Semantic Web 0 (2022) 1–0
IOS Press

Deep Understanding of Everyday Activity Commands for Household Robots

Sebastian Höffner [a,*], Robert Porzel [a], Maria M. Hedblom [c], Mihai Pomarlan [b], Vanja Sophie Cangalovic [b], Johannes Pfau [a], John A. Bateman [b] and Rainer Malaka [a]
[a] *Digital Media Lab, TZI, University of Bremen, Germany*
[b] *Computational Linguistics, FB10, University of Bremen, Germany*
[c] *Jönköping Artificial Intelligence Laboratory, JAIL, Jönköping University, Sweden*
E-mail: {hoeffner,porzel,pomarlan,vanja,jpfau,bateman,malaka}@uni-bremen.de, maria.hedblom@ju.se

Editor: Katja Hose, Database and Web Technologies, Aalborg Universitet, Denmark
Solicited reviews: Stefano Borgo, Laboratory for Applied Ontology, Istituto di Scienze e Tecnologie della Cognizione, Italy; Christopher Myers, Air Force Research Laboratory, Wright-Patterson AFB, Ohio, USA; Five anonymous reviewers

Abstract. Going from natural language directions to fully specified executable plans for household robots involves a challenging variety of reasoning steps. In this paper, a processing pipeline to tackle these steps for natural language directions is proposed and implemented. It uses the ontological Socio-physical Model of Activities (SOMA) as a common interface between its components. The pipeline includes a natural language parser and a module for natural language grounding. Several reasoning steps formulate simulation plans, in which robot actions are guided by data gathered using human computation. As a last step, the pipeline simulates the given natural language direction inside a virtual environment. The major advantage of employing an overarching ontological framework is that its asserted facts can be stored alongside the semantics of directions, contextual knowledge, and annotated activity models in one central knowledge base. This allows for a unified and efficient knowledge retrieval across all pipeline components, providing flexibility and reasoning capabilities as symbolic knowledge is combined with annotated sub-symbolic models.

Keywords: Natural language understanding, Simulation, Grounding, Everyday activities, SOMA, Robotics

1. Introduction

Performing household activities such as cooking and cleaning have, until recently, been the exclusive provenance of human participants. However, the development of robotic agents that can perform different tasks of increasing complexity is slowly changing this state of affairs, creating new opportunities in the domain of household robotics. Most commonly, any robot activity starts with the robot receiving directions or commands for that specific activity. Today, this is mostly done via programming languages or pre-defined user interfaces, but this changes rapidly. For everyday activities, instructions could be given either verbally from a human or through written texts such as recipes and procedures found in online repositories, e.g., from wikiHow. From the perspective of the robot, these instructions tend to be vague and imprecise as natural language generally employs ambiguous, abstract, and non-verbal cues. Often, instructions omit vital semantic components such as determiners, quantities, or even the objects they refer

*Corresponding author. E-mail: hoeffner@uni-bremen.de.

to [1, 2]. Still, asking a human to, for instance, *"take the cup to the table"* will typically result in a satisfactory outcome. Humans excel despite many uncertain variables existing in the environment: for example, the cups might all have been put away in a cupboard or the path to the table could be blocked by chairs.

In comparison, artificial agents lack the same depth of symbol grounding between linguistic cues and real world objects, as well as the capacity for insight and prospection to reason about instructions in relation to the real world and the changing states of that world. In order to turn an underspecified text, issued or taken from the web, into a detailed robotic action plan, various processing steps are necessary, some based on symbolic reasoning and some on numeric simulations or data sets. The research question in this paper is therefore the following: how can ontological knowledge be used to extract and evaluate parameters from a natural language direction in order to simulate it formally? The solution proposed in this work is the Deep Language Understanding (DLU) processing pipeline. This pipeline employs the ontological Socio-physical Model of Activities (SOMA) [3], which serves not only to define the interfaces of the multi-component pipeline but also to connect numeric data and simulations with symbolic reasoning processes.

1.1. Scope of DLU

Directions, instructions delivered textually, and commands, delivered verbally, are special instructions as they always demand an action. In contrast, other instructions can also be descriptions or specifications of circumstances. For example, *"take the cup to the table"* demands the addressee to perform an action. Sentences such as "knives must be placed to the right of the plate" or "the onions should be brown after 20 minutes" might entail directions or commands, but they do not explicitly call to action. In this work, the focus lies on directions as an important special case of instructions.

More specifically, DLU only handles directions in which a trajector needs to be moved by an agent along a trajectory, thus directions which can be described using a SOURCE_PATH_GOAL (SPG)[1] schema. Example directions which DLU does not handle are:

	Direction	Explanation
(1)	*"Take the cheese out with the bottle"*	Tool use is currently not part of DLU as, at the time of writing, this requires additional modeling in SOMA and Streaming Construction Grammar (SCG).
(2)	*"Put the bottle back"*	*Back* and other concepts requiring keeping track of a history are out of scope for DLU right now.
(3)	*"Go to the fridge"*	DLU does not support actions which manipulate the state of the actor.
(4)	*"Stay away from the cooker"*	Directions demanding avoidance of actions or indirectly describe actions are also not covered.

DLU also only handles directions in isolation. That means, if a previous direction is referenced or sets up a certain pre-condition for another direction, SCG will not know about it and not provide a correct semantic specification as described in Section 3.2.

The context in DLU is always the physical context, that is a scene in form of a semantic map as discussed in Section 3.3. No other context is taken into account, neither previous instructions nor social interactions or information from other knowledge graphs.

Another important point is that DLU assumes error-free communication; thus faulty directions such as *"Take the chair to the table"* when in fact instead of chair the user meant cup will not be corrected by DLU, mostly because of the isolation and context arguments above.

To evaluate the DLU pipeline, the direction *"take the cup to the table"* is used, as it has several interesting properties: a) *"Take"* makes no specific assumptions about the means, but together with *"to"* it entails a movement with a trajectory. b) *"The cup"* could easily be replaced with other items, based on similar functional properties or

[1] Following convention, image-schematic notions are written in small capitals.

affordances. c) *"To the table"* does not explicitly specify that the cup should be place on top of it, but only that it needs to be close to it. d) While two items are involved in the direction, one remains stationary. e) Additionally, the task is rather forgiving in terms of precision. While covering a pot requires placing a lid exactly on top of it, placing the cup on the table allows for more freedom in location (albeit not in orientation). f) Lastly, taking a cup to a table is usually a mundane exercise for humans. The direction is thus prototypical of everyday activities and the directions DLU handles.

In summary, this work builds upon the existing SOMA ontology [3], the semantic parser SCG [4, 5], and the human computation game Kitchen Clash [6]. In this work, a pipeline is build around these three modules to parse directions into semantic specifications. To ground the directions inside a simulated physical context, this work introduces a prototype heuristic for grounding. Additionally, this work generalizes data collected using Kitchen Clash, stores it inside an ontology for task specific retrieval, and successfully simulates its generated task plans in a kitchen environment in Unity.

2. Semantic Grounding and Ontologies

Research on natural language understanding for artificial agents dates back at least to the late 1940s. Recent examples include various artificial neural networks such as BERT [7] and GPT-2 [8] which can extract information from written text, answer questions, or tell stories when presented with a short teaser. These and similar machine learning techniques power chatbots and virtual assistants that rely on natural language understanding to answer questions[2] or control smart home appliances [10]. Additionally, there are systems that generate scenes from natural language [11, 12] and methods to control non-player characters in computer games with unstructured natural language commands [13]. All of these systems present various possible solution strategies to natural language understanding, but the problem of symbol grounding remains. The symbol grounding problem can be described with the *semiotic triangle*, which relates *concepts*, *symbols*, and *objects* to each other [14, 15].

Dealing with the transition from symbols in language to actions and objects in the real world is partly a problem due to the fact that the underlying deep understanding of natural language is not a direct mapping from either syntax or grammar. Instead, semantics appears in the form of linguistic constructs that correspond to particular mental patterns of meaning [16]. A theory for how these constructs manifest in the cognitive sphere can be constructed by grounding it in the embodied experiences of agents, thereby offering a seamless transition to simulations and robotics research. By looking to the embodied theories of cognition [17], recorded human activity, simulation, and action execution of robots can be used as a foundation for how to construct meaningful ontological structures. One proposal for how experiences turn into meaningful constructs is through conceptual patterns called image schemas [18]. They are spatiotemporal relationships that to varying degrees capture the specifics of particular static relationships and dynamic transformations of objects, agents and environments. Relationships like movement between points (SPG), relative distance in the NEAR_FAR schema and VERTICALITY are important image schemas to consider when performing object manipulations. While image schemas are predominantly studied in cognitive and linguistic settings, turning the theory into a formally applicable method has been initiated in terms of using them as logically formulated theories, hierarchically structured based on their internal compositionality [19].

In DLU, the first steps of using image-schematic relationships as ontological micro-theories included in SOMA are introduced with the purpose to ground robotic action plans into the large body of linguistic research on embodied semantics. The theories define slots, such as *trajector* for an SPG, which can be filled using parameters (cf. Section 3.5), motivated by Bergen and Chang's embodied construction grammar [20]. To extract the theories from natural language, the SCG parser [5] is employed.

While this application is a novel research contribution, employing ontologies for knowledge representation and reasoning in autonomous robot control is an established field of research with developments in both service and industrial robotics (see [21] for an overview). One example in the industrial robotics domain is the ROSETTA project [22, 23]. The initial scope of this project was reconfiguration and adaptation of robot-based manufacturing

[2] E.g., Alexa (https://developer.amazon.com/alexa), Almond [9] (https://almond.stanford.edu/), Google assistant (https://assistant.google.com/), Siri (https://www.apple.com/siri/)

cells, however, the authors have further developed their activity modeling for coping with a wider range of industrial tasks. Other authors have focused on modeling industrial task structure, part geometry features, or task teaching from examples (e.g. [24–26]). These industrial tasks tend to be more structured and less demanding in terms of flexibility compared to the everyday activity domain that is the focus of this paper.

Another ontology is described by Eberhart et al. [27], who use a domain specific ontology to model instructions for cognitive tasks. In contrast to this work, they do not simulate instructions and work in a more abstract scenario. However, their ontology is similar in spirit to some modules in SOMA [3] as both provide models of task instructions.

An approach to activity modeling in the service robotics domain is presented by Tenorth and Beetz [28]. The scope of their work is similar to this work, as the authors also consider how activity knowledge can be used to fill knowledge gaps in abstract instructions given to a robotic agent performing everyday activities. However, the scope of the work presented here is wider, as it is also considered here how knowledge can be used for the integration of numeric approaches and reasoning. Another difference is that, in Tenorth and Beetz' modeling, there is no distinction between physical and social context, and therefore, it is less expressive than SOMA which is used here.

A more general approach to activity modeling for robotic agents is presented by the IEEE-RAS working group Ontologies for Robotics and Automation (ORA) [29]. The group has the goal of defining a standard ontology for various sub-domains of robotics, including a model for object manipulation tasks. It has defined a core ORA ontology [30], as well as additional modules for industrial tasks such as kitting [31].

Interactive task learning, the work by Kirk et al. [32], also presented by Gluck and Laird [33], is similar to DLU in that they also simulate natural language instructions. In contrast to DLU, their focus in on task learning for puzzles and games, while DLU puts the focus on robust task execution for household tasks. Learning in DLU is mostly an off-line process and, for example, seeded using human computation.

3. System Overview

The DLU processing pipeline consists of the interconnected components depicted in Figure 1. In this section, the most important components are described.

DLU starts by parsing a sentence into semantic specifications (c) (Figure 1) based on SOMA (a) using the Streaming Construction Grammar (SCG) parser (b) [5]. Next, DLU determines referents (f) for the extracted semantic specifications in a pre-defined semantic map (d) of a kitchen scene. Additionally, the pipeline extracts image schema theories (e) from the semantic specifications. Each image schema theory has a set of parameters which DLU fills with values. These values are either references to objects or other image schemas, or they are values from evaluating pre-trained models. For example, one model discussed below contains information on where glasses can be placed on top of a table. Eventually, all knowledge from the previous steps—parsing, grounding, theory extraction—is combined into a sequence of trajectories (g), which DLU executes inside a game engine (h).

3.1. SOMA and other Ontologies

Across the DLU pipeline the Socio-physical Model of Activities (SOMA) is employed as a common interface. SOMA is based on the DOLCE+DnS Ultralite (DUL) foundational framework and its plugin Information Objects ontology (IOLite) [3, 34, 35]. Consequently, SOMA has two knowledge branches; one physical and one social [3], which leads to a distinction between objects and actions in the physical branch on the one hand, as well as roles and tasks in the social branch on the other. Beßler et al. explain that axiomatizations in the physical branch express physical contexts which can be classified by axiomatization in the social context [3]. For example, a cup and its properties of being a designed physical artifact would be described using parts of the physical branch, but its potential usage or affordances would be axiomatized within the social branch. SOMA is built out of multiple modules for different aspects. For example, the *SAY* module defines the linguistic theories and knowledge required by SCG, and the *HOME* module defines typical household objects. For DLU, SOMA is an optimal choice ontology as it is designed to describe actions with high precision.

To keep the ontology prefix definitions in the article at a minimum, all of the used prefixes are defined in Listing 1.

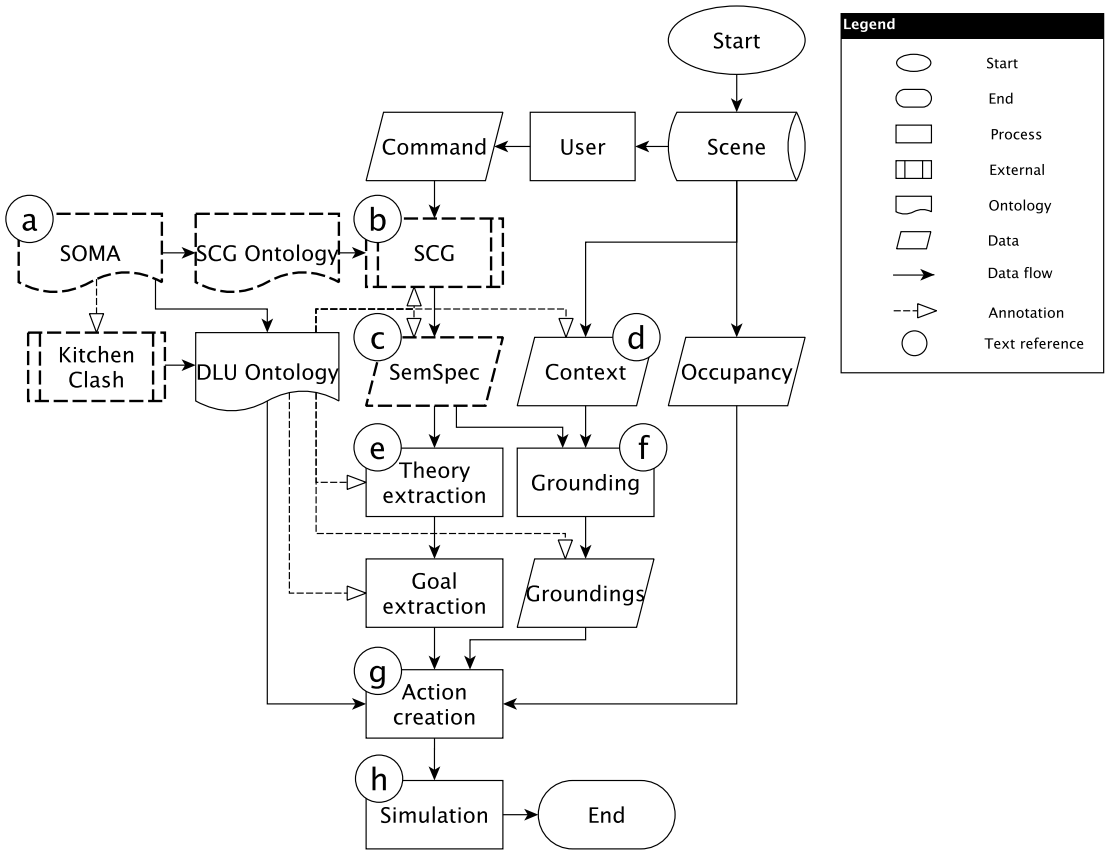

Figure 1. The architecture of the implemented natural language understanding pipeline DLU. A user supplies a direction to be performed in a given scene. The semantic parser SCG (b) parses the direction into a semantic specification (c). Together with the context information (d), the semantic specification is grounded (f) in the scene. Subsequent reasoning steps create action plans (g) and simulate them (h). Interoperability between the modules is achieved with the help of SOMA (a). The data flow between *Kitchen Clash* [6] and the DLU ontology and from the DLU to the action creation is slightly unexpected: in DLU, the models derived via human computation in Kitchen Clash are stored inside the ontology. They are retrieved during the action creation to facilitate human-like behavior. *Note:* This diagram represents the actual *use* of data, not the implementation: the output of (f) is available for (e), but currently not exploited there. *Dashed modules: previous work. Dashed white arrows: ontological annotations.*

3.2. SCG / Semantic Specification

To extract a semantic specification from a sentence, the Streaming Construction Grammar (SCG) parser [5] is employed. Given a textual input, it produces a set of triples which describe the sentence's syntax as well as semantics. The underlying grammar maps the semantics onto SOMA concepts where possible[3], and employs a custom set of ontologies for linguistic features. The produced terms are tagged as syntactic or semantic concepts to allow for straightforward filtering. This is currently used in DLU, which operates on semantics as discussed in Section 3.5. An example of semantic triples can be found in Listing 2.

3.3. Context / Semantic Map

For a direction in the scope of DLU to be executed, an environment with objects to manipulate is mandatory. This environment is a *context* or a *scene*. Figure 2 shows a kitchen scene in which actions in DLU are simulated. In the semantic map, each object in the kitchen scene is already annotated with unique identifiers and semantic

[3]If a mapping is not possible, the output will be less detailed.

```
PREFIX cg: <.../clj/scg/grammars/cooking.clj#>
PREFIX dlu: <.../dlu/dlu/EASE-DLUext.owl#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX li: <.../master/linguisticInfo.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX scg: <https://gitlab.informatik.uni-bremen.de/vanja/scg#>
PREFIX soma: <http://www.ease-crc.org/ont/SOMA.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Listing 1: All prefixes for ontology namespaces used in this paper. Some are shortened for legibility, the full list is available online https://osf.io/bp34y/.

```
<urn:uuid:18b08e99-5576-4d83-8066-8a58244c02ab>
        a                       dlu:Cup ,
                                soma:DesignedContainer ,
                                li:Referent ;
        scg:tag                 cg:semantic ;
        cg:has-number-value     li:Singular ;
        cg:identifiability      cg:indefinite .
```

Listing 2: An excerpt from SCG output: semantic triples involving a cup in n3/turtle-notation. The term 18b08e99-…has three is-a relations: it is a **dlu:Cup**, a **soma:DesignedContainer**, and a linguistic **li:Referent**. This cluster of triples is relevant to the semantics of the sentence thus tagged with **cg:semantic**. Additional linguistic properties of the term 18b08e99 is that it is described by a singular and has an indefinite article. In short, this semantic specification closely describes the *a cup* part of *"take the cup to the table."* Note: SCG uses different prefixes, here they are adjusted for consistency.

labels aligned with SOMA[4]. This allows for storing knowledge about the scene in the same format as the semantic specifications. In essence, the context resembles a semantic map of the scene. However, no semantic mapping of the scene is performed in DLU – instead, for the time being, an a priori semantic map is assumed to be provided. This makes sense, as the semantic mapping approaches have become very good in creating semantic maps from real world data, grounding them ontologically, and in refining them using CAD models [36–39].

3.4. Grounding

In DLU, the parts of the semiotic triangle are defined in SOMA. The symbols are part of the natural language direction, e.g., the word "cup". The objects are the entities in the kitchen scene, e.g., the table entity has a mesh, textures, and defined physical properties. And the mapping from the symbols to the concepts is performed by SCG. With the semantic map, the mapping from the objects to the concepts are given a priori.

The missing link, the mapping between the objects and symbols, is called *grounding* in DLU; this is also known as *referencing* or can be described as *framing*. One might wonder why the missing link needs to be established at all: there is a link from the symbols via concepts to objects and vice-versa. However, these links are not sufficient to fully disambiguate a sentence under a given context. For example, if in a kitchen scene one cup is on top of a shelf and another cup stands on a counter top, *"take the cup to the table"* might refer to either. But from the context it should be clear which of the cups is the correct one, as otherwise the speaker would have made more specific instructions. To resolve these ambiguities, we need to find the direct mapping between specific objects and symbols.

Grounding is implemented using an ad-hoc heuristic in DLU. The implementation is a prototype placeholder for future sophisticated methods. The heuristic designed for DLU is: a symbol references an object if the mean path depth of their common types to **dul:PhysicalObject** is maximal. This means, that after parsing *"take*

[4]The concepts are from the DLU ontology, which is a descendant of SOMA.

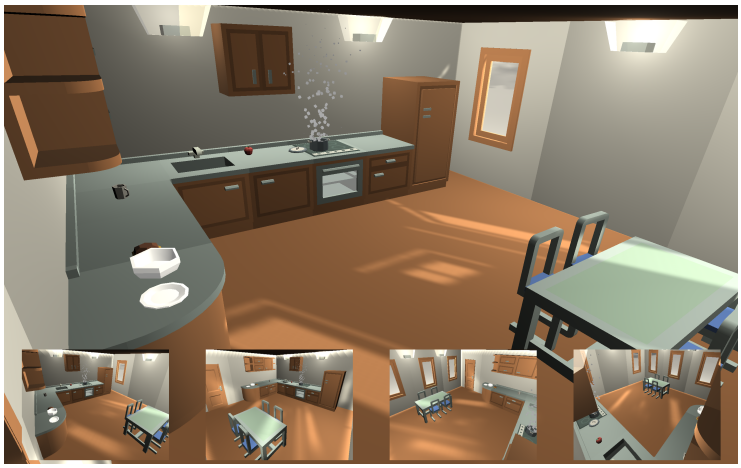

Figure 2. The kitchen scene. All objects—chairs, bowls, pots, cupboards, etc.—have semantic labels aligned with SOMA and unique identifiers, e.g., **<urn:cup_0>**.

```
<urn:Cup> rdf:type dlu:Cup .
<urn:Cup> rdf:type soma:DesignedContainer .
<urn:Cup> scg:tag cg:semantic .

<urn:cup_0> rdf:type soma:DesignedContainer .
<urn:cup_0> rdf:type dlu:Cup .
<urn:cup_0> rdf:type owl:NamedIndividual .

<urn:fridge_0> rdf:type soma:DesignedContainer .
<urn:fridge_0> rdf:type soma:Fridge .
<urn:fridge_0> rdf:type owl:NamedIndividual .
```

Listing 3: Semantic specifications for the term referred to as **<urn:Cup>** and for the objects **<urn:cup_0>** and **<urn:fridge_0>** from the semantic map. Here, the tag **cg:semantic** distinguishes symbols such as **<urn:Cup>** from objects, which are of type **owl:NamedIndividual**. This is because in SOMA, there is no distinction between symbols and objects, there exist only terms. A term which references an object assumes the type of its object's concept. A term which references a symbol analogously assumes the type of its symbol's concept. The tag **cg:semantic** is added by SCG to the symbol-terms and the **owl:NamedIndividual** property exists in the semantic map for each object-term.

the cup to the table" with SCG and storing the results alongside the semantic map in the knowledge base, DLU might have—among others—the triples in Listing 3 available. In Listing 3, there is a symbol **<urn:Cup>** from the semantic specification and the two objects **<urn:cup_0>** and **<urn:fridge_0>** from the semantic map. During the grounding step, DLU has to find a mapping between **<urn:Cup>** and **<urn:cup_0>**, but avoid matching the symbol cup to the fridge. Given the examples in Listing 3, the SPARQL query Listing 4 returns pairs and object types as can be seen in Table 1.

The Prolog predicate[5] `rdf_db:rdf_reachable/5` finds the depth of a path in a SPARQL query, which is used for the heuristic. For each type, its path depth from **dul:PhysicalObject** is determined. The path depth is the minimal required number of subclass relations from the object type to **dul:PhysicalObject**. In the example, **soma:DesignedContainer** requires three relations, while **dlu:Cup** requires four. For each symbol-object pair, e.g., **<urn:Cup>** and **<urn:cup_0>**, DLU can thus find a mean path depth value and arbitrarily

[5] https://www.swi-prolog.org/pldoc/man?section=semweb-rdf11#rdf_reachable/5

```
SELECT DISTINCT ?symbol ?object ?objtype
WHERE {
    ?symbol rdf:type ?objtype .
    ?symbol scg:tag cg:semantic .

    ?object rdf:type ?objtype .
    ?object rdf:type owl:NamedIndividual .
}
```

Listing 4: `SELECT` statement to select all symbol-object pairs with the same assigned object types.

select one of the highest valued ones, thus referencing the cup with `<urn:cup_0>` but not with the fridge, see Table 1. Finally, this knowledge is asserted by introducing a `dul:isReferenceOf` to the knowledge base for further computations.

3.5. Theory and goal extraction

"Take the cup to the table" is a `soma:Command` (cf. Figure 3). In SOMA, a direction classifies a state transition, i.e., the intention of a direction is to transform a scene from one scene state, the initial scene, into another scene state, the terminal scene [40]. It is important to note that the terminal scene is not derived explicitly by DLU, it can only be observed via simulation. For each scene state, a set of schemas or theories has to be satisfied. Most directions leave the initial scene free of constraints except for ensuring the existence of several objects, e.g., a cup and a table. The focus of DLU lies in the terminal scene, as this defines which schemas or theories need to be satisfied in the final state after the action. In the example case, this is a theory of proximity: i.e., the cup has to be in the proximity of the table. It is not possible to be more specific here on linguistic grounds because "take to" does not constrain whether the cup should be on top of or below the table, or in any other relation. A possible resolution to this dilemma is human computation, as described in Section 3.6. Here, the focus is on the extraction of theories which give information on how the new state should be achieved.

The direction *"take the cup to the table"* evokes a SPG schema [18], denoted by a `soma:SourcePathGoalTheory`. In DLU, SPGs are modeled similarly to Bergen and Chang's version in their embodied construction grammar by introducing a theory with the roles trajector, source, path, goal, and means [20]. Each of the roles can be filled by a reference to an object in the scene, a property of an object, a reference to another theory, or some specific numerical value. The initial roles in the example direction, *"take the cup to the table,"* can be seen in Listing 5. The trajector is a reference to the cup in the scene and the source is the pose of that cup, extracted from the knowledge base. Note that the path is an empty list—DLU has no information about the path and so determines the cup's trajectory in subsequent steps. The goal is a reference to another term in the knowledge base: the PROXIMALTHEORY, which should be satisfied at the terminal scene, after executing the direction.

By having theories reference other theories, the tree of theory references can be traversed to filter those theories which lead towards our goal state. In the example sentence, this applies only to the SPG theory, which references the proximal theory. The proximal theory satisfies the terminal scene of the direction and has no further references.

Table 1

Candidate groundings for the symbol `<urn:Cup>` to different objects. The grounding heuristic assigns mean depth values to each pair of candidate groundings. The pair `<urn:Cup>`, `<urn:cup_0>` has the highest, and thus the most specific, value. Hence, this pair is selected.

Symbol	Object	Object type	Depth	Mean depth
`<urn:Cup>`	`<urn:cup_0>`	`dlu:Cup`	4	$\frac{4+3}{2} = 3.5$
`<urn:Cup>`	`<urn:cup_0>`	`soma:DesignedContainer`	3	
`<urn:Cup>`	`<urn:fridge_0>`	`soma:DesignedContainer`	3	3

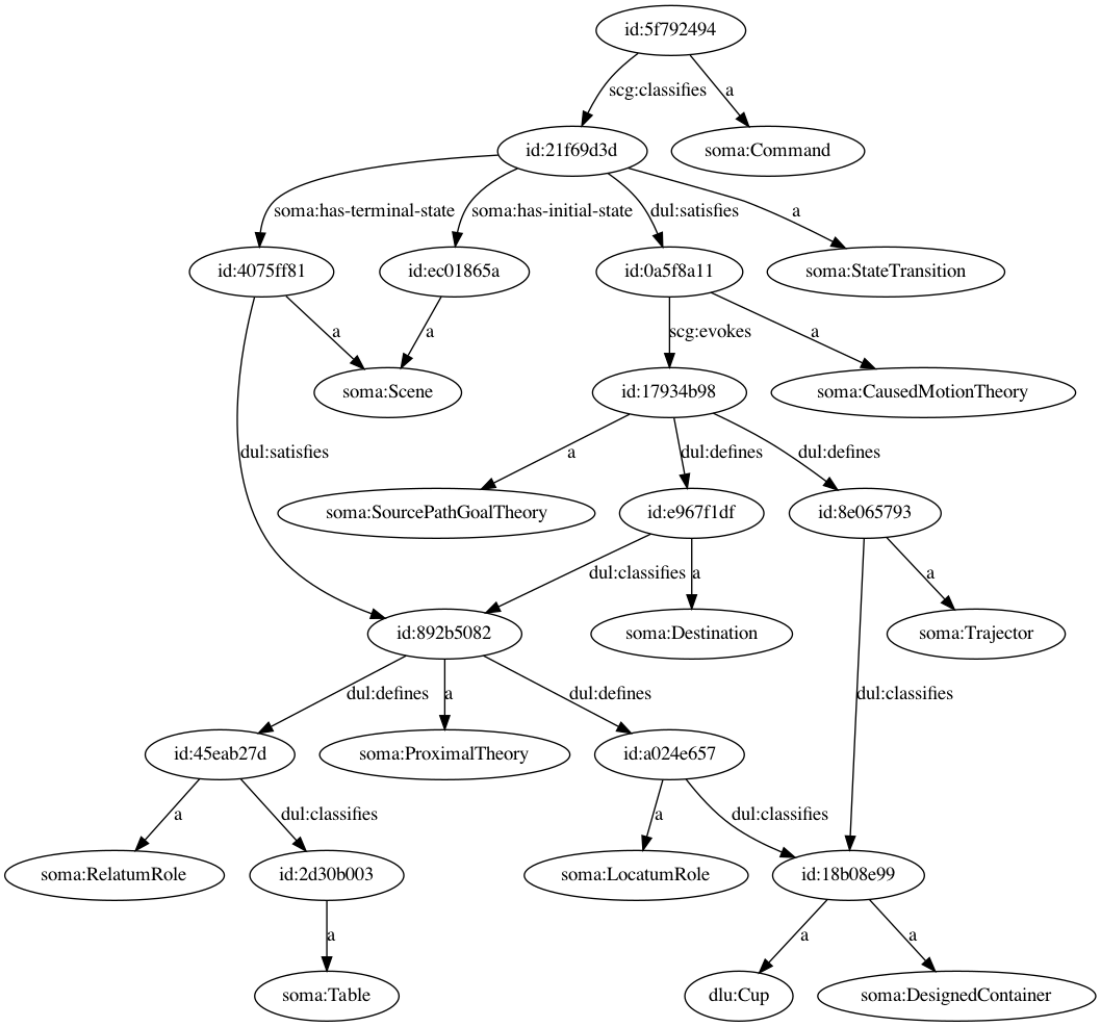

Figure 3. An extract of the semantic specification of *"take the cup to the table."* A full image can be found online at https://osf.io/dmcjq/. Each node represents a term—either an entity (starting with *id:*) or a concept. The edges between the nodes are the relations between the terms. For example, term 5f792494 is a `soma:Command` and `scg:classifies` a `soma:StateTransition`, term 21f69d3d. 21f69d3d is a `soma:StateTransition` with initial and terminal scenes, which has to satisfy a `soma:CausedMotionTheory`, and so on.

Thus, it is possible to infer that DLU only needs to process the SPG and the proximal theory itself to calculate the required actions to reach the goal of the direction.

3.6. Action Creation

There are several accounts of what "actions" are (e.g., [41]). In this work, an atomic action is the process of a trajector following a trajectory. It is possible to generate such actions from the extracted schemas in order to reach the goal of the direction. For this, a distinction between two types of theories has to be made: *resolvable* and *executable* theories. While this is mostly an implementation detail, it allows for picking theories from which actions can be built: each executable theory eventually describes a trajectory, while each resolvable theory adds information to such trajectories.

In *"take the cup to the table,"* there is one executable theory, the SPG mentioned above. Eventually, it should describe a trajectory from the original location of the cup to a pose somewhere in the proximity of the table, as is

```
<SourcePathGoalTheory urn:uuid:17934b98...
    Trajector: urn:N4d059 (cup_0)
    Source: Pose(-1.2, 0.4, 0.8, 0, -0.9, 0, -0.4)
    Path: []
    Goal: <ProximalTheory urn:uuid:892b5082...
              Locatum: urn:N4d059... (cup_0)
              Relatum: urn:N8e6c2... (table_0)>
    Means: None>
```

Listing 5: The initial roles for the SPG schema evoked by *"take the cup to the table."*

defined by the proximal theory. Finding this trajectory requires two steps: first, resolving the proximal theory to a proper pose, then using the initial pose and the resolved pose to plan a path between the two.

One can see the proximity theory as a specialization of Johnson's NEAR_FAR schema [18], as it conveys some information about nearness. However, nearness in itself is context dependent and relies on a reference frame. To solve the problem of precision in spatial closeness, DLU uses human computation [42]. Pfau et al. collected human activity data in a virtual reality (VR) kitchen environment [6] for various tasks[6]. Inside the VR environment, users could chop vegetables, move objects around, and perform other cooking related tasks. Simultaneously as the users performed tasks in the VR kitchen, their hands and the object positions were tracked, as well as collision data between individual objects. The collisions between glasses and the table surface were used to gather data where cups are in a proximity relation to a table, as the kitchen did not contain any cups. Several Gaussian mixture models (GMMs) were fit on the collision positions and the one with lowest Bayesian information criterion (BIC) was selected as the model[7] for "glasses in proximity of a table". Figure 4 depicts the raw collisions and the GMM on a normalized table surface.

Having distinct models for proximity relations between cups and tables, between lamps and chairs, between houses and trees, etc., offers a way to select the correct model that is required. To achieve this in DLU, the model of proximity relations between cups and tables is stored in the ontology as an instance of a **`soma:SoftwareImplementation`** which concretely expresses an instance of a proximal theory.[8] To store the model, it is serialized using `dill`[9] and written as a **`xsd:base64Binary`** into the knowledge base. This allows SPARQL queries to be run to select a model and, after deserialization, evaluate it. For the proof of concept of DLU, cups and glasses are treated as the same object class. Otherwise, an additional reasoning step to select models for objects similar to other objects would be required.

After selecting a candidate model for the proximity theory, it is evaluated to sample a goal pose for the cup in the example and replace the reference to the proximity theory in the SPG (cf. Listing 5) with the concrete sampled pose. As a final step, a trajectory between the start and goal pose is generated to build a path. Currently, this means a linear interpolation between the start and goal pose, but DLU provides an occupancy grid of the scene to allow for more sophisticated path planning in the future, e.g., using the A* algorithm.

3.7. Simulation

Physics engines are popular tools to simulate real life phenomena. Computational fluid dynamics (CFD) solvers such as OpenFOAM [43] or Autodesk CFD[10] focus on high accuracy but are not necessarily suited for real time simulations. In contrast, there are many physics engines which focus on speed, such as the PhysX SDK[11] or the

[6] The complete dataset can be found online: https://osf.io/rx9jg/

[7] GMMs with 1–6 Gaussians, using different covariance types. Lowest BIC was 143.82 for three Gaussians with spherical covariances. Adapted from https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_selection.html

[8] A full example can be found at https://osf.io/c63w4/

[9] The standard way to serialize Python objects, `pickle`, cannot handle anonymous functions well. Hence, `dill` (https://dill.rtfd.io) is used.

[10] Autodesk Inc., Autodesk CFD, https://www.autodesk.com/products/cfd/overview

[11] NVIDIA Corp., PhysX SDK, https://developer.nvidia.com/physx-sdk

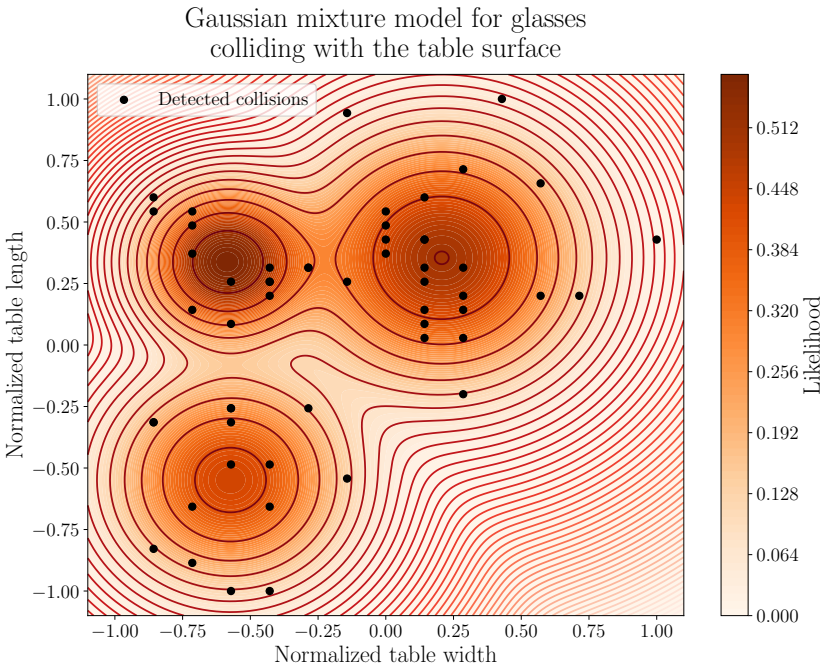

Figure 4. The positions of the collisions of glasses with the table surface in VR, together with the resulting GMM. Collisions were projected onto the same horizontal area and show the collisions from the top. The table width and length are normalized to $[-1, 1]$, so that the data can be mapped onto arbitrary (rectangular) tables. Multiple collisions at $x = -1$ were removed prior to calculating the GMM, as they formed a straight line at the edge of the table.

Bullet SDK [44]. They are embedded in a variety of applications, e.g., Robot Operating System (ROS) employs the Bullet SDK, the OpenAI Gym [45] offers bindings to MuJoCo[12], and Unity3D[13] integrates the PhysX SDK. In addition to these general purpose physics engines, there exist specialized tools, e.g, to predict folk psychological phenomena [46].

In DLU, Unity3D is used to perform naive physics simulations. It executes the actions described above in the scene and records the actual trajectories of all objects in the kitchen scene. After a simulation run, the results are evaluated visually until automatic evaluations are developed, as described in the future work in Section 5.1.

4. Discussion of System Performance and Limitations

As discussed in Section 1.1, the direction *"take the cup to the table"* is used to evaluate the DLU pipeline. The evaluation criterion is that the final simulation matches common sense expectations. In future work, DLU can be used as a platform to compare different evaluation strategies. One possible strategy is briefly discussed in Section 5.1, but in general out of the scope of this paper.

DLU is built upon a few assumptions. First, it assumes error-free natural language directions and the existence of a semantic map. Further, it expects that each direction maps to an action as defined above and can be achieved by moving objects along a trajectory in the environment. Also, it presumes that a real robot can perform that action when given a trajectory and a trajector, allowing it to leave out the robot—instead, objects "fly" around. For the model of proximity relations between glasses and tables, it is assumed that collision events are a good enough indicator of proximity. Lastly, the hypothesis is that once a sufficiently large set of functional relationships and

[12]Roboti LLC, MuJoCo, http://www.mujoco.org
[13]Unity Technologies, Unity, https://unity.com

image-schematic micro-theories is modeled in DLU, the pipeline will scale up and be able to understand more directions.

When the DLU pipeline is evaluated under these assumptions and given the above criterion, *"take the cup to the table"* results in a simulation that human observers, such as DLU's authors, would consider an appropriate execution of the task. Still, individual components of the pipeline can be observed in isolation to reveal their strengths and weaknesses. SOMA and SCG have been evaluated by their authors [3, 4].

The grounding step (Section 3.4) works for many objects in the scene, given the parser provides enough information. With insufficient or imprecise information, for example, labeling all objects as generic `soma:PhysicalObject`s will cause the grounding step to select arbitrary objects. It also does not work for complex object descriptions, e.g., "the red cup" might select a blue cup. When multiple objects are present, such as six identical cups or two fridges, the system will not be able to distinguish between these. However, as the grounding step is merely a prototype heuristic and should be replaced with a more sophisticated method in the future, it is sufficient for this work.

In the schema or theory extraction step, only those theories for which queries were written can be found. To date, these are the ones which are relevant in *"take the cup to the table,"* namely SPG, CAUSEDMOTIONTHEORY, and PROXIMALTHEORY. There is also only one model available for the proximal theory of glasses being in the proximity of tables. Under the assumption that one can exchange glasses and tables for other objects, this makes it possible to move objects around the scene and place them on top of each other. In DLU, it is shown that it is possible to store such a model with proper annotations and retrieve it to use it without enforcing specific constraints on the model. This allows sharing and reusing models across different ontologies. As path planning is not the focus of this work, DLU interpolates linearly between the start and goal pose of an object. However, since DLU provides occupancy information as well as object data such as available meshes, future versions can employ existing path planning solutions.

5. Conclusion and Future Work

This paper introduced DLU, a natural language processing pipeline which is capable of simulating natural language directions using SOMA as a common interface for individual components. On the basis of *"take the cup to the table,"* this work showed that the architecture is successful in simulating an everyday activity task.

DLU runs on a live instance at https://litmus.informatik.uni-bremen.de/dlu/[14], and all required resources to run DLU locally are compiled at the Open Science Framework (OSF)[15]. The source code, software, and hardware information used to build and run DLU are available at https://osf.io/nbxsp/. The ontologies and their metrics are available at https://osf.io/e7uck/.

The external modules, SOMA and SCG, are under active development and as they progress to become more advanced and complete, DLU benefits by being able to parse more directions and harvest better ontological descriptions. For the grounding module, other systems to replace the heuristic are currently being evaluated. Possible options are a system to ground unknown synonyms [47] and end-to-end machine learning models to ground directions such as "go to the red pole" [48, 49].

Other future work includes building models on different relations between different objects to scale up the knowledge base, e.g., SUPPORT or functional relationships such as coverage. In addition to broadening the application, it also offers to validate model selection procedures, e.g., whether exchanging a cup for a glass is an option, and if one can resort to a model of SUPPORT between two physical objects. To decide which object classes are possible replacements candidates, it is planned to use semantic similarities, e.g., word2vec [50], and affordance-based models, fueled by human computation tasks. One major step in future work will be to reiterate the last two steps: that is, if the simulation fails, to re-sample the parameters and try again. For this, an integration of *schemasim* is planned, which allows checking whether a configuration of objects in a scene satisfies a set of expectations [51], and thus to decide whether the simulation succeeded.

[14] A video demonstration is available at https://osf.io/t2mnw/, in case the live instance no longer exists.
[15] https://osf.io/nm86g/

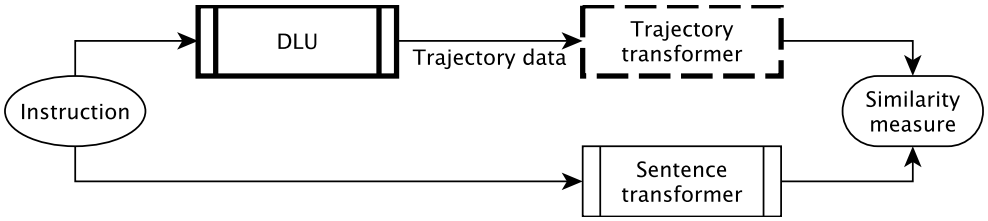

Figure 5. A possible platform to process the output of DLU with a *trajectory transformer* and to compare the result with the semantic embedding from a sentence transformer. The similarity between the outputs of the transformers would be a good measure for simulation correctness. To build a good trajectory transformer model, data from human computation will be vital in the training process. *Bold line: this work. Dashed line: future work.*

5.1. DLU as an Evaluation Platform

As introduced in Section 4, one future plan for DLU is to use it as an evaluation platform to compare different evaluation strategies. Simulating tasks is much cheaper and faster than performing tasks on a real robot platform and the modular design of DLU will allow individual evaluation of exchanged components. But a simulation alone is not enough. When the simulation places a cup inside the sink or puts it into the fridge rather than placing it on top of a table, it is important to distinguish the different outcomes: another algorithm must rate the resulting trajectory data as successful or not. Currently, to the best of the authors' knowledge, no generic evaluation criterion exists to determine a successful task execution of an arbitrary task. DLU will be of assistance to develop and test such platforms.

One possible platform the authors envision could draw from ideas in recent advances of transformers [52], and is sketched in Figure 5. A sentence transformer [53] provides a semantic embedding, a vector which encodes the semantics of a sentence. Analogously, a similar model, say a *trajectory transformer*, could be used to represent the essence of the trajectories recorded using DLU – or on a real robot. Such a trajectory transformer will be trained with a sentence transformer using data gathered by human computation. The outputs of the sentence transformer and the trajectory transformer should thus match, and could be compared using common methods in transformer training, e.g., a cosine similarity. This way, DLU could work as a platform to compare different components for language understanding, and, by replacing the approach presented here, also for comparison of evaluation strategies.

Acknowledgements

The authors wish to thank Susanne Putze, Dmitry Alexandrovsky, Nina Wenig, the reviewers Stefano Borgo, Christopher Myers, and five anonymous reviewers for their valuable feedback. Stefano Borgo provided examples to help us narrow down the scope of DLU, they are discussed in Section 1.1. The research reported in this paper has been (partially) supported by the FET-Open Project #951846 "MUHAI – Meaning and Understanding for Human-centric AI" funded by the EU Program Horizon 2020 as well as the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 "EASE – Everyday Activity Science and Engineering", University of Bremen (https://ease-crc.org/). The research was conducted in subprojects H02, P01 and P05.

References

[1] R. Porzel, V.S. Cangalovic and J.A. Bateman, Filling Constructions: Applying Construction Grammar in the Kitchen, in: *Proceedings of the 11th International Conference on Construction Grammar*, 2021, URL: https://muhai.org/images/papers/Porzel-Cangalovic-ICCG2021.pdf.

[2] J. Ruppenhofer and L. Michaelis, A Constructional Account of Genre-Based Argument Omissions, *Constructions and Frames* (2010), DOI: https://doi.org/10.1075/cf.2.2.02rup.

[3] D. Beßler, R. Porzel, M. Pomarlan, A. Vyas, S. Höffner, M. Beetz, R. Malaka and J. Bateman, Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents, in: *Proceedings of the 12th International Conference on Formal Ontology in Information Systems*, Bolzano, Italy, 2021, pp. 1–16, arXiv: https://arxiv.org/abs/2011.11972.

[4] V.S. Cangalovic, Cooking up a Grammar: Incremental Grammar Extension for Domain-Specific Parsing, Bachelor's thesis, University of Bremen, Bremen, 2018.

[5] V.S. Cangalovic, R. Porzel and J.A. Bateman, Streamlining Formal Construction Grammar, in: *Proceedings of the ICCG11 Workshop on Constructional Approaches in Formal Grammar*, 2021, URL: https://muhai.org/images/papers/Cangalovic-Porzel-ICCG2021.pdf.

[6] J. Pfau, R. Porzel, M. Pomarlan, V.S. Cangalovic, S. Grudpan, S. Höffner, J. Bateman and R. Malaka, Give MEANinGS to Robots with Kitchen Clash: A VR Human Computation Serious Game for World Knowledge Accumulation, in: *Entertainment Computing and Serious Games*, Vol. 11863, E. van der Spek, S. Göbel, E.Y.-L. Do, E. Clua and J. Baalsrud Hauge, eds, Springer International Publishing, Cham, 2019, pp. 85–96, DOI: https://doi.org/10.1007/978-3-030-34644-7_7. ISBN 978-3-030-34644-7.

[7] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, DOI: https://doi.org/10.18653/v1/N19-1423, arXiv: https://arxiv.org/abs/1810.04805.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, Language Models Are Unsupervised Multitask Learners, 2019, URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[9] G. Campagna, R. Ramesh, S. Xu, M. Fischer and M.S. Lam, Almond: The Architecture of an Open, Crowdsourced, Privacy-Preserving, Programmable Virtual Assistant, in: *Proceedings of the 26th International Conference on World Wide Web*, 2017, DOI: https://doi.org/10.1145/3038912.3052562.

[10] P. Schoutsen, Almond & Ada: Privacy-Focused Voice Assistant, *Home assistant blog* (2019), URL: https://www.home-assistant.io/blog/2019/11/20/privacy-focused-voice-assistant/.

[11] A. Chang, W. Monroe, M. Savva, C. Potts and C.D. Manning, Text to 3D Scene Generation with Rich Lexical Grounding, in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, Beijing, China, 2015, pp. 53–62, DOI: https://doi.org/10.3115/v1/P15-1006.

[12] B. Coyne and R. Sproat, WordsEye: An Automatic Text-to-Scene Conversion System, in: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001, DOI: https://doi.org/10.1145/383259.383316.

[13] B. Andrus and N. Fulda, Immersive Gameplay via Improved Natural Language Understanding, in: *International Conference on the Foundations of Digital Games*, ACM, Bugibba Malta, 2020, pp. 1–4, DOI: https://doi.org/10.1145/3402942.3403024. ISBN 978-1-4503-8807-8.

[14] S. Harnad, The Symbol Grounding Problem, *Physica D: Nonlinear Phenomena* **42**(1–3) (1990), 335–346, DOI: https://doi.org/10.1016/0167-2789(90)90087-6.

[15] L. Steels, The Symbol Grounding Problem Has Been Solved, so What's Next?, in: *Symbols and Embodiment: Debates on Meaning and Cognition*, M. de Vega, A. Glenberg and A. Graesser, eds, Oxford University Press, 2008, DOI: https://doi.org/10.1093/acprof:oso/9780199217274.003.0012. ISBN 978-0-19-921727-4.

[16] C.J. Fillmore, The Mechanisms of "Construction Grammar", in: *Proceedings of the Fourteenth Annual Meeting of the Berkeley Linguistics Society*, 1988, DOI: https://doi.org/10.3765/bls.v14i0.1794.

[17] L. Shapiro, *Embodied Cognition*, Routledge, 2011, DOI: https://doi.org/10.4324/9781315180380.

[18] M. Johnson, *The Body in the Mind*, The University of Chicago Press, 1987, OCLC: https://www.worldcat.org/oclc/1178931442.

[19] M.M. Hedblom, *Image Schemas and Concept Invention: Cognitive, Logical, and Linguistic Investigations*, Cognitive Technologies, Springer Computer Science, 2020, DOI: https://doi.org/10.1007/978-3-030-47329-7.

[20] B.K. Bergen and N. Chang, Embodied Construction Grammar in Simulation-Based Language Understanding, in: *Constructional Approaches to Language*, Vol. 3, J.-O. Östman and M. Fried, eds, John Benjamins Publishing Company, Amsterdam, 2005, pp. 147–190, DOI: https://doi.org/10.1075/cal.3.08ber. ISBN 978-90-272-1823-0 978-1-58811-579-9 978-90-272-1826-1 978-90-272-9470-8.

[21] A. Olivares-Alarcos, D. Beßler, A. Khamis, P. Gonçalves, M. Habib, J. Bermejo, M. Barreto, M. Diab, J. Rosell, J. Quintas, J. Olszewska, H. Nakawala, E. Pignaton de Freitas, A. Gyrard, S. Borgo, G. Alenyà, M. Beetz and H. Li, A Review and Comparison of Ontology-Based Approaches to Robot Autonomy, *The Knowledge Engineering Review* (2019), DOI: https://doi.org/10.1017/S0269888919000237.

[22] R. Patel, M. Hedelind and P. Lozan-Villegas, Enabling Robots in Small-Part Assembly Lines: The "ROSETTA Approach" – an Industrial Perspective, in: *ROBOTIK*, 2012, URL: https://ieeexplore.ieee.org/document/6309522.

[23] M. Stenmark, J. Malec, K. Nilsson and A. Robertsson, On Distributed Knowledge Bases for Robotized Small-Batch Assembly, *IEEE Transactions on Automation Science and Engineering* (2015), DOI: https://doi.org/10.1109/TASE.2015.2408264.

[24] S. Balakirsky, Z. Kootbally, T. Kramer, A. Pietromartire, C. Schlenoff and S. Gupta, Knowledge Driven Robotics for Kitting Applications, *Robotics and Autonomous Systems* (2013), DOI: https://doi.org/10.1016/j.robot.2013.04.006.

[25] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert and A. Knoll, Intuitive Instruction of Industrial Robots: Semantic Process Descriptions for Small Lot Production, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, DOI: https://doi.org/10.1109/IROS.2016.7759358.

[26] A.S. Polydoros, B. Großmann, F. Rovida, L. Nalpantidis and V. Krüger, Accurate and Versatile Automation of Industrial Kitting Operations with SkiROS, in: *Towards Autonomous Robotic Systems – 17th Annual Conference (TAROS)*, 2016, DOI: https://doi.org/10.1007/978-3-319-40379-3_26.

[27] A. Eberhart, C. Shimizu, C. Stevens, P. Hitzler, C.W. Myers and B. Maruyama, A Domain Ontology for Task Instructions, in: *Knowledge Graphs and Semantic Web*, Vol. 1232, B. Villazón-Terrazas, F. Ortiz-Rodríguez, S.M. Tiwari and S.K. Shandilya, eds, Springer International Publishing, Cham, 2020, pp. 1–13, DOI: https://doi.org/10.1007/978-3-030-65384-2_1. ISBN 978-3-030-65383-5 978-3-030-65384-2.

[28] M. Tenorth and M. Beetz, Representations for Robot Knowledge in the KnowRob Framework, *Artificial Intelligence* **247** (2017), 151–169, DOI: https://doi.org/10.1016/j.artint.2015.05.010.
[29] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer and E. Miguelanez, An IEEE Standard Ontology for Robotics and Automation, in: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012, URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=911827.
[30] E. Prestes, J.L. Carbonera, S.R. Fiorini, V.A.M. Jorge, M. Abel, R. Madhavan, A. Locoro, P. Goncalves, M.E. Barreto, M. Habib, A. Chibani, S. Gérard, Y. Amirat and C. Schlenoff, Towards a Core Ontology for Robotics and Automation, *Robotics and Autonomous Systems* (2013), DOI: https://doi.org/10.1016/j.robot.2013.04.005.
[31] S.R. Fiorini, J.L. Carbonera, P. Gonçalves, V.A.M. Jorge, V.F. Rey, T. Haidegger, M. Abel, S.A. Redfield, S. Balakirsky, V. Ragavan, H. Li, C. Schlenoff and E. Prestes, Extensions to the Core Ontology for Robotics and Automation, *Robotics and Computer-Integrated Manufacturing* (2015), DOI: https://doi.org/10.1016/j.rcim.2014.08.004.
[32] J.R. Kirk, A. Mininger and J.E. Laird, A Demonstration of Interactive Task Learning, in: *IJCAI*, 2016, URL: https://soar.eecs.umich.edu/pubs/kirk_ijcai16.pdf.
[33] K.A. Gluck and J.E. Laird (eds), *Interactive Task Learning: Humans, Robots, and Agents Acquiring New Tasks through Natural Interactions*, The MIT Press, 2019, DOI: https://doi.org/10.7551/mitpress/11956.001.0001. ISBN 978-0-262-34942-0.
[34] A. Gangemi and V. Presutti, Ontology Design Patterns, in: *Handbook on Ontologies*, Springer, 2009, DOI: https://doi.org/10.1007/978-3-540-92673-3_10.
[35] C. Masolo, S. Borgo, A. Gangemi, N. Guarino and A. Oltramari, WonderWeb Deliverable D18 Ontology Library, Technical Report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web, 2003, URL: http://wonderweb.man.ac.uk/deliverables/documents/D18.pdf.
[36] S. Albrecht, T. Wiemann, M. Günther and J. Hertzberg, Matching CAD Object Models in Semantic Mapping, in: *Workshop Semantic Perception, Mapping and Exploration*, Shanghai, China, 2011, URL: https://kbs.informatik.uos.de/files/pdfs/icra2011_albrecht.pdf.
[37] A. Pronobis and P. Jensfelt, Large-Scale Semantic Mapping and Reasoning with Heterogeneous Modalities, in: *2012 IEEE International Conference on Robotics and Automation*, IEEE, St Paul, MN, USA, 2012, pp. 3515–3522, DOI: https://doi.org/10.1109/ICRA.2012.6224637. ISBN 978-1-4673-1405-3 978-1-4673-1403-9 978-1-4673-1578-4 978-1-4673-1404-6.
[38] R. Capobianco, J. Serafin, J. Dichtl, G. Grisetti, L. Iocchi and D. Nardi, A Proposal for Semantic Map Representation and Evaluation, in: *2015 European Conference on Mobile Robots (ECMR)*, IEEE, Lincoln, United Kingdom, 2015, pp. 1–6, DOI: https://doi.org/10.1109/ECMR.2015.7324198. ISBN 978-1-4673-9163-4.
[39] H. Deeken, T. Wiemann, K. Lingemann and J. Hertzberg, SEMAP - a Semantic Environment Mapping Framework, in: *2015 European Conference on Mobile Robots (ECMR)*, IEEE, Lincoln, United Kingdom, 2015, pp. 1–6, DOI: https://doi.org/10.1109/ECMR.2015.7324176. ISBN 978-1-4673-9163-4.
[40] R. Porzel and V.S. Cangalovic, What Say You: An Ontological Representation of Imperative Meaning for Human-Robot Interaction, in: *Proceedings of the JOWO – Ontology Workshops*, Bolzano, Italy, 2020, URL: http://ceur-ws.org/Vol-2708/robontics4.pdf.
[41] J.R. Flanagan, M.C. Bowman and R.S. Johansson, Control Strategies in Object Manipulation Tasks, *Current Opinion in Neurobiology* (2006), DOI: https://doi.org/10.1016/j.conb.2006.10.005.
[42] L. von Ahn, Human Computation, PhD thesis, Carnegie Mellon University, 2005, URL: https://www.proquest.com/docview/305006687.
[43] H.G. Weller, G. Tabor, H. Jasak and C. Fureby, A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques, *Computers in Physics* (1998), DOI: https://doi.org/10.1063/1.168744.
[44] E. Coumans and Y. Bai, PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning, 2016, URL: https://pybullet.org.
[45] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, OpenAI Gym (2016), arXiv: https://arxiv.org/abs/1606.01540.
[46] P.W. Battaglia, J.B. Hamrick and J.B. Tenenbaum, Simulation as an Engine of Physical Scene Understanding, *Proceedings of the National Academy of Sciences* **110**(45) (2013), 18327–18332, DOI: https://doi.org/10.1073/pnas.1306572110.
[47] O. Roesler, A. Aly, T. Taniguchi and Y. Hayashi, Evaluation of Word Representations in Grounding Natural Language Instructions Through Computational Human-Robot Interaction, in: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, Daegu, Korea (South), 2019, pp. 307–316, DOI: https://doi.org/10.1109/HRI.2019.8673121. ISBN 978-1-5386-8555-6.
[48] V. Kurenkov, B. Maksudov and A. Khan, Task-Oriented Language Grounding for Language Input with Multiple Sub-Goals of Non-Linear Order, *CoRR* (2019), arXiv: https://arxiv.org/abs/1910.12354.
[49] D.S. Chaplot, K.M. Sathyendra, R.K. Pasumarthi, D. Rajagopal and R. Salakhutdinov, Gated-Attention Architectures for Task-Oriented Language Grounding, in: *AAAI Conference on Artificial Intelligence*, 2018, arXiv: https://arxiv.org/abs/1706.07230.
[50] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient Estimation of Word Representations in Vector Space, Technical Report, Google Inc., 2013, arXiv: https://arxiv.org/abs/1301.3781.
[51] M. Pomarlan and J. Bateman, Embodied Functional Relations: A Formal Account Combining Abstract Logical Theory with Grounding in Simulation, in: *Proceedings of the 11th International Conference on Formal Ontology in Information Systems*, FOIS, 2020, DOI: https://doi.org/10.3233/FAIA200668.
[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser and I. Polosukhin, Attention Is All You Need, in: *Advances in Neural Information Processing Systems*, Vol. 30, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, Curran Associates, Inc., 2017, URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.
[53] N. Reimers and I. Gurevych, Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks (2019), arXiv: https://arxiv.org/abs/1908.10084.