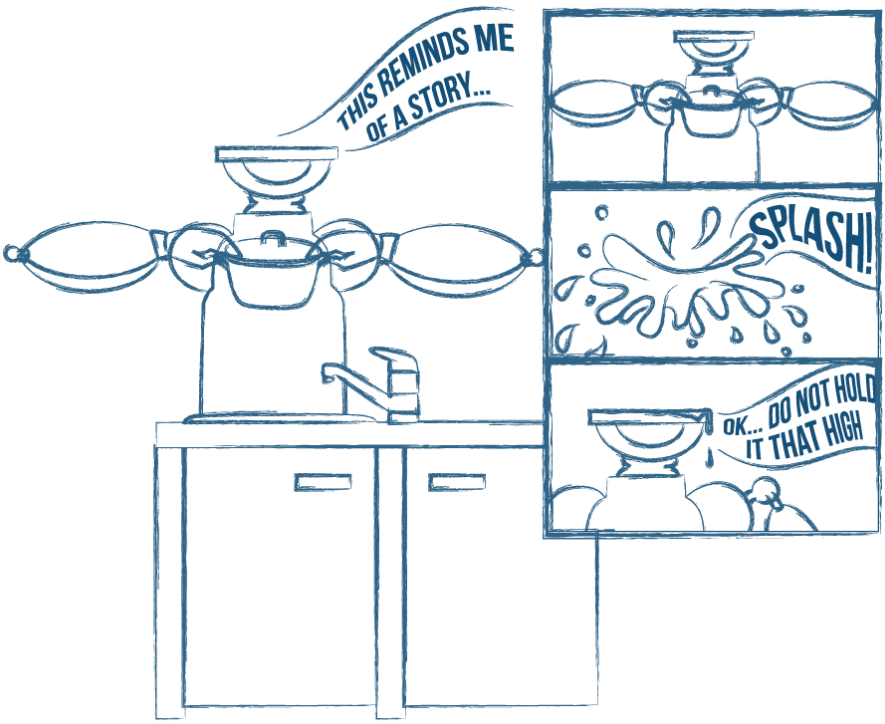
NEEM Handbook

The EASE Researchers

CRC Everyday Activity Science and Engineering (EASE)
University Bremen, Am Fallturm 1, 28359 Bremen
ai-office@cs.uni-bremen.de

Summary. The Collaborative Research Center EASE is an interdisciplinary research initiative at the University of Bremen that attempts to advance our understanding of how human-scale manipulation tasks can be mastered by robotic agents. The challenge is that the same task needs to be executed by the robot in different ways depending on, for example, what tools are available, and how the environment is shaped. The key to solve this issue is *generalization*. However, the robot needs to know more then what step it needs to execute next – it further needs to decide on how the next step is carried out through motions of its body, and interactions with its environment. In this document, we will describe how these types of information are represented in the EASE system, how such data-sets are acquired, and how they are stored, maintained, and curated using a centralized web-service. The goal of this effort is to establish representations and infrastructure for a shared experience storage with annotated data-sets of agents performing everyday activities, and to use these data-sets as ground truth data to find generalizations that do not abstract away from movements, and naive physics.



Contents

1

Introduction

D. BESSLER, S. KORALEWSKI

This document, referred to as the "NEEM Handbook" hereafter, describes the EASE system for episodic memories of everyday activities. The NEEM Handbook will be updated along the progress in the CRC EASE . It is thought to provide EASE researchers with compact but still comprehensive information about what information is contained in NEEMs , and how it is represented.

Narrative Enabled Episodic Memories

When somebody talks about the deciding goal in the last soccer world championship many of us can "replay" the episode in our "mind's eye". Those episodic memories can be seen as abstract descriptions which allow us to recall detailed pieces of information from any experienced activity. Having those detailed memories, we can use them to learn general knowledge or map similar memories to unknown situations, so we know how to behave in the given situation.

EASE integrates episodic memories deeply into the knowledge acquisition, representation, and processing system. For every activity the agent performs, observes, prospects and reads about, it creates an episode and stores it in its memory. An episode is best understood as a video recording that the agent makes of the ongoing activity. In addition, those videos are enriched with a very detailed story about the actions, motions, their purposes, effects and the agent's sensor information during the activity.

We define the episodic memories created by our system narrative-enabled episodic memories (NEEMs). A NEEM consists of the *NEEM experience* and the *NEEM narrative*. The NEEM experience captures low-level data such as the agent's sensor information, e.g. images and forces, and records of poses of the agent and its detected objects. NEEM experiences are linked to NEEM narratives, which are stories of the

episode described symbolically. These narratives contain information regarding the tasks, the context, intended goals, observed effects, etc. The NEEM-experience and NEEM-narrative combined are so rich of information that the agent can replay an episode to experience the seen activity anytime again.

NEEMs are representations of experiences acquired through experimentation, reading, observing, mental simulation, etc. The main goal is to establish a common vocabulary used to annotate experience data across different tasks, scientific disciplines, and modalities of acquisition, and to define models for the representation of experience data. The vocabulary is not just a set of atomic labels, but each label has a formal definition in an ontology. These definitions are done such that a set of *competency questions* about an activitiy can be answered by a knowledge base that is equipped with the ontology and a collection of NEEMs .

The NEEM model is formally defined in form of an OWL ontology which is based on the DOLCE+DnS Ultralite (DUL) upper-level ontology [1]. DUL is a carefully designed ontology that seeks to model general categories underlying human cognition without making any discipline-specific assumptions. Our extensions of DUL mainly focus on characterizing different aspects of activities that were not considered in much detail in DUL, but are relevant for the autonomous robotics scope. These extensions are part of an ontology that we have called SOMA [1]. A NEEM is made of several patterns defined either in DUL or in SOMA .

While it is possible to create the representations listed in this document through a custom exporter, it is not advised to do so. Instead, it is advised to interface with the KnowRob knowledge base [2]. KnowRob provides an interface based on predicate logics that allows to interact with NEEMs . The language is a collection of predicates that can be called by users to ask certain types of competency questions covering different aspects of activitiy, or to add labels and relationships in the NEEM-narrative . We will provide example expressions in this document that highlight how the knowledge base can be used to interact with NEEMs .

1.1 Scope

Our work aims to provide knowledge modelling for robotic manipulation and autonomous robot control. Such knowledge must include aspects of interaction forces and motion characteristics of objects participating in an action, since it is these physical and geometric considerations that are crucial to determining whether, or to what degree, an action is successful. Ideally, the knowledge we formalize in this collection of ontologies will allow a single, general control program to adapt itself, via reasoning, and generate adequate behavior in a large variety of settings, for different tasks and participating objects.

[1] `https://ease-crc.github.io/soma`

[2] `https://github.com/knowrob/knowrob`

The kinds of knowledge a robot needs for competent performance of its tasks are varied. Usually, knowledge modelling in robotics and AI has focused on a symbolic level, of actions treated as black boxes that relate to a larger plan by means of their preconditions and effects. Actions are also very underspecified when described in spoken commands. This abstract level of description however is insufficient; the physical details of the actions matter. For example, the angle and speed with which a pitcher is moved, and the amount of liquid in it, determines whether there will be spillage. A robot needs to choose appropriate parameters for its actions, and infer these parameters when they are left unspecified in a command.

Such inference requires the robotic agent to be equipped with common-sense and intuitive physics knowledge, as well as an abstract task and object model, and knowledge of how to apply these models in a given situation. SOMA is an attempt to support each of these requirements. A brief list of some of the over-arching competency questions follows.

- *How are actions conceptualized?* What is an Action, how does it relate to other concepts an Agent might have about the world? What is the purpose of an Action?
- *What is the structure of an Action?* How do several actions make up another? What objects participate in an action and with what roles?
- *How are qualitative and quantitative features of the world represented?* What is the parameter set of an action? What regions can values for these parameters occupy? What is a good parameterization and how can one be found?
- *How are objects conceptualized?* What roles can an object play? What actions can it take part in? What kinds of objects are necessary for an action?
- *How is an Action recorded and described?* What is the relevant data to capture how an action unfolded? What are the relevant pieces of contextual information for describing an action that has actually occurred? What was the outcome of the action, in particular, to what extent did it match the goal?

1.2 Outline

The purpose of this document is to provide an overview about version 1.0 of NEEMs . This is, first of all, how NEEMs are represented using ontological categories and relationships, and time series data. The purpose of this document is not to provide a full overview about the ontological modelling underlying NEEMs , but rather to concentrate on aspects that are directly relevant when a NEEM is created. These are the representations that are used in the NEEM-narrative (Section 3) and NEEM-experience (Section 4). In addition, NEEMs are situated in an environment, and acquired by some agent executing a task by interacting with its environment (the NEEM-background). However, the agent and the environment may be involved in

many different NEEMs such that their representation is seperated from the representation of the NEEM-narrative and NEEM-experience (Section 2). Second, this document provides an overview about software tools that were developed to support the acquisition of NEEMs in different modalities such as a tool that can be hooked into a robot control system to auto-generate NEEMs (Section 5). Finally, the document provides information about the NEEM-hub , which is an infrastructure software for the management and curation of NEEMs (Section 6).

2

NEEM-Background

D. BESSLER, M. POMARLAN, A. VYAS

The NEEM-background represents the (physical) context of NEEMs . More concretely, the NEEM-background represents the environment where events took place, and the agents that are involved. These are representations of physical objects, their parts, properties, and relationships between them.

Each NEEM must have exactly one associated NEEM-background . This is important as only the objects and their properties represented in the NEEM-background may be involved in events that occur in NEEMs . Consider, for example, a robot fetching a cup in a kitchen environment to prepare a coffee. The cup would be part of the NEEM-background while the fetching event carried out by the robot would be represented in the NEEM-narrative (Chapter 3).

The way how a task can be solved best depends on what is available in the environment. The suitability of an object to be used to perform a certain task is often derived from the class of objects it belongs to, e.g., that a knife can be used for cutting. The NEEM model defines a set of more general object classes such as *agent* and *artifact* (Section 2.1). These are used to classify each object represented in the NEEM-background . The usability of an object is, however, ultimately grounded in its properties, and, e.g., a dulled knife may proof to be unusable to perform a cutting task. It is thus also relevant to characterize object properties as they correlate with how an agent may solve its task. Consequently, we treat types of object properties as classes organized in a taxonomy (Section 2.2).

NEEMs may characterize different aspects of the environment depending on what information is accessible when the NEEM is acquired. We organize different characteristics in so called *views* (Section 2.3). Each view has its own set of types and relationships to represent the environment from a specific viewpoint such as appearance or kinematics.

NEEMs are heterogenous representations that may include additional data files. These are, first of all, time series data bases that are annotated by the NEEM-narrative . In adddition, some widely used data formats for the representation of objects are supprted (Section 2.4). Such data files may be stored within the NEEM-background , associated to objects it represents, and used to enrich knowledge about the environment.

2.1 Types of Objects

Objects and agents that appear in an environment are classified as *physical objects*. Physical objects are exactly the objects you can point on, as they have a location in space.

The most common physical objects in non-natural environments are *artifacts*. An artifact is an item that has certain structure, often to serve a particular purpose such as to use it in a certain way, or to enjoy looking at it in case of, e.g., an art piece. Artifacts that were created with a purpose in mind are called *designed artifacts*. Most objects in human-made environments belong to this category. Note that, e.g., a *container* is not a designed artifact, as also objects that were not designed as such may serve as container. Consequently, the class *designed container* is used for the objects that were designed to be used as a container. Other examples of designed artifacts are *tools* and *appliances* designed for specific tasks or agents, and *components* designed to fit together to form a larger whole.

Another category of objects are *physical bodies*. Most commonly one would use this category for *substances* that appear in the environment such as a blob of dough, or the coffee inside of a cup. However, it is more appropiate to classify the substance as *designed substance* in case it was created with a purpose in mind which is, e.g., the case for the dough that is made according to a recipe, and supposed to be eaten after being baked.

Agents that appear in a NEEM are classified as *physical agents*. The difference to other types of objects is that agents have intentions, execute actions, and attempt to achieve goals, e.g., by following a plan and moving their body in a way to generate interactions with the environment to cause intended effects. Each agent is composed of *body parts* organized in a skeletal structure. Interactions with the environment are carried out through *effectors* such as arms, legs, or hands. Effectors that are used for grasping are called *prehensile effectors*.

there is no class body part

The last top-level category in our object taxonomy is *physical place*. Places are objects with a specific location such as the surface of a table, or the campus of the University. Each NEEM refers at least to the place where it was acquired, which is usually a room in a building with objects that can be used to perform certain everyday tasks.

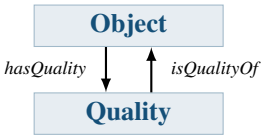
show object taxonomy Figure

provide details about e.g. how many labels are available?

2.2 Properties of Objects

Qualities are the properties of an object that are not part of it, but cannot exist without it. This is, for example, the quality of having a shape – a quality inherited by all physical objects. Another example is the quality of a floor being slippery. A robot navigating on such a floor could use this knowledge to avoid, for example, spillage when moving on the floor with a coffee-filled cup. The quality concept does not directly encode the value of the object property, but only focusses on characteristics of the property itself. This is mainly useful in cases where individual aspects of an entity are considered in the domain of discourse.

Intent	To represent the qualities of an object.
Competency Questions	*What qualities does this object have? What objects have this quality?*
Defined in	DUL.owl

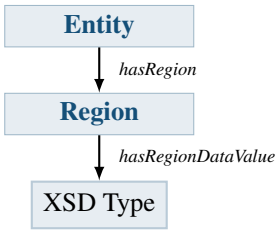


Expression	Meaning
has_quality(x,y)	*y* is a quality of *x*

Each object property has one value at a time. The value of an object property is called *region*. The value itself is an element, or a sub-region in some dimensional space such as *time interval* or *space region*. A region may be a finite set of discrete labels, allowing for "qualitative" descriptions, but more often a region is some dimensional space allowing "quantitative" descriptions. A Region may contain a single point, in cases where the value of a property is known precisely. Note that the domain of the relation *hasRegion* is not *Quality* but *Entity*. This is to allow assigning regions to entities without explicating the quality as a concept (quality-as-relation).

Seba: maybe some more example would be nice. Or refering to the example with the slippery floor.

Intent	To represent values of attributes of things.
Competency Questions	*What is the value for the attribute of that entity? Which entities have a certain value on that parameter/attribute/feature?*
Defined in	DUL.owl



Expression	Meaning
has_region(x,y)	*y* is a region of *x*
has_data_value(x,y)	*y* is a data value of *x*

2.3 Views on Objects

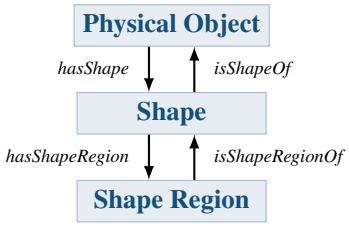
The NEEM-background may represent several different *views* on the same object highlighting different characteristics that are fused in the NEEM-background to form a more complete representation of the environment. Each view has its own vocabulary to describe objects including view-specific types of objects, qualities, and relations, and has a distinct set of competency questions that may be answered in case a NEEM represents the view. A NEEM may not represent each supported view, however, it is recommended to represent as many as possible.

TODO: Mihai: explain how EASE researcher may represent appearance of objects

2.3.1 Appearance

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Intent	To represent the shape of physical objects.
Competency Questions	*What is the shape of this object? Which objects have this shape?*
Defined in	SOMA.owl

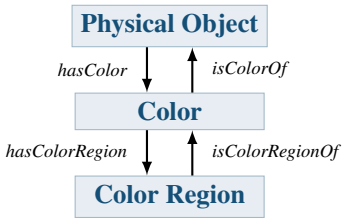


Expression	Meaning
object_dimensions(x,d,w,h)	d, w, h are the depth,width,height of the bounding box of x
object_shape(x,s,o)	s is the data of a shape of x located at o

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

TODO: write about how appearance of objects is represented

Intent	To represent the color of physical objects.
Competency Questions	*What is the color of this object? Which objects have this color?*
Defined in	SOMA.owl



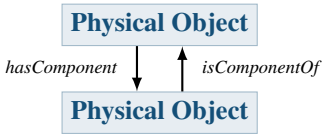
Expression	Meaning
object_color_rgb(x,[r,g,b])	r,g,b is the RGB color data of x

2.3.2 Components

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

TODO: write about how components of objectsare represented

Intent	To represent proper parthood of objects.
Competency Questions	*What is this object component of? What are the components of this object?*
Defined in	DUL.owl



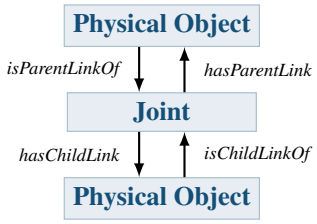
Expression	Meaning
has_component(x,y)	y is a proper part of x

2.3.3 Kinematics

Kinematics, also often referred to as *geometry of motion*, describes how objects may move without considering the influence of forces. The kinematic state of an object is represented using a set of object properties such as its position, velocity and acceleration. However, objects often can not move freely but are constrained in their movement with respect to some reference object. This is, for example, the case for walls without doors preventing movement from one room to another, or for two objects that

are attached to each other via a *joint* and thus restricting movement relative to each other (kinematic coupling). Kinematically coupled objects are often part of a bigger hierarchical structure, and one of the linked objects, the parent link of the joint, is the one closer to the root of the structure then the child link of the joint.

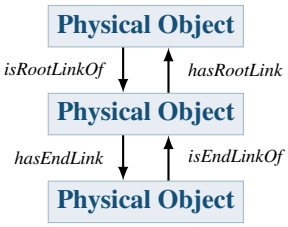
Intent	To represent how parts are attached to each other via a joint.
Competency Questions	*Is there a joint between these objects? What objects are connected by this joint?*
Defined in	SOMA.owl



Expression	Meaning
xxx_xxxxx(x,y)	**todo**

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

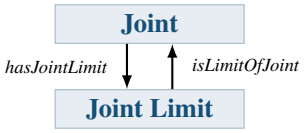
Intent	To represent objects as chains of parts.
Competency Questions	*What is the base link of this object? What are the end links of this object?*
Defined in	SOMA.owl



Expression	Meaning
xxx_xxxxx(x,y)	**todo**

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

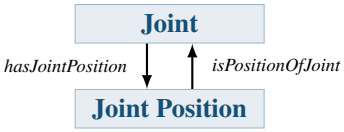
Intent	To represent the hard limits of a joint.
Competency Questions	*How far can this joint move into some direction?*
Defined in	SOMA.owl



Expression	Meaning
xxx_xxxxx(x,y)	**todo**

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Intent	To represent the position of a joint.
Competency Questions	*What is the position of this joint?*
Defined in	SOMA.owl



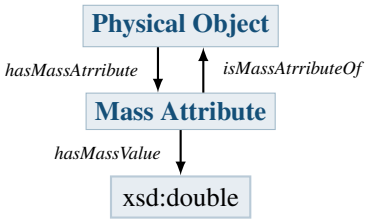
Expression	Meaning
xxx_xxxxx(x,y)	**todo**

2.3.4 Dynamics

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

TODO: write about how physical attributes of objects are represented

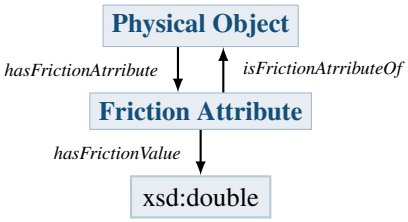
Intent	To represent the quantity of matter which a body contains.
Competency Questions	*What is the mass of this object?*
Defined in	SOMA.owl



Expression	Meaning
xxx_xxxxx(x,y)	**todo**

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Intent	To represent the resistance that one surface or object encounters when moving over another.
Competency Questions	*What is the friction of this object?*
Defined in	SOMA.owl



Expression	Meaning
xxx_xxxxx(x,y)	**todo**

2.4 Data Formats

Representations in the NEEM-background may be enriched through additional data files. Data files are stored with the NEEM-background , and loaded by the EASE knowledge base when a NEEM is activated. They may encode information that can be directly represented in the NEEM , however, it is not necessary in such a case to duplicate the information. A data file may be loaded at runtime, and used by the knowledge base in combination with other representations to answer questions about an activity.

2.4.1 URDF

The Unified Robot Description Format (URDF) is widely used for the representation of kinematics in robotics. That is how objects are organized in a skeletal structure of links and joints, and how links may move relative to each other in case of being connected via a flexible joint. URDF was designed to represent robot kinematics. It is, however, also often used for other types of objects with movable parts, for example to represent how a door is attached to a shelve via a joint, and what limits the joint has, but can also be used to represent completely static environments (via fixed joints). URDF further allows to represent a set of properties for links and joints, such as what the mass of a link is, or what the hard and soft limits of a joint are.

URDF organizes objects and their parts in a common coordinate system, and represents an initial configuration of all links and joints. The origin of this coordinate system is often called *world* or *map* frame. Each object has an associated frame in this coordinate system with a position relative to the parent frame in the skeletal structure. Frame names are further used to identify entities in the knowledge base, and logged position data that corresponds to objects described in the URDF file. To make this connection, it is important that each representation refers to the same frame names, in the same coordinate system.

more explicit: what are the properties in OWL to make the link?

From the point of view of URDF, the world is only made of links and joints. Joints are further classified based on how they operate, and have different sets of parameters quantifying their kinematics depending on their type. It is, however, not possible to represent that links belong to a certain category, or that a chain of links forms a component of some type. However, we can use the information encoded in URDF files to enrich NEEM-background representations, and on the other hand, use the NEEM-background to provide classifications for links in URDF files.

Links in URDF files may have multiple associated shapes. Two different shape types are distinguished: collision and visualization shapes. Each link has usually one shape of each kind. Shapes are either represented as geometrical primitives such as spheres or boxes, or refer to an external mesh file in which case this mesh file needs to be stored as an additional data source in the NEEM-background (next Section).

2.4.2 DAE

The preferred format for meshes is Collada (DAE). The reason primarly being that it is widely supported by modeling tools and rendering engines. If possible, the mesh should be accompanied by high-resolution textures in PNG or JPG format. The more realistic a mesh is modeled, the more immersive the experience of humans interacting with it in VR, and the better the perception models that can be trained by images generated by placing the object in a virtual scene.

what is the optimal size of meshes?

3

NEEM-Narrative

D. BESSLER, M. POMARLAN, A. VYAS

The narrative part of NEEMs represents a "story" of what has happened. But the story is more detailed then it would be when told from one human to another as it includes details about movements and interactions that would usually not be spelled out in a human conversation. The reason to include this type of information is that it is extremely difficult to generalize robotic behaviour, and in particular how the robot needs to move its body to accomplish its goal with varying context such as different environemnts. The vision is that a library of *contextualized* motions and interactions will help to uncover models underlying everyday activities.

The story represented in the NEEM-narrative provides context for the NEEM-experience acquired during an activitiy. This is, first of all, that labels are assigned to the time intervals where something relevant has happened such as the execution of a task, the interaction between objects, or the occurrence of a motion. Labels correspond to classes that are organized in a taxonomy (Section 3.1). This allows to learn models at different levels of abstraction, e.g., for more general or specific variants of a task. The NEEM-narrative further represents relationships between instances of these classes, such as that an object plays a role during an event, or that a movement has happened as part of executing an action (Section 3.2).

3.1 Taxonomy

Here in the following section, we will discuss taxonomy for the concepts used to describe NEEM-narrative part. A taxonomy provides the hierarchical relationships among various concepts and defines the terminology for each concept. One way to classify an entity in ontology is through taxonomies, which concentrates on *what there actually is?* as compared to conceptual classification which is more about *how*

an entity is interpreted?. For instance, the concepts *spoon* and *knife* are subclasses of the concept *cutlery*.

3.1.1 Event

An event can be described as any physical, social or mental process that occurs or happens. In DUL upper level ontology, an event is considered as *perdurant* entity, which unfolds over time and can have part of it's presence during any given snapshot of time. All though event taxonomy is not the scope of this section, it is important to clear some of the concepts related to event. Such as *what is an event type? Which are the sub-concepts of an event considered for the scope of NEEM-narrative ?*. An event has mainly three sub-concepts defined in SOMA which are Accident, Action and Process.

When causes are considered unknown and/or irrelevant for an event then it falls under accident category. It is worth to note distinction between everyday uses of accident definition which concentrates on a causal structure and responsibility for an event, such as who made a mistake bringing about the event; in "traffic accident", where we want to know who is responsible. Such an event does NOT fall under the definition of Accident here. An example of Accident would be a fair coin landing Heads: the causal chain for why this exact face landed is not important, all that matters is the brute fact that the coin landed Heads.

An Action can be defined as an event where at least one agent participates, such that this agent performs a dedicated task, defined by a plan or workflow, which it executes through the Action. Here workflow refers to a plan that defines role(s), task(s), and specific structure of tasks that needs to be executed.

A Process is an Event for which no such commitments for involvement of an agent have been made. For the scope SOMA ontology, we put emphases on process as a top-level class for events with no agentive participant.

An event type describes how an event should be interpreted, executed, expected, and seen which is "definedIn" according to some description [3]. It is broadly used to classify an event, *ProcessType* and *Task* are some of the *EventType* sub-classes. An Action is an Event which is conceptually classified by a Task, similarly Process is an Event which is classified by ProcessTypes.

3.1.2 Task

Task is an *EventType* that classifies an action to be executed. The actions which needs to be taken in order to execute the task are organized according to a Plan, which is however not the same as defining a task. For example task of cleaning a table

[3] `http://www.ontologydesignpatterns.org/ont/dul/DUL.owl`

that can be defined by a plan which is executed by taking several actions. Where as simpler task such as turning left or right can be executed by taking single action. Task can be further classified into three main categories, *PhysicalTask*, *MentalTask* and *CommunicationTask*.

A task which requires to execute an action through which an agent has to manipulate representations stored in its own cognition can be categorized as *MentalTask*. Dreaming, imagining, prospecting, reasoning, retrospecting falls under mental task.

PhysicalTask requires an agent to perform certain physical activities such as actuating, constructing, modifying physical object, looking at, placing, navigating and perceiving.

CommunicationTask in which two or more agents shares information. This is used to classify special kind of events that have participants as agents. The means of information exchange is physical however the scope of interest here is to identify which agent communicates and what kind of information it has.

3.1.3 Motions

An *EventType* which deals with movements of an agent and motions it makes during task execution. Motion taxonomy includes concepts such as *BodyMovement* which concentrates on the movements of agents body parts. *DirectedMotion* which involves a destination and directed path for an agent to follow. An *UndirectedMotion* is opposite to directed where agent does not have any particular destination but it is important to know that the agent has moved and motion has occurred. And at last *FluidFlow* is the process by which fluid moves or has moved from one location to other.

3.1.4 Roles

Roles are concepts used to describe how an object participates in an Event or a Relation. An object can play different roles throughout its lifetime. For example, a knife may play the *PatientRole* of a grasping action, i.e. the object mainly affected by the action, and play an *InstrumentRole* in a cutting action. Other roles include *AgentRole*, *CausalProcessRole*, *AnswerRole*, etc. Further role sub-categories will be defined under appendix section.

Seba: An example with using those terms would be nice.

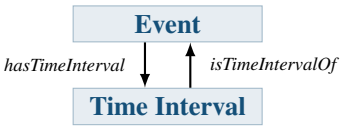
3.2 Relationships

3.2.1 Occurrence

The basic building blocks of NEEMs are the events that occur when an agent interacts with its environment through movements of its body. An event is defined as *any*

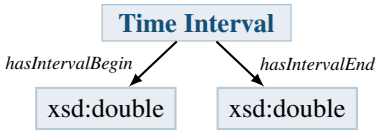
physical, social, or mental process, event, or state. Events have an associated time interval that determines the time at which the event occurs. Time data is represented as unix timestamps using XSD types.

Intent	To represent the temporal extension of events.
Competency Questions	*What is the time interval associated to this event?*
Defined in	DUL.owl



Expression	Meaning
occurs(x)	*x* is an occurence
is_time_interval(x)	*x* is a time interval
has_time_interval(y,x)	*x* is the time interval of event *y*

Intent	To quantify when something has happened.
Competency Questions	*When did it happen?*
Defined in	SOMA.owl

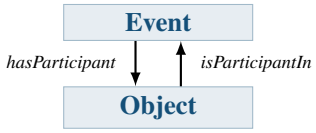


Expression	Meaning
occurs(x) during [y,z]	*x* occurs between the occurrences of *y* and *z*
occurs(x) since y	*x* and *y* begin at the same time
occurs(x) until y	*x* and *y* end at the same time

3.2.2 Participation

Events always involve some objects that play a certain role during the event. The role of being the *patient* of some event being an example. This is that the event is directed towards the object. It is not always directly observable what the role of an object might be, however, it is less problematic to just state that the object *has participated* in some event without naming a role.

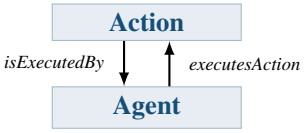
Intent	To represent participation of an object in an event.
Competency Questions	*Which objects do participate in this event? In which events does this object participate in?*
Defined in	DUL.owl



Expression	Meaning
has_participant(x,y)	*y* is involved in event *x*

Agents are defined as *agentive objects, either physical (e.g. a robot, a human or a whale) or social (e.g. a corporation, an institution or a community)*, Actions are defined as events with *at least one agent that is participating in it, and that is executing a task*. An example would be an robot that is grasping an object. In that case the robot is the agent and grasping would be the a task executed in an action. Actions can be executed by multiple agents.

Intent	To represent that an agent has executed an action.
Competency Questions	*Which agent did execute this action? Which actions are executed by this agent?*
Defined in	SOMA.owl



Expression	Meaning
is_executed_by(x,y)	*y* is executed by *x*

3.2.3 Composition

Because an Event often has relevant internal structure, it is necessary to represent relations between it and the other events that make up its composition. In SOMA, we define a *hasPhase* property to represent such relations. This is a transitive property, similar to its superproperty in DUL, *hasPart*.

Intent	To represent how an event is composed of phases.
Competency Questions	*What are the phases of this action? Is this a phase of some action?*
Defined in	SOMA.owl

Expression	Meaning
has_phase(x,y)	*y* is a phase of *x*

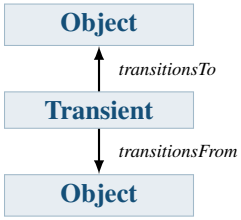
3.2.4 Transformation

Objects typically undergo changes by taking part in actions or processes. Such changes typically take one of two forms: either some property of an object changes value, or the object itself modifies its ontological characterization. As an example, a wad of dough being transported from the table to the inside of an oven has changed its position, but is still, at this point, a wad of dough. Left in a hot oven for enough time however, the wad of dough becomes bread.

The variation with time of object qualities has been described in section **??**. The object transformation pattern, described here, handles the changes of an object's ontological classification through time. An immediate problem is that ontological characterizations are necessarily discrete – there is a finite number of classes an object can belong to – while change in the physical world is continuous. In the example above, the wad of dough ceases to be dough after a few moments in the oven, because its chemical composition is changing away from the composition of dough. Nonetheless, it is not bread yet.

For practical purposes however, human beings often do not care about the exact classification of an object undergoing ontological change; only the endpoints are important. Also, there is a tendency to loosely apply ontological classification to the changing object, as if it were on either side of the transformation. The cooking wad of dough could be referred to as dough, or it could be referred to as bread. It is both, and neither. While sufficient for causal discourse, such carelessness would not work in a formal system. Hence, the approach in SOMA is to define a class of objects called *Transient*, and it is to this class that objects undergoing ontological classification change belong to. A Transient *transitionsFrom* some object with a specific classification (e.g. *Dough*) and *transitionsTo* another object with a specific classification (e.g. *Bread*). These two relations can be combined into the *transitionsBack* relation, to cover situations where an object changes in essential ways during an event, but returns to being itself after the event completes, such as a catalyst in a chemical reaction.

Intent	Ontological classification for objects undergoing type changes.
Competency Questions	*What sort of object is this? What objects "went" into the making of another? What is the outcome of some process of change acting on an object? Does an object preserve or restore its identity after change?*
Defined in	SOMA.owl



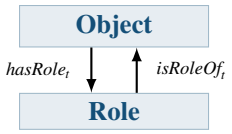
Expression	Meaning
transitionsFrom('A','B')	By entering some process of change, object 'B' becomes Transient object 'A'.
transitionsTo('A','B')	By completing some process of change, transient 'A' becomes object 'B'.
transitionsBack('A','B')	Object 'B' entered some process of change during which its ontological classification is unclear and it is replaced by Transient 'A', but after the process completes object 'B' is restored.

3.2.5 Conceptualization

The classification of entities is done from multiple viewpoints. The most essential one is *what the entitiy really is*. This is reflected in the taxonomy. However, entities may further be classified according to social aspects such as intention, purpose, etc. This type of classification is based on the *conceptualization* of entities. In particular, conceptualizations of objects and events are used to classify them in the scope of some activity. For this purpose, a comprehensive collection of concepts is used to classify objects and events from a conceptual viewpoint (**todo: insert link to list of all concepts**).

An object that participates in an event usually plays a certain role during the event. Some objects are designed to be used in certain ways, thus playing certain roles in specific tasks. This is, for example, a box which is designed to be used as a container to store items. However, roles may also be taken by objects that are not designed to be used as such. The box could, for example, also be used as a door stopper, but surely it would be inappropriate to classify it as such taxonomically. Instead, roles are defined as concepts that are used to classify the objects that particpate in an event from a conceptual viewpoint.

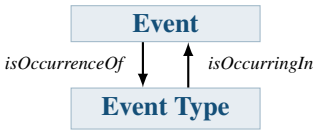
Intent	To represents objects and the roles they play.
Competency Questions	*What role does this object play? Which objects do play that role?*
Defined in	DUL.owl



Expression	Meaning
has_role(x,y)	*y* is a role of *x*
has_role(x,y) during z	*y* is a role of *x* during the occurrence of *z*

The conceptualization of an event is about how it should be interpreted, executed, expected, seen, etc. One aspect is that a single event may contribute to multiple goals, such as when an ingredient is fetched that is partly used in a step of a cooking recipe, and partly eaten raw to satisfy hunger. In such a case, the taxonomical category of the event would be unclear in case it should describe the goal to which the event contributes. Another aspect is that intentions of agents may not always be known such that the classification of events based on their goals is difficult, and different viewpoints on the same event may exist. Hence, when referring to goals, intentions, etc. we rather employ conceptual classification. This allows us to represent several different classifications of the same event in one or more situational contexts, for example an interpretations of the same event from different viewpoints.

Intent	To represent how events should be interpreted, executed, expected, seen, etc.
Competency Questions	*What are the events that are classified by this concept? What are the concepts that classify this event?*
Defined in	SOMA.owl



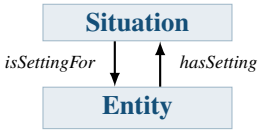
Expression	Meaning
is_occurring_in(x,y)	*y* is an occurrence that is classified by *x*
is_executed_in(x,y)	*y* is an action that executes the task *x*

3.2.6 Contextualization

An episode is seen as a *relational context* created by an observer that creates a view on a set of entities such as actions that were perfomed, and objects that played a role. We say that an episode is a *setting for* each entity that is relevant for the scope of the

episode. As an example, consider the statement *"this morning the robot made a mess on the floor while preparing coffee"*, where the preparation of coffee in the morning is the setting for the robot, the floor, and the actions that were perfomed. Several specializations of the general *is setting for* relation exist that can be used to distinguish between entities based on their type – these are, among others, *includesObject* and *includesAgent*. However, assuming hierachical organization of objects (i.e. all objects are part of some map), and events (i.e. an event is composed into sub-events) only those entities at the root of the composition (e.g. the map) are required to be included explicitely in the episode.

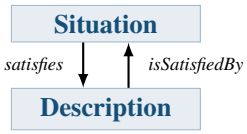
Intent	To represent that entites are included in a situation.
Competency Questions	*What are the entities that are relevant for this situation? What are the situations where this entity is relevant?*
Defined in	DUL.owl



Expression	**Meaning**
is_setting_for(x,y)	*x* is a setting for *y*

Episodes refer to concrete occurences with actual objects that are involved, and actual events that occur. The conceptualization of an episode is an abstraction that refers to concepts instead that are used to classify entities that are included in the episode. Such conceptualizations are called *descriptions*. We say that an episode *satisfies* a description in case the view represented by the episode is consistent with the conceptualization given by the description. Diagnosis being one example of a description of an episode. Stating that the performance of the robot was *amateurish* when it made a mess on the floor while preparing coffee is one example for describing an epsiode. Another type of descriptions are plans that are used to conceptualize the structure of an activity.

Intent	To represent a conceptualization of a situation.
Competency Questions	*How can this situation be conceptualized? What are the situations that are consitent with this conceptualization?*
Defined in	DUL.owl



Expression	**Meaning**
satisfies(x,y)	*x* is consistent with *y*

4

NEEM-Experience

D. BESSLER, S. KORALEWSKI

NEEM-experience captures low-level information about experienced activities represented as time series data streams. This data has often no or only unfeasible lossless representation as facts in a knowledge base. To make this data *knowledgable*, procedural hooks are defined in the ontology to compute symbols from the experience data, and to embed these symbols in logic-based reasoning.

The data is stored in a NoSQL database using JSON documents. Each individual type of data is stored in a separate collection named according to the type of data stored in the collection. When imported, the knowledge system stores the data in a MongoDB [4] server, for which the knowledge system implements a client for querying the data during question answering. The query cursor concept employed in MongoDB integrates nicely with backtracking based search employed in the knowledge system. It further scales well to large amount of data and can be distributed amongst clusters through built-in automatic sharding.

The data in NEEM-experiences is represented as time series and indexed in time order. The different experience data types need to define a dedicated time key for computing the search index.

The experience data in NEEMs has individual characteristics regarding the format, compressed representation, and what symbols the knowledge system can abstract from the data. In this chapter, we describe these aspects for the experience data types covered in NEEM version 1.0 .

[4] https://www.mongodb.com/

4.1 Pose Data

A robotic system typically has many mobile components arranged in a kinematic chain. Each component in a kinematic chain has an associated named coordinate frame such as world frame, base frame, gripper frame, head frame, etc. Coordinate systems are always 3D, with x forward, y left, and z up. 6 DOF relative poses are assigned to the different frames. These are usually updated with about 10 Hz during movements, and expressed relative to the parent in the kinematic chain to avoid updates when only the parent frame moves. The transformation tree is rooted in the dedicated world frame node (also often called map frame).

The data is used by the EASE knowledge system to answer questions such as:

- Where was the head frame relative to the world frame, 5 seconds ago?
- What is the pose of the object in the gripper relative to the base?
- What is the pose of the base frame in the map frame?

Pose data is saved in MongoDB collections named “tf”, the format is described below.

4.1.1 Format

The pose data structure has fields for encoding the translation and rotation of a coordinate frame. The parent frame and time stamp of pose estimation are stored in the *header* field of the data structure. The transform coordinate frame is assigned to the *child_frame_id* field.

Note that static frames may be recorded at lower frequency – about every two seconds. This usually reduces the data size significantly. At the moment, no other motion data compression, such as motion JPEG, is supported.

Field	Type	Description
tf	dict	–
header	dict	–
seq	uint32	consecutively increasing ID
stamp	time	time stamp of this transform
frame_id	string	parent coordinate frame of this transform
child_frame_id	string	coordinate frame of this transform
transform	dict	–
translation	dict	–
x	float64	*x* axis translation
y	float64	*y* axis translation
z	float64	*z* axis translation
rotation	dict	–
x	float64	*x* component of quaternion
y	float64	*y* component of quaternion
z	float64	*z* component of quaternion
w	float64	*w* component of quaternion

Fig. 1. The pose data structure in the EASE system.

5

NEEM-Acquisition

S. KORALEWSKI

This chapter focuses on the acquisition process of NEEMs . At first, we will provide the tools and procedures to acquire episodic memories from robots performing experiments. The second section focuses on the NEEM acquisition from virtual reality.

Each section will contain an example NEEM to provide insights how the representation, described in Chapter **??**, is utilized to represent performed activities by robots or by humans. In addition, each example NEEM is available on the NEEM-hub for downloading.

5.1 Robot NEEMs

This section focuses on describing how to generate NEEMs from experiments performed by the robot.

5.1.1 Prerequisite

Before you are intending to generate your episodic memories, make sure you are familiar with the Cognitive Robot Abstract Machine (CRAM)[5] system and installed it on your machine. CRAM is a cognitive-enabled planning framework which allows to design high-level plan for robots. This section requires that your robot plan is written in CRAM to be able to generate NEEMs . However, once you are familiar with our

@Seba: Add more detailed description about CRAM

[5] http://cram-system.org/cram

planning and logging components, you will be able to port those components to your preferred planning tool.

In addition to having a valid CRAM plan, you will need the following software components to be installed:

- A MongoDB server with at least version 3.4.10[6]
- KnowRob [7]
- SOMA ontology[8]
- CRAM ontology[9]

5.1.2 Recording Narrative Enabled Episodic Memories

Our recording mechanism captures every executed CRAM action and its parameter. In addition, the logger puts the actions in relation to each other by creating a hierarchy which is described in Section **??**.

Before you can begin to record your own NEEMs , you need to include the "cram-cloud-logger" package into your CRAM plan. After you included the package, you need to enable the logging via:

```
(setf ccl::*is-logging-enabled* t)
```

Listing 5.1. Enabling NEEM Logging in a CRAM plan

The only things left to do start the logging before the plan execution and after the execution to finishing it. It can look like the following:

```
(ccl::start-episode)
(urdf-proj:with-simulated-robot (demo::demo-random nil ))
(ccl::stop-episode)
```

Listing 5.2. Steps to Record an Episode for a CRAM Plan

The generate NEEM will be stored per default in "~/knowrob-memory". Keep in mind to have KnowRob launched before starting the NEEM recording. You can start KnowRob via:

```
roslaunch knowrob_memory knowrob.launch
```

Listing 5.3. How to Start KnowRob

[6] https://www.mongodb.com/
[7] https://github.com/knowrob/knowrob
[8] https://github.com/ease-crc/soma
[9] https://github.com/ease-crc/cram_knowledge

5.1.3 Data

Per default your generated NEEM will be stored under "~/knowrob-memory/<timestamp >". Your NEEM will consist of at least 5 folders - *annotations*, *inferred*, *roslog*, *ros_tf* and *triples*. In the following, we will give an overview which information is contained in those folders:

@Seba add proper descriptions

annotations description

inferred description

ros_tf description

triples description

5.1.4 Add Semantic Support to your Designed Plans

The disadvantage of having a strong semantic knowledge representation is that the used ontology requires updates when semantic knowledge has to be extended. Currently, SOMA focuses on the support to record table setting/cleaning-up experiments. If you want for instance create NEEMs for e.g. autonomous cars, you will need to extend the SOMA ontology and the recording mechanism with your required actions, parameters and objects. How to extend the SOMA ontology is described in Section **??**. In the following subsection, we will describe how you can extend the CRAM recording mechanism to support your required semantic knowledge.

New Tasks Definition

If you want to semantically record new tasks such as e.g. accelerating or breaking, you need to define them in the ontology as described in Section **??**. Unknown tasks will be logged as *PhysicalTask*. The *PlanExecution* instance pointing to the*PhysicalTask* will have a comment attached with the statement "Unknown Action: <CRAM-ACTION-NAME>", where <CRAM-ACTION-NAME>" is the action name you defined in your plan. After you defined your new actions in SOMA , please open the "knowrob-action-name-handler.lisp" in the "cram-cloud-logger" package and add your new actions in the format:

```
(defun init-action-name-mapper ()
   (let ((action-name-mapper (make-instance 'ccl::
    cram-2-knowrob-mapper))
     (definition
         '(("reaching" "Reaching")
           ("retracting" "Retracting")
           ("lifting" "Lifting")
```

```
            ("opening" "Opening")
            ("<CRAM-ACTION-NAME>" "<SOMA-ONTOLOGY-NAME>")
```

Listing 5.4. Linking the CRAM Action to the Ontology Concept

where <CRAM-ACTION-NAME>is the name of your action used in the cram plan and <SOMA-ONTOLOGY-NAME>. With this step you added successfully the support of the new action to CRAM NEEM episodic memory logger.

Adding New Objects

Unknown objects will be logged as instance of *DesignedArtifact*. In addition, a comment like "Unknown Object: <CRAM-OBJECT-TYPE>" will be attached to this instance. <CRAM-OBJECT-TYPE> indicates which object you need to define in the ontology . After you added your object to the ontology, open the "utils-for-perform.lisp" in the "cram-cloud-logger" package and include the new object in the hash table generate in "get-ease-object-lookup-table" like:

set reference

```
(defun get-ease-object-lookup-table()
  (let ((lookup-table (make-hash-table :test 'equal)))
    (setf (gethash "BOWL" lookup-table) "'http://www.ease-crc.
    org/ont/EASE-OBJ.owl#Bowl'")
    (setf (gethash "CUP" lookup-table) "'http://www.ease-crc.
    org/ont/EASE-OBJ.owl#Cup'")
    (setf (gethash "<CRAM-OBJECT-TYPE>" lookup-table) "'<
    SOMA-ONTOLOGY-ENTITY-URL>'")
 lookup-table))
```

Listing 5.5. Linking the CRAM Object to the Ontology Concept

where the key is <CRAM-OBJECT-TYPE>, the object type used in the CRAM plan, and <SOMA-ONTOLOGY-ENTITY-URL>the value which is the uri pointing to the object concept created in SOMA .

Adding New Failure

Unknown CRAM failures will be logged as instance of *Failure*. In addition, a comment like "Unknown failure: <CRAM-FAILURE-NAME>" will be attached to the instance. <CRAM-FAILURE-NAME> indicates which failure you need to define in the ontology. After you added your failure to the ontology, open the "failure-handler.lisp" in the "cram-cloud-logger" package and add your new failure in the format:

set reference

```
(defun init-failure-mapper ()
  (let ((failure-mapper (make-instance 'ccl::
    cram-2-knowrob-mapper))
    (definition
```

```
    '(("cram-common-failures:low-level-failure" "
LowLevelFailure")
    ("cram-common-failures:actionlib-action-timed-out" "
ActionlibTimeout")
    ("<CRAM-FAILURE-NAME>" "<SOMA-ONTOLOGY-NAME>")
```

Listing 5.6. Linking the CRAM Action to the Ontology Concept

where <CRAM-FAILURE-NAME> is the name of your failure defined in the cram plan and <SOMA-ONTOLOGY-NAME>. With this step you added successfully the support of the new failure to CRAM NEEM episodic memory logger.".

Adding New Rostopic

Per default, we log the rostopics tf and tf_static. If you need to log additional topics, open "memory.pl" in the "knowrob_memory" package from KnowRob and include your topic in the "mem_episode_start(Episode)" function. After the NEEM generation, the data will be stored in the created NEEM folder under the file "roslog/¡rostopic¿.bson".

Adding New Parameters

Unknown parameters will be logged as comment attached to the corresponding *PlanExecution* instance. The comment statement "Unknown Parameter: PARAMETER-NAME -####- PARAMETER-VALUE/¿" The current parameter types are represented

@Ontology group: Please make sure that is available in the ontology

1. Integer/Floats
2. Posen
3. Spatial Relations
4. Link to entities of other ontologies such as http://knowrob.org/kb/PR2.owl#pr2_right_arm

Before you want to model your parameter what data type your parameter is. If it is a complex object, you need to consider how you want to to represent it an the ontology. For simpler representaion such has a discrete domain representation, you might represented as the domain values as *Region* and add the model the parameter as a subconcept of *Parameter*. More information about the concepts *Region* and *Parameter* can be found in .

reference to Region and parameter

5.1.5 Adding New Reasoning Tasks

@Ontology group: How to log the result of the reasoning query ?

5.1.6 Next steps

Add neem2narrative

After you have generate your NEEM , you can use the tool to generate an cvs file for your NEEM . Keep in mind that the csv is a abstraction of NEEM-narrative and can be used to make data-mining on explicit knowledge. For more sophisticated analysis, you will need to use KnowRob . We use this general analysis to identify bottlenecks in our plan execution. We also showed that with a collection of NEEMs we are able to improve the robot's performance. The tools for the feature extraction can be found here Now that you we encourage you to generate your NEEMs and share them via our NEEM-hub .

Af

Add link

5.1.7 Example

Add belief state example

What about reference to semantic map ?

Provide link for downloading real log file

In this chapter we will show a generated NEEM based on the version 1.0 . You can download the log used in this example here. We will not go into the details about the NEEM-experience meaning the tf since we just stored the exact tf message in the database. Therefore it is not required to have detail explanation since the idea of tf is already understood.

This log file represents a experiment where the PR2 tired to grasp five objects which were located on a kitchen counter and bring each object to our table and place them there. We defined grapsing + placing as a transporting. This experiment was performed in projection meaning in simulation. CRAM is not used using a belief state during execution in projection, therefore we will not show how a belief state is represented. The object which were precived or grapsed are represent as strings.

How we can differ between those two goal Locations ?

Figure 2 show how a transporting task for a cup is represented. To summarize what happend during the transporitng task in the experiment, The task was performed not successfully in projection. The failure was that the cup was unfetchable for the PR2 . The PR2 used the left arm to grasp the cup from a connected space region and transported to object to a specific pose in the map. This task has also one sub action. Given the task the agent inferred that to achieve this task, it has to perform a PickingUpAnObject task. You can also read from the log that the PR2 2 tried already an transporting task on another object and that this task is also not the last transporitng task.

```
Individual: TRANSPORTING_KVRJlsOG
Facts:
goalLocation knowrob;ConnectedSpaceRegion_SgfniCWn
goalLocation knowrob;Pose_eLrKlpcE
arm pr2;pr2_left_arm
endTime knowrob;timepoint_1526640320.499945
failure CRAM-COMMON-FAILURES:OBJECT-UNFETCHABLE
nextAction knowrob;TRANSPORTING_ysHEUIwW
objectType "CUP"\todo{add figure to connected space region
}
performedInProjection true
previousAction knowrob;TRANSPORTING_FkRlsFGa
startTime knowrob;timepoint_1526640316.535484
subAction knowrob;PickingUpAnObject_iLTYaxBE
taskContext TableSetting
taskSuccess false
```

Fig. 2. Example for a Transporting Task

The next transporting represented in . . . 3 represents a transporting task of a bowl object. In contrast to the previous task, this task was succuessfully performed. Again the PR2 used only the left arm. The PR2 grasped the object from a connected space region which is an kitchen counter 4. It placed the object in the specific pose in the map which is displayed in 5.

```
Individual: TRANSPORTING_ysHEUIwW
Facts:
  goalLocation knowrob;ConnectedSpaceRegion_YOXaigqs
  goalLocation knowrob;Pose_GIcRIGdP
  subAction knowrob;PickingUpAnObject_CTGdupxa
    subAction knowrob;PuttingDownAnObject_uBRtvLcg
    arm pr2;pr2_right_arm
    endTime knowrob;timepoint_1526640332.733415
    nextAction knowrob;TRANSPORTING_HRgQoeEw
    objectType rdf:resource="BOWL"
    performedInProjection true
    previousAction knowrob;TRANSPORTING_KVRJlsOG
    startTime rdf:resource="&knowrob;timepoint_1526640320
  .561752
  taskContext "TableSetting"
  taskSuccess true
```

Fig. 3. Example for a second Transporting Task

```
Individual: ConnectedSpaceRegion_YOXaigqs
Facts:
  onPhysical knowrob:iai_kitchen_sink_area_counter_top
```

Fig. 4. Example for Connected Space Region

```
Individual: Pose_GIcRIGdP
  Facts:
  quaternion 0.0 0.0 1.0 0.0
  translation -0.7599999904632568 1.190000057220459
  0.9300000071525574
```

Fig. 5. Example for a Pose

5.2 VR NEEMs

6

NEEM-Hub

S. KORALEWSKI

6.1 Publishing

In this chapter we have to describe, how the NEEM narrative and NEEM experience should be represented e.g. using OWL and MongoDB and how the partners can upload those file to openEASE.

6.2 Maintaining

References

[1] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. WonderWeb deliverable D18 ontology library (final). Technical report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web, 2003.