A Framework for Integrating Symbolic and Sub-symbolic Representations

Keith Clark
Imperial College

Bernhard Hengst
University of NSW

Maurice Pagnucco
University of NSW

David Rajaratnam
University of NSW

Peter Robinson
University of Queensland

Claude Sammut
University of NSW

Michael Thielscher
University of NSW

Abstract

This paper establishes a framework that hierarchically integrates symbolic and sub-symbolic representations in an architecture for cognitive robotics. It is formalised abstractly as nodes in a hierarchy, with each node a sub-task that maintains its own belief-state and generates behaviour. An instantiation is developed for a real robot building towers of blocks, subject to human interference; this hierarchy uses a node with a concurrent multi-tasking teleo-reactive program, a node embedding a physics simulator to provide spatial knowledge, and nodes for sensor processing and robot control.

1 Introduction

A physical symbol system as the sole basis for artificial intelligence has been criticised by many researchers. Allen Newell and Herbert Simon introduced the *physical symbol system hypothesis (PSSH)* [Newell and Simon, 1976] implying that human thinking is a kind of symbol manipulation process, and that we can build machines to mimic human intelligence. Detractors include Rodney Brooks who showed that robots with superior behaviour do not necessarily use higher level symbols at all [Brooks, 1990]. Recently the paradigm has shifted more to probabilistic robotics [Thrun *et al.*, 2005]. Nilsson [2006] analyses some of the attacks against the PSSH and grants the need to supplement symbol systems with non-symbolic processes in intelligent systems, mostly for perceptual and motor activities close to the environment.

Our architecture for cognitive robotics accommodates both symbolic and sub-symbolic representation. The architecture comprises nodes operating at different spatial and temporal scales, linked in a hierarchy. Each node is a kind of sub-task that maintains a belief state about an abstract representation of part of the robot's environment. It generates behaviour based on the belief state. We are not proposing an expressive language to include probabilities symbolically, but rather that nodes can use different representations to interconnect symbolic or probabilistic models and behaviour.

The two main contributions of this paper are:

1. *The formalisation of a general architecture for cognitive robotics and proof that cyclic updates of the hierarchy of nodes are well defined.*
2. *The instantiation of the architecture with a Baxter robot*[1] *tasked to build multiple towers. The main features are:*
   - *A symbolic node with a concurrent multi-tasking extension of Nilsson's Teleo-Reactive (TR) rule based robot agent programming language.*
   - *A spatial node using a rigid-body simulator acting as the "mind's eye" of the robot. The physics simulator introduces common sense real-world spatial knowledge that would otherwise be cumbersome to represent by a formal symbol system.*
   - *Controller nodes that process robot sensory input and generate robot motor actions.*

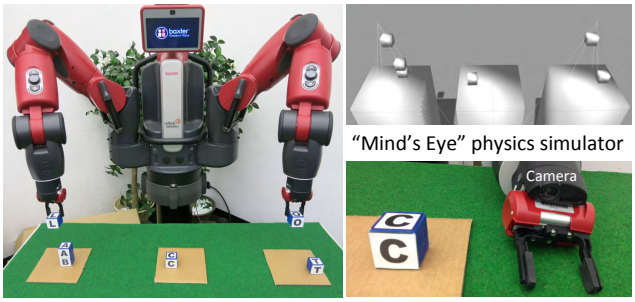


Figure 1: Baxter in blocks-world. The belief state is reflected in the "mind's eye" (physics simulator). A closeup of a block and the arm end-effector showing the co-location of the gripper and camera.

In the rest of this paper we position our robot architecture in related work and formalise the architectural framework using a motivating example. We instantiate the architecture with a real-world concrete example, namely a two armed robot building towers in a blocks-world environment. Finally, we discuss robustness, limitations, and future work.

2 Related Work

Several cognitive architectures have been proposed. Prominent ones include SOAR [Laird *et al.*, 1987] and ACT-R [An-

[1] The Baxter robot is built by Rethink Robotics, a company founded by Rodney Brooks.

derson, 1993]. These architectures are based on the physical symbol system hypothesis, but have been extended to interface with an external environment. ICARUS [Langley and Choi, 2006] has many similarities to SOAR and ACT-R, and is directly grounded in perception and action, with conceptual inference and skill more basic than problem solving. A well known sub-symbolic reactive architecture is Subsumption [Brooks, 1986] in which higher-level layers can subsume the roles of lower levels by suppressing their outputs.

Other related work includes: GWT [Baars, 1988], a cognitive architecture to account for a multitude of communicating brain processes; RL-TOPS [Ryan and Pendrith, 1998], a hybrid system for combining teleo-reactive planning and reinforcement learning (RL); MAXQ task-hierarchies [Dietterich, 2000]; CRAM [Beetz *et al.*, 2010], a cognitive robot abstract machine; and MALA [Haber and Sammut, 2012], a multi-agent blackboard based cognitive architecture inspired by Minsky's *Society of Mind* [Minsky, 1986].

Our architectural commitments are closest to the Robot Control System (RCS) reference model architecture [Albus and Meystel, 2001]. The RCS architecture was also the inspiration for the triple-tower architecture [Nilsson, 2001] used to illustrate the operation of a teleo-reactive program for block-stacking tasks. The triple-tower instantiates the RCS architecture for modelling at the symbolic level. However, it provides no explicit representation of sub-symbolic processes, instead assuming that there simply exists some underlying sensory system from which symbolic facts are generated.

In contrast to all of these existing architectures, which are either described informally or without connection to real-world robotic implementations, we formally axiomatise our model and instantiate it in a real robotic setting.

3 The Architectural Framework

The following formalisation presents a hierarchical system consisting of interconnected abstract nodes, each with their own internal belief state and reasoning mechanism. We use the following simple running example as an explanatory aid.

Example. *A self-driving car is tasked to park between* $0.2m$ *and* $0.3m$ *rear-to-wall from an arbitrary initial position. The car has camera and laser distance sensors providing Gaussian measurements to the wall. At each time-step it is able to move forward or backward by a small displacement, or stop.*

3.1 Nodes

Nodes are tasked to achieve a goal, and have two primary functions: world-modelling and behaviour-generation.

World-Modelling. The notion of a world-model appears in many disciplines under different names (e.g., internal state, hidden state, belief state). We adopt the term *belief state* as the representation of a node's internal world-model. The terminology for updating a world-model similarly varies (e.g, filtering, localisation, tracking, belief revision) and we adopt the generic term: *update*. Update may depend on sensing observations, or may be an *expectation* update based on a node's actions. Finally, we distinguish between *sensing* and *observation*. Sensing is the process of extracting observations, the latter of which is used to update a belief state.

Behaviour-Generation. The belief state helps the node select the next action. A function that maps states to actions is a *policy*. Policies may be provided directly or determined by planning or decision-theoretic methods. Typical examples are GOLOG programs [Levesque *et al.*, 1997], Teleo-Reactive Programs [Nilsson, 1994], and RL policies.

Definition 1. *A cognitive language is a tuple* $\mathcal{L} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O})$*, where* $\mathcal{S}$ *is a set of belief states,* $\mathcal{A}$ *is a set of actions,* $\mathcal{T}$ *is a set of task parameters, and* $\mathcal{O}$ *is a set of observations. A cognitive node is a tuple* $N = (\mathcal{L}, \Pi, \lambda, \tau, \gamma, s^0, \pi^0)$ *s.t:*

- $\mathcal{L}$ *is the cognitive language for* N*, with initial belief state* $s^0 \in \mathcal{S}$*.*
- Π *a set of policies such that for all* $\pi \in \Pi$*,* $\pi : \mathcal{S} \to 2^{\mathcal{A}}$*, and an initial policy* $\pi^0 \in \Pi$*.*
- *A policy selection function* $\lambda \colon 2^{\mathcal{T}} \to \Pi$*, s.t.* $\lambda(\{\}) = \pi^0$*.*
- *A observation update operator* $\tau : 2^{\mathcal{O}} \times \mathcal{S} \to \mathcal{S}$*.*
- *An action update operator* $\gamma : 2^{\mathcal{A}} \times \mathcal{S} \to \mathcal{S}$*.*

Definition 1 provides a very abstract characterisation of a node, allowing for an arbitrary representation and reasoning mechanism for individual nodes. For example, it can encapsulate both stochastic and symbolic nodes.

Example. *Let* $E = \{\langle x, \sigma_x \rangle \mid x, \sigma_x \in \mathbb{R},\ 0 \leq x < +\infty$*, and* $-\infty < \sigma_x < +\infty\}$*, consist of position estimates represented as a Gaussian distribution with mean* x *and standard deviation* σ_x*. Stochastic node* N_1 *defines the cognitive language* $(\mathcal{S}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{O}_1)$ *as follows. The set of belief states consist of (car) position estimates:* $\mathcal{S}_1 = E$*, with the initial belief state* $s_1^0 = \langle 0.8, 0.1 \rangle$ *being a known starting position and standard deviation. The set of actions represents small independent moves:* $\mathcal{A}_1 = \{\delta_x \in \{-0.01, 0.0, 0.01\}\}$*. The task parameters represent the requirement to move forwards or backwards:* $\mathcal{T}_1 = \{F_1, B_1\}$*. Finally, the observations represent camera-laser pairs of observations:* $\mathcal{O}_1 = \{\langle c, l \rangle \mid c, l \in E\}$*.*

There are three policies: to move forwards, to move backwards, and to stop: $\Pi_1 = \{\pi_F, \pi_B, \pi_S\}$*, where* $\pi_F(s^x) = \{0.01\}$*,* $\pi_B(s^x) = \{-0.01\}$*, and* $\pi_S(s^x) = \{0.0\}$*, for every* $s^x \in \mathcal{S}_1$*. Policy* π_S *is also the initial policy.*

The policy selection function λ_1 *respectively maps sets* $\{F_1\}$*,* $\{B_1\}$*,* $\{\} \in 2^{\mathcal{T}_1}$ *to policies* π_F*,* π_B*,* π_S*, with the action update operator being:*

$$\gamma_1(\{\delta_x\}, \langle x, \sigma_x \rangle) = \langle x + \delta_x, \sigma_x + v \rangle$$

where v *represents the increase in standard deviation due to process noise during the last time-step.*

The observation update operator is:

$$\tau_1(\{\langle \langle x_c, \sigma_{x_c} \rangle, \langle x_l, \sigma_{x_l} \rangle \rangle\}, \langle x, \sigma_x \rangle) = \langle x', \sigma_{x'} \rangle$$

where $\langle x_c, \sigma_{x_c} \rangle$*,* $\langle x_l, \sigma_{x_l} \rangle$ *are independent camera and laser distance observations, and* $\langle x', \sigma_{x'} \rangle$ *is the corrected state after the observation update:*

$$x' = \frac{x_c \sigma_x^2 \sigma_{x_l}^2 + x_l \sigma_x^2 \sigma_{x_c}^2 + x \sigma_{x_c}^2 \sigma_{x_l}^2}{\sigma_x^2 \sigma_{x_l}^2 + \sigma_x^2 \sigma_{x_c}^2 + \sigma_{x_c}^2 \sigma_{x_l}^2}$$

$$\sigma_{x'} = \frac{\sigma_{x_c}^2 \sigma_{x_l}^2 \sigma_x^2}{\sigma_{x_c}^2 + \sigma_{x_l}^2 + \sigma_x^2}$$

The action update and observation update operators define a Kalman filter estimating the belief state of the car.

The example shows how a stochastic node can be modelled within our framework, where it is used to provide immediate sensing and control of a car. We now show how to represent the state and behaviour of the car at a symbolic level.

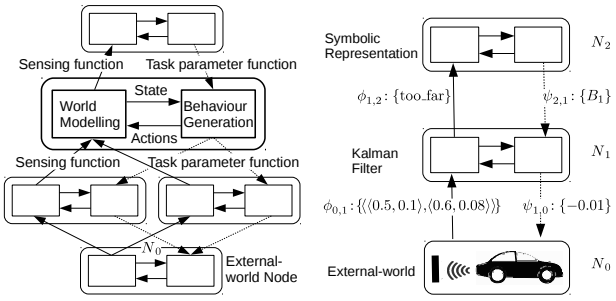
Example. *The symbolic node N_2 contains the cognitive language $(\mathcal{S}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{O}_2)$ as follows. The set of belief states $\mathcal{S}_2 = \{too\_far, too\_close, on\_target\}$, where on_target is the initial belief state. The set of actions $\mathcal{A}_2 = \{F_2, B_2\}$ yields action sets $\{F_2\}$, $\{B_2\}$, $\{\}$, respectively to move the car forwards, backwards, and stop. The task parameter set is empty $\mathcal{T}_2 = \{\}$ as N_2 is a top-level node. Hence, the set of policies Π_2 consists only of the initial policy π_2^0 : on_target $\mapsto \{\}$, too_far $\mapsto \{B_2\}$, too_close $\mapsto \{F_2\}$. Finally, the set of observations is identical to the set of beliefs: $\mathcal{O}_2 = \mathcal{S}_2$.*

The observation update operator, τ_2, replaces the current state with an observation (i.e., for all x and s, $\tau_2(\{x\}, s) = x$), while the action update operator, γ_2, makes no change to the state (i.e., for all a and s, $\gamma_2(a, s) = s$).

Note, due to space restrictions a number of the functions in the example are only partially defined. However, as part of the interconnection of nodes we shall ensure that the inputs to these functions will be constrained to only the defined cases.

3.2 Cognitive Hierarchy

Nodes in the architecture are interlinked in a hierarchy, similar to that of a *task-graph* [Dietterich, 2000]. Nodes in the hierarchy represents sub-tasks, equipped with their own state representations and reasoning mechanisms. The lowest level node is a proxy for the external world, consisting of physical sensors and actuators. Sensing information is passed up the hierarchy to the top-level nodes, while actions are passed down, eventually resulting in physical actions (Figure 2(a)).



(a) Typical node hierarchy (b) Autonomous parking

Figure 2: Observations are propagated up the hierarchy, revising beliefs. Actions are generated from belief states and translate into task parameters for lower-level nodes. The node with two children (Diagram (a)) could represent concurrently setting behaviours for the two arms of a humanoid robot.

Definition 2. *A cognitive hierarchy is a tuple $H = (\mathcal{N}, N_0, F)$ s.t:*

- *$\mathcal{N}$ is a set of cognitive nodes; $N_0 \in \mathcal{N}$ is a distinguished node corresponding to the external environment.*
- *F is a set of function pairs $\langle \phi_{i,j}, \psi_{j,i} \rangle \in F$ that connect nodes $N_i, N_j \in \mathcal{N}$ where:*
  - *$\phi_{i,j} : \mathcal{S}_i \to 2^{\mathcal{O}_j}$ is a sensing function, and*
  - *$\psi_{j,i} : 2^{\mathcal{A}_j} \to 2^{\mathcal{T}_i}$ is a task parameter function.*
- *Sensing graph: each $\phi_{i,j}$ represents an edge from node N_i to N_j and forms a directed acyclic graph with N_0 as the unique source node of the graph.*
- *Action graph: the set of task parameter functions forms a converse to the sensing graph such that N_0 is the unique sink node of the graph.*

Definition 2 models the connection between nodes as consisting of pairs of sensing and task parameter functions. The sensing function extracts observations from a lower-level node, while the task parameter function translates a higher-level node's action set to a task parameter set, which is then used to select the active policy for a node. It is worth noting that the pairing of these functions is not a restrictive requirement and it is still possible to model a connection between two nodes with actions but no sensing and vice-versa. This would simply correspond to a sensing, or task parameter, function that always returns the empty set.

Finally, the external world is modelled as a distinguished node, N_0. Sensing functions allow other nodes to observe properties of the external world, and task parameter functions allow actuator values to be modified, but N_0 doesn't "sense" properties of other nodes, nor does it generate task parameters for those nodes. Beyond this external interface the internal behaviour of N_0 is considered to be opaque.

Continuing the running example we can now see how the nodes are connected (displayed graphically in Figure 2(b)).

Example. *With reference to nodes N_1 and N_2 defined earlier, and Figure 2(b), we specify a cognitive hierarchy $H = (\mathcal{N}, N_0, F)$ where $\mathcal{N} = \{N_0, N_1, N_2\}$ and $F = \{\langle \phi_{0,1}, \psi_{1,0} \rangle, \langle \phi_{1,2}, \psi_{2,1} \rangle\}$. For the pair $\langle \phi_{1,2}, \psi_{2,1} \rangle \in F$ we define:*

$$\phi_{1,2} : \{(x < 0.2, \cdot)\} \mapsto \{too\_close\}, \{(0.2 < x \leq 0.3, \cdot)\} \mapsto \{on\_target\}, \{(x > 0.3, \cdot)\} \mapsto \{too\_far\}.$$

$$\psi_{2,1} : \{F_2\} \mapsto \{F_1\}, \{B_2\} \mapsto \{B_1\}, \{\} \mapsto \{\}.$$

The internal representation of N_0 is opaque. Instead we observe that the sensing function from N_0 to N_1 takes a state of N_0 and returns camera and laser state observations. Similarly, the matching task parameter function simply sets a required distance for the vehicle control sub-system:

$$\phi_{0,1}(s_0 \in \mathcal{S}_0) = \{\langle \langle x_c, \sigma_{x_c} \rangle, \langle x_l, \sigma_{x_l} \rangle \rangle\}$$

$$\psi_{1,0} : \{0.01\} \mapsto \{0.01\}, \{-0.01\} \mapsto \{-0.01\}, \{0\} \mapsto \{0\}$$

3.3 Active Cognitive Hierarchy

A cognitive hierarchy captures only static properties and additional details are required to model dynamic components, such as the active belief state, policy, and actions.

Definition 3. *An active cognitive node is a tuple $Q = (N, s, \pi, a)$ where: 1) N is a cognitive node with $\mathcal{S}$, Π, and $\mathcal{A}$ being the corresponding belief states, policies, and actions*

respectively, 2) $s \in \mathcal{S}$ is the current belief state, $\pi \in \Pi$ is the current policy, and $a \in 2^{\mathcal{A}}$ is the current set of actions.

Defining an active cognitive node naturally leads to an *active cognitive hierarchy* as a collection of active nodes.

Definition 4. *An active cognitive hierarchy is a tuple $\mathcal{X} = (H, \mathcal{Q})$ where H is a cognitive hierarchy with set of cognitive nodes $\mathcal{N}$ such that for each $N \in \mathcal{N}$ there is a corresponding active cognitive node $Q = (N, s, \pi, a) \in \mathcal{Q}$ and vice-versa.*

An active cognitive hierarchy captures the complete state of an operational system at some instant. The initial state of such a system is an *initial active cognitive hierarchy* such that the active cognitive nodes in the hierarchy are generated from their corresponding cognitive nodes with each node's initial belief state, initial policy, and the empty set of actions.

3.4 Cognitive Process Model

So far we have formalised an abstraction of an active cognitive hierarchy, but it remains to show how this hierarchy can change over time. For this we introduce the notion of a *process model*. Nodes in an active cognitive hierarchy can be viewed as independent processes operating at synchronised discrete time steps. At each time step the belief state for each node is updated and actions taken. The intention of a process model is that it allows a cognitive robot to operate in a physical environment. Therefore we now construct a model that allows for sensing and actions up and down the cognitive hierarchy. The model operates in two phases; a sensing update phase followed by an action selection phase.

In order to characterise the sensing update for an active cognitive hierarchy we first characterise the process update of the beliefs of a single active cognitive node.

Definition 5. *Let $\mathcal{X} = (H, \mathcal{Q})$ be an active cognitive hierarchy with $H = (\mathcal{N}, N_0, F)$. The sensing process update of $\mathcal{X}$ with respect to an active cognitive node $Q_i = (N_i, s_i, \pi_i, a_i) \in \mathcal{Q}$, written as* **SensingUpdate**$'(M, Q_i)$ *is:*

- *$\mathcal{X}$, if there does not exist a node N_x such that $\langle \phi_{i,x}, \psi_{x,i} \rangle \in F$,*
- *an active cognitive hierarchy $\mathcal{X}' = (H, \mathcal{Q}')$ where $\mathcal{Q}' = \mathcal{Q} \backslash \{Q_i\} \cup \{Q_i'\}$ and $Q_i' = (N_i, \tau_i(O, s_i), \pi_i, a_i)$ s.t:*
$$O = \bigcup \{\phi_{x,i}(s_x) \mid \langle \phi_{x,i}, \psi_{i,x} \rangle \in F \text{ where } Q_x = (N_x, s_x, \pi_x, a_x) \in \mathcal{Q}\}$$

While somewhat complicated in structure, Definition 5 is intuitively simple. An active cognitive hierarchy is updated with respect to a particular node by gathering the sensing observations for all connected nodes that are lower in the hierarchy and updating the node's belief state through its observation update function. By updating each node the belief state of the system as a whole is updated.

Definition 6. *Let $\mathcal{X} = (H, \mathcal{Q})$ be an active cognitive hierarchy with $H = (\mathcal{N}, N_0, F)$ and Φ be the sensing graph induced by the sensing functions in F. The sensing process update of $\mathcal{X}$, written* **SensingUpdate**$(\mathcal{X})$*, is an active cognitive hierarchy:*

$$\mathcal{X}' = \textbf{SensingUpdate}'(\ldots \textbf{SensingUpdate}'(\mathcal{X}, Q_0), \ldots Q_n)$$

where the sequence $[Q_0, \ldots, Q_n]$ consists of all active cognitive nodes of the set $\mathcal{Q}$ such that the sequence satisfies the partial ordering induced by the sensing graph Φ.

Sensing update (Definition 6) successively updates the hierarchy ensuring that a node is only updated once the nodes on which its sensing depends are first updated. The order is intuitive but also guarantees a well-defined update process.

Lemma 1. *For any cognitive model $\mathcal{X}$ the sensing process update of $\mathcal{X}$ is well-defined.*

Proof Sketch: Can be proved by induction on the update sequence. Every valid sensing update sequence satisfies the sensing graph's partial ordering, so a node's belief state is updated only after lower level nodes are updated. Hence, each node's update is fully determined. □

Defining action update follows the same pattern as sensing update by first characterising update for a single node.

Definition 7. *Let $\mathcal{X} = (H, \mathcal{Q})$ be an active cognitive hierarchy with $H = (\mathcal{N}, N_0, F)$. The action process update of $\mathcal{X}$ with respect to an active cognitive node $Q_i = (N_i, s_i, \pi_i, a_i) \in \mathcal{Q}$, written as* **ActionUpdate**$'(M, Q_i)$ *is an active cognitive hierarchy $\mathcal{X}' = (H, \mathcal{Q}')$ where $\mathcal{Q}' = \mathcal{Q} \backslash \{Q_i\} \cup \{Q_i'\}$ and $Q_i' = (N_i, \gamma_i(a_i', s_i), \pi_i', a_i')$ s.t:*

- *if there does not exist node N_x such that $\langle \phi_{x,i}, \psi_{i,x} \rangle \in F$ then: $\pi_i' = \pi_i$ and $a_i' = \pi_i(s_i)$,*
- *else:*
$$\pi_i' = \lambda_i(T) \text{ and } a_i' = \pi_i'(s_i),$$
$$T = \bigcup \{\psi_{x,i}(a_x) \mid \langle \phi_{i,x}, \psi_{x,i} \rangle \in F \text{ where } Q_x = (N_x, s_x, \pi_x, a_x) \in \mathcal{Q}\}$$

Definition 7 is slightly more involved than Definition 5. Firstly, the actions of the nodes higher in the hierarchy are used to generate the task parameters of the current node. From this set a new active policy is selected, which in turn is used to generate the current actions. Finally, these actions are used to update the node's belief state. From here, the definition of action update proceeds in the expected manner.

Definition 8. *Let $\mathcal{X} = (H, \mathcal{Q})$ be an active cognitive hierarchy with $H = (\mathcal{N}, N_0, F)$ and Ψ be the action graph induced by the task parameter functions in F. The action process update of $\mathcal{X}$, written* **ActionUpdate**$(\mathcal{X})$*, is an active cognitive model:*

$$\mathcal{X}' = \textbf{ActionUpdate}'(\ldots \textbf{ActionUpdate}'(\mathcal{X}, Q_n), \ldots Q_0)$$

where the sequence $[Q_n, \ldots, Q_0]$ consists of all active cognitive nodes of the set $\mathcal{Q}$ such that the sequence satisfies the partial ordering induced by the action graph Ψ.

It is worth noting that action update (Definition 8) is almost identical to sensing update except that the update takes place in the opposite order. Naturally, it is also well-defined.

Lemma 2. *For any active cognitive hierarchy $\mathcal{X}$ the action process update of $\mathcal{X}$ is well-defined.*

Proof. Identical pattern to proof for Lemma 1. □

Finally, we can combine the sensing and action process updates for a model into a general process update for the model.

Definition 9. *Let $\mathcal{X} = (H, \mathcal{Q})$ be an active cognitive hierarchy with $H = (\mathcal{N}, N_0, F)$. The process update of $\mathcal{X}$, written* **Update**$(\mathcal{X})$*, is an active cognitive hierarchy:*

$$\mathcal{X}' = \textbf{ActionUpdate}(\textbf{SensingUpdate}(\mathcal{X}))$$

Theorem 3. *The cognitive process model* **Update** *is well-defined.*

Proof. Follows directly from Lemmas 1 and 2. □

The cognitive process model performs what one would intuitively require of an operational cognitive system; updating nodes up the hierarchy and propagating actions back down the hierarchy, resulting in changes to the physical actuators of a robot. Furthermore the fact that it is well-defined guarantees that there is a single unique update to the hierarchy.

4 Baxter and Blocks-World

We instantiate our architectural framework with an off-the-shelf Baxter robot tasked to build towers of uniquely labelled blocks. Baxter's two arms can be operated concurrently and have an overlapping work-space. Each arm end-effector is equipped with a two-pronged gripper and a camera that point down at the table as shown in Figure 1. There are three tables. Each arm is able to reach its own table (designated left or right) and a shared table. Baxter's sensor information includes the arm position and arm effort measurements, 2D camera images, and gripper states. Effector actions move the arms to various positions and open or close the gripper.

The framework is instantiated in ROS using five nodes: a *Controller* node for each arm to sense the world and control arm movement, a *Spatial* node to model the blocks-world scene and quantitatively specify arm pickup and putdown goals, and a symbolic *TR* node to build multiple block towers. Baxter is taken to be the external world node.

Arm Controller Nodes

The world model belief state for each arm includes the position, visual persistence and identity of blocks currently in view, the position of the arm, arm effort, and whether the arm is gripping a block. Specifically the language for each arm node, $\mathcal{L}_{Controller} = \{(blockId, (x, y, z, p)), arm(x, y, z), gripper, effort\}$, where: $blockId \in \{A, B, C, L, T, O\}$; (x, y, z, p) represents the position in metres of the centroid of the block in metres in Baxter's torso coordinate frame; and $p \in \{0, 1, 2, 3, 4\}$ is the visual persistence that is incremented when a block is identified in an image frame and decremented when not; $arm(x, y, z)$ is the position of the arm end-effector in metres, also in the torso frame; $gripper \in \{\text{True}, \text{False}\}$ indicating whether the grippers are holding a block; and $effort$ measures the push-force of the arm.

The camera is used to locate and identify each block in the visual field. We employ OpenCV to find squares representing blocks and to identify block symbols. The size and location of a square in the image yields its 3D position relative to the camera. Given fixed camera offsets from each arm position we can determine the block position in Baxter's reference frame. Arm position is determined by an action update based on the last commanded pose of the arm. The observation update from the external environment updates the visual persistence to ensures that any block in view is not sensitive to momentary occlusions, mis-identification, passing shadows, etc. Only blocks with a visual persistence of 4 are passed to the *Spatial* node in a ROS message.

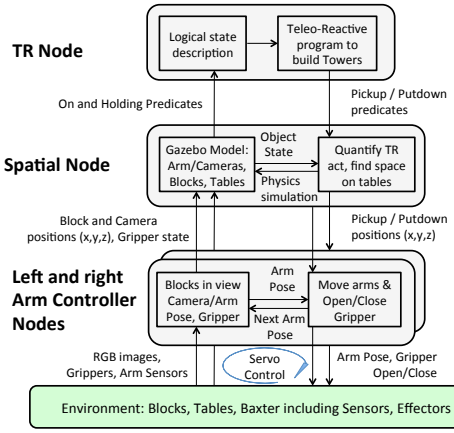


Figure 3: Cognitive graph for Baxter in blocks-world.

The node behaviour-generation consists of picking up and putting down blocks at specific positions determined by the *Spatial* node. We rely on Baxter's inverse kinematic function to move arms to commanded poses. Baxter's sensors and effectors are not accurate enough to pick up or put down blocks by dead-reckoning. Instead, arm controller nodes use a visual feedback control loop through the environment, adjusting the arm pose effector to target a block in the camera image. In this way grippers visually servo into the correct position for grabbing a block. When putting blocks down, arm $effort$ is used in a feedback control loop to determine when the block has made contact with the table or another block.

Spatial Node

The spatial node employs the Gazebo physics simulator to represent the blocks and tables as 3D objects in Baxter's torso coordinate frame. The simulator is used to predict the next belief state based on its implicit knowledge of physics. For example, gravity will ensure that blocks are stacked correctly, even if the gripper releases the block slightly before it makes contact with the block or table below.

The belief state of the spatial node is essentially the scene-graph and includes the identity and quantitative position of each block, the quantitative position of the arm cameras, and the state of the grippers. Each camera position belief state is updated from the belief state of arm positions communicated from the arm controller nodes. The sensing update of the identity and position of blocks is predicated on whether the blocks are expected to be seen by each camera. For example, if a block is placed on a table (or removed) in view of the camera, the block will be created (or deleted) in the simulator, subject to a high visual persistence. Blocks not in the camera field-of-view will not be removed from the simulator, but assumed to have object permanence. A block that is in the

field-of-view of a camera, but occluded by another block in the tower will not be removed. In this way the spatial simulator sensing update operator maintains the belief state of the blocks scene by integrating and reasoning about the sensory information from each of the two arms.

Policies move each arm to pickup or putdown blocks at positions specified quantitatively as (x, y, z) locations in Baxter's coordinate frame. After each such action the policy moves the arm to its respective home position over its table to help refresh the model. The task parameter function maps higher level symbolic TR move commands to quantitative values that select appropriate quantitative move policies. For example, *putdown(right, shared_table)* selects the policy actioning the right arm to place the held block to a clear position on the share table in the simulator, while *pickup(left, A)* selects a policy that results in an action for the left arm to pickup block A from its position in the simulator.

The 3D quantitative representation of the physics simulator can perceptually anchor, relate and remember sensed objects, and provide a basis for abstracting symbolic representations.

Symbolic TR Node

The TR node receives sensed observations of the spatial node's belief state in terms of an abstracted set of logical facts such as *on(A, left_table)* and *holding(left, B)*. These are then used as part of the TR node's symbolic belief state.

A multi-tasking teleo-reactive programming language [Clark and Robinson, 2015] is used as the node's only policy to concurrently build user-specified towers of blocks. The language comprise sequences of Guard $\rightarrow$ Action rules grouped into parameterised procedures. The Guards are deductive queries to a rapidly changing belief state that is updated through sensing. For multi-tasking, the granularity of interleaving robotic resources is specified by declaring certain procedures as task atomic. A queuing convention for resource acquisition prevents starvation, while tasks co-ordinate their use of resources using the agent's belief state.

The multi-tasking features in our example use a single task atomic procedure – to build a tower of blocks. It can be used by each task of an agent building any number of block towers using two robot arms in parallel. Both arms may need to be used by each task at different times since the blocks are distributed over three tables and each arm can only reach two tables. Clashing of arms over the shared table is primarily avoided by making the tables extra resources.

The node sends symbolic commands such as $pickup(left, A)$ and $putdown(right, shared\_table)$ to the *Spatial* node, which are interpreted quantitatively in the *Spatial* node's language.

Results

We conducted several experiments tasking Baxter to build towers of blocks starting from different initial configurations. Our aim was to test the robustness and limits of the instantiated architecture at each level of the abstraction hierarchy.

Persistence of vision keeps the world from going pitch black every time we blink our eyes. In the *Controller* nodes, the persistence parameter in the belief state plays a similar role. Blocks do not disappear in the belief state when a hand is moved rapidly between the camera and a block, or when a human manually places one block on top of another, momentarily blurring the robot's view. Letting the camera dwell on a position for a longer duration than (an approximate) blink of an eye, results in the creation or destruction of blocks in the belief-state of the *Controller* node.

Object permanence is the understanding that objects continue to exist even when they cannot be observed. The Gazebo simulator in the *Spatial* node shows object permanence even when objects are outside the visual field of either camera, or occluded by other objects. Figure 1 shows block "B" on the right table in the simulator, even though it is occluded by block "A". Block "C" on the shared table is not seen by either camera, yet retained in the simulator.

We found that blocks may not be recognised in poor lighting conditions and that grasping may fail if blocks are not oriented orthogonally to the reference frame as assumed. The physics model will incorrectly model block behaviour if supporting blocks have not been previously seen, or are mistakenly seen elsewhere. The TR program fails if unknown block symbols are introduced.

The *TR* node is demonstrated to handle unforeseen contingencies. A video[2] of Baxter building the tower "LAB" on the right table and "COT" on the left table, shows the TR program coping with both help and hindrance from a human.

5 Conclusion

In this paper we developed a formal framework for the integration of symbolic and sub-symbolic components in a cognitive hierarchy. Formalised at an abstract level the framework provides both a model for the construction of cognitive robotic systems, as well as a tool for the analysis of existing architectures (e.g., SOAR), for example, in how they integrate different reasoning components. Consequently, the formalism complements, rather than competes with, these existing architectures and systems. Finally, due to its well-defined properties, the framework is an important step towards the development of provable safety guarantees in robotic systems. This is especially important for implementing robots that operate in unstructured real-world environments.

A real-world instantiation of the framework was outlined, consisting of a multi-node Baxter robot capable of building towers of blocks. This implementation was evaluated empirically to determine the robustness and limitations of the implementation at each level of the hierarchy.

Our framework lays the foundation for spatially modelling other physical agents, including humans, to investigate multi-agent systems in adversarial and collaborative settings. Our interests include general game playing, machine learning and autonomous adaptation at both the symbolic and sub-symbolic levels, and trust studies between humans and machines in mixed-task initiatives. The inclusion of a physics simulator provides opportunities for the agent to predict outcomes and explore courses of action without committing to them in the real world. Human modelling also opens the way to study communication via gesturing, and theory-of-mind.

[2] http://robocup.web.cse.unsw.edu.au/baxter/20160127-LABCOT-HIx4.mp4

Acknowledgments

We thank the anonymous reviewers of an earlier version for their helpful comments and suggestions that led to a more concise formalisation of our model. This material is based upon work supported by the Asian Office of Aerospace Research and Development (AOARD) under Award No: FA2386-15-1-0005. This research was also supported under Australian Research Council's (ARC) *Discovery Projects* funding scheme (project number DP 150103035). Michael Thielscher is also affiliated with the University of Western Sydney.

Disclaimer

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the AOARD.

References

[Albus and Meystel, 2001] James S. Albus and Alexander M. Meystel. *Engineering of Mind: An Introduction to the Science of Intelligent Systems*. Wiley-Interscience, 2001.

[Anderson, 1993] John R. Anderson. *Rules of the Mind*. Lawrence Erlbaum Associates Inc, New Jersey, July 1993.

[Baars, 1988] Bernard J. Baars. *A Cognitive Theory of Consciousness*. Cambridge University Press, 1988.

[Beetz *et al.*, 2010] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM – a cognitive robot abstract machine for everyday manipulation in human environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017, Taipei, Taiwan, October 2010.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, Mar 1986.

[Brooks, 1990] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems 6*, pages 3–15, 1990.

[Clark and Robinson, 2015] Keith Clark and Peter Robinson. Robotic agent programming in TeleoR. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5040–5047, May 2015.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research (JAIR)*, 13:227–303, 2000.

[Haber and Sammut, 2012] Adam Haber and Claude Sammut. A cognitive architecture for autonomous robots. *Advances in Cognitive Systems*, 2:257–275, December 2012.

[Laird *et al.*, 1987] John E. Laird, Allen Newell, and Paul S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, September 1987.

[Langley and Choi, 2006] Pat Langley and Dongkyu Choi. A unified cognitive architecture for physical agents. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 1469–1474. AAAI Press, 2006.

[Levesque *et al.*, 1997] Hector J. Levesque, Raymond Reiter, Yves. Lespérance, Fangzhen. Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[Minsky, 1986] Marvin Minsky. *The Society of Mind*. Simon & Schuster, Inc., New York, NY, USA, 1986.

[Newell and Simon, 1976] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, March 1976.

[Nilsson, 1994] Nils J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research (JAIR)*, 1:139–158, 1994.

[Nilsson, 2001] Nils J. Nilsson. Teleo-reactive programs and the triple-tower architecture. *Electronic Transactions on Artificial Intelligence*, 5:99–110, 2001.

[Nilsson, 2006] Nils J. Nilsson. The physical symbol system hypothesis: Status and prospects. In Max Lungarella, Fumiya Iida, Josh C. Bongard, and Rolf Pfeifer, editors, *50 Years of Artificial Intelligence, Essays Dedicated to the 50th Anniversary of Artificial Intelligence*, volume 4850 of *Lecture Notes in Computer Science*, pages 9–17. Springer, 2006.

[Ryan and Pendrith, 1998] Malcolm R. K. Ryan and Mark D. Pendrith. RL-TOPS: An architecture for modularity and re-use in reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 481–487, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[Thrun *et al.*, 2005] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.