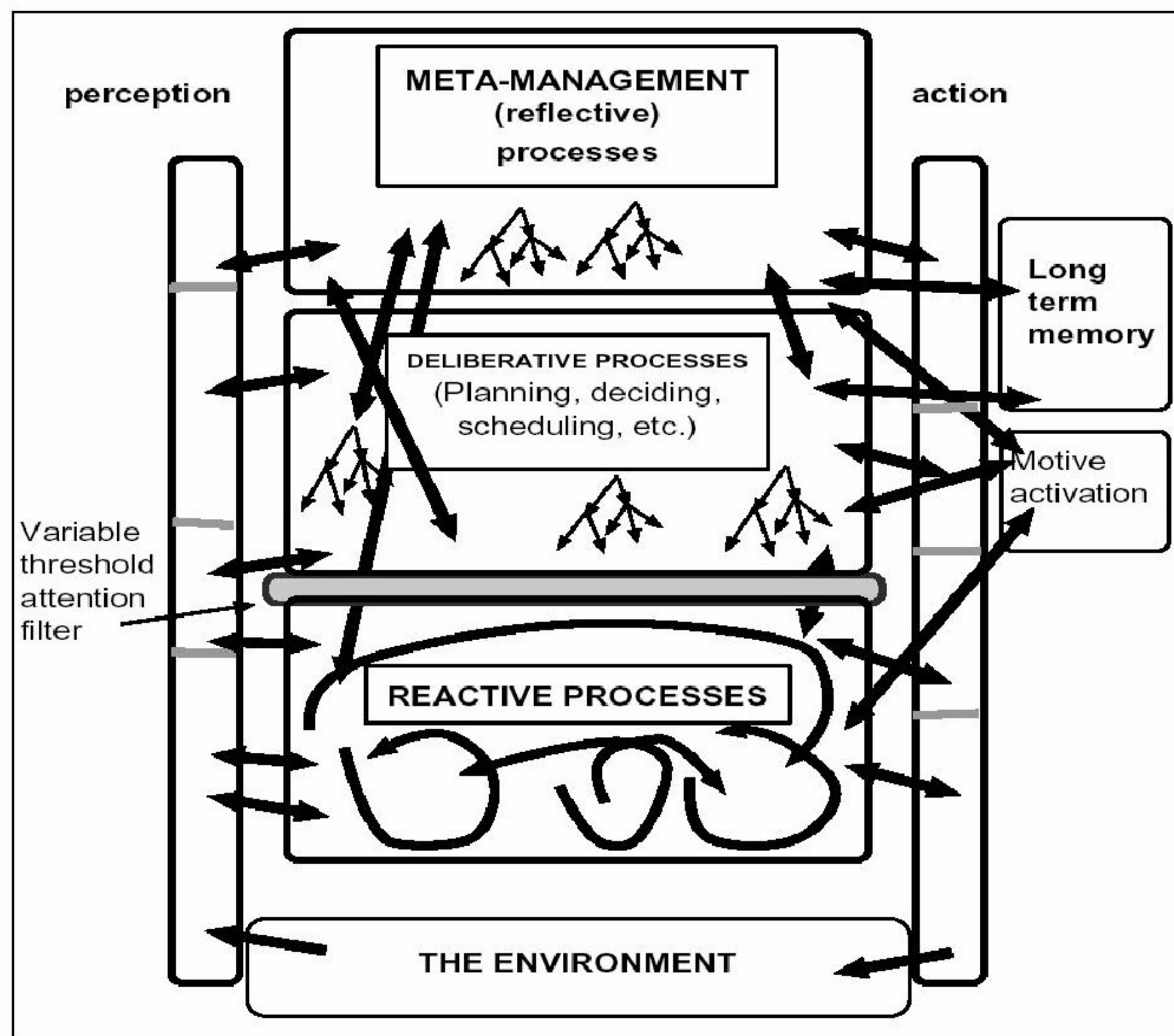
Figure 2: A double-tree system.

IDA

general

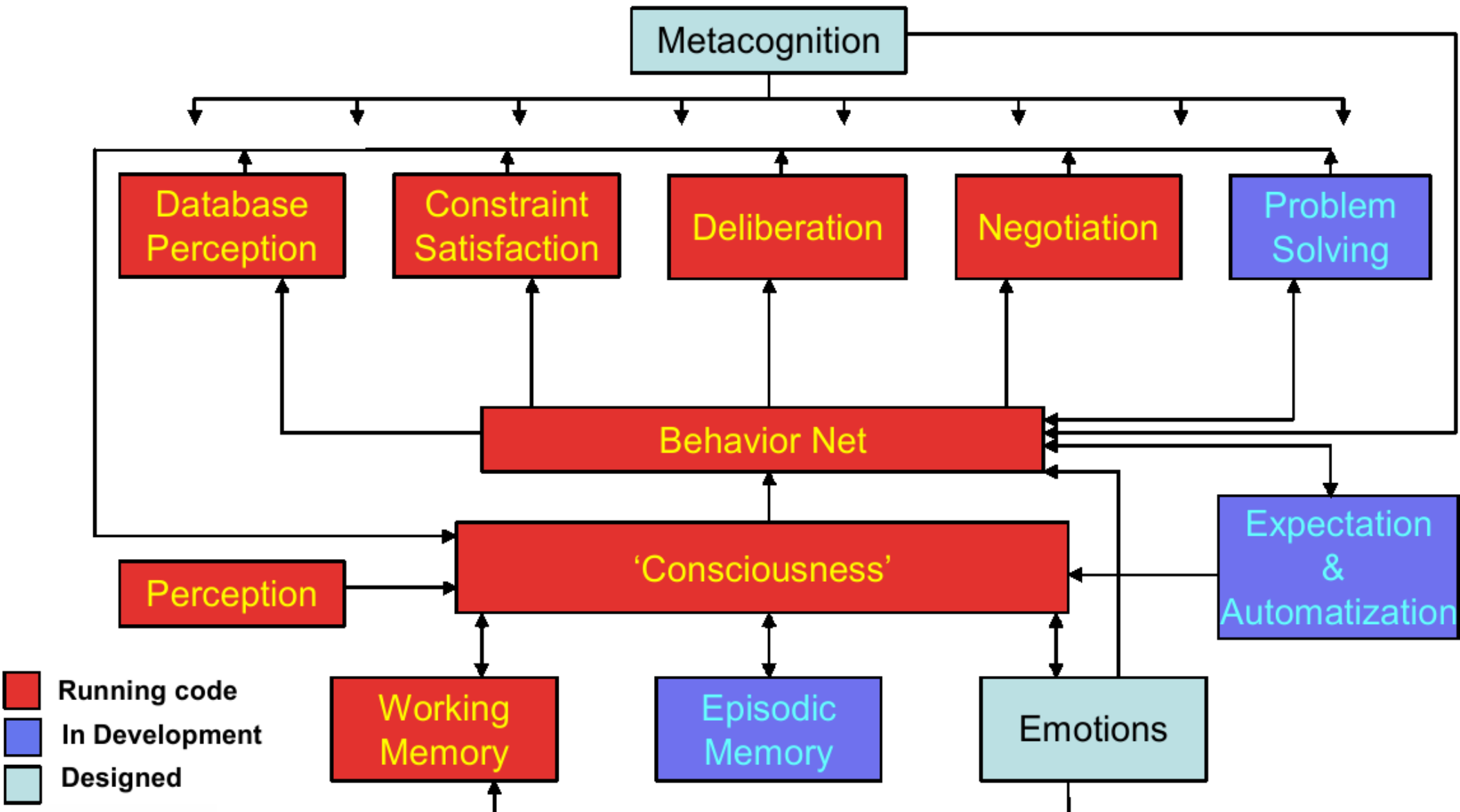
cognitive

architecture





# IDA's Architecture



# IDA'S Modules and Mechanisms

- Perception—Copycat Architecture—Hofstadter
- Action Selection—Behavior Net—Maes
- Episodic Memory—Sparse Distributed Memory—Kanerva
- Emotions—Pandemonium Theory—Jackson
- Metacognition—Fuzzy Classifier Systems—Holland
- Learning—Copycat Architecture, Reinforcement
- Constraint Satisfaction—Linear Functional
- Language Generation—Pandemonium Theory
- Deliberation—Pandemonium Theory
- 'Consciousness' —Pandemonium Theory

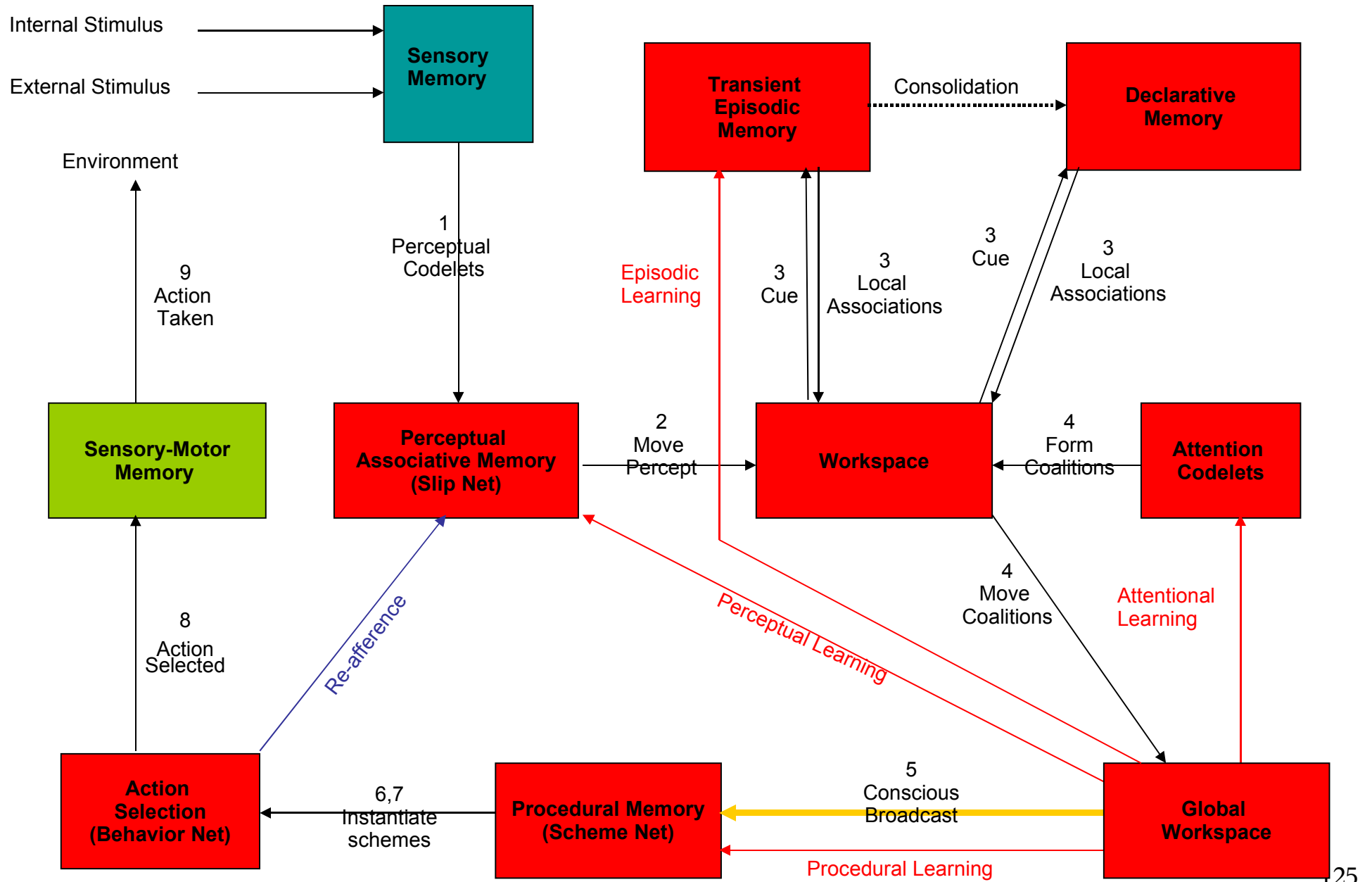
# (IDA) Action Selection Paradigm of Mind

- Best viewed as degreed rather than as Boolean
- Aggregate rather than monolithic
- Enabled by disparate mechanisms
- Overriding task to produce the next action
- Operates on sensations to create information
- Reconstructs memories (prior information)
- Is implementable on machines

# (IDA) Cognitive Cycle Processing

- **Hypothesis:** Like IDA's, human cognitive processing is via a continuing sequence of Cognitive Cycles
- **Duration:** Each cognitive cycle takes roughly 200 ms with steps 1 through 5 occupying about 80 ms
- **Overlapping:** Several cycles may have parts running simultaneously in parallel
- **Seriality:** Consciousness maintains serial order and the illusion of continuity
- **Start:** with perception or action selection

# IDA



# (IDA) Virtual Machine on a Brain

- Entities include qualia, objects, categories, feelings, intentions, internal images, internal speech, etc.
- Relations include cause, before, on top of, isa, is not, can drink from, etc.
- Processes include perception, memory, action selection, learning, etc.
- Note the partial ontology just created.
- Cognition: the endless cycle of deciding what to do next.

# IDA finds jobs for sailors

- Communicates with sailors in English via email
- Selects jobs to offer a sailor, taking into account
  - the Navy's policies and needs
  - the sailor's preferences
- Deliberates about feasible dates
- Negotiates with the sailor about job selection over the course of several emails

# IDA Consciousness

- What's in the spotlight
- Limited capacity
- Coalition of codelets
- Message from these codelets broadcast to all other codelets
- “...serves to disseminate a small amount of information to a vast unconscious audience...”
- “The payoff for limited capacity seems to be vast access.”

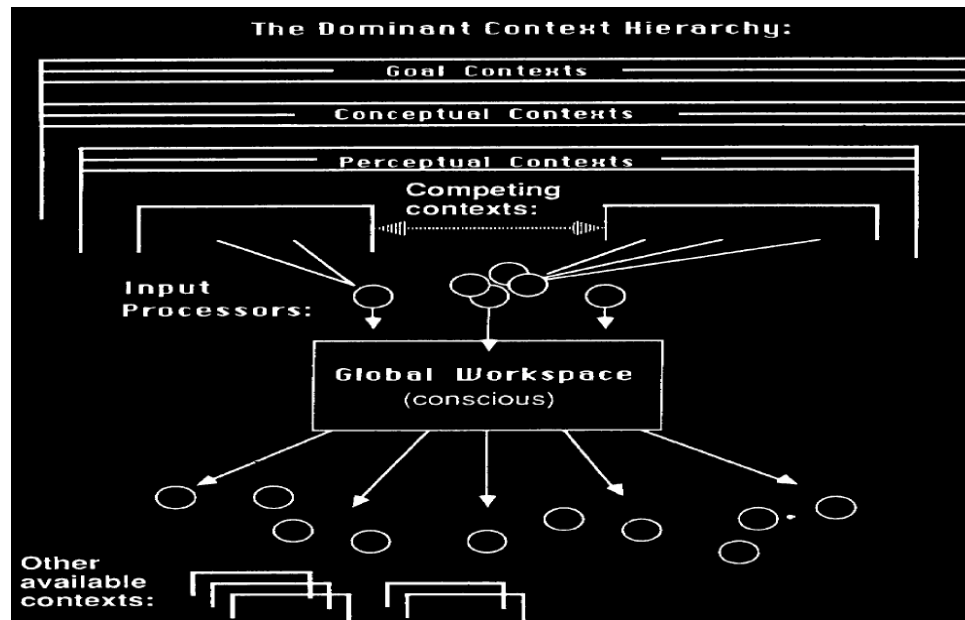


# (IDA) Tickets to the Spotlight

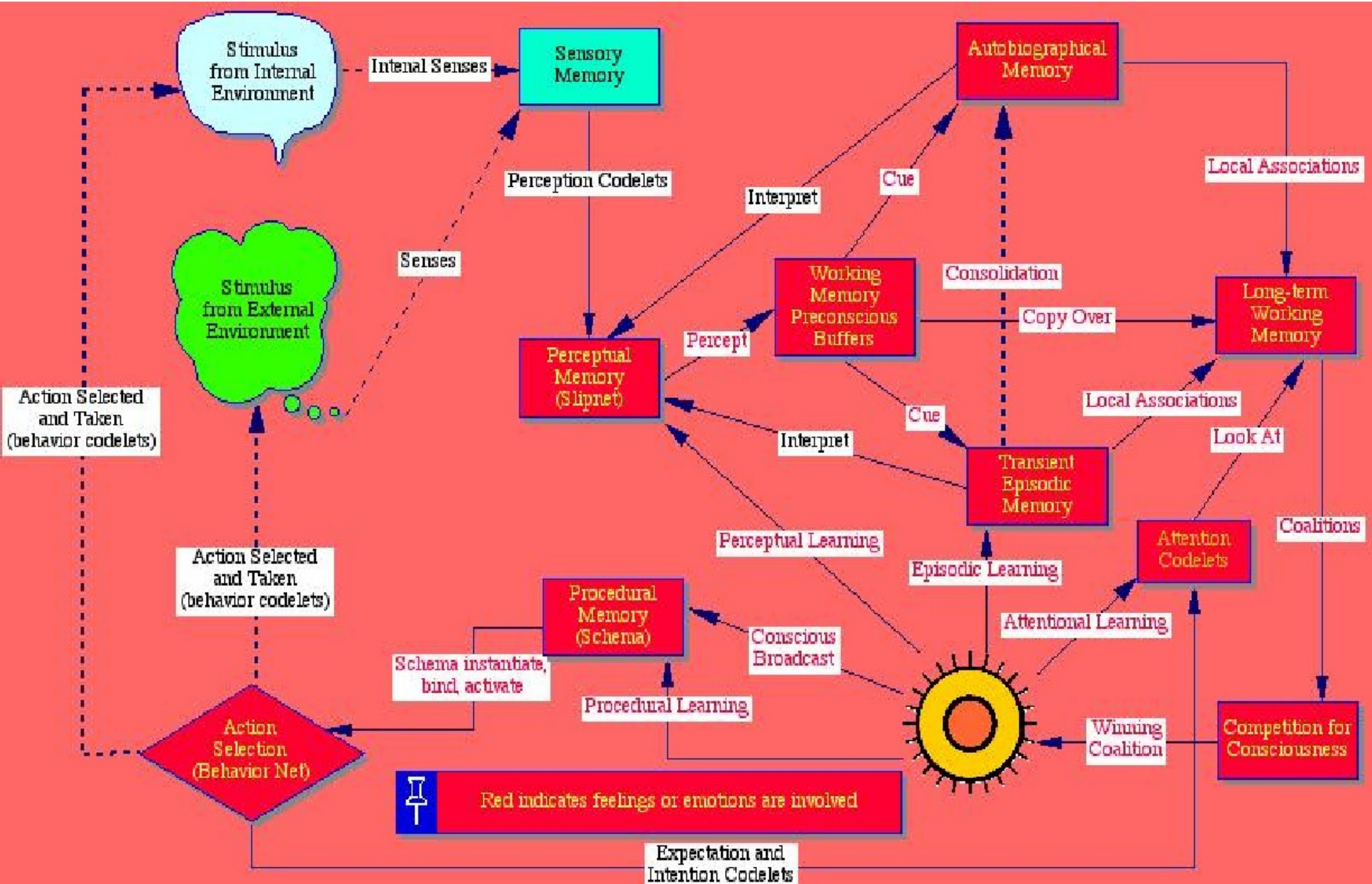
- Novelty, Relevancy, Informativeness
- Problems, Inconsistency, Violated expectations
- Whatever can't be dealt with by unconscious, automatic processors.
- Conscious imagery and inner speech allow metacognitive reflection and control
- Self-system maintains stability under changing internal and external conditions

# (IDA) Contexts

- coalitions of processors (codelets)
- include unconscious expectations and intentions
- Similar to but not the same as: Frames, Scripts, Schemas, Semantic nets
- Perceptual, Conceptual, Goal, Cultural



# IDA



# (IDA) Slipnet = Simple Activation Net

- Directed graph of nodes, representing concepts, and labeled links
- Links represent relations between nodes
- Nodes support activation, links pass it
- Slipnet does not learn
- Nodes don't decay, but activation does
- Slipnet is long-term memory
- Temperature control of stability (temperature inversely measures understanding of situation)
- Activation passes from node to node until the slipnet stabilizes
- All slipnet nodes are feature detectors

# (IDA) Schema = Hierarch. Behavior Net

- Triple: context (makes schema more likely), action, result (should be more likely after action)
- Spin-off schemas built when a relation between items and actions is discovered
- Composite actions (implemented by schemas) coordinated to achieve some goal
- Synthetic item — a state not expressible as some combination of current states
- Synthetic items permit the invention of radically new concepts, for example conservation
- Schemas keep track of reliability statistically

# (IDA) Finding Reliable Schema

- Schema mechanism looks for results that follow from actions, reliably or not
- If a result follows unreliably, the mechanism looks for added context to improve reliability
- When successful, it spins off a new schema adding the newly discovered context to a copy of the old schema
- Plan—a set of reliable schemas coordinated to achieve some specified result.

<u>Preconditions</u> i k l m	Competence Module x	<u>add list</u> a b c	<u>delete list</u> l n
	Activation		

(IDA)  
Strict

Behavi  
or

Nets

- do not learn
- hand-coded

<u>Preconditions</u> b c d	Competence Module y	<u>add list</u> a n	<u>delete list</u> l m
	Activation		

successor links  
predecessor links  
conflictor links



# Behavior Net „Fuzzy” Plans

- Sequence of competencies transform present situation into desired one
- Sequence can become highly activated by forward spreading from current state & backward spreading from a goal state
- May occur in competition with other sequences striving towards other goals
- In LIDA (strict) behavior nets are not learnable



# IDA Codelets

- Small pieces of code each performing a simple, specialized task
- Many watch for a chance to act
- Most subserve some high level entity, e.g. behavior, slipnet node
- Some codelets work on their own, e.g.
  - watching for incoming mail
  - checking for time and place conflicts
- Specialized perception codelets find features and activate appropriate nodes in the slipnet

# IDA Sparse Distributed Memory

- random access (constant time)
- similar to a mix of Hopfield nets and self organizing maps, based on binary code vectors and Hamming distance, input space = output space
- writing: nearby code vectors move their source vectors towards input
- reading: fixpoint on coordinate-wise majority rule reading the sources of nearby code vectors

# NARS Methodology

- Minimalism: not to maximize the system's performance, but to minimize its theoretical assumptions and technical instruments, while still achieving desired performance.
- There are scientific and engineering reasons for following a unified approach.
- Many such attempts have failed, but they might have followed wrong ideas. (General Problem Solver, Fifth Generation Computers)
- The system should:
  - rely on constant processing capacity,
  - be open to unexpected tasks,
  - learn from experience.

# NARS Semantics

- The truth value of a sentence is determined by available evidence in the experience:

$$F = W^+/W, \quad C = W/(W+1)$$

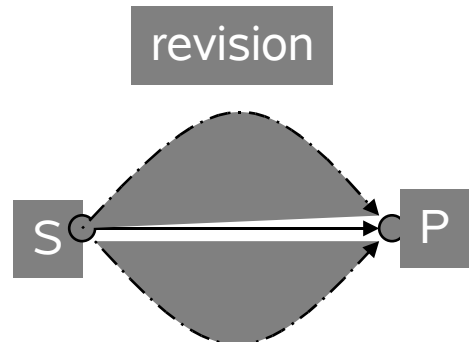
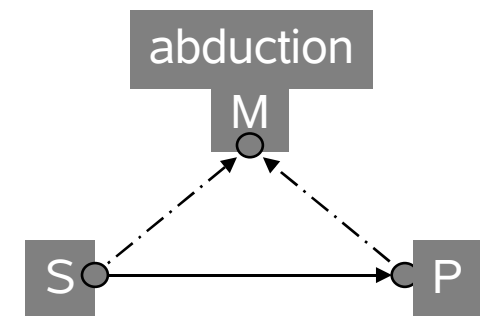
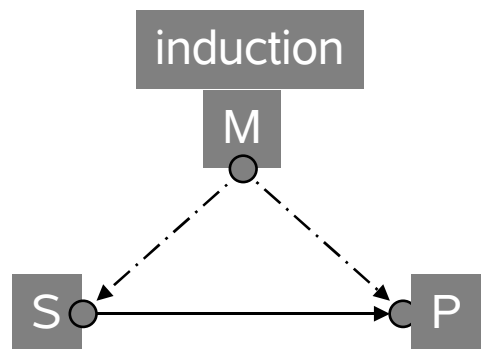
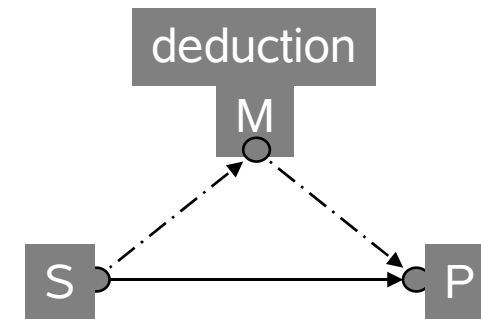
- Truth value uniformly represents randomness, fuzziness, and ignorance.
- The meaning of a term is defined by its experienced relations with other terms.

bird  $\rightarrow$  animal [1.0, 0.9]

subject  $\rightarrow$  predicate  
[frequency, confidence]

extension of subject inside  
extension of predicate

intension of predicate inside  
intension of subject



# NARS Semantics

- *complete inheritance*: base case when  $F=C=1$
- extension and intension:
  - $E_T = \{x | x \sqsubset T\}$  and  $I_T = \{x | T \sqsubset x\}$
- when premises are complete inh.:
  - $W^+ = |E_S \cap E_P| + |I_P \cap I_S|$ ,  $W = |E_S| + |I_P|$
- generalized to truth values

## DEDUCTION

$$\begin{array}{l} M \subset P \langle F_1, C_1 \rangle \\ S \subset M \langle F_2, C_2 \rangle \end{array}$$

---


$$S \subset P \langle F_d, C_d \rangle$$

$$\begin{array}{l} F_d = F_1 F_2 / (F_1 + F_2 - F_1 F_2) \\ C_d = C_1 C_2 (F_1 + F_2 - F_1 F_2) \end{array}$$

## INDUCTION

$$\begin{array}{l} M \subset P \langle F_1, C_1 \rangle \\ M \subset S \langle F_2, C_2 \rangle \end{array}$$

---


$$S \subset P \langle F_i, C_i \rangle$$

$$\begin{array}{l} F_i = F_1 \\ C_i = F_2 C_1 C_2 / (F_2 C_1 C_2 + 1) \end{array}$$

## REVISION

$$\begin{array}{l} S \subset P \langle F_1, C_1 \rangle \\ S \subset P \langle F_2, C_2 \rangle \end{array}$$

---


$$S \subset P \langle F_r, C_r \rangle$$

$$\begin{array}{l} F_r = \frac{F_1 C_1 (1 - C_2) + F_2 C_2 (1 - C_1)}{C_1 (1 - C_2) + C_2 (1 - C_1)} \\ C_r = \frac{C_1 (1 - C_2) + C_2 (1 - C_1)}{C_1 (1 - C_2) + C_2 (1 - C_1) + (1 - C_1)(1 - C_2)} \end{array}$$

## ABDUCTION

$$\begin{array}{l} P \subset M \langle F_1, C_1 \rangle \\ S \subset M \langle F_2, C_2 \rangle \end{array}$$

---


$$S \subset P \langle F_a, C_a \rangle$$

$$\begin{array}{l} F_a = F_2 \\ C_a = F_1 C_1 C_2 / (F_1 C_1 C_2 + 1) \end{array}$$

# (NARS) Compound Terms

- Compound terms: *sets*, *intersections*, *differences* and *images* in extensional and intensional versions, *products*.
- Variants of the inheritance relation: *similarity*, *instance*, and *property*.
- New inference rules are added to carry out compound *composition* and *decomposition*.
- Related changes in memory and control.

# NARS Higher-Order Reasoning

- Two higher-order relations, *implication* and *equivalence*, are defined between statements.
- Compound statements: *negations*, *conjunctions*, and *disjunctions*.
- The implication relation is used to carry out *conditional and hypothetical inferences*.
- Variable terms are used to carry out *general and abstract inferences*. Variable can be independent or dependent on other variables.
- Some rules are the same (e.g. deduction, abduction, induction), some are new.

# NARS Procedural Reasoning

- An *event* has a time-dependent truth-value.
- Events can be simultaneous or one can happen before another.
- New operators and relations are formed, such as sequential conjunction (“,”), parallel conjunction (“;”), predictive implication (“ $\Rightarrow$ ”), retrospective implication (“ $\Leftarrow$ ”), and concurrent implication (“ $\mid\Rightarrow$ ”).

operations = executable  
events

Deduction	Abduction	Induction
$M \Rightarrow P$	$P \Leftarrow M$	$M \Rightarrow P$
$S \Rightarrow M$	$S \Rightarrow M$	$M \Leftarrow S$
<hr/>	<hr/>	<hr/>
$S \Rightarrow P$	$S \Rightarrow P$	$S \Rightarrow P$

the system issues execution commands and  
collects execution consequences by I/O



# NARS Control Strategy

- Task: a question or an assertion to assimilate.
- Beliefs and tasks are links of the belief network, a concept is a node with all its links.
- Concepts, tasks and beliefs have *priority-values*.
- High-priority concept is selected probabilistically, some its task and belief are processed by an inference rule.
- Factors influence the priority of an item: quality of the item, usefulness of the item in history, and relevance of the item to the current context.
- Events have *desirability-values*, the system uses *decision-making* procedure to create new goals from desirable and achievable events.

# (NARS) Defence of Logical Approach

- „In its original and broad sense, “logic” is just the attempt of capturing valid patterns of inference in a content-independent manner, and “inference” is just the process by which new knowledge is derived from existing knowledge.”
- „Non-Axiomatic Logic of NARS is fundamentally different from traditional mathematical logic, in that it is an attempt to capture the principle of adaptation with insufficient knowledge and resources. In this logic, a “term” is an identifiable item or pattern in the system’s experience; a “statement” is a relation between two terms indicating their substitutability; the “truth-value” of a statement measures how a statement is supported or refuted by the system’s experience; the “meaning” of a term indicates the role it plays in the system’s experience; the function of an “inference rule” is to accomplish a single inference step, which build term(s) and/or statement(s) to summarize the information in existing ones; and an “reasoning process” is a sequence of steps to carry out the tasks needed by the system for surviving and adapting.”

# Novamente

- Components of the system have been commercially deployed:
  - **Biomind OnDemand** product for bioinformatic data analysis
  - **ImmPort: NIH Web portal** with Biomind/Novamente based analytics on the back end
  - **INLINK** language processing system developed for INSCOM (Army Intelligence)
- Work in progress:
  - *Electric Sheep Company: Virtual Pets* (early 2008)
  - Virtual Agents with rudimentary English capability (2010)
- “Software and mathematics alone, no matter how advanced, cannot create an AGI. Intelligence most naturally emerges through situated and social experience.”
- AGI-Sim based on CrystalSpace, no much physics yet but robot-like steering.

# PsyNet Philosophy of Mind

- **Association.** Patterns, when given attention, spread some of this attention to other patterns that they have previously been associated with in some way. Every idea in the memory is an active agent, continually acting on those ideas with which the memory associates it.
- **Hierarchical network** (inheritance). Patterns are habitually in relations of control over other patterns that represent more specialized aspects of themselves. **Heterarchical network** (similarity). (Hierarchical + heterarchical = **dual network**.)
- **Differential attention allocation.** Patterns that have been valuable for goal-achievement are given more attention, and are encouraged to participate in giving rise to new patterns.
- **Pattern creation.** Patterns that have been valuable for goal-achievement are mutated and combined with each other to yield new patterns.
- **Credit Assignment.** Habitual patterns in the system that are found valuable for goal-achievement are explicitly reinforced and made more habitual.
- **Self structure.** A portion of the network of patterns forms into an approximate image of the overall network of patterns.

# Novamente's “Atom Space”

- Atoms = Nodes or Links
- Atoms have
  - Truth values (probability + weight of evidence)
  - Attention values (short and long term importance)
- The Atomspace is a weighted, labeled hypergraph
  - ConceptNodes
    - “tokens” for links to attach to
  - PredicateNodes
  - ProcedureNodes
  - PerceptNodes
    - Visual, acoustic percepts, etc.
  - NumberNodes
  - Logical links
    - InheritanceLink
    - SimilarityLink
    - ImplicationLink
    - EquivalenceLink
  - Intensional logical relationships
  - HebbianLinks
  - Procedure evaluation links

# Novamente mechanisms

- A **map** = a fuzzy set of nodes or links that corresponds to abstract concept or schema (event); they obey emergent dynamics similar to that of nodes and are habitually activated together, either all at once or in a particular habitual sequence.
- **Probabilistic reasoning** carried directly on the hypergraph. (Probabilistic Logic Networks, successor of Probabilistic Term Logic.)
- **Evolutionary learning** carried out using MOSES (descendant of BOA for evolving programs/scripts).
- **Attention allocation** by combination of inference and evolutionary pattern mining.

# Novamente Past Applications

- Biomind:
  - BOA uncovers patterns in labeled datasets (microarray gene expression), and learn classification models
  - PTL incorporates background knowledge: gene and protein function, research papers, gene sequence alignment, protein interactions, and pathways
- INLINK:
  - PTL learns disambiguation by interactive knowledge entry.
  - BOA Pattern Mining is used to spontaneously create queries that are judged interesting.

# More on Atoms

- **ConceptNodes**, which derive their meaning via interrelationships with other nodes
- **PerceptNodes** nodes representing perceptual inputs into the system (e.g., pixels, points in time, etc.)
- **TimeNodes** representing moments and intervals of time
- **PredicateNodes** representing complex patterns (procedures that output truth values)
- **SchemaNodes** embodying procedures (procedures that output Atoms)
- **Inheritance** links implement hierarchical network
- **Similarity** links implement heterarchical network
- Procedures can be represented as terms or as variable-free combinators (see Curry and Feys)



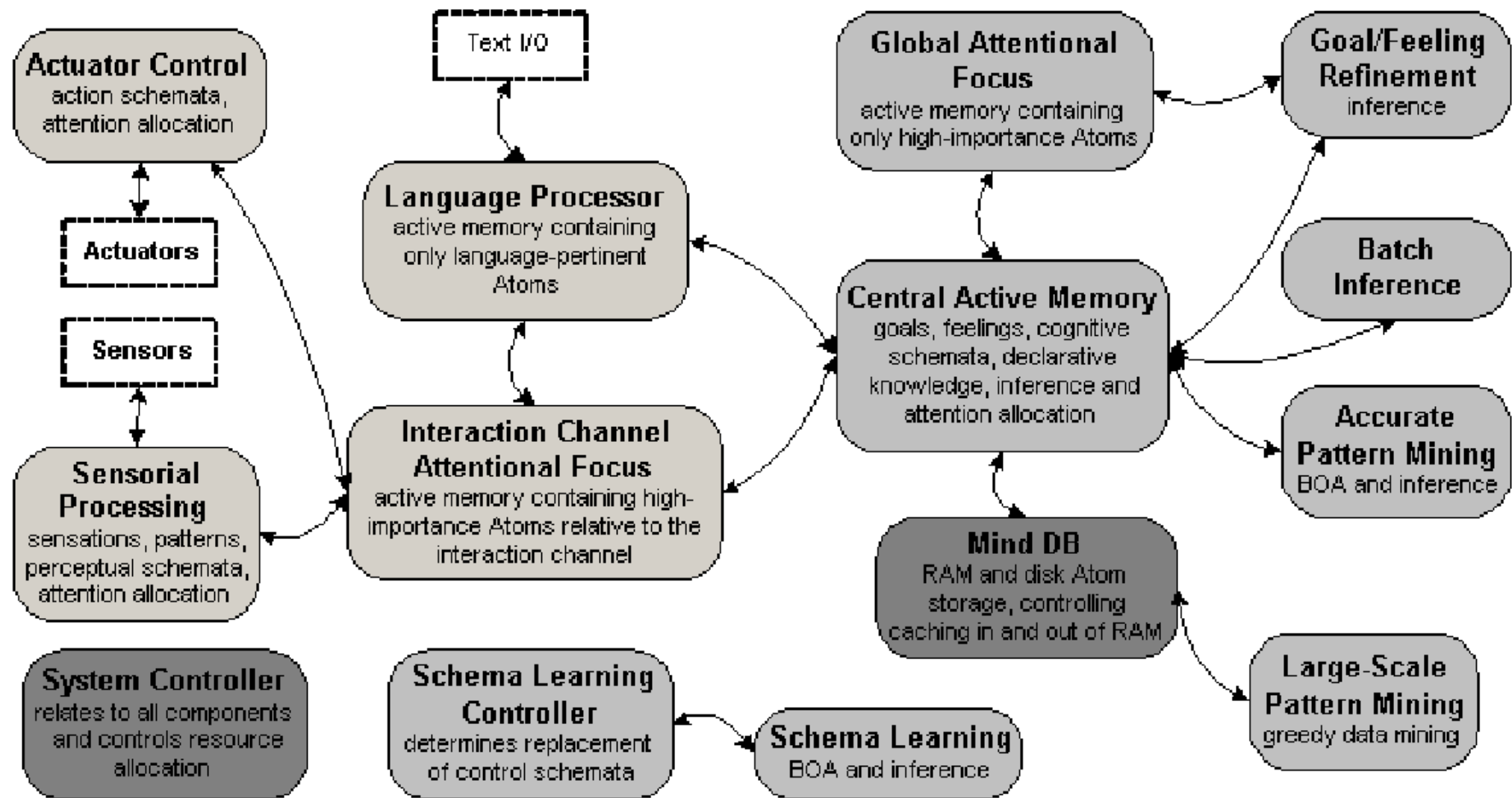
Node Variety	Description
<b>Perceptual Nodes</b>	These correspond to perceived items, like WordInstanceNode, CharacterInstanceNode, NumberInstanceNode, PixelInstanceNode
<b>Procedure Nodes</b>	These contain small programs called “schema,” <sup>4</sup> and are called SchemaNodes. Action Nodes that carry out logical evaluations are called PredicateNodes.
<b>ConceptNodes</b>	This is a “generic Node” used for two purposes. An individual ConceptNode may represent a category of Nodes. Or, a Map of ConceptNodes may represent a concept.
<b>Psyche Nodes</b>	These are GoalNodes and FeelingNodes, which are special PredicateNodes that play a special role in overall system control, in terms of monitoring system health, and orienting overall system behavior.

Link Variety	Description
<b>Logical links</b>	These represent symmetric or asymmetric logical relationships , either among Nodes (InheritanceLink, SimilarityLink), or among links and PredicateNodes (e.g. ImplicationLink, EquivalenceLink)
<b>MemberLink</b>	These denote fuzzy set membership
<b>Associative links</b>	These denote generic relatedness, including HebbianLink learned via Hebbian learning, and a simple AssociativeLink representing relationships derived from natural language or from databases.
<b>ExecutionOutputLink</b>	These indicate input-output relationships among SchemaNodes and PredicateNodes and their arguments
<b>Action-Concept links</b>	Called ExecutionLinks and EvaluationLinks, these form a conceptual record of the actions taken by SchemaNodes or PredicateNodes
<b>ListLink and concatListLink</b>	These represent internally-created or externally-observed lists, respectively

Map Type	Description
<b>Concept map</b>	a map consisting primarily of conceptual Nodes
<b>Percept map</b>	a map consisting primarily of perceptual Nodes, which arises habitually when the system is presented with environmental stimuli of a certain sort
<b>Schema map</b>	a distributed schema
<b>Predicate map</b>	a distributed predicate
<b>Memory map</b>	a map consisting largely of Nodes denoting specific entities (hence related via MemberLinks and their kin to more abstract Nodes) and their relationships
<b>Concept-percept map</b>	a map consisting primarily of perceptual and conceptual Nodes
<b>Concept-schema map</b>	a map consisting primarily of conceptual Nodes and SchemaNodes
<b>Percept-concept-schema map</b>	a map consisting substantially of perceptual, conceptual and SchemaNodes
<b>Event map</b>	a map containing many links denoting temporal relationships
<b>Feeling map</b>	a map containing FeelingNodes as a significant component
<b>Goal map</b>	a map containing GoalNodes as a significant component

# Novamente Architecture

- Each of functionally specialized *Lobes* (or **Units**) contains a hypergraph and a number of *MindAgents*.
- Some **MindAgents** perform basic system maintenance, other apply PLN (PTL) and MOSES (BOA) inferences in conjunction with simple heuristics to carry out particular cognitive tasks like procedure learning, probabilistic inference on declarative knowledge, language parsing.
- The **Mind OS** builds on a distributed processing framework to enable distributed MindAgents to act efficiently on large populations of Nodes and Links



**Figure 1.** High-level architecture of a complex Novamente instantiation. Each component is a Lobe, which contains multiple atom types and mind agents. Lobes may span multiple machines, and are controlled by schemata which may be adapted/replaced by new ones learned by Schema Learning, as decided by the Schema Learning Controller. The diagram shows a configuration with a single interaction channel, that contains sensors, actuators and linguistic input; real deployments may contain multiple channels, with different properties. (From Looks et al, 2004)

<b>MindAgent</b>	<b>Function</b>	<b>Development Status</b>
<b>First-Order Inference</b>	Acts on first-order logical links, producing new logical links from old using the formulas of Probabilistic Term Logic	Complete
<b>LogicalLinkMining</b>	Creates logical links out of nonlogical links	Complete
<b>Evolutionary Predicate Learning</b>	Creates PredicateNodes containing predicates that predict membership in ConceptNodes	Complete
<b>Clustering</b>	Creates ConceptNodes representing clusters of existing ConceptNodes (thus enabling the cluster to be acted on, as a unified whole, by precise inference methods, as opposed to the less-accurate map-level dynamics)	Complete
<b>Activation Spreading</b>	Spreads activation among Atoms in the manner of a neural network	Complete
<b>Importance Updating</b>	Updates Atom “importance” variables and other related quantities	Implemented in prototype form
<b>Concept Formation</b>	Creates speculative, potentially interesting new ConceptNodes	Implemented in prototype form
<b>Evolutionary Optimization</b>	A “service” MindAgent, used for schema and predicate learning, and overall optimization of system parameters	Complete
<b>Hebbian Association Formation</b>	Builds and modifies HebbianLinks between Atoms, based on a PTL-derived Hebbian reinforcement learning rule	Implemented in prototype form
<b>Evolutionary Schema Learning</b>	Creates SchemaNodes that fulfill criteria, e.g. that are expected to satisfy given GoalNodes	Partially implemented
<b>Higher-Order Inference</b>	Carries out inference operations on logical links that point to links and/or PredicateNodes	Partially implemented
<b>Logical Unification</b>	Searches for Atoms that mutually satisfy a pair of PredicateNodes	Not yet implemented
<b>Predicate/Schema Formation</b>	Creates speculative, potentially interesting new SchemaNodes	Not yet implemented
<b>Schema Execution</b>	Enacts active SchemaNodes, allowing the system to carry out coordinated trains of action	Partially implemented
<b>Map Encapsulation</b>	Scans the AtomTable for patterns and creates new Atoms embodying these patterns	Not yet implemented
<b>Map Expansion</b>	Takes schemata and predicates embodied in nodes, and expands them into multiple Nodes and links in the AtomTable (thus transforming complex Atoms into Maps of simple Atoms)	Not yet implemented
<b>Homeostatic Parameter Adaptation</b>	Applies evolutionary programming to adaptively tune the parameters of the system	Implemented in prototype form

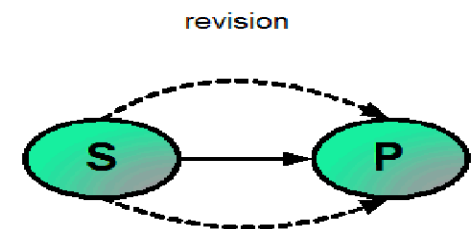
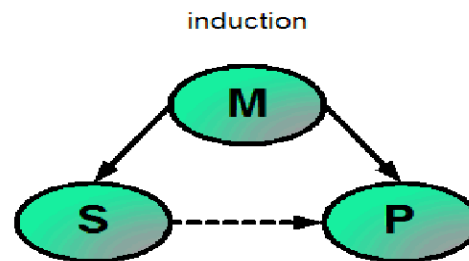
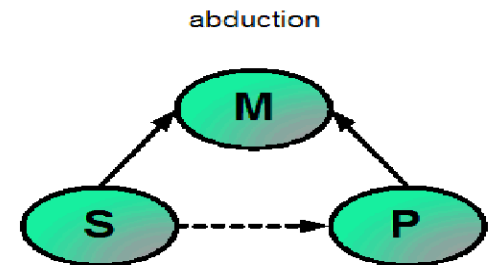
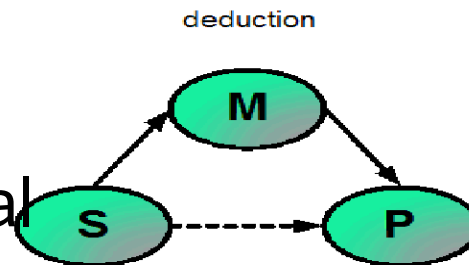
# Probabilistic Logic Networks

- More complex than logic in NARS
- Gracefully deals with inconsistencies e.g. by iteratively correcting premises (example: sensory input to agree with understanding)

- Higher-order PTL deals with links pointing to links

- Distinction between intensional and extensional (in NARS these are symmetric)

- Inheritance  $A \rightarrow B =$   
 Subset  $A \rightarrow B$  or  
 Intensional Inheritance  $A \rightarrow B$



← Old Link   ← - - - New Link

Figure 4. First-Order PTL Inference on InheritanceLinks

# MOSES Meta-Optimizing Semantic Evolutionary Search

- *The properties of programs and program spaces can be leveraged as inductive bias to reduce the burden of manual representation-building, leading to competent program evolution.*
- Programs are normalized for better correlation of syntactic and semantic distance
- Programs are aligned before cross-over
- Programs are generated according to a learned distribution



# The Fundamental Cognitive Dynamic

$$S(t+1) = B( F(S(t) + I(t)) )$$

Forward: create new mental forms by combining existing ones

Backward: seek simple explanations for the forms in the mind, including the newly created ones. The explanation itself then comprises additional new forms in the mind

Forward: ...

Backward: ...

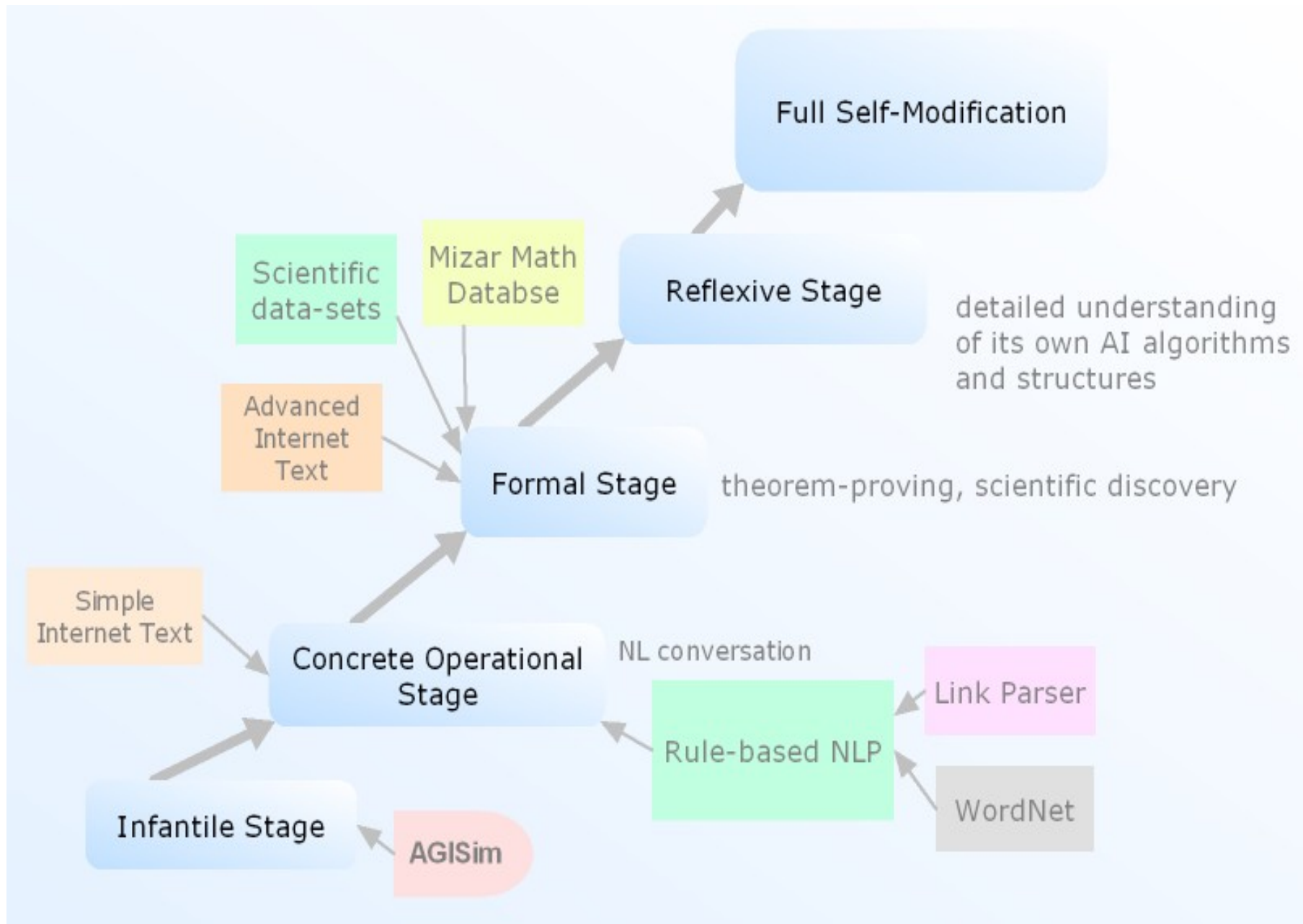
Etc.

... Combine ... Explain ... Combine ... Explain ... Combine ...

# Intelligence and Self

- The reflexive process of *flexibly recognizing patterns in oneself and then improving oneself based on these patterns* is the “basic algorithm of intelligence”
- The phenomenal self, a key aspect of intelligent systems, is the result of an intelligent system *recognizing itself as a pattern* in its (internal and external) behaviors

# Stages of Cognitive Development



# Sources

- *„Essentials of General Intelligence: The direct path to AGI”* Peter Voss, 2002
- *„A Gentle Introduction to the Universal Algorithmic Agent AIXI”* Marcus Hutter, 2003

# Sources

- „*An Introduction to SNePS 3*”, Stuart Shapiro 2000
- „*SNePS 2.6.1 User's Manual*”, Stuart Shapiro, The SNePS Implementation Group 2004
- „*SNePS: A Logic for Natural Language Understanding and Commonsense Reasoning*”, Stuart C. Shapiro 1999
- „*Metacognition in SNePS*”, Stuart C. Shapiro, William J. Rapaport, Michael Kandefer, Frances L. Johnson, and Albert Goldfain 2006

# Sources

- „*An Introduction to the Soar Cognitive Architecture*”, Tony Kalus, Frank Ritter 2003 (slides)
- „*Cognitive Theory, SOAR*”, Richard L. Lewis 1999
- „*The Soar User’s Manual Version 8.6.3*” John E. Laird, Clare Bates Congdon, Karen J. Coulter 2006

# Sources

- „*An Integrated Theory of the Mind*”, John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, Yulin Qin, 2004
- „ACT-R 6.0 Reference Manual”, Dan Bothell, 2006
- „Rational Cognition in OSCAR” John L. Pollock,
-

# Sources

- „*Augmenting BDI Agents with Deliberative Planning Techniques*”, A. Walczak, L. Braubach, A. Pokahr, W. Lamersdorf, 2006
- „*A Goal Deliberation Strategy for BDI Agent Systems*”, Alexander Pokahr, Lars Braubach, Winfried Lamersdorf 2005
- „*Jadex: A BDI Reasoning Engine*”, Alexander Pokahr, Lars Braubach, Winfried Lamersdorf, 2005



# Sources

- „*TouringMachines: Autonomous Agents with Attitudes*”, Innes A. Ferguson 1992
- „*TouringMachines: an architecture for dynamic, rational, mobile agents*”, Innes A. Ferguson 1992

# Sources

- *„From 2001 to 2001: Common Sense and the Mind of HAL”, Douglas B. Lenat*
- *„The Cyc® System: Notes on Architecture” Nick Siegel, Keith Goolsbey, Robert Kahlert, and Gavin Matthews 2004*
- *„Guiding Inference with Policy Search Reinforcement Learning” Cynthia Matuszek, Pace Reagan Smith, Michael Witbrock, Matthew E. Taylor 2007*
- *„Autonomous Classification of Knowledge into an Ontology” M. E. Taylor, C. Matuszek, B. Klimt, M. Witbrock 2007*

# Sources

- „*Common Sense Reasoning – From Cyc to Intelligent Assistant*”, Kathy Panton, Cynthia Matuszek, Douglas Lenat, Dave Schneider, Michael Witbrock, Nick Siegel, and Blake Shepard 2006

# Sources

- „*Human-level Intelligence Laboratory*” directed by Nick Cassimatis ([www.cassimatis.com](http://www.cassimatis.com))
- „*Grammatical Processing Using the Mechanisms of Physical Inference*”, Nicholas Cassimatis 2004

# Sources

- „*The Lazy Learning Package*” Mauro Birattari and Gianluca Bontempi
- „*Growing Neural Gas. Experiments with GNG, GNG with Utility and Supervised GNG*” Jim Holmström
- „*Churchland on Connectionism*” Aarre Laasko
- MIT AI Lab, Humanoid Robotics Group, Cog project
- „*Building Behaviors Developmentally: A New Formalism*”, Brian Scassellati 1998

# Sources

- „*Hierarchical Temporal Memory. Concepts, Theory, and Terminology*” Jeff Hawkins and Dileep George, 2007
- „*SAIL and Dav Developmental Robot Projects: the Developmental Approach to Machine Intelligence*”, Juyang Weng
- „*From Neural Networks to the Brain: Autonomous Mental Development*”, J. Weng and W. S. Hwang 2006
- "Novelty and Reinforcement Learning in the Value System of Developmental Robots", X. Huang and J. Weng, 2002

# Sources

- „*Hierarchical Discriminant Regression*”, W. Hwang and J. Weng, 2000
- „*Incremental Hierarchical Discriminant Regression*”, J. Weng and W. Hwang, 2007
- „*Action Chaining by a Developmental Robot with a Value System*”, Y. Zhang and J. Weng, 2002

# Sources

- „*How Minds Work: A Cognitive Theory of Everything, Full Tutorial*”, Stan Franklin, University of Memphis Cognitive Computing Research Group 2006
- „*From NARS to a Thinking Machine*”, Pei Wang 2006 (also AGI Workshop presentation)
- „*Return to Term Logic*”, Pei Wang 1997



# Sources

- „*A Comparison of the Novamente AI Design with the Human Mind/Brain*”, Ben Goertzel 2005
- „*Novamente. A Practical Architecture for Artificial General Intelligence*” presentation, Ben Goertzel 2006