Learning Robot Activities from Experiences: An Ontology-based Approach

Bernd Neumann, Lothar Hotz and Andreas Günter

Technical Report FBI-HH-B-300/13

University of Hamburg, Department of Informatics
Cognitive Systems Laboratory

November 2013

Roboteraktivitäten aus Erfahrungen lernen: Ein ontologiebasierter Ansatz

Bernd Neumann, Lothar Hotz und Andreas Günter

{neumann, hotz, guenter}@informatik.uni-hamburg.de

Zusammenfassung

In diesem Bericht beschreiben wir Methoden, mit denen ein Roboter aus Erfahrungen lernen kann, und illustrieren dies mit ersten experimentellen Ergebnissen. Das Lernen erfolgt mithilfe von hybrider Wissensrepräsentation auf der Basis von OWL 2, erweitert um Konstrukte zur quantitativen Repräsentation von zeitlichen und räumlichen Informationen. Als zentrale Lernaufgabe untersuchen wir, wie ein neues Konzept aus positiven Beispielen abgeleitet werden kann. Dazu werden die Erfahrungen konzeptualisiert und durch die Konstruktion eines "Guten Gemeinsamen Subsumierers" (GCS) verbunden. Gegenüber dem Kleinsten Gemeinsamen Subsumierer (LCS) ermöglicht der GCS kognitiv plausiblere Lernergebnisse. Ein ähnliches Verfahren wird angewandt, wenn der Roboter ein gelerntes Konzept auf eine neue Situation anwenden soll, die nicht vom Konzept abgedeckt wird. Als dritte Lernsituation untersuchen wir das Verfeinern eines bestehenden Konzeptes aufgrund eines Negativbeispiels. Unsere Lernmethoden sind in mehrerer Hinsicht innovativ. Konzepte werden eng mit Fakten verbunden, die als Kontext relevant sein könnten. Auf diese Weise können konkrete Episoden maßvoll verallgemeinert werden. Lernbeispiele werden autonom oder nur auf groben Anweisungen basierend aus aufgezeichneten Episoden extrahiert, die auch irrelevante Beobachtungen enthalten können. Um Korrespondenzen zwischen konzeptualisierten Beispielen herzustellen, verwenden wir Ideen aus der Kognitionswissenschaft zum analogen Schließen ("structure-mapping theory"). Die Forschungsarbeiten und Experimente wurden im Rahmen des Projektes RACE durchgeführt, in dem ein Roboter lernt, Kellnerdienste in einem Restaurant zu leisten.

Learning Robot Activities from Experiences: An Ontology-based Approach

Bernd Neumann, Lothar Hotz and Andreas Günter

{neumann, hotz, guenter}@informatik.uni-hamburg.de

Abstract

This report describes methods and first experiments for a robot learning from experiences. Learning is performed within a hybrid knowledge representation framework based on the ontology language OWL 2 and extensions for quantitative spatial and temporal information. The central learning task considered is to establish a new concept based on positive examples. This is accomplished by conceptualizing each example and constructing a "Good" (rather than a Least) Common Subsumer of the conceptualizations in order to obtain cognitively plausible learning results. Another learning task arises when the robot must apply a learnt concept to a new situation which is not covered by the concept. A third learning situation concerns the refinement of learnt concepts by negative examples. Our approach is new in several respects. First, concepts are conjoined with factual knowledge representing relevant context, thus allowing tight models and conservative generalizations of episodes. Second, learning examples are autonomously extracted from episodes or with the help of crude instructions. Thus examples may also contain irrelevant observations. Third, correspondence between examples and concepts is established using ideas of the structure-mapping theory in Cognitive Science, emphasizing the correspondence of relations. Research and experiments are based on work in Project RACE[1] where a robot plays the part of a restaurant waiter.

1. Introduction

There is widespread agreement in robotics and in the knowledge-based systems community that robots, in order to become useful in complex real-world domains, must be equipped with learning capabilities. Robots cannot be programmed from scratch to perform well in dynamic, partially unknown environments, they must be able to learn autonomously or by instructions. In this paper, we investigate learning of high-level activity concepts represented in the formal knowledge-representation framework OWL 2. Learning will be based on experiences recorded by the robot and, at times, instructions by a human instructor. Examples are taken from the restaurant domain with a robot performing as a waiter. Activities such as serving a guest are typically composed of several levels of subactivities, down to elementary robot capabilities such as moving or grasping. Hence ontological representations encompass compositional hierarchies besides customary taxonomies.

To convey a first understanding of the learning tasks investigated in this paper, consider the three scenarios sketched in Fig. 1.

In Scenarios A and B, the robot - here called "trixi" - receives detailed instructions how to serve a coffee to a guest and learns that these activities constitute a "ServeACoffee":

Instructions for Scenario A: "Move to counter1, grasp mug1-A, move to south of table1, place mug1-A at placement area west – this is a ServeACoffee."

[1] RACE is funded by the EC Seventh Framework Program theme FP7-ICT-2011-7, grant agreement no. 287752.

Instructions for Scenario B: "Move to counter1, grasp mug1-B, move to north of table1, place mug1-B at placement area east – this is also a ServeACoffee."

In Scenario C, it is assumed that the robot has learnt a concept from the two examples and will serve the coffee to the placement area south right of guest1-C.

Instructions for Scenario C: "Do a ServeACoffee to guest1-C at table2."

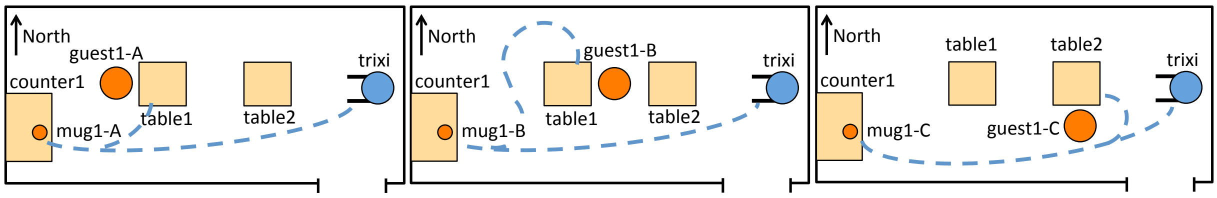


Fig. 1: a) Scenario A b) Scenario B c) Scenario C

In all scenarios, we assume that the robot knows the location of the guest and of the placement areas on the table. However, it does not know which placement area to approach for guest1-C. Part of the learning task is therefore to generalize the recorded experiences of Scenarios A and B and create a concept which can be applied to similar but not identical situations. In general, we consider learning scenarios where a service robot, equipped with a repertoire of basic robot operations, is incrementally instructed by examples and expected to autonomously apply its enhanced competence to new situations, possibly requiring further generalizations.

Learning structured conceptual representations from examples is not a new topic. Winston's seminal work on learning block structures such as bridges is a well-known starting point [Winston 75]. One can distinguish three major lines of research which have contributed to today's understanding of the field:

(i) the development of well-understood and standardized knowledge representation formalisms, (ii) research into Cognitive Robotics, connecting symbolic high-level representations with quantitative low-level operations, and (iii) advances in learning and reasoning models in Cognitive Sciences. In the following, we will review contributions of these lines of research on which we have built our work.

The possibilities for a standardized knowledge representation have been much improved by the development of the Semantic Web and the Web Ontology Language OWL [Hitzler et a. 08]. Exploiting work in Description Logics (DL), OWL representations can be processed by powerful symbolic reasoners. As an application in Computer Vision, it has been shown in [Bohlken et al. 11] that conceptual definitions of compositional structures for activity monitoring can be represented in OWL and can be used to automatically generate corresponding recognition procedures. Quantitative temporal characterizations were attached to symbolic representations via the Semantic Web Rule Language SWRL [Grosof et al. 03]. SWRL rules can support reasoning with concrete data (ABoxes in DL terminology) but cannot be considered by general DL reasoners.

One drawback pervading OWL representations is the problem to express the sameness of entities occurring in several parts of a compositional structure, for example, the sameness of an agent performing several coherent activities. However, recent work by the group of Hitzler [Krötzsch et al. 11] has established "nominal schemas" as a well-founded way to express sameness by identical variable names, similar to variables in Datalog. We adopt nominal schemas for our work, although this extension of OWL is not yet standardized.

Learning DL concepts from examples has been investigated before [Cohen and Hirsh 94, Lehmann 09, Lehmann and Hitzler 07], and it is well understood how concept expressions can be generalized or refined if they must be adapted to new examples. In principle, version space learning as introduced in [Mitchell 82] can be applied to define the space of concepts which correctly classify positive and negative examples. In our approach, we have designed model learning as a cognitively plausible strategy for ontology evolution without bookkeeping of version space boundaries.

The second line of research influencing our work concerns methods for seamlessly connecting symbolic high-level representations with low-level robot operations, in particular with numeric spatial and temporal data. Formally, this is possible by extending a DL by "concrete domains" [Lutz 03], already partially realized in OWL by certain datatype properties. In practice, it has proved useful to support symbolic reasoning in OWL by dedicated constraint solvers [Günther et al. 12, Bohlken et al. 13]. Such a knowledge representation and reasoning architecture has also been chosen for the robot system in RACE [Rockel et al. 13], providing the framework for learning in this paper. Numeric values are mainly used for temporal information, i.e. duration of activities and temporal relations between activities. Furthermore, each value is represented as an uncertainty range. It is interesting that this kind of numerical representation can be homogeneously integrated into learning procedures based on generalization and refinement.

Cognitive Science is a third line of research which has influenced our work, in particular in terms of insights about analogical reasoning and transfer learning for humans. We will show that in order to create a concept from examples in a realistic setting, a correspondence analysis must be performed which is closely related to the correspondence analysis in analogical reasoning [Holyoak et al. 01, Kean and Costello 01]. Cognitive models for learning also illustrate the need for a judicious generalization of properties, depending on domain characteristics [Wilson et al. 01]. In our restaurant domain, for example, when learning a service activity, it is more appropriate to abstract from the colour of a guest's clothing, but not from a guest's position at a table.

In Section 2, following this introduction, we describe the knowledge representation conventions adopted for our learning work. In Section 3, we present our learning approach, called Ontology-based Learning from Examples (OLE), in detail. This allows the robot to form a new activity concept from examples, to adapt an existing concept to cover a new task, or to refine a concept based on a negative example.

In Section 4, we evaluate OLE using several other scenarios in the restaurant domain. It will be shown that the learning procedure can lead to desirable activity models with very few examples, given an appropriate ontology. More examples may be required, however, if recorded examples contain many irrelevant details, initially leading to overly restricted concepts.

The paper ends with conclusions, summarizing the work and pointing out some open problems for future research.

2. Knowledge Representation Conventions

Terminological Knowledge

Concepts in OLE are represented in an ontology using a restricted version of the Web Ontology Language OWL 2 and Protégé as an editor. The concept definitions are also the basis for a hierarchical planner as well as other components of the robotic system realized in RACE [Rockel et al. 13]. For textual concept representations, we use the Manchester Syntax in this paper[2]. Following the conventions of OWL, a concept will be often called 'class' when we deal with OWL representations. As an example of an activity class, the definitions of PlaceObject1-A and classes used as property fillers are listed below.

```
Class:  PlaceObject1-A
SubclassOf:    PlaceObject
          that    hasHolding exactly 1 {?Holding1-A}
          and      exactly 1 {?On1-A}
          and     hasBefore exactly 1 Before1-A

Class:  Holding1-A
SubClassOf:    Holding
EquivalentTo: {?Holding1-A}
          that    hasRobot value trixi
          and     hasPassiveObject {?Mug1-A}

Class:  On1-A
SubclassOf:    On
EquivalentTo: {?On1-A}
          that    hasPhysicalEntity {?Mug1-A}
          and     hasArea value paWest1

Class:  Before1-A
SubclassOf:    Before
          that    hasFirst exactly 1 {?Holding1-A}
          and     hasSecond exactly 1 {?On1-A}
          and     hasBeforeRange exactly 1 TimeRange
```

Listing 1: Class definitions of PlaceObject1-A and its property fillers

Class names begin with upper-case, individuals with lower-case letters, property names with the prefix 'has'. The postfix '-A' is part of the class names and used here to mark classes conceptualized from the robot's recording of Scenario A (see Fig. 1). In addition to the properties spelled out above, the concepts inherit properties from the ontological ancestor Occurrence:

```
          and     hasStartTime only TimeRange
          and     hasFinishTime only TimeRange
          and     hasDuration only TimeRange
```

The datatype TimeRange is used to express an uncertainty range of a time point and is a shorthand for two separate properties with numerical fillers, e.g.

```
          and     hasStartTimeLowerBound only Int
          and     hasStartTimeUpperBound only Int
```

We currently use only a subset of OWL 2 with the syntax shown in Listing 2. In DL terminology, the syntax corresponds to an Attribute Language with full existential quantification and number restriction (ALEN).

[2] http://www.w3.org/TR/owl2-manchester-syntax/

Note the restrictive use of class expressions for property fillers: If a filler requires a more expressive definition, a class name must be introduced and defined in a separate class definition, as shown for Holding1-A and On1-A. The reason for this restrictive grammar is our interest in keeping a simple syntactical correspondence between class definitions and the constituents of episodes, see below. It is apparent that named concepts as property fillers can be replaced by their definitions, creating nested concept definitions and a reduced number of names. Also note the absence of negation. This is motivated by our primary interest in modelling conceptualizations of episodes recorded under an open-world assumption (OWA).

```
Class: <className>
SubclassOf: <className>
        [ ‘that‘ [inverse] <propertyName>  <restriction>
        { ‘and‘ [inverse] <propertyName>  <restriction> } ]

<restriction> ::= ‘only‘ <classExpression> |
                  ‘some‘ <classExpression> |
                  ‘exactly‘ <integer> [<classExpression>] |
                  ‘min‘ <integer> [<classExpression>] |
                  'max' <integer> [<classExpression>] |
                  'value' <individual>

<classExpression> ::= <className> |
                  <nominalSchema>

<nominalSchema> ::= '{' <individualVariable> '}'
```

Listing 2: Syntax of restricted grammar for concept definitions

Class definitions can represent hierarchical compositional structures by letting a parent class (an aggregate) refer to its components via partonomical properties. These properties have the common parent property 'hasPart'. We often refer to the root of a compositional hierarchy as *root concept* or *root class*.

We now explicate the use of nominal schemas as property fillers. A nominal schema specifies an individual of a particular class with a variable name which may reoccur as a property filler in several places and must be instantiated with the same known individual [Krötzsch et al. 11] which may be, however, any value of its class, different from the usual individuals. As pointed out in the introduction, expressing sameness of individuals is important for compositional hierarchies. For our knowledge representation purposes, the class of the individuals of a nominal schema {?X} is represented by a class definition with the additional axiom EquivalentTo: {?X}. The scope of a nominal schema is taken to be the ontology of the domain. In the examples above, {?Holding1-A}, {?Mug1-A}, and {On1-A} are nominal schemata.

Assertional Knowledge

The assertional knowledge of a robot comprises episodes which are stored as experiences in the robot memory. An episode consists of dynamic knowledge which describes spatially and temporally coherent occurrences in the restaurant as viewed by the robot, and permanent (or background) knowledge about the robot's environment which is assumed to be valid for all times.

Using the YAML syntax, assertional knowledge is described in terms of 'fluents', each specifying one occurrence or state of the world in a coherent time interval. For example, the following fluent specifies that the robot trixi is located at the premanipulation area pmaSouth1 from time unit 23 to 50 during Episode A (constituting an instance of At1-A):

```
!Fluent
Class_Instance:         [At1-A, at1-A]
StartTime:      [23, 23]
FinishTime:     [50, 50]
Properties:
        -       [hasPhysicalEntity, Robot, trixi]
        -       [hasArea, PMA, pmaSouth1]
---
```

A fluent has the following syntax:

```
'!Fluent'
'Class_Instance: [' <className> ',' <individualName> ']'
'StartTime: ' <timeRange>
'FinishTime: ' <timeRange>
'Properties:'
{       '-      [' <propertyName> ',' <className> ',' <individualName> ']' }
'---'
```

3. Concept Formation and Adaption

As illustrated by the scenarios shown in Fig. 1, the OLE approach to concept learning includes a supervised learning task where an instructor provides concrete examples and a name for a new concept (ServeACoffee in Scenarios A and B), and an unsupervised learning task where the robot must adapt a concept to a new situation (Scenario C). We believe that learning situations of this kind may play a key role when employing service robots in new domains. Technically, we realize both learning tasks by two procedures: *conceptualizing* an example and adapting a concept to a conceptualized positive example or, more generally, *merging* two corresponding concepts. In effect, this approach allows to formulate learning solely based on conceptual expressions, known as the "single-representation trick" [Cohen and Feigenbaum 82]. We also sketch a procedure for *refining* a concept in order to exclude a negative example.

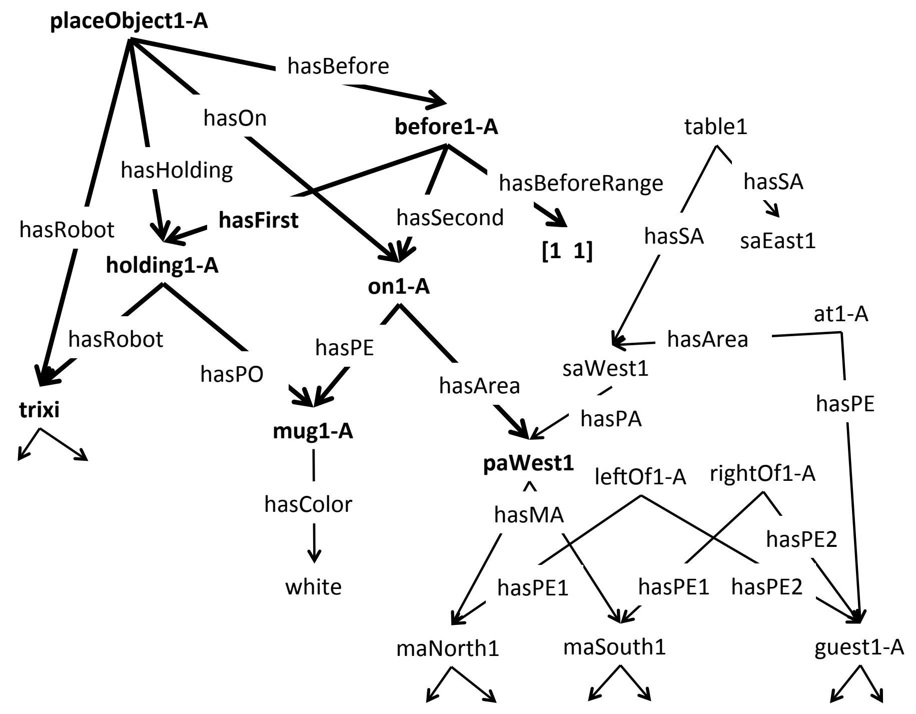
3.1 Conceptualizing Experiences

In this section we describe the conceptualization of occurrences selected from an episode as parts of a new concept CN. The process consists of two steps: (i) Determining the fluents which are relevant parts of CN, and (ii) creating the new class definitions for CN and its parts.

Determining Relevant Fluents

A complete description of all terminological and assertional knowledge necessary for a complex task such as ServeACoffee typically involves many class definitions and assertions. We call this knowledge the *support* of a concept, and in a concrete instance, the support of that instance. It is not possible to give an undisputable definition or computational procedure for determining support - this is the qualification problem in AI [McCarthy and Hayes 69]. We therefore employ a heuristic to the effect that all aspects of the scene are considered to be likely support which are either constituents of the task or temporally and spatially related to it, as observed by the robot.

It is useful to visualize the support structure of a task by means of a *property graph* with individuals as nodes and properties as directed edges. A node connected to another node by an edge will sometimes be referred to as "property triplet". Fig. 2 shows the property graph for placeObject1-A, using definitions of Listing 1, among others. As components, this robot activity comprises a precondition holding1-A with the properties hasRobot and hasPO (hasPassiveObject), and a postcondition on1-A with the properties hasPE (hasPhysicalEntity) and hasArea. Temporal properties of occurrences, specifying stat time and finish time, are not included for clarity. The graph also shows some of the observations which the robot has made: guest1-A at the sitting area saWest1, right of the manipulation area maSouth1 and left of the manipulation area maNorth1. Permanent knowledge includes the robot trixi, table1, its sitting areas saWest1 and saEast1, the corresponding placing areas, manipulation areas and premanipulation areas (not all shown). Class information for individuals are not represented in the graph, but are of course available.



Legend:

PE	PhysicalEntity	PO	PassiveObject	SA	SittingArea
PA	PlacingArea	MA	ManipulationArea	PMA	PremanipulationArea
➔	property for direct support	→	property for indirect support		

Fig. 2: Property graph for an instance of the concept PlaceObject with related permanent knowledge and observations.

In our learning context we want to distinguish between nodes which are closely related to a root node, the *direct support,* and those representing additional observations, the *indirect support*. We define the direct support to comprise all nodes which can be reached from the root node via directed partonomical properties (the components of the aggregate hierarchy), and in addition all property fillers of these nodes which have been part of the activity plan and hence included in the episode. Note that there is no guarantee that all nodes of the direct support are captured in a recorded episode due to the robot's limited sensory activities.

In Fig. 2, the nodes of the direct support of placeObject1-A are shown in bold, all other nodes are indirect support. Note that maSouth1 will be direct support in the larger context of ServeACoffee.

Depending on the scene and the observations of the robot, the indirect support may comprise many nodes, e.g. about physical entities located near to the robot or the guest, or about other occurrences happening in the temporal and spatial vicinity of the robot activity, such as a person passing by. It is the task of the conceptualization procedure to hypothesize which nodes of the indirect support are relevant for a newly created concept and must be included in an ontological definition. As a first approach, we make use of a heuristic which measures relevance in terms of the smallest distance of a node to any of the nodes of the direct support. Distance is measured by counting the number of property edges irrespective of their direction. The success of this relevance heuristic depends largely on the observations taken by the robot. If they reflect scene objects which are spatially and temporally close to the robots actions, they may very well indicate relevance. A more profound way for deriving a relevance ranking would require robot experiences and background knowledge beyond our learning scenarios.

In our experiments we have included nodes within a distance of 2, whereby connections via the two legs of a reified relation (e.g. at, on) where counted as 1. Fig. 2 shows that the important instances table1, guest1-A and saWest1 (the sitting area west of table1) are included as a result of the relevance analysis, whereas saEast1 (the sitting area east of table1) is deemed irrelevant.

Conceptualization

We now describe the conceptualization of instances relevant for a new concept CN. The operations are performed on the fluents corresponding to the relevant nodes of the property graph, obtained by the relevance analysis described above. The goal is to produce class definitions in OWL for future use. To this end, conceptualization abstracts from absolute times and expresses the temporal structure of the direct support of CN by durations and offsets. In the case of a robot activity, conceptualizations will allow to plan and execute the same activities, in the case of an observed occurrence, they will allow to recognize the same occurrences. Above all, they provide a conceptualized input for example-based adaptation of an existing concept.

It is convenient to formulate the conceptualization rules as transformations of fluents to class definitions obeying the restricted form described in Section 2. The following rules are used:

Class definition	
<className> for 'Class:'	<individualName> of fluent with capitalized first letter
<className> for 'SubclassOf:'	<className> of fluent
<propertyName>	<propertyName> of fluent

<restriction>	property filler is permanent knowledge: 'value' <individualName> property filler is non-permanent knowledge: 'exactly 1' <individualName> of fluent with capitalized first letter*
<timeRange> for 'hasStartTime' and 'hasFinishTime'	[-inf, inf]
<timeRange> for 'hasDuration'	[(1-q)(minimal duration of fluent), (1+q)(maximal duration of fluent)] with 0≤q≤1 as tolerance constant

*Identical non-permanent property fillers occurring in several fluents are conceptualized as a nominal schema, see Section 2. The variable name is marked with a preceding '?'.

Table 1: Conceptualization rules for transforming fluents into class definitions

As an example, consider the conceptualization of the fluent on1-A in Fig. 2.

```
!Fluent
Class_Instance: [On, on1-A]
StartTime:      [100, 100]
FinishTime:     [250, 250]
Properties:
        -       [hasPhysicalEntity, PhysicalEntity, mug1-A]
        -       [hasArea, Area, paWest1]
---
```

The conceptualization produces the following class definition (a tolerance q=0.1 has been used for the duration):

```
Class: On1-A
EquivalentTo: {?On1-A}
SubClassOf: On
        that    hasPhysicalEntity {?Mug1-A}
        and     hasArea value paWest1
        and     hasStartTime exactly 1 [-inf  inf]
        and     hasFinishTime exactly 1 [-inf  inf]
        and     hasDuration exactly 1 [135  165]
```

Since two class definitions refer to the same instance of On1-A, this is represented as a nominal schema. Note that mug1-A has been conceptualized to Mug1-A (represented as a nominal schema) and thus can be instantiated by any mug, while paWest1 is part of the robot's permanent knowledge and remains a specific individual.

Constructing a New Top-level Concept Definition

The last step of conceptualization is the construction of a new concept definition with the name CN, properties linking CN to the component concepts, and specified temporal relations. For the example illustrated by Fig. 1a, the new concept ServeACoffee has the following definition:

```
Class:  ServeACoffee-A
SubclassOf:   RobotActivity
        that    hasDrive exactly 1 {?Drive1-A}
        and     hasGrasp exactly 1 {?Grasp1-A}
        and     hasDrive exactly 1 {?Drive2-A}
        and     hasPutObject exactly 1 {?PutObject1-A}
        and     hasAt exactly 1 {?At1-A}
        and     hasRightOf exactly 1 {?RightOf1-A}
        and     hasBefore exactly 1 Before1-A
        and     hasBefore exactly 1 Before2-A
        and     hasBefore exactly 1 Before3-A
```

Listing 3: Concept definition of ServeACoffee-A derived from Scenario A

Before1-A, Before2-A and Before3-A relate the activities Drive1-A, Grasp1-A, Drive2-A and PutObject1-A pairwise, analog to the example in Section 2. The definitions of these concepts are also part of the conceptualization.

At1-A and RightOf1-A are conceptualizations of the support of ServeACoffee-A. The concepts are temporally valid throughout the ServeACoffee, we omit the temporal relations for brevity.

The properties hasDrive, hasGrasp, etc. must also be defined, but they can be reused whenever a corresponding activity concept occurs as a component in a concept definition.

3.2 Adapting a Concept to a Positive Example

Modifying an existing concept such that it subsumes a new example is a frequent step in a learning curriculum. In a knowledge-based setting as in OLE, a principled way to perform this step is to conceptualize the example (as shown in the preceding subsection) and then compute the Least Common Subsumer (LCS) of the old concept and the conceptualized example. An effective way to do this for DL languages similar to our sublanguage of OWL has been presented by [Baader et al. 99]. Computing the LCS of two concepts essentially amounts to determining the product tree of the description trees corresponding to the concepts and intersecting the subclass memberships. Our language is somewhat richer because of a larger variety of restrictions on concept fillers (min and max on number restrictions, individuals as fillers) and a taxonomical hierarchy which permits other common parent concepts than intersections of subclass memberships. On the other hand, our concept expressions are not nested, leading to a different graphical representation with named concepts as nodes.

In view of the fact that a formal LCS may not always exist, and following similar ideas as in [Baader et al. 07], we will be content to compute a cognitively plausible "good" common consumer, abbreviated GCS.

The scenarios in Fig. 1 exemplify two different learning situations. One is the creation of a concept for ServeACoffee after experiencing Scenarios A and B, this can be done off-line based on episodes describing *complete* examples. When the robot is asked, however, to perform a ServeACoffee for a guest in Scenario C, the existing concept has to be adapted on-line to be applicable to partially unfolded scene where only the situation before performing the task is available for concept adaptation. We will refer to this task as adaptation to an *incomplete* example. Both tasks, adaptation of a concept to a complete and adaptation to an incomplete example, can be performed by essentially the same GCS computations, while requiring different alignment procedures.

Alignment

In order to perform component-based generalization of two concepts, they must be structurally aligned. We adopt ideas of analogical reasoning [Gentner 86, Gentner et al. 03] where structure mapping has been investigated in detail in a cognitive context. As one of the key principles, analogical structures require a tight agreement of corresponding relations but little or no agreement between corresponding entities. Accordingly, our alignment process follows the property structure and establishes correspondence mainly based on coinciding property names. Differently from analogy construction, however, classes do play a role, and corresponding classes should not be taxonomically distant.

For adaptation to complete examples, we may assume that both conceptual descriptions have identical single roots which necessarily form a corresponding pair. Further correspondences can then be determined by following the property graphs. For adaptation to incomplete examples, alignment may be more difficult. Roughly, it amounts to searching for large coinciding property structures, similar to searching for graph isomorphisms, except for the specific tolerance requirements. Computational aspects of searching for isomorphisms in conceptual graphs are treated thoroughly in [Chein and Mugnier 09].

In general, a conceptualized example cannot be perfectly aligned with an existing concept due to two reasons. One is that reified spatial relations may not have a corresponding partner. The other reason is a different set of robot activities at the top-level, then leading to further discrepancies at lower levels.

In accordance with our learning approach which is basically a procedure to generalize from sufficiently informative examples, we do not consider alignment problems due to missing observations, although this may practically occur. Instead, we assume that the existing concept and the conceptualized example typically contain more detail than relevant for the concept.

In the case of spatial relations we therefore retain corresponding pairs for the adaptation step, but omit excess relations in either the existing concept or the example, assuming that they are irrelevant. It must be noted, however, that it is not obvious in general which spatial relations correspond to each other (and will be retained) and which are deemed irrelevant. The relevance rating introduced in Section 3.1 provides some guidance.

If the example contains a different set of robot activities, however, the alignment fails and the conceptualized example is made a new concept, sibling of the existing concept and with a taxonomical parent assigned to both. This way, a simple form of disjunction is realized. Several learning scenarios presented in Section 4 require the introduction of a new concept because of non-corresponding robot activities.

A detailed investigation and tuning of the alignment procedure must still be performed.

GCS of Corresponding Classes

The GCS computation of two concepts takes the two property graphs of the concepts as input. Hence each concept description typically comprises several class definitions and related permanent knowledge.

The generalization steps are comparable to LCS computation in DLs, however due to our interest in compact descriptions and the use of restricted concept expressions, there are some differences, manifest in more numerous but shorter class definitions. In Table 2 we list key generalization rules of our GCS. The entities refer to the grammatical structure of properties as described in Section 2. LNCS stands for least named common subsumer, i.e. the closest taxonomical parent. MSC is the most specific concept for an individual.

Property Graph 1	**Property Graph 2**	**Element of resulting property graph (GCS)**
<className1>	<className2>	<newClassName> subclassOf LNCS (<className1>, <className2>)
<className1>	<individual2>	<newClassName> subclassOf LNCS (<className1>, MSC(<individual2>))

<individual1>	<individual2>	if <individual1> = <individual2>: <individual1>
		else: <newClassName> subclassOf MSC(<individual1>, <individual2>) universal restriction 'only'

'max' <int1>	'max' <int2>	'max' max(<int1>, <int2>)
'min' <int1>	'min' <int2>	'min' min(<int1>, <int2>)
interval [<int1> <int2>]	interval [<int3> <int4>]	interval [min(<int1>, <int3>), max(<int2>, <int4>)]
<propertyName1>	<propertyName2>	LNCS (<propertyName1>, <propertyName2>)

Table 2: Key generalization rules for the Good Common Subsumer (GCS)

Restrictions by nominal schemas are in principle treated the same way as restrictions by class names. A common new variable name is used for all occurrences of a nominal schema. Furthermore, the new class definition corresponding to the nominal schema has the conjunct 'EquivalentTo:' <nominal schema>. Individuals generalized to a class give rise to a new nominal schema, if they are property fillers for more than one class definition.

After the generalization phase, a cleaning-up process is carried out to avoid unnecessary new class names. This pertains to class definitions where the generalization has led to a class already existing in the ontology.

Part of property graph A	Part of property graph B	GCS
Class: Holding1-A SubClassOf: Holding that hasRobot value trixi and hasPassiveObject exactly 1 Mug1-A	Class: Holding1-B SubClassOf: Holding that hasRobot value trixi and hasPassiveObject exactly 1 Mug1-B	Class: Holding1-AB SubClassOf: Holding that hasRobot value trixi and hasPassiveObject exactly 1 Mug
Class: Mug1-A SubclassOf: Mug that hasColor value white	Class: Mug1-B SubclassOf: Mug that hasColor value yellow	Class: Mug SubclassOf: DrinkingVessel that hasColor exactly 1 Color

Table 3: Illustrating examples for computing the GCS, taken from Scenarios A and B.

To illustrate the process, consider the examples in Table 3. The color values 'white' and 'yellow' are generalized to any color, hence an intermediate concept Mug-AB is changed to Mug in the bottom-up cleaning phase because this definition already exists in the ontology.

Interestingly, the process may lead to property structures where an individual has a conceptual property, for example, [table1 hasSA SA1-AB]. The meaning of SA1-AB, "any sitting area of table1", cannot be expressed this way, of course, and one must reverse the property arrow by using the inverse property: [SA1-AB inverse hasSA value table1].

3.3 Adapting a Concept to a Negative Example

Learning from positive examples as described in the preceding section is conservative in the sense that unnecessary taxonomical generalizations are avoided. Nevertheless, conceptualizations may prove too general, and instructions may inform the robot about situations which are negative examples for an existing concept. There may be several reasons:

(i) The existing taxonomy is too coarse, preventing a necessary differentiation. For example, there may be no class for standard table items such as pepper, salt and decoration.

(ii) There are no useful properties which could help to distinguish the positive and negative examples.

(iii) There are distinguishing features, but the heuristically determined support of a learnt concept has not included this information.

In this section, we sketch two re-learning procedures which resort to recorded episodes in order to adapt a concept to a negative example. This requires, of course, that a learnt concept is linked with its positive examples.

Adding an Affordance Property

It is well-established in robotics to characterize physical or conceptual entities by ways to make use of them, called affordances. For example, if one can sit on an object, it is characterized by the affordance 'sittable'. In the learning situation addressed above in (ii), an affordance property can be added to establish the necessary distinguishing property. To provide a solution where the robot need not invent new names, we introduce the meta-concept 'ActivityName' with all acitivity names of the ontology as possible instances, and postulate that all scene objects have the property 'hasAffordance only ActivityName'. This way, the affordance 'sittable' can be expressed in a concept as the property 'hasAffordance value sit' where 'sit' is an activity name.

In our re-learning situation, the concept which needs differentiation must receive the appropriate affordance property, and all recorded positive examples must be extended accordingly.

Extending the Support

A second way of distinguishing positive from negative examples searches for additional features (properties or scene components) which have not been included in the original conceptualization, but may be recorded in the episodes which have provided the positive examples. For example, if a guest had not been included in the ServeACoffee conceptualization of Scenarios A and B, the impoverished concept would have allowed to place a coffee at any placement area, provoking a negative example. By revisiting

Episodes A and B and extending the support to include a guest, the necessary differentiation can be achieved.

4. Experimental Results

In this section, we describe experimental learning results achieved with different learning tasks in various scenarios. Because of the large data volumes we cannot provide a complete coverage but restrict our documentation to the most interesting generalizations resulting from the GCS.

ServeACoffee Scenarios

The GCS of the conceptualizations of Scenarios A and B produces several generalizations concerning robot destination, placement area, manipulation area, premanipulation area, mug, and spatial relations of the guest. Table 4 shows interesting examples, new concepts are marked with the postfix '-AB'.

	Conceptualization A	Conceptualization B	ServeACoffee-AB
1	... hasToArea value pmaSouth1	... hasToArea value pmaNorth1	... hasToArea only {?PMA1-AB}
2	... hasArea value paWest1	... hasArea value paEast1	... hasArea only {?PA1-AB}

3	maWest1 hasPMA pmaWest1	maEast1 hasPMA pmaEast1	Class: MA1-AB
			EquivalentTo: {?MA1-AB}
			SubclassOf: MA
			that hasPMA only {?PMA1-AB}

4	Class_Instance: [EastWestTable, table1] Properties: [hasSA, SA, saWest1]	Class_Instance: [EastWestTable, table1] Properties: [hasSA, SA, saEast1]	Class: SA1-AB EquivalentTo: {?SA1-AB} SubclassOf: SA that inverse hasSA value table1
5	Class: At1-A SubclassOf: At that hasArea value saWest1 and hasPE exactly 1 Guest	Class: At1-B SubclassOf: At that hasArea value saEast1 and hasPE exactly 1 Guest	Class: At1-AB SubclassOf: At that hasArea only {?SA1-AB} and hasPE exactly 1 Guest
6	Class: RightOf1-A SubclassOf: RightOf that hasFirst value maSouth1 and hasSecond exactly 1 Guest	Class: RightOf1-B SubclassOf: RightOf that hasFirst value maNorth1 and hasSecond exactly 1 Guest	Class: RightOf1-AB SubclassOf: RightOf that hasFirst only {?MA1-AB} and hasSecond exactly 1 Guest

Table 4: Generalizations by combining conceptualizations of Scenarios A and B

Comments to Table 4:

Lines 1 and 2: After applying ServeACoffee to two different premanipulation areas and placing areas of table1, respectively, the concept is generalized to apply to all such areas of table1. The corresponding area concepts are represented by nominal schemas because the same instances occur as fillers of several properties within the support of ServeACoffee.

Line 4: The new concept for representing all sitting areas of table1 must be related to table1 with the inverse of the existing property hasSA.

Lines 5 and 6: The spatial relations involving a guest now refer to the same sitting areas and manipulation areas of table1 which are also the destination of ServeACoffee.

In Scenario C, the learnt concepts must be applied to a new situation where the guest sits at another table (table2) in a sitting area distinct from any previously encountered sitting areas. The property graph for this situation is shown in Fig. 3.

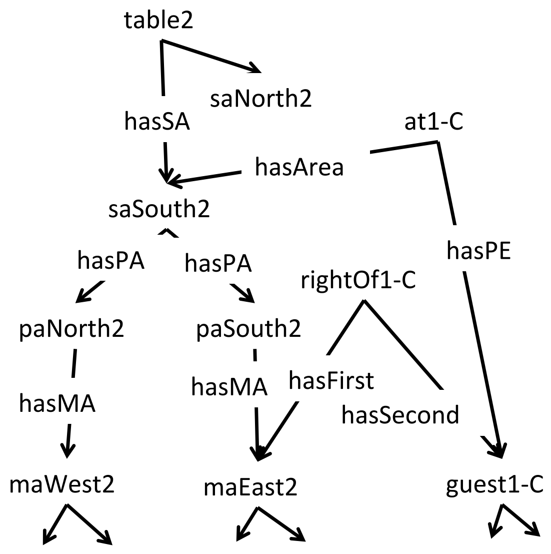


Fig. 3: Property graph of Scenario C before performing ServeACoffee

After conceptualization, it is correctly aligned with the property graph of ServeACoffee-AB and the adapted concept SeveACoffee-ABC is determined, with key generalizations shown in Table 5.

	ServeACoffee-AB	Conceptualization C	ServeACoffee-ABC
1	Class: SA1-AB EquivalentTo: {?SA1-AB} SubclassOf: SA that inverse hasSA value table1	Class_Instance: [Table, table2] Properties: - [hasSa, SA, saSouth2]	Class: SA EquivalentTo: {?SA1-ABC} SubclassOf: SA that inverse hasSA only Table
2	Class: At1-AB SubclassOf: At that hasArea only {?SA1-AB} and hasPE exactly 1 Guest	Class: At1-C SubclassOf: At that hasArea value saSouth2 and hasPE exactly 1 Guest	Class: At1-ABC SubclassOf: At that hasArea only {?SA1-ABC} and hasPE exactly 1 Guest
3	Class: RightOf1-AB SubclassOf: RightOf that hasFirst only {?MA1-AB} and hasSecond exactly 1 Guest	Class: RightOf1-C SubclassOf: RightOf that hasFirst value maEast2 and hasSecond exactly 1 Guest	Class: RightOf1-ABC SubclassOf: RightOf that hasFirst only {?MA1-ABC} and hasSecond exactly 1 Guest

Table 5: Generalizations by combining the conceptualization of Scenarios A and B with the conceptualization of Scenario C.

Comments to Table 5:

Line 1: The sitting area saSouth2 of table2 in Conceptualization C causes a generalization of table1 to an arbitrary table.

Lines 2 and 3: The spatial relations involving a guest now refer to sitting areas and manipulation areas of any table which is the destination of ServeACoffee.

ServeACoffee-ABC can be paraphrased as follows: Move to counter1, grasp a mug, move to a premanipulation area belonging to a manipulation area right of the guest, place the mug on the placing area belonging to the sitting area where the guest is located.

Deal-with-obstacles Scenarios

In these scenarios, illustrated in Fig. 4, it is assumed that the robot has learnt a ServeACoffe as described in the previous subsection. The robot has again the task to serve a coffee to a guest west of table1, but encounters an obstacle in the southern manipulation area from which a coffee is normally served.

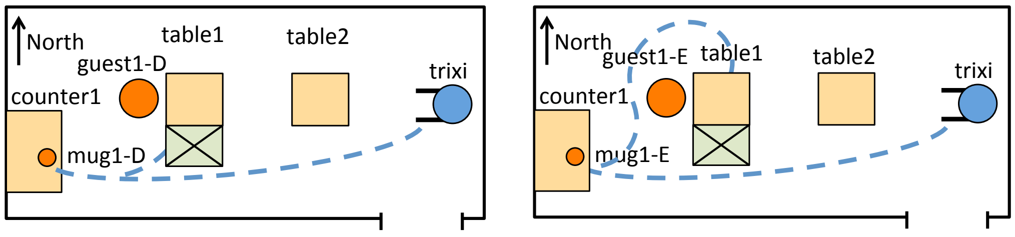


Fig. 4: a) Scenario D b) Scenario E

In Scenario D, this obstacle is a person, and the robot is instructed to wait until the person has moved away. The robot follows the instruction and learns a new concept ServeACoffeeBlocked-D. In Scenario E, a sidetable blocks the manipulation area, and the robot tries to apply the learnt concept, generalizing it to subsume any physical entity as obstacle. As the robot waits for the sidetable to move away, the instructor tells the robot not to wait in this case but to move to the northern premanipulation area and place the coffee on the western placement area.

Both learning situations feature a negative example for an existing concept and require a structurally modified new concept. In Scenario D, the robot first follows a plan based on the existing concept ServeACoffee which fails because of the obstacle. The robot continues following the instructions, records the episode and after conceptualization creates the new concept ServeACoffeeBlocked-D. Parts of the property graph are shown in Fig. 5. The temporal relations, omitted for graphical clarity, require that a PutObject1-D is carried out after the blocking event At1-D has terminated.

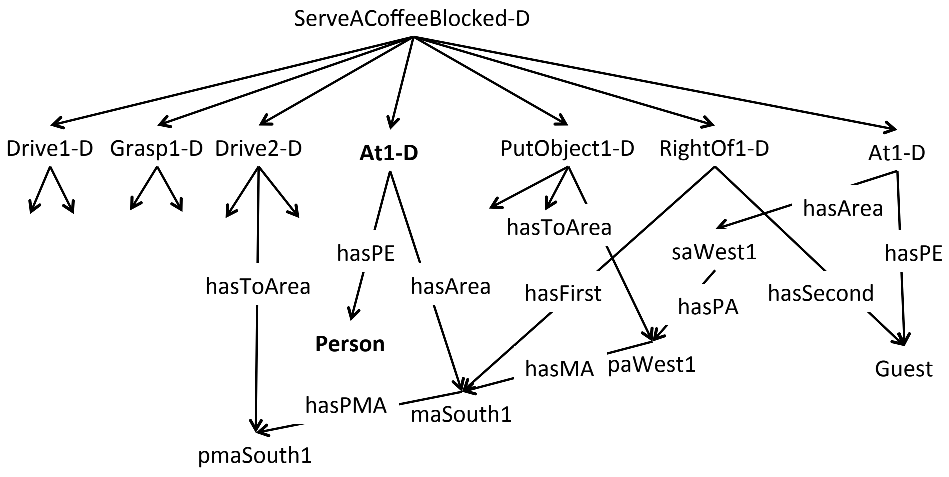


Fig. 5: Conceptualization of Episode D. PutObject1-D begins when At1-D ends.

In Scenario E, the robot first applies ServeACoffee unaware of the obstacle and, after failure, adapts ServeACoffeeBlocked-D by generalizing Person to PhysicalEntity (PE) in order to cover the new situation with the sidetable as obstacle (procedure presented in Section 3.2). Following the instructions, the robot does not wait but proceeds to pmaNorth1 at the north of the table and performs the putObject. The conceptualization of this episode results in ServeACoffeeBlocked-E, parts of the property graph are shown in Fig. 6. The three concepts, ServeACoffee, ServeACoffeeBlocked-D, and ServeACoffeeBlocked-E are preserved, while the concept with the tentative generalization of Person to PhysicalEntity is abandoned.

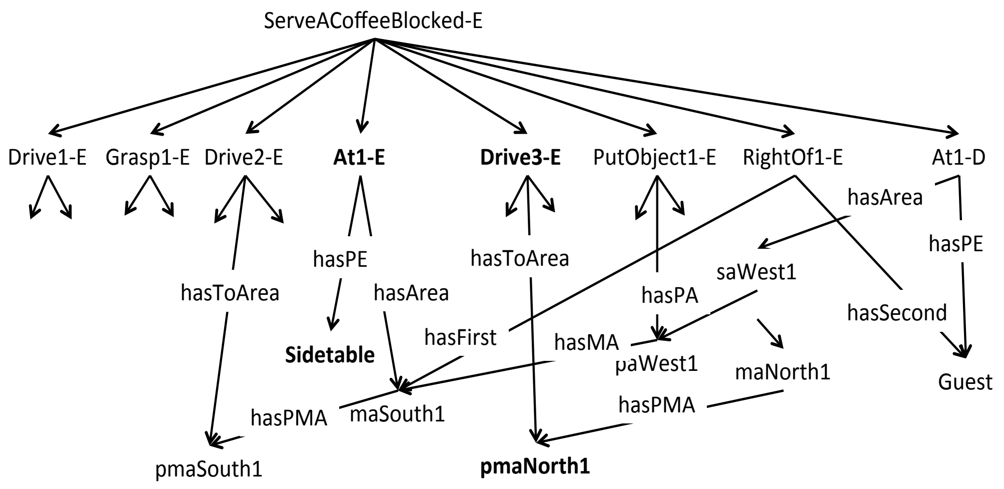
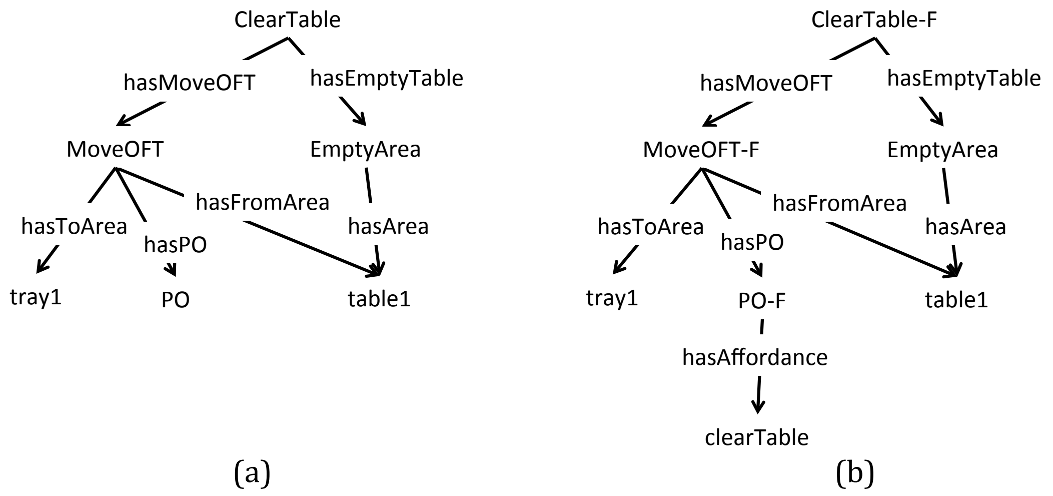


Fig. 6: Conceptualization of Episode E. An additional Drive3-E begins after observing At1-E. PutObject1-E begins after Drive3-E.

Clear-table-smartly Scenarios

These scenarios provide learning situations for the task of clearing a restaurant table. Typically, all used dishes have to be cleared while certain objects such as a pepper-and-salt dish and table decoration must not be cleared. We assume that the robot has initially been instructed or has learnt that ClearTable amounts to taking all objects from the table. Hence ClearTable has a property graph as shown in Fig. 7a. MoveOFT (MoveObjectFromTo) effects the removal of a single object, hasMoveOFT is not cardinality-restricted and hence can occur several times. Temporal constraints ensure that EmptyArea concludes the process, temporal relations are omitted in the figure for clarity. PO (PassiveObject) subsumes all non-living physical entities.

In Scenario F, the robot has to clear a table with a plate, knife and vase, and is instructed not to remove the vase. Since ClearTable includes a vase, the robot considers this situation a negative example and must refine the concept to exclude it. There is no ontological distinction between the physical entities of the positive examples on one side and the negative example, therefore the robot refines PO by the affordance 'hasAffordance value clearTable' as described in Section 3.3, and extends the positive examples for PO, plate and knife, accordingly. Note that this happens in an ongoing episode. In future situations, using ClearTable-F, the robot will not clear any object without the affordance property. It is a matter of the instructor point out omissions as positive examples, so that the robot can add missing affordance properties.



Legend: MoveOFT = MoveObjectFromTable
PO = PassiveObject PO-F = revised concept of PassiveObject

Fig. 7: Property graph of ClearTable before (a) and after (b) learning in Scenario F

5. Conclusions

We have presented several methods for learning or refining conceptual descriptions based on examples, formalized within an OWL-based knowledge representation framework. While the principles of conceptual learning are well understood for many years, our approach deals with several new aspects. Firstly, our representation formalism is used by an integrated robot system operating in real-world scenarios, collecting experiences in a robot memory, and sharing a common ontology for recording experiences, learning, planning, scene interpretation, and other reasoning tasks. Hence the robot can make immediate use of learning results. Representations include quantitative spatial and temporal information for real-world grounding.

Secondly, we have shown that concepts and episodes represented in a restricted OWL 2 dialect can be conveniently transformed into property graphs as a basis for structural matching. This way, ideas of analogical reasoning can be realized, allowing complex conceptual descriptions to be applied to new situations.

Thirdly, our approach considers realistic learning scenarios where instructions may be vague and the robot may be uncertain about relevant scene components. We therefore propose relevance analysis based on the semantic distance between contextual scene components and robot activities.

Finally, due to the experiences stored in the robot memory, the framework can be smoothly extended from learning based on single examples to learning based on a large body of experiences with statistically relevant features.

References

[Baader et al. 99]
Baader, F,; Küsters, R.; Molitor, R.: Computing Least Common Subsumers in Description Logics with Existential Restrictions. In Proc. IJCAI-99, Vol. 1, Morgan Kaufmann, 1999, 96-101.

[Baader et al. 07]
Baader, F.; Sertkaya, B.; Turhan, A.-Y.: Computing the least common subsumer w.r.t. a background terminology. Journal of Applied Logic, 5(3), 2007

[Bohlken et al. 11]
Bohlken, W.; Neumann, B.; Hotz, L.; Koopmann, P.; Ontology-Based Realtime Activity Monitoring Using Beam Search. In: Crowley, J.L. et al. (eds.): ICVS 2011, LNCS 6962, Springer, Heidelberg (2011), 112-121

[Bohlken et al. 13]
Bohlken, W.; Koopmann, P.; Hotz, L.; Neumann, B.: Towards Ontology-Based Realtime Behaviour Interpretation. In Guesgen, H.W. and Marsland, S. (eds.): Human Behavior Recognition Technologies: Intelligent Applications for Monitoring and Security, IGI Global, 2013, 33-64

[Chein and Mugnier 09]
Chein, M.; Mugnier, M.-L.: Graph-based Knowledge Representation. Springer, 2009.

[Cohen and Feigenbaum 82]
Cohen, W.W.; Feigenbaum, E.A. (eds.): The Handbook of Artificial Intelligence, Vol. 3., William Kaufmann, 1982, 368-369.

[Cohen and Hirsh 94]
Cohen, W.W.; Hirsh, H.: Learning the CLASSIC Description Logic: Theoretical and Experimental Results. In Proc. Principles of Knowledge Representation and Reasoning (KR-94), Morgan Kaufmann, 1994, 121-133

[Grosof et al. 03]
Grosof, B.N.; Horrocks, J.; Volz, R.; Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. . In Proc. of the 12th Int. World Wide Web Conference (WWW 2003), 2003, 48–57

[Günther et al. 12]
Günther, M.; Hertzberg, J.; Mansouri, M.; Pecora, F.; Saffiotti, A.: Hybrid reasoning in perception: A case study. In Proc. SYROCO. Dubrovnik: IFAC., 2012

[Hitzler et a. 08]
Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y.: Semantic Web, Springer, 2008

[Holyoak et al. 01]
Holyoak, K.J.; Gentner, D.; Kokinov, B.N.: Introduction: The Place of Analogy in Cognition. In: Gentner, D.; Holyoak, K.J.; Kokinov, B.N. (eds.), The Analogical Mind,The MIT Press, 2001, 1-20

[Kean and Costello 01]
Keane, M.T., Costello, F.: Setting Limits on Analogy: Why Conceptual Combination is not Structural Alignment. In: Gentner, D.; Holyoak, K.J.; Kokinov, B.N. (eds.), The Analogical Mind,The MIT Press, 2001, 287-312

[Krötzsch et al. 11]
Krötzsch, M.; Maier, F.; Krisnadhi, A.A.; Hitzler, P.: A Better Uncle for OWL. In: In Proc. of the World Wide Web Conference (WWW 2011), 2011, 645–654

[Lehmann 09]
Lehmann, J.: DL-Learner: Learning Concepts in Description Logics.Journal of Machine Learning Research (JMLR) 10, 2009, 2639-2642

[Lehmann and Hitzler 07]
Lehmann, J.; Hitzler, P.: A Refinement Operator Based Learning Algorithm for the ALC Description Logic. In: ILP 2007, Springer LNCS 4894, 2007, 147-160

[Lutz 03]
Lutz, C.: Description Logics with Concrete Domains - a Survey. In: Advances in Modal Logic, Vol. 4, 2003, 265-296

[McCarthy and Hayes 69]
McCarthy, J.; Hayes, P.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie (eds.), Machine Intelligence, Vol. 4, Edinburgh University Press, 1969, 463–502.

[Mitchell 82]
Mitchell, T.M.: Generalization as search. Artificial Intelligence 18 (2), 1982, 203–226

[Rockel 13]
Rockel, S.; Neumann, B.; Zhang, J.; Dubba, K.S.R.; Cohn, A.G.; Konecny, S.; Mansouri, M.; Pecora, F.; Saffioti, A.; Günther, M.; Stock, S.; Hertzberg, J.; Tome, A.M.; Pinho, A.; Seabra Lopes, L.; von Riegen, S.; Hotz, L.: An Ontology-based Multi-level Robot Architecture for Learning from Experiences. In: Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium 2013, March 25-27, Stanford University, USA

[Wilson et al. 01]
Wilson, W.H.; Halford, G.S.; Gray, B.; Phillips, S.: The STAR-2 Model for Mapping Hierarchically Structured Analogs. In: Gentner, D.; Holyoak, K.J.; Kokinov, B.N. (eds.), The Analogical Mind,The MIT Press, 2001, 125-160

[Winston 75]
Winston, P.H.: Learning Structural Descriptions from Examples. In Winston, P.H., The Psychology of Computer Vision, McGraw Hill, 1975