Deliverable number: D3.3

Deliverable Title: Generalization by compilation procedures
Type (Internal, Restricted, Public): PU

Authors: Mohamad Javad Aein, Norbert Krüger, Leon Bodenhagen, Ales Ude, Minija Tamosiunaite, Florentin Wörgötter

Contributing Partners: UGOE, SDU, JSI

Project acronym: ACAT

Project Type: STREP

Project Title: Learning and Execution of Action Categories

Contract Number: 600578

Starting Date: 01-03-2013

Ending Date: 30-04-2016

Contractual Date of Delivery to the EC: 31-08-2015
Actual Date of Delivery to the EC: 04-09-2015

Content

1. Executive summary

This document presents the theoretical background for generalization by compilation procedures applied in the ACAT project. Essentially, here we are presenting material on which we base extension of the sub-symbolic compilation procedures presented in D3.2 to a bigger number of actions, as our reviewers have requested in the reviewers' report for the year two of the Project.

In the current deliverable we are providing only the theoretical background, while actual demonstration of the generalization by compilation will be provided at the end of the year three, together with the final demonstrators.

We explain our changes in object role nomenclature required for the expansion of the action set and introduce Action Data Table (ADT) extensions required for the generalization we are presenting here. In total 22 actions are analyzed and the possibilities for information re-use between those actions (i.e. generalization) are indicated.

2. Introduction

In the deliverable D3.2 and the WP3 report in the PPR of the year two of the project we have presented our general approach on compilation of new instructions, given the textual ontology and the library of previous robot execution examples presented in the form of ADTs. We have illustrated the approach in D3.2 and PPR by a small number of examples (mostly pick and place) of ADT re-use. The aim of this deliverable is to analyze the extent to which the methods presented previously can be used for a bigger set of actions for cross compilation and generalization between them.

Now our set of actions consists of the 22 actions extracted from the two ACAT scenarios, augmented with some general purpose actions which were not directly indicated in the instruction sheets provided in deliverable D5.1 but are many times used in table-top operations and might be frequently required in industrial assembly or chemical laboratory tasks. Note, the nomenclature of actions used in the two scenarios is not very diverse with pick&place, peg in hole (insert), screw/unscrew and button press operations prevailing.

In order to introduce new actions we had to adjust object role definitions, originally introduced in the ACAT term glossary provided in PPR 2, where new definitions are given in Chapter 3.

Now we have expanded the ADTs by including the movement primitives into the ADTs. The changes are introduced in Chapter 4.

In Chapter 5 we are providing explanations of how to perform action temporal segmentation and discuss the segment interpretation used in generalization procedures based on several actions. The entire set of actions with appropriate interpretations is presented in the Appendix of this document.

In Chapter 6 we are discussing cross-compilation and generalization issues based on action representations given in chapters 3-5.

Finally, in Chapter 7 we present conclusions and indications for future work. Note, on top of the cross-compilation and generalization theoretical analysis discussed in this deliverables we will make actual cross-compilation implementations for the final demo of the ACAT project.

3. Adjustments of object role definitions.

When analyzing an instruction we attribute roles to the objects indicated in the sentence (main object, primary object, secondary object, etc., see page 13 of the PPR year 2). Adjusted role definitions are given in Table 1.

Here we have to add new roles “load” and “container” for pouring and pipetting actions, which were not included into our demonstrators previously. We have slightly re-defined primary and secondary object, as to increase generalizability. For the concise definition of changes see the table.

Table 1. Changes on object role definitions

Object	ACAT term glossary	modified definition
Hand	The hand (always present)	The object that performs the action. The object which is free at the beginning and the end of action. (free means not touching any other object, except the tool)(always present)
Tool	The entity grasped by the hand to perform an action instead of the hand (optional)	Same
Main	The object which is first touched by hand/tool (always present)	Same
Primary	The object which is first touched/untouched by the main object. (Optional)	The first object which makes a T-->N transition with the main object and it is not hand/tool (Optional)
Secondary	The object which is second touched/untouched by the main object. (Optional)	The first object which makes a N-->T transition with the main object and it is not hand/tool (Optional)
Main support	The object on which the main object rests, where this “resting relation” will not change during the action.(Optional)	Same
Primary support	The object on which the primary object rests, where this “resting relation” will not change during the action.(Optional)	Same
Secondary support	The object on which the secondary object rests, where this “resting relation” will not change during the action.(Optional)	Same
Tool support	The object on which the tool object rests.(Optional,may need re-definition)	Same
Load	-	The object which is not directly touched by the hand, touches/untouches the main and untouches/touches another object
Container	-	The object which touches/untouches the load

4. Adjustments of ADTs

After experiments performed in year two of the project, we have found that in addition to the two hierarchical levels of action temporal segmentation previously included in the ADTs: *action primitives* and *action chunks* (see Fig. 1 in D3.2), it is practical to add another finer level of temporal segmentation which we call "movement primitives" (See Fig. 1A below). Figure 1B, by means of an example, shows the meaning of the three hierarchical levels: *action primitive*, *action chunk* and the *movement primitive*. Note that the action chunk level is defined by touching or un-touching of objects in the scene during the manipulation and thus is defined in an objective way. *Action primitives* and *movement primitives* rely on units of action conventionally used in robotic systems.

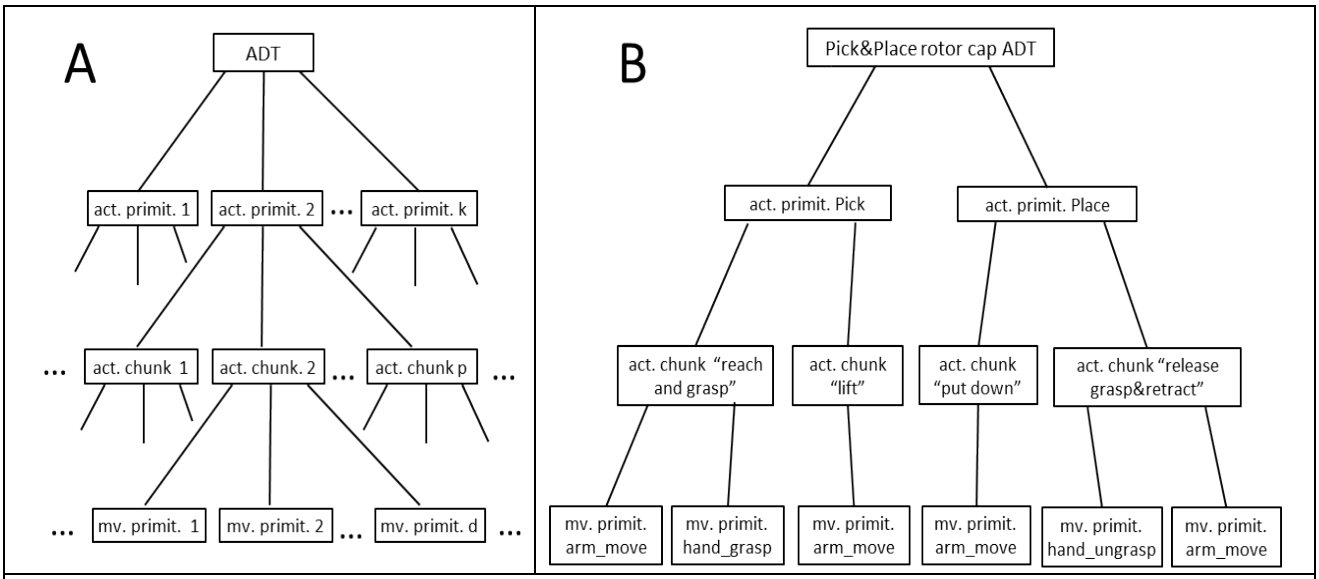

Fig. 1. Three hierarchical levels of ADT action representation and an example based on Pick&Place action.

The *movement primitives* as such were introduced already in D3.2 but now we have decided to save them in the ADTs together with the other action information. We consider *movement primitives* to be the smallest temporal segments of action with explicit symbolic interpretation.

We use the following *movement primitives*:

1) arm_move(goal_pose);
2) arm_rotate(quaternion);
3) arm_move_periodic(f);
4) arm_insert(peg_length);
5) arm_excert_force(goal_force);
6) hand_move(goal_angles);
7) hand_grasp;
8) hand_ungrasp.

These or similar movement primitives are used in many robotic systems containing an arm and a hand thus our movement-primitive-based representations has a certain degree of generality.

The primitives are now added to the ADT for each action chunk. Stating specificity of the representation: each movement primitive belongs to a specific action chunk and thus movement primitives are not allowed to cross action chunk boundaries. In each action chunk we now provide a list of movement primitives with start and end time as well as movement primitive-specific parameters. The first movement primitive in an action chunk has a start time corresponding to the start time of the action chunk, while last movement primitive has an end time corresponding to the end time of the action chunk. Otherwise movement primitives can overlap or be kept sequential. E.g. the "arm_move" and "hand_grasp" in the action chunk "reach and grasp" can be executed in a sequence in a "simple" robotic system, however can be executed in an overlapping manner in a system with advanced grasp implementation.

In addition, movement primitives are parameterized with a small number of parameters. E.g. For the *arm_move* movement primitive we indicate the goal pose. For the *arm_rotate* movement primitive we indicate the quaternion to indicate rotation angles. For the *arm_move_periodic* we indicate the frequency *f* of the periodic movement. For the *arm_excert_force* movement primitive we indicate the goal force. However the *hand_move* primitive is currently parameterized with the hand_angles which is not task space description as hand movements are very difficult to describe in a way that generalizes across platforms.

And excerpt from an ADT with movement primitive description is provided in Fig. 2.

```
<movement_primitives>

<movement_primitive>
<mp_name>arm_move</mp_name>
<mp_startime>chunk_start</mp_startime>
<mp_endtime>1408093815.420</mp_endtime>
<mp_parameters>
<pose>
        <position>0.620 0.593 0.287</position>
        <quaternion>0.935286 -0.350021 0.049885 -0.015384</quaternion>
        <pose_reliability>1.0</pose_reliability>
</pose>
</mp_parameters>
</movement_primitive>

<movement_primitive>
<mp_name>hand_grasp</mp_name>
<mp_startime>1408093815.420</mp_startime>
<mp_endtime>chunk_end</mp_endtime>
<mp_parameters>void</mp_parameters>
</movement_primitive>

</movement_primitives>
```

Fig. 2. Excerpt from an action chunk "reach and grasp" (first action chunk for a Pick&Place ADT-action) with two movement primitives indicated. Note, the start of the first movement primitive is kept together with the start of the action chunk and the end of the second movement primitive is kept together with the end of the action chunk.

5. Action interpretation for generalization

In this chapter we will explain our approach to action segmentation and interpretation which we will further use for generalization procedures. Here we mainly concentrate on temporal action segmentation (action chunks and movement primitives discussed in Chapter 4 above), as those segments are the units which then are re-used in generalization procedures.

Interpretation concerns relational readout of the existing ADT information. As ADTs are given in absolute coordinates, here we are indicating which relational information is important, i.e. which relations between the objects and TCP need to be kept in ADT re-use.

In Fig. 3 below we show a *pick&place* action, based on the nomenclature of object roles given in Table 1 above.

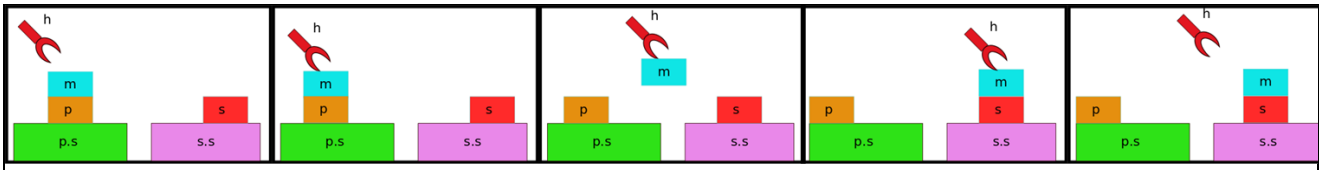

Fig. 3. Phases of touching and un-touching of different objects in a *pick&place* action. Notations: h- hand, m – main object, p –primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Figure 3 allows us to define a Semantic Event Chain (SEC) for the *pick&place* action. In the consecutive panes of the figure it is indicated how touching and un-touching relations change throughout the action. The corresponding SEC is provided in Table 2 below.

Table 2. Semantic Event Chain for a *pick&place* action shown in Fig. 3. "T" means relation between the objects "touching", "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2, P3 are given required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, p.s	N	N	N	N	N
main, s.s	N	N	N	N	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	hand_ungrasp	
P2	arm_move(main)			arm_move(sec.)	
P3	hand_grasp			arm_move(free)	

In Table 2 it is indicated how the touching and non-touching relations change during the action (i.e. the SEC of an action is shown). The action segment that leads from one SEC column to the other we call an action chunk, as indicated in D3.2. In the last rows of the Table 2 it is indicated which movement primitives are required to obtain the next SEC state (i.e. which movement primitives

correspond to an action chunk). One or more movement primitives can correspond to one action chunk.

The arguments given together with the movement primitives in the table (e.g. main, prim. sec.) show the required interpretation in the existing ADTs. Specifically, they show in relation to which object, indicated in the ADT, the movement shall be interpreted. This is additional information which we use for generalization, but which is not directly indicated in the ADT. Note, an ADT is considered to be "raw" information potentially extractable without manual intervention and interpretations for re-use of the information come on top.

Continuing with the explanation of the arguments of the movement primitives provided in Table 2:

- *arm_move(main)* associated to SEC column 1 (action chunk No 1) means that the arm has to move to the relative pose in respect to main object as given at the end of the first action chunk in the existing ADT;
- *arm_move(prim.)* associated to SEC column 2 (action chunk No 2) means that the arm has to execute a movement with the end pose relative to the primary object as given in the existing ADT;
- *arm_move(sec.)* associated to SEC column 3 (action chunk No 3) means that the arm has to execute a movement with the end poses relative to the secondary object as given in the existing ADT; an existing ADT will show an approach to the secondary object in this case ;
- *arm_move(sec.)* associated to SEC column 4 (action chunk No 4) means that the arm has to execute a movement with the end poses relative to the secondary object as given in the existing ADT; however differently from the same primitive in SEC column 3, here a retract of the arm in respect to the secondary object would be indicated in the ADT;
- *arm_move(free)* associated to SEC column 4 (action chunk No 4) means that the arm has to execute a movement with the free end pose (no requirements in respect to objects included in the ADT)

The movement primitive *hand_move(pregrasp)* notes preparation of the correct pose of a hand for grasping. Different actions would need different poses of the hand, like e.g. for pushing or punching or pulling one would need different poses of the hand. Movement primitives *hand_grasp* and *hand_ungrasp* in the current version do not have parameters and are not associated to any objects, however potentially could be extended with parameters (e.g. grasping force and this would then depend on the main object).

Here we will show one example of how we define re-use of (generalization on) action segments. For this here we show another example of an action (*shake*) including the temporal segmentation as for the previous *pick&place* action. The Fig. 3 as well as the semantic event chain (top of Table 2) for the action *shake* is the same as for the action *pick&lace*. However, there are differences in movement primitives for some action chunks (see Table 3 bottom).

Comparing Table 2 and Table 3 one can see that the semantic event chain transitions are the same. For the action chunks No 1, 2 and 4 also the movement primitive sequences are the same. This indicates the similarity of the two actions. If the main object is the same (or similar, according to textual ontology or even according to the shape as given by CAD models), one can re-use the movement primitives from *pick&place* (actions cunk No 1) to perform the action *shake* or vice versa. If primary object is the same or similar, one can re-use movement primitives from action chunk No 2. If, in addition, the secondary object is the same or similar movement primitives from the action chunk No 4 can be re-used. The movement primitive from action chunk No 3

arm_move(sec.) can also be re-used. The movement primitive arm_move_periodic(main) in the action chunk No 3 action *shake* means that the frequency and amplitude depend on the main object. As this movement primitive does no exist in the *pick&place* action it cannot be re-used from *pick&place*. However if the action *shake* has been recorded on the ADT and the main objects (i.e. objects which one has to shake) match, then we can re-use amplitude and frequency from the movement primitive arm_move_periodic in the previous shaking example.

Table 3. Semantic Event Chain for an action *shake*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2 are given required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, p.s	N	N	N	N	N
main, s.s	N	N	N	N	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move_periodic (main)	hand_ungrasp	
P2	arm_move(main)		arm_move(sec.)	arm_move(sec.)	
P3	hand_grasp			arm_move(free)	

Definition of all 22 actions we are considering here (figures as Fig. 3 and tables as Table 2) are provided in the Appendix. Chapter 6 shows possibilities for re-use between the actions provided in the appendix.

6. Movement primitive-based generalization

In Chapter 5 we have shown how we parameterize actions and have shown one example how we are doing movement primitive re-use (generalization) in different actions. Here we will go through all 22 actions discussed in this deliverable (see the Appendix) and define the possibilities for information re-use based on coinciding action chunks and movement primitives provided in the ADTs. Here we note one more time that re-use is only eligible if appropriate objects (main, primary, secondary) coincide or are similar (based on textual ontology or shape similarity). Also one shall consider, that here as in any type of generalization mistakes may happen. We will consider this issue in more detail at the end of this chapter.

Further we will provide an action list which we are considering in this deliverable as well as in the final demos on generalization. Action description with appropriate Semantic Event Chains and movement primitives are given in the Appendix. Some actions are bound to the ACAT scenarios, but also many actions are of general importance, e.g. *pick&place* or *push* are very frequently used

in any human or robot table-top operation. Another example is *screwing* and *unscrewing*. You will need screwing and unscrewing caps on bottles in the chemical laboratory and screwing and unscrewing screws in industrial assembly and in other scenarios, too.

Note that most of the actions given below are linguistically notated by different verbs. However, we have included two action groups (corresponding to *pick&place* and *push* verbs) where linguistically actions are notated by the same verb, however contextual details and robotic execution differ. E.g. we can pick an object (in an action *pick&place*) and place it on top of some other object or we can place it to the side of some other object. In these two cases not only "place" action segment may differ but also grasps and approach directions may differ, too. Linguistic contexts of the verb will also differ (e.g. "Place a bottle on the plate" vs. "Place a bottle next to the plate").

Next we provide an action list:

1. Poke/Press button
2. Push
3. Pull
4. Pick&Place from side to side
5. Pick&Place from top to top
6. Pick&Place from top to side
7. Drop
8. Screw (a cap or a screw)
9. Unscrew (a cap or a screw)
10. Insert (peg in hole)
11. Push apart
12. Push together
13. Push from X to Y
14. Cut
15. Chop
16. Stir
17. Shake
18. Pipette
19. Pour
20. Align (by grasping)
21. Rotate (by grasping an object which has a rotation axis)
22. Punch

Possibilities for action component re-use between actions are presented in Table 4. Possibility of re-use of *some* components is marked with a plus signs and possibility of re-use of many components is marked with a letter "h" (high similarity). One shall use this table as a look-up table in generalization algorithms.

The table is mainly symmetric with a small amount of exceptions. Exceptions arise due to some actions requiring more specific components than the others. E.g. grasp of a spoon for stirring can be transferred to grasping of a spoon for dropping it in a bowl with dirty tools. However opposite transfer is not possible, as one cannot be sure that for dropping of a spoon someone used the same grasp which is appropriate for stirring.

There are several very specific actions, like e.g. pipetting or pouring which are not similar to other actions and cross-compilation is not possible. However, there are groups of actions e.g. one consisting of pick&place, drop, shake, insert, screw/unscrew, another consisting or poking, varieties of pushing, punching, where re-use of movement primitives can be performed between different actions.

Table 4. Possibilities for action information re-use. Minus sign means no re-use possibilities. Plus sign means some components can be re-used. "h" means high potential for re-use (many components can be re-used).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1		**h**	-	-	-	-	-	-	-	-	**+**	**+**	**+**	-	-	-	-	-	-	-	-	**+**
2	**h**		-	-	-	-	-	-	-	-	**+**	**+**	**+**	-	-	-	-	-	-	-	-	-
3	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-		**+**	**h**	**+**	-	-	**+**	-	-	**+**	-	-	-	**h**	-	-	**+**	**+**	-
5	-	-	-	**+**		**h**	**h**	**+**	**+**	**+**	-	-	-	-	-	-	**h**	-	-	**h**	**+**	-
6	-	-	-	**h**	**h**		**+**	**+**	**+**	**+**	-	-	-	-	-	-	**+**	-	-	**h**	**+**	-
7	-	-	-	**+**	**h**	**+**		**+**	**+**	**+**	-	-	-	-	-	-	**+**	-	-	-	-	-
8	-	-	-	-	**+**	**+**	**+**		**h**	**+**	-	-	-	-	-	-	**+**	-	-	**+**	**h**	-
9	-	-	-	-	**+**	**+**	**+**	**h**		**+**	-	-	-	-	-	-	**+**	-	-	**+**	**h**	-
10	-	-	-	**+**	**+**	**+**	**+**	**+**	**+**		-	-	-	-	-	-	**+**	-	-	**+**	**+**	
11	**+**	**+**	-	-	-	-	-	-	-	-		**+**	**h**	-	-	-	-	-	-	-	-	**+**
12	**+**	**+**	-	-	-	-	-	-	-	-	**+**		**h**	-	-	-	-	-	-	-	-	**+**
13	**+**	**+**	-	**+**	-	-	-	-	-	-	**h**	**h**		-	-	-	-	-	-	-	-	**+**
14	-	-	-	-	**+**	**+**	**+**	-	-	-	-	-	-		**+**	-	-	-	-	-	-	-
15	-	-	-	-	**+**	**+**	**+**	-	-	-	-	-	-	**+**		-	-	-	-	-	-	-
16	-	-	-	-	**+**	**+**	**+**	-	-	-	-	-	-	-	-		-	-	-	-	-	-
17	-	-	-	**h**	**h**	**+**	**+**	**+**	**+**	**+**	-	-	-	-	-	-		-	-	-	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-
20	-	-	-	**+**	**h**	**h**	**+**	**+**	**+**	**+**	-	-	-	-	-	-	-	-	-		**+**	-
21	-	-	-	**+**	**+**	**+**	**+**	**h**	**h**	**+**	-	-	-	-	-	-	-	-	-	**+**		-
22	**+**	**+**	-	-	-	-	-	-	-	-	**+**	**+**	-	-	-	-	-	-	-	-	-	

Here we formulate some general rules, based on movement primitives, which allow re-use of the movement primitives between different actions.

1. If the main object is the same in the new instruction and the ADT (or the shapes match), all the corresponding movement primitives for the first action chunk can be re-used if the table above allows re-use.
2. If both main and primary objects are the same in the new instruction and the ADT, all the corresponding movement primitives from the first two chunks can be re-used if the table allows re-use.
3. If the secondary object is the same in the new instruction and the ADT, all the movement primitives performed in respect to the secondary object can be re-used if the table allows re-use (many times it would be the last two action chunks).

These are essentially the same rules as given in PPR 2 for instruction compilation at the sub-symbolic level. However, now we formulate those rules based on movement primitives (earlier movement primitive-based formulation was not given). In addition we are indicating in Table 4 between which actions we allow to transfer information.

It is obvious that straightforward re-use, as formulated in those rules will sometimes lead to mistakes. Here we do not consider any considerable clutter of the scene, where approach and retract directions would depend on other objects in the scene which are not mentioned in the instruction as such. In such cases one would need movement planning, possibly using ADT information for biasing the solution. Also, re-usability heavily depends on the precision required. If we are manipulating bigger objects, like cups and plates on a table or e.g. rotor caps in IASSES scenario, quite a large tolerance for precision can be expected. However if one deals with magnets which have to be inserted into a rotor in the IASSES scenario, much more precision is needed and e.g. an ADT-derived approach direction if taken from an action of insertion of a magnet of a different size or different shape might not be re-usable. Thus a user needs to decide how far to trust the ADTs, and for more accurate operations one needs human in the loop for generalization based on ADTs. However in less precise service robotics applications, like e.g. cooking scenarios frequently used as show-cases in robotics the re-use can go much further and with much less human supervision.

7. Conclusions and future work

In this deliverable we have provided a theoretical framework for action generalization through compilation procedures in the ACAT project. For 22 actions we provide movement primitive sequences which can be directly used in action execution. The movement primitive parameterization is explained in Chapter 4 and in Chapter 5 we explain how we interpret existing ADT material (at movement primitive level) for re-use. In the Appendix we provide movement primitive interpretation for each action, i.e., we are indicating in relation to which objects one has to interpret poses and trajectories when re-using ADT information in the new context.

We have introduced a table which shows when re-use information from one action into another is allowed and when it is not allowed. In addition we have provided rules based on action chunks and movement primitives for re-use of information in the cases when the re-use is allowed.

The rules for information re-use between actions will be used for generalization in execution of new instructions in robotic setup towards the end of the project and demonstrated at the third ACAT project review meeting.

Appendix. Movement primitives for 22 actions

In this appendix we provide tables indicating action chunk sequences and movement primitive sequences for the actions analyzed in this deliverable. The actions are illustrated by figures showing different action phases based on touching and non-touching relations. The tables in the upper rows show Semantic Even Chains (i.e. how touching and non-touching relations change throughout the action, SEC) and in the lower rows they show movement primitives which are needed to induce SEC transitions. The arguments in the brackets at movement primitives show in respect to which object to interpret the existing ADT material when defining an appropriate movement primitive in the new situation. For explanation how to interpret those tables see Chapter 5.

Actions: poke, push, pull, punch, rotate

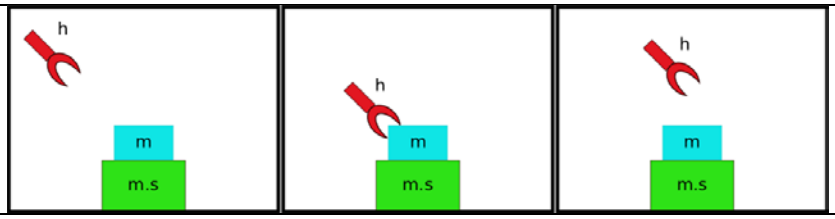

Fig. A1. Phases of touching and un-touching of different objects in actions: poke, push, pull, punch, rotate. Notations: h- hand, m – main object, m.s. – main support.

Table A1. Semantic Event Chain and movement primitives for actions poke, push, pull, punch, rotate. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2 are given for each action required to make a transition to the next SEC state.

SEC columns	1	2	3
hand, main	N	T	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	
Poke, push			
P1	hand_move(straighten)	arm_move(main)	
P2	arm_move(main)	arm_move(free)	
Pull			
P1	hand_move(hook)	arm_move(main)	
P2	arm_move(main)	arm_move(free)	
Punch			
P1	hand_move(fist)	arm_move(main)	
P2	arm_move(main)	arm_move(free)	

Rotate			
P1	hand_move(pregrasp)	arm_rotate	
P2	arm_move(main)	hand_ungrasp	
P3	hand_grasp	arm_move(main)	
P4		arm_move(free)	

Action: pick&place from side to side

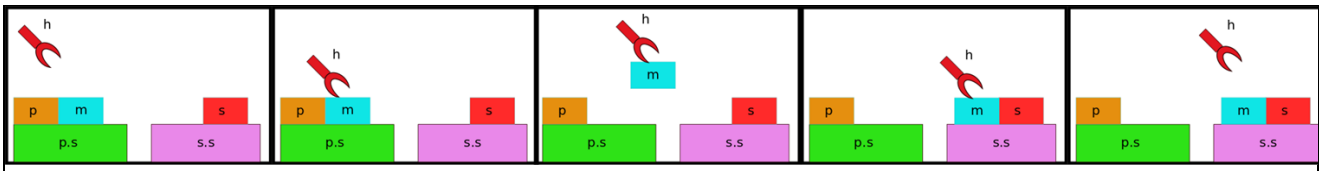

Fig. A2. Phases of touching and un-touching of different objects in action *pick&place from side to side*. Notations: h- hand, m – main object, p – primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Table A2. Semantic Event Chain and movement primitives for action *pick&place from side to side*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2, P3 are given for each action chunk required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, p.s	T	T	N	N	N
main, s.s	N	N	N	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	hand_ungrasp	
P2	arm_move(main)			arm_move(sec.)	
P3	hand_grasp			arm_move(free)	

Action: pick&place from top to top, screw, unscrew

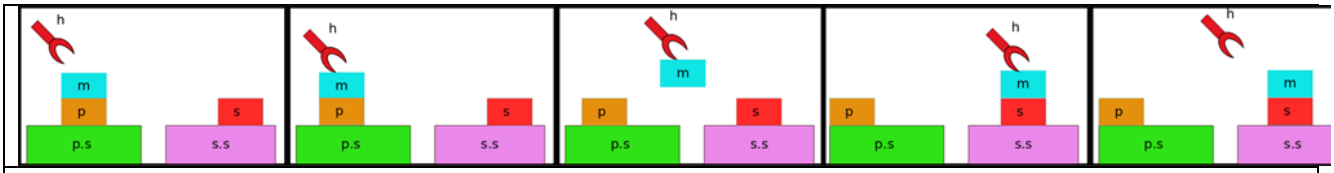

Fig. A3. Phases of touching and un-touching of different objects in action *pick&place from top to top*, *screw*, *unscrew*, *insert*. Notations: h- hand, m – main object, p – primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Table A3. Semantic Event Chain and movement primitives for action *pick&place from top to top*, *screw, unscrew, insert*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2, P3 are given for each action chunk.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, p.s	N	N	N	N	N
main, s.s	N	N	N	N	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
pick&place2					
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	hand_ungrasp	
P2	arm_move(main)			arm_move(sec.)	
P3	hand_grasp			arm_move(free)	
Screw					
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	hand_move	
P2	arm_move(main)			hand_ungrasp	
P3	hand_grasp			arm_move(sec)	
P4				arm_move(free)	
Unscrew					
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	hand_ungrasp	
P2	arm_move(main)			arm_move(sec.)	
P3	hand_grasp			arm_move(free)	
P4	hand_rotate				
Insert					
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	arm_insert	
P2	arm_move(main)			hand_ungrasp	
P3	hand_grasp			arm_move(sec.)	
P4				arm_move(free)	

Action: pick&place from top to side

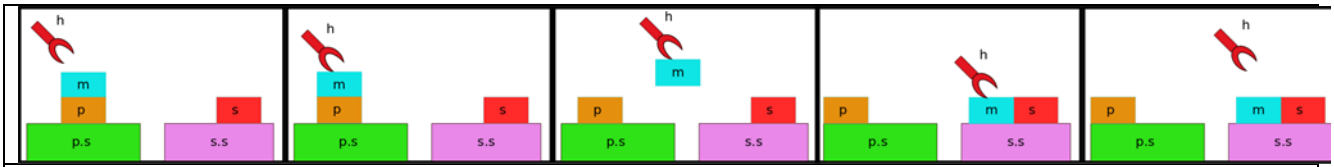

Fig. A4. Phases of touching and un-touching of different objects in action *pick&place from top to side*. Notations: h- hand, m – main object, p – primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Table A4. Semantic Event Chain and movement primitives for action *pick&place from top to side*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2, P3 are given for each action chunk required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, p.s	N	N	N	N	N
main, s.s	N	N	N	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	hand_ungrasp	
P2	arm_move(main)			arm_move(sec.)	
P3	hand_grasp			arm_move(free)	

Action: drop

Table A5. Semantic Event Chain and movement primitives for action *drop*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2, P3 are given for each action chunk required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5
hand, main	N	T	T	N	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	N	T
main, s.s	N	N	N	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move(sec.)	arm_move(free)	
P2	arm_move(main)		hand_ungrasp		
P3	hand_grasp				

Action: push apart

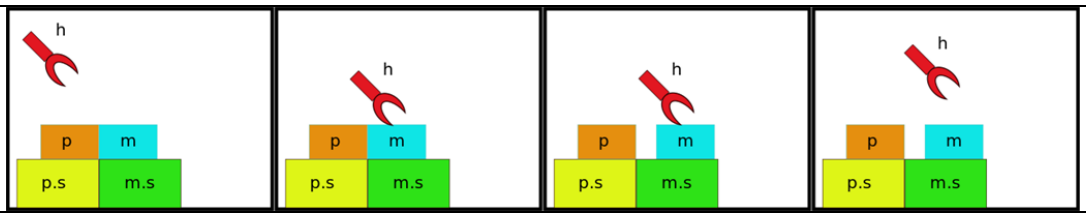

Fig. A5. Phases of touching and un-touching of different objects in action *push apart*. Notations: h- hand, m – main object, p – primary object, , m.s. – main support, p.s. – primary support.

Table A6. Semantic Event Chain and movement primitives for action *push apart*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives are given for each action chunk required to make a transition to the next SEC state. There is only one movement primitive per action chunk in this action.

SEC columns	1	2	3	4
hand, main	N	T	T	N
main, primary	T	T	N	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	
P1	arm_move(main)	arm_move(prim.)	arm_move(main)	
P2			arm_move(free)	

Action: push together

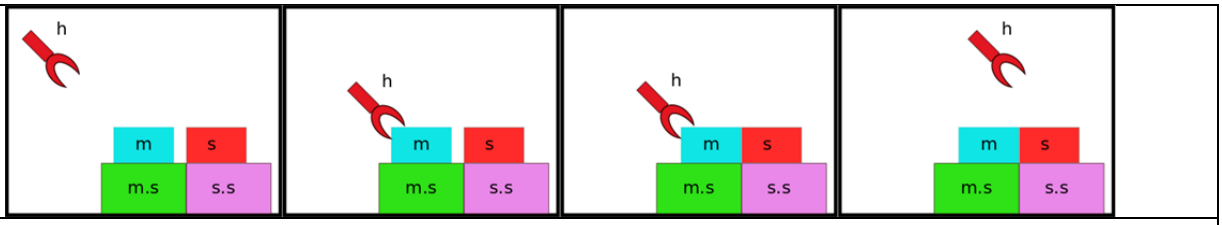

Fig. A6. Phases of touching and un-touching of different objects in action *push together*. Notations: h- hand, m – main object, s – secondary object, , m.s. – main support, s.s. – secondary support.

Table A7. Semantic Event Chain and movement primitives for action *push together*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives are given for each action chunk required to make a transition to the next SEC state. There is only one movement primitive per action chunk in this action.

SEC columns	1	2	3	4
hand, main	N	T	T	N
main, secondary	N	N	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	
P1	arm_move(main)	arm_move(sec.)	arm_move(sec.)	
P2			arm_move(free)	

Action: push from primary to secondary

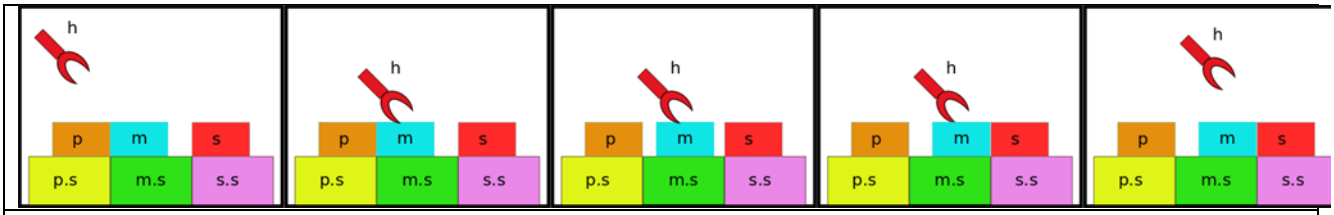

Fig. A7. Phases of touching and un-touching of different objects in action *push from primary to secondary*. Notations: h- hand, m – main object, p – primary object, s – secondary object, , m.s. – main support, p.s. – primary support, s.s. – secondary support.

Table A8. Semantic Event Chain and movement primitives for action *push from primary to secondary*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives are given for each action chunk required to make a transition to the next SEC state. There is only one movement primitive per action chunk in this action.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	arm_move(main)	arm_move(prim.)	arm_move(sec.)	arm_move(sec.)	
P2				arm_move(free)	

Actions: cut and chop

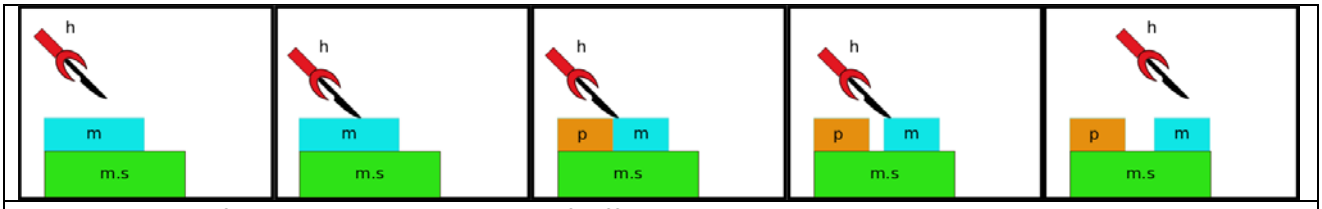

Fig. A8. Phases of touching and un-touching of different objects in action *cut and chop*. Notations: h- hand, m – main object, m.s. – main support. Tool is shown as the shape of the knife. Phases of grasping and releasing tool are not shown here.

Table A9. Semantic Event Chain and movement primitives for actions *cut* and *chop*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives are given for each action chunk required to make a transition to the next SEC state; m.s. means "main support", t.s. means "tool support". Grasping and releasing of the tool is not shown.

SEC columns	1	2	3	4	5
tool, main	N	T	T	T	N
main, primary	A	A	T	N	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
Cut					
P1	arm_move(main)	arm_move_periodic (main)	arm_move (primary)	arm_move(t.s.)	
P2		arm_excert_force			
Chop					
P1	arm_move(main)	arm_move(m.s.)	arm_move(primary)	arm_move(t.s.)	

Action: stir

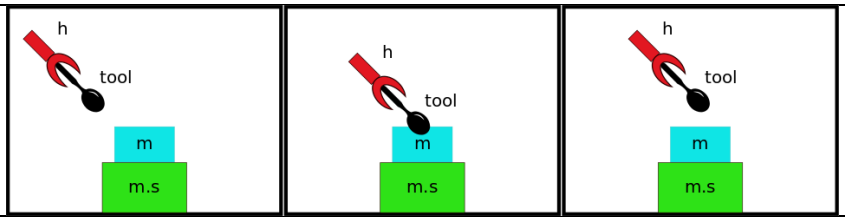

Fig. A9. Phases of touching and un-touching of different objects in action *stir*. Notations: h- hand, m – main object, m.s. – main support. Phases of grasping and releasing tool are not shown here.

Table A10. Semantic Event Chain and movement primitives for action *stir*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives are given for each action chunk required to make a transition to the next SEC state; t.s. means tool support.

SEC columns	1	2	3	4	5	6	7
hand, tool	N	T	T	T	T	T	N
tool, main	N	N	N	T	N	N	N
tool, t.s.	T	T	N	N	N	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	for transition 5->6 (action chunk No 5)	for transition 6->7 (action chunk No 6)	
P1	hand_move (pregrasp)	arm_move (tool,main)	arm_move (main)	arm_move_ periodic (main)	arm_move (t.s.)	arm_move (free)	
P2	arm_move (tool)			arm_move (main)	ungrasp		
P3	hand_grasp						

Action: shake

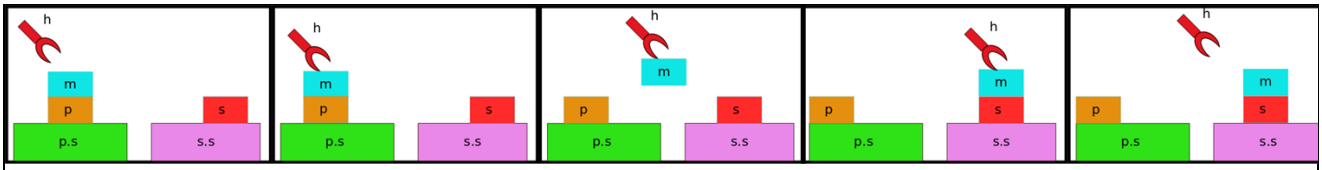

Fig. A10. Phases of touching and un-touching of different objects in action shake. Notations: h- hand, m – main object, p – primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Table A11. Semantic Event Chain for an action Shake. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2 are given required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, s.s	N	N	N	N	N
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	for transition 3->4 (action chunk No 3)	for transition 4->5 (action chunk No 4)	
P1	hand_move(pregrasp)	arm_move(prim.)	arm_move_periodic (main)	hand_ungrasp	
P2	arm_move(main)		arm_move(sec.)	arm_move(sec.)	
P3	hand_grasp			arm_move(free)	

Action: pipette

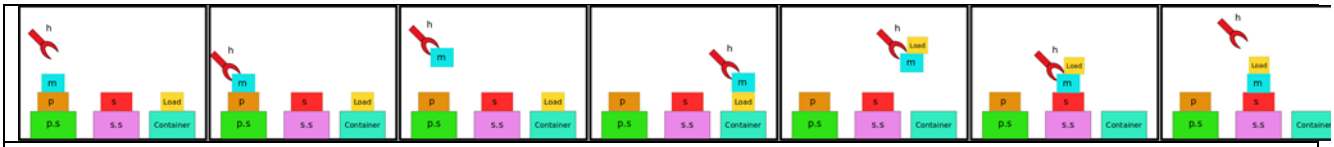

Fig. A11. Phases of touching and un-touching of different objects in action *pipette*. Notations: h- hand, m – main object, p – primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Table A12. Semantic Event Chain for an action *pipette*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2 are given required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5	6	7
hand, main	N	T	T	T	T	T	N
main, primary	T	T	N	N	N	N	N
main, secondary	N	N	N	T	T	T	T
main, load	N	N	N	T	T	T	T
load, container	T	T	T	T	N	N	N
movement primitives	transition 1->2 (act. ch. No 1)	transition 2->3 (act. ch. No 2)	transition 3->4 (act. ch. No 3)	transition 4->5 (act. ch. No 4)	transition 5->6	transition 6->7	
P1	hand_move (pregrasp)	arm_move (prim.)	arm_move (load)	arm_move (container)	arm_move (secondary)	arm_move(sec)	
P2	arm_move (main)		press_pipette		release_pipette	arm_move(free)	
P3	hand_grasp						

Action: pour

Fig. A12. Phases of touching and un-touching of different objects in action pour. Notations: h- hand, m – main object, p – primary object, s –secondary object, p.s. – primary support, s.s. – secondary support.

Table A13. Semantic Event Chain for an action *pour*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives P1, P2 are given required to make a transition to the next SEC state.

SEC columns	1	2	3	4	5	6	7
hand, main	N	T	T	T	T	T	N
main, primary	T	T	N	N	N	N	N
main, secondary	N	N	N	N	N	T	T
main, load	T	T	T	T	N	N	N
load, container	N	N	N	T	T	T	T
movement primitives	transition 1->2 (act. ch. No 1)	transition 2->3 (act. ch. No 2)	transition 3->4 (act. ch. No 3)	transition 4->5 (act. ch. No 4)	transition 5->6	transition 6->7	
P1	hand_move (pregrasp)	arm_move (prim.)	arm_move (container)	arm_move (container)	arm_move (secondary)	hand_ungrasp	
P2	arm_move (main)					arm_move (container)	
P3	hand_grasp					arm_move(free)	

Action: align by grasp

Table A14. Semantic Event Chain and movement primitives for action *align by grasp*. "T" means relation between the objects touching, "N" means relation between objects non-touching. On the bottom of the table movement primitives are given for each action chunk required to make a transition to the next SEC state. There is only one movement primitive per action chunk in this action.

SEC columns	1	2	3
hand, main	N	T	N
main, primary	T	T	T
movement primitives	for transition 1->2 (action chunk No 1)	for transition 2->3 (action chunk No 2)	
P1	hand_move(pregrasp)	arm_rotate(primary)	
P2	arm_move(main)	hand_ungrasp	
P3	hand_grasp	arm_move(free)	