Object schemas for grounding language in a responsive robot

Kai-yuh Hsiao[a*], Stefanie Tellex[a], Soroush Vosoughi[a], Rony Kubat[a], Deb Roy[a]

[a]MIT Media Laboratory, MIT, Cambridge, Massachusetts, USA

Abstract

We introduce an approach for physically-grounded natural language interpretation by robots which reacts appropriately to unanticipated physical changes in the environment and dynamically assimilates new information pertinent to ongoing tasks. At the core of the approach is a model of *object schemas* that enables a robot to encode beliefs about physical objects in its environment using collections of coupled processes responsible for sensorimotor interaction. These *interaction processes* run concurrently in order to ensure responsiveness to the environment, while coordinating sensorimotor expectations, action planning, and language use. The model has been implemented on a robot that manipulates objects on a tabletop in response to verbal input. The implementation responds to verbal requests such as "Group the green block and the red apple," while adapting in real-time to unexpected physical collisions and taking opportunistic advantage of any new information it may receive through perceptual and linguistic channels.

Keywords: *object schema; language grounding; human-robot interaction; semantics*

*Corresponding author. [a]Emails: {eepness, stefie10, soroush, kubat, dkroy}@media.mit.edu

Object-centred, behaviour-based language grounding

Natural language is an efficient means by which humans convey beliefs and desires to other people. Analogously, a robot system that uses natural language could communicate more efficiently with humans. The conceptual structures required for language use could also help the robot interact with an environment generally designed by humans for human use.

For a robot with a physical body to make sense of language, the robot's internal representations need to be *grounded* in the real world, meaning that the robot must have processes that relate its words and concepts to its perceptions and actions in systematic ways. For instance, a language-using robot must identify the perceptual referents of nouns, and it must translate verbs into appropriate motor behaviour. A physical robot, however, also needs to stay responsive to changes in its environment. This is the premise of behaviour-based robot design, as argued by Brooks [8] and numerous others. In this paper we present a model for language grounding that addresses the need for responsiveness without sacrificing the ability to form structured internal representations of a kind necessary to support symbolic communication.

At the core of the model are *object schemas*, in which running processes are organized according to the physical object towards which they are directed, and by which they are causally affected. The object schema processes run concurrently to ensure responsiveness of the robot to its environment. The object schemas built from the processes provide object representations for planning and language use. The model builds upon an underlying theoretical object schema framework developed in [32, 33].

To demonstrate the efficacy of these constructs, we have implemented a robotic system in which the object schemas assist the coordination of vision, touch, motor action, planning, and language use, in a tabletop domain. Figure 1 shows the robot, Trisk, performing a behaviour sequence enabled by the model.

In this paper we first situate our approach with respect to related work in the field. Then, we explain the model and describe selected aspects of the implementation in greater detail.

Background

Our model brings together behaviour-based design and language grounding, combined through a sensorimotor schema representation of objects. In this section we review some related work and highlight some benefits of our model.

SMPA and non-linguistic behaviour-based robots

From the days of Shakey (the first language-using robot [28]) the “traditional” approach for robot planning and control was what Brooks eventually called the “sense-model-plan-act” approach (SMPA). The idea behind SMPA is that sensory input is interpreted as a model of the external world, the model is used for planning purposes, and then the plans are used to guide action. The SMPA model naturally supports language use. For example, verbal commands can be translated into plans that are inserted into the plan layer of an SMPA system. Likewise, verbal descriptions of world state can be translated into elements of the model layer.

As an alternative to the SMPA approach, Brooks [7, 8] developed the subsumption architecture, an early example of behaviour-based robot control. Behaviour-based approaches focus first and foremost on the responsiveness of the robot to unexpected (and thus unmodelled) changes in the environment. The subsumption architecture maintains responsiveness by decomposing a task into separate behaviours and having a dedicated process handle all aspects of each specific behaviour, from sensory input to motor output.

Behaviour processes interact through layered control interrupts. Brooks identified the use of internal models of the environment as the primary culprit for impeding responsiveness to change, and as a result, argued against explicit internal models altogether. This prohibition, however, leaves no clear path for language processing, which is fundamentally a "representation-hungry" task [14].

Later work on behaviour-based robots have introduced some degree of representation and planning by layering these elements on top of reactive processes. For instance, some of the three-layer systems developed to date [10, 11, 19, 20] allow reactive behaviours (such as collision-avoidance) to compete with more complex planned behaviours (such as mapping and path planning) as part of an action selection process. Representations in such systems focus primarily on behaviours and task decomposition.

Object representation

Behaviour-based robots have been successfully deployed in a number of real world systems, demonstrating how much can be accomplished with a minimum of representation. However, the raw sensorimotor processing of the subsumption architecture and the action representations of the later systems have limitations with respect to human communication and acting in human-centric environments. Adding representations of physical objects is important for several reasons:

Communicating about objects Human language use is full of statements about objects, suggesting that explicitly representing objects will enable the robot to efficiently handle such language.

Planning actions Representing objects would allow a robot to plan coherent sequences of actions towards goals expressed in terms of objects and their states.

Coordinating expectations The conception of objects and object permanence may provide an efficient means for a robot to generate structured predictions of incoming sensory input, and to predict changes in the environment as a result of specific actions taken by the robot.

Because of the importance of objects in communication and planning, our object schema model organizes processes according to the object they target. There are several other systems that explicitly represent physical objects. Here we relate them to our approach:

Object persistence and tracking Several projects inspired by behaviour-based design, such as those led by Blumberg et al. [12, 24] and later Breazeal et al. [5, 6] remain reactive while updating object representations continuously. However, in these models much of the object processing is decoupled from the object representations themselves, making it inconvenient to represent object-directed *affordances* (i.e., actions that can be taken and states that can be achieved by the agent, due to the object [21]) such as "liftable" and "graspable." Because the processes that target an object are contained within our object schemas, affordance information can be stored alongside the corresponding processes in our object representation. This simplifies the grounding of affordance-related words.

Affordance learning Several systems do represent object affordances explicitly, for instance for learning affordances [17, 18, 27, 39]. Learning is a vital future direction for our work as well, but our current emphasis is on developing a richer affordance-centred representation of objects that can support language use and responsive behaviour.

The model that is presented in this paper is influenced by the Piagetian idea of a sensorimotor schema [29]. Both Arbib [1] and Drescher [16] have developed computational interpretations of Piaget's concept of

schemas. Drescher's schema mechanism is an early attempt to fully implement schemas and test them in a simulated world. Drescher strived for a learning system that minimized "innate" structures, preferring a highly data-driven learning system that constructed primitive object schemas out of very low level sensorimotor primitives. In practice the implementations based on this approach failed to scale to environments beyond extremely simple grid-worlds. In contrast, our approach provides fairly rich built-in structures in the form of robotic controllers and perceptual processing routines with the goal of implementing more capable systems that demonstrate robust real world object-directed behaviour. Drescher's work suggests future directions for introducing structure learning capabilities into our approach by "deconstructing" our innate schema structures and enabling the system to reconstruct schemas based on experience.

Under the schema view, objects are a useful abstraction that brings structure to otherwise incoherent and overwhelming sensorimotor experience. A system with no object representation would have to explicitly encode an enormous network of sensorimotor expectations just to gain basic abilities like object persistence, while a system with an object representation can expect that a green object will continue to appear green. Object schemas thus enable the system to organize perceptions and actions in ways that matter to the system's programmed goals. Our model provides a computational/mechanistic interpretation of the schema concept with sufficient detail to implement on a physical robot.

Reactive grounded language robots

Several mobile robot systems include the ability to take verbal commands [3, 25, 26, 27]. Some of these robots can also run collision-avoidance behaviours in competition with verbally-commanded behaviours, using a mechanism like those in a subsumption-based system. In essence, such systems use SMPA-based language grounding grafted atop a behaviour-based reactive robot, granting the benefits of both a full representation and behaviour-based responsiveness. Figure 2 shows a depiction of this model.

However, our object schema approach is specifically *not* adding SMPA to a reactive robot. Rather than temporally interspersing responsive behaviours with language use and planning, our approach constructs object representations for language use and planning *using* the responsive behaviours, so a sensorimotor event can rapidly alter linguistic and planning structures and processes. A collision with an object not only causes our robot to pull back to avoid damage, but the process handling the collision also provides updated knowledge of the object's expected location, altering the target location for subsequent verbally-commanded grasping actions. In contrast, the reactive SMPA approach would use separate processes for collision-handling and for building a model from sensory input to locate objects for verbal command processing. Our object schemas are thus more adept at leveraging information from the reactive level. A high-level overview of our approach is depicted in Figure 3 to contrast with the preceding SMPA diagram.

Other grounded language systems

Numerous previous efforts have addressed language grounding either in physical or virtual worlds (see [31] for a survey of models of word grounding, and a collection of recent papers on the topic introduced in [35]). Winograd's SHRDLU [41] carried out natural language commands in a symbolic "simulator" of a blocks-world manipulation robot. While Winograd abstracted away many of the core issues we focus on here related to interacting with and representing real objects, SHRDLU's linguistic performance remains impressive to this day.

Many other language-grounding systems such as [2, 30, 36] connect aspects of language use or word meaning to perceptual features grounded in objects, but do not address the underlying problem of how a machine might conceive of objects in the first place. Several other researchers pairing language and robotics [13, 38, 40] focus on evolving novel languages between agents or robots trying to communicate. Our focus is

on representations specifically appropriate to pre-existing human languages that can be used in human-robot interaction tasks.

In our previous work we developed a robot called Ripley that is able to respond to spoken commands such as “Hand me the blue one on your left” [23, 34]. Ripley was not, however, responsive to changes or corrections, and action failures had no impact on decision-making beyond a simple retry mechanism. In contrast, our new approach uses a planning mechanism to make decisions, respond to sudden changes, and replan dynamically.

Interaction processes and the object schema model

As described in the “Reactive grounded language robots” section above, an SMPA model built on top of a reactive robot system could carry out verbal tasks, but its reactive processes would function independently of the verbal processing. In contrast, our object schema model coordinates language use, planning, and sensorimotor expectations by using object representations built *out of* the reactive processes. We call these processes *interaction processes* (in this context we sometimes just call them “processes” for brevity), and they form the basic building block of the model. The interaction processes are simultaneously organized into object schemas, which represent objects, and *plan hierarchies*, which coordinate action planning. For language purposes, the object schemas and plan hierarchies both serve as representations that language can be grounded to. For example, noun phrases can connect to the object schemas, and verb phrases in commands can connect to the plan hierarchies. In this section we describe the model in greater detail.

Interaction processes and histories

The fundamental building block of the model is the interaction process. Interaction processes perform tasks such as taking sensory input, producing motor output, and manipulating internal records. The interaction processes run concurrently and continuously, thus ensuring responsiveness. Each interaction process provides information to other processes by writing data to shared memory. By building object schemas and plan hierarchies out of interaction processes, information in the shared memory is coordinated between the sensorimotor modalities, planning, and language use.

Interaction processes have a main loop that can be started and stopped, and some processes exit when their main loops complete their programmed tasks. These tasks could include finding interesting visual regions or reaching the robot arm towards an object. Figure 4 shows a visual notation for four interaction processes with various run states (represented by the left-side semicircle) and various finish states (represented by the right-side semicircle).

Interaction histories The data that each interaction process writes to shared memory is called its *interaction history*. The interaction histories allow processes to provide information to each other. This use of shared memory somewhat resembles a traditional blackboard model, except that each process can only write to its dedicated space. There are two primary types of history data: *attribute data*, which is returned while a process is running, and *completion data*, which is returned when a process' loop is *completed* (e.g., finishing an attempt to touch an object).

Attribute data contains information to be incorporated into knowledge of the *attributes* of an object (such as colour, shape, and location). The attributes of an object can be continuous (e.g., colour and weight as measured from sensory input) or discrete (e.g., a labelled category for colour or weight). Completion data contains information about whether a process completed successfully or not, as well as timing information about the process' execution. Completion data, as compiled over time, is incorporated into knowledge of the *affordances* of an object (such as liftability or graspability). Figure 5 shows the two types of history data deriving from interaction processes.

Process classes Each interaction process in the system is an instance of a *process class*. Each process class defines the execution loop that its instances run. The execution loops for some of these classes exit at a defined point (e.g., finishing a grasping action) and the process reports completion data to its history. Other execution loops cycle forever, causing the process to write attribute data to its history until it is destroyed (e.g., its object schema is deleted). The following are the major types of process classes:

Sensory processes are interaction processes that monitor incoming sensory data and write relevant data to their interaction histories. Subtypes in our implementation include collision detectors, visual trackers, and touch trackers. The visual trackers watch visual input and provide continuously updated visual information about an object. The touch trackers watch touch sensor input to determine if an object is being grasped. If the object is grasped, the touch trackers output the weight of the object and the location of the object based on the hand's location. Each sensory process is intended to loop forever.

Action processes are interaction processes that, when activated by the planning system, send motor commands to the robot. Examples include a process to move the robot arm away from a collision, or to grasp the fingers around an object. An action process reports successful or failed completion when its attempt to perform its programmed actions is done.

Plan fragment processes operate in the planning system to coordinate actions and preconditions. For instance, the plan fragment for grasping an object could require the precondition that the hand be open and positioned at the object's location. Then, it would trigger the action for closing the fingers. Plan fragment processes declare themselves complete when their attempt to perform their programmed sequence is done. These processes are often created in response to verbal commands.

Condition processes continuously assess whether a specified condition is true or not, and when not true they trigger the planning system to search for plan fragment and action processes that can render them true. Condition processes loop endlessly until removed from their plan hierarchy, although for planning purposes they add a piece of completion data to their histories when their conditions are true.

Translation processes convert interaction history data from other processes to a new form, such as a transformation from visual 2-D data to 3-D coordinates for targeting the arm, or a categorization from continuous colour data to a discrete colour category. These processes loop forever, and the results of their translations are written to their own interaction histories. The discrete categories (such as colour or shape labels) are used for connecting noun phrases to object schemas.

Coordination processes maintain the integrity of object schemas by manipulating other interaction processes. Each object schema has a coordination process which monitors interactions between processes within that object schema. For instance, if visual and touch tracking disagree about an object's location, the coordination process can detach one of the processes from the object schema to resolve the conflict. These processes also loop forever, until destroyed along with their object schemas.

Reference processes receive noun phrases from speech input and attempt to connect the noun phrases to object schemas with matching attributes. For instance, "the green block" leads to a reference process that reads interaction histories of each active object schema to find one that best fits "green" and "block." These processes exit and report completion when a matching object schema is identified.

Object schemas and the belief context

Interaction processes can be incorporated into (*bound to*) structures called object schemas. Each object schema represents one physical object in the system's environment, and consists of a container that holds interaction processes that are considered to be "about" the represented object. Object schemas are stored in the *belief context*, which is a structure that stores object schemas along with accumulated affordance information. The belief context constitutes the set of beliefs that the system currently holds about its environment and its affordances.

When an interaction process is bound to an object schema, history data for that process becomes associated with the object schema as well, and is treated as attribute or affordance information for the represented object. The object schemas act as discrete entities for planning and language use. Figure 6 depicts an object schema that consists of several interaction processes and, by association, their histories.

Not all interaction processes are bound to object schemas. Processes that provide incoming sensory input (such as collision detection and visual segmentation) typically remain unbound.

Object expectations When an interaction process is part of an object schema, its interaction history can be used to generate expectations of future history data. Expectation data is stored with the interaction process, alongside the history data that is being predicted. There are two types of expectation data: *expected attributes*, which predict attribute data, and *expected affordances*, which predict completion data. Expectations are used by the planning system to select a course of action. These expectations of future interactions would not be as straightforward to compute without the use of an object-centred framework, as mentioned in "Object representation," above. The visual notation for expectations is given in Figure 7.

Completion statistics In addition to object schemas, the belief context also stores statistics on completed action and plan fragment processes. These statistics are indexed by the attribute data of the object schema that the processes were bound to. As an example, if a lift action towards a heavy red apple failed, then the belief context would note the failure of lifting with respect to the attributes for "heavy," "red," and "apple." Over time, trends emerge in these statistics; the "heavy" attribute might be a good predictor of difficulty lifting an object. These statistics are used to compute expected affordances, which in turn are used by the planning system to make decisions based on success likelihood. This learning of affordances for objects and attributes is the main type of learning present in our current system, although perceptual categories such as shapes and colours are also learned in a supervised manner in the current system.

The association of success statistics with discrete object attributes such as "heavy" has an interesting ramification. At first glance, discrete attributes such as colour and weight labels ("red" or "heavy") seem relevant to language use but have no efficacy in the robot's planning and actions. Linking attributes to success statistics allows the attributes to predict affordances, thus linking the attributes to the physical capabilities of the robot. Discrete object attributes then mean something in terms of the embodiment of the system, rather than only as a means of jointly referencing objects. This relates to the reason an embodied agent would represent discrete attributes in the first place -- awareness of discrete categories such as "heavy" helps to categorize objects based on their physical affordances.

Plan hierarchies and the planning system

The planning system stores and manipulates plan hierarchies. Like object schemas, plan hierarchies are structures composed of interaction processes. Each hierarchy is a tree structure that organizes action, condition, and plan fragment processes into a coherent sequence to address a primary motivation at the root of the tree.

The root of each plan hierarchy is a primary motivation, which represents a basic "need" that the system attempts to satisfy by attaching appropriate interaction processes to the hierarchy. The primary motivations have *priority scores* that vary over time, reflecting the changing priority for each basic need. The priority scores are passed along the hierarchy from parents to children.

The non-root nodes are interaction processes that the planning system attempts to run to completion. For each node *X*, *X*'s children are the interaction processes which must complete before *X* can. For instance, a node responsible for closing the hand around an object will have a child node that ensures that the robot's hand is at the object's position before attempting to perform the grasp. A part of such a hierarchy is shown in Figure 8.

Plan hierarchies are built such that their leaf nodes form a sequence of action processes that will serve the primary motivation. Across all the plan hierarchies, only the action process with the highest priority score is permitted to run at a given time. Thus, the primary motivations compete according to their current priority score to build hierarchies and have the robot take actions in their service.

The construction of a plan hierarchy proceeds as follows:

1. For the primary motivation with the highest priority score, the planning system examines a list of action and plan fragment processes that are known to satisfy the primary motivation. When multiple processes are available, the expectations computed in the belief context are consulted to determine which is most likely to succeed. The best action or plan fragment process is then attached as a child node, and the priority score is propagated to it.

2. When a plan fragment process is the highest priority leaf node, the first element in its sequence (either an action or a condition process) is created and attached as a child. The priority score is passed to the child, which eventually leads to the satisfaction of preconditions and the execution of actions. When a child completes, it is detached by the planning system and the next element in the plan fragment's sequence is created and processed.

3. When a condition process is the highest-priority leaf node, the planning system conducts a search and attaches a child in the same way as for a primary motivation.

4. Action processes are always leaf nodes. Action processes compete to execute based on the priority scores propagated from the primary motivations. When an executing action process is complete it is detached from its parent so the hierarchy's sequence can continue.

By choosing appropriate plan fragments to coordinate actions and conditions, the planning system executes a coherent sequence of actions that satisfies the strongest current motivation of the system. The planning system interacts with the belief context by using object schemas as targets for action and also as sources of likelihood data to decide between possible actions. Because each process in a plan hierarchy is

also part of an object schema, changes recorded in interaction histories can rapidly affect likelihood data and cause revision of the plan hierarchy.

Language interaction

Incoming speech is provided to the model in the form of a parse tree. The tokens in the parse tree can then be connected to representations in the model. This primarily involves adding structures to the planning system in response to verb phrases, and connecting a noun phrase to an object schema that it refers to.

When a verb phrase is received (in a command, for instance), it is assigned an appropriate structure in the planning system, such as a condition process that reports satisfaction when the command is executed. By attaching the structure to a primary motivation and setting the priority score, the planning system then takes responsibility for carrying out the relevant actions.

Object-directed interaction processes are typically part of the object schema of their targeted object. However, an object-directed process based on speech input has no initial target, because speech input can only provide a noun phrase that describes the target object; it cannot identify a specific object schema without further processing. Because the object-directed processes cannot join an object schema until the system identifies the appropriate object schema, they are connected to a *reference process* until the correct object schema is found.

Reference processes are the interaction processes responsible for connecting object schemas to noun phrases. They search the object schemas in the current belief context for one with discrete attributes (which are automatically produced by translation processes) that correspond to the tokens in the noun phrase. For each noun phrase, a reference process is created and provided with the tokens from the phrase. The process then searches the current object schemas for one with the appropriate discrete attributes. When a match is found, all object-directed interaction processes that are connected to the reference process are added to the matching object schema. Speech-driven action and plan fragment processes in plan hierarchies cannot be activated until their reference processes are resolved and they have been added to object schemas.

Summary

In the model, interaction processes handle sensory input, motor output, and internal recordkeeping. These interaction processes are organized into object schemas, each representing one physical object, and plan hierarchies, each representing one coherent sequence of actions. Language is processed by connecting noun phrases to object schemas and verb phrases to structures in the plan hierarchies.

This approach presents several benefits relative to the other approaches mentioned in the Background section. Our model extends behaviour-based systems by explicitly representing objects, and it extends language-grounding systems by maintaining responsiveness to the environment. Furthermore, by building object schemas from reactive processes, our model gains several features over an SMPA language system added to a reactive robot:

Computational scalability Instead of performing a full round of processing on current sensory inputs, our object schema model permits each bit of sensory input to affect only specific aspects of planning and language use. For example, a collision with an object can trigger a cascade of processing that directly changes the target position for subsequent grasping actions, while an SMPA approach would wait for the next modelling cycle to process all sensory input. By decoupling the processing of sensory inputs from each other, our approach decreases the latency time from sensory input to motor action, and renders our approach more amenable to parallelization across multiple processors or computers. Decisions can be made and altered in our model based on the interaction of a small subset of processes, rather than waiting for a complete model to

be generated from new sensory input. As grounded language systems increase in complexity from the current set of relatively limited domains, our approach offers a path for scalability.

Common currency Our model, by collecting all processes directed towards an object into the object schema, provides a convenient means of interaction between language, planning, and sensory processes. For example, hearing "the apple is heavy" enables our system to alter its plans based on this information. The system would also have arrived at the same decision if it had prior physical experience with the apple, or with objects with visual similarity to the apple. Likewise, if it has to lift the apple anyway, it can decide for itself how heavy the apple is, regardless of what it had been told. This convergence of multiple modalities, including language, would be more complicated to implement in an SMPA model.

Multimodal affordances Finally, our object schemas provide a convenient way to represent affordances, such as liftability and graspability, for grounding affordance-related words. This coordination of information from multiple sources in an object-centric fashion would also be difficult in a model where planning and action are represented separately from the objects themselves.

Implementation walkthrough

The model has been implemented on a robot platform, named Trisk. In this section we describe the physical and sensorimotor systems of the robot, and then we discuss key aspects of the implemented model as it produces a sequence of behaviours.

The robot and the sensorimotor systems

Our robot platform, named Trisk (Figure 1 shows the robot in action), is a six degree of freedom (DOF) robotic arm with a four-DOF Barrett Hand as its end effector, situated in front of a table on which manipulable objects are placed. Six-axis force-torque sensors on each of the three fingers of the hand enable sensing of forces and collisions, including awareness of successful grasps. The downwards force of an object on the finger sensors provide the weight of the object. Two cameras (only one is currently active for simplicity) sit in a head mounted on a four-DOF neck, which allows the system to adjust its view of the environment and look up at the human for interactivity purposes.

Visual input from the active camera is sent through a colour-based segmentation algorithm (based on CMVision [9]) that groups contiguous regions by colour. The current set of objects used for robot interactions consists of simple objects of uniform colour, so colour segmentation suffices for our purposes. Visual input is also processed on demand by a mean-shift tracker [15] based on edge and colour profile, and a 2-D shape recognition algorithm based on shape contexts [4] when requested by vision-related interaction processes. Visual information thus derived includes the size, shape, colour, and location of each object, and is discretized by translation processes for matching with incoming description words (such as "red" and "ball"). Visual locations are transformed into arm-space coordinates for grasping by assuming that objects are on the plane of the table and performing the appropriate vector math.

The motor control modules for the robot's arm and hand compute forward and inverse kinematics, so the hand can be brought via a smooth trajectory towards reachable 3-D coordinates. The fingers can be spread to enable grasping, or moved together to tap an object. Touch input from the fingers is used along with arm kinematic information to provide the location, direction, and magnitude of contact forces between the fingers and physical objects. Collisions are detected by monitoring forces on the fingertip sensors, torques on the arm motors, and computing positions relative to hard-coded nearby surfaces.

It should be noted that the robot's hand and arm significantly occludes the camera view of the workspace. The kinematics of the arm and hand are used to create what we call an "occlusion zone" in the

visual input. In this area of visual input, apparent visual objects are considered untrustworthy and thus not instantiated by the model.

Speech input is collected by a microphone headset worn by the human, and passed through the Sphinx 4 free speech recognizer and a probabilistic Earley parser (from [22]). The resulting parse trees provide structured tokens to be connected to object schemas and plan hierarchies.

An example behaviour sequence

In this section we narrate a sequence of behaviours (most of which is pictured in Figure 1) and discuss how the implementation produces this sequence. The sequence consists of the following behaviours:

1. The robot is looking at a table with a blue cup and a yellow ball on it. A green block and a red apple are then placed on the table. The system tracks the objects visually, maintaining the object representations throughout new visual frames.

2. The human says "Group the green block and the red apple." The robot reaches towards the red apple in order to move it towards the green block.

3. The human interrupts the robot's action, saying "The red apple is heavy." The robot reaches for the green block instead.

4. The robot misses the green block, and its wrist collides with the block. The robot immediately pulls away from the block, avoiding damage. Assuming it collided with the block, the system revises its location estimate for the block.

5. The robot adjusts its targeting, lifts the block, and moves it next to the red apple.

Behaviour 1

The scenario begins with the robot's head facing the table with the blue cup and yellow ball. At this point, two object schemas already exist within the system, one representing the cup and the other representing the ball. Both of these object schemas have been created in response to the visual input.

Visual tracking The green block is then placed on the table. Figure 9 depicts the processing of the subsequent frame of visual input. This processing requires extracting information about both previously-known objects, as well as starting to represent and track the green block.

The three visual regions are detected by the unbound sensory process that handles visual segmentation, and written to its interaction history. Each of the pre-existing object schemas includes a visual tracking process for identifying regions in new visual frames that correspond to their objects. Past history data for each object schema leads to visual expectations for the object's current appearance. These expectations are compared to the regions in the new frame. The closest match for each object schema is found according to a distance metric and then "claimed" by the schema's visual tracker.

New schema creation The third region written by the segmenter process goes unclaimed. As depicted in Figure 10, the unclaimed region triggers the creation of a new object schema. New object schemas start out

with a coordination process, which is responsible for subsequently creating the other processes that will maintain the object schema's data and expectations.

At the outset, a typical object schema consists of:

- The coordination process that bootstraps the other necessary processes and removes processes when inconsistent data is detected.

- Visual and touch tracking processes to assimilate incoming sensory data.

- Translation processes to convert continuous sensory data (such as weight and colour) into discrete attribute categories (corresponding to word labels), and to convert between visual and touch coordinate systems.

The visual tracker for each object provides information derived directly from the visual input frame, such as average colour, centroid location, and occupied pixel list. The translation processes then convert these into other attributes, such as colour category, shape, and location in physical space. The expected location in physical space is used for subsequent reaching and grasping actions.

Action, condition, and plan fragment processes are later added to some object schemas due to planning and language activity. An action process will be bound to an object schema if the action is targeted towards the corresponding object (such as moving towards or lifting the object). A condition process will be bound to an object schema if the condition being tested involves an attribute of the object schema (such as the object being at a specific location). Likewise, a plan fragment process is bound to an object schema if actions and conditions within the plan fragment are targeted towards the object.

In subsequent frames, the green block is tracked by its new object schema. The red apple is then handled similarly.

Behaviour 2

The human says “Group the green block and the red apple.” Speech recognition and the language parser convert this verbal input into a parse tree. The parse tree triggers the creation of a plan hierarchy to coordinate the requested actions. The noun phrases in the parse tree must be connected to object schemas. In order to narrate these mechanisms, we frame the discussion by first describing the full range of verbal inputs and motivations of our implementation.

Types of speech input We divide our speech inputs into four types:

Queries Requests for information that require a verbal response, such as “Is the green block near the red apple?” and “Describe the block.”

Directives Requests for the robot to perform motor action, such as “Touch the red apple” and “Move the block to the right.”

Assertives Statements about the state of an object, such as “The red apple is heavy.”

Correctives Partial directives which cause replacement in the immediately preceding full directive, such as “Pick up the red ball... no, the green ball.” Correctives are handled in a preprocessing step, by making the substitution into the preceding directive.

In this example, the speech input is a directive, and leads to a plan hierarchy that attempts to execute a sequence of actions.

Primary motivations The implementation makes use of three primary motivations:

Safety The system makes plans to avoid taking damage. In the current implementation this entails avoiding and moving away from physical collisions.

Social The system attempts to interact with the human partner by answering queries and carrying out requested actions.

Curiosity The system explores objects by attempting to grasp and lift them. In doing so, the system learns about object affordances by automatically compiling statistics about completed action processes.

Speech inputs that request action are processed by attaching a plan fragment process to the social motivation. For a directive, the verb is used to determine the sequence of actions that constitutes the plan fragment. The social motivation is then given a sufficient priority score to cause the planning system to carry out the given task.

Reference processes Once the plan fragment process inherits the priority score, the planning system will attempt to expand its plan hierarchy. However, processes in a plan hierarchy cannot be handled by the planning system until it is clear which object schemas they are directed towards.

The parse tree for the directive contains two noun phrases, “the green block” and “the red apple.” A reference process is generated for each noun phrase and assigned to an expression based on the parse tree. As an example, the reference process for “the red apple” is assigned the following expression:

```
(refexpr (= function
            (lambda (x) (p_and
                          (red x) (apple x))))
         (= definite "definite")
         (= cardinality "1"))))
```

The reference process runs a search function that matches object schemas based on category attributes provided by translation processes that were trained in a supervised fashion. When an object schema with matching attributes is identified, the processes connected to the reference process (in this case, the plan fragment process for “group”) are then bound to the object schema so planning can continue.

Plan hierarchy construction Once an object schema is found for each of the two noun phrases, the planning system can continue by expanding the first sequence element of the “group” plan fragment. The sequence consists of a single condition, which tests whether the locations associated with the two schemas are

near each other. This "grouped" condition process is created and attached as the child of the plan fragment. It then inherits the priority, and the planning system attempts to satisfy this condition.

The expected results of each plan fragment is programmed with the plan fragment, so the planning system can search the list of expected results to find suitable plan fragments. The planning system can select one of two "move object" plan fragments, one that moves one object and one that moves the other.

Expected affordances In order to select between two possible plan fragment processes, the planning system creates both within their respective object schemas. Once this is done, the belief context automatically assesses the expected affordances. This involves examining the history data of the object (i.e., how many times the object itself has been successfully moved) and the attribute-linked completion statistics (i.e., for the red apple, how often red objects are moved successfully, and how often apples are moved successfully).

Based on these statistics, a success likelihood is generated for each of the prospective plan fragment processes. The planning system will read these likelihoods and then select the plan fragment with the best chance of success. Figure 11 depicts the plan hierarchy as it considers the choice. Figure 12 shows the structure of this hypothetical hierarchy after it has been fully built and executed.

Behaviour 3

The robot has moved its open hand towards the apple in order to move it. At this point, the human says "The red apple is heavy." The robot changes its mind and reaches for the green block instead. This behaviour is accomplished by processing the assertive input, updating expected affordances, and revising the plan.

Assertive input and expected attributes "The red apple is heavy" is a speech input that asserts a belief about an object. First, a reference process identifies the red apple's object schema. Then, the label "heavy" is written as an expected attribute for the history data of the weight-categorizing translation process.

Affordance monitoring and plan revisions As soon as the expectation of "heavy" is added to the apple's object schema, the belief context re-evaluates the expected affordance of whether the apple can be moved by the robot. Objects with the "heavy" attribute often cause lifting and moving processes to fail, and so the belief context records that any plan fragment processes attempting to lift and move the apple are likely to fail.

This in turn causes the active "move object" plan fragment to expect failure, leading to a revision of the plan hierarchy. The system opts to move the block instead, and the robot proceeds to do so.

Behaviour 4

As the robot hand approaches the block, its wrist collides with the block rather than encircling the block with the fingers. It immediately pulls away from the block

Collision handling and claiming unbound data Upon detecting the collision, the collision detection process writes information about the collision to its history. This is noticed by the primary motivation for safety, which immediately increases its priority score to the maximum possible. The action process for moving away from a collision runs immediately, and pulls the arm away. The safety motivation's priority score then drops back to zero.

The collision location provided by the collision detection process is also monitored by the touch trackers of all object schemas. Based on distance from an object's location estimate, each touch tracker can claim the new collision location as information about its object, writing the collision location to its interaction history. This is analogous to the claiming and writing of visual segment data by the visual trackers.

In this example, the collision location is close to the block's expected location, and so the block's touch tracker revises its location estimate in physical space based on the collision.

Behaviour 5

After the interruption by the safety motivation, the social motivation resumes control. The "move hand" action process was deactivated before it completed, and so the plan hierarchy restarts the action process, which makes use of the new location estimate provided by the green block's touch tracker. The hand arrives at the green block, closes around it, moves it next to the apple, and releases it.

Multimodal tracking As the hand closes around the green block and begins to lift it, the touch tracker for the green block begins writing weight and location information to its interaction history. The touch tracker's location estimate is based on the location from the forward-kinematics for the hand.

However, the visual tracker is continuing to provide visual locations based on the visual region associated with the block. Supposing that the visual tracker and touch tracker were to provide location estimates with substantial discrepancy, the coordination process for the object schema then has to step in and reconcile the information. It does this by removing the touch tracker from the object schema. The newly-unbound touch tracker will be used as the basis for a new object schema, just as unclaimed visual regions lead to new object schemas.

By concurrently tracking objects in multiple modalities, the model potentially provides a foundation for coordinating expectations between modalities in greater detail. For example, physically rotating an object may alter its perceived visual shape. This is left for future work. The model could also benefit from the ability to merge and split object schemas, to handle cases where multimodal tracking discovers an error such as one perceived object turning out to be two, or vice-versa.

Discussion

The value of our model lies in its ability to retain a responsive coupling between its language use, planning, and sensorimotor modalities. Here, we review some of the behaviours that underlie this ability:

1. If the robot is reaching for an object and it misses slightly, colliding its wrist with the object, it will respond to the collision by moving away from the point of contact. The point of contact will immediately be claimed by the object schema of the targeted object, which revises its expected physical location, which causes the robot to adjust its grasping target accordingly.

2. If the robot is reaching to move a red ball, and it accidentally lifts a nearby green block instead, the visual trackers will observe that the red ball is not moving, and will put down the green block and retry the grasp. This is the result of a discrepancy at the object schema level between the grasp trackers and the visual trackers, which is resolved by the coordination process.

3. When the human says "Group the red apple and the green block... The red apple is heavy," the new verbal information about the apple's weight is represented as an expectation in the same way as the equivalent physical interaction. This leads to a change in the expected affordances in the object schema, which leads to a re-evaluation of the planning system's hierarchy. The robot will reach for the green block instead.

4. Suppose the human says "Group the red apple and the green block," without specifying any additional object information. If the robot then attempts to lift the red apple and repeatedly fails, this affects the expected affordances in the object schema in the same way, leading to the same re-evaluation.

The data-sharing in our model that enables these sorts of responsive behaviours is made possible by the presence of interaction processes in both the object representations and the plan representations. In Example 1, the touch tracker notices the touch contact near the expected location of its object schema, and claims the point of contact as another piece of expectation information. In Example 2, the two trackers and the coordination process form a short information path between the touch and vision modalities, enabling a local correction. In Example 3, new verbal information adjusts an expected affordance, so the belief context alters the success likelihood for the "move object" plan fragment. This plan fragment process is also present in the plan hierarchy, enabling immediate re-evaluation by the planning system. Example 4 shows the same mechanism with direct physical interaction.

Conclusion

We have developed and implemented a model of object schemas that brings together responsive interactive robotics with natural language understanding. By building object schemas and plan hierarchies from the processes that govern responsive interaction, planning and language use are capable of leveraging information from the reactive processes directly and immediately. This stands in contrast to previous hybrid approaches that graft behaviour-based reactive control layers to SMPA-based architectures.

Although there are many limitations to the model, the most critical one in our view is that currently all schema structures are manually constructed by human designers. As a result, we have had to take great care in creating physical environments that are consistent with the hand-coded structures of our robot. In the future we plan to turn this situation around by enabling robots to construct object schemas -- and higher order relational schemas as well -- that are consistent with whatever environment they find themselves in.

Acknowledgments

This paper is partly based upon work supported under a National Science Foundation Graduate Research Fellowship. We thank Bruce Blumberg and Rod Brooks for discussions that helped shape the work presented here. We also thank Jonathan Huggins, Thananat Jitapunkul, Peter Lai, John McBean, Kailas Narendran, Philipp Robbel, Kleovoulos Tsourides, David Wang, and Jonathan Wang, for their time and effort with numerous aspects of the robot system implementation.

References

[1] M.A.Arbib, T. Iberall, and D. Lyons. *Schemas that integrate vision and touch for hand control.* In *Vision, Brain, and Cooperative Computation*, M.A. Arbib and A.R. Hanson, eds., MIT Press, 1987, pp. 489–510.

[2] C. Bauckhage, J. Fritsch, K. Rohlfing, S. Wachsmuth, and G. Sagerer. *Evaluating integrated speech and image understanding.* In Proceedings of the IEEE International Conference on Multimodal Interfaces (ICMI), 2002, pp. 9–14.

[3] P. Beeson, M. MacMahon, J. Modayil, A. Murarka, B. Kuipers, and B. Stankiewicz. *Integrating multiple representations of spatial knowledge for mapping, navigation, and communication.* In Proceedings of the AAAI 2007 Spring Symposium on Control Mechanisms for Spatial Knowledge Processing in Cognitive / Intelligent Systems, 2007.

[4] S. Belongie, J. Malik, and J. Puzicha. *Shape matching and object recognition using shape contexts.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(24), 2002, pp. 509–522.

[5] M. Berlin, J. Gray, A.L. Thomaz, and C. Breazeal. *Perspective taking: An organizing principle for learning in human-robot interaction.* In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI), 2006.
[6] C. Breazeal, M. Berlin, A. Brooks, J. Gray, and A.L. Thomaz. *Using perspective taking to learn from ambiguous demonstrations.* Journal of Robotics and Autonomous Systems, Special Issue on Robot Programming by Demonstration, 54(5), 2006.
[7] R.A. Brooks. *A robust layered control system for a mobile robot*. IEEE Journal of Robotics and Automation, 2, 1986, pp.14–23.
[8] R.A. Brooks. *Intelligence without representation*. Artificial Intelligence, 47, 1991, pp.139–160.
[9] J. Bruce, T. Balch, and M. Veloso. *Fast and inexpensive colour image segmentation for interactive robots.* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2000.
[10] J. J. Bryson. *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*. PhD thesis, Massachusetts Institute of Technology, 2001.
[11] J. J. Bryson and L. A. Stein. *Modularity and design in reactive intelligence*. In Proceedings of the International Joint Conference on Artificial Intelligence, 2001, pp. 1115–1120.
[12] R. Burke, D. Isla, M. Downie, Y. Ivanov, and B. Blumberg. *Creature smarts: The art and architecture of a virtual brain*. In Proceedings of the Game Developers Conference, 2001, pp. 147–166.
[13] A. Cangelosi. *The grounding and sharing of symbols*. Pragmatics and Cognition, 14, 2006, pp. 275–285.
[14] A. Clark and J. Toribio. *Doing with representing.* Synthese, 101, 1994, pp. 401–431.
[15] D. Comaniciu and P. Meer. *Mean shift: A robust approach toward feature space analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5), 2002.
[16] G. Drescher. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, 1991.
[17] P. Fitzpatrick. *From First Contact to Close Encounters: A Developmentally Deep Perceptual System for a Humanoid Robot*. PhD thesis, Massachusetts Institute of Technology, 2003.
[18] P. Fitzpatrick and G. Metta. *Grounding vision through experimental manipulation.* Philosophical Transactions of the Royal Society: Mathematical, Physical, and Engineering Sciences, 361(1811), 2003, pp. 2165–2185.
[19] E. Gat. *Integrating planning and reaction in a heterogeneous asynchronous architecture for controlling mobile robots.* In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI), 1992.
[20] E. Gat. *Three-layer architectures*. In *Artificial Intelligence and Mobile Robots*, D. Krotenkamp, R.P. Bannasso, and R. Murphy, eds., AAAI Press, 1998.
[21] J. J. Gibson. *The Ecological Approach to Visual Perception*. Erlbaum, 1979.
[22] P. Gorniak and D. Roy. *Situated language understanding as filtering perceived affordances.* Cognitive Science, 31(2), 2007, pp.197–231.
[23] K. Hsiao and D. Roy. *A habit system for an interactive robot*. In AAAI Fall Symposium: From Reactive to Anticipatory Cognitive Embodied Systems, 2005.
[24] D. Isla, R. Burke, M. Downie, and B. Blumberg. *A layered brain architecture for synthetic creatures.* In Proceedings of International Joint Conferences on Artificial Intelligence, 2001.
[25] L.S. Lopes and J.H. Connell. *Semisentient robots: Routes to integrated intelligence.* IEEE Intelligent Systems, 16, 2001, pp. 10–14.
[26] L.S. Lopes and A. Teixeira. *Human-robot interaction through spoken language dialogue.* In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.

[27] J. Modayil and B. Kuipers. *Where do actions come from? Autonomous robot learning of objects and actions.* In Proceedings of the AAAI 2007 Spring Symposium on Control Mechanisms for Spatial Knowledge Processing in Cognitive / Intelligent Systems, 2007.
[28] N. J. Nilsson. *Shakey the robot.* Technical Report 323, AI Center, SRI International, 1984.
[29] J. Piaget. *The Construction of Reality in the Child.* Basic Books, 1955.
[30] T. Regier and L. Carlson. *Grounding spatial language in perception: An empirical and computational investigation.* Journal of Experimental Psychology, 130(2), 2001, pp. 273–298.
[31] D. Roy. *Grounding words in perception and action: Computational insights.* Trends in Cognitive Science, 9(8), 2005, pp. 389–396.
[32] D. Roy. *Semiotic schemas: A framework for grounding language in action and perception.* Artificial Intelligence, 167(1-2), 2005, pp.170–205.
[33] D. Roy. *A mechanistic model of three facets of meaning.* In *Symbols and Embodiment*, M.D. Vega, G. Glennberg, and G. Graesser, eds., Oxford University Press, 2008.
[34] D. Roy, K. Hsiao, and N. Mavridis. *Mental imagery for a conversational robot.* IEEE Transactions on Systems, Man, and Cybernetics, 34, 2004, pp. 1374–1383.
[35] D. Roy and E. Reiter. *Connecting language to the world.* Artificial Intelligence, 167(1-2), 2005, pp. 1–12.
[36] J.M. Siskind. *Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic*. Journal of Artificial Intelligence Research, 15, 2001, pp. 31–90.
[37] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. *Spatial language for human-robot dialogs*. IEEE Transactions on SMC Part C, Special Issue on Human-Robot Interaction, 34(2), 2004, pp. 154–167.
[38] L. Steels and T. Belpaeme. *Coordinating perceptually grounded categories through language: A case study for colour.* Behavioural and Brain Sciences, 28, 2005, pp. 469–529.
[39] A. Stoytchev. *Behaviour-grounded representation of tool affordances.* In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2005.
[40] P. Vogt and F. Divina. *Social symbol grounding and language evolution.* Interaction Studies, 8(1), 2007, pp. 31–52.
[41] T. Winograd. *Understanding Natural Language.* Academic Press, 1972.

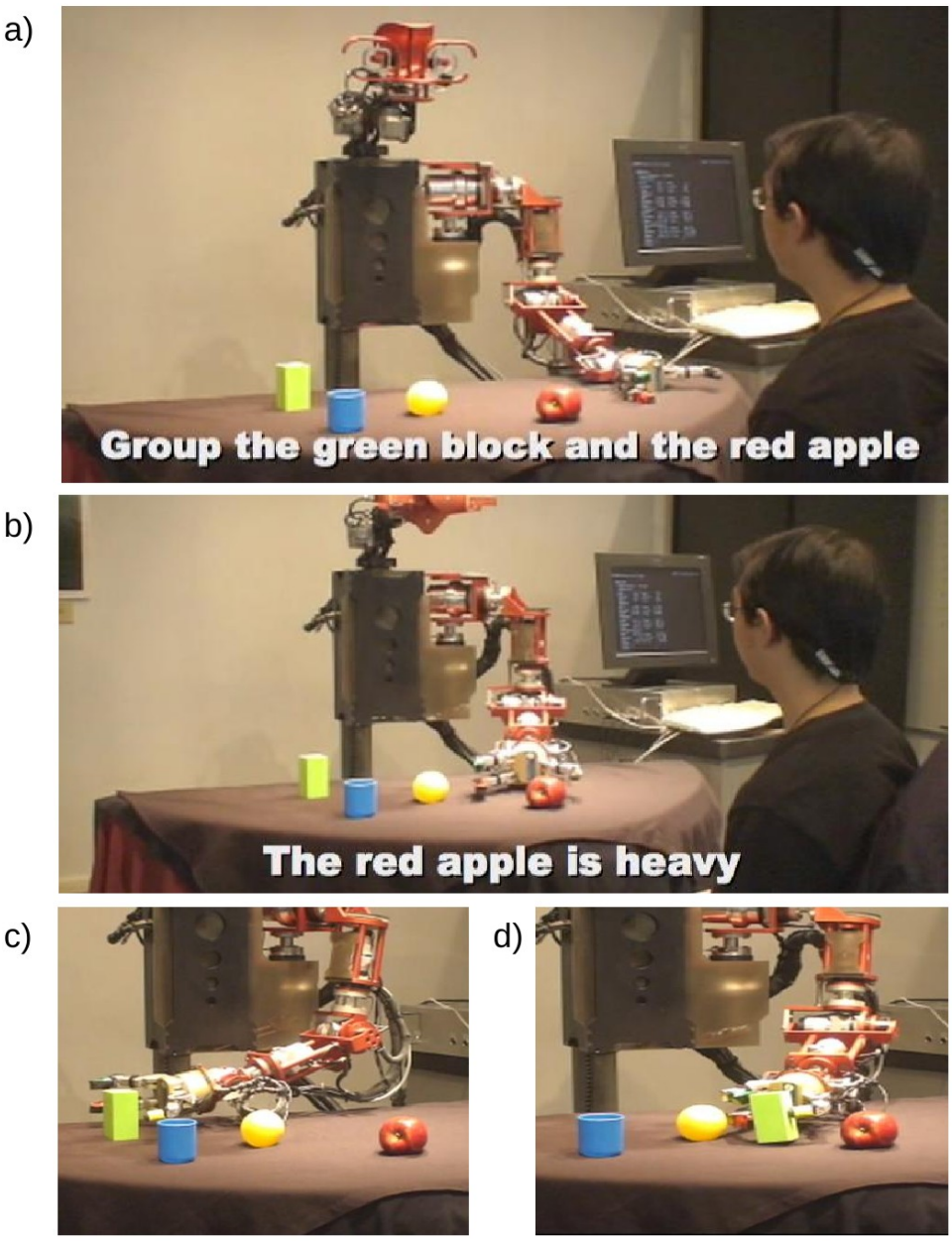

Figure 1. The robot, Trisk, is facing a scene that includes a red apple and a green block. a) Trisk is told, "Group the green block and the red apple." This request could be satisfied by moving the block towards the apple, or vice versa. The robot decides to move the apple. b) While the robot reaches for the apple, the human adds, "The red apple is heavy." Knowing that heavy objects are more difficult to lift, the robot changes its plan and c) moves the green block d) towards the apple instead. This example demonstrates the responsiveness of the system to new information.

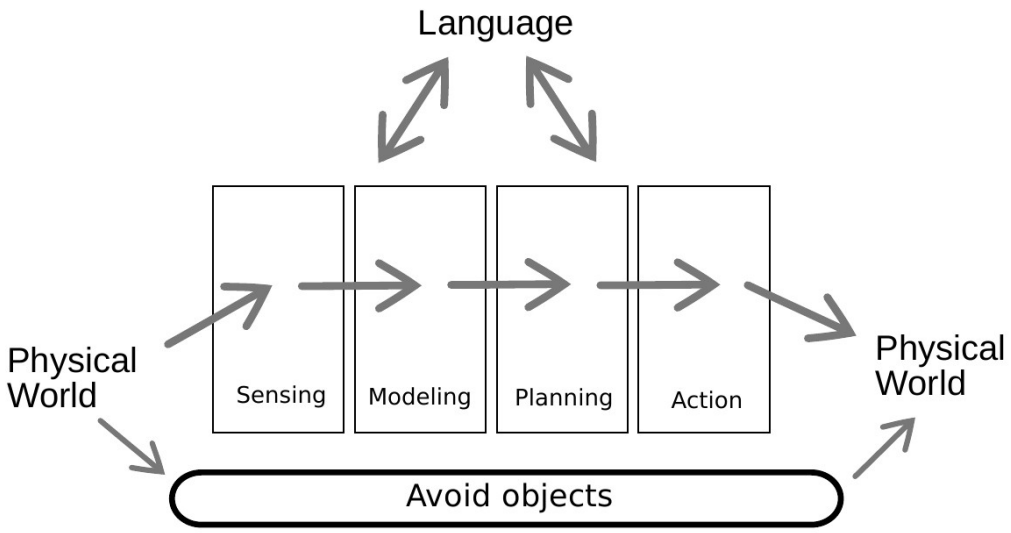

Figure 2. An SMPA hybrid model connects language use to a model of the physical world. Responsive behaviours can be run at a higher priority independently of the SMPA modules, enabling responsiveness but remaining separate from the main system.

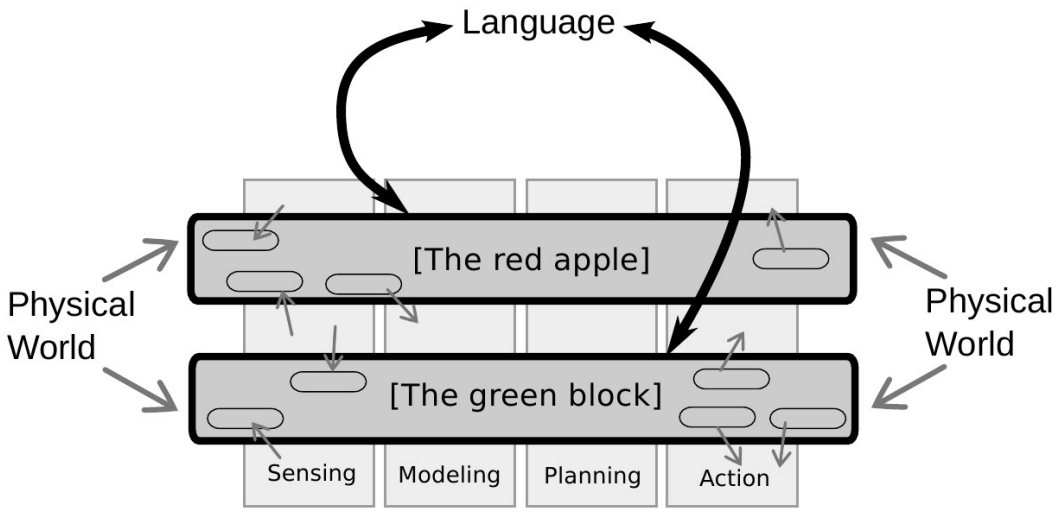

Figure 3. In our model, responsive processes (small ellipses, with small arrows depicting information passed in and out) are organized into object schemas (large boxes). The processes handle the various functions needed to maintain the object schema and connect to language.

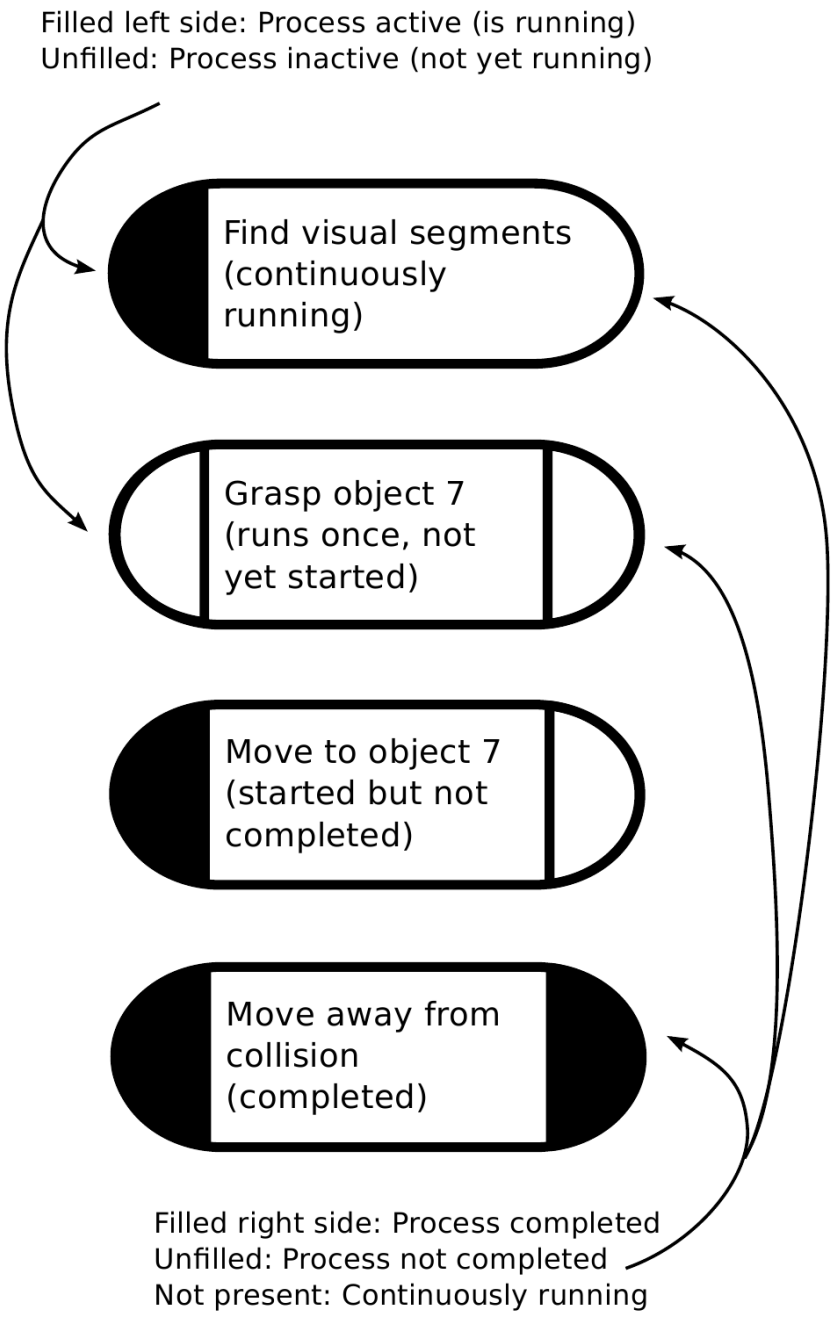

Figure 4. Visual notation for four example interaction processes with various run states and finish states.

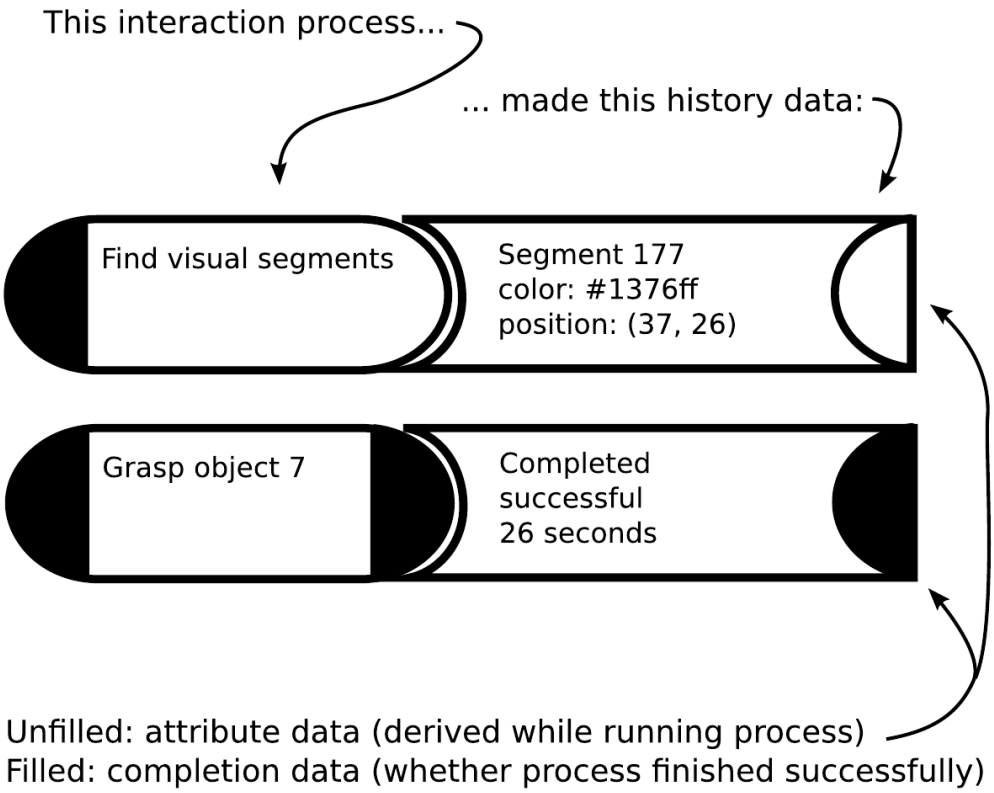

Figure 5. Visual notation for interaction history data generated by interaction processes. For simplicity, not all history data will be shown for processes depicted in such diagrams.

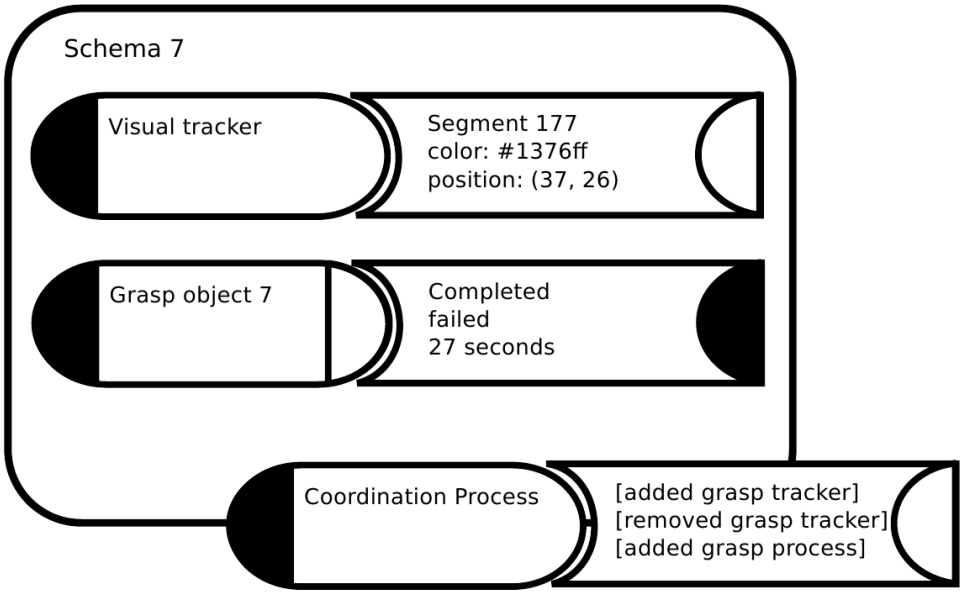

Figure 6. An object schema, containing three interaction processes relevant to its represented object. The reference process is depicted on the edge of the object schema to denote that it controls the object schema by including or removing other processes from the schema. For simplicity, not all processes in an object schema will be shown in these diagrams.

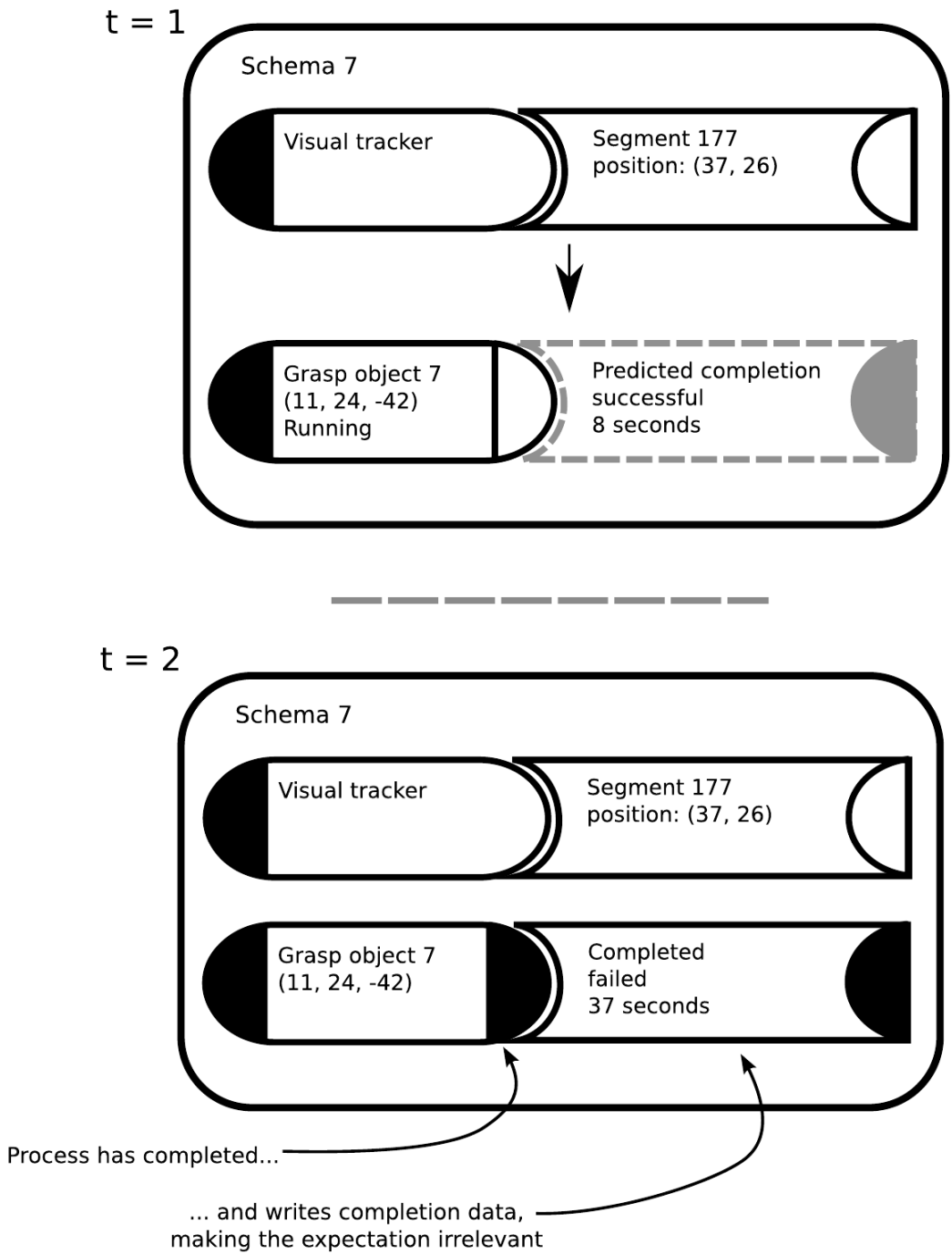

Figure 7. Top: The visual location associated with this example object schema leads to (denoted by the small arrow) an expectation that the corresponding grasp action will complete successfully. Bottom: The grasp action is completed and reports a failure.

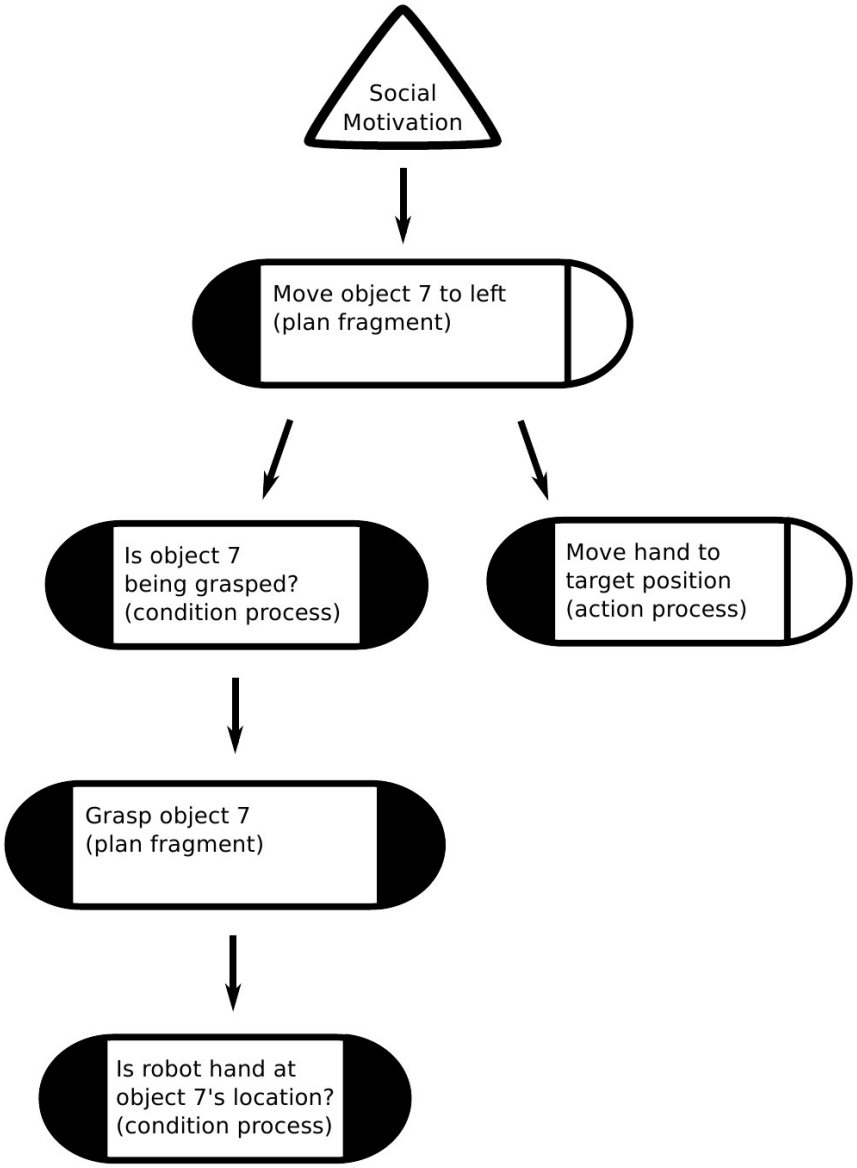

Figure 8. An example plan hierarchy, showing the primary "social" motivation connected to a tree of interaction processes. The arrows denote a parent-child relation, and indicate that the parent cannot be completed until the child completes. The tree consists of action processes, condition processes, and plan fragment processes.

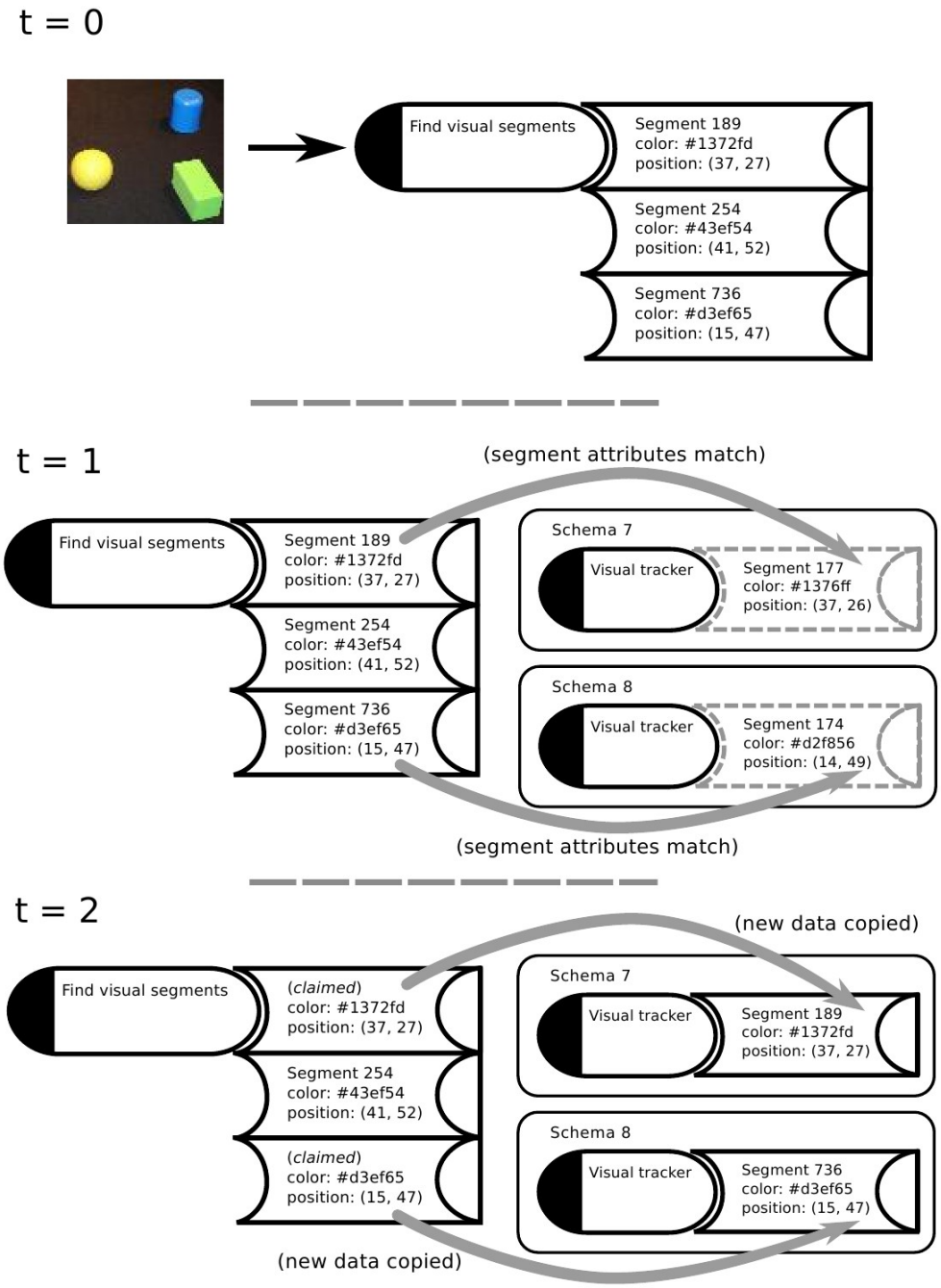

Figure 9. Top: The visual segmenter produces a list of coloured regions from a frame of visual input. Middle: Based on their histories, the visual trackers of two object schemas have expected attributes for how their visual regions will appear in the new frame. Bottom: The visual trackers have claimed the regions from the visual segmenter's interaction history, and copied the new visual data to their own histories.

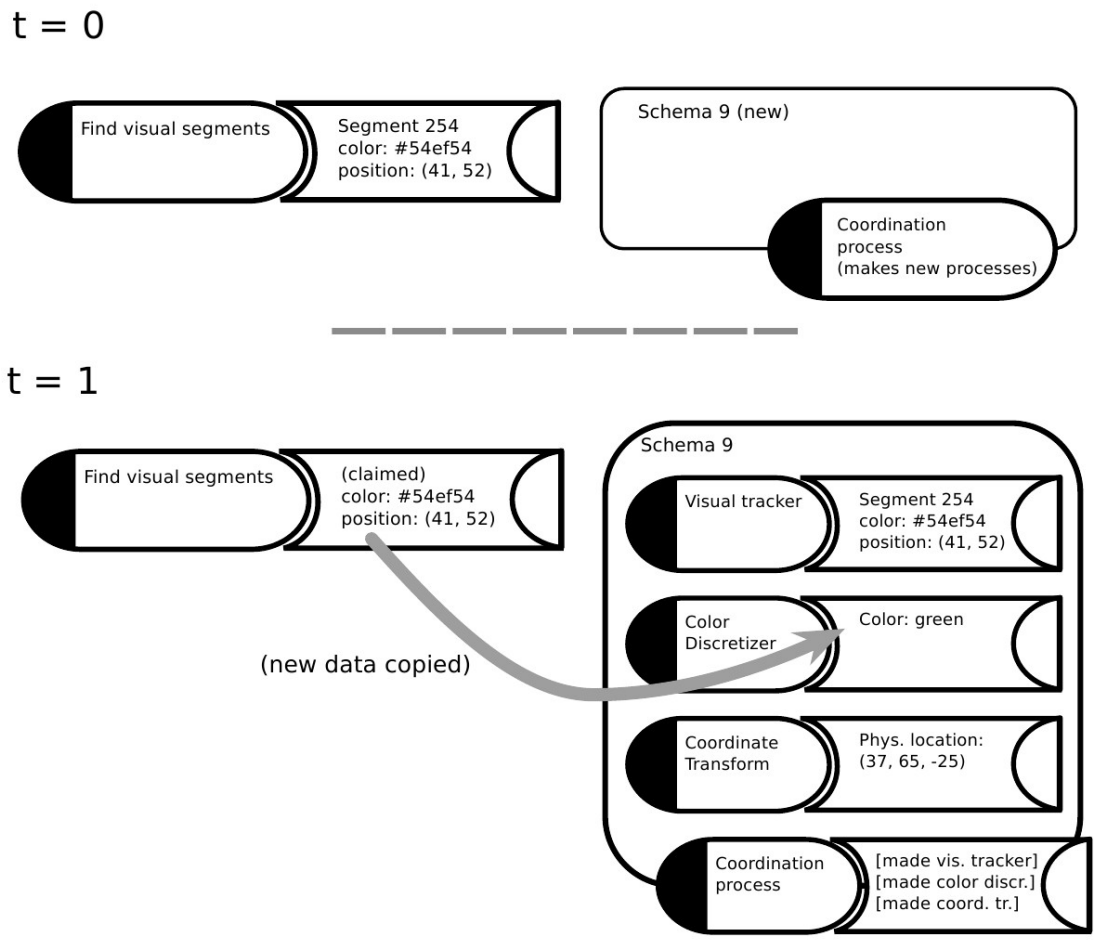

Figure 10. Top: The unclaimed visual segment from Figure 9 triggers the creation of a new object schema in the belief context. Bottom: The new schema's coordination process immediately creates trackers and translators which populate the history data for the object schema.

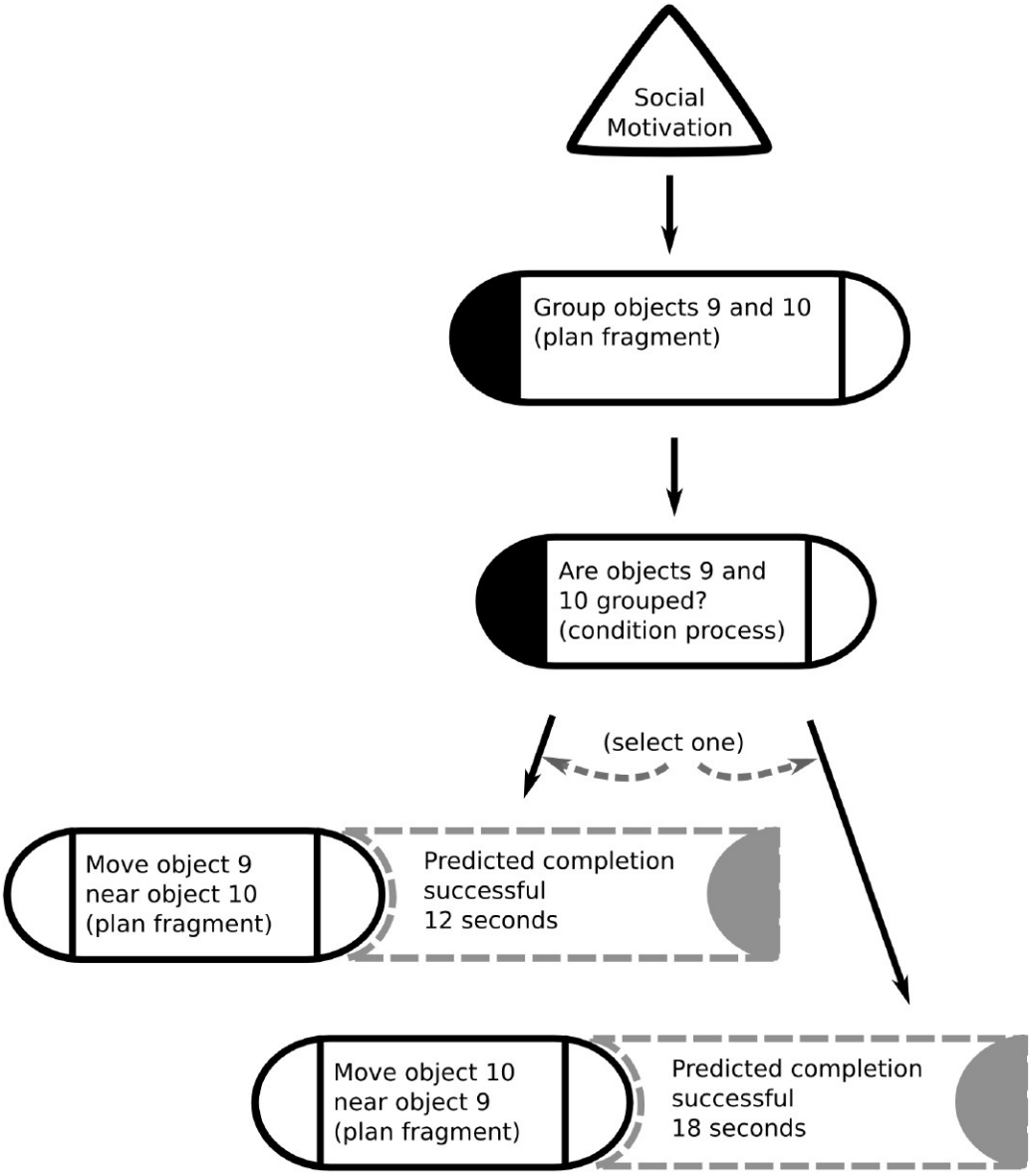

Figure 11. The planning system must choose between two plan fragments, both of which might satisfy the "grouped" condition process.

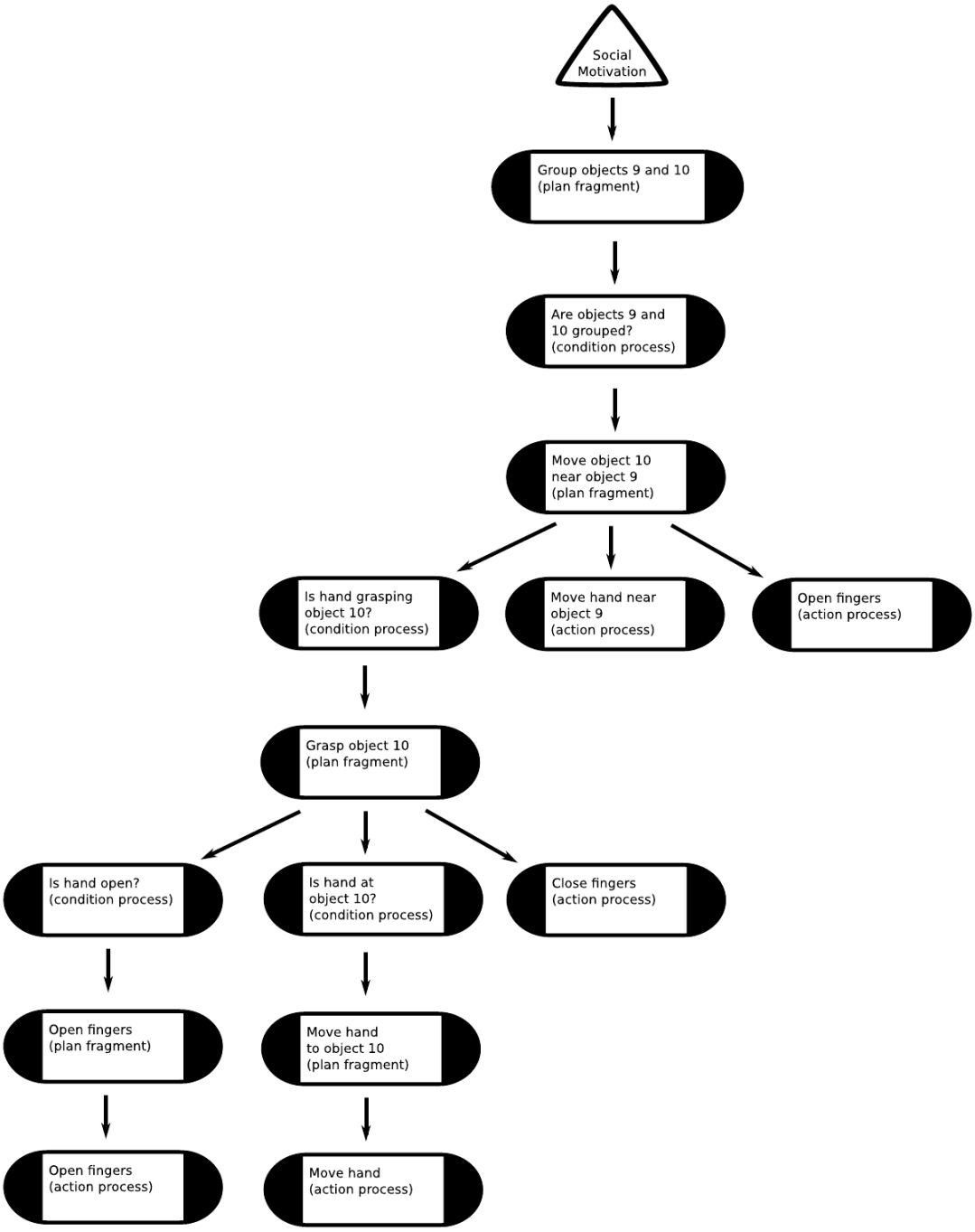

Figure 12. A completed plan hierarchy for grouping the green block (object 9) and the red apple (object 10). Action processes are typically deleted when they complete, but they are depicted here to show the entire structure. Temporal sequences for children of plan fragments are shown left-to-right.