

# A balancing act: Ordering algorithm and image-schematic action descriptors for stacking objects by household robots

Kaviya Dhanabalachandran<sup>1</sup>, Maria M. Hedblom<sup>2</sup> and Michael Beetz<sup>1</sup>

<sup>1</sup>*Institute of Artificial Intelligence, University of Bremen, Germany*

<sup>2</sup>*Jönköping Artificial Intelligence Laboratory, Jönköping University, Sweden*

## Abstract

Optimising object order in stacking problems remains a hard problem for cognitive robotics research. In this paper, we continue our work on using the spatiotemporal relationships called image schemas to represent affordance spaces founded on object properties. Based on object properties, we introduce a stacking-order algorithm and describe the action descriptors using an image-schematic event segmentation format by describing a small subset using the Image Schema Logic ISL<sup>FOL</sup>.

## Keywords

object stacking, cognitive robotics, affordances, algorithms, action representation

## 1. Introduction and Problem Space

Despite the past decades progress in developing increasingly sophisticated software and hardware, research in artificial intelligence and cognitive robotics is still struggling to accurately represent and execute tasks that are learned by a human children in early infancy, a phenomenon known as Moravec's paradox [1]. One such task is the pick-and-place task. Appearing deceptively simple, the task includes visual mastery of gaze control, understand spatial depth and positioning, as well as object recognition and object permanence. It requires object manipulation abilities for moving the agent's own body (or body parts), understand the spatial and force-dynamic relationships of grasping objects. Additionally, it needs to be able to identify object properties and reason how their affordances behave under particular conditions. In this paper, we look at a particular complex pick-and-place task by focusing on how to stack object on the vertical axis. In addition, to the complexities of the pick-and-place task, stacking objects require a deeper understanding of the properties and affordances of the objects being stacked. For instance, it is not possible to stack heavy objects on top of flexible objects, nor it is (under normal circumstances) possible to stack a flat object onto a convex surface.

The motivation for this research agenda is the common occurrence of stacking objects in

---


*The Eighth Joint Ontology Workshops (JOWO'22), August 15-19, 2022, Jönköping University, Sweden*

✉ kaviya@uni-bremen.de (K. Dhanabalachandran); maria.hedblom@ju.se (M. M. Hedblom); beetz@uni-bremen.de (M. Beetz)

🆔 0000-0002-0419-5242 (K. Dhanabalachandran); 00070-0001-8308-8906 (M. M. Hedblom); 0000-0002-7888-7444 (M. Beetz)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

household activities. Kitchen utensils are stacked on top of one another in the cupboard and objects are carried on trays in particular arrangement.

To provide the household robots with a more intelligent understanding for stacking, we will use inspiration from cognitive science that demonstrate how human children core down object properties and action segments into meaningful components. We will then proceed to utilise these components by formally representing action descriptors involved in stacking and present an ordering algorithm that takes the object properties into account to provide a stable stack.

## 2. Foundation and Related Work

### 2.1. Theoretical Foundation

Our work is based on the hypothesis that there exists a finite number of conceptual building blocks that represent the salient features of object relations, including those providing balance and stackability. These are often referred to as *image schemas* and encompass spatiotemporal relationships between objects, agents and their environments<sup>1</sup> [2, 3]. The theory stems from the notion of embodied cognition, a common theoretical framework in cognitive robotics research (e.g. [4, 5]) which assumes that intelligent behaviour, in all its forms, stems from the body's sensorimotor experiences in its environment. Within this framework the image schemas represent the atomic yet salient features that distinguish one particular situation from another. For instance, the image schema CONTACT is present in a situation in which two objects are physically touching and the image schema LINK defines objects that (may or may not be) touching but have a causal relationship to one another. The salient difference is that in CONTACT, moving one object will not necessarily affect the other object, but if they are LINKED, moving one object will automatically move the other object as well. Likewise, the salient features of an object like a *cup* is defined by its ability to CONTAIN liquids, and objects with flat sturdy surfaces like a *tray* are defined by their affordance to SUPPORT other objects. In most cases, an event like stacking objects can be described as a combination of different vertical pick-and-place tasks (VERTICALITY + SOURCE\_PATH\_GOAL) with the SUPPORT and CONTAINMENT constraints of any involved objects.

Due to the conceptually-rich content of image schemas and their finite number, combinations of them can be argued to describe all kinds of spatiotemporal situations and events [6, 7]. For robotic research, this means that it is possible to formally represent the physical states of both the initial state and the goal state of particular actions, but also in detail describe the changes over time that constitute the actions that leads to these changes. In previous work we have investigated this for different scenarios (see [8, 9]) and in Section 4 we will demonstrate this for stacking using ISL<sup>FOL</sup>.

---

<sup>1</sup>The theory was originally developed in cognitive linguistics to explain the high number of spatial metaphors in abstract language, but has become a common hypothesis in many research areas dealing with semantic relevance in relation to spatial reasoning. Not all research fields would agree on defining them as spatiotemporal relationships, but it is a useful delimitation in cognitive robotics.

## 2.2. Formal Framework

**The representation language ISL<sup>FOL</sup>:** We base our representation on the expressive combination language the Image Schema Logic ISL<sup>FOL</sup> [3]. Following a popular temporalisation strategy (as in [10]), temporal structures are the primary model-theoretic objects, in turn based on Linear Temporal Logic over the reals (LTL). At each moment of time, we allow for the employment of secondary semantics to represent complex propositions. These atomic representations are topological assertions in 3D Euclidean space based on the Region Connection Calculus (RCC) [11], relative movement dimension using Qualitative Trajectory Calculus (QTC) [12] and relative object position using Ligozat’s Cardinal Directions (CD) [13]. Quantification is used to separate different sortal objects, whilst otherwise the syntax of the language follows a standard multi-modal logic paradigm. For more details on the logic we refer to [3, 14].

**The robotics framework:** While preliminary, the robotics framework we utilise is based on KnowRob [15], a knowledge representation and reasoning system. Knowrob uses Web ontology language (OWL) based on description logics to represent the knowledge. Prolog, a logic-based programming language is used to reason over the knowledge base and to assert new facts that are computed. We use the formal affordance model defined in [16] and represented in SOMA [17]. The formal model comprises of concepts including Affordance, Disposition and Role where Dispositions are properties of objects that takes on the role of a bearer and acts as a description of Affordance. With the model defined in [16], for an event requiring a certain affordance to execute a task, the task can be executed only if there is a presence of two suitably disposed objects with a trigger and a bearer role in the environment. This model is used in the algorithm for stacking.

## 2.3. Previous Work

**Image-schematic event representation for robotics:** In [7], the authors demonstrate how it is possible to divide the event of cracking eggs into bowls into image-schematic components. In previous work, we looked at how robotic action descriptors could be cored down into their image-schematic relationships. In [8], we defined image-schematic relations like VERTICALITY, SOURCE\_PATH\_GOAL, CONTAINMENT, LINK, CONTACT, SUPPORT to describe the action of cutting. The functional properties of objects and scene description defined using image schema concepts are used in [18] to construct a simulation of the scene and estimate the parameters necessary for performing the action of pouring. While in [19], the author uses qualitatively described functional object relations in a spatial arrangement to enable the agent to understand the causal structure embedded in the scene.

**Object stacking:** Robotics research on object manipulation comes in many forms, including work on stacking objects with a focus on the complexity of balancing objects on top of one another. The major challenge is to learn the physics of how shapes of different level of complexity can be stacked on top of one another without falling. [20] approach the problem by using a neural network to learn the geometric affordances. In [21], a reinforcement learning algorithm

was used to train a system how to stack a series of complex shapes while simultaneously ignoring irrelevant object features such as colour.

In more domestic environments, in which household items are to be stacked for more practical reasons, other challenges presents themselves. In [22], the authors introduce a method for stacking objects on shelves based on crowd sourced data.

**Learning functional relationships and affordances:** To perform a successful stacking, it is important to reason about the object affordance property. There are works (e.g. [20, 23, 24]) using machine learning to learn the extracted features representing the physical properties of objects in a scene to be able to reason about the individual object behaviour and how they behave in pairwise object interactions. But the problem is the machine learning models does not enable an understanding of how the object features relate to functional properties. There needs to be a semantic component as in image schemas to perform object stacking and also be able to explain in case of failure, when certain objects do not in the stack or failure in action when a robot executes the task of object stacking.

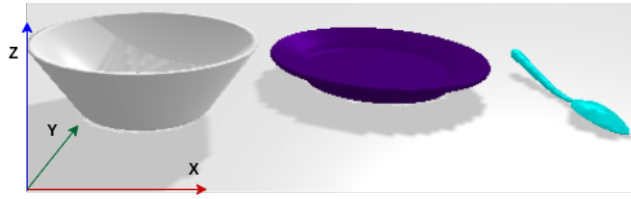
### 3. Stacking order algorithm

Computing the stacking order is a search problem to find a stable configuration that can be stacked from the given set of objects in a scene. The search space of this problem is  $n!$ , where  $n$  is the number of objects. Only rigid objects are considered in this work and the stack height is limited to 5 objects.

Algorithm 1 presents the stacking order by computing the disposition properties of the objects. By Turvey’s [25] definition of dispositions, there needs to be two objects whose disposition properties complement each other, for the affordances of the objects to be realised if the situation demands. For the task of stacking to be performed, one of the objects assumes the role of a bearer with the ability to support other objects and the second object with a trigger role can be placed on top of other objects. The roles of the objects can be interchanged, with the object that is in need of support acting as a bearer and the supporting object acting as a trigger. The important aspect is the presence of two suitably disposed objects that can enable stacking. The Rule *onTopOf* is used to verify if the considered objects satisfy the condition to be stacked and does not mean that the object  $O_1$  is placed on top of  $O_2$ . The rule is defined below,

$$\begin{aligned} \forall O_1, O_2 : onTopOf(O_2, O_1) \leftrightarrow \exists D_1, R_1 : & Object(O_1) \wedge Object(O_2) \wedge Disposition(D_1) \wedge \\ & Role(R_1) \wedge hasDisposition(O_1, D_1) \wedge hasTrigger(D_1, R_1) \wedge \\ & hasRole(O_2, R_1) \wedge canStack(O_1, O_2) \end{aligned} \quad (1)$$

It comprises of the unary predicates *Object*, *Role* and *Disposition* which are used to assert that  $O_1$  and  $O_2$  are instances of *Object*,  $R_1$  is of type *Role* and  $D_1$  is a *Disposition*. The predicate *hasDisposition* relates the object  $O_1$  with a suitable disposition  $D_1$ , similarly *hasTrigger* is a relation between  $D_1$  and the role type  $R_1$  that can act as a trigger for  $D_1$  and *hasRole* relates the object  $O_2$  to role  $R_1$  if the object can take on the role. The last predicate *canStack* is used



**Figure 1:** Object set: Bowl, Plate, Spoon

to perform geometric reasoning if the objects  $O_1$  and  $O_2$  playing the trigger and bearer roles afforded by the disposition  $D_1$  can be stacked.

The rule above explains how to check if two objects can be stacked. When a set of objects is considered, they are separated into two groups based on their disposition property. The separation of the objects is done with the reasoning that any object that can support or contain other objects with the disposition *Deposition* and *Containment* respectively, should be placed at the bottom of the stack. Likewise, objects without such dispositions such as spoons, as well as spherical objects that has the property of *Rollability*, are at the top.

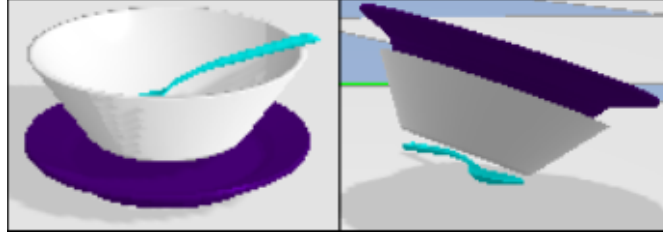
The objects are modelled with their top facing the positive *Z-axis* (height) and the length along the *X-axis* and the width along the *Y-axis*, see Figure 1. This is identical to how they are to be placed on a surface to ensure stability. To avoid comparing each object with the rest of the objects, we use a simple heuristic of sorting the objects based on their diagonal length along the *XY plane*. This is motivated with how it ensures the object with a large bottom surface is at the bottom of the stack.

Consider an example scenario with an ordered list of objects comprising of a plate, a spoon and a bowl as shown in Figure 1. First, we will apply Rule 1 to the plate and the bowl. The plate, having the disposition property of *Deposition*, needs a trigger role of type *DepositedObject*, as the flat nature of the plate affords to *SUPPORT* other objects when placed on top of it. Based on its physical features, the bowl can play the role of a *DepositedObject* and the relation *canStack* holds as well. Thus, the bowl can be placed on the plate. Continuing, with the bowl and the spoon, the bowl with the disposition property of *Containment* needs a trigger of type *ContainedObject*, as the concave shape allows *CONTAINMENT* of objects inside. If the spoon can be contained in the given bowl then the task of *Stacking* can be realised<sup>2</sup>. The obtained stack is similar to the left stack in Figure 2. The unstable arrangement of objects is one of the configurations that will not be considered by the algorithm as the spoon placed at the bottom will not offer any support to other objects or contain other objects.

In the next section, we define the actions necessary for stacking. Stacking is performed by transporting the objects in the order provided by the algorithm. The defined action sequence has to be performed recursively until all the objects are stacked.

---

<sup>2</sup>Note that dispositions often correlate with their image-schematic representations, but that we have chosen to keep their syntactic representation in the text.



**Figure 2:** Stacking objects with two different arrangements, stable and unstable stack

## 4. Action representation of stacking

In order to be able to execute successful object stacking, the event of stacking one object on top of another needs to be cored down and explained in terms of each of the individual actions. Stacking is a repetition of the task of transporting objects until all objects in our set have been placed on top of one another following the stacking algorithm.

The actions are classified into atomic and compound as described in the taxonomy of actions, see [26]. An atomic action is defined as below,

**Definition 1.** *An action causing a single change in the image schematic relation is an atomic action.*

For example, consider the action of *Lifting*, before the action is executed an object is in contact with the robot gripper and is supported by a surface. Once the action is executed, there exists no contact between the surface and the lifted object and the support is offered by the gripper. The *Lifting* action leads to a change in *SUPPORT* relationship, hence, *Lifting* is an atomic action. Compound actions are a composition of atomic actions or compound actions. By that definition, an action like *transporting* is a compound action comprising of two other compound actions: picking up and placing. Each of the compound action consists of smaller atomic descriptors that can be described in image-schematic terms.

Further, the pre- and post scene of the compound action is defined in terms of the spatial relations between the involved objects. The defined spatial relations and the object roles must hold for a successful task execution. By defining the pre- and post scene of an action, it is possible to infer the necessary sequence of actions to execute a task. The representation of the action with its initial and the terminal scene has the following syntax:

$\text{Action}(I) \leftrightarrow (\text{InitialScene}(T_1) \rightarrow \text{TerminalScene}(T_2))$ , where  $T_1, T_2$  are time points and  $I$  is an interval which begins at  $T_1$  and ends at  $T_2$  during which the action happens.

This means that to execute an Action, the InitialScene is a prerequisite and once the action is performed, it results in the TerminalScene, the goal state of the Action. The action takes place during the time interval  $T$  resulting in the end state at  $T_2$ , given the prerequisite is satisfied at some time point  $T_1$ . The ' $\rightarrow$ ' does not point to implication, it rather means that the defined goal state is achieved if the action is successfully executed under the given context. The initial scene of an action comprises of objects that can take on suitable roles and relevant spatial relation

---

**Algorithm 1** Rules for Stacking

---

**Input** *Objectlist*  $\leftarrow$  given set of objects in scene

**Output** *OrderedObjects*  $\leftarrow$  Stacking order of given objects

*ObjectsNotInStack*  $\leftarrow$  Objects that are not part of the stack

```
1: procedure COMPUTESORDEREDOBJECTLIST
2:   OrderedObjects  $\leftarrow \phi$ 
3:   Sortedlist  $\leftarrow$  Objectlist sorted based on their diagonal length in descending order ignoring the height
4:   /* considers objects that can support other objects */
5:   for object O in Sortedlist do
6:     if hasDisposition(O, D) and D in [Deposition, Containment] then
7:       Remove O from Sortedlist
8:       if OrderedObjects is empty then
9:         OrderedObjects  $\leftarrow$  OrderedObjects  $\cup$  O
10:      else
11:        /* Apply the Rule 1 */
12:        if onTopOf(O, Top) then
13:          OrderedObjects  $\leftarrow$  OrderedObjects  $\cup$  O
14:        end if
15:      end if
16:    end if
17:    Top  $\leftarrow$  OrderedObjects[N-1], N  $\leftarrow$  size of OrderedObjects
18:  end for
19:  /* considers objects that cannot support other objects */
20:  Top  $\leftarrow$  OrderedObjects[N-1], N  $\leftarrow$  size of OrderedObjects
21:  for object O in Sortedlist do
22:    /* Apply the Rule 1 */
23:    if onTopOf(O, Top) then
24:      Sortedlist.pop()
25:      OrderedObjects  $\leftarrow$  OrderedObjects  $\cup$  O
26:    end if
27:  end for
28:  ObjectsNotInStack = ObjectList - OrderedObjects
29:  Return OrderedObjects, ObjectsNotInStack
30: end procedure
```

---

among objects. Note that when the action is executed, the relationships between the objects and the roles the objects initially have, can change.

**For a transporting** action to happen, the prerequisite is the presence of a SUPPORT relation between two physical objects taking the roles of *Deposit* and *DepositedObject* at a time point  $T_1$ . When the task is completed successfully, there is again a SUPPORT relation between the

transported object and an object with the *Deposit* role at a time point  $T_2$ . The transporting predicate is parameterised by 3 objects and a time interval,  $O_1, O_2, O_3$  and  $I$ , where  $O_1$  is transported from  $O_2$  to  $O_3$  during  $I$ . The predicate *hasInterval* relates the *TimeInterval*  $I$  instance with the starting time point  $T_1$  and the end point  $T_2$ .

$$\begin{aligned} \forall O_1 O_2, O_3, I : \text{transporting}(O_1, O_2, O_3, I) \leftrightarrow & (\exists T_1, T_2 \text{ Object}(O_1) \wedge \text{Object}(O_2) \wedge \text{Object}(O_3) \wedge \\ & \text{Timepoint}(T_2) \wedge \text{TimeInterval}(I) \wedge T_2 > T_1 \wedge \\ & \text{hasInterval}(I, T_1, T_2) \wedge \text{hasRole}(O_1, \text{DepositedObject}) \wedge \\ & \text{hasRole}(O_2, \text{Deposit}) \wedge \text{supportedBy}(O_1, O_2, T_1) \\ & \rightarrow \text{hasRole}(O_3, \text{Deposit}) \wedge \text{supportedBy}(O_1, O_3, T_2)) \end{aligned}$$

**Picking up** comprises of several atomic actions: *LookingFor*, *LookingAt*, *MovingTo*, *Grasping* and *Lifting*. *LookingFor* and *LookingAt* are movements that are necessary to perceive the object. *MovingTo* is a movement towards the object so that the object is reachable. *Lifting* and *Grasping* involve interaction with objects, the gripper of the agent comes in contact with the object in *Grasping* and in *Lifting*, the agent is in control of the object and is being supported by the gripper. Similar to transporting, there needs to be a *SUPPORT* relation before and after the action is executed successfully. For picking up of  $O_1$  from  $O_2$  to happen, there needs to be a *SUPPORT* relation between the object and the surface. There is an  $O_3$  involved in picking up and the object  $O_1$  is supported by the agent once the action is complete. The action happens over a time interval which starts at  $T_1$  and ends at  $T_2$ .

$$\begin{aligned} \forall O_1, O_2, I : \text{pickingUp}(O_1, O_2, I) \leftrightarrow & (\exists T_1, T_2 : \text{Object}(O_1) \wedge \text{Timepoint}(T_1) \wedge \text{Timepoint}(T_2) \\ & \wedge \text{TimeInterval}(I) \wedge T_2 > T_1 \wedge \text{hasInterval}(I, T_1, T_2) \wedge \\ & \text{Object}(O_2) \wedge \text{hasRole}(O_2, \text{Deposit}) \wedge \\ & \text{hasRole}(O_1, \text{DepositedObject}) \wedge \text{supportedBy}(O_1, O_2, T_1) \\ & \rightarrow \exists O_3 \text{ Agent}(O_3) \wedge \text{hasRole}(O_3, \text{AgentRole}) \wedge \\ & \text{supportedBy}(O_1, O_3, T_2)) \end{aligned}$$

We provide the definition of the actions involving objects using ISL logic. The operator  $U$  is Until from Temporal Logic,  $\leftarrow$  and  $\rightsquigarrow$  are used as in Qualitative Trajectory Calculus to denote the movement of the object away from, and towards the other object respectively. *Grasping* defined below is a combination of the image-schematic aspects of *LINK* and *CONTAINMENT*. When the grasping action is performed, the object  $O_1$  and gripper  $O_2$  are initially disconnected and eventually the object becomes a part of the gripper and there exists a contact force between the gripper and the object.

$$\begin{aligned} \forall O_1 : \text{grasping}(O_1) \leftrightarrow & \exists O_2 \text{ Object}(O_1) \wedge \text{Gripper}(O_2) \wedge \\ & DC(O_1, O_2) U (TPP(O_1, O_2) \wedge \text{force}(O_1, O_2)) \end{aligned}$$

*Lifting* is equivalent to a combination of image schema concepts namely *UP*, *SOURCE\_PATH* and *SUPPORT*. During the execution of *Lifting*, the object  $O_1$  to be lifted is supported by the arm  $O_2$  while the arm supporting the object, moves along an upwards path  $P$  away from the support



surface  $O_3$ . The execution is complete when the arm lies above the support surface and there is no contact between the object and support surface.

$$\begin{aligned} \forall O_1 : \text{lifting}(O_1) \leftrightarrow & \exists O_2, O_3, P \text{ Object}(O_1) \wedge \text{Arm}(O_2) \wedge \text{Object}(O_3) \wedge \\ & \text{Path}(P) \wedge \text{TPP}(O_1, O_2) \wedge \text{force}(O_1, O_2) \wedge \\ & ((O_1 \leftrightarrow O_3 \wedge \text{contact}(O_1, P)) \cup \\ & (\neg(\text{contact}(O_1, P)) \wedge \neg(\text{contact}(O_2, O_3)) \wedge \text{Above}(O_1, O_3))) \end{aligned}$$

**Placing** The placing action is performing picking up in reverse order, which comprises of the following atomic actions: MovingTo, Lowering, Releasing, MovingAway. MovingTo is the action of an agent moving towards the object to achieve reachability and MovingAway is the act of moving away from the object once it is placed stable on a surface. Again, we define Lowering and Releasing actions which involve objects using ISL. The condition necessary for placing to be executed is the existence of a SUPPORT that is provided by the agent  $O_2$  to an object  $O_1$  at some time point  $T_1$ . When the placing action is performed successfully during the time interval  $I$ , there exists no contact between the agent and the object, and there is a SUPPORT relation between the placed object  $O_1$  and the object  $O_3$  that can afford to support at time point  $T_2$ .

$$\begin{aligned} \forall O_1, O_2, I : \text{placing}(O_1, O_2, I) \leftrightarrow & (\exists O_3, T_1, T_2 \text{ Object}(O_1) \wedge \text{Timepoint}(T_1) \wedge \text{Timepoint}(T_2) \wedge \\ & \text{Timeinterval}(I) \wedge T_2 > T_1 \wedge \text{Agent}(O_3) \wedge \\ & \text{hasInterval}(I, T_1, T_2) \wedge \text{hasRole}(O_1, \text{DepositedObject}) \wedge \\ & \text{hasRole}(O_3, \text{AgentRole}) \wedge \text{supportedBy}(O_1, O_3, T_1) \\ & \rightarrow \text{Object}(O_2) \wedge \text{hasRole}(O_2, \text{Deposit}) \wedge \\ & \text{supportedBy}(O_1, O_2, T_2)) \end{aligned}$$

Lowering, an atomic action of Placing is a combination of , PATH\_GOAL and SUPPORT. As in Lifting, there exists a contact force between the object  $O_1$  and the arm  $O_2$  during the execution phase and the arm moves towards the supporting surface  $O_3$  along a path  $P$ . The action is successfully completed, if at the end, there exists a contact between the object held by the gripper and the object with a supporting surface.

$$\begin{aligned} \forall O_1 : \text{lowering}(O_1) \leftrightarrow & \exists O_2, O_3, P \text{ Object}(O_1) \wedge \text{Arm}(O_2) \wedge \text{Object}(O_3) \wedge \\ & \text{Path}(P) \wedge \text{TPP}(O_1, O_2) \wedge \text{force}(O_1, O_2) \wedge \\ & ((O_1 \rightsquigarrow O_3 \wedge \text{contact}(O_1, P) \wedge \text{Above}(O_1, O_3)) \cup \\ & (\neg(\text{contact}(O_1, P)) \wedge (\text{contact}(O_2, O_3)))) \end{aligned}$$

The next action is Releasing, which begins with a contact between the gripper  $O_2$  and the object  $O_1$  and ends with the gripper and the object being disconnected. Once the action is complete, the object has no contact with the agent.

$$\begin{aligned} \forall O_1 : \text{releasing}(O_1) \leftrightarrow & \exists O_2 \text{ Object}(O_1) \wedge \text{Gripper}(O_2) \wedge \\ & (\text{TPP}(O_1, O_2) \wedge \text{force}(O_1, O_2)) \cup \text{DC}(O_1, O_2) \end{aligned}$$

## 5. Discussion and Future Work

The complexity of designing methods for cognitive robots to successfully stack objects, extends the problem for efficient manipulation. Performing a successful manipulation, requires an understanding of how the objects behave in a particular arrangement. In this paper, we utilised object affordances and introduced an algorithm for intelligent stacking based on individual object properties, and a rule that acts on objects pairwise to check if on top of relation holds between them. Taking inspiration from human cognition, we argue that stacking objects of different properties require the agent to understand the underlying rules pertaining to how a particular object interacts with one another. Our inclination to use object affordances stems from the fact that they can be described to relate to spatiotemporal relationships, image schemas, which also constitute the fundamental steps in action execution. Further, we showcase how image-schematic event representation in the format of ISL<sup>FOL</sup>, can be used to describe the action descriptors. The described action descriptors can then be used for failure monitoring and also in case of failures in execution, for explaining the reason behind a failure. For example, with the image-schematic definition of Lifting, there needs to be a support relation between the lifted object and the gripper involved in lifting and the gripper in contact with the object must be moving away from the supporting surface in such a way the gripper lies above the supporting surface. In case of an object slipping while lifting, this can be detected with our definition of lifting as the support relation between the lifted object and the gripper is violated.

Much work (e.g. [27, 23]) has used simulation as a means to understand the scene. Considering an instance of stacking performed only with a physics simulator, the number of random samples required by the simulator to understand the object properties when they interact with other objects is huge. Also a part of the generated samples might consist of spatial arrangement of objects that violates what the object affords. For example, in Figure 2, there is an object constellation with spoon at the bottom with a bowl on the top. However, this is an invalid combination according to our qualitative description while this is a completely valid sample considered by the simulator. Generating samples that respect our qualitative descriptions of the scene and object properties can significantly reduce the number of samples needed.

Due to the current theoretical nature of the work, evaluations of the underlying ideas are still lacking. In the future, we will rectify this, by investigating how the formalised image schemas can be used by the robot control programs. To use image-schema formalisms along with robot control programs, it is necessary to represent and quantify the defined spatiotemporal relations using physics based simulators. The image schematic event segmentation defined in 4 can be used to infer the sequence of actions that the robot has to repeat to perform stacking until all the objects received from the algorithm 1 are in the stack. To execute the inferred actions successfully, we need failure handling. With the action descriptors defined in ISL logic, failure monitoring can be performed. To combine the action sequence for stacking and failure monitoring in a modular fashion, behaviour trees will be used. For instance, a behaviour tree can be generated with a parallel node consisting of two children, one for performing the action execution and the other monitoring the failures based on the action executed. The action execution node consists of a child node for each atomic action to be executed and will execute the actions in sequence until all of them succeed. We intend to use Giskard, [28], a constraint-based robot controller for executing the stacking task. As Giskard already uses behaviour trees

for planning and executing the goals, it is relatively less complex to integrate our approach with Giskard. And with regards to the rules of stacking, we are interested in a long term research goal of extracting the rules by letting the agent interact with the environment. For this we want to use observational data for modelling the relation between the physical attributes of the object that influence the stacking stability and also collect intervention data by allowing the agent to interact with the environment similar to a curiosity-driven exploration in simulation [29].

## Acknowledgements

The research reported in this paper has been partially supported by the Federal Ministry for Economic Affairs and Energy BMWi within the Knowledge4Retail project, subproject semantic Digital Twin 01MK20001M (<https://knowledge4retail.org>).

## References

- [1] H. Moravec, *Mind children: The future of robot and human intelligence*, Harvard University Press, 1988.
- [2] M. Johnson, *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*, The University of Chicago Press, Chicago and London, 1987.
- [3] M. M. Hedblom, *Image Schemas and Concept Invention: Cognitive, Logical, and Linguistic Investigations*, Cognitive Technologies, Springer Computer Science, 2020.
- [4] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, et al., *Rearrangement: A challenge for embodied ai*, arXiv preprint arXiv:2011.01975 (2020).
- [5] L. Smith, M. Gasser, *The development of embodied cognition: Six lessons from babies*, *Artif. Life* 11 (2005) 13–30. URL: <https://doi.org/10.1162/1064546053278973>. doi:PATH10.1162/1064546053278973.
- [6] T. Oakley, *Image schema*, in: D. Geeraerts, H. Cuyckens (Eds.), *The Oxford Handbook of Cognitive Linguistics*, Oxford University Press, Oxford, 2010, pp. 214–235.
- [7] M. M. Hedblom, O. Kutz, R. Peñaloza, G. Guizzardi, *Image schema combinations and complex events*, *KI-Künstliche Intelligenz* 33 (2019) 279–291.
- [8] K. Dhanabalachandran, V. Hassouna, M. M. Hedblom, M. Küempel, N. Leusmann, M. Beetz, *Cutting events: Towards autonomous plan adaption by robotic agents through image-schematic event segmentation*, in: *Proceedings of the 11th on Knowledge Capture Conference, K-CAP '21*, Association for Computing Machinery, New York, NY, USA, 2021, p. 25–32. URL: <https://doi.org/10.1145/3460210.3493585>. doi:PATH10.1145/3460210.3493585.
- [9] M. M. Hedblom, M. Pomarlan, R. Porzel, R. Malaka, M. Beetz, *Dynamic action selection using image schema-based reasoning for robots*, in: *Proc. of the Joint Ontology Workshops*, 2021.
- [10] M. Finger, D. M. Gabbay, *Adding a Temporal Dimension to a Logic System*, *Journal of Logic, Language and Information* 1 (1993) 203–233.
- [11] D. A. Randell, Z. Cui, A. G. Cohn, *A spatial logic based on regions and connection*, in: *Proc. 3rd Int. Conf. on Knowledge Rep. and Reas.*, 1992.

- [12] N. Van De Weghe, A. G. Cohn, G. De Tré, P. D. Maeyer, A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems, *Control and cybernetics* 35 (2006) 97–119.
- [13] G. Ligozat, Reasoning about cardinal directions, *J. Vis. Lang. Comput.* 9 (1998) 23–44.
- [14] M. M. Hedblom, O. Kutz, T. Mossakowski, F. Neuhaus, Between contact and support: Introducing a logic for image schemas and directed movement, in: F. Esposito, R. Basili, S. Ferilli, F. A. Lisi (Eds.), *AI\*IA 2017: Advances in Artificial Intelligence*, 2017, pp. 256–268.
- [15] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoglu, G. Bartels, KnowRob 2.0 – A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents, in: *2018 IEEE Int. Conf. on Robotics and Automation, ICRA 2018*, Brisbane, Australia, May 21–25, 2018, 2018, pp. 512–519. doi:PATH10.1109/ICRA.2018.8460964.
- [16] D. Beßler, R. Porzel, M. Pomarlan, M. Beetz, R. Malaka, J. Bateman, A formal model of affordances for flexible robotic task execution, in: *ECAI 2020*, IOS Press, 2020, pp. 2425–2432.
- [17] D. Beßler, R. Porzel, M. Pomarlan, A. Vyas, S. Höffner, M. Beetz, R. Malaka, J. Bateman, Foundations of the socio-physical model of activities (soma) for autonomous robotic agents, *arXiv preprint arXiv:2011.11972* (2020).
- [18] M. Pomarlan, M. M. Hedblom, R. Porzel, Panta Rhei: curiosity-driven exploration to learn the image-schematic affordances of pouring liquids, in: *Proceedings of the 29th Irish Conference on Artificial Intelligence and Cognitive Science*, Dublin, Ireland, 2021.
- [19] M. Pomarlan, J. A. Bateman, Embodied functional relations: A formal account combining abstract logical theory with grounding in simulation., in: *FOIS*, 2020, pp. 155–168.
- [20] O. Groth, F. B. Fuchs, I. Posner, A. Vedaldi, Shapestacks: Learning vision-based physical intuition for generalised object stacking, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 702–717.
- [21] A. X. Lee, C. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, C. Fantacci, J. E. Chen, A. Raju, R. Jeong, M. Neunert, A. Laurens, S. Saliceti, F. Casarini, M. Riedmiller, R. Hadsell, F. Nori, Beyond pick-and-place: Tackling robotic stacking of diverse shapes, 2021. *arXiv:2110.06192*.
- [22] N. Abdo, C. Stachniss, L. Spinello, W. Burgard, Robot, organize my shelves! tidying up objects by predicting user preferences, in: *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 1557–1564.
- [23] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, K. Kavukcuoglu, Interaction networks for learning about objects, relations and physics, 2016. *arXiv:1612.00222*.
- [24] D. Raposo, A. Santoro, D. Barrett, R. Pascanu, T. Lillicrap, P. Battaglia, Discovering objects and their relations from entangled scene representations, *arXiv preprint arXiv:1702.05068* (2017).
- [25] M. Turvey, *Ecological foundations of cognition: Invariants of perception and action*. (1992).
- [26] P. Zech, E. Renaudo, S. Haller, X. Zhang, J. Piater, Action representations in robotics: A taxonomy and systematic classification, *The International Journal of Robotics Research* 38 (2019) 518–562.
- [27] P. W. Battaglia, J. B. Hamrick, J. B. Tenenbaum, Simulation as an engine of physical scene understanding, *Proceedings of the National Academy of Sciences* 110 (2013) 18327–18332.
- [28] Z. Fang, G. Bartels, M. Beetz, Learning models for constraint-based motion parameteriza-

tion from interactive physics-based simulation, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2016, pp. 4005–4012.

- [29] M. Gasse, D. Grasset, G. Gaudron, P.-Y. Oudeyer, Causal reinforcement learning using observational and interventional data, 2021. [arXiv:2106.14421](https://arxiv.org/abs/2106.14421).