



Learning Compositional Robot Activities from Examples

Bernd Neumann, Lothar Hotz, Pascal Rost

Cognitive Systems Laboratory, University of Hamburg



Main Topics of RACE


RACE = "Robustness by Autonomous Competence Enhancement"

"Exploitation of past robot experiences for improving future robot performance by an integrated sub-symbolic/symbolic approach."

non-symbolic

- **Integration by OWL ontology + semantic attachments via constraint solver**
- **Focus on example-based learning of robot activities represented in compositional hierarchies (EBCL)**




 **Experimental Restaurant Domain**

Robot has basic skills (moving, grasping, perceiving)

- learns tasks of a restaurant waiter from coarse instructions
- improves competence based on experiences
- adapts concepts to new situations

Vision of Woolworth robot ...

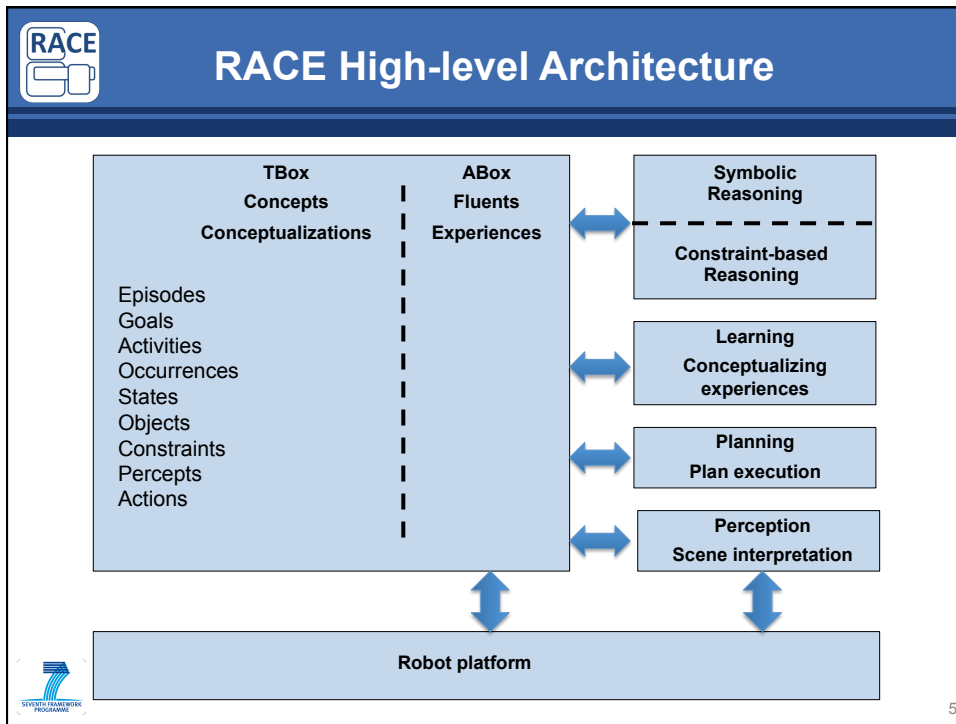
 3

 **PR2 in Mock-up Restaurant**

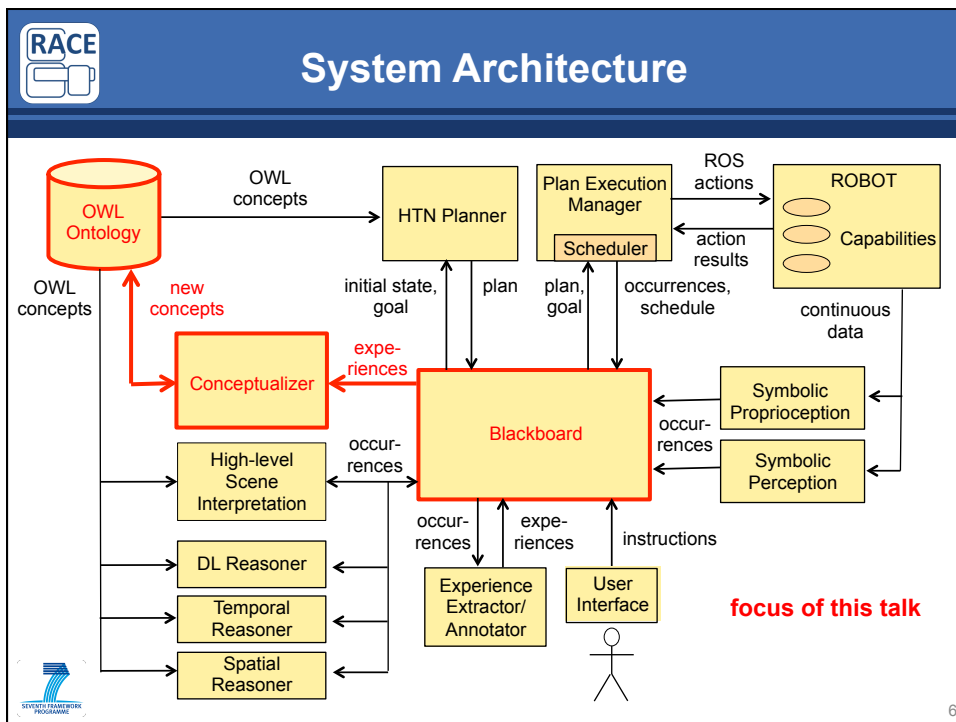


The image shows a PR2 robot in a mock-up restaurant environment. The robot is positioned in the center, facing a table with chairs. The room contains several tables and chairs, a whiteboard, and a clock on the wall. The robot has 'PR2' and 'ROS' labels on its body.


 4



5





6



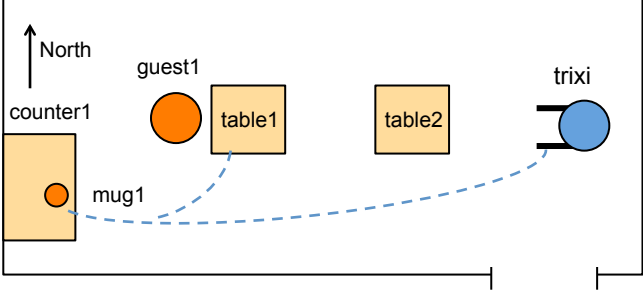
Agenda

- **A learning scenario**
- **Knowledge representation conventions**
- **Conceptualizing examples – relevance analysis**
- **Constructing a Good Common Subsumer**
- **Other learning scenarios**
- **Conclusions**


7




Learning to Serve a Coffee (A)



"Move to counter1, grasp mug1, move to **south** of table1, place mug1 at placement area **west** – this is a ServeACoffee"

Concept creation for ServeACoffee activity from a single example


8

RACE

Learning to Serve a Coffee (B)

North
counter1
mug2
table1
guest2
table2
trixi

"Move to counter1, grasp mug2, move to *north* of table1, place mug2 at placement area *east* – this is also a ServeACoffee"

➔ Generalization of ServeACoffee concept to cover both examples

SEVENTH FRAMEWORK PROGRAMME

9

RACE

Learning to Serve a Coffee (C)


North
counter1
mug3
table1
guest3
table2
trixi

"Do a ServeACoffee to guest3 at table2"

➔ Further generalization of ServeACoffee concept for application to a new situation

SEVENTH FRAMEWORK PROGRAMME


10




Problems

Main problems addressed in this talk:

- **What is relevant for a new concept?**
"Qualification problem" in AI (McCarthy & Hayes 69)
- **What are reasonable generalizations?**
- **Under which conditions does EBCL converge to an intended target behaviour?**


11





Representing Activity Concepts in OWL

ServeACoffee \sqsubseteq RobotActivity

- hasMove1 ∇ (Move □ hasToArea {counter1})
- hasGrasp =1 (Grasp □ hasObject EXACTLY 1
 (Mug □ hasColor {white})
 □ hasHolding EXACTLY 1
 (Holding □ hasObject EXACTLY 1 Mug)
- hasMove2 ∇ (Move □ hasToArea ManipulationArea)
- hasPlace =1 (Place □ hasObject EXACTLY 1 Mug
 □ hasHolding EXACTLY 1
 (Holding □ hasObject EXACTLY 1 Mug)
 □ hasToArea PlacingArea
 □ hasOn EXACTLY 1
 (On □ hasObject EXACTLY 1 Mug
 □ hasArea EXACTLY 1 PlacingArea)
- hasDuration =1 Int \leq 120

hasMove1, hasGrasp, hasMove2, hasPlace \sqsubseteq hasPart


12



Restricted Syntax for Example-based Learning


```

<conceptDefinition> ::= <conceptName> '⊆' <conceptName> { '∩' <roleRestriction> }


<roleRestriction> ::= <roleName> '∇' <conceptName> |
                    <roleName> [<numberRestriction>] <conceptName> |
                    <roleName> <individualList> |
                    <roleName> <dataType>


<individualList> ::= '{' {<individual>} '}'

```



- no nested expressions
- role restrictions by named concepts or primitives
- no OR, no negation
- use of SWRL rules to express sameness
- allows easy correspondence with examples


13




Example of Restricted Syntax

```

ServeACoffee ⊆ RobotActivity
...
□ hasGrasp =1 (Grasp
  □ hasObject EXACTLY 1
    (Mug □ hasColor {white}))
...

```




```


ServeACoffee ⊆ RobotActivity
...
□ hasGrasp Grasp1-A
...

Grasp1-A ⊆ Grasp
□ hasObject Mug1-A
□ hasAgent {trixi}

Mug1-A ⊆ Mug
□ hasColor {white}
...

```


14

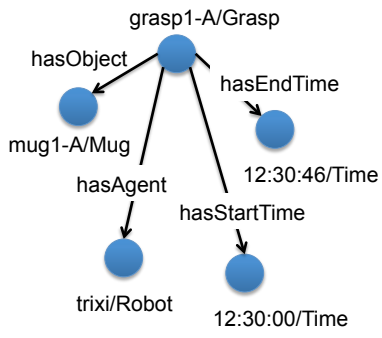



Assertional Knowledge (ABox)


Syntax:
 $\{IN: CN \wedge (RN_1 CN_1 IN_1) \wedge (RN_2 CN_2 IN_2) \dots \}$

Example:
 $\{grasp1-A: Grasp$
 $\wedge (hasAgent Robot trixi)$
 $\wedge (hasObject Mug mug1-A)$
 $\wedge (hasStartTime Time 12:30:00)$
 $\wedge (hasEndTime Time 12:30:46)\}$

Graphical representation:






15



Learning Curriculum

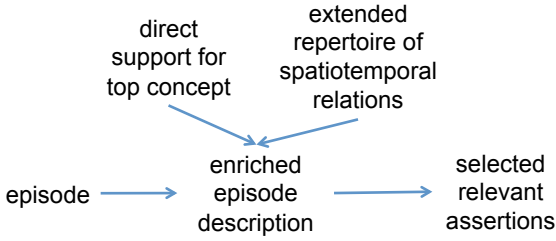
	Episode 1	Episode 2	Episode N
<i>Relevance Analysis</i>	↓	↓	↓
	Example 1	Example 2	Example N
<i>Conceptualization</i>	↓	↓	↓
	Concept Support 1	Concept Support 2	Concept Support N
<i>Merging</i>	↓	↓	↓
	↓	Concept Support 1 .. 2	Concept Support 1 .. N


16




Relevance Analysis


- **Episodes contain**
 - background knowledge
 - coarse instructions
 - robot activities
 - robot observations
- **Heuristic for dumb robot: All assertions in "close" spatiotemporal vicinity of the robot's activities are relevant**



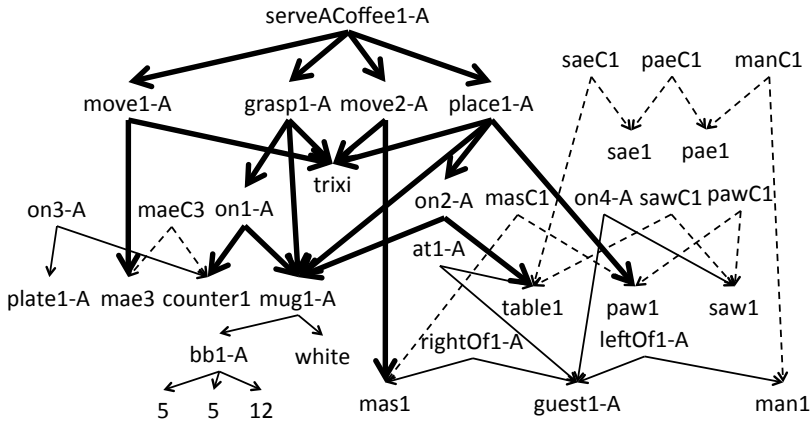
```

graph LR
    episode --> enriched[enriched episode description]
    enriched --> selected[selected relevant assertions]
    direct[direct support for top concept] --> enriched
    extended[extended repertoire of spatiotemporal relations] --> enriched
    
```


17




Assertion Graph for an Example



```

graph TD
    serve[serveACoffee1-A] --> move1[move1-A]
    serve --> grasp[grasp1-A]
    serve --> move2[move2-A]
    serve --> place[place1-A]
    move1 --> on3[on3-A]
    move1 --> mae3[maeC3]
    grasp --> on1[on1-A]
    grasp --> trixi[trixi]
    move2 --> on2[on2-A]
    move2 --> mas1[masC1]
    place --> on4[on4-A]
    place --> sawC1[sawC1]
    place --> pawC1[pawC1]
    on3 --> plate[plate1-A]
    on3 --> counter[counter1]
    on1 --> counter
    on1 --> mug[mug1-A]
    on2 --> at1[at1-A]
    on4 --> table[table1]
    on4 --> paw[paw1]
    on4 --> saw[saw1]
    counter --> bb1[bb1-A]
    counter --> white[white]
    at1 --> fight[fightOf1-A]
    at1 --> left[leftOf1-A]
    table --> fight
    table --> left
    paw --> left
    saw --> left
    fight --> mas1[mas1]
    left --> guest[guest1-A]
    left --> man1[man1]
    sawC1 -.-> sae1[sae1]
    sae1 -.-> saeC1[saeC1]
    pawC1 -.-> pae1[pae1]
    pae1 -.-> paeC1[paeC1]
    manC1[manC1] -.-> man1
    
```

Legend: bold edges = support before learning
 thin edges = relevant roles for new indirect support
 broken edges = relevant roles of permanent knowledge


18

Conceptualization of an Assertion Graph

Conceptualization of an example generates a conceptual description for exactly the same activity, to be carried out at any time.

Examples:

<pre>{grasp1-A: Grasp ^ (hasAgent Robot trixi) ^ (hasObject Mug mug1-A) ^ (hasStartTime Time 12:30:00) ^ (hasEndTime Time 12:30:46)}</pre>		<pre>Grasp1-A ⊆ Grasp □ hasAgent {trixi} □ hasObject Mug1-A □ hasStartTime Time □ hasEndTime Time</pre>
<pre>{mug1-A: Mug ^ (hasColor Color white)}</pre>		<pre>Mug1-A ⊆ Mug □ hasColor {white}</pre>

19

Conceptualization Rules

Object Descriptor	Old Concept	New Concept
{IN: CN'} leaf of assertion graph	CN	CN'' ⊆ {IN} CN'' ≠ CN → change
{IN: CN' ∧ ... ∧ (RN _k CN _k ' IN _k) ... }	CN ⊆ CN ₀ □ ... □ RN _k CN _k □ ...	CN'' ⊆ CN' □ ... □ RN _k CN _k ' ... CN' = CN ₀ , all CN _k ' unchanged → CN'' = CN else CN'' = new concept name
{IN: CN' ∧ ... ^ (hasStartTime Int [int ₁ , int ₂]) ^ (hasEndTime Int [int ₃ , int ₄]) ^ (hasDuration Int [int ₅ , int ₆])}	CN ⊆ CN ₀ □ ... □ hasStartTime Int □ hasEndTime Int □ hasDuration Int	CN'' ⊆ CN' □ ... □ hasStartTime Int □ hasEndTime Int □ hasDuration {(1-q)*int ₅ , (1+q)*int ₆ }

20

Alignment of two Concept Graphs

Determine graph homomorphism by finding alignment with least distance. Omit non-corresponding indirect support.

Distance between two nodes or two edges:
 = distance to LNCS (Least Named Common Subsumer)

Distance between two node complexes compiled from

- distance of center nodes
- distance of directly connected edges and satellite nodes

PlaceObject1-A

hasOn ↓

On1-A

hasPE ↙ ↘ hasArea

Mug1-A paWest1

PlaceObject1-B

hasOn ↓

On1-B

hasPE ↙ ↘ hasArea

Cup1-B paEast1

21

Merging two Concept Graphs

Graph A

PlaceObject1-A

hasOn ↓

On1-A

hasPE ↙ ↘ hasArea

Mug1-A paWest1

↓

{white}

Graph B

PlaceObject1-B

hasOn ↓

On1-B

hasPE ↙ ↘ hasArea

Mug1-B paEast1

↓

{yellow}

➔

Graph AB

PlaceObject1-AB

hasOn ↓

On1-B

hasPE ↙ ↘ hasArea


Mug1-B PA

↓

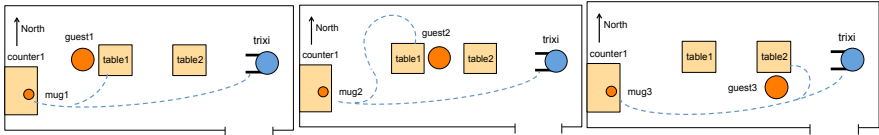
{white, yellow}


Computation of a "Good Common Subsumer" (Bader et al. 07)

22




Results of Learning ServeACoffee from Scenarios A, B, C






ServeACoffee-ABC:

" Move to the manipulation area at counter1, grasp a white or yellow mug, move to the manipulation area at the table right of the guest, place the mug at placement area assigned to the sitting area where the guest is located."


23





Using Negative Examples to Remove Irrelevant Support

Conceptual descriptions learnt from few examples may include many assertions as indirect support which are actually irrelevant.

Heuristic for pruning indirect support:
Remove all assertions which are also indirect support of negative examples.

- Picture at wall
- Chair at irrelevant table
- Robot is initially at door


24




Adapting a Concept to a Negative Example


Careful generalization from positive examples helps to avoid negative examples, but ...

- **taxonomy may not allow necessary differentiation**
e.g. what are "standard table items" which need not be cleared from a table?
- **concept may be too general because relevant support has not been determined adequately**
e.g. color of mugs may be important

Two adaptation procedures for negative examples:


- **Use affordance property to differentiate between positive and negative examples**
e.g. hasAffordance clearTable
- **Refine concept by re-learning with more relevant context**



25



Properties of EBCL

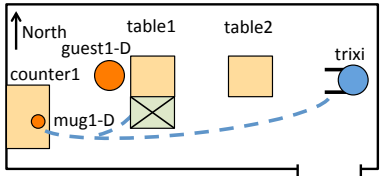
- **Learning compositional concept descriptions based on existing concepts can be used to create concepts at increasingly high abstraction levels.**
- **EBCL constructs the most specific descriptions covering all positive examples.**
- **Given an observant robot, the learnt concept descriptions will monotonously increase in generality and eventually reach the intended target description.**


26



Dealing with Obstacles

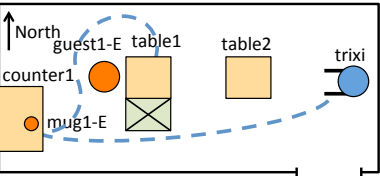
Scenario D



Person blocks manipulation area south. Robot is instructed to wait until path is unblocked.


➔ Robot conceptualizes *ServeACoffeeBlocked-D* including waiting.


Scenario E



Sidetable blocks manipulation area south. Robot generalizes *ServeACoffeeBlocked-D* and waits. Robot is instructed not to wait but to move to the manipulation area north.

➔ Robot conceptualizes *ServeACoffeeBlocked-E* including serving from the north.


27



Using Affordance Property (1)

Robot experiences several obstacles and is instructed how to behave:

{person, robot}	➔	wait!	(ServeACoffeeBlocked-D)
{sidetable, chair, bag}	➔	go to other side!	(ServeACoffeeBlocked-E)

Taxonomical grouping by means of affordance property:


```

Class: Person (dto. robot)
SubclassOf: PhysicalEntity
that hasAffordance ServeACoffeeBlocked-D ←
and ...


Class: sidetable (dto. chair, bag)
SubclassOf: PhysicalEntity
that hasAffordance ServeACoffeeBlocked-E ←
and ...

```

activity which can be performed with this kind of physical entity

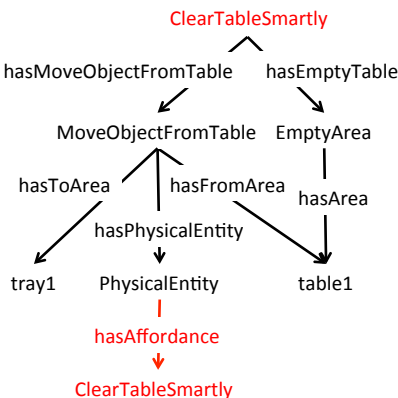

28


14




Using Affordance Property (2)

Robot learns to clear a table except for standard items (vase, pepper-and-salt, menu, etc.)





29




Learning to Avoid Toppling Over


Experiment planned for final review of RACE in January 2015:
Robot learns that jerky motion and sharp corners cause tall objects on a tray to topple over.



Noisy acceleration magnitude values recorded by robot
 [... 3 4 3 3 0 1 1 1 3 8 3 6 4 2 3 2 ...]
 toppling-over event recorded by robot


EBCL singles out high acceleration as characteristic feature of toppling events.


30



Conclusions

- **Symbolic learning of OWL concepts can include non-symbolic information**
- **EBCL can be modelled with graphical structures known from Cognitive Science (Gentner 86)**
- **Learning of new activity concepts can be achieved with few examples**
- **Limitations of OWL (and DLs in general) partly spoil the fun**



31



Trixi in TAMS Restaurant



32