Cognitive Systems Research 92 (2025) 101375

Contents lists available at ScienceDirect

Cognitive Systems Research

journal homepage: www.elsevier.com/locate/cogsys

Robot manipulation in everyday activities with the CRAM 2.0 cognitive architecture and generalized action plans☆

Michael Beetz[a], Gayane Kazhoyan[a], David Vernon[b],*

[a] *Institute for Artificial Intelligence, University of Bremen, Am Fallturm 1, Bremen, 28359, Germany*
[b] *Carnegie Mellon University Africa, Kigali, Rwanda*

ARTICLE INFO

Action editor: Serge Thill

Keywords:
Cognitive robotics
Cognitive architecture
Robot manipulation
Everyday activity

ABSTRACT

The CRAM 2.0 robot cognitive architecture provides a framework for knowledge-based instantiation of robot manipulation design patterns for everyday activities. These design patterns take the form of generalized action plans, which are transformed by CRAM 2.0 into parameterized low-level motion plans, using knowledge and reasoning with a contextual model to identify the motion parameter values that will successfully perform the actions required to accomplish the task. In this way, CRAM 2.0 performs implicit-to-explicit manipulation, mapping an under-specified high-level goal to the specific low-level motions required to accomplish the goal. We demonstrate the ability of a CRAM-controlled robot to carry out everyday activities in a kitchen environment.

1. Cognitive robotics and manipulation tasks in everyday activities

One of the main goals of cognitive robotics is for robot agents to be able to accomplish everyday manipulation tasks without having to be provided with detailed instructions, i.e., without users having to specify how the goal is to be achieved (Sandini, Sciutti, & Vernon, 2021). Consider the activity of setting a table for a meal and tidying up afterwards, shown in Fig. 1 and also in a video recording of this activity.[1] The robot fetches the required items and arranges them on the table. To complete the activity successfully, the robot has to select an appropriate way to carry out every object transportation task, depending on the type of the object to be transported (e.g., spoon, bowl, cereal box, milk box, or mug), its current and target location (e.g., drawer, refrigerator, cupboard, or table), and the task context (e.g., setting or cleaning the table, loading the dishwasher, or throwing away items). As the robot motions required for each action are not specified in the activity description, which is typically framed in goal-oriented terms, the robot must infer what it has to do. The CRAM[2] robot cognitive architecture[3] accomplishes this by adaptively instantiating design patterns for everyday activities. These design patterns take the form of generalized action plans, which are transformed by CRAM 2.0 into parameterized low-level motion plans, using knowledge and reasoning to identify the motion parameter values that will successfully perform the actions required to accomplish the task, complete the activity, and achieve the goal.

In general, carrying out everyday activities presents four fundamental challenges.

First, as we have noted above, everyday activities are typically stated in an underdetermined manner. For example, orders to “set the table”, “load the dishwasher”, and “prepare breakfast” do not fully specify the intended goal state, even if there are specific expectations about the outcome of the activity. Consequently, the robot must acquire the missing knowledge to accomplish the task and meet those expectations. Some of this knowledge is provided a priori, some from the context of the activity, and some can be learned by experience.

Second, competence in accomplishing everyday manipulation tasks requires the ability to make decisions based on knowledge, past experience, and prediction of the outcome of each constituent action. The knowledge required includes common sense, such as knowing that the tableware to be placed on the table should be clean and that clean tableware is typically stored in cupboards. It also requires intuitive

☆ This work was supported by the German Research Foundation DFG, Germany, as part of the Collaborative Research Center (Sonderforschungsbereich) 1320 “EASE — Everyday Activity Science and Engineering”, University of Bremen (http://www.ease-crc.org).

* Correspondence to: Carnegie Mellon University Africa, Kigali Special Economic Zone, Kigali, Rwanda.
E-mail addresses: mbeetz@uni-bremen.de (M. Beetz), kazhoyan@uni-bremen.de (G. Kazhoyan), vernon@cmu.edu (D. Vernon).

[1] https://www.ease-crc.org/link/video-ease-robot-day.

[2] CRAM is an acronym for Cognitive Robot Abstract Machine. The first version dates from 2010 (Beetz, Jain, Mösenlechner, & Tenorth, 2010). As we will see, CRAM 2.0 represents a significant advance on the concepts and functionality encapsulated in this first version, building on and extending the original principals.

[3] We qualify CRAM 2.0 as a *robot* cognitive architecture to distinguish it from cognitive architectures that also model human cognition; see Section 2.

https://doi.org/10.1016/j.cogsys.2025.101375
Received 17 April 2024; Received in revised form 3 June 2025; Accepted 6 June 2025
Available online 21 June 2025

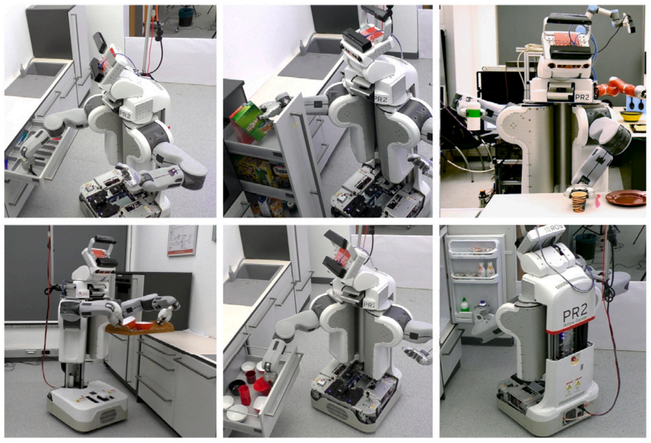


Fig. 1. Different object grasps selected by the context model based on the object, task, and context.

physics knowledge, e.g., that objects should be placed with their center of gravity close to the support surface to avoid them toppling over. Domain knowledge might include the fact that plates are breakable, so they must be handled with care. Experience allows the robot agent to improve the robustness and efficiency of its actions by tailoring behavior to specific contexts. Prediction enables the robot to infer the outcome of any action it performs.

Third, the cognitive robot must reason about its actions at the motion level, predicting the effect of motion parameter values on the physical effects of the motion. This allows a robot to identify the best way to achieve the intended outcomes and avoid unwanted side effects.

Fourth, a cognitive robot should be able to answer questions about what it is doing, why it is doing it, how it is doing it, what it expects to happen when it does it, and how it could do it differently. This ability is important because it allows the cognitive robot agent to assess the impact of not doing something effectively or failing to do it at all. This understanding also allows a cognitive robot to discover possibilities for improving the way it currently does things.

CRAM 2.0 addresses these four challenges of cognitive robot manipulation through a combination of novel representations and processes that distinguish it from both earlier versions of CRAM and other cognitive robot architectures. These mechanisms include (a) reasoning over structured representations of tasks, environments, robots, and actions, (b) perception- and effect-guided control of body motion, (c) experience-, prospection-, and competence-based decision making; and (d) introspective execution control. Together, these capabilities enable flexible, goal-directed behavior that adapts to underdetermined goals and complex task contexts in everyday human-scale environments. Core concepts such as *generalized action plans*, *designators*, *belief states*, and *digital twins* are introduced formally in Sections 3 and 4.

To interpret underdetermined task specifications, CRAM 2.0 employs *generalized action plans*: symbolic plan templatesor design patterns for task categories such as fetching, placing, or pouring. At execution time, these plans are instantiated by querying a semantic knowledge base (KnowRob 2.0) to infer missing parameters based on current perception and task-specific knowledge. This enables the system to convert vague, high-level instructions (e.g., “set the table”) into fully grounded motion behavior without requiring environment-specific programming.

To generate context-specific motion, CRAM combines symbolic designators, the Giskard constraint-based motion executive, a belief state maintained by the perception executive, and models of expected and desired world dynamics, including task goals, intended effects, and effects to be avoided. These components allow CRAM to infer and adapt motion parameters during execution, balancing hard constraints (e.g., collision avoidance, stability) and soft constraints (e.g., stereotypicality of motion patterns). Context is interpreted broadly, encompassing affordances, task semantics, learned priors, and predicted consequences, supporting motion generalization across robots and environments with minimal reconfiguration.

To make robust and context-aware decisions, CRAM 2.0 integrates simulation and experience-based reasoning. Its digital twin reasoning system (DTKR&R) simulates the outcomes of potential actions in a virtual model of the world, while narrative-enabled episodic memories (NEEMs) log and abstract prior experiences. This allows the robot to evaluate feasibility, anticipate physical consequences, and select strategies that are likely to succeed in the current situation.

In parallel, CRAM 2.0 supports introspective execution by maintaining explicit, perceptually grounded representations of tasks, objects, and actions, continuously updated throughout the task. These representations enable the robot to monitor its behavior, explain its choices, recover from errors, and adapt plans dynamically. In contrast to architectures that treat execution as a black-box process, CRAM 2.0 offers transparent, causally-grounded, introspective and explainable manipulation. These capabilities are demonstrated concretely in the table-setting scenario of Section 5, where a single plan library is used to generate task-appropriate actions across multiple robots and environments.

We now proceed to discuss in detail the capabilities outlined above, first by providing a brief overview of the field of cognitive architectures, second by describing the structure and operation of the CRAM 2.0 robot cognitive architecture, and third by explaining the CRAM 2.0 contextual model of robot agency and discussing the core elements of this model. We then provide examples of a CRAM-controlled robot carrying out manipulation tasks in everyday activities in a kitchen.

2. Cognitive architectures

The concept of a cognitive architecture, introduced by Newell (1990), is a framework that integrates all the elements required for a system to exhibit the abilities that are considered to be characteristic of a cognitive agent. Core cognitive abilities include perception, action, learning, adaptation, anticipation & prospection, motivation, autonomy, internal simulation, attention, action selection, memory, reasoning, and meta-reasoning (Kotseruba & Tsotsos, 2020; Vernon, von Hofsten, & Fadiga, 2016). A cognitive architecture determines the overall structure and organization of a cognitive system, including the component parts or modules (Sun, 2004), the relations between these modules, and the essential algorithmic and representational details within them (Langley, 2006).

There are three different types of cognitive architecture, each derived from the three paradigms of cognitive science: the cognitivist, the emergent, and the hybrid (Vernon, 2022).

Cognitivist cognitive architectures, often referred to as symbolic cognitive architectures (Kotseruba & Tsotsos, 2020), focus on the aspects of cognition that are relatively constant over time and that are independent of the task (Langley, Laird, & Rogers, 2009; Ritter & Young, 2001), with knowledge providing the task-specific element. In many symbolic systems, much of the knowledge incorporated in the model is provided by the designer, possibly drawing on years of experience working in the problem domain. Machine learning is increasingly used to augment this knowledge.

Emergent cognitive architectures focus on the development of the agent from a primitive state to a fully cognitive state over its life-time. As such, an emergent cognitive architecture is both the initial state from which an agent subsequently develops and the encapsulation of the dynamics that drive that development, typically exploiting non-symbolic processes and representations. Since the emergent paradigm holds that the body of the cognitive agent plays a causal role in the cognitive process, emergent cognitive architectures often reflect the

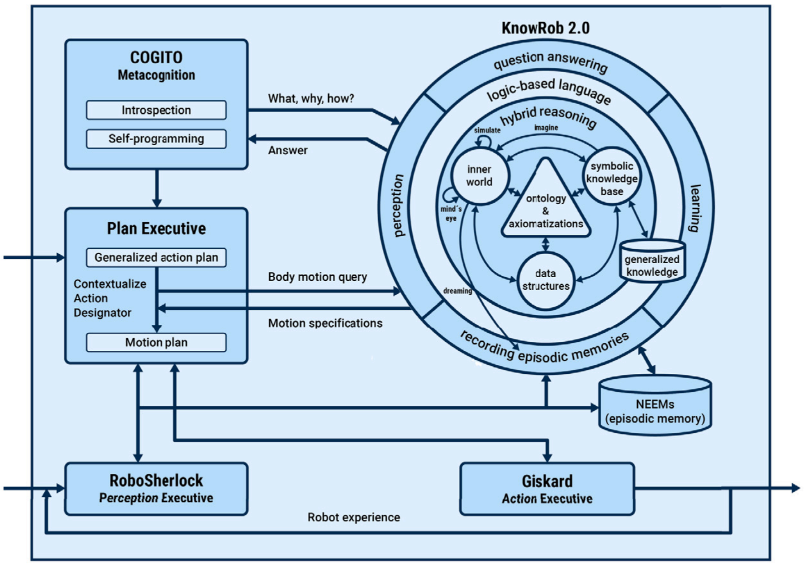


Fig. 2. A schematic representation of the CRAM 2.0 cognitive architecture with its five main components: the plan executive, the KnowRob 2.0 KR&R system, the metacognition system, the perception executive, and the action executive. In KnowRob 2.0, the NEEMs (episodic memory) knowledge base and generalized knowledge base together comprise the episodic memory knowledge base (see text for details).

structure and capabilities of the physical body and its morphological development (Naya-Varela, Faina, & Duro, 2021).

Hybrid systems endeavour to combine the strengths of the symbolic and emergent paradigms, typically integrating symbolic and non-symbolic processing. Hybrid cognitive architectures are the most prevalent type: forty-eight of the eighty-four cognitive architectures surveyed by Kotseruba and Tsotsos (2020) are hybrid.

Most cognitive architectures have their origins in cognitive science and focus on modeling human cognition, e.g., Soar (Laird, 2012; Laird, Newell, & Rosenbloom, 1987), ACT-R (Anderson, 1996; Anderson et al., 2004), CLARION (Sun, 2007, 2016, 2017), and LIDA (Franklin, Madl, D'Mello, & Snaider, 2014; Ramamurthy, Baars, D'Mello, & Franklin, 2006), all of which are classified as hybrid by Kotseruba and Tsotsos (2020). A few of these cognitive architectures have been applied in robotics, e.g., Soar and LIDA, and ACT-R/E (Trafton et al., 2013). However, other cognitive architectures have been designed specifically in the context of robotics research, e.g., ArmarX (Vahrenkamp, Wächter, Kröhnert, Welke, & Asfour, 2015) and ISAC (Kawamura, Gordon, Ratanaswasd, Erdemir, & Hall, 2008), and do not claim to be models of human cognition. The CRAM cognitive architecture, to be described in the next section, is one of these.

3. The CRAM 2.0 cognitive architecture

The CRAM 2.0 cognitive architecture[4] comprises five components: (i) the plan executive; (ii) a KR&R system referred to as KnowRob 2.0; (iii) a perception executive referred to as RoboSherlock; (iv) an action executive referred to as Giskard; and (v) a metacognition system referred to as COGITO; see Fig. 2. In the following, we will describe each component in turn.

3.1. The plan executive

The plan executive executes generalized action plans written in the CRAM plan language (CPL). It does so by determining what knowledge is required by the motion plan and formulating a query for the corresponding motion parameter values. The query is then answered using the contextual model encapsulated in KnowRob 2.0, using knowledge embedded in the plan, and using perceptual information provided by the perception executive RoboSherlock. The resultant motion plans are then executed by the action executive Giskard.

CPL[5] is an extension of Lisp, exploiting macros and functions to create a domain-specific language (DSL) for programming cognitive robots. It makes heavy use of multi-threading, especially for fluents, one of the key extensions. A fluent is a proxy object for some Common Lisp object, typically interfacing to some sensor, and it is used as a variable that allows a separate thread to monitor and act on a change in the value of that object, blocking until a value is available or until the value changes.

CPL supports concurrency, with various sequential execution constructs and parallel execution constructs, each offering different ways to advance the thread of execution. Failure in executing actions is a normal occurrence in robotics, so CPL allows failures to be thrown and caught using the `handle-failure` function. This function specifies the failure-handling code and provides control over the number of attempts to re-try the code that threw the failure, after first having taken some remedial action. It also has a fully-featured Prolog interpreter, written in Lisp, to provide for the definition of facts and predicates, and for reasoning, to be used when handling failures.

One of the main constructs in CPL has already been introduced: the designator. To the robot plan programmer, designators are objects containing sequences of key–value pairs of symbols, the second value element of each pair acting as a placeholder for information that is required by the CRAM plan, or a placeholder for the execution of a motor command. The information is acquired and the motor command executed by resolving the designator at run time in the current context of the task action. Information is acquired by querying a priori knowledge embedded in the plan, by querying knowledge in KnowRob 2.0, and by accessing sensor data through the perception executive, either directly or via a feature in KnowRob 2.0 referred to as a computable predicate, about which we will say more in Section 3.2.

As we will see in Section 4, there are four types of designator: motion designators (e.g., for motor commands), location designators (e.g., for 3D poses), object designators (e.g., for grasp configurations),

[4] The CRAM 2.0 open-source software is available at http://www.cram-system.org

[5] A detailed description of CPL in beyond the scope of this article; for a comprehensive treatment, see Mösenlechner (2016).

and action designators (e.g., for high-level goal-directed functionality); see Fig. 10. Each type of designator has a different resolution strategy. For example, motion designators are resolved by the action executive. Object designators provide a direct interface between CRAM plans and the RoboSherlock perception executive. The key–value pairs in the designator's properties describe the object that is to be perceived and RoboSherlock resolves the designator by extracting the required information from the sensor data. Location designators are typically resolved as robot poses that are appropriate for manipulating an object. Resolution is accomplished in two steps: (i) generation of a lazy list of candidate poses, and (ii) testing that candidate poses are feasible. This facilitates a general generation process and a specific filter process to remove the invalid solutions. Action designators are usually resolved using the Prolog inference engine to convert the high-level symbolic action into constitutent lower-level action designators, motion designators, location designators, and object designators, all of which are then resolved accordingly.

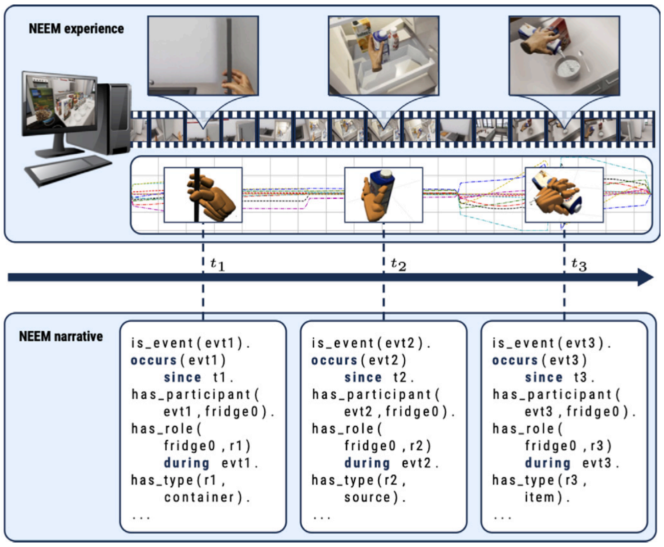
Because the plan executive uses explicit symbolic knowledge structures to represent the generalized action plans, the computational processes they initiate, the motions they generate, and the physical effects that the motions cause (Mösenlechner, Demmel, & Beetz, 2010), as well as the causal relationships between these knowledge structures, CRAM (specifically KnowRob 2.0, to be described next) can answer queries about what the robot does, why it does it, how it does it, and what happens when it does it. This allows CRAM to diagnose its behavior by inferring answers to questions such as: "Is the goal of the action achievable?" and "Did the action fail because the robot did not see the object?". These knowledge structures allow the robot to identify the subplans that are responsible for certain effects, and provides the basis for plan transformation (Kazhoyan, Niedzwiecki, & Beetz, 2020).

3.2. KnowRob 2.0: The knowledge representation & reasoning subsystem

KnowRob 2.0 (Beetz et al., 2018) is CRAM's KR&R framework. KnowRob 2.0 enables robots to acquire open-ended manipulation skills, reason about how to perform manipulation actions, and acquire commonsense knowledge. It has enabled robot agents to accomplish complex manipulation tasks such as making pizza, conducting chemical experiments, and setting tables.

KnowRob 2.0 is implemented in Prolog and it is exposed as a conventional first-order time interval logic knowledge base. However, many logic expressions are constructed on-demand from sensorimotor data computed in real-time, through algorithms for motion planning and inverse kinematics, for instance, and from log data stored in noSQL databases. It provides the background common sense intuitive-physics knowledge required by the plan executive to implement its goal-directed under-determined generalized action plans, e.g., how to grasp an object, depending on the object's shape, weight, softness, and other properties; how it has to be held while moving it, e.g., upright to avoid spilling its contents; and where the object is normally located. Some knowledge is specified *a priori*, some is derived from experience, and some is the result of simulated execution of candidate actions using a high-fidelity virtual reality physics engine simulator.

KnowRob 2.0 comprises five core elements embedded in a multi-formalism reasoning shell, exposed through a logic-based language layer to an interface shell that provides functionality for perception, question answering, experience acquisition, and knowledge learning; see Fig. 2. The five elements are: (i) a central set of knowledge ontologies and axiomatizations; (ii) an episodic memory knowledge base encapsulating the robot's experiences; (iii) an inner world knowledge base and virtual reality physics engine simulator; (iv) a symbolic knowledge base with abstracted symbolic seensorimotor data, logical axioms, and inference rules; and (v) a virtual knowledge base comprising a set of data-structures for parameterized motion control and path planning. We now describe each of these in more detail.

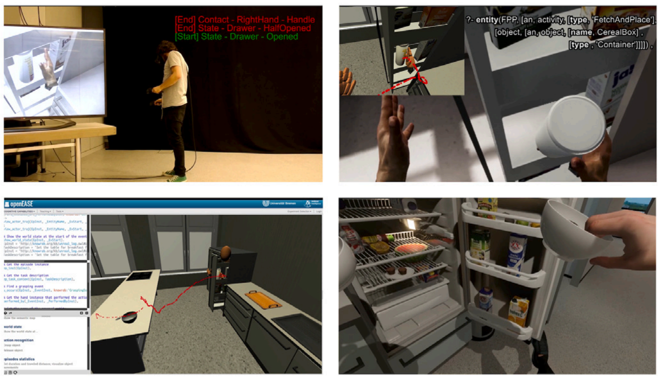


```
is_event(evt1).
occurs(evt1)
    since t1.
has_participant(
    evt1,fridge0).
has_role(
    fridge0,r1)
    during evt1.
has_type(r1,
    container).
...
```

```
is_event(evt2).
occurs(evt2)
    since t2.
has_participant(
    evt2,fridge0).
has_role(
    fridge0,r2)
    during evt2.
has_type(r2,
    source).
...
```

```
is_event(evt3).
occurs(evt3)
    since t3.
has_participant(
    evt3,fridge0).
has_role(
    fridge0,r3)
    during evt3.
has_type(r3,
    item).
...
```

Fig. 3. Schematic visualization of a NEEM: sub-symbolic experiential episodic data and motor control procedural data, augmented by a symbolic description of the episode as it unfolds.



```
[End] Contact - RightHand - Handle
[End] State - Drawer - HalfOpened
[Start] State - Drawer - Opened
```

```
?- entity(FPP, [an, activity, [type, 'FetchAndPlace'],
    [object, [an, object, [name, CerealBox] ,
        [type , 'Container']]]]) .
```

Fig. 4. Collecting NEEMs from human demonstration in a virtual kitchen environment.

3.2.1. Knowledge ontologies and axiomatizations

CRAM employs a collection of ontologies (Bateman, Beetz, Beßler, Bozcuoğlu, & Pomarlan, 2018) within a top-level ontology called SOMA (socio-physical model of activities). SOMA is a parsimonious extension of DUL (Dolce Upper Lite) ontology, where additional concepts and relations provide deeper semantics of autonomous activities, objects, agents, and environments. SOMA has been complemented with various sub-ontologies that provide background knowledge on everyday activities, including models of actions, robots, and affordances (Beßler et al., 2020), and failures modes (Diab et al., 2019).

3.2.2. Episodic memory knowledge base

One of the distinguishing aspects of KnowRob 2.0 is its focus on episodic memory, specifically narrative enabled episodic memories: NEEMs. These are CRAM's generalization of episodic memory (Tulving, 1972), which refers to a type of declarative memory that contains autobiographical events, encapsulating non-symbolic experiential episodic data, motor control procedural data, as well as a descriptive semantic annotation that explains what is happening in the NEEM experience; see Fig. 3. CRAM collects and stores NEEMs, and processes them to abstract away from specific episode contexts and learn the generally applicable commonsense and naive physics knowledge needed for mastering everyday activities.

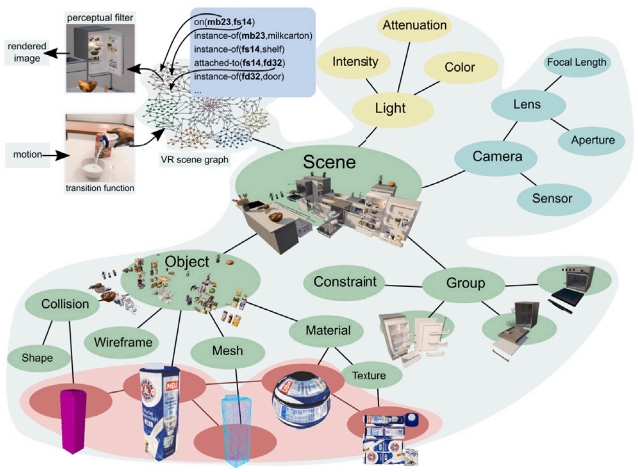


Fig. 5. Symbolic knowledge in KnowRob is grounded in a virtual reality scene graph representation (top left of the diagram) which, in turn, provides the basis for digital twin knowledge representation and reasoning (exploded view of part of the scene graph in the center, bottom, and right of the diagram).

KnowRob 2.0 provides a query language in order to retrieve information from NEEMs. The set of questions that can be asked about a given NEEM is provided by the KnowRob 2.0 ontology. One can retrieve all entities of a given entity category in the KnowRob 2.0 ontology and describe each entity using the attributes defined for the respective entity category. In addition, the relations defined in the ontology can be used to constrain combinations of entities. The assertions about entities and relations are automatically generated from the ontology. The integration of the NEEM database, referred to as the NEEM Hub, with the SOMA ontology yields a powerful hybrid symbolic/non-symbolic framework for reasoning about activities, and it forms the basis of the CRAM contextual model.

The creation of the CRAM contextual model was informed by interpretation and abstraction of human everyday activity data. These data include 100 multi-modal recordings of varied fetch and place table setting episodes. These episode recordings are transformed into a machine-understandable representation, semantically rooted in a common ontology of robot and human agency, and a common representation of activities.

The data encapsulated in NEEMS is generated by humans, both as a result of physical actions carried out by sensorized humans in the real world, and as a result of actions carried out by humans operating in a high-fidelity photorealistic virtual reality environment. The NEEMs were captured using AMEvA (Automated Models of Everyday Activities) (Haidu & Beetz, 2019), a computer system that can observe, interpret, and record fetch-and-place tasks and automatically abstract the observed activities into action models (Flanagan, Bowman, & Johansson, 2006). Fig. 4 shows the operation of AMEvA in which a human is performing `fetch&place` tasks in a virtual kitchen environment. NEEMs can also be used at execution time to diagnose and recover from execution failures (Bartels, Beßler, & Beetz, 2019).

3.2.3. Inner world knowledge base

The inner world knowledge base, also called a digital twin knowledge representation and reasoning system (DTKR&R), facilitates geometric reasoning using a photo-realistic virtual reality system and physics engine. The DTKR&R uses scene graph data structures that are the implementation basis of virtual environments and integrates them with the symbolic knowledge representation system, providing every entity that is relevant for the robot with a symbolic name; see Fig. 5. This symbolic name is then axiomatized in the symbolic knowledge base by asserting it as an instance of an object category defined in the ontological knowledge base, and providing formally-stated background knowledge about the entity. The relationship between the symbolic knowledge base and the inner world means that every relevant physical entity of the artificial world is formalized in the knowledge base and every symbolic physical entity is also an entity in the inner world. This means that the symbolic representations are grounded, not in physical objects, but in their counterparts in the inner digital twin world. However, this inner world is continually synchronized with the real world though perceptual data provided by the perception executive. In this way, the DTK&R inner world represents the belief state of a robot agent. Thus, to assess the information content of the belief state at a time instant t_i during an everyday activity episode, we can capture an image from the artificial world at time instant t_i and compare it with an image captured by a real camera in the real environment. Fig. 6 shows an example of an image showing the belief state generated by this process and the comparison with the real state of the environment at the same time instant.

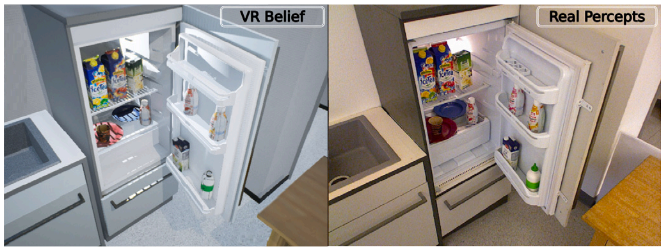


Fig. 6. Expectation for belief state estimation generated through imagistic reasoning. The belief image on the left is a rendering of the symbolic belief state of the robot agent.

Knowledge in the belief state can be retrieved by asking queries, such as: “which containers open counterclockwise?”, “which objects are electrical devices?”, or “what is a storage space for perishable items?”. The refrigerator is an answer to all of these queries. If the robot then asks for knowledge about the refrigerator, the accessible knowledge would include the part hierarchy of the refrigerator, including the 3D models of the parts, and the articulation model of the refrigerator that tells the robot how to open and close it.

Note that, in order to achieve this question answering capability, we make the *weak closed world assumption*; that is, in its reasoning processes, CRAM maintains a belief about where each object is and which state it has, but concurrently it monitors its percepts for new objects and updates its belief state whenever a new object is detected. We call the closed-world assumption *weak* because the robot is still required to detect novel objects, for example if somebody puts a new object in the environment. Domain knowledge can be provided as prior knowledge through a hand-coded ontology, which contains valuable manipulation knowledge, such as a container can be opened by generating a motion implied by the articulation model of the container (e.g., the knowledge that a screw top cap can be removed by twisting the cap). Assuming a weak closed world, CRAM-based robot agents can also answer queries that require prospective capabilities. An example of such a query is: “what do I expect the inside of the refrigerator to look like when I open it?” Answering this query requires the robot agent to visually render the scene inside the refrigerator given its current belief state.

Since the DTKR&R allows KnowRob 2.0 to simulate the outcome of candidate action and to establish the feasibility of that action, the inner world knowledge base serves two roles: (i) a representation of the belief state of the robot about itself and the world (as described above), and (ii) a reasoning mechanism for determining the outcome of candidate actions. As such, the inner world knowledge base encapsulates two types of knowledge: current beliefs about robot and the world, and projected internal simulation of future states. It also acts as a learning mechanism, generating NEEMs off-line when running simulations of activities with varying control parameters. These are recorded and transferred to the episodic memory knowledge base.

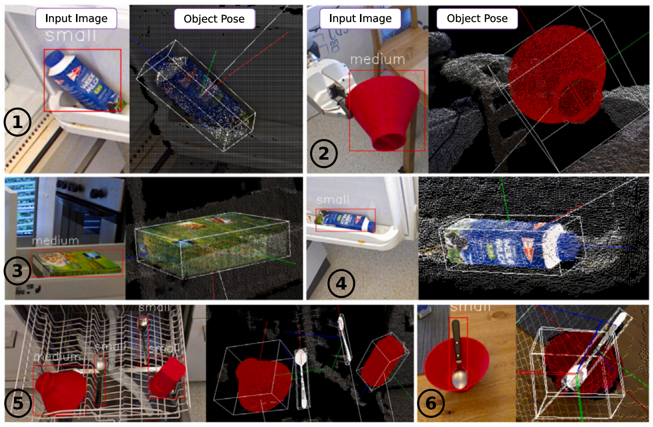


Fig. 7. Deep network-based object detection and object pose estimation in six challenging configurations.

3.2.4. *Symbolic knowledge base*

The symbolic knowledge base provides information about the entities in the robot's environment, including objects, object parts, object articulation models, environments composed of objects, actions, and events. It uses an entity description language that allows partial description of entities in terms of both symbolic and sub-symbolic properties.

3.2.5. *Virtual knowledge base of data structures*

The virtual knowledge base provides computable predicates that facilitate the integration of non-symbolic data, e.g., perceptual information, into the reasoning process, allowing symbolic queries of non-symbolic data. This allows run-time sensorimotor states to be integrated into the knowledge base at run-time and to be used in reasoning in the same way as symbolic knowledge.

3.3. *RoboSherlock: The perception executive*

The resolution of object designators, providing information about the objects in the robot's visual field of view, is accomplished by the RoboSherlock perception executive (Beetz et al., 2015). RoboSherlock provides a symbolic language in which perception tasks are stated in terms of object descriptions, object hypotheses, and task descriptions. Using these descriptions, CRAM can describe a red bottle using the object designator: `(an object (type bottle) (color red))` (see Fig. 10). The action designator `detecting` applied to an object designator causes the perception system to detect objects in the sensor data that satisfy the description and return the detected hypotheses (again, see Fig. 10).

RoboSherlock uses an extensible ensemble of experts to accomplish perception tasks. RoboSherlock perception experts are special-purpose routines that are employed in the respective perception contexts. Thus, rather than applying a general-purpose plate detector, RoboSherlock uses context-specific plate detectors. During table setting, it might use one that detects the topmost white horizontal lines in cupboards, while it uses detectors for ovals when cleaning the table. It might also use texture detection when it assumes the plates to not be clean, and so on. Thus, given a perception task and the current context information, RoboSherlock recruits the appropriate perception experts to generate candidate solutions for the given perception task and generates context-specific perception pipelines. The candidate solutions proposed by experts are then tested by simulation and ranked to find the best solution. The advantage of RoboSherlock over other perception approaches is that it can combine knowledge with perception and use knowledge-enabled reasoning about objects and scenes to tailor perception capabilities to the respective contexts and thereby make it more effective, robust, and efficient.

RoboSherlock exploits the power of RobotVQA (Kenfack, Siddiky, Balint-Benczedi, & Beetz, 2020), a scene-graph and deep-learning-based visual question answering system for robot manipulation. At the heart of RobotVQA lies a multi-task deep-learning model that infers formal semantic scene graphs from RGB(D) images of the scene at a rate of approximately 5 frames per second. The graph is made up of the set of scene objects, their description (category, color, shape, 6D pose, material, mask) and their spatial relations. Moreover, each of the facts in the graph is assigned a probability as a measure of uncertainty. The estimated scene graphs are represented using a probabilistic lightweight description logic.

Fig. 7 shows some of the current capabilities of RoboSherlock in more complex scenes, including a cluttered fridge, dishwasher, and oven.

RoboSherlock maintains the internal belief state implemented through virtual reality technologies; see Fig. 6. The knowledge base of the robot is populated with object models that consist of CAD models, including the part structure and possible articulation models, a texture model, as well as encyclopedic, commonsense, and intuitive physics knowledge about the object. This imagination-based scene perception approach has the advantage that the robot has perfect knowledge about everything that is contained in the belief state. A second advantage is that the robot can compute detailed and realistic image-level simulations about what it expects to see. These simulations are used to estimate object poses accurately and to save computational resources by rendering the belief state from the camera pose as an image and by comparing it with the image captured by the robot camera in the real environment.

As RoboSherlock is developed, it increasingly uses self-supervised learning methods leveraging these inner-world models of the robot environments (Kenfack et al., 2020; Mania & Beetz, 2019), which together with the images contained in the episodic memories (i.e., NEEMs) (Bálint-Benczédi & Beetz, 2018; Balint-Benczedi, Marton, Durner, & Beetz, 2017), are sufficient to learn robust real-world perception methods. This framework enables robot agents to use their environment and object representations in order to generate training data for supervised learning for perception tasks. For training, the framework not only allows the creation of typical scenes in the environment but also the generation of distributions for typical robot behaviors. This way, the distribution of training data can be tuned to specific kinds of environments and tasks.

3.4. *Giskard: The action executive*

Given a motion plan created by the plan executive, the Giskard Action Executive provides semantic constraint- and optimization-based motion control for manipulation actions. Giskard can be tasked with motion goals and objectives, such as "keep holding a door handle and move the handle according to the articulation model that the handle is part of". These two motion objectives are sufficient to open and close all containers of the kitchen furniture: the oven, the dishwasher, the refrigerator, the drawers, and the cupboards. Giskard can also execute these motion objectives for different robots; see Fig. 8.

As constraint- and optimization-based control is a mathematical optimization problem, Giskard transforms object-based action and motion specifications into mathematically formalized motion tasks. While the optimization-based motion generation can compute good body motions, it does so based on idealized and abstract models of the robot capabilities. In many situations the model assumptions are not satisfied. This happens, for example, if the robot moves different body parts at the same time and the composed motion causes inaccuracies in hand motions that are too large for grasping objects successfully, or the motion generation does not take into account the inaccuracies of the estimation of the robot pose. Such motion generation problems

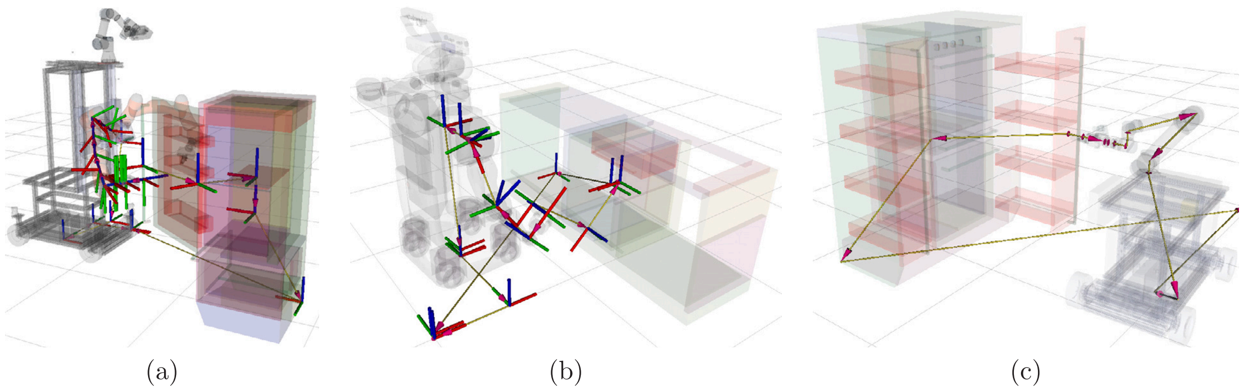


Fig. 8. Three different robots executing the action request “open the container holding the object to be placed on the table” for different objects located in different containers.

can often be better approached through experience-based learning, for example, by learning manipulation strategies using reinforcement learning. This has been investigated for learning hand manipulation strategies to open and close containers using tactile-based manipulation procedures which are linked to the declarative aspects of the developed tactile state detections (Meier, Haschke, & Ritter, 2020). Such processes include movement guidance, tactile servoing, tactile exploration with respect to shape or moveability, and different forms of task-related force and touch-based control, e.g., when unscrewing a lid or cutting a piece of bread.

3.5. COGITO: The metacognition system

Metacognition is an ability to reason about the performance of the system and adapt the cognitive architecture to improve its performance. In the CRAM 2.0 cognitive architecture, this maifests as an ability to reason about the generalized action plans, the plan interpreter, and the context model encapsulated in KnowRob 2.0, and then self-program to adapt its capabilities. This self-programming precondition of metacognition in the CRAM 2.0 cognitive architecture is satisfied because CPL is an extension of the Lisp programming language. There are two properties of the Lisp language that facilitate metacognitive capabilities: (i) programs as data, and (ii) the existence of metacircular interpreters. In Lisp, programs are represented as nested lists, that is as Lisp data structures. This means that Lisp programs can inspect and modify themselves. In other words, CPL plans can in principle inspect and modify CPL plans. The concept of metacircular interpretation is that an interpreter for a programming language can be implemented in the language itself. This metacircular interpretation process can be used to make the interpretation of a robot control program explicit and represent it for introspection and meta-reasoning. In other words, the CPL interpreter can inspect and modify itself.

We return to the discussion of metacognition in Section 7 where we outline planned extensions to the CRAM 2.0 cognitive architecture.

4. The CRAM contextual model of robot agency

4.1. Introduction

The CRAM stance on robot agency is that robot agents, in general, and cognitive robots, in particular, are controlled by programs: formally represented symbol structures that prescribe the computations for inferring the body motions to be executed by the robots. CRAM 2.0 represents these programs as plans that can be interpreted, reasoned about, and manipulated at execution time.[6] In this section, we describe the plan language in which key components of the program are expressed. These are *generalized action plans* and *motion plans*. We explain how motions are computed from underdetermined action descriptions in the form of generalized action plans, and we introduce the concept of a *contextual model* that captures how motions are implemented, i.e., how specific and appropriate motions are selected when interpreting the generalized action plan. We discuss the digital twin knowledge representation & reasoning (DTKR&R) system and introduce narrative-enabled episodic memories (NEEMs) which capture knowledge of specific robot experiences. In the next section, we describe the CRAM 2.0 cognitive architecture that realizes this model and the role that DTKR&R and NEEMs play in that realization.

4.2. Implicit-to-explicit manipulation

Our guiding principle is that actions performed by cognitive agents are goal-directed, and they are guided by prospection (Vernon, von Hofsten, & Fadiga, 2011). Because goals are typically stated in general, abstract terms, this means that cognitive robots carrying out everyday activities must be able to achieve what we term *implicit-to-explicit manipulation*, i.e., mapping an under-specified high-level goal to the specific low-level motions required to accomplish the goal. The likelihood of success in accomplishing the action is evaluated by internal simulation in the same domain as that in which the goal was formulated, i.e., the perceptual domain comprising observations of the world before starting the task and observations of the world after completing it, rather than at the symbolic level at which the action is stated.

This guiding principle is realized through an action description programming language — the CRAM Plan Language (CPL) — which allows manipulation actions to be carried out by saying *what* action is to be carried out, but without having to say exactly *how* it has to be carried out. We refer to this as an *underdetermined action description*: a high-level abstract specification of the action required to carry out an everyday task.

The key construct in CPL is the *generalized action plan*: a computational expression of the underdetermined action description. It is a high-level plan which is automatically expanded into a low-level parameterized motion plan, supplied with appropriate parameter values, and then executed. A generalized action plan is effectively a design pattern for a given type of action. The specification of the generalized action plan corresponds to the instantiation of the design pattern. A generalized action plan is also underdetermined: not all the knowledge required to execute the plan is specified. The required knowledge takes the form of the values of the parameters of the motion primitives into which the generalized action plan is expanded. These motion parameter values maximize the likelihood that the associated robot motions successfully accomplish the desired action. These values are

[6] Parts of the control program can also be effected using non-symbolic representations, including action policies implemented by training artificial neural networks.

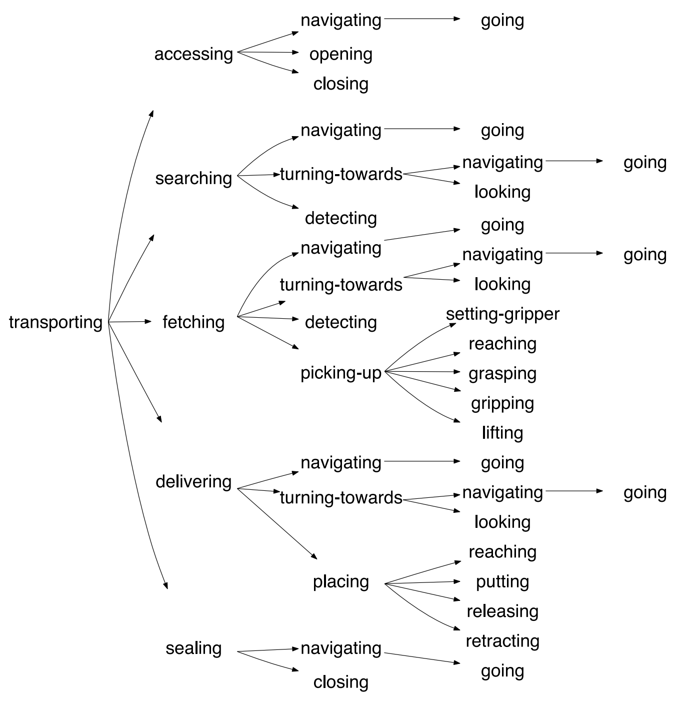
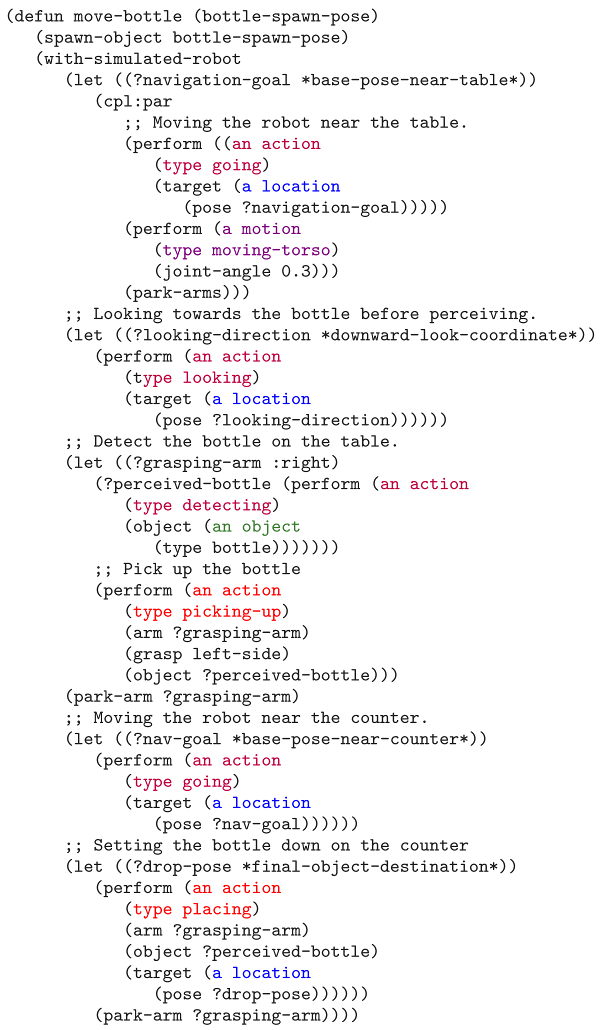


Fig. 9. The hierarchy of action designators for the PR2 robot. Note that the resolution of an action hierarchy at one level may cause the instantiation of more than one instance of a given action designator at the next level. Here, for the sake of compact presentation, we show just a single instance. Action designators that are leaf nodes in this tree are referred to as atomic action designators. These are subsequently resolved into motion designators.

provided by a *contextual model*, i.e., an ontology-dependent knowledge-base and reasoning system, to map from desired outcomes of an action to the motion parameter values that are most likely to succeed in accomplishing the desired action. Expansion of the generalized action plan and identification of motion parameter values is a process referred to as *contextualization*. CPL operates on an element of the generalized action plan called an *action designator*, a placeholder for yet-to-be-determined information. The designator is resolved and the related information is determined at run time based on the current context of the task action.

There are four types of designator in CPL: action, object, location, and motion designators. Action designators and motion designators are both concerned with robot control. Action designators focus on achieving some goal state, e.g., having set the table for a meal or having placed dirty dishes in the dishwasher, whereas motion designators are concerned with the physical movements and the control of some actuator, e.g., setting the gaze direction of the robot head, moving the end-effector to a given pose, or actuating the gripper. Location designators are concerned with poses in general, e.g., a list of positions and orientations at which the robot should stand in order to perform some manipulation task, or a list of positions where a robot should direct its gaze when looking for some specific object. Object designators are concerned with the properties of objects in the robot's environment, e.g., the pose of the object and its physical characteristics. Object designators provide an interface to the perception system since the pose of an object will typically be determined at run time.

There is a hierarchy of action designators; see Fig. 9. Consequently, the resolution of an action designator can involve the instantiation and resolution of other action designators. The action designators at the lowest level in the hierarchy are referred to as atomic action designators. Ultimately, all action designators are resolved into more concrete motion, location, and object designators. Specifically, atomic action designators are resolved directly into the motion designators that



```
(defun move-bottle (bottle-spawn-pose)
   (spawn-object bottle-spawn-pose)
   (with-simulated-robot
      (let ((?navigation-goal *base-pose-near-table*))
         (cpl:par
            ;; Moving the robot near the table.
            (perform ((an action
               (type going)
               (target (a location
                  (pose ?navigation-goal)))))
            (perform (a motion
               (type moving-torso)
               (joint-angle 0.3)))
            (park-arms)))
      ;; Looking towards the bottle before perceiving.
      (let ((?looking-direction *downward-look-coordinate*))
         (perform (an action
            (type looking)
            (target (a location
               (pose ?looking-direction))))))
      ;; Detect the bottle on the table.
      (let ((?grasping-arm :right)
         (?perceived-bottle (perform (an action
            (type detecting)
            (object (an object
               (type bottle)))))))
         ;; Pick up the bottle
         (perform (an action
            (type picking-up)
            (arm ?grasping-arm)
            (grasp left-side)
            (object ?perceived-bottle)))
      (park-arm ?grasping-arm)
      ;; Moving the robot near the counter.
      (let ((?nav-goal *base-pose-near-counter*))
         (perform (an action
            (type going)
            (target (a location
               (pose ?nav-goal))))))
      ;; Setting the bottle down on the counter
      (let ((?drop-pose *final-object-destination*))
         (perform (an action
            (type placing)
            (arm ?grasping-arm)
            (object ?perceived-bottle)
            (target (a location
               (pose ?drop-pose))))))
         (park-arm ?grasping-arm))))
```

Fig. 10. A generalized action plan to fetch and place a bottle. High-level action designators are in red, low-level action designators in purple, location designators in blue, object designators in green, and motion designators in violet.

form the motion plan. Designator resolution is accomplished either by querying knowledge embedded in the plan, by querying knowledge in CRAM's knowledge base, or by accessing sensorimotor data through perception. Resolving a motion designator results in motion of the robot body.

Let us now look more closely at the four key elements of implicit-to-explicit manipulation: the high-level generalized action plan, the low-level motion plan, the process of contextualization, and the contextual model.

4.3. Generalized action plans

A CRAM-based robot agent is equipped with a generalized action plan for each action category, which typically corresponds to action verbs such as fetch, place, pour, and cut. A generalized action plan specifies an action schema, i.e., a design pattern that specifies how actions of this category can be executed. A generalized action plan is the CPL counterpart of a underdetermined action description and it is executed by resolving one or more high-level action designators in the generalized plan. In CPL, this resolution is accomplished with the `perform` function; see Fig. 10. An action designator has a specific `type`, specified by a key–value pair, e.g., `type picking_up`.

The generalized action plan in Fig. 10 is a design pattern for picking and placing any object (in this case a bottle) from its current location

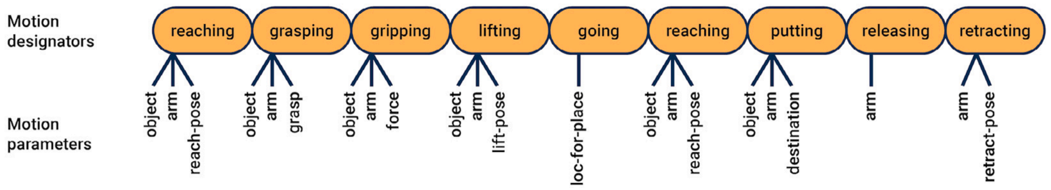


Fig. 11. Motion plan for the `fetch&place` generalized action plan, showing the motion designators that comprise each motion phase depicted in orange boxes, together with their associated motion parameters.

and placing it at some destination. CPL, in general, and a generalized action plan, in particular, lets the programmer state underdetermined action descriptions which have to be contextualized by the robot agent at execution time. For example, in the `picking-up` action designator, it is not specified where the object is located in the environment. These and other parameter values are specified as key–value pairs, e.g., `arm <value>`, `grasp <value>`, `object <value>`, `target <value>`, `object <value>`, which have to be inferred from the robot's knowledge, and augmented at run time with perceptual information. Some of the values of the key–value pairs are provided as input by the generalized action plan parameter values, e.g., `?grasping_arm`, others take the form of a constituent designator, e.g., a lower-level `type action` designator such as `type going`, a `motion` designator such as `type moving-torso`, a `location` designator, or an `object` designator.

Resolving the `location` designator yields a pose, e.g., `?navigation-goal`, `?looking-direction`, `?nav_goal`, and `?drop-pose`. Resolving the `object` designator by `performing` an `action` of `type detecting` yields information about the object, e.g., its pose and physical properties.

The underdetermined nature of the plan means that if the location of the object is not known then the robot has to search for it. The search process will be more efficient if the robot knows places where the object is likely to be. The robot can infer this knowledge from its knowledge base. Similarly, the manner in which the object has to be picked up can only be decided after the object is found and its geometry and state as well as the scene context are known.

4.4. Motion plans

The resolution of high-level action designators generates motion schemata that tell the plan executive how to execute the constituent motions. The motion schemata are represented as motion plans that structure the motion into motion phases. An approach similar to that described by Flanagan et al. (2006) has been adopted, but with a finer-grained representation of motion primitives. For example, the motion plan for picking up a detected object within the robot's reach and placing it at another location is shown in Fig. 11.

A motion plan for fetch & place actions comprises four different phases: reaching, lifting, transporting, and releasing (Flanagan et al., 2006). Each motion phase has a goal. When the goal is achieved, the start of the subsequent motion phase is triggered. Goals can be force-dynamic events, e.g., the robot finger coming into contact with the object to be grasped, or a perceptually distinctive event, e.g., a milk carton becoming visible when a fridge door is opened. Each motion phase has several parameters that are set in order to match the motion to the current situation. The question that then arises is: how are the parameter values selected to achieve the desired outcome and avoid unwanted side effects? We answer this question in the next section.

4.5. Contextualization

Mapping from a generalized action plan to a motion plan with parameter values is accomplished by the contextualization process. It has three steps, as follows.

1. Select the generalized action plan[7] that corresponds to the underdetermined description of the action that is to be performed, and insert the arguments required for that specific action to be performed. As we have seen, one of these arguments is the action type, e.g., `type picking_up`. Others include the type of the object to be manipulated and the destination location. These arguments are typically designators of some kind, i.e., an action, object, or location designator.
2. Expand the instantiated generalized action plan by adding the parameters needed to execute the motion plan, e.g., the arm or grasp pose to use. These motion parameters are identified automatically in CRAM by resolving the high-level action designator into the motion plan's constituent motion designators, by way of the action designator hierarchy; see Fig. 9.
3. Submit a query to the CRAM knowledge base for the values of the motion parameters that will produce the robot movements required to achieve the goal.

4.6. Contextual model

The query in the third step is answered by a knowledge representation & reasoning (KR&R) system. The KR&R system plays the role of the contextual model, exploiting the constraints of a priori contextual knowledge and current perceptual information. It uses prospection, effected with the CRAM's digital twin knowledge representation and reasoning (DTKR&R) system, to simulate plan execution, and thereby verify that the selected motion parameter values produce robot body motions that are likely to succeed in accomplishing the desired action.

Once the parameter values have been determined, and the query has been answered by the KR&R system, the motion plan is executed adaptively by the CRAM action executive.

The contextualization of an underdetermined action description, expressed as a generalized action plan, into well-defined body motions requires KR&R methods that can combine both abstract and concrete reasoning. Abstract reasoning enables open question answering by composing and chaining generalized and modular knowledge pieces. For example, through reasoning, robots can combine the knowledge that open filled containers should be held upright to avoid spilling with the knowledge that a milk carton is a container filled with milk and the knowledge that the milk carton is open, to conclude that the milk carton should be held upright when carried to the table. The power of generalized abstract knowledge is that it applies to all containers, even the ones that the robot does not know yet. At the same time, the robot needs to make concrete inferences to decide on the body motion. It has to pick the hand shape for picking up the milk carton, the grasp points, and the grasp force. This requires it to reason about the shape of the milk carton, its parts, its stability, the surface friction, the articulation model for opening and closing the milk carton, etc. In addition, contextualizing underdetermined action descriptions often requires predictive inferences, e.g., to assess the risk of the object falling out of the hand when grasping it in a particular way, taking into

[7] The selection of the generalized action plan is done by the CPL programmer.

consideration whether the object has to be re-grasped before placing it at the intended location, or for assessing alternative body motions for accomplishing the action. To cover this range of inference tasks, CRAM employs a digital twin knowledge representation and reasoning (DTKR&R) system which we described in Section 3.2.

5. Demonstration of the CRAM-controlled robot carrying out everyday activities

The current abilities of CRAM 2.0 to accomplish everyday activities has been demonstrated in an extensive validation exercise requiring a robot agent to set a table for a meal and clear up afterwards. The operation of the robot when carrying out these activities is captured on video (see Footnote 1). The approach we have adopted exploits the `fetch&place` generalized action plan. As we have noted, this plan can be executed with different types of robots and it can be used with various objects and in different environments. The plan is automatically contextualized for each individual object transportation task. Thus, the robot infers the body motions required to achieve the respective object transportation task depending on the type and state of the object to be transported (e.g., a spoon, bowl, cereal box, milk box, or mug), the original location (e.g., the drawer, the high drawer, or the table), and the task context (e.g., setting or cleaning the table, loading the dishwasher, or throwing away items) (Kazhoyan, Stelter, Kenfack, Koralewski, & Beetz, 2021). In doing this, it avoids unwanted side effects (e.g., knocking over a glass when placing a spoon on the table). The body motions generated to perform the actions are varied and complex and, when required, include subactions such as opening and closing containers, as well as coordinated, bimanual manipulation tasks. Fig. 1 shows some examples of the robot grasping objects in different ways during this exercise. All of these grasp strategies are inferred from the robot's knowledge base: the spoon in the drawer is grasped from the top because it is a very flat object; the tray is grasped with two hands because the center of mass would be too far outside the hand for a single-hand grasp; the mug is grasped from the side and not at the rim because the purpose of grasping it is pouring liquid into the mug.

6. Evaluation: Addressing the four challenges

We have evaluated CRAM 2.0 with respect to the four fundamental challenges of everyday cognitive manipulation that we have posed in the introduction: underdetermined task specification, context-specific behavior generation, knowledge-, experience-, and prediction-based decision making, and action-level explanation and introspection. Evaluations were conducted across a diverse set of real and simulated environments using both physical and virtual robot platforms. The results demonstrate that CRAM 2.0 generalizes across tasks, robots, and environments without requiring plan reengineering. Importantly, the simulated experiments are reproducible within the Virtual Research Building (VRB, 2025), which provides open-source code, openly accessible semantic knowledge bases, and interactive simulation environments running as Docker software containers. This infrastructure enables transparent, scalable, and repeatable experimentation across robot platforms and manipulation tasks.

Challenge 1: Underdetermined task specification. CRAM 2.0 demonstrates a robust ability to execute high-level, underdetermined task commands by transforming symbolic goal descriptions into fully grounded behavior at runtime. Through generalized action plans, abstract instructions such as "set the table" or "clean up" are resolved into context-appropriate low-level actions by querying semantic knowledge and resolving symbolic designators based on real-time perception. This dynamic plan instantiation enables the robot to autonomously infer the necessary objects, tools, locations, and motions required to complete a task, without requiring per-task procedural programming.

This capability was evaluated on four real robot platforms — PR2, Toyota HSR, Donbot, and Stretch — in three physical environments: the Apartment Lab, the Kitchen Lab, and the RoboCup@Home arena at the University of Bremen. Across these settings, the same generalized action plan was reused without modification to execute variations of object transportation tasks including fetching, placing, and clearing. In parallel, CRAM 2.0 was deployed on ten simulated robot agents in three virtual environments within the Object Transportation Virtual Research Lab (VRL1, 2025), confirming that execution-time plan instantiation generalizes across morphology, embodiment, and physical context. The manipulation task of getting an object out of a container is reproducible in The Body Motion Problem Virtual Research Lab (VRL2, 2025). All evaluations were run on publicly accessible infrastructure to ensure full reproducibility and benchmarking.

Challenge 2: Context-specific behavior generation. CRAM 2.0 produces context-sensitive motion behavior by resolving symbolic designators into concrete motion parameters based on a structured, dynamically updated belief state. This belief state encodes semantic, physical, and geometric properties of the environment, enabling the robot to interpret objects not only as spatial entities, but in terms of their task roles, physical properties, and affordances. Motion plans are computed and executed using Giskard, a constraint- and optimization-based whole-body controller that ensures generated trajectories satisfy all relevant constraints. These include constraints such as collision avoidance, kinematic feasibility, and correspondence with the Flanagan model of human reaching and placing. Importantly, context in CRAM 2.0 encompasses not just spatial configuration but also abstract factors such as task semantics, affordance relations, goal trade-offs, and experience-informed predictions, all of which contribute to the selection and generation of appropriate motions.

Again, this capability was validated across object transportation tasks in real and simulated experiments in the Object Transportation Virtual Research Lab (VRL1, 2025). The capabilities of the perception executive RoboKudo are reproducible in the RoboKudo Perception Executive Laboratory (EL1, 2025) and of the motion executive in GISKARD Motion Executive Laboratory (EL2, 2025). In real environments, the system adjusted motion plans for objects with different properties, small and big, rigid, fragile, and soft, single- and multi-part objects, and objects with articulation. All these adaptations were made without altering the high-level plan code: only the symbolic context and perceptual state differed. In simulation, the VRL infrastructure provided a reproducible testbed for evaluating motion constraint satisfaction and behavior generalization. Detailed access to the motion specification format allow external researchers to inspect how motion constraints were respected and how decisions were guided by symbolic and experiential context.

Challenge 3: Knowledge-, experience-, and prediction-based decision making. CRAM 2.0 tightly integrates symbolic reasoning, episodic experience, and internal simulation to support informed decision making. The robot anticipates and evaluates possible outcomes using its digital twin reasoning system (DTKR&R), while consulting narrative-enabled episodic memories (NEEMs) to reuse prior experience for parameter instantiation. This enables context-sensitive selection of action parameters based on likely success, history of similar situations, and inferred physical constraints, resulting in more robust and informed behavior.

Experience-based (Koralewski, Kazhoyan, & Beetz, 2019) and prediction-based (Kazhoyan & Beetz, 2019) decision making were shown to outperform the baseline performance of robot agents doing only knowledge-based decision making. Simulated outcomes were used to select successful configurations before execution, and NEEMs informed default choices based on prior trials. In the Virtual Research Building, mechanisms can be explored to equip the robot with task knowledge in the Knowledge Service Laboratory (KSL, 2025) and advanced methods for prediction-based decision making in the TraceBot Laboratory (TBL, 2025).

Challenge 4: Action-level explanation and introspection. CRAM 2.0 supports introspective reasoning by maintaining explicit symbolic representations of tasks, actions, parameters, and effects, and linking them to execution-level outcomes during task execution. These representations enable the robot to assess its ongoing behavior, detect inconsistencies between expected and observed states, and explain what it is doing and why. When failure occurs, CRAM 2.0 can diagnose causes (e.g., misperceived object pose, unreachable target) and trigger recovery procedures based on structured introspection and plan monitoring.

The action-level explanation and introspection capabilities of CRAM 2.0 are best validated through loading NEEMs of object transportation episodes into the interactive knowledge service. Using NEEM replay and visualization tools, researchers can explore the internal decision-making rationale and verify that failures are correctly attributed and recovered from, reinforcing the architecture's transparency and robustness. This can be done in the Knowledge Service Laboratory (KSL, 2025).

7. Discussion and future extensions

As demonstrated so well in the survey by Kotseruba and Tsotsos (Kotseruba & Tsotsos, 2020), the design and implementation of a cognitive architecture is an exercise that extends over many years, often several decades, and rarely comes to an end. The CRAM robot cognitive architecture is no different: the first version dates from 2010 (Beetz et al., 2010), with new and improved functionality being added over the intervening years, e.g., real-time perception, adaptive motion execution, elaborate failure handling, advanced reasoning, and extensions for different robot platforms, environments, and applications.

CRAM 2.0 differs from other cognitive robot architectures, some of which we mentioned in Section 2, by comprehensively reasoning and manipulating representations of models of robots, tasks, environments, and actions during execution. This is accomplished through tightly coupling symbolic reasoning, perception, motion control, and introspection within a physically grounded robotic system. To position CRAM 2.0 within the cognitive architecture landscape, we compare it to representative systems across our four core challenges of everyday manipulation: underdetermined task specification, context-specific behavior generation, experience- and prediction-based decision making, and action-level introspection, as follows (also, see Table 1).

CRAM 2.0 uniquely addresses underdetermined task specification through generalized action plans instantiated via contextual queries to a semantic knowledge base. This runtime grounding contrasts with rule-based systems like SOAR and ACT-R, which require fully specified procedures, and with robotic platforms like ArmarX (Vahrenkamp et al., 2015) that partially exhibit such cognitive flexibility at the behavior module level, and more recently through statistical motion sequence predictions using generative AI methods. For context-specific behavior generation, CRAM integrates a symbolic belief state with Giskard's constraint-based motion control, enabling the satisfaction of both hard and soft constraints. Other architectures either lack embodiment, e.g., SOAR & ACT-R, or do not propagate symbolic task constraints into motion control, e.g., DIARC (Schermerhorn et al., 2006; Scheutz et al., 2013).

In decision making, CRAM combines digital twin simulation (DTKR&R) and narrative-enabled episodic memories (NEEMs) to select context-appropriate actions based on prior experience and simulated outcomes. Most symbolic systems support neither prediction nor physical memory reuse. Finally, CRAM provides structured introspection by linking goals, plans, parameters, and execution effects, enabling real-time explanation and recovery. In contrast, systems like SOAR and DIARC offer limited introspective capacity, often disconnected from physical execution. CRAM 2.0 thus uniquely supports transparent, embodied reasoning in real-world manipulation.

The long term vision is for CRAM to exploit transformational learning (Kazhoyan et al., 2020) in two complementary ways: by specialization and by generalization. Both approaches are discussed in the remainder of this section. While they represent future extensions to the CRAM cognitive architecture, they are solidly based on the computational foundations described above.

Beneath the familiarity of everyday activities often lies a complexity that can be computationally intractable, especially when you factor in the flexibility that humans exhibit when carrying out these activities, the variety of circumstances in which they carry them out, and the underdetermined manner in which they are described (Vernon et al., 2021). This complexity is characterized by the fact that the mapping encapsulated in the CRAM contextual model is embedded in a very high-dimensional space, i.e., mapping from an under-specified high-level goal to the specific low-level motion parameter values required to accomplish the action successfully. One of the central ideas of the CRAM cognitive architecture is that, for everyday activities, the contextual model does not need to capture all the dimensions of this space: subsets of these dimensions are often sufficient to accomplish the actions successfully. These subsets are manifolds, which we refer to as PEAMs — pragmatic everyday activity manifolds — and they serve to render tractable the solution of problems that in their full generality are intractable. A PEAM, therefore, represents a form of specialization. It is achieved using the constraints that knowledge of everyday activities and the environment bring to bear on the problem. One of the future goals of CRAM is to identify and exploit the PEAMs that will result in a robot agent mastering everyday activities.

Generalization through metacognitive induction complements the PEAM solution strategy by exploring patterns among generalized actions plans, seeking ways to transform generalized action plans, either by carrying out the action in a more efficient and effective manner, or by accomplishing the outcome of the action in a different way. For example, instead of depending on a generalized action plan to carry dishes one by one to the dishwasher, a more general plan might first stack them if, as in the case of plates, they are stackable, then carry them together, and transfer them from the stack into the dishwasher. Alternatively, if they are not stackable, they might be placed on a tray, carried, and transferred to the dishwasher from the tray. The embedding of this form of induction and transformational learning in the CRAM metacognition system is also one of the main goals for the future.

Robot transformers, e.g., RT-1 (Brohan et al., 2022), RT-2 (Brohan et al., 2023), and RT-X (O'Neill, Rehman, Gupta, et al., 2024), foundation models for robotics (Firoozi et al., 2024; Hu et al., 2024; Xiao et al., 2023; Yang et al., 2023), e.g., RFM-1 (Sohn et al., 2024), including Vision-Language Models (VLMs) for robot manipulation (Stone et al., 2023) and Vision-Language-Action models (VLAs), e.g., OpenVLA (Kim et al., 2024), CogACT (Li et al., 2024), and GROOT N1 (Nvidia, 2025), all share with CRAM the capability to predict the motions a robot must perform to accomplish specific tasks within dynamically changing environments. These transformers and foundation models learn to anticipate the appropriate robot motions by learning probability distributions over robot motions, tasks, and contexts informed by various modalities such as images of robot manipulation scenes, instructional texts, videos of robot tasks, and firsthand manipulation task data. This distribution aims to foresee the next motion that will lead to successful task completion, based on the current task and context.

In contrast, CRAM approaches motion prediction by generating and refining an explicitly represented model of the intended action—the action designator—and inferring how it can be successfully carried out. CRAM's explicit action representation allows for an in-depth analysis of the planned action, albeit at the expense of requiring extensive knowledge bases and carefully designed representations. Meanwhile, robot transformers and foundation models learn to make decisions autonomously from a wealth of multimodal data. These data-driven systems are capable of handling a wide range of action variations, drawing from an extensive pool of experiences and training data. Consequently, a robot's performance is significantly influenced by the quality and quantity of action data used for training. Moreover, since motion

Table 1
Comparative positioning of CRAM 2.0 against other cognitive architectures across four manipulation challenges: ✓= fully addressed, ∼ = partially addressed, ×= not addressed. GAPS: Generalized Action Plans.

Challenge	CRAM 2.0	SOAR	ACT-R	DIARC	ArmarX
Underdetermined task specification	✓ (GAPs, Runtime grounding)	×	×	∼	∼
Context-Specific behavior generation	✓ (Belief, Constraints)	×	×	∼	✓ (Object-Action Complexes)
Prediction and Experience use	✓ (DTKR&R, NEEMs)	∼	∼	∼	✓ (ArmarX episodic memories)
Action explanation and Introspection	✓ (Symbolic, Causal, Embodied)	∼	∼	∼	∼

prediction in these systems is managed through a series of probabilistic inferences, there is a risk of generating impractical or ineffective motions, with potential inconsistencies in the motion sequence.

Robot transformers satisfy the first three core challenges of cognitive robot manipulation: they resolve underdetermined task specifications through multimodal learning, generate context-specific behavior based on joint probability distributions over tasks and environments, and incorporate prior experience through large-scale training. However, they do not yet support structured introspection and explanation of actions due to the absence of explicit task and world models. This lack of model-based reasoning limits their ability to explain what they are doing, why, and how alternatives could be considered. This observation suggests a promising research direction: integrating the probabilistic capabilities of robotic transformers with the model-based reasoning of CRAM 2.0. Drawing on dual-process theory (Kahneman, 2003; Stanovich & West, 2000), this hybrid approach would deploy robot transformers for fast, intuitive action generation (System 1), and CRAM's symbolic mechanisms for deliberate, introspective reasoning (System 2), enabling both efficient execution and transparent decision-making in complex environments.[8]

8. Summary and conclusions

The CRAM cognitive architecture is founded on the concept of a contextual model and a conceptual framework for accomplishing everyday manipulation tasks through implicit-to-explicit manipulation. Both are based on the following three premises:

1. For each manipulation action category, such as `fetch`, `place`, `pour`, `cut`, `wipe`, we can provide a general motion plan schema with motion phases and phase-specific motion parameters that can generate a range of body motions to achieve the respective goal of the action in a large variety of contexts.
2. A request for performing actions can be represented by generalized action plans that are contextualized through knowledge, reasoning, and perception in order to infer the motion parameter values that generate a body motion to achieve the goal of the action description.
3. The knowledge needed to refine action descriptions can be represented as generalized and modular knowledge chunks that can be composed through a reasoning engine to derive appropriate motion parameterizations for novel tasks, situations, and contexts.

The CPL programmer selects the generalized action plan that corresponds to the underdetermined description of the action that is to be performed, and inserts the parameters required for that specific action to be performed. Leveraging explicitly-represented knowledge & reasoning, inner-world simulation-based prospection, and perception, CRAM then contextualizes this generalized action plan to identify the parameter values that will accomplish the desired action. The action executive then adaptively executes the body motions, as demonstrated in Section 5.

[8] Ron Sun recently suggested a similar approach when discussing the integration of large language models (LLMs) with his Clarion cognitive architecture (Sun, 2024).

In conclusion, CRAM is a practical realization of cognition-enabled robotics and its application in carrying out everyday activities. It provides both the principled foundations and the practical computational means to endow robots with a capacity for flexible, context-sensitive manipulation in everyday activities. It accomplishes this through a robot cognitive architecture that also provides the foundation for advances to generalize implicit-to-explicit manipulation to produce new outcomes and new actions in novel scenarios (Vernon et al., 2021).

CRediT authorship contribution statement

Michael Beetz: Writing – review & editing, Writing – original draft, Conceptualization. **Gayane Kazhoyan:** Writing – review & editing, Writing – original draft, Conceptualization. **David Vernon:** Writing – review & editing, Writing – original draft, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Michael Beetz reports financial support was provided by German Research Foundation DFG. David Vernon, corresponding author, serves as an associate editor of Cognitive Systems Research. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the many people who have contributed to the development of the CRAM cognitive architecture over the past several years. These include Ferenc Bálint-Benczédi, Georg Bartels, Daniel Beßler, Andrei Haidu, Franklin Kenghagho Kenfack, Sebastian Koralewski, Lorenz Mösenlechner, Daniel Nyga, Simon Stelter, and Moritz Tenorth.

Data availability

No data was used for the research described in the article.

References

Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist*, *51*, 355–365.
Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060.
Bálint-Benczédi, F., & Beetz, M. (2018). Variations on a theme:"it's a poor sort of memory that only works backwards". In *2018 IEEE/RSJ international conference on intelligent robots and systems* (pp. 8390–8396). IEEE.
Balint-Benczedi, F., Marton, Z.-C., Durner, M., & Beetz, M. (2017). Storing and retrieving perceptual episodic memories for long-term manipulation tasks. In *Proceedings of the 2017 IEEE international conference on advanced robotics* (pp. 25–31). Hong-Kong, China: IEEE, Finalist for Best Paper Award.
Bartels, G., Beßler, D., & Beetz, M. (2019). Episodic memories for safety-aware robots. *KI-Künstliche Intelligenz*, *33*(2), 123–130.
Bateman, J., Beetz, M., Beßler, D., Bozcuoğlu, A. K., & Pomarlan, M. (2018). Heterogeneous ontologies and hybrid reasoning for service robotics: The EASE framework. In A. Ollero, A. Sanfeliu, L. Montano, N. Lau, & C. Cardeira (Eds.), *ROBOT 2017: third iberian robotics conference* (pp. 417–428). Springer International Publishing.

Beetz, M., Balint-Benczedi, F., Blodow, N., Nyga, D., Wiedemeyer, T., & Marton, Z.-C. (2015). RoboSherlock: Unstructured information processing for robot perception. In *IEEE international conference on robotics and automation*. Seattle, Washington, USA: Best Service Robotics Paper Award.

Beetz, M., Beßler, D., Haidu, A., Pomarlan, M., Bozcuoglu, A. K., & Bartels, G. (2018). Knowrob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *IEEE international conference on robotics and automation, ICRA 2018* (pp. 512–519).

Beetz, M., Jain, D., Mösenlechner, L., & Tenorth, M. (2010). Towards performing everyday manipulation activities. *Robotics and Autonomous Systems*, *58*(9), 1085–1095.

Beßler, D., Porzel, R., Mihai, P., Beetz, M., Malaka, R., & Bateman, J. (2020). A formal model of affordances for flexible robotic task execution. In *Proc. of the 24th European conference on artificial intelligence*.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., et al. (2023). RT-2: Vision-language-action models transfer web knowledge to robotic control.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., et al. (2022). RT-1: Robotics transformer for real-world control at scale.

Diab, M., Pomarlan, M., Beßler, D., Akbari, A., Rosell, J., Bateman, J., et al. (2019). An ontology for failure interpretation in automated planning and execution. In *Fourth iberian robotics conference* (pp. 381–390). Springer.

EL1 (2025). Robokudo perception executive laboratory. https://vrb.ease-crc.org/explore-labs/robokudo-lab.

EL2 (2025). GISKARD motion executive laboratory. https://vrb.ease-crc.org/explore-labs/giskard-lab.

Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., et al. (2024). Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*.

Flanagan, J. R., Bowman, M. C., & Johansson, R. S. (2006). Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*, *16*(6), 650–659.

Franklin, S., Madl, T., D'Mello, S., & Snaider, J. (2014). LIDA: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, *6*(1), 19–41.

Haidu, A., & Beetz, M. (2019). Automated models of human everyday activity based on game and virtual reality technology. In *2019 international conference on robotics and automation* (pp. 2606–2612). IEEE.

Hu, Y., Xie, Q., Jain, V., Francis, J., Patrikar, J., Keetha, N., et al. (2024). Toward general-purpose robots via foundation models: A survey and meta-analysis. arXiv:2312.08782.

Kahneman, D. (2003). A perspective on judgement and choice — mapping bounded rationality. *American Psychologist*, *58*(9), 697–720.

Kawamura, K., Gordon, S. M., Ratanaswasd, P., Erdemir, E., & Hall, J. F. (2008). Implementation of cognitive control for a humanoid robot. *International Journal of Humanoid Robotics*, *5*(4), 547–586.

Kazhoyan, G., & Beetz, M. (2019). Executing underspecified actions in real world based on online projection. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5156–5163). http://dx.doi.org/10.1109/IROS40897.2019.8967867.

Kazhoyan, G., Niedzwiecki, A., & Beetz, M. (2020). Towards plan transformations for real-world mobile fetch and place. In *IEEE international conference on robotics and automation (ICRA)*. http://dx.doi.org/10.1109/ICRA40945.2020.9197446.

Kazhoyan, G., Stelter, S., Kenfack, F. K., Koralewski, S., & Beetz, M. (2021). The robot household marathon experiment. In *IEEE international conference on robotics and automation (ICRA)*. Retrieved from https://arxiv.org/abs/2011.09792.

Kenfack, F. K., Siddiky, F. A., Balint-Benczedi, F., & Beetz, M. (2020). RobotVQA — A scene-graph- and deep-learning-based visual question answering system for robot manipulation. In *IEEE/RSJ international conference on intelligent robots and systems*. Las Vegas, USA.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., et al. (2024). Openvla: An open-source vision-language-action model. arXiv:2406.09246.

Koralewski, S., Kazhoyan, G., & Beetz, M. (2019). Self-specialization of general robot plans based on experience. *IEEE Robotics and Automation Letters*, *4*(4), 3766–3773. http://dx.doi.org/10.1109/lra.2019.2928771.

Kotseruba, I., & Tsotsos, J. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*(1), 17–94.

KSL (2025). Knowledge service laboratory. https://vrb.ease-crc.org/explore-labs/openease-lab.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Laird, J. E., Newell, A., & Rosenbloom, P. (1987). Soar: an architecture for general intelligence. *Artificial Intelligence*, *33*(1–64).

Langley, P. (2006). Cognitive architectures and general intelligent systems. *AI Magazine*, *27*(2), 33–44.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*(2), 141–160.

Li, Q., Liang, Y., Wang, Z., Luo, L., Chen, X., Liao, M., et al. (2024). Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. arXiv:2411.19650.

Mania, P., & Beetz, M. (2019). A framework for self-training perceptual agents in simulated photorealistic environments. In *2019 international conference on robotics and automation* (pp. 4396–4402). IEEE.

Meier, M., Haschke, R., & Ritter, H. J. (2020). From geometries to contact graphs. *12397*, In *Artificial neural networks and machine learning - ICANN 2020 - 29th international conference on artificial neural networks, bratislava, slovakia, September 15-18, 2020, proceedings, part II* (pp. 546–555). Springer, http://dx.doi.org/10.1007/978-3-030-61616-8_44, Retrieved from https://doi.org/10.1007/978-3-030-61616-8_44 (Best Paper Award).

Mösenlechner, L. (2016). *The cognitive robot abstract machine: a framework for cognitive robotics* (Ph.D. thesis), Technical University of Munich.

Mösenlechner, L., Demmel, N., & Beetz, M. (2010). Becoming action-aware through reasoning about logged plan execution traces. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2231–2236). http://dx.doi.org/10.1109/IROS.2010.5650989.

Naya-Varela, M., Faina, A., & Duro, R. J. (2021). Morphological development in robotic learning: a survey. *IEEE Transactions on Cognitive and Developmental Systems*.

Newell, A. (1990). *Unified theories of cognition*. Cambridge MA: Harvard University Press.

Nvidia (2025). *GR00T N1: An open foundation model for generalist humanoid robots*: *Tech. Rep.*.

O'Neill, A., Rehman, A., Gupta, A., et al. (2024). Open X-Embodiment: robotic learning datasets and RT-x models. arXiv:2310.08864.

Ramamurthy, U., Baars, B., D'Mello, S. K., & Franklin, S. (2006). LIDA: A working model of cognition. In D. Fum, F. D. Missier, & A. Stocco (Eds.), *Proceedings of the 7th international conference on cognitive modeling* (pp. 244–249).

Ritter, F. E., & Young, R. M. (2001). Introduction to this special issue on using cognitive models to improve interface design. *International Journal of Human-Computer Studies*, *55*, 1–14.

Sandini, G., Sciutti, A., & Vernon, D. (2021). Cognitive robotics. In M. Ang, O. Khatib, & B. Siciliano (Eds.), *Encyclopedia of robotics*. Springer.

Schermerhorn, P., Kramer, J., Brick, T., Anderson, D., Dingler, A., & Scheutz, M. (2006). DIARC: A testbed for natural human-robot interactions. In *Proceedings of AAAI 2006 mobile robot workshop*.

Scheutz, M., Briggs, G., Cantrell, R., Krause, E., Williams, T., & Veale, R. (2013). Novel mechanisms for natural human-robot interactions in the diarc architecture. In *Workshop at the 27th AAAI conference on artificial intelligence*. North America.

Sohn, A., Nagabandi, A., Florensa, C., Adelberg, D., Wu, D., Farooq, H., et al. (2024). Introducing RFM-1: Giving robots human-like reasoning capabilities. https://covariant.ai/insights/introducing-rfm-1-giving-robots-human-like-reasoning-capabilities.

Stanovich, K. E., & West, R. F. (2000). Individual differences in reasoning: Implications for the rationality debate. *Behavioral and Brain Sciences*, *23*, 645–665.

Stone, A., Xiao, T., Lu, Y., Gopalakrishnan, K., Lee, K.-H., Vuong, Q., et al. (2023). Open-world object manipulation using pre-trained vision-language models.

Sun, R. (2004). Desiderata for cognitive architectures. *Philosophical Psychology*, *17*(3), 341–373.

Sun, R. (2007). The importance of cognitive architectures: an analysis based on CLARION. *Journal of Experimental & Theoretical Artificial Intelligence*, *19*(2), 159–193.

Sun, R. (2016). *Anatomy of the Mind: Exploring Psychological Mechanisms and Processes with the Clarion Cognitive Architecture*. Oxford University Press.

Sun, R. (2017). The CLARION cognitive architecture: Toward a comprehensive theory of the mind. In S. E. F. Chipman (Ed.), *The oxford handbook of cognitive science*. Oxford University Press.

Sun, R. (2024). Can a cognitive architecture fundamentally enhance LLMs? Or vice versa?. arXiv:2401.10444.

TBL (2025). TraceBot laboratory. https://vrb.ease-crc.org/explore-labs/the-tracebot-laboratory.

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, F. P., Khemlani, S. S., & Schultz, A. C. (2013). ACT-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, *2*(1), 30–55.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving, & W. Donaldson (Eds.), *Organization of memory* (pp. 381–403). New York: Academic Press.

Vahrenkamp, N., Wächter, M., Kröhnert, M., Welke, K., & Asfour, T. (2015). The robot software frameworks armarx. *Information Technology*, *57*(2), 99–111.

Vernon, D. (2022). Cognitive architectures. In A. Cangelosi, & M. Asada (Eds.), *Cognitive robotics* (pp. 191–212). MIT Press.

Vernon, D., Albert, J., Beetz, M., Chiou, S.-C., Ritter, H., & Schneider, W. X. (2021). Action selection and execution in everyday activities: A cognitive robotics & situation model perspective. *Topics in Cognitive Science*, 1–19.

Vernon, D., von Hofsten, C., & Fadiga, L. (2011). A roadmap for cognitive development in humanoid robots. *Cognitive systems monographs (COSMOS)*: *vol. 11*, Berlin: Springer.

Vernon, D., von Hofsten, C., & Fadiga, L. (2016). Desiderata for developmental cognitive architectures. *Biologically Inspired Cognitive Architectures*, *18*, 116–127.

VRB (2025). Virtual research building. https://vrb.ease-crc.org.

VRL1 (2025). Object transportation virtual research lab. https://vrb.ease-crc.org/explore-labs/object-transportation-lab.

VRL2 (2025). The body motion problem virtual research lab. https://vrb.ease-crc.org/explore-labs/body-motion-problem/.

Xiao, X., Liu, J., Wang, Z., Zhou, Y., Qi, Y., Cheng, Q., et al. (2023). Robot learning in the era of foundation models: A survey. arXiv:2311.14379.

Yang, S., Nachum, O., Du, Y., Wei, J., Abbeel, P., & Schuurmans, D. (2023). Foundation models for decision making: Problems, methods, and opportunities. arxiv:2303.04129v1.