A Robot Waiter Learning from Experiences

Bernd Neumann[1], Lothar Hotz[2], Pascal Rost[2], and Jos Lehmann[1]

[1] Department of Informatics,
University of Hamburg, Germany
{neumann,jlehmann}@informatik.uni-hamburg.de
[2] Hamburger Informatik Technology Center, Department of Informatics,
University of Hamburg, Germany
{hotz,7rost}@informatik.uni-hamburg.de

Abstract. In this contribution, we consider learning tasks of a robot simulating a waiter in a restaurant. The robot records experiences and creates or adapts concepts represented in the web ontology language OWL 2, extended by quantitative spatial and temporal information. As a typical task, the robot is instructed to perform a specific activity in a few concrete scenarios and then expected to autonomously apply the conceptualized experiences to a new scenario. Constructing concepts from examples in a formal knowledge representation framework is well understood in principle, but several aspects important for realistic applications in robotics have remained unattended and are addressed in this paper. First, we consider conceptual representations of activity concepts combined with relevant factual knowledge about the environment. Second, the instructions can be coarse, confined to essential steps of a task, hence the robot has to autonomously determine the relevant context. Third, we propose a "Good Common Subsumer" as opposed to the formal "Least Common Subsumer" for the conceptualization of examples in order to obtain cognitively plausible results. Experiments are based on work in Project RACE[3] where a PR2 robot is employed for recording experiences, learning and applying the learnt concepts.

Keywords: Machine Learning, ontology, robot activities

1 Introduction

Learning capabilities are essential prerequisits for robots to become useful in complex real-world domains. 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.

[3] This work is supported by the RACE project, grant agreement no. 287752, funded by the EC Seventh Framework Program theme FP7-ICT-2011-7.

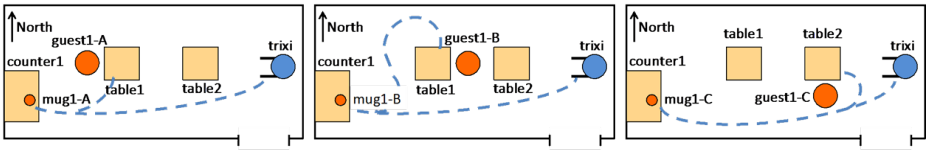
To convey a first understanding of the learning tasks investigated in this paper, consider the three scenarios sketched in Figure 1. In Scenarios A and B, the robot - here called "trixi" - receives detailed instructions how to serve a coffee to a guest and learns that these activities constitute a "ServeACoffee":

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."

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."



(a) Scenario A (b) Scenario B (c) Scenario C

Fig. 1: Scenarios

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 [18]. 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 (see in the following), (ii) research into Cognitive Robotics, connecting symbolic high-level representations with quantitative low-level operations (see [13, 7, 3] and [15]), and (iii) advances in learning and reasoning models in Cognitive Sciences (see [9, 17, 10] and [15]).

Our choice of a knowledge representation framework is motivated by previous work on high-level representations of events and activities [3] and the recent development of the standardized knowledge representation language OWL 2 supporting such representations. Furthermore, extensions of OWL 2 in terms of quantitative representations for spatial and temporal information [7] have made it possible to represent robot activities coherently from high-level symbolic ac-

tivity concepts down to low-level quantitative commands and sensor input. As a drawback, standard OWL representations cannot yet represent the sameness of entities typically occurring in the compositional structures for activity models. Fortunately, recent work by the group of Hitzler [8] 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.

Learning from examples within a DL framework has been investigated before [6, 12], 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 [14] can be applied to define the space of concepts which correctly classify positive and negative examples. In our approach, concept learning has been designed as a cognitively plausible strategy for ontology evolution without bookkeeping of version space boundaries. This is achieved by a learning curriculum essentially based on careful generalizations and avoiding disjunctive expressions. Correspondences between concepts and examples are established following the structure-mapping theory of Cognitive Science [9].

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 learning 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. Future work will also cover research on the formal properties of the presented algorithm, such as consistency, completeness, or complexity.

2 Knowledge Representation Conventions

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 [16]. For textual concept representations, we use the Manchester Syntax[4] in this paper. 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.

Listing 1.1: Class definitions of PlaceObject1-A and its property fillers

```
Class:  PlaceObject1-A
SubclassOf:     PlaceObject
```

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

```
        that    hasHolding exactly 1 {?Holding1-A}
        and     hasOn 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
```

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 Figure 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).

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).

Listing 1.2: Syntax of restricted grammar for concept definitions

```
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> '}'
```

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 [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 Schemas.

The assertional knowledge of a robot comprises episodes which are stored as experiences in the robot memory. An episode consists of dynamic factual 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. Assertional knowledge is stored in triplets relating individuals via properties to other individuals, as customary for DL languages.

3 Concept Formation and Adaption

As illustrated by the scenarios shown in Figure 1, the OLE approach to concept learning includes a supervised learning task where an instructor provides a name for a new concept (ServeACoffee in Scenarios A and B) and its essential components, 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, finding a common subsumer for two corresponding concepts. In effect, this approach allows to formulate learning solely based on conceptual expressions, known as the "single-representation trick" [5]. We also sketch a procedure for refining a concept in order to exclude a negative example.

3.1 Conceptualizing Examples

```
Conceptualize Episode
Input:   - Ontology
         - Episode, background knowledge (recorded by the robot)
         - Robot activities from episode constituting a new concept
           (specified by instructor)
Process: - Determine relevant assertions from episode
         - Conceptualize assertions
         - Create new class definitions
Output:  - New class definitions updating the ontology
```

Conceptualizing a robot activity carried out in a scenario amounts to establishing a generic description for the ontology of the robot with the purpose that it can be used as a template for future robot activities and other cognitive tasks. The conceptualization procedure is structured as shown above.

In the following, we describe an approach for extracting the assertions relevant for a concept from an episode E. Let A be the high-level robot activities specified by the instructor as constituents of an instance c of the new concept C. We define the *support* of c as the assertions involving A, and all assertions in E connected to A. This can be visualized by a *property graph* with individuals as nodes, c as root node, and properties as directed edges. Figure 2 shows the property graph for placeObject1-A, using definitions of Listing 1, among others. As constituents A, 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 start 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.

We define the *direct support* of c 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. In Figure 2, the nodes of the direct support of placeObject1-A are shown in bold, all other nodes are indirect support. In general, an episode may contain many details experienced by the robot and connected to the direct support, e.g. by spatial or temporal relations.

To filter out irrelevant information, we make use of a heuristic which measures *relevance* in terms of the smallest semantical distance of a node to any of the nodes of the direct support. Distance is determined by counting the number of property edges irrespective of their direction. 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. Figure 2 shows that the impor-

tant 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.

The next step in the conceptualization procedure is to transform relevant assertional knowledge into conceptual descriptions. We take a strictly conservative approach regarding generalizations and let a conceptualization represent exactly the same activities, however at an arbitrary time (modulo some tolerance regarding durations). Hence given the same environment and the updated ontology, the robot will be able to carry out the learnt activities. Permanent assertional knowledge remains assertional and is associated with the new concept via the relevant property graph. The exact rules for transforming assertional knowledge into conceptual representations are described in [15].

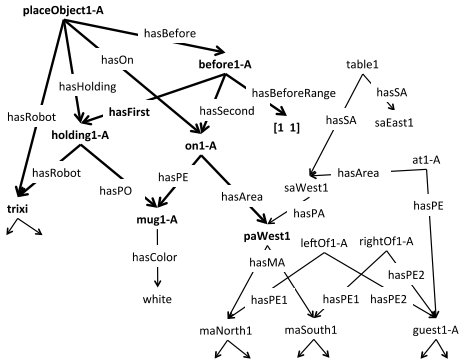


Fig. 2: Property graph for an instance of the concept PlaceObject with related permanent knowledge and observations.

3.2 Adapting a Concept to a Positive Example

Modifying an existing concept such that it covers a new example is a frequent step in a learning curriculum. This can be done by conceptualizing the example (as described in the preceding subsection) and then computing the Least Common Subsumer (LCS) of the old concept and the conceptualized example. An effective way to do this for Description Logics (DL) similar to our sublanguage of OWL has been presented in [1]. 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. In view of the fact that a formal LCS may not always exist, and following similar ideas as in [2], we will be content to compute a cognitively plausible "good" common consumer, abbreviated GCS.

The adaptation procedure has the following structure:

```
Input:   - Ontology
         - Old concept
         - Positive example (complete or incomplete)
Process: - Conceptualize example
         - Align example to old concept
         - Compute "Good Common Subsumer" (GCS)
Output: - New concept subsuming conceptualized example and old concept
        - Updated ontology
```

The scenarios in Figure 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 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 [9] 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 [4].

If a conceptualized example cannot be perfectly aligned with an existing concept because of non-corresponding reified spatial or temporal relations, these are treated as irrelevant and omitted. If a discrepancy is due to non-corresponding robot activities, however, alignment fails and the conceptualized example is made a new concept.

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 1, 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.

Table 1: Key generalization rules for the Good Common Subsumer (GCS)

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>)

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 a Nominal Schema has the conjunct 'EquivalentTo:' <ominalSchema>. 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. Illustrating examples for GCS computations are provided in Section 4.

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:

1. 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.
2. There are no useful properties which could help to distinguish the positive and negative examples.
3. 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. Hence, if a concept needs refinement, it must receive the appropriate affordance property, and all recorded positive examples must be extended accordingly.

Extending the Support As a second way of distinguishing positive from negative examples one can search 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.

4.1 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 2 shows interesting examples, new concepts are marked with the postfix '-AB'.

Comments to Table 2:

Rows 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

Table 2: Generalizations by combining conceptualizations of Scenarios A and B

	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

Nominal Schemas because the same instances occur as fillers of several properties within the support of ServeACoffee.

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

Rows 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 Figure 3.

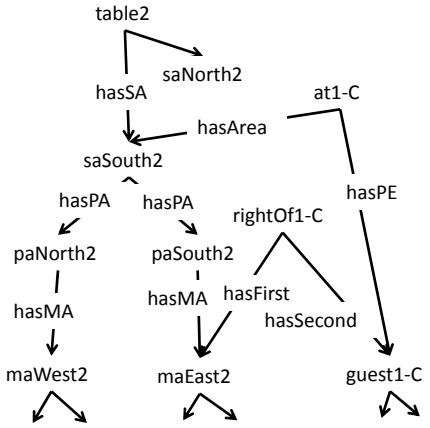


Fig. 3: Property graph of Scenario C before performing ServeACoffee

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 3.

Table 3: Generalizations by combining the conceptualization of Scenarios A and B with the conceptualization of Scenario C.

	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

Comments to Table 3:

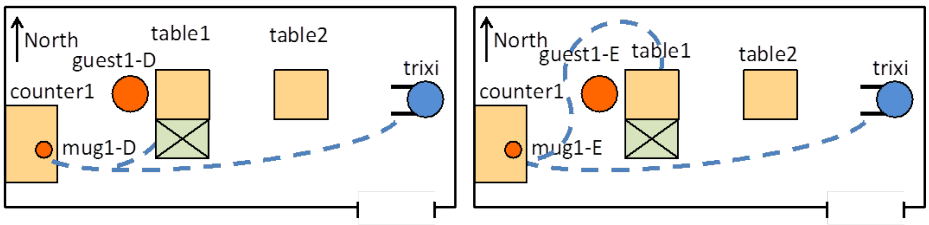
Row 1: The sitting area saSouth2 of table2 in Conceptualization C causes a generalization of table1 to an arbitrary table.

Rows 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.

4.2 Deal-with-obstacles Scenarios

In these scenarios, illustrated in Figure 4, it is assumed that the robot has learnt a ServeACoffee 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.



(a) Scenario D (b) Scenario E

Fig. 4: Scenarios D and 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 Figure 5. The temporal relations, omitted for graphical clarity, require that a PutObject1-D is carried out after the blocking event At1-D has terminated.

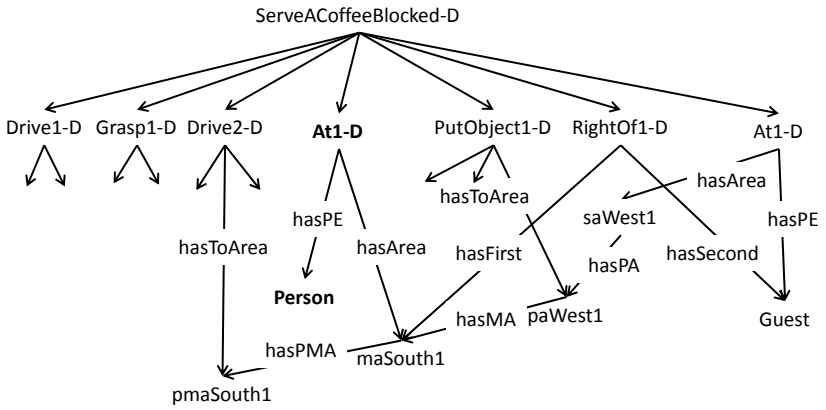


Fig. 5: Conceptualization of Episode D. PutObject1-D begins when At1-D ends.

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 Figure 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.

In a further experimental scenario, Clear-table-smartly, the robot learns to clear all items from a table except for a vase. The initial concept ClearTable lets the robot clear all passive objects (POs). When the robot clears the vase, the

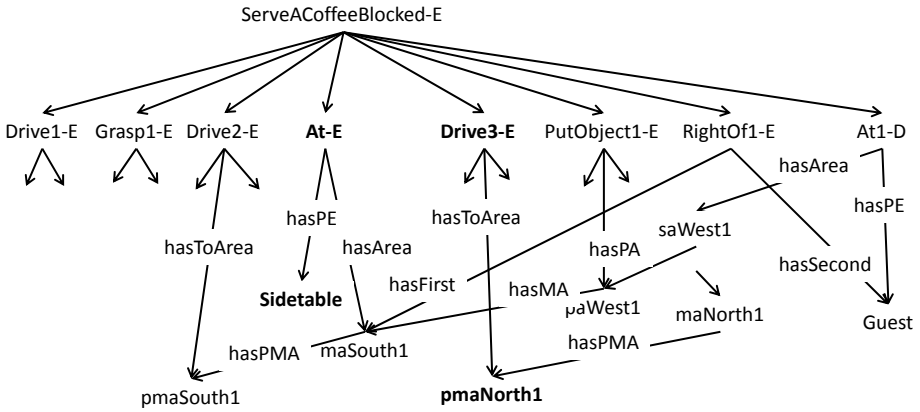


Fig. 6: Conceptualization of Episode E. An additional Drive3-E begins after observing At1-E. PutObject1-E begins after Drive3-E.

instructor marks this as a negative example. To adapt to the negative example, the robot refines PO in the concept ClearTable by the affordance property 'hasAffordance value clearTable' and extends the positive examples accordingly. For details see [15].

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. Thus, 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 have therefore proposed 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

1. Baader, F., Küsters, R., Molitor, R.: Computing Least Common Subsumers in Description Logics with Existential Restrictions. In: In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99). vol. 1, pp. 96–101. Morgan Kaufmann (1999)
2. 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)
3. Bohlken, W., Koopmann, P., Hotz, L., Neumann, B.: Towards ontology-based real-time behaviour interpretation. In: Guesgen, H., Marsland, S. (eds.) Human Behavior Recognition Technologies: Intelligent Applications for Monitoring and Security. pp. 33–64. IGI Global (2013)
4. Chein, M., Mugnier, M.L.: Graph-based Knowledge Representation. Springer (2009)
5. Cohen, W., Feigenbaum, E.: The Handbook of Artificial Intelligence, vol. 3. William Kaufmann (1982)
6. Cohen, W., Hirsh, H.: Learning the classic description logic: Theoretical and experimental results. In: Proc. Principles of Knowledge Representation and Reasoning (KR-94) (1994)
7. Günther, M., Hertzberg, J., Mansouri, M., Pecora, F., Saffiotti, A.: Hybrid reasoning in perception: A case study. In: Proc. SYROCO. IFAC, Dubrovnik (Sep 5-7 2012)
8. Hitzler, P., Krötzsch, M., Rudolph, R., Sure, Y.: Semantic Web (2008)
9. Holyoak, K., Gentner, D., Kokinov, B.: Introduction: The place of analogy in cognition. In: Gentner, D., Holyoak, K., Kokinov, B. (eds.) The Analogical Mind. pp. 1–20. The MIT Press (2001)
10. Keane, M., Costello, F.: Setting limits on analogy: Why conceptual combination is not structural alignment. In: Holyoak, K., Gentner, D., Kokinov, B. (eds.) The Analogical Mind. pp. 287–312. The MIT Press (2001)
11. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for owl. In: In Proc. of the World Wide Web Conference (WWW 2011). pp. 645–654 (2011)
12. Lehmann, J.: Dl-learner: Learning concepts in description logics. Journal of Machine Learning Research (JMLR) 10 (2009)
13. Lutz, C.: Description logics with concrete domains - a survey. Advances in Modal Logic 4 (2003)
14. Mitchell, T.: Generalization as search. Artificial Intelligence 18(2), 203–226 (1982)
15. Neumann, B., Hotz, L., Günter, A.: Learning robot activities from experiences: An ontology-based approach. Tech. Rep. TR FBI-HH-B-300/13, University of Hamburg, Department of Informatics Cognitive Systems Laboratory (2013)
16. Rockel, S., Neuman, B., Zhang, J., Dubba, K.S.R., Cohn, A.G., Š. Konečný, Mansouri, M., Pecora, F., Saffiotti, A., Günther, M., Stock, S., Hertzberg, J., Tomé, A.M., Pinho, A.J., Lopes, L.S., 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. Stanford (USA) (March 2013)
17. Wilson, W., Halford, G., Gray, B., Phillips, S.: The star-2 model for mapping hierarchically structured analogs. In: Holyoak, K., Gentner, D., Kokinov, B. (eds.) The Analogical Mind. pp. 125–160. The MIT Press (2001)
18. Winston, P.: The Psychology of Computer Vision, chap. Learning structural descriptions from examples, pp. 157–209. McGraw-Hill, New York (1975)