

```

1   T<- (desig-action-grounding ?action-designator (close-container ?arm
2   7gripper-opening
3   ?distance
4   ?left-reach-poses
5   ?left-grasp-poses
6   ?right-grasp-poses
7   ?left-lift-poses)
8   ?right-lift-poses
9   ?left-2nd-lift-poses)
10  ?right-2nd-lift-poses)
11  ?cont-name
12  ?environment-ob))
13
14  (spec-property ?action-designator (type ?closing))
15  (spec-property ?action-designator (object ?closed-object-designator))
16  (spec-property ?container-designator (type ?container-type))
17  (obj-int-object-type-subtype ?container ?container-type)
18  (spec-property ?container-designator (script ?script-environment))
19  (> (spec-property ?action-designator (arm ?arm))
20  (true)
21  (and (cran-robot-interfaces:robot ?robot)
22  (cran-robot-interfaces:arm ?robot ?arm)))
23
24  (spec-property ?action-designator (?distance ?distance))
25  (lisp-fun man-int get-connecting-point ?container-arm ?connecting-joint)
26  (lisp-fun cl-robot:cl-arm ?name)
27  (lisp-fun btr:btr-world ?world)
28  (lisp-fun btr:btr-object ?world ?btr-environment ?environment-ob)
29
30
31
32  (lisp-fun obj-int get-object-type-gripper-opening ?container-type ?gripper-opening)
33  (lisp-fun get-container-pose-and-transform ?container-name ?btr-environment
34  ?container-pose ?container-transform)
35
36
37
38  (lisp-fun obj-int get-object-grasping-poses ?container-name
39  :container-prismatic ?left :close ?container-transform ?left-poses)
40  (lisp-fun obj-int get-object-grasping-poses ?container-name
41  :container-prismatic ?right :close ?container-transform ?right-poses)
42  (lisp-fun cran-mobile-pick-place:plane extract-pick-up-manipulation-poses
43
44
45
46
47  ?arm ?left-poses ?right-poses
48  (?left-reach-poses ?right-reach-poses
49  ?left-grasp-poses ?right-grasp-poses
50  ?left-lift-poses ?right-lift-poses))
51
52  (> (lisp-pred identity ?left-poses)
53  (equal ?left-lift-poses ?left-lift-poses ?left-2nd-lift-poses))
54  (equal (NIL NIL) ?left-lift-poses ?left-2nd-lift-poses))
55
56
57
58
59
60
61
62
63
64  (> (lisp-pred identity ?right-lift-poses)
65  (equal ?right-lift-poses ?right-lift-poses ?right-2nd-lift-poses))
66  (equal (NIL NIL) ?right-lift-poses ?right-2nd-lift-poses))
67
68

```



```

1   T<- (desig-action-grounding ?action-designator (close-container ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15  (spec-property ?action-designator (type ?closing))
16  (spec-property ?action-designator (object ?closed-object-designator))
17  (spec-current-designator ?object-designator ?current-object-design)
18
19  (spec-property ?current-object-design (type ?object-type))
20  (spec-property ?current-object-design (name ?object-name))
21  (> (spec-property ?action-designator (arm ?arm))
22  (true)
23  (and (lisp-fun man-int get-object-obt-transform ?current-object-design ?object-transform)
24  (lisp-fun man-int calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face))
25  (> (spec-property ?action-designator (grasp ?grasp))
26  (true)
27  (and (lisp-fun man-int get-action-grasps ?object-type ?arm ?object-transform ?grasps)
28  (member ?grasp ?grasps)))
29  (> (spec-property ?action-designator (object-half-pose ?object-half-pose))
30  (true)
31  (comment "Please infer where to close the container, or less the query system to infer it here"))
32  (lisp-fun man-int get-action-grasping-effort ?object-type ?effort)
33  (lisp-fun man-int get-action-gripper-opening ?object-type ?gripper-opening)
34  (equal ?objects ?current-object-design)
35  (> (equal ?arm ?left)
36  (and (lisp-fun man-int get-action-trajectory :closing ?arm ?grasp T ?objects
37  ?right-poses)
38  (lisp-fun man-int get-fra-poses-by-label ?left-closing-poses :closing-up
39  ?left-closing-up-poses)
40  (lisp-fun man-int get-fra-poses-by-label ?left-closing-poses :closing-down
41  ?left-closing-down-poses))
42  (and (equal ?left-closing-up-poses NIL)
43  (equal ?left-closing-down-poses NIL)))
44  (> (equal ?arm ?right)
45  (and (lisp-fun man-int get-action-trajectory :closing ?arm ?grasp T ?objects
46  ?right-poses)
47  (lisp-fun man-int get-fra-poses-by-label ?right-closing-poses :closing-up
48  ?right-closing-up-poses)
49  (lisp-fun man-int get-fra-poses-by-label ?right-closing-poses :closing-down
50  ?right-closing-down-poses))
51  (and (equal ?right-closing-up-poses NIL)
52  (equal ?right-closing-down-poses NIL)))
53  (> (design-prop ?action-designator :collision-mode ?collision-mode))
54  (true)
55  (equal ?collision-mode NIL)
56  (design-designator :action (type ?closing)
57  (object ?current-object-design)
58  (object-name ?object-name)
59  (arm ?arm)
60  (grasp ?grasp)
61  (effort ?effort)
62  (script ?script)
63  (left-closing-up-poses ?left-closing-up-poses)
64  (right-closing-up-poses ?right-closing-up-poses)
65  (left-closing-down-poses ?left-closing-down-poses)
66  (right-closing-down-poses ?right-closing-down-poses)
67  (collision-mode ?collision-mode))
68  ?resolved-action-designator)

```

Figure 1: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Halving* (55 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
40	32	12	3

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *object* instead of *container*)
- adds variables and their handling from the example designator, despite being unnecessary for the use case (here: *?object-half-pose*)
- generated designator ends by mapping variables to designator "parts"

```

1  | <- (desig-action-grounding ?action-designator (?close-container ?arm
2  |   | gripper-opening
3  |   | ?distance
4  |   | ?left-reach-poses
5  |   | ?right-reach-poses
6  |   | ?left-poses
7  |   | ?right-grasp-poses
8  |   | ?left-lift-poses
9  |   | ?right-lift-poses
10 |   | ?left-2nd-lift-poses
11 |   | ?right-2nd-lift-poses
12 |   | ?joint-name
13 |   | ?environment-obj)
14 |   | (spec:property ?action-designator (type :closing))
15 |   | (spec:property ?action-designator (object ?container-designator))
16 |   | (spec:property ?container-designator (type ?container-type))
17 |   | (obj-int-object-type-subtype ?container ?container-type)
18 |   | (spec:property ?container-designator (part-of ?bsr-environment))
19 |   | (spec:property ?action-designator (?arm ?arm))
20 |   | (true)
21 |   | (and)
22 |   | (cram-robot-interfaces:robot ?bsr)
23 |   | (cram-robot-interfaces:arm ?bsr ?arm))
24 |   | (spec:property ?action-designator (distance ?distance))
25 |   | (lisp-fun get-container-link ?container-name ?bsr-environment ?container-link)
26 |   | (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
27 |   | (lisp-fun get-connecting-joint ?connecting-joint ?joint-name)
28 |   | (bsr-world ?world)
29 |   | (lisp-fun bsr-object ?world ?bsr-environment ?environment-obj)
30 |   | (lisp-fun get-container-poses-and-transform ?container-type ?gripper-opening)
31 |   | (lisp-fun get-container-poses-and-transform ?container-name ?bsr-environment
32 |   | (Container-poses ?container-transform)
33 |   | (lisp-fun get-object-grasping-poses ?container-name
34 |   | (contains-prismatic-left-close ?container-transform ?left-poses)
35 |   | (lisp-fun get-object-grasping-poses ?container-name ?right-poses)
36 |   | (contains-prismatic-right-close ?container-transform ?right-poses)
37 |   | (lisp-fun cram-mobile-pick-place-plans: extract-pick-up-manipulation-poses
38 |   | ?arm ?left-poses ?right-poses
39 |   | (left-reach-poses ?right-reach-poses)
40 |   | (right-reach-poses ?left-reach-poses)
41 |   | (left-lift-poses ?right-lift-poses))
42 |   | (> (lisp-pred identity ?left-lift-poses)
43 |   | (equal ?left-lift-poses (?left-lift-poses ?left-2nd-lift-poses)))
44 |   | (equal (NIL NIL) (?left-lift-poses ?left-2nd-lift-poses)))
45 |   | (> (lisp-pred identity ?right-lift-poses)
46 |   | (equal ?right-lift-poses (?right-lift-poses ?right-2nd-lift-poses)))
47 |   | (equal (NIL NIL) (?right-lift-poses ?right-2nd-lift-poses)))

```



```

1  | <- (desig-action-grounding ?action-designator (?p-plans:close-container ?resolved-action-designator))
2  |   | (spec:property ?action-designator (type :closing))
3  |   | (spec:property ?action-designator (object ?object-designator))
4  |   | (desig-current-designator ?object-designator ?current-object-design)
5  |   | (spec:property ?current-object-design (name ?object-name))
6  |   | (lisp-fun man-int-get-object-transform ?current-object-design ?object-transform)
7  |   | (lisp-fun man-int-calculate-object-faces ?object-transform ?flacing-robot-face ?bottom-face)
8  |   | (lisp-fun man-int-get-action-gripping-effort ?current-object-design ?effort)
9  |
10 |   | (lisp-fun man-int-get-action-gripper-opening ?current-object-design ?gripper-opening)
11 |   | (equal ?objects (?current-object-design))
12 |   | (> (lisp-fun man-int-get-action-trajectory ?closing ?object-transform ?objects
13 |   | ?trajectory)
14 |   | (lisp-fun man-int-get-traj-poses-by-label ?trajectory ?reaching ?reach-poses)
15 |   | (lisp-fun man-int-get-traj-poses-by-label ?trajectory ?grasping ?grasp-poses))
16 |   | (desig-designator ?action (type :closing)
17 |   | (object ?current-object-design)
18 |   | (name ?object-name)
19 |   | (gripper-opening ?gripper-opening)
20 |   | (effort ?effort)
21 |   | (reach-poses ?reach-poses)
22 |   | )
23 |   | )
24 |   | )
25 |   | )
26 |   | )
27 |   | )
28 |   | )
29 |   | )
30 |   | )
31 |   | )
32 |   | )
33 |   | )
34 |   | )
35 |   | )
36 |   | )
37 |   | )
38 |   | )
39 |   | )
40 |   | )
41 |   | )
42 |   | )
43 |   | )
44 |   | )
45 |   | )
46 |   | )
47 |   | )

```

Figure 2: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Holding* (23 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
16	40	6	1

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *object* instead of *container*)
- generated designator ends by mapping variables to designator "parts"

```

1  (<- (design-action-grounding ?action-designator (close-container ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    (?left-lift-pose)
9    (?right-lift-pose)
10   (?left-2nd-lift-pose)
11   (?right-2nd-lift-pose)
12   ?joint-name
13   ?environment-obj)))
14 (spec:property ?action-designator (:type :closing))
15 (spec:property ?action-designator (:object ?container-designator))
16 (spec:property ?container-designator (:type ?container-type))
17 (obj:int!object-type-subtype :container ?container-type)
18 (spec:property ?container-designator (:urdf-name ?container-name))
19 (spec:property ?container-designator (:part-of ?btr-environment))
20 (- (spec:property ?action-designator (:arm ?arm)))
21   (true)
22   (and (cram-robot-interfaces:robot ?robot)
23         (cram-robot-interfaces:arm ?robot ?arm)))
24 (spec:property ?action-designator (:distance ?distance))
25 (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
26 (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
27 (lisp-fun cl-urdfname ?connecting-joint ?joint-name)
28 (btr:bullet-world ?world)
29 (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
30 (lisp-fun obj:int!get-object-type-gripper-opening ?container-type ?gripper-opening)
31 (lisp-fun get-container-pose-and-transform ?container-name ?btr-environment
32   ?container-pose ?container-transform)
33 (lisp-fun obj:int!get-object-grasping-poses ?container-name
34   :container-prismatic :left :close ?container-transform ?left-poses)
35 (lisp-fun obj:int!get-object-grasping-poses ?container-name
36   :container-prismatic :right :close ?container-transform ?right-poses)
37 (lisp-fun cram-mobile:pick-place-plans :extract-pick-up-manipulation-poses
38   ?arm ?left-poses ?right-poses
39   ?left-reach-poses ?right-reach-poses
40   ?left-grasp-poses ?right-grasp-poses
41   ?left-lift-poses ?right-lift-poses))
42 (> (lisp-pred identity ?left-lift-poses)
43   (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose)))
44   (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
45 (> (lisp-pred identity ?right-lift-poses)
46   (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose)))
47   (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose))))
```



```

1  (<- (design-action-grounding ?action-designator (close-container ?arm
2    ?gripper-closing
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    (?left-lift-pose)
9    (?right-lift-pose)
10   (?left-2nd-lift-pose)
11   (?right-2nd-lift-pose)
12   ?joint-name ?environment-obj)))
13
14 (spec:property ?action-designator (:type :closing))
15 (spec:property ?action-designator (:object ?container-designator))
16 (spec:property ?container-designator (:type ?container-type))
17 (obj:int!object-type-subtype :container ?container-type)
18 (spec:property ?container-designator (:urdf-name ?container-name))
19 (spec:property ?container-designator (:part-of ?btr-environment))
20 (- (spec:property ?action-designator (:arm ?arm)))
21   (true)
22   (and (cram-robot-interfaces:robot ?robot)
23         (cram-robot-interfaces:arm ?robot ?arm)))
24 (spec:property ?action-designator (:distance ?distance))
25 (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
26 (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
27 (lisp-fun cl-urdfname ?connecting-joint ?joint-name)
28 (btr:bullet-world ?world)
29 (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
30 (lisp-fun obj:int!get-object-type-gripper-closing ?container-type ?gripper-closing)
31 (lisp-fun get-container-pose-and-transform ?container-name ?btr-environment
32   ?container-pose ?container-transform)
33 (lisp-fun obj:int!get-object-grasping-poses ?container-name
34   :container-prismatic :left :closed ?container-transform ?left-poses)
35 (lisp-fun obj:int!get-object-grasping-poses ?container-name
36   :container-prismatic :right :closed ?container-transform ?right-poses)
37 (lisp-fun cram-mobile:pick-place-plans :extract-pick-up-manipulation-poses
38   ?arm ?left-poses ?right-poses
39   ?left-reach-poses ?right-reach-poses
40   ?left-grasp-poses ?right-grasp-poses
41   ?left-lift-poses ?right-lift-poses))
42 (> (lisp-pred identity ?left-lift-poses)
43   (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose)))
44   (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
45 (> (lisp-pred identity ?right-lift-poses)
46   (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose)))
47   (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose))))
```

Figure 3: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Opening* (46 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
1	2	5	40

Findings:

- added / deleted lines are due to different spacing in generated designator
- 4 / 5 changes are names / descriptors (*opening -i closing*) that are changed wrong, since its about the gripper opening during the manipulation process, which is still necessary
- last change is due to one missing parentheses

```

1  <- (design.action-grounding ?action-designator (:close-container ?arm
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

```

1  <- (design.action-grounding ?action-designator (:close ?container-desig ?effort ?opening))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

Figure 4: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Picking Up* (6 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
1	42	4	1

Findings:

- tba

```

1  :- (<- (desig.action-grounding ?action-designator (close-container ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    (?left-lift-pose)
9    (?right-lift-pose)
10   (?left-2nd-lift-pose)
11   (?right-2nd-lift-pose)
12   ?joint-name
13   ?environment-obj))
14   (spec:property ?action-designator (:type :closing))
15   (spec:property ?action-designator (:object ?container-designator))
16   (spec:property ?container-designator (:type ?container-type))
17   (obj-int-to-object-type-subtype :container ?container-type)
18   (spec:property ?container-designator (:urdf-name ?container-name))
19   (spec:property ?container-designator (:part-of ?btr-environment))
20   (-> (spec:property ?action-designator (:arm ?arm))
21     (true)
22     (and (cram-robot-interfaces:robot ?robot)
23       (cram-robot-interfaces:arm ?robot ?arm)))
24     (spec:property ?action-designator (:distance ?distance))
25     (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
26     (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
27     (lisp-fun cl-urdf:name ?connecting-joint ?joint-name)
28     (btr-bullet-world ?world)
29     (lisp-fun btr-object ?world ?btr-environment ?environment-obj)
30     (lisp-fun obj-int-get-object-type-gripper-opening ?container-type ?gripper-opening)
31     (lisp-fun get-container-pose-and-transform ?container-name ?btr-environment
32       (?container-pose ?container-transform))
33     (lisp-fun obj-int-get-object-grasping-poses ?container-name
34       :container-prismatic :left :clos ?container-transform ?left-poses)
35     (lisp-fun obj-int-get-object-grasping-poses ?container-name
36       :container-prismatic :right :close ?container-transform ?right-poses)
37     (lisp-fun cram-mobile-pick-place-plan extract-pick-up-manipulation-poses
38       ?arm ?left-poses ?right-poses
39       (?left-reach-poses ?right-reach-poses
40         ?left-grasp-poses ?right-grasp-poses
41         ?left-lift-poses ?right-lift-poses))
42     (-> (lisp-pred identity ?left-lift-poses)
43       (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose))
44       (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
45     (-> (lisp-pred identity ?right-lift-poses)
46       (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose))
47       (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose)))))

1  :- (<- (desig.action-grounding ?action-designator (close ?container-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14   (spec:property ?action-designator (:type :closing))
15   (spec:property ?action-designator (:object ?container-designator))
16   (spec:property ?container-designator (:type :container))
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33   (-> (lisp-fun obj-int-get-container-closing-poses ?container-name ?container-designator
34     (?closing-poses))
35     (lisp-fun extract-closing-manipulation-poses ?closing-poses ?left-reach-poses
36       (?right-reach-poses ?left-grasp-poses ?right-grasp-poses
37         ?left-retract-poses ?right-retract-poses)
38       (or (and (cram-robot-interfaces:robot ?robot)
39         (cram-robot-interfaces:arm ?robot :left)
40         (cram-robot-interfaces:arm ?robot :right))
41       (format "WARNING: Could not find robot or arms for closing container ~a.~%" ?container-designator))))
```

Figure 5: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Placing Down* (15 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
5	37	8	2

Findings:

- designator definition contains only one variable instead of 13
- generated designator focuses on the necessary poses but nothing else
- generation adds block about `cram-robot-interfaces` (3 lines)

```

1  [+< (design-action-grounding ?action-designator (close-container ?arm)
2    ?grpper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    (?left-lift-pose)
9    (?right-lift-pose)
10   (?left-2nd-lift-pose)
11   (?right-2nd-lift-pose)
12   ?cont-name
13   ?environment-obj)]
14   (
  


```

1 [+< (design-action-grounding ?action-designator (close-container ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```


```

Figure 6: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Pouring* (45 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
35	37	8	2

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *object* instead of *container*)
- high focus on finding the fitting (gripping) poses for two robot arms
- generated designator ends by mapping variables to designator "parts"

```

1  | <- (design-action-grounding ?action-designator (close-container ?arm
2  |   ?gripper-opening
3  |   ?distance
4  |   ?left-reach-poses
5  |   ?right-reach-poses
6  |   ?left-grasp-poses
7  |   ?right-grasp-poses
8  |   ?left-lift-poses)
9  |   ?right-lift-poses)
10 |   ?left-2nd-lift-poses)
11 |   ?right-2nd-lift-poses)
12 |   ?point-name
13 |   ?environment-obj)
14 | 
15 |   (spec-property ?action-designator :type ?closing)
16 | 
17 |   (spec-property ?action-designator :object ?resolved-action-designator)
18 | 
19 |   (spec-property ?container-designator :type ?container-type)
20 | 
21 |   (obj-int-object-type-subtype ?container ?container-type)
22 | 
23 |   (spec-property ?container-designator :current-name ?container-name)
24 | 
25 |   (spec-property ?container-designator :part-of ?btr-environment)
26 | 
27 |   (-> (spec-property ?action-designator :arm ?arm))
28 | 
29 |   (true)
30 | 
31 | 
32 | 
33 | 
34 | 
35 | 
36 | 
37 | 
38 | 
39 | 
40 | 
41 | 
42 | 
43 | 
44 | 
45 | 
46 | 
47 | 
48 | 
49 | 
50 | 
51 | 
52 | 
53 | 
54 | 
55 | 
56 | 
57 | 
58 | 
59 | 
60 | 
61 | 

```



```

1  | <- (design-action-grounding ?action-designator (close-container ?resolved-action-designator))
2  | 
3  | 
4  | 
5  | 
6  | 
7  | 
8  | 
9  | 
10 | 
11 | 
12 | 
13 | 
14 | 
15 | 
16 | 
17 | 
18 | 
19 | 
20 | 
21 | 
22 | 
23 | 
24 | 
25 | 
26 | 
27 | 
28 | 
29 | 
30 | 
31 | 
32 | 
33 | 
34 | 
35 | 
36 | 
37 | 
38 | 
39 | 
40 | 
41 | 
42 | 
43 | 
44 | 
45 | 
46 | 
47 | 
48 | 
49 | 
50 | 
51 | 
52 | 
53 | 
54 | 
55 | 
56 | 
57 | 
58 | 
59 | 
60 | 
61 | 

```

Figure 7: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Slicing* (40 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
29	36	8	3

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *object* instead of *container*)
- high focus on finding the fitting (gripping) poses for two robot arms
- generated designator ends by mapping variables to designator "parts"

```

1  |<- (design-action-grounding ?action-designator (close-container ?arm
2  |  |?gripper-opening
3  |  |?distance
4  |  |?left-reach-poses
5  |  |?right-reach-poses
6  |  |?left-grasp-poses
7  |  |?right-grasp-poses
8  |  |(?left-lift-poses)
9  |  |(?right-lift-poses)
10 |  |(?left-2nd-lift-poses)
11 |  |(?right-2nd-lift-poses)
12 |  |?point-name
13 |  |?environment-object)
14 |  |(spec:property ?action-designator :type :closing-container)
15 |  |(spec:property ?action-designator :object ?container-designator)
16 |  |(spec:property ?container-designator :type ?container-type))
17 |  |(obj-int:property ?container-designator ?current-name ?container-name)
18 |  |(spec:property ?container-designator :parent ?left-environment)
19 |  |(spec:property ?container-designator :arm ?arm))
20 |  |(> (spec:property ?action-designator :arm ?arm))
21 |  |
22 |  |(and (cram-robot-interfaces:robot ?robot)
23 |  |(cram-robot-interfaces:arm ?arm)
24 |  |(spec:property ?action-designator :collision-distance ?distance)
25 |  |(lisp-fun get-container-link ?container-name ?distance ?container-link)
26 |  |(lisp-fun get-connecting-joint ?container-link ?connecting-joint)
27 |  |(lisp-fun c-lift:name ?connecting-joint ?point-name)
28 |  |(lisp-fun btr:btr ?btr-environment ?environment-object)
29 |  |(lisp-fun obj-int:get-object-type ?gripper-opening ?container-type ?gripper-opening)
30 |  |(lisp-fun get-container-poses-and-transform ?container-name ?btr-environment
31 |  |(lisp-fun get-container-link ?container-link ?container-transform)
32 |  |(lisp-fun obj-int:get-object-type ?container-name ?container-name)
33 |  |(lisp-fun obj-int:get-object-poses ?container-name ?container-name)
34 |  |(lisp-fun obj-int:get-object-grasping-poses ?container-name
35 |  |(lisp-fun container-prismatic ?left-close ?container-transform ?left-poses)
36 |  |(lisp-fun container-prismatic ?right-close ?container-transform ?right-poses)
37 |  |(lisp-fun crm:mobile-pick-place-plan :extract:pick-up-manipulation-poses
38 |  |(lisp-fun ?arm ?left-poses ?right-poses
39 |  |(lisp-fun ?left-reach-poses ?right-reach-poses
40 |  |(lisp-fun ?left-grasp-poses ?right-grasp-poses
41 |  |(lisp-fun ?left-lift-poses ?right-lift-poses))
42 |  |(> (isp-pred identity ?left-poses))
43 |  |(lisp-pred identity ?right-poses)
44 |  |(lisp-pred identity ?left-lift-poses))
45 |  |(lisp-pred identity ?right-lift-poses))
46 |  |(lisp-pred identity ?left-2nd-lift-poses))
47 |  |(lisp-pred identity ?right-2nd-lift-poses))
48 |  |(lisp-pred identity ?left-2nd-lift-poses))
49 |  |(lisp-pred identity ?right-2nd-lift-poses))

```



```

1  |<- (design-action-grounding ?action-designator (close-container ?resolved-action-designator))
2  |  |(spec:property ?action-designator :type :closing-container)
3  |  |(spec:property ?action-designator :grasp ?grasp))
4  |  |(spec:property ?action-designator :arm ?arm))
5  |  |(spec:design-prop ?action-designator ?container ?container-designator)
6  |  |(design-current-designator ?container-designator ?current-container-designator)
7  |  |(spec:property ?current-container-designator :type ?container-type))
8  |  |(spec:property ?current-container-designator :current-container-designator)
9  |  |(equal ?container ?current-container-designator))
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |  |(design-design-prop ?action-designator :collision-mode ?collision-mode)
26 |  |(lisp-fun (equal ?arm ?left)
27 |  |(and (lisp-fun get-trajectory ?closing-container ?arm ?current-grasp T ?container ?lists)
28 |  |(lisp-fun man-int:get-traj-poses-by-label ?lists ?closing-container
29 |  |?left-closing-poses)
30 |  |(lisp-fun man-int:get-traj-poses-by-label ?lists :initial
31 |  |?left-initial-poses)
32 |  |(and (spec:property ?left-closing-poses NIL)
33 |  |(equal ?left-initial-poses NIL)))
34 |  |(lisp-fun (equal ?arm ?right)
35 |  |(and (lisp-fun get-trajectory ?closing-container ?arm ?current-grasp T ?container ?lists)
36 |  |(lisp-fun man-int:get-traj-poses-by-label ?lists ?closing-container
37 |  |?right-closing-poses)
38 |  |(lisp-fun man-int:get-traj-poses-by-label ?lists :initial
39 |  |?right-initial-poses))
40 |  |(and (spec:property ?right-closing-poses NIL)
41 |  |(equal ?right-initial-poses NIL)))
42 |  |(design-designator ?action-designator ?closing-container)
43 |  |(collission-mode ?closing-container)
44 |  |(left-closing-poses ?left-closing-poses)
45 |  |(right-closing-poses ?right-closing-poses)
46 |  |(left-initial-poses ?left-initial-poses)
47 |  |(right-initial-poses ?right-initial-poses))
48 |  |(spec:property ?resolved-action-designator))

```

Figure 8: Left: Manual designator for the action *Closing* (47 lines). Right: Generated designator based on *Wiping* (32 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
20	35	12	0

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *designator* instead of *type*)
- generated designator focuses on the necessary poses and trajectories
- generated designator ends by mapping variables to designator "parts"

```

1  :-> (design action-grounding ?action-designator (slice-in-half ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73

```

```

1  :-> (design action-grounding ?action-designator (halve ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    ?left-lift-poses
9    ?right-lift-poses
10   ?left-2nd-lift-poses
11   ?right-2nd-lift-poses
12   ?joint-name
13   ?environment-obj))
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73

```

Figure 9: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Closing* (47 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
31	42	14	2

Findings:

- designator definition contains 13 variables instead of only one
- some minor renaming (e.g. *cutting* instead of *halving*)
- generated designator does not cover the "cutting" part but focuses mainly on poses
- generated designator does not end by mapping variables to designator "parts"
- generation adds block about **cram-robot-interfaces** (2 lines)

```

1  ;< (design-action-grounding ?action-designator (slice-in-half ?resolved-action-designator))
2  (spec:property ?action-designator (:type :shaving))
3  (spec:property ?action-designator (?object ?object-designator))
4  (design-current-designator ?object-designator ?current-object-desig)
5  (spec:property ?current-object-desig (:type ?object-type))
6  (spec:property ?current-object-desig (:name ?object-name))
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1  ;< (design-action-grounding ?action-designator (ip-plans-cut ?resolved-action-designator))
2  (spec:property ?action-designator (:type :shaving))
3  (spec:property ?action-designator (?object ?object-designator))
4  (design-current-designator ?object-designator ?current-object-desig)
5  (spec:property ?current-object-desig (:type ?object-type))
6  (spec:property ?current-object-desig (:name ?object-name))
7  (lisp-fun-man-int-get-object-transform ?current-object-desig ?object-transform)
8  (lisp-fun-man-int-calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
9  (> (spec:property ?action-designator (?arm ?arm))
10  (true)
11  (man-int:robot-free-hand ?_ ?arm))
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

Figure 10: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Holding* (48 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
3	13	29	16

Findings:

- some minor renaming (e.g. *cutting* instead of *halving*)
- generated designator does not cover the necessary poses for "halving" (*?object-half-pose*)

```

1  :-> (desig.action-grounding ?action-designator (slice-in-half ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13  |  :-> (spec.property ?action-designator (:type :halving))
14  |  (spec.property ?action-designator (:object ?object-designator))
15  |  (desig.current-object-designator ?object-designator ?current-object-design)
16  |  (spec.property ?current-object-design (:type ?object-type))
17  |  (spec.property ?current-object-design (:name ?object-name))
18  |  (-> (spec.property ?action-designator (:arm ?arm))
19  |         (true)
20  |         (man-int:robot-free-hand ?, ?arm))
21  |         (lisp-fun man-int:get-object-old-transform ?current-object-design ?object-transform)
22  |         (lisp-fun man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face))
23  |         (-> (man-int:object-rotationally-symmetric ?object-type)
24  |             (equal ?rotationally-symmetric t)
25  |             (equal ?rotationally-symmetric nil))
26  |             (-> (spec.property ?action-designator (:grasp ?grasp))
27  |                 (true)
28  |                 (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
29  |                     (member ?grasp ?grasps)))
30  |                 (-> (spec.property ?action-designator (:object-half-pose ?object-half-pose))
31  |                     (true)
32  |                     (format "Please infer where to cut the object, or use the query system to infer it here"))
33  |                     (+> (lisp-fun man-int:get-action-grasping-effort ?object-type ?effort)
34  |                         (lisp-fun man-int:action-gripper-opening ?object-type ?gripper-opening)
35  |                         (equal ?object-type ?current-object-design))
36  |                         (-> (equal ?arm ?arm)
37  |                             (and (lisp-fun man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
38  |                                 ?left-halving-pose)
39  |                                 (lisp-fun man-int:get-traj-poses-by-label ?left-halving-pose :halving-up
40  |                                 ?left-slice-up-poses)
41  |                                 (lisp-fun man-int:get-traj-poses-by-label ?left-halving-pose :halving-down
42  |                                 ?left-slice-down-poses))
43  |                                 (and ?left-slice-up-poses NIL)
44  |                                 (equal ?left-slice-down-poses NIL))
45  |                                 (-> (equal ?arm ?right)
46  |                                     (and (lisp-fun man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
47  |                                         ?right-halving-pose)
48  |                                         (lisp-fun man-int:get-traj-poses-by-label ?right-halving-pose :halving-up
49  |                                         ?right-slice-up-poses)
50  |                                         (lisp-fun man-int:get-traj-poses-by-label ?right-halving-pose :slice-down
51  |                                         ?right-slice-down-poses))
52  |                                         (and ?equal ?right-slice-up-poses NIL)
53  |                                         (equal ?right-slice-down-poses NIL))
54  |                                         (-> (desig:design-prop ?action-designator (:collision-mode ?collision-mode))
55  |                                             (true)
56  |                                             (equal ?collision-mode nil))
57  |                                             (desig:designator ((type :halving)
58  |                                                 (:object ?current-object-design)
59  |                                                 (:object-name ?object-name)
60  |                                                 (:arm ?arm)
61  |                                                 (:gripper-opening ?gripper-opening)
62  |                                                 (:effort ?effort)
63  |                                                 (:grasp ?grasp)
64  |                                                 (:left-slice-up-poses ?left-slice-up-poses)
65  |                                                 (:right-slice-up-poses ?right-slice-up-poses)
66  |                                                 (:left-slice-down-poses ?left-slice-down-poses)
67  |                                                 (:right-slice-down-poses ?right-slice-down-poses)
68  |                                                 (:collision-mode ?collision-mode))
69  |                                                 ?resolved-action-designator)))
1  :-> (desig.action-grounding ?action-designator (halve ?arm
2  |  ?gripper-opening
3  |  ?distance
4  |  ?left-reach-poses
5  |  ?right-reach-poses
6  |  ?left-grasp-poses
7  |  ?right-grasp-poses
8  |  (?left-lift-pose)
9  |  (?right-lift-pose)
10 |  (?left-2nd-lift-pose)
11 |  (?right-2nd-lift-pose)
12 |  ?joint-name ?object))
13  |  (spec.property ?action-designator (:type :cutting))
14  |  (spec.property ?action-designator (:object ?object-designator))
15  |  (spec.property ?object-designator (:type ?food))
16  |  (spec.property ?object-designator (:part-of ?bitr-environment))
17  |  (spec.property ?object-designator (:urdf-name ?object-name))
18  |  (-> (spec.property ?action-designator (:arm ?arm))
19  |         (and (cram-robot-interfaces:robot ?robot)
20  |             (cram-robot-interfaces:arm ?robot ?arm)))
21
22
23
24
25
26  |  (spec.property ?action-designator (:distance ?distance))
27
28  |  (lisp-fun get-object-link ?object-name ?bitr-environment ?object-link)
29
30
31
32
33  |  (lisp-fun get-connecting-joint ?object-link ?connecting-joint)
34  |  (lisp-fun cram-robot-interface:robot ?world)
35  |  (lisp-fun ltr:object ?world ?bitr-environment)
36  |  (lisp-fun obj:int:get-object-type-gripper-opening ?food ?gripper-opening)
37  |  (lisp-fun obj:int:get-object-poses ?object-name ?bitr-environment
38  |  (?object-pose ?object-transform))
39
40
41  |  (lisp-fun obj:int:get-object-grasping-poses ?object-name
42  |  ?food ?left:open ?object-transform ?left-poses)
43
44
45
46  |  (lisp-fun obj:int:get-object-grasping-poses ?object-name
47  |  ?food ?right:open ?object-transform ?right-poses)
48  |  (lisp-fun cram-mobile:pick-place-plans:extract-pick-up-manipulation-poses
49  |  ?arm ?left-poses ?right-poses
50  |  (?left-reach-poses ?right-reach-poses
51  |  ?left-grasp-poses ?right-grasp-poses)
52
53  |  (?left-lift-poses ?right-lift-poses))
54  |  (-> (lisp-pred identity ?right-lift-poses)
55  |      (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose))
56  |      (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
57
58
59
60
61
62
63
64
65  |  (-> (lisp-pred identity ?right-lift-poses)
66  |      (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose)))
67  |      (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose)))
68
69

```

Figure 11: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Opening* (45 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
29	42	13	3

Findings:

- designator definition contains 12 variables instead of only one
- some minor renaming (e.g. *cutting* instead of *halving*)
- generated designator does not cover the "cutting" part but focuses mainly on poses
- generated designator does not end by mapping variables to designator "parts"
- generation adds block about `cram-robot-interfaces` (2 lines)

```

1  (defun (<:desig-action-grounding ?action-designator [slice-in-half ?resolved-action-designator])
2    (spec:property ?action-designator (:type :halving))
3    (spec:property ?action-designator (:object ?object-designator))
4    (design-current-designator ?object-designator ?current-object-desig)
5    (spec:property ?current-object-desig (:type ?object-type))
6    (spec:property ?current-object-desig (:name ?object-name))
7    (spec:property ?action-designator (:arm ?arm))
8    (true)
9    (man-int:robot-free-hand ?_ ?arm))
10   (lisp-fun man-int:get-object-old-transform ?current-object-desig ?object-transform)
11   (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
12   (> (man-int:object-rotationally-symmetric ?object-type)
13     (equal ?rotationally-symmetric NIL))
14   (> (spec:property ?action-designator (:grasp ?grasp)))
15     (true)
16   (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
17     (member ?grasp ?grasps)))
18   (> (spec:property ?action-designator (:object-half-pose ?object-half-pose))
19     (true)
20   (format "Please infer where to cut the object, or use the query system to infer it here"))
21   (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
22   (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
23   (equal ?objects (?current-object-desig))
24   (> (equal ?arm ?left)
25     (and (lisp-fun man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
26       ?left-halving-poses)
27       (lisp-fun man-int:get-traj-poses-by-label ?left-halving-poses :halving-up
28         ?left-slice-up-poses)
29       (lisp-fun man-int:get-traj-poses-by-label ?left-halving-poses :halving-down
30         ?left-slice-down-poses))
31     (and (equal ?left-slice-up-poses NIL)
32       (equal ?left-slice-down-poses NIL)))
33   (> (equal ?arm ?right)
34     (and (lisp-fun man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
35       ?right-halving-poses)
36       (lisp-fun man-int:get-traj-poses-by-label ?right-halving-poses :halving-up
37         ?right-slice-up-poses)
38       (lisp-fun man-int:get-traj-poses-by-label ?right-halving-poses :slice-down
39         ?right-slice-down-poses))
40     (and (equal ?right-slice-up-poses NIL)
41       (equal ?right-slice-down-poses NIL)))
42   (> (desig-prop ?action-designator (:collision-mode ?collision-mode))
43     (true))
44   (equal ?collision-mode NIL)
45   (desig-designator :action ((:type :halving)
46     (object ?current-object-desig)
47     (object-name ?object-name)
48     (arm ?arm)
49     (gripper-opening ?gripper-opening)
50     (effort ?effort)
51     (grasp ?grasp)
52     (left-halving-poses ?left-slice-up-poses)
53     (left-slice-up-poses ?left-slice-up-poses)
54     (right-slice-up-poses ?right-slice-up-poses)
55     (left-slice-down-poses ?left-slice-down-poses)
56     (right-slice-down-poses ?right-slice-down-poses)
57     (collision-mode ?collision-mode))
58     (?resolved-action-designator)))
59

```



```

1  (defun (<:desig-action-grounding ?action-designator [halve ?current-object-desig ?knife ?cutting-plane])
2    (spec:property ?action-designator (:type :halving))
3    (spec:property ?action-designator (:object ?object-designator))
4    (design-current-designator ?object-designator ?current-object-desig)
5    (spec:property ?current-object-desig (:type ?object-type))
6    (spec:property ?current-object-desig (:name ?object-name))
7    (spec:property ?action-designator (:knife ?knife))
8    (true)
9    (and (cram-robot-interfaces:robot ?robot)
10      (cram-robot-interfaces:end-effector ?robot ?knife)))
11   (lisp-fun obj-int:get-object-transform ?current-object-desig ?object-transform)
12   (lisp-fun obj-int:calculate-object-faces ?object-transform (?top-face ?bottom-face))
13   (lisp-fun obj-int:get-object-type-cutting-plane ?object-type ?top-face ?bottom-face ?cutting-plane))
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

Figure 12: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Picking Up* (13 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
3	48	4	6

Findings:

- tba

```
1  |(<- (design-action-grounding ?action-designator (:slice-in-half ?resolved-action-designator))
2  |  (spec-property ?action-designator (type :halving))
3  |  (spec-property ?action-designator (object ?object-designator))
4  |  (spec-property ?current-object-designator ?current-object-desig)
5  |  (spec-property ?current-object-desig (type ?object-type))
6  |  (spec-property ?current-object-desig (name ?object-name))
7  |  (> (spec-property ?action-designator (arm ?arm))
8  |    (true)
9  |    (man-int-robot-free-hand ??arm))
10 |    (lisp-fun man-int-get-object-old-transform ?current-object-desig ?object-transform)
11 |
12 |  (lisp-fun man-int-calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
13 |  (> (man-int-object-rotationally-symmetric ?object-type)
14 |    (equal ?rotationally-symmetric t)
15 |    (equal ?rotationally-symmetric nil))
16 |  (> (spec-property ?action-designator (:grasp ?grasp))
17 |    (true)
18 |    (and (lisp-fun man-int-get-action-grasps ?object-type ?arm ?object-transform ?grasps)
19 |      (member ?grasp ?grasps)))
20 |
21 |
22 |  (> (spec-property ?action-designator (:object-half-pose ?object-half-pose))
23 |    (true)
24 |    (format "Please infer where to cut the object, or use the query system to infer it here"))
25 |    (lisp-fun man-int-get-action-gripping-effect ?object-type ?effort)
26 |    (lisp-fun man-int-get-action-gripper-opening ?object-type ?gripper-opening)
27 |    (equal ?objects (Current-object-designator))
28 |    (> (equal ?arm ?left)
29 |      (and (lisp-fun man-int-get-action-trajectory :halving) ?arm ?grasp T ?objects
30 |        ?left-halving-pose)
31 |        (lisp-fun man-int-get-traj-poses-by-label ?left-halving-pose :halvin-up
32 |          ?left-slice-up-poses)
33 |        (lisp-fun man-int-get-traj-poses-by-label ?left-halving-pose :halvin-down
34 |          ?left-slice-down-poses))
35 |      (and (equal ?left-slice-up-poses NIL)
36 |        (equal ?left-slice-down-poses NIL)))
37 |    (> (equal ?arm ?right)
38 |      (and (lisp-fun man-int-get-action-trajectory :halving) ?arm ?grasp T ?objects
39 |        ?right-halving-pose)
40 |        (lisp-fun man-int-get-traj-poses-by-label ?right-halving-pose :halving-up
41 |          ?right-slice-up-poses)
42 |        (lisp-fun man-int-get-traj-poses-by-label ?right-halving-pose :slice-down
43 |          ?right-slice-down-poses))
44 |      (and (equal ?right-slice-up-poses NIL)
45 |        (equal ?right-slice-down-poses NIL)))
46 |    (> (design-prop ?action-designator (:collision-mode ?collision-mode))
47 |      (true)
48 |      (equal ?collision-mode nil))
49 |    (design-designator :action (type :halving)
50 |      (object ?current-object-design)
51 |      (object-name ?object-name)
52 |      (arm ?arm)
53 |      (gripper-opening ?gripper-opening)
54 |      (effort ?effort)
55 |      (grasp ?grasp)
56 |      (left-slice-up-poses ?left-slice-up-poses)
57 |      (right-slice-up-poses ?right-slice-up-poses)
58 |      (left-slice-down-poses ?left-slice-down-poses)
59 |      (right-slice-down-poses ?right-slice-down-poses)
60 |      (collision-mode ?collision-mode))
61 |    ?resolved-action-designator)
62 |
63 |(<- (design-action-grounding ?action-designator (halve ?current-object-designator ?arm)
64 |  (left-cut-poses ?right-cut-poses)
65 |  (left-cut-poses ?right-cut-poses)
66 |  (left-retract-poses ?right-retract-poses)
67 |  (left-retract-poses ?right-retract-poses))
68 |
69 |  (spec-property ?action-designator (type :cutting))
70 |  (> (spec-property ?action-designator (arm ?arm))
71 |    (spec-property ?action-designator (object ?object-designator))
72 |
73 |    (or (cpoe:object-in-hand ?object-designator ?arm)
74 |      (and (format "Wanted to cut an object ~a with arm ~a, ~a, but it's not in the arm. ~%" ?object-designator ?arm)))
75 |    )
76 |  )
77 |
78 |  (> (spec-property ?action-designator (:object ?object-designator))
79 |    (and (cram-robot-interfaces:robot ?robot)
80 |      (cram-robot-interfaces:arm ?robot ?arm)
81 |      (cpoe:object-in-hand ?object-designator ?arm)))
82 |    (once (or (cpoe:object-in-hand ?object-designator ?arm)
83 |      (spec-prop ?current-object-designator (type ?object-type)))
84 |      (spec-property ?current-object-designator (name ?object-name))
85 |      (obj-int:object-type-grasp ?object-type ?grasp)
86 |      (lisp-fun obj-int-get-object-transform ?current-object-designator ?object-transform)
87 |      (lisp-fun obj-int-get-object-pose ?current-object-designator ?object-pose)
88 |      (lisp-fun obj-int-get-object-bounding-box ?current-object-designator ?object-bbox)
89 |      (lisp-fun extract-cutting-poses ?arm ?object-pose ?object-bbox ?object-transform)
90 |
91 |    )
92 |
93 |    (lisp-fun extract-manipulation-poses ?arm ?left-cut-poses ?right-cut-poses
94 |      ?left-reach-poses ?right-reach-poses ?left-retract-poses ?right-retract-poses)))
95 |  )
```

Figure 13: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Placing Down* (30 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
21	49	8	1

Findings:

- designator definition contains 4 variables instead of only one
 - generated designator does not cover the "cutting" part but focuses mainly on poses / trajectories
 - generated designator does not end by mapping variables to designator "parts"
 - generation adds block about `cram-robot-interfaces` (3 lines)

1	<pre>(-> (design-action-grounding :action-designator (:slice-in-half ?resolved-action-designator)) (spec:property :action-designator (:type :halving)) (spec:property :action-designator (:object ?object-designator)) (design:current-designator ?object?object-designator) (spec:property ?current-object-desig (:type ?object-type)) (spec:property ?current-object-desig (:name ?object-name)) (lisp-fun man-int: get-object-transform ?current-object-designator) (lisp-fun man-int: calculate-object-faces ?object-transform :?facing-robot-face ?bottom-face) (-> (man-int-object-rotationally-symmetric ?object-type) (=equal ?rotationally-symmetric t) (=equal ?rotationally-symmetric nil)) (-> (spec:property :action-designator (:grasp ?grasp)) (true) (and (lisp-fun man-int: get-action-grasps ?object-type ?arm ?object-transform ?grasps) (member ?grasp ?grasps))) (-> (spec:property :action-designator (:object-half-pose ?object-half-pose)) (format "Please infer where to cut the object, or use the query system to infer it here") (lisp-fun man-int: get-action-gripping-effort ?object-type ?effort) (lisp-fun man-int: get-action-gripper-opening ?object-type ?gripper-opening) (=equal ?objects (current-object-design)) (-> (=equal ?arm :left) (and (lisp-fun man-int: get-action-trajectory :halving ?arm ?grasp T ?objects :?left-halving-pose) (lisp-fun man-int: get-traj-poses-by-label :left-halving-pose :halvin-up :?left-slice-up-poses) (lisp-fun man-int: get-traj-poses-by-label :left-halving-pose :halving-down :?left-slice-down-poses)) (and (=equal ?left-slice-up-poses NIL) (=equal ?left-slice-down-poses NIL))) (-> (=equal ?arm :right) (and (lisp-fun man-int: get-action-trajectory :halving ?arm ?grasp T ?objects :?right-halving-pose) (lisp-fun man-int: get-traj-poses-by-label :right-halving-pose :halving-up :?right-slice-up-poses) (lisp-fun man-int: get-traj-poses-by-label :right-halving-pose :slice-down :?right-slice-down-poses)) (and (=equal ?right-slice-up-poses NIL) (=equal ?right-slice-down-poses NIL))) (-> (design:desig-prop :action-designator (:collision-mode ?collision-mode)) (true) (=equal ?collision-mode nil)) (design:designator :action (:type :halving) (:object ?current-object-design) (:object-type ?object-type) (:object-name ?object-name) (:arms ?arms)))</pre>	1	<pre>(-> (design-action-grounding :action-designator (:halve ?resolved-action-designator)) (spec:property :action-designator (:type :cutting)) (spec:property :action-designator (:object ?object-designator)) (design:current-designator ?object?object-designator) (spec:property ?current-object-desig (:type ?object-type)) (spec:property ?current-object-desig (:name ?object-name)) (lisp-fun man-int: get-object-transform ?current-object-designator) (lisp-fun man-int: calculate-object-faces ?object-transform :?facing-robot-face ?bottom-face) (-> (spec:property :action-designator (:arms ?arms)) (true) (and (man-int:robot-free-hand ?_?arm) (=equal ?arms ?arm)))) (-> (spec:property :action-designator (:grasp ?grasp)) (true) (and (member ?arm ?arms) (lisp-fun man-int: get-action-grasps ?object-type ?arm ?object-transform ?grasps) (member ?grasp ?grasps))) (-> (spec:property :action-designator (:left ?left)) (lisp-fun man-int: get-action-gripping-effort ?object-type ?effort) (lisp-fun man-int: get-action-gripper-opening ?object-type ?gripper-opening) (=equal ?objects (current-object-design)) (-> (=member ?left ?arms) (and (lisp-fun man-int: get-action-trajectory :cutting ?left ?grasp T ?objects :?left-cutting-pose) (lisp-fun man-int: get-traj-poses-by-label :left-cutting-pose :approach :?left-approach-poses) (lisp-fun man-int: get-traj-poses-by-label :left-cutting-pose :cutting :?left-cutting-poses)) (and (=equal ?left-cutting-poses NIL) (=equal ?left-cutting-poses NIL))) (-> (=member ?right ?arms) (and (lisp-fun man-int: get-action-trajectory :cutting ?right ?grasp T ?objects :?right-cutting-pose) (lisp-fun man-int: get-traj-poses-by-label :right-cutting-pose :approach :?right-approach-poses) (lisp-fun man-int: get-traj-poses-by-label :right-cutting-pose :cutting :?right-cutting-poses)) (and (=equal ?right-cutting-poses NIL) (=equal ?right-cutting-poses NIL))) (-> (design:desig-prop :action-designator (:collision-mode ?collision-mode)) (true) (=equal ?collision-mode nil)) (design:designator :action (:type :cutting) (:object ?current-object-design) (:object-type ?object-type) (:object-name ?object-name) (:arms ?arms)))</pre>
2		2	<pre>(:grasp ?grasp) (left-approach-poses ?left-approach-poses) (right-approach-poses ?right-approach-poses) (left-cutting-poses ?left-cutting-poses) (right-cutting-poses ?right-cutting-poses) (collision-mode ?collision-mode) ?resolved-action-designator)</pre>

Figure 14: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Pouring* (53 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
5	10	29	19

Findings:

- mainly renaming (e.g. *cutting* instead of *halving*)
 - generated designator does not cover the necessary poses for "halving" (*?object-half-pose*)
 - generated designator shortens the mapping of variables to designator "parts" (misses the *:gripper-opening*)

```

1  |  (spec-action-grounding ?action-designator (:slice-in-half ?resolved-action-designator))
2  |  (spec-property ?action-designator (:type :halving))
3  |  (spec-property ?action-designator (:object ?object-designator))
4  |  (design-current-designator ?object-designator ?current-object-desig)
5  |  (spec-property ?current-object-desig (:type ?object-type))
6  |  (spec-property ?current-object-desig (:name ?object-name))
7  |  (spec-property ?action-designator (:arm ?arm))
8  |  (true)
9  |  (man-int:robot-free-hand ? ?arm)
10 |  (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
11 |  (> (man-int:object-rotationally-symmetric ?object-type)
12 |  (equal ?rotationally-symmetric t)
13 |  (equal ?rotationally-symmetric nil))
14 |  (spec-property ?action-designator (:grasp ?grasp))
15 |  (true)
16 |  (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
17 |  (member ?grasp ?grasps)))
18 |  (> (spec-property ?action-designator (:object-half-pose) ?object-half-pose))
19 |  (true)
20 |  (format "Please infer where to cut the object, or use the query system to infer it here")
21 |  (> (spec-property ?action-designator (:object-half-pose)))
22 |  (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
23 |  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
24 |  (equal ?objects (current-object-designator))
25 |  (> (equal ?arm :left)
26 |  (and (lisp-fun man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
27 |  ?left-halving-pose)
28 |  (lisp-fun man-int:get-traj-poses-by-label ?left-halving-pose :halving-up
29 |  ?left-slice-up-poses)
30 |  (lisp-fun man-int:get-traj-poses-by-label ?left-halving-pose :halving-down
31 |  ?left-slice-down-poses)
32 |  (and (equal ?left-slice-up-poses NIL)
33 |  (equal ?left-slice-down-poses NIL)))
34 |  (> (equal ?arm :right)
35 |  (and (lisp-fun man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
36 |  ?right-halving-pose)
37 |  (lisp-fun man-int:get-traj-poses-by-label ?right-halving-pose :halving-up
38 |  ?right-slice-up-poses)
39 |  (lisp-fun man-int:get-traj-poses-by-label ?right-halving-pose :slice-down
40 |  ?right-slice-down-poses)
41 |  (and (equal ?right-slice-up-poses NIL)
42 |  (equal ?right-slice-down-poses NIL)))
43 |  (> (design-desig-prop ?action-designator (:collision-mode ?collision-mode))
44 |  (true)
45 |  (equal ?collision-mode nil))
46 |  (design-designator action (:type :halving)
47 |  (:object ?current-object-design)
48 |  (:object-name ?object-name)
49 |  (:arm ?arm)
50 |  (:gripper-opening ?gripper-opening)
51 |  (:effort ?effort)
52 |  (:grasp ?grasp)
53 |  (:left-slice-up-poses ?left-slice-up-poses)
54 |  (:right-slice-up-poses ?right-slice-up-poses)
55 |  (:left-slice-down-poses ?left-slice-down-poses)
56 |  (:right-slice-down-poses ?right-slice-down-poses)
57 |  (:collision-mode ?collision-mode))
58 |  ?resolved-action-designator))

1  |  (spec-action-grounding ?action-designator (:slice ?resolved-action-designator))
2  |  (spec-property ?action-designator (:type :slicing))
3  |  (spec-property ?action-designator (:object ?object-designator))
4  |  (design-current-designator ?object-designator ?current-object-desig)
5  |  (spec-property ?current-object-desig (:type ?object-type))
6  |  (spec-property ?current-object-desig (:name ?object-name))
7  |  (spec-property ?action-designator (:arm ?arm))
8  |  (true)
9  |  (man-int:robot-free-hand ? ?arm)
10 |  (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
11 |  (> (man-int:object-rotationally-symmetric ?object-type)
12 |  (equal ?rotationally-symmetric t)
13 |  (equal ?rotationally-symmetric nil))
14 |  (spec-property ?action-designator (:grasp ?grasp))
15 |  (true)
16 |  (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
17 |  (member ?grasp ?grasps)))
18 |  (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
19 |  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
20 |  (equal ?objects (current-object-designator))
21 |  (> (equal ?arm :left)
22 |  (and (lisp-fun man-int:get-action-trajectory :slicing ?arm ?grasp T ?objects
23 |  ?left-slicing-pose)
24 |  (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-pose :slice-up
25 |  ?left-slice-up-poses)
26 |  (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-pose :slice-down
27 |  ?left-slice-down-poses)
28 |  (and (equal ?left-slice-up-poses NIL)
29 |  (equal ?left-slice-down-poses NIL)))
30 |  (> (equal ?arm :right)
31 |  (and (lisp-fun man-int:get-action-trajectory :slicing ?arm ?grasp T ?objects
32 |  ?right-slicing-pose)
33 |  (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-pose :slice-up
34 |  ?right-slice-up-poses)
35 |  (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-pose :slice-down
36 |  ?right-slice-down-poses)
37 |  (and (equal ?right-slice-up-poses NIL)
38 |  (equal ?right-slice-down-poses NIL)))
39 |  (> (design-desig-prop ?action-designator (:collision-mode ?collision-mode))
40 |  (true)
41 |  (equal ?collision-mode nil))
42 |  (design-designator action (:type :slicing)
43 |  (:object ?current-object-design)
44 |  (:object-name ?object-name)
45 |  (:arm ?arm)
46 |  (:gripper-opening ?gripper-opening)
47 |  (:effort ?effort)
48 |  (:grasp ?grasp)
49 |  (:left-slice-up-poses ?left-slice-up-poses)
50 |  (:right-slice-up-poses ?right-slice-up-poses)
51 |  (:left-slice-down-poses ?left-slice-down-poses)
52 |  (:right-slice-down-poses ?right-slice-down-poses)
53 |  (:collision-mode ?collision-mode))
54 |  ?resolved-action-designator))

```

Figure 15: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Slicing* (55 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
0	3	11	44

Findings:

- deleted lines describe block that is used for inferring the position / pose necessary for meeting the exact halve
- deleted lines are the same that are added when the reference and generated action are swapped (Fig. 58)
- remaining changes are places where the keyword *slicing* was not replaced by *halving*

```

1  |(< (desig.action-grounding ?action-designator (slice-in-half ?resolved-action-designator))
2  |  (spec:property ?action-designator (:type :halving))
3  |  (spec:property ?action-designator ?object ?object-designator)
4  |  (spec:current-designator ?object-designator ?current-object-designator)
5  |  (spec:property ?current-object-designator (:type ?object-type))
6  |  (spec:property ?current-object-designator (:name ?object-name))
7  |  (spec:property ?action-designator (:arm ?arm))
8  |  (true)
9  |  (man-int:robot-free-hand 2 ?arm)
10 |  (lisp-fun:man-int:get-object-old-transform ?current-object-designator ?object-transform)
11 |  (lisp-fun:man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
12 |  (> (man-int:object-rotationally-symmetric ?object-type)
13 |    (equal ?rotationally-symmetric t)
14 |    (equal ?rotationally-symmetric nil))
15 |  (spec:property ?action-designator (:grasp ?grasp))
16 |  (true)
17 |  (and (lisp-fun:man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
18 |    (member ?grasp ?grasps))
19 |  (> (spec:property ?action-designator (:object-half-pose ?object-half-pose))
20 |    (true)
21 |    (format "Please infer where to cut the object, or use the query system to infer it here")
22 |    (lisp-fun:man-int:get-action-gripping-effort ?object-type ?effort)
23 |    (lisp-fun:man-int:get-action-gripper-opening ?object-type ?gripper-opening)
24 |    (equal ?objects ?current-object-designator)
25 |    (> (equal ?arm :left)
26 |      (and (lisp-fun:man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
27 |        ?left-halving-pose)
28 |        (lisp-fun:man-int:get-traj-poses-by-label ?left-halving-pose :halving-up
29 |          ?left-slice-up-poses)
30 |        (lisp-fun:man-int:get-traj-poses-by-label ?left-halving-pose :halving-down
31 |          ?left-slice-down-poses))
32 |        (and (equal ?left-slice-up-poses NIL)
33 |          (equal ?left-slice-down-poses NIL)))
34 |      (> (equal ?arm :right)
35 |        (and (lisp-fun:man-int:get-action-trajectory :halving ?arm ?grasp T ?objects
36 |          ?right-halving-pose)
37 |            (lisp-fun:man-int:get-traj-poses-by-label ?right-halving-pose :halving-up
38 |              ?right-slice-up-poses)
39 |            (lisp-fun:man-int:get-traj-poses-by-label ?right-halving-pose :slice-down
40 |              ?right-slice-down-poses))
41 |            (and (equal ?right-slice-up-poses NIL)
42 |              (equal ?right-slice-down-poses NIL)))
43 |          (> (design:design-prop ?action-designator (:collision-mode ?collision-mode))
44 |            (true)
45 |            (equal ?collision-mode nil))
46 |            (design:designator ?action (:type :halving)
47 |              (object ?current-object-designator)
48 |              (object-name ?object-name)
49 |              (arm ?arm)
50 |              (gripper-opening ?gripper-opening)
51 |              (effort ?effort)
52 |              (grasp ?grasp)
53 |              (left-slice-up-poses ?left-slice-up-poses)
54 |              (right-slice-up-poses ?right-slice-up-poses)
55 |              (left-slice-down-poses ?left-slice-down-poses)
56 |              (right-slice-down-poses ?right-slice-down-poses)
57 |              (collision-mode ?collision-mode)))
58 |            ?resolved-action-designator)))
1  |(< (desig.action-grounding ?action-designator (:type :cutting))
2  |  (spec:property ?action-designator (:grasp ?grasp))
3  |  (spec:current-designator ?object-designator ?current-object-designator)
4  |  (spec:property ?current-object-designator (:type ?object-type))
5  |  (spec:property ?current-object-designator (:name ?object-name))
6  |  (spec:property ?action-designator (:arm ?arm))
7  |  (true)
8  |  (desig:desig-prop ?action-designator (:object ?object ?object-designator))
9  |  (desig:current-designator ?object-designator ?current-object-designator)
10 |  (spec:property ?current-object-designator (:type ?object-type))
11 |  (spec:property ?current-object-designator (?equal ?object ?current-object-designator))
12 |  (equal ?grasp ?pinching)
13 |  (lisp-fun:differentiate-object-types ?grasp ?object ?current-grasp)
14 |  (equal ?grasp ?current-grasp)
15 |  (design:design-prop ?action-designator (:collision-mode ?collision-mode))
16 |  (true)
17 |  (lisp-fun:man-int:get-traj-poses-by-label ?lists :cutting
18 |    ?left-cutting-poses)
19 |  (lisp-fun:man-int:get-traj-poses-by-label ?lists :initial
20 |    ?left-initial-poses)
21 |  (and (equal ?left-cutting-poses NIL)
22 |    (equal ?left-initial-poses NIL)))
23 |  (> (equal ?arm :right)
24 |    (and (lisp-fun:man-int:get-traj-poses-by-label ?lists :cutting
25 |      ?right-cutting-poses)
26 |        (lisp-fun:man-int:get-traj-poses-by-label ?lists :initial
27 |          ?right-initial-poses))
28 |        (and (equal ?right-cutting-poses NIL)
29 |          (equal ?right-initial-poses NIL)))
30 |        (design:designator ?action (:type :cutting)
31 |          (collision-mode ?collision-mode)))
32 |        (true)
33 |        (left-cutting-poses ?left-cutting-poses)
34 |        (right-cutting-poses ?right-cutting-poses)
35 |        (left-initial-poses ?left-initial-poses)
36 |        (right-initial-poses ?right-initial-poses)))
37 |  ?resolved-action-designator))

```

Figure 16: Left: Manual designator for the action *Halving* (58 lines). Right: Generated designator based on *Wiping* (35 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
5	28	27	3

Findings:

- some minor renaming (e.g. *cutting* instead of *halving*)
- generated designator does not cover the necessary poses for "halving" (*?object-half-pose*)
- generated designator shortens the mapping of variables to designator "parts" (misses *:object*, *:object-name*, *:arm*, *:gripper-opening*, *:effort* and *:grasp*)

```

1  <- (desig.action-grounding ?action-designator (bp-plans:pick-up ?resolved-action-designator))
2
3   (spec:property ?action-designator (type holding))
4   (spec:property ?action-designator (object ?object-designator))
5   (desig.current-designator ?object-designator ?current-object-desig)
6   (spec:property ?current-object-desig (type ?object-type))
7   (spec:property ?current-object-desig (name ?object-name))
8   (> (spec:property ?action-designator (arm ?arm))
9     (true)
10    (man-int:robot-free-hand ?, ?arm))
11    (lisp-fun man-int:get-object-transform ?current-object-desig ?object-transform)
12    (lisp-fun man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
13    (> (man-int:object-rotationally-symmetric ?object-type)
14      (equal ?rotationally-symmetric t)
15      (equal ?rotationally-symmetric nil))
16    (> (spec:property ?action-designator (grasp ?grasp))
17      (true)
18      (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
19        (lisp-fun man-int:get-action-grasps ?object-type ?effort)
20        (lisp-fun man-int:get-action-grasps ?object-type ?gripper-opening)
21        (equal ?objects (?current-object-design)))
22        (> (equal ?arm ?left)
23          (and (lisp-fun man-int:get-action-trajectory :picking-up) ?arm ?grasp T ?objects
24            ?left-trajectory)
25            (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :reaching
26              ?left-reach-poses)
27            (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :grasping
28              ?left-grasp-poses))
29            (and (equal ?left-reach-poses NIL)
30              (equal ?left-grasp-poses NIL))
31            (> (equal ?arm ?right)
32              (and (lisp-fun man-int:get-action-trajectory :picking-up) ?arm ?grasp T ?objects
33                ?right-trajectory)
34                (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :reaching
35                  ?right-reach-poses)
36                (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :grasping
37                  ?right-grasp-poses))
38                (and (equal ?right-reach-poses NIL)
39                  (equal ?right-grasp-poses NIL))
40                (desig.designator :action (type :picking-up)
41                  (object ?current-object-desig)
42                    (object-name ?object-name)
43                    (arm ?arm)
44                    (gripper-opening ?gripper-opening)
45                    (effort ?effort)
46                    (grasp ?grasp)
47                    (left-reach-poses ?left-reach-poses)
48                    (right-reach-poses ?right-reach-poses)
49                    (left-grasp-poses ?left-grasp-poses)
50                    (right-grasp-poses ?right-grasp-poses)
51                    (hold holding))
52                    ?resolved-action-designator)))
53
54
1  <- (desig.action-grounding ?action-designator (hold ?arm
2   ?gripper-opening
3   ?object-designator))
4
5   (spec:property ?action-designator (type holding))
6   (spec:property ?action-designator (object ?object-designator))
7
8
9
10  (> (spec:property ?arm ?arm))
11  (true)
12  (and (cram-robot-interfaces:robot ?robot)
13    (cram-robot-interfaces:arm ?robot ?arm)))
14
15
16
17
18
19
20
21  (lisp-fun obj-int:get-object-type-gripper-opening
22    (spec:property ?object-designator (type ?object-type))
23    (gripper-opening))
24    (spec:property ?object-designator (part-of ?environment))
25    (lisp-fun cram-robot-interfaces:object ?environment ?object-designator ?object)
26    (lisp-fun cram-robot-interfaces:grasp-object ?arm ?object ?gripper-opening)))
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

Figure 17: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Closing* (15 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
9	46	2	4

Findings:

- designator definition contains three variables instead of only one
- generated designator does not end by mapping variables to designator "parts"
- generation adds block about **cram-robot-interfaces** (2 lines)

```

1  :- (<- (design-action-grounding ?action-designator (hold ?resolved-action-designator))
2    (spec:property ?action-designator :type :holding))
3    (spec:property ?action-designator :object ?object-designator)
4    (design current-designator ?object-designator ?current-object-design)
5    (spec:property ?current-object-design :type ?object-type))
6    (spec:property ?current-object-design :name ?object-name))
7    (-> (spec:property ?action-designator :arm ?arm))
8    (true)
9    (man-int:robot-free-hand ?,?arm))
10   (lisp-fun man-int:get-object-transform ?current-object-design ?object-transform)
11   (lisp-fun man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
12   (-> (man-int:object-rotationally-symmetric ?object-type)
13     (=equal ?rotationally-symmetric t)
14     (=nil ?rotationally-symmetric nil))
15   (-> (spec:property ?action-designator :grasp ?gasp))
16   (true)
17   (-> (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
18     (member ?gasp ?grasps)))
19   (lisp-fun man-int:get-action-grapping-effort ?object-type ?effort)
20   (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
21   (=equal ?objects (?current-object-design))
22   (-> (=equal ?arm :left)
23     (and (lisp-fun man-int:get-action-trajectory :picking-up ?arm ?gasp T ?objects
24       ?left-trajectory)
25       (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :reaching
26       ?left-reach-poses)
27       (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :grasping
28       ?left-grasp-poses)))
29   (-> (=nil ?left-reach-poses NIL)
30     (=nil ?left-grasp-poses NIL))
31   (-> (=equal ?arm :right)
32     (and (lisp-fun man-int:get-action-trajectory :picking-up ?arm ?gasp T ?objects
33       ?right-trajectory)
34       (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :reaching
35       ?right-reach-poses)
36       (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :grasping
37       ?right-grasp-poses)))
38   (-> (=nil ?right-reach-poses NIL)
39     (=nil ?right-grasp-poses NIL))
40   (-> (design-designator :action ((type :holding)
41     :object ?current-object-design)
42     :object ?object-name)
43     :object ?arm)
44     :gripper-opening ?gripper-opening)
45     :effort ?effort)
46     :grasp ?gasp)
47   (-> (left-reach-poses ?left-reach-poses)
48     (right-reach-poses ?right-reach-poses)
49     (left-grasp-poses ?left-grasp-poses)
50     (right-grasp-poses ?right-grasp-poses)
51     (hold ?holding)))
52   ?resolved-action-designator)

1  :- (<- (design-action-grounding ?action-designator (hold ?resolved-action-designator))
2    (spec:property ?action-designator :type :holding))
3    (spec:property ?action-designator :object ?object-designator)
4    (design current-designator ?object-designator ?current-object-design)
5    (spec:property ?current-object-design :type ?object-type))
6    (spec:property ?current-object-design :name ?object-name))
7    (-> (spec:property ?action-designator :arm ?arm))
8    (true)
9    (man-int:robot-free-hand ?,?arm))

10   (-> (spec:property ?action-designator :grasp ?gasp))
11   (true)
12   (-> (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?grasps)
13     (member ?gasp ?grasps)))
14   (lisp-fun man-int:get-action-grapping-effort ?object-type ?effort)
15   (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
16   (=equal ?objects (?current-object-design))
17   (-> (=equal ?arm :left)
18     (and (lisp-fun man-int:get-action-trajectory :holding ?arm ?gasp T ?objects
19       ?left-holding-pose)))
20   (-> (=equal ?left-holding-pose NIL))
21   (-> (=equal ?arm :right)
22     (and (lisp-fun man-int:get-action-trajectory :holding ?arm ?gasp T ?objects
23       ?right-holding-pose))
24     (=nil ?right-holding-pose NIL))
25   (-> (design-design-prop :action-designator (:collision-mode ?collision-mode))
26     (true)
27     (=nil ?collision-mode))
28   (-> (design-designator :action ((type :holding)
29     :object ?current-object-design)
30     :object ?object-name)
31     :object ?arm)
32     :gripper-opening ?gripper-opening)
33     :effort ?effort)
34     :grasp ?gasp)
35   (-> (left-holding-pose ?left-holding-pose)
36     (right-holding-pose ?right-holding-pose)
37     (=nil ?collision-mode)))
38   ?resolved-action-designator)

```

Figure 18: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Halving* (38 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
7	21	9	22

Findings:

- some minor renaming (e.g. *holding* instead of *picking up*)
- generated designator does not cover the necessary calculations regarding trajectories for reaching and/or grasping
- generated designator shortens the mapping of variables to designator "parts" (misses *:left-grasp-poses*, *:right-grasp-poses* and *:hold* but adds *:collision-mode*)

```

1  <- (design-action-grounding ?action-designator (np-plans:pick-up ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

```

1  <- (design-action-grounding ?action-designator (hold ?arm
2    ?gripper-closing
3    ?object-designator))
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

Figure 19: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Opening* (10 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
4	46	2	4

Findings:

- designator definition contains three variables instead of only one
- generated designator does not end by mapping variables to designator "parts"
- generated designator contains almost no functionality (no trajectory calculations, etc.)
- generation adds block about `cram-robot-interfaces` (2 lines)

```

1  :- (<- (desig:action-grounding ?action-designator (pp-plans:pick-up ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

(spec:property ?action-designator (:type :holding))
(spec:property ?action-designator (:object ?object-designator))
(desig:current-designator ?object-designator ?current-object-design)
(spec:property ?current-object-design (:type ?object-type))
(spec:property ?current-object-design (:name ?object-name))
(> (spec:property ?action-designator (:arm ?arm))
(true)
(man:int:robot-free-hand ?_?arm))
(lisp-fun man:int:get-object-transform ?current-object-design ?object-transform)
(lisp-fun man:int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
(> (man:int:object-rotationally-symmetric ?object-type)
(equal ?rotationally-symmetric t)
(equal ?rotationally-symmetric nil))
(> (spec:property ?action-designator (:grasp ?grasp))
(true)
(and (lisp-fun man:int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
(member ?grasp ?grasps)))
(lisp-fun man:int:get-action-gripping-effort ?object-type ?effort)
(lisp-fun man:int:get-action-gripper-opening ?object-type ?gripper-opening)
(equal ?objects ?current-object-design)
(> (equal ?arm :left)
(and (lisp-fun man:int:get-action-trajectory :picking-up ?arm ?grasp T ?objects
?left-trajectory)
(lisp-fun man:int:get-traj-poses-by-label ?left-trajectory :reaching
?left-reach-poses)
(lisp-fun man:int:get-traj-poses-by-label ?left-trajectory :grasping
?left-grasp-poses)
(and (equal ?left-reach-poses NIL)
(equal ?left-grasp-poses NIL))
(> (equal ?arm :right)
(and (lisp-fun man:int:get-action-trajectory :picking-up ?arm ?grasp T ?objects
?right-trajectory)
(lisp-fun man:int:get-traj-poses-by-label ?right-trajectory :reaching
?right-reach-poses)
(lisp-fun man:int:get-traj-poses-by-label ?right-trajectory :grasping
?right-grasp-poses)
(and (equal ?right-reach-poses NIL)
(equal ?right-grasp-poses NIL))
(desig:designator :action ((:type :picking-up)
(:object ?current-object-design)
(:object-name ?object-name)
(:arm ?arm)
(:gripper-opening ?gripper-opening)
(:effort ?effort)
(:grasp ?grasp)
(:left-reach-poses ?left-reach-poses)
(:right-reach-poses ?right-reach-poses)
(:left-grasp-poses ?left-grasp-poses)
(:right-grasp-poses ?right-grasp-poses)
(:hold :holding))
?resolved-action-designator)))

```

1  :- (<- (desig:action-grounding ?action-designator (hold ?current-object-design ?arm  

?gripper-effort ?gripper-opening  

?left-hold-poses ?right-hold-poses))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

(spec:property ?action-designator (:type :holding))
(spec:property ?action-designator (:object ?object-designator))
(desig:current-designator ?object-designator ?current-object-design)
(spec:property ?current-object-design (:type ?object-type))
(spec:property ?current-object-design (:name ?object-name))
(> (spec:property ?action-designator (:arm ?arm))
(true)
(and (cram:robot-interfaces:robot ?robot)
(cram:robot-interfaces:arm ?robot ?arm)))
(lisp-fun obj:int:get-object-gripping-effort ?object-type ?gripper-effort)
(lisp-fun obj:int:get-object-gripper-opening ?object-type ?gripper-opening)
(lisp-fun obj:int:get-object-holding-poses
?object-name ?object-type :left ?left-hold-poses)
(lisp-fun obj:int:get-object-holding-poses
?object-name ?object-type :right ?right-hold-poses))

Figure 20: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Picking Up* (18 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
5	39	6	7

Findings:

- tba

```

1  (-> (spec:property ?action-designator (:type :holding)))
2  (spec:property ?action-designator (:object ?object-designator))
3  (desig:current-designator ?object-designator ?current-object-designator)
4  (spec:property ?current-object-designator (:type ?object-type))
5  (spec:property ?current-object-designator (:name ?object-name))
6  (-> (spec:property ?action-designator (:arm ?arm)))
7
8  (true)
9  (and (lisp-fun man-int:get-object-transform ?current-object-designator ?object-transform)
10 (lisp-fun man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face))
11 (-> (man-int:object-rotationally-symmetric ?object-type)
12 (equal ?rotationally-symmetric t)
13 (equal ?rotationally-symmetric nil))
14 (-> (spec:property ?action-designator (:grasp ?grasp)))
15 (true)
16 (and (lisp-fun man-int:get-action-grasp ?object-type ?arm ?object-transform ?grasp)
17 (member ?grasp ?grasps))
18 (lisp-fun man-int:get-action-grasping-effort ?object-type ?effort)
19 (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
20 (lisp-fun man-int:current-object-designator ?current-object-designator)
21 (-> (equal ?objects ?current-object-designator))
22 (-> (equal ?arm :left)
23 (and (lisp-fun man-int:get-action-trajectory :picking-up) ?arm ?grasp T ?objects
24 ?left-trajectory)
25 (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :reaching
26 ?left-reach-poses)
27 (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :grasping
28 ?left-grasp-poses))
29 (and (equal ?left-reach-poses NIL)
30 (equal ?left-grasp-poses NIL)))
31 (-> (equal ?arm :right)
32 (and (lisp-fun man-int:get-action-trajectory :picking-up) ?arm ?grasp T ?objects
33 ?right-trajectory)
34 (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :reaching
35 ?right-reach-poses)
36 (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :grasping
37 ?right-grasp-poses))
38 (and (equal ?right-reach-poses NIL)
39 (equal ?right-grasp-poses NIL)))
40 (desig:designator :action (:type :picking-up)
41 (object ?current-object-designator)
42 (object-name ?object-name)
43 (arm ?arm)
44 (gripper-opening ?gripper-opening)
45 (effort ?effort)
46 (grasp ?grasp)
47 (left-reach-poses ?left-reach-poses)
48 (right-reach-poses ?right-reach-poses)
49 (left-grasp-poses ?left-grasp-poses)
50 (right-grasp-poses ?right-grasp-poses)
51 (hold :holding))
52 (?resolved-action-designator)))

```



```

1  (-> (spec:property ?action-designator (:type :holding)))
2  (spec:property ?action-designator (:object ?object-designator))
3
4
5
6
7
8
9
10
11
12
13
14
15  (-> (spec:property ?action-designator (:object ?object-designator)))
16  (cpeo:object-in-hand ?object-designator ?arm))
17  (and (cram-robot-interfaces robot ?robot)
18  (cram-robot-interfaces arm ?robot ?arm))
19  (cpeo:object-in-hand ?object-designator ?arm)))
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

```

Figure 21: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Placing Down* (8 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
4	48	2	2

Findings:

- generated designator does not end by mapping variables to designator "parts"
- generated designator contains almost no functionality (no trajectory calculations, etc.)
- generation adds block about **cram-robot-interfaces** (3 lines)

```

1  <- (design-action-grounding ?action-designator (ip-plans :pick-up ?resolved-action-designator))
2   (spec:property ?action-designator (:type :holding))
3   (spec:property ?action-designator (:object ?object-designator))
4   (design-current-designator ?object-designator ?current-object-design)
5   (spec:property ?current-object-design (:type ?object-type))
6   (spec:property ?current-object-design (:name ?object-name))
7   (-> (spec:property ?action-designator (:arm ?arm)))
8   (true)
9   (-> (man-int:robot-free-hand ?_ ?arm))
10  (lisp-fun man-int:get-object-faces ?current-object-design ?object-transform)
11  (lisp-fun man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
12  (-> (man-int:object-rotationally-symmetric ?object-type))
13  (equal ?rotationally-symmetric t)
14  (equal ?rotationally-symmetric nil)
15  (-> (spec:property ?action-designator (:grasp ?grasp)))
16  (true)
17
18  (-> (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
19   (member ?grasp ?grasps)))
20  (lisp-fun man-int:calc-grasping-effort ?object-type ?effort)
21  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
22  (equal ?objects (?current-object-design))
23  (-> (equal ?arm :left))
24  (-> (and (lisp-fun man-int:get-action-trajectory :picking-up ?arm) ?grasp T ?objects
25   :left-trajectory))
26  (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :reaching)
27  (-> (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :grasping)
28  :left-grasp-poses))
29  (-> (and (equal ?left-reach-poses NIL)
30  (equal ?left-grasp-poses NIL)))
31  (-> (equal ?arm :right))
32  (-> (and (lisp-fun man-int:get-action-trajectory :picking-up ?arm) ?grasp T ?objects
33   :right-trajectory))
34  (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :reaching)
35  (-> (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :grasping)
36  :right-grasp-poses))
37  (-> (and (equal ?right-reach-poses NIL)
38  (equal ?right-grasp-poses NIL)))
39
40  (-> (design-designator :action (:type :picking-up)
41   :object ?current-object-design)
42
43   (:object-name ?object-name)
44   (:arm ?arm)
45   (:gripper-opening ?gripper-opening)
46   (:effort ?effort)
47   (:grasp ?grasp)
48   (:left-reach-poses ?left-reach-poses)
49   (:right-reach-poses ?right-reach-poses)
50   (:left-grasp-poses ?left-grasp-poses)
51   (:right-grasp-poses ?right-grasp-poses)
52   (:hold :holding))
53
54  ?resolved-action-designator))

```



```

1  <- (design-action-grounding ?action-designator (ip-plans :hold ?resolved-action-designator))
2   (spec:property ?action-designator (:type :holding))
3   (spec:property ?action-designator (:object ?object-designator))
4   (design-current-designator ?object-designator ?current-object-design)
5   (spec:property ?current-object-design (:type ?object-type))
6   (spec:property ?current-object-design (:name ?object-name))
7   (-> (spec:property ?action-designator (:arms ?arms)))
8   (true)
9   (-> (and (man-int:robot-free-hand ?_ ?arm)
10  (equal ?arms ?arm)))
11
12
13
14
15  (-> (spec:property ?action-designator (:grasp ?grasp)))
16  (true)
17  (+ (and (member ?arm ?arms)
18   (lisp-fun man-int:get-action-grasps ?object-type ?arm ?grasps)
19   :member ?grasp ?grasps)))
20  (lisp-fun man-int:calc-grasping-effort ?object-type ?effort)
21  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
22  (equal ?objects (?current-object-design))
23  (-> (member ?left ?arms))
24  (-> (and (lisp-fun man-int:get-action-trajectory :holding :left) ?grasp T ?objects
25   :left-holding-pose))
26  (lisp-fun man-int:get-traj-poses-by-label ?left-holding-pose :approach)
27
28
29
30  (-> (equal ?left-approach-poses))
31
32  (-> (member ?right ?arms))
33  (-> (and (lisp-fun man-int:get-action-trajectory :holding :right) ?grasp T ?objects
34   :right-holding-pose))
35  (lisp-fun man-int:get-traj-poses-by-label ?right-holding-pose :approach)
36
37
38  (-> (equal ?right-approach-poses))
39
40  (-> (design-designator :action (:type :holding)
41   :object ?current-object-design)
42
43   (:object-type ?object-type)
44   (:object-name ?object-name)
45   (:arms ?arms))
46
47
48  (:grasp ?grasp)
49  (:left-approach-poses ?left-approach-poses)
50  (:right-approach-poses ?right-approach-poses))
51
52
53
54  ?resolved-action-designator))

```

Figure 22: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Pouring* (39 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
4	17	20	15

Findings:

- some minor renaming (e.g. *approach* instead of *reach*)
- omits the rotational trajectory calculation
- generated designator shortens the mapping of variables to designator "parts" (misses *:arm*, *:gripper-opening*, *:effort*, *:left-grasp-poses*, *:right-grasp-poses* and *:hold* but adds *:object-type* and *:arms*)

Diagram illustrating the resolution of action-grounding for the action `:pick-up`.

Left Panel (Resolved Action Desig.):

- Initial Desig.:** `(design-action-grounding :action-designator (:plans :pick-up) ?resolved-action-designator)`
- Properties:**
 - `(spec:property :action-designator (:type :holding))`
 - `(spec:property :action-designator (:object ?object-designator))`
 - `(design:current-designator :object-designator ?current-object-desig)`
 - `(spec:property ?current-object-desig (:type ?object-type))`
 - `(spec:property ?current-object-desig (:name ?object-name))`
 - `(-> (spec:property :action-designator (:arm ?arm))
 (true)
 (man-int:robot-free-hand ?_?arm))`
 - `(lisp-fun man-int:get-object-transform ?current-object-desig ?object-transform)`
 - `(lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))`
 - `(-> (man-int:object-rotationally-symmetric ?object-type)
 (equal ?rotationally-symmetric t)
 (equal ?rotationally-symmetric nil))`
 - `(-> (spec:property :action-designator (:grasp ?grasp))
 (true))`
 - `(and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
 (member ?grasp ?grasps))`
 - `(lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)`
 - `(lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)`
 - `(equal ?objects (?current-object-designator))`
 - `(-> (equal ?arm ?efort)
 (and (lisp-fun man-int:get-action-trajectory :picking-up ?arm ?grasp T ?objects
 ?left-trajectory)
 (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :reaching
 ?left-reach-poses)
 (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :grasping
 ?left-grasp-poses)))
 (and (equal ?left-reach-poses NIL)
 (equal ?left-grasp-poses NIL)))
 (-> (equal ?arm ?right)
 (and (lisp-fun man-int:get-action-trajectory :picking-up ?arm ?grasp T ?objects
 ?right-trajectory)
 (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :reaching
 ?right-reach-poses)
 (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :grasping
 ?right-grasp-poses)))
 (and (equal ?right-reach-poses NIL)
 (equal ?right-grasp-poses NIL)))
 (design:designator :action ((:type :picking-up)
 (:object ?current-object-designator)
 (:object-name ?object-name)
 (:arm ?arm)
 (:grasp ?grasp)
 (:left-reach-poses ?left-reach-poses)
 (:right-reach-poses ?right-reach-poses)
 (:left-grasp-poses ?left-grasp-poses)
 (:right-grasp-poses ?right-grasp-poses)
 (:hold :holding))
 ?resolved-action-designator)))`

Right Panel (Resolved Action Desig.):

- Initial Desig.:** `(design-action-grounding :action-designator (:type :holding) ?resolved-action-designator)`
- Properties:**
 - `(spec:property :action-designator (:type :holding))`
 - `(spec:property :action-designator (:object ?object-designator))`
 - `(design:current-designator :object-designator ?current-object-desig)`
 - `(spec:property ?current-object-desig (:type ?object-type))`
 - `(spec:property ?current-object-desig (:name ?object-name))`
 - `(-> (spec:property :action-designator (:arm ?arm))
 (true)
 (man-int:robot-free-hand ?_?arm))`
 - `(> (spec:property :action-designator (:grasp ?grasp))
 (true))`
 - `(lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
 (member ?grasp ?grasps)`
 - `(lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)`
 - `(lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)`
 - `(equal ?objects (?current-object-designator))`
 - `(-> (lisp-fun man-int:get-action-trajectory :holding ?arm ?grasp T ?objects
 ?holding-pose))`
 - `(design:designator :action ((:type :holding)
 (:object ?current-object-designator)
 (:object-name ?object-name)
 (:arm ?arm)
 (:grasp ?grasp)
 (:holding-pose ?holding-pose))
 ?resolved-action-designator))`

Figure 23: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Slicing* (27 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
2	27	6	19

Findings:

- generated designator does not cover the necessary calculations regarding (rotational) trajectories for reaching and/or grasping
 - generated designator shortens the mapping of variables to designator "parts" (misses *:left-reach-poses*, *:right-reach-poses*, *:left-grasp-poses*, *:right-grasp-poses* and *:hold* but adds a *:holding-pose*)

```

1  | (< (desig:action-grounding ?action-designator (pp-plans:pick-up ?resolved-action-designator)))
2  |   (spec:property ?action-designator (:type :holding))
3  |   (spec:property ?action-designator (:object ?object-designator))
4  |   (spec:current-designator ?object-designator ?current-object-desig)
5  |   (spec:property ?current-object-desig (:type ?object-type))
6  |   (spec:property ?current-object-desig (:name ?object-name))
7  |   (-> (spec:property ?action-designator (:arm ?arm))
8  |         (true)
9  |         (man-int:robot-free-hand ?_ ?arm))
10 |         (lisp-fun man-int:get-object-transform ?current-object-desig ?object-transform)
11 |         (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
12 |         (-> (man-int:object-rotationally-symmetric ?object-type)
13 |             (equal ?rotationally-symmetric t)
14 |             (equal ?rotationally-symmetric nil))
15 |             (-> (spec:property ?action-designator (:grasp ?grasp))
16 |                   (true)
17 |                   (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
18 |                         (member ?grasp ?grasps)))
19 |                     (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
20 |                     (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
21 |                     (equal ?objects (?current-object-desig)))
22 |                     (-> (equal ?arm :left)
23 |                           (and (lisp-fun man-int:get-action-trajectory :picking-up) ?arm ?grasp T ?objects
24 |                               ?left-trajectory)
25 |                               (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :reaching)
26 |                               ?left-reach-poses)
27 |                               (lisp-fun man-int:get-traj-poses-by-label ?left-trajectory :grasping)
28 |                               ?left-grasp-poses)
29 |                               (and (equal ?left-reach-poses NIL)
30 |                                   (equal ?left-grasp-poses NIL)))
31 |                               (-> (equal ?arm :right)
32 |                                   (and (lisp-fun man-int:get-action-trajectory :picking-up) ?arm ?grasp T ?objects
33 |                                       ?right-trajectory)
34 |                                       (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :reaching)
35 |                                       ?right-reach-poses)
36 |                                       (lisp-fun man-int:get-traj-poses-by-label ?right-trajectory :grasping)
37 |                                       ?right-grasp-poses)
38 |                                       (and (equal ?right-reach-poses NIL)
39 |                                           (equal ?right-grasp-poses NIL)))
40 |                                       (design:designator :action ((:type :picking-up)
41 |                                         (:object ?current-object-desig)
42 |                                         (:object-name ?object-name)
43 |                                         (:arm ?arm)
44 |                                         (:gripper-opening ?gripper-opening)
45 |                                         (:effort ?effort)
46 |                                         (:grasp ?grasp)
47 |                                         (:left-reach-poses ?left-reach-poses)
48 |                                         (:right-reach-poses ?right-reach-poses)
49 |                                         (:left-grasp-poses ?left-grasp-poses)
50 |                                         (:right-grasp-poses ?right-grasp-poses)
51 |                                         (:hold :holding)))
52 |                                         ?resolved-action-designator)))

```

Figure 24: Left: Manual designator for the action *Holding* (52 lines). Right: Generated designator based on *Wiping* (6 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
3	49	2	1

Findings:

- designator definition contains five variables instead of only one
- generated designator contains no functionality (no trajectory calculations, etc.)
- generated designator does not end by mapping variables to designator "parts"

```

1  (<- (design-action-grounding ?action-designator (open-container ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-lift-poses
7    ?right-lift-poses
8    ?left-lift-poses)
9    (?left-lift-pose)
10   (?right-lift-pose)
11   (?left-2nd-lift-pose)
12   (?right-2nd-lift-pose)
13   ?joint-name ?environment-obj))
14
15  (spec:property ?action-designator (:type :opening))
16  (spec:property ?action-designator (:object ?container-designator))
17  (spec:property ?container-designator (:type ?container-type))
18  (obj:int-object-type-subtype :container ?container-type)
19  (spec:property ?container-designator (:urdf-name ?container-name))
20  (spec:property ?container-designator (:part-of ?btr-environment))
21  (-> (spec:property ?action-designator (:arm ?arm))
22    (true)
23    (and (cram-robot-interfaces:robot ?robot)
24      (cram-robot-interfaces:arm ?robot ?arm)))
25    (spec:property ?action-designator (:distance ?distance))
26    (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
27    (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
28    (lisp-fun cl-urdfname ?connecting-joint ?joint-name)
29    (btr:bullet-world ?world)
30    (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
31    (lisp-fun obj-intget-object-type-gripper-opening ?container-type ?gripper-opening)
32    (lisp-fun get-container-poses-and-transform ?container-name ?btr-environment
33      (?container-prismatic ?left-open ?container-transform ?left-poses)
34      (?container-prismatic ?right-open ?container-transform ?right-poses)
35      (?lisp-fun cram-mobile-pick-place-plans:extract-pick-up-manipulation-poses
36        ?arm ?left-poses ?right-poses
37        (?left-reach-poses ?right-reach-poses
38          ?left-grasp-poses ?right-grasp-poses
39          ?left-lift-poses ?right-lift-poses)
40        (> (lisp-pred identity ?left-lift-poses)
41          (equal) ?left-lift-poses ?left-lift-poses ?left-2nd-lift-pose)))
42        (equal) (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
43        (-> (lisp-pred identity ?right-lift-poses)
44          (equal) ?right-lift-poses ?right-lift-poses ?right-2nd-lift-pose)
45          (equal) (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose)))
46
47
1  (<- (design-action-grounding ?action-designator (open-container ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-release-poses
7    ?right-release-poses
8
9
10
11
12
13  ?joint-name
14  ?environment-obj))
15
16  (spec:property ?action-designator (:type :opening))
17  (spec:property ?action-designator (:object ?container-designator))
18  (spec:property ?container-designator (:type ?container-type))
19  (obj:int-object-type-subtype :container ?container-type)
20  (spec:property ?container-designator (:urdf-name ?container-name))
21  (spec:property ?container-designator (:part-of ?btr-environment))
22  (-> (spec:property ?action-designator (:arm ?arm))
23    (true)
24    (and (cram-robot-interfaces:robot ?robot)
25      (cram-robot-interfaces:arm ?robot ?arm)))
26    (spec:property ?action-designator (:distance ?distance))
27    (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
28    (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
29    (lisp-fun cl-urdfname ?connecting-joint ?joint-name)
29    (btr:bullet-world ?world)
30    (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
31    (lisp-fun obj-intget-object-type-gripper-opening ?container-type ?gripper-opening)
32    (lisp-fun get-container-poses-and-transform ?container-name ?btr-environment
33      (?container-prismatic ?left-open ?container-transform ?left-poses)
34      (?container-prismatic ?right-open ?container-transform ?right-poses)
35      (?lisp-fun cram-mobile-pick-place-plans:extract-release-manipulation-poses
36        ?arm ?left-poses ?right-poses
37        (?left-reach-poses ?right-reach-poses
38          ?left-grasp-poses ?right-grasp-poses
39          ?left-lift-poses ?right-lift-poses)
40        (> (lisp-pred identity ?left-lift-poses)
41          (equal) ?left-lift-poses ?left-lift-poses ?left-2nd-lift-pose)))
42        (equal) (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
43        (-> (lisp-pred identity ?right-lift-poses)
44          (equal) ?right-lift-poses ?right-lift-poses ?right-2nd-lift-pose)
45          (equal) (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose)))
46
47

```

Figure 25: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Closing* (36 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
1	11	5	30

Findings:

- ChatGPT replaces *grasping* and *picking up* with *release*, which has a different meaning
- additional line is due to a difference in formatting
- ChatGPT does not generate the necessary poses for lifting and how they are handled

```

1  | <- (design-action-grounding ?action-designator (open-container ?arm)
2  |   ?gripper-opening
3  |   ?distance
4  |   ?left-reach-poses
5  |   ?right-reach-poses
6  |   ?left-grasp-poses
7  |   ?right-grasp-poses
8  |   ?left-lift-poses
9  |   ?right-lift-poses
10 |   ?left-2nd-lift-poses
11 |   ?right-2nd-lift-poses
12 |   ?joint-name ?environment-obj)
13 |
14 |   (spec:property ?action-designator (:type :opening))
15 |   (spec:property ?action-designator ?object ?container-designator)
16 |   (spec:property ?container-designator (:type ?container-type))
17 |   (obj-int:object-type-subtype ?container ?container-type)
18 |   (spec:property ?container-designator (curdf-name ?container-name))
19 |   (spec:property ?container-designator (part-of ?btr-environment))
20 |   (> (spec:property ?action-designator (?arm ?arm))
21 |     (true)
22 |     (and (cram-robot-interfaces:robot ?robot)
23 |       (cram-robot-interfaces:arm ?robot ?arm)))
24 |
25 |   (spec:property ?action-designator (:distance ?distance))
26 |   (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
27 |   (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
28 |   (lisp-fun get-connecting-joint ?connecting-joint ?joint-name)
29 |   (bullet-world ?world)
30 |   (lisp-fun btr-object ?world ?btr-environment ?environment-obj)
31 |   (lisp-fun obj-int:get-object-type ?gripper-opening ?container-type ?gripper-opening)
32 |   (lisp-fun get-container-poses-and-transform ?container-name ?btr-environment
33 |     ?container-poses ?container-transform)
34 |   (lisp-fun obj-int:get-object-grasping-poses ?container-name
35 |     :container-prismatic ?left-open ?container-transform ?left-poses)
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |   (lisp-fun obj-int:get-object-grasping-poses ?container-name
47 |     :container-prismatic ?right-open ?container-transform ?right-poses)
48 |   (lisp-fun cram-mobile-pick-place-plans:extract-pick-up-manipulation-poses
49 |     ?arm ?left-poses ?right-poses
50 |     ?left-reach-poses ?right-reach-poses
51 |     ?left-grasp-poses ?right-grasp-poses
52 |     ?left-lift-poses ?right-lift-poses)
53 |   (> (lisp-pred identity ?left-lift-poses)
54 |     (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose))
55 |     (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
56 |
57 |
58 |
59 |
60 |
61 |
62 |
63 |
64 |   (> (lisp-pred identity ?right-lift-poses)
65 |     (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose))
66 |     (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose)))
67 |
68 |

```



```

1  | <- (design-action-grounding ?action-designator (open-container ?resolved-action-designator))
2  |
3  |
4  |
5  |
6  |
7  |
8  |
9  |
10 |
11 |
12 |
13 |
14 |   (spec:property ?action-designator (:type :opening))
15 |   (spec:property ?action-designator ?object ?object-designator)
16 |   (spec:current-designator ?object-designator ?current-object-designator)
17 |
18 |   (spec:property ?current-object-designator (:type ?object-type))
19 |   (spec:property ?current-object-designator (?name ?object-name))
20 |   (> (spec:property ?action-designator (?arm ?arm))
21 |     (true)
22 |     (and
23 |       (man-int:robot-free-hand ? ?arm)
24 |       (lisp-fun man-int:get-object-old-transformation ?current-object-designator ?object-transform)
25 |       (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
26 |       (spec:property ?action-designator (?grasp ?grasp))
27 |       (true)
28 |       (and
29 |         (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
30 |           (member ?grasp ?grasps))
31 |         (spec:property ?action-designator (?object-half-pose ?object-half-pose))
32 |         (true)
33 |         (format "Please infer where to open the container, or use the query system to infer it here")
34 |         (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
35 |         (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
36 |         (equal ?objects (?current-object-designator))
37 |         (> (equal ?arm ?arm)
38 |             (and
39 |               (lisp-fun man-int:get-action-trajectory ?opening ?arm ?grasp T ?objects
40 |                 ?left-opening-poses)
41 |               (lisp-fun man-int:get-traj-poses-by-label ?left-opening-pose :opening-up
42 |                 ?left-opening-up-poses)
43 |               (lisp-fun man-int:get-traj-poses-by-label ?left-opening-pose :opening-down
44 |                 ?left-opening-down-poses)
45 |               (and (equal ?left-opening-up-poses NIL)
46 |                 (equal ?left-opening-down-poses NIL)))
47 |             (> (equal ?arm ?right)
48 |                 (and
49 |                   (lisp-fun man-int:get-action-trajectory ?opening ?arm ?grasp T ?objects
50 |                     ?right-opening-poses)
51 |                   (lisp-fun man-int:get-traj-poses-by-label ?right-opening-pose :opening-up
52 |                     ?right-opening-up-poses)
53 |                   (lisp-fun man-int:get-traj-poses-by-label ?right-opening-pose :opening-down
54 |                     ?right-opening-down-poses)
55 |                   (and (equal ?right-opening-up-poses NIL)
56 |                         (equal ?right-opening-down-poses NIL)))
57 |             (> (design-desig-prop ?action-designator (?collision-mode ?collision-mode))
58 |                 (true)
59 |                 (equal ?collision-mode nil))
60 |             (design-designator action ((?type :opening)
61 |               (?object ?current-object-designator)
62 |               (?object-name ?object-name)
63 |               (?arm ?arm)
64 |               (?gripper-opening ?gripper-opening)
65 |               (?effort ?effort)
66 |               (?grasp ?grasp)
67 |               (?left-opening-up-poses ?left-opening-up-poses)
68 |               (?right-opening-up-poses ?right-opening-up-poses)
69 |               (?left-opening-down-poses ?left-opening-down-poses)
70 |               (?right-opening-down-poses ?right-opening-down-poses)
71 |               (?collision-mode ?collision-mode)))
72 |             ?resolved-action-designator)
73 |
74 |
75 |
76 |
77 |
78 |
79 |
80 |
81 |
82 |
83 |
84 |
85 |
86 |
87 |
88 |
89 |
90 |
91 |
92 |
93 |
94 |
95 |
96 |
97 |
98 |
99 |
100 |
101 |
102 |
103 |
104 |
105 |
106 |
107 |
108 |
109 |
110 |
111 |
112 |
113 |
114 |
115 |
116 |
117 |
118 |
119 |
120 |
121 |
122 |
123 |
124 |
125 |
126 |
127 |
128 |
129 |
130 |
131 |
132 |
133 |
134 |
135 |
136 |
137 |
138 |
139 |
140 |
141 |
142 |
143 |
144 |
145 |
146 |
147 |
148 |
149 |
150 |
151 |
152 |
153 |
154 |
155 |
156 |
157 |
158 |
159 |
160 |
161 |
162 |
163 |
164 |
165 |
166 |
167 |
168 |
169 |
170 |
171 |
172 |
173 |
174 |
175 |
176 |
177 |
178 |
179 |
180 |
181 |
182 |
183 |
184 |
185 |
186 |
187 |
188 |
189 |
190 |
191 |
192 |
193 |
194 |
195 |
196 |
197 |
198 |
199 |
200 |
201 |
202 |
203 |
204 |
205 |
206 |
207 |
208 |
209 |
210 |
211 |
212 |
213 |
214 |
215 |
216 |
217 |
218 |
219 |
220 |
221 |
222 |
223 |
224 |
225 |
226 |
227 |
228 |
229 |
230 |
231 |
232 |
233 |
234 |
235 |
236 |
237 |
238 |
239 |
240 |
241 |
242 |
243 |
244 |
245 |
246 |
247 |
248 |
249 |
250 |
251 |
252 |
253 |
254 |
255 |
256 |
257 |
258 |
259 |
259 |
260 |
261 |
262 |
263 |
264 |
265 |
266 |
267 |
268 |
269 |
270 |
271 |
272 |
273 |
274 |
275 |
276 |
277 |
278 |
279 |
279 |
280 |
281 |
282 |
283 |
284 |
285 |
286 |
287 |
288 |
289 |
289 |
290 |
291 |
292 |
293 |
294 |
295 |
296 |
297 |
298 |
299 |
299 |
300 |
301 |
302 |
303 |
304 |
305 |
306 |
307 |
308 |
309 |
309 |
310 |
311 |
312 |
313 |
314 |
315 |
316 |
317 |
318 |
319 |
319 |
320 |
321 |
322 |
323 |
324 |
325 |
326 |
327 |
328 |
329 |
329 |
330 |
331 |
332 |
333 |
334 |
335 |
336 |
337 |
338 |
339 |
339 |
340 |
341 |
342 |
343 |
344 |
345 |
346 |
347 |
348 |
349 |
349 |
350 |
351 |
352 |
353 |
354 |
355 |
356 |
357 |
358 |
359 |
359 |
360 |
361 |
362 |
363 |
364 |
365 |
366 |
367 |
368 |
369 |
369 |
370 |
371 |
372 |
373 |
374 |
375 |
376 |
377 |
378 |
379 |
379 |
380 |
381 |
382 |
383 |
384 |
385 |
386 |
387 |
388 |
389 |
389 |
390 |
391 |
392 |
393 |
394 |
395 |
396 |
397 |
398 |
399 |
399 |
400 |
401 |
402 |
403 |
404 |
405 |
406 |
407 |
408 |
409 |
409 |
410 |
411 |
412 |
413 |
414 |
415 |
416 |
417 |
418 |
419 |
419 |
420 |
421 |
422 |
423 |
424 |
425 |
426 |
427 |
428 |
429 |
429 |
430 |
431 |
432 |
433 |
434 |
435 |
436 |
437 |
438 |
439 |
439 |
440 |
441 |
442 |
443 |
444 |
445 |
446 |
447 |
448 |
449 |
449 |
450 |
451 |
452 |
453 |
454 |
455 |
456 |
457 |
458 |
459 |
459 |
460 |
461 |
462 |
463 |
464 |
465 |
466 |
467 |
468 |
469 |
469 |
470 |
471 |
472 |
473 |
474 |
475 |
476 |
477 |
478 |
479 |
479 |
480 |
481 |
482 |
483 |
484 |
485 |
486 |
487 |
488 |
489 |
489 |
490 |
491 |
492 |
493 |
494 |
495 |
496 |
497 |
498 |
499 |
499 |
500 |
501 |
502 |
503 |
504 |
505 |
506 |
507 |
508 |
509 |
509 |
510 |
511 |
512 |
513 |
514 |
515 |
516 |
517 |
518 |
519 |
519 |
520 |
521 |
522 |
523 |
524 |
525 |
526 |
527 |
528 |
529 |
529 |
530 |
531 |
532 |
533 |
534 |
535 |
536 |
537 |
538 |
539 |
539 |
540 |
541 |
542 |
543 |
544 |
545 |
546 |
547 |
548 |
549 |
549 |
550 |
551 |
552 |
553 |
554 |
555 |
556 |
557 |
558 |
559 |
559 |
560 |
561 |
562 |
563 |
564 |
565 |
566 |
567 |
568 |
569 |
569 |
570 |
571 |
572 |
573 |
574 |
575 |
576 |
577 |
578 |
579 |
579 |
580 |
581 |
582 |
583 |
584 |
585 |
586 |
587 |
588 |
589 |
589 |
590 |
591 |
592 |
593 |
594 |
595 |
596 |
597 |
598 |
599 |
599 |
600 |
601 |
602 |
603 |
604 |
605 |
606 |
607 |
608 |
609 |
609 |
610 |
611 |
612 |
613 |
614 |
615 |
616 |
617 |
618 |
619 |
619 |
620 |
621 |
622 |
623 |
624 |
625 |
626 |
627 |
628 |
629 |
629 |
630 |
631 |
632 |
633 |
634 |
635 |
636 |
637 |
638 |
639 |
639 |
640 |
641 |
642 |
643 |
644 |
645 |
646 |
647 |
648 |
649 |
649 |
650 |
651 |
652 |
653 |
654 |
655 |
656 |
657 |
658 |
659 |
659 |
660 |
661 |
662 |
663 |
664 |
665 |
666 |
667 |
668 |
669 |
669 |
670 |
671 |
672 |
673 |
674 |
675 |
676 |
677 |
678 |
679 |
679 |
680 |
681 |
682 |
683 |
684 |
685 |
686 |
687 |
688 |
689 |
689 |
690 |
691 |
692 |
693 |
694 |
695 |
696 |
697 |
698 |
699 |
699 |
700 |
701 |
702 |
703 |
704 |
705 |
706 |
707 |
708 |
709 |
709 |
710 |
711 |
712 |
713 |
714 |
715 |
716 |
717 |
718 |
719 |
719 |
720 |
721 |
722 |
723 |
724 |
725 |
726 |
727 |
728 |
729 |
729 |
730 |
731 |
732 |
733 |
734 |
735 |
736 |
737 |
738 |
739 |
739 |
740 |
741 |
742 |
743 |
744 |
745 |
746 |
747 |
748 |
749 |
749 |
750 |
751 |
752 |
753 |
754 |
755 |
756 |
757 |
758 |
759 |
759 |
760 |
761 |
762 |
763 |
764 |
765 |
766 |
767 |
768 |
769 |
769 |
770 |
771 |
772 |
773 |
774 |
775 |
776 |
777 |
778 |
779 |
779 |
780 |
781 |
782 |
783 |
784 |
785 |
786 |
787 |
788 |
789 |
789 |
790 |
791 |
792 |
793 |
794 |
795 |
796 |
797 |
798 |
799 |
799 |
800 |
801 |
802 |
803 |
804 |
805 |
806 |
807 |
808 |
809 |
809 |
810 |
811 |
812 |
813 |
814 |
815 |
816 |
817 |
818 |
819 |
819 |
820 |
821 |
822 |
823 |
824 |
825 |
826 |
827 |
828 |
829 |
829 |
830 |
831 |
832 |
833 |
834 |
835 |
836 |
837 |
838 |
839 |
839 |
840 |
841 |
842 |
843 |
844 |
845 |
846 |
847 |
848 |
849 |
849 |
850 |
851 |
852 |
853 |
854 |
855 |
856 |
857 |
858 |
859 |
859 |
860 |
861 |
862 |
863 |
864 |
865 |
866 |
867 |
868 |
869 |
869 |
870 |
871 |
872 |
873 |
874 |
875 |
876 |
877 |
878 |
879 |
879 |
880 |
881 |
882 |
883 |
884 |
885 |
886 |
887 |
888 |
889 |
889 |
890 |
891 |
892 |
893 |
894 |
895 |
896 |
897 |
898 |
899 |
899 |
900 |
901 |
902 |
903 |
904 |
905 |
906 |
907 |
908 |
909 |
909 |
910 |
911 |
912 |
913 |
914 |
915 |
916 |
917 |
918 |
919 |
919 |
920 |
921 |
922 |
923 |
924 |
925 |
926 |
927 |
928 |
929 |
929 |
930 |
931 |
932 |
933 |
934 |
935 |
936 |
937 |
938 |
939 |
939 |
940 |
941 |
942 |
943 |
944 |
945 |
946 |
947 |
948 |
949 |
949 |
950 |
951 |
952 |
953 |
954 |
955 |
956 |
957 |
958 |
959 |
959 |
960 |
961 |
962 |
963 |
964 |
965 |
966 |
967 |
968 |
969 |
969 |
970 |
971 |
972 |
973 |
974 |
975 |
976 |
977 |
978 |
979 |
979 |
980 |
981 |
982 |
983 |
984 |
985 |
986 |
987 |
988 |
989 |
989 |
990 |
991 |
992 |
993 |
994 |
995 |
995 |
996 |
997 |
998 |
999 |
999 |
1000 |
1001 |
1002 |
1003 |
1004 |
1005 |
1006 |
1007 |
1008 |
1009 |
1009 |
1010 |
1011 |
1012 |
1013 |
1014 |
1015 |
1016 |
1017 |
1018 |
1019 |
1019 |
1020 |
1021 |
1022 |
1023 |
1024 |
1025 |
1026 |
1027 |
1028 |
1029 |
1029 |
1030 |
1031 |
1032 |
1033 |
1034 |
1035 |
1036 |
1037 |
1038 |
1039 |
1039 |
1040 |
1041 |
1042 |
1043 |
1044 |
1045 |
1046 |
1047 |
1048 |
1049 |
1049 |
1050 |
1051 |
1052 |
1053 |
1054 |
1055 |
1056 |
1057 |
1058 |
1059 |
1059 |
1060 |
1061 |
1062 |
1063 |
1064 |
1065 |
1066 |
1067 |
1068 |
1069 |
1069 |
1070 |
1071 |
1072 |
1073 |
1074 |
1075 |
1076 |
1077 |
1078 |
1079 |
1079 |
1080 |
1081 |
1082 |
1083 |
1084 |
1085 |
1086 |
1087 |
1088 |
1089 |
1089 |
1090 |
1091 |
1092 |
1093 |
1094 |
1095 |
1096 |
1097 |
1097 |
1098 |
1099 |
1099 |
1100 |
1101 |
1102 |
1103 |
1104 |
1105 |
1106 |
1107 |
1108 |
1109 |
1109 |
1110 |
1111 |
1112 |
1113 |
1114 |
1115 |
1116 |
1117 |
1118 |
1119 |
1119 |
1120 |
1121 |
1122 |
1123 |
1124 |
1125 |
1126 |
1127 |
1128 |
1129 |
1129 |
1130 |
1131 |
1132 |
1133 |
1134 |
1135 |
1136 |
1137 |
1138 |
1138 |
1139 |
1140 |
1141 |
1142 |
1143 |
1144 |
1145 |
1146 |
1147 |
1148 |
1149 |
1149 |
1150 |
1151 |
1152 |
1153 |
1154 |
1155 |
1156 |
1157 |
1158 |
1159 |
1159 |
1160 |
1161 |
1162 |
1163 |
1164 |
1165 |
1166 |
1167 |
1168 |
1169 |
1169 |
1170 |
1171 |
1172 |
1173 |
1174 |
1175 |
1176 |
1177 |
1178 |
1178 |
1179 |
1180 |
1181 |
1182 |
1183 |
1184 |
1185 |
1186 |
1187 |
1187 |
1188 |
1189 |
1190 |
1191 |
1192 |
1193 |
1194 |
1195 |
1195 |
1196 |
1197 |
1198 |
1199 |
1199 |
1200 |
1201 |
1202 |
1203 |
1204 |
1205 |
1206 |
1207 |
1208 |
1209 |
1209 |
1210 |
1211 |
1212 |
1213 |
1214 |
1215 |
1216 |
1217 |
1218 |
1219 |
1219 |
1220 |
1221 |
1222 |
1223 |
1224 |
1225 |
1226 |
1227 |
1228 |
1229 |
1229 |
1230 |
1231 |
1232 |
1233 |
1234 |
1235 |
1236 |
1237 |
1238 |
1238 |
1239 |
1240 |
1241 |
1242 |
1243 |
1244 |
1245 |
1246 |
1247 |
1248 |
1249 |
1249 |
1250 |
1251 |
1252 |
1253 |
1254 |
1255 |
1256 |
1257 |
1258 |
1259 |
1259 |
1260 |
1261 |
1262 |
1263 |
1264 |
1265 |
1266 |
1267 |
1268 |
1269 |
1269 |
1270 |
1271 |
1272 |
1273 |
1274 |
1275 |
1276 |
1277 |
1278 |
1278 |
1279 |
1280 |
1281 |
1282 |
1283 |
1284 |
1285 |
1286 |
1287 |
1288 |
1288 |
1289 |
1290 |
1291 |
1292 |
1293 |
1294 |
1295 |
1296 |
1296 |
1297 |
1298 |
1299 |
1299 |
1300 |
1301 |
1302 |
1303 |
1304 |
1305 |
1306 |
1307 |
1308 |
1309 |
1309 |
1310 |
1311 |
1312 |
1313 |
1314 |
1315 |
1316 |
1317 |
1318 |
1319 |
1319 |
1320 |
1321 |
1322 |
1323 |
1324 |
1325 |
1326 |
1327 |
1328 |
1329 |
1329 |
1330 |
1331 |
1332 |
1333 |
1334 |
1335 |
1336 |
1337 |
1338 |
1338 |
1339 |
1340 |
1341 |
1342 |
1343 |
1344 |
1345 |
1346 |
1347 |
1348 |
1349 |
1349 |
1350 |
1351 |
1352 |
1353 |
1354 |
1355 |
1356 |
1357 |
1358 |
1359 |
1359 |
1360 |
1361 |
1362 |
1363 |
1364 |
1365 |
1366 |
1367 |
1368 |
1369 |
1369 |
1370 |
1371 |
1372 |
1373 |
1374 |
1375 |
1376 |
1377 |
1378 |
1379 |
1379 |
1380 |
1381 |
1382 |
1383 |
1384 |
1385 |
1386 |
1387 |
1388 |
1388 |
1389 |
1390 |
1391 |
1392 |
1393 |
1394 |
1395 |
1396 |
1396 |
1397 |
1398 |
1399 |
1399 |
1400 |
1401 |
1402 |
1403 |
1404 |
1405 |
1406 |
1407 |
1408 |
1409 |
1409 |
1410 |
1411 |
1412 |
1413 |
1414 |
1415 |
1416 |
1417 |
1418 |
1419 |
1419 |
1420 |
1421 |
1422 |
1423 |
1424 |
1425 |
1426 |
1427 |
1428 |
1429 |
1429 |
1430 |
1431 |
1432 |
1433 |
1434 |
1435 |
1436 |
1437 |
1438 |
1439 |
1439 |
1440 |
1441 |
1442 |
1443 |
1444 |
1445 |
1446 |
1447 |
1448 |
1449 |
1449 |
1450 |
1451 |
1452 |
1453 |
1454 |
1455 |
1456 |
1457 |
1458 |
1459 |
1459 |
1460 |
1461 |
1462 |
1463 |
1464 |
1465 |
1466 |
1467 |
1468 |
1469 |
1469 |
1470 |
1471 |
1472 |
1473 |
1474 |
1475 |
1476 |
1477 |
1478 |
1478 |
1479 |
1480 |
1481 |
1482 |
1483 |
1484 |
1485 |
1486 |
1487 |
1488 |
1489 |
1489 |
1490 |
1491 |
1492 |
1493 |
1494 |
1495 |
1496 |
1497 |
1498 |
1499 |
1499 |
1500 |
1501 |
1502 |
1503 |
1504 |
1505 |
1506 |
1507 |
1508 |
1509 |
1509 |
1510 |
1511 |
1512 |
1513 |
1514 |
1515 |
1516 |
1517 |
1518 |
1519 |
1519 |
1520 |
1521 |
1522 |
1523 |
1524 |
1525 |
1526 |
1527 |
1528 |
1529 |
1529 |
1530 |
1531 |
1532 |
1533 |
1534 |
1535 |
1536 |
1537 |
1538 |
1539 |
1539 |
1540 |
1541 |
1542 |
1543 |
1544 |
1545 |
1546 |
1547 |
1548 |
1549 |
1549 |
1550 |
1551 |
1552 |
1553 |
1554 |
1555 |
1556 |
1557 |
1558 |
1559 |
1559 |
1560 |
1561 |
1562 |
1563 |
1564 |
1565 |
1566 |
1567 |
1568 |
1569 |
1569 |
1570 |
1571 |
1572 |
1573 |
1574 |
1575 |
1576 |
1577 |
1578 |
1579 |
1579 |
1580 |
1581 |
1582 |
1583 |
1584 |
1585 |
1586 |
1587 |
1588 |
1589 |
1589 |
1590 |
1591 |
1592 |
1593 |
1594 |
1595 |
1596 |
1597 |
1598 |
1599 |
1599 |
1600 |
1601 |
1602 |
1603 |
1604 |
1605 |
1606 |
1607 |
1608 |
1609 |
1609 |
1610 |
1611 |
1612 |
1613 |
1614 |
1615 |
1616 |
1617 |
1618 |
1619 |
1619 |
1620 |
1621 |
1622 |
1623 |
1624 |
1625 |
1626 |
1627 |
1628 |
1629 |
1629 |
1630 |
1631 |
1632 |
1633 |
1634 |
1635 |
1636 |
1637 |
1638 |
1639 |
1639 |
1640 |
1641 |
1642 |
1643 |
1644 |
1645 |
1646 |
1647 |
1648 |
1649 |
1649 |
1650 |
1651 |
1652 |
1653 |
1654 |
1655 |
1656 |
1657 |
1658 |
1659 |
1659 |
1660 |
1661 |
1662 |
1663 |
1664 |
1665 |
1666 |
1667 |
1668 |
1669 |
1669 |
1670 |
1671 |
1672 |
1673 |
1674 |
1675 |
1676 |
1677 |
1678 |
1679 |
1679 |
1680 |
1681 |
1682 |
1683 |
1684 |
1685 |
1686 |
1687 |
1688 |
1689 |
1689 |
1690 |
1691 |
1692 |
1693 |
1694 |
1695 |
1696 |
1697 |
1698 |
1699 |
1699 |
1700 |
1701 |
1702 |
1703 |
1704 |
1705 |
1706 |
1707 |
1708 |
1709 |
1709 |
1710 |
1711 |
1712 |
1713 |
1714 |
1715 |
1716 |
1717 |
1718 |
1719 |
1719 |
1720 |
1721 |
1722 |
1723 |
1724 |
1725 |
1726 |
1727 |
1728 |
1729 |
1729 |
1730 |
1731 |
1732 |
1733 |
1734 |
1735 |
1736 |
1737 |
1738 |
1739 |
1739 |
1740 |
1741 |
1742 |
1743 |
1744 |
1745 |
1746 |
1747 |
1748 |
1749 |
1749 |
1750 |
1751 |
1752 |
1753 |
1754 |
1755 |
1756 |
1757 |
1758 |
1759 |
1759 |
1760 |
1761 |
1762 |
1763 |
1764 |
1765 |
1766 |
1767 |
1768 |
1769 |
1769 |
1770 |
1771 |
1772 |
1773 |
1774 |
1775 |
1776 |
1777 |
1778 |
1779 |
1779 |
1780 |
1781 |
1782 |
1783 |
1784 |
1785 |
1786 |
1787 |
1788 |
1789 |
1789 |
1790 |
1791 |
1792 |
1793 |
1794 |
1795 |
1796 |
1797 |
1798 |
1799 |
1799 |
1800 |
1801 |
1802 |
1803 |
1804 |
1805 |
1806 |
1807 |
1808 |
1809 |
1809 |
1810 |
1811 |
1812 |
1813 |
1814 |
1815 |
1816 |
1817 |
1818 |
1819 |
1819 |
1820 |
1821 |
1822 |
1823 |
1824 |
1825 |
1826 |
1827 |
1828 |
1829 |
1829 |
1830 |
1831 |
1832 |
1833 |
1834 |
1835 |
1836 |
1837 |
1838 |
1839 |
1839 |
1840 |
1841 |
1842 |
1843 |
1844 |
1845 |
1846 |
1847 |
1848 |
1849 |
1849 |
1850 |
1851 |
1852 |
1853 |
1854 |
1855 |
1856 |
1857 |
1858 |
1859 |
1859 |
1860 |
1861 |
1862 |
1863 |
1864 |
1865 |
1866 |
1867 |
1868 |
1869 |
1869 |
1870 |
1871 |
1872 |
1873 |
1874 |
1875 |
1876 |
1877 |
1878 |
1878 |
1879 |
1880 |
1881 |
1882 |
1883 |
1884 |
1885 |
1886 |
1887 |
1888 |
1889 |
1889 |
1890 |
1891 |
1892 |
1893 |
1894 |
1895 |
1896 |
1897 |
1898 |
1899 |
1899 |
1900 |
1901 |
1902 |
1903 |
1904 |
1905 |
1906 |
1907 |
1908 |
1909 |
1909 |
1910 |
1911 |
1912 |
1913 |
1914 |
1915 |
1916 |
1917 |
1918 |
1919 |
1919 |
1920 |
1921 |
1922 |
1923 |
1924 |
1925 |
1926 |
1927 |
1928 |
1929 |
1929 |
1930 |
1931 |
1932 |
1933 |
1934 |
1935 |
1936 |
1937 |
1938 |
1939 |
1939 |
1940 |
1941 |
1942 |
1943 |
1944 |
1945 |
1946 |
1947 |
1948 |
1949 |
1949 |
1950 |
1951 |
1952 |
1953 |
1954 |
1955 |
1956 |
1957 |
1958 |
1959 |
1959 |
1960 |
1961 |
1962 |
1963 |
1964 |
1965 |
1966 |
1967 |
1968 |
1969 |
1969 |
1970 |
1971 |
1972 |
1973 |
1974 |
1975 |
1976 |
1977 |
1978 |
1978 |
1979 |
1980 |
1981
```

```

1  #:-(<- (design-action-grounding ?action-designator (open-container ?arm
2    ?upper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    ?left-1st-lift-poses
9    ?right-1st-lift-poses
10   ?left-2nd-lift-poses
11   ?right-2nd-lift-poses
12   ?joint-name ?environment-obj))
13   (spec-property ?action-designator (:type :opening))
14   (spec-property ?action-designator (:object ?container-designator))
15   (spec-property ?container-designator (:type ?container-type))
16   (obj-int-object-type-subtype :container ?container-type)
17   (spec-property ?container-designator (:url-name ?container-name))
18   (spec-property ?container-designator (:part-of ?btb-environment))
19
20  (> (spec-property ?action-designator (arm ?arm)
21    (true)
22    (and (cram-robot-interfaces:robot ?robot)
23      (cram-robot-interface:arm ?robot ?arm)))
24    (spec-property ?action-designator (?distance ?distance)
25      (spec-property ?distance (?bt-environment ?container-link)
26        (isp-fun get-connecting-joint ?container-link ?connecting-joint)
27        (isp-fun cl-urdf:name ?connecting-joint ?joint-name)
28        (btr:btr-world ?world)
29        (isp-fun btr:object ?world ?bt-environment ?environment-obj)
30
31    (lisp-fun obj-int:get-object-type? :gripper-opening ?container-type ?gripper-opening)
32    (lisp-fun get-container-pose-and-transform ?container-name ?bt-environment
33      (?container-pose ?container-transform))
34
35    (lisp-fun obj-int:get-object-grasping-poses ?container-name
36      :container-prismatic-left :open ?container-transform ?left-poses)
37    (lisp-fun obj-int:get-object-grasping-poses ?container-name
38      :container-prismatic-right :open ?container-transform ?right-poses)
39    (lisp-fun cram-mobile-pick-place-plans:extract-pick-up-manipulation-poses
40      ?arm ?left-poses ?right-poses
41      (?left-reach-poses ?right-reach-poses
42        ?left-grasp-poses ?right-grasp-poses
43
44        ?left-lift-poses ?right-lift-poses))
45    (> (isp-pred identity ?left-lift-poses
46      (equal ?left-lift-poses ?left-1st-lift-poses ?left-2nd-lift-poses)
47      (equal (nil nil) ?left-lift-poses ?left-2nd-lift-poses)))
48    (> (isp-pred identity ?right-lift-poses
49      (equal ?right-lift-poses ?right-lift-poses ?right-2nd-lift-poses)
50      (equal (nil nil) ?right-lift-poses ?right-2nd-lift-poses)))
51
52
53
54
55
56
57
58
59

```



```

1  #:-(<- (design-action-grounding ?action-designator (sp-plans:open-container ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

Figure 27: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Holding* (47 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
36	35	9	2

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *object* instead of *container*)
- designator calculates trajectories and poses in a different manner
- designator (falsely) ends by mapping the variables to designator "parts"

```

1  (-> (design:action-grounding ?action-designator (open ?container ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    (?left-lift-pose)
9    (?right-lift-pose)
10   (?left-2nd-lift-pose)
11   (?right-2nd-lift-pose)
12   ?joint-name ?environment-obj))
13   (spec:property ?action-designator (:type opening))
14   (spec:property ?action-designator (:object ?container-desig))
15   (spec:property ?container-designator (:type ?container-type))
16   (obj-int:object-type-subtype .container ?container-type)
17   (spec:property ?contained-designator (curld-name ?container-name))
18   (spec:property ?contained-designator (part-of ?btr-environment))
19   (> (spec:property ?action-designator (:arm ?arm))
20     (true)
21     (and (cram-robot-interfaces:robot ?robot)
22       (cram-robot-interfaces:arm ?arm ?am)))
23   (spec:property ?action-designator (:distance ?distance))
24   (lisp-fun obj-int:container-link ?container-name ?btr-environment ?container-link)
25   (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
26   (lisp-fun cl-urdf:name ?connecting-joint ?joint-name)
27   (btr:bullet-world ?world)
28   (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
29   (lisp-fun obj-int:get-object-type:gripper-opening ?container-type ?gripper-opening)
30   (lisp-fun get-container-pose-and-transform ?container-name ?btr-environment
31     (?Container-pose ?Container-transform))
32   (lisp-fun obj-int:get-object-grasping-poses ?Container-name
33     ?Container-prismatic:left :open ?Container-transform ?left-poses)
34   (lisp-fun obj-int:get-object-grasping-poses ?Container-name
35     ?Container-prismatic:right :open ?Container-transform ?right-poses)
36   (lisp-fun cram:mobile:pick-place-plans:extract:pick-up-manipulation-poses
37     ?arm ?left-poses ?right-poses
38     (?left-reach-poses ?right-reach-poses
39       ?left-grasp-poses ?right-grasp-poses
40       ?left-lift-poses ?right-lift-poses))
41   (> (lisp-pred identity ?left-lift-poses)
42     (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose))
43     (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
44   (> (lisp-pred identity ?right-lift-poses)
45     (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose))
46     (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose))))

```



```

1  (-> (design:action-grounding ?action-designator (open ?container-desig ?arm ?effort))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

Figure 28: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Picking Up* (9 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
0	37	4	5

Findings:

- tba

```

1  | (spec:property ?action-designator (:type :opening))
2  |   ?gripper-opening
3  |   ?distance
4  |   ?left-reach-poses
5  |   ?right-reach-poses
6  |   ?left-grasp-poses
7  |   ?right-grasp-poses
8  |   (?left-lift-poses)
9  |   (?right-lift-poses)
10 |   (?left-2nd-lift-poses)
11 |   (?right-2nd-lift-poses)
12 |   (?left-name ?environment-obj)
13 | 
14 |   (spec:property ?action-designator (:type :opening))
15 |   (spec:property ?action-designator (:object ?container-designator))
16 |   (spec:property ?container-designator (:type ?container-type))
17 |   (obj-int-object-type-subtype :container ?container-type)
18 |   (spec:property ?container-designator (:part-of ?btr-environment))
19 |   (> (spec:property ?action-designator (:arm ?arm))
20 |     (true)
21 |     (and (cram-robot-interface :robot ?robot)
22 |       (cram-robot-interface :arm ?robot ?arm)))
23 |   (spec:property ?action-designator (:distance ?distance))
24 |   (lisp:fun get-container-link ?container-name ?distance ?container-link)
25 |   (lisp:fun get-connecting-joint ?container-link ?connecting-joint)
26 |   (lisp:fun cl-urdf-name ?connecting-joint ?joint-name)
27 |   (lisp:fun btr-object-world ?btr-environment ?environment-obj)
28 |   (lisp:fun obj-int-get-object-type ?gripper-opening ?gripper-opening)
29 |   (lisp:fun obj-int-get-container-opening-poses ?container-name ?opening-poses)
30 |   (lisp:fun get-container-pose-and-transform ?container-name ?btr-environment)
31 |   (lisp:fun container-pose ?container-name ?transform)
32 |   (lisp:fun obj-int-get-prismatic-poses ?container-name)
33 |   (lisp:fun obj-int-get-prismatic-poses ?container-name ?left-poses)
34 |   (lisp:fun obj-int-get-object-grasping-poses ?container-name)
35 |   (lisp:fun container-prismatic :left :open ?open ?container-transform ?right-poses)
36 |   (lisp:fun cram-mobile-pick-place-plans extract-pick-up-manipulation-poses
37 |     ?arm ?left-poses)
38 |   (lisp:fun cram-mobile-pick-place-plans ?left-reach-poses)
39 |   (lisp:fun ?left-reach-poses ?right-reach-poses)
40 |   (lisp:fun ?left-reach-poses ?right-reach-poses ?left-lift-poses)
41 |   (> (lisp:pred identity ?left-lift-poses)
42 |     (equal ?left-lift-poses ?right-lift-poses ?left-2nd-lift-poses))
43 |   (lisp:pred nil ?left-lift-poses ?right-lift-poses)
44 |   (> (lisp:pred identity ?right-lift-poses)
45 |     (equal ?right-lift-poses ?right-lift-poses ?right-2nd-lift-poses))
46 |   (equal ?NIL ?NIL) (equal ?right-lift-poses ?right-2nd-lift-poses)))

```



```

1  | (spec:property ?action-designator (:type :opening))
2  |   (spec:property ?action-designator (:object ?container-designator))
3  |   (spec:property ?container-designator (:type ?container))
4  | 
5  |   (spec:property ?container-designator (:name ?container-name))
6  | 
7  | 
8  | 
9  | 
10 | 
11 | 
12 | 
13 | 
14 | 
15 | 
16 | 
17 | 
18 | 
19 | 
20 | 
21 | 
22 | 
23 | 
24 | 
25 | 
26 | 
27 | 
28 | 
29 | 
30 | 
31 | 
32 | 
33 | 
34 | 
35 | 
36 | 
37 | 
38 | 
39 | 
40 | 
41 | 
42 | 
43 | 
44 | 
45 | 
46 |

```

Figure 29: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Placing Down* (12 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
5	39	5	2

Findings:

- designator definition contains only one variable instead of 13
- designator contains only definitions, no assignments or calculations

```

1  #:-( desig:action-grounding ?action-designator (open-container ?arm
2    ?upper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    (?left-lift-pose)
8    (?right-lift-pose)
9    (?left-2nd-lift-pose)
10   (?right-2nd-lift-pose)
11   ?joint-name ?environment-object))
12
13   (spec:property ?action-designator (:type :opening))
14   (spec:property ?action-designator (:object ?container-designator))
15   (spec:property ?container-designator (:type ?container-type))
16   (obj-int:object-type-subtype ?container ?container-type)
17
18   (spec:property ?container-designator (:uri-id ?name ?container-name))
19
20   (spec:property ?container-designator (:part-of ?sit-environment))
21
22
23
24  (-> (spec:property ?action-designator (:arm ?arm)))
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```



```

1  #:-( desig:action-grounding ?action-designator (open-container ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

Figure 30: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Pouring* (45 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
35	36	8	2

Findings:

- designator definition contains only one variable instead of 13
- some minor renaming (e.g. *object* instead of *container*)
- designator calculates trajectories and poses in a different manner
- designator (falsely) ends by mapping the variables to designator "parts"

```
1  :- (<- (design-action-grounding ?action-designator (open-container ?arm
2    ?gripper-opening
3    ?distane
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    ?left-lift-poses)
9    ?left-lift-poses)
10   ?left-2nd-lift-poses)
11   ?right-2nd-lift-poses)
12   ?joint-name ?environment-obj))
13   (spec-property ?action-designator :type :opening)
14   (spec-property ?action-designator :object ?container-designator)
15   (spec-property ?container-designator :type :container-type)
16   (obj-int-object-type-subtype :container ?container-type)
17   (spec-property ?container-designator :urdf-name ?container-name)
18   (spec-property ?container-designator :part-of ?btr-environment))
19   (-> (spec-property ?action-designator :arm ?arm))
20   (true)
21   (and
22     (cram-robot-interfaces robot ?robot)
23     (cram-robot-interfaces arm ?robot ?arm)))
24   (spec-property ?action-designator :distance ?distance))
25   (lisp-fun get-container-link ?container-name ?btr-environment ?container-link)
26   (lisp-fun get-connecting-joint ?container-link ?connecting-joint)
27   (lisp-fun cl-urdf.name ?connecting-joint ?joint-name)
28   (btr-bullet-world ?world)
29
30   (lisp-fun btr-object ?world ?btr-environment ?environment-obj)
31
32   (lisp-fun obj-int-get-object-type-gripper-opening ?container-type ?gripper-opening)
33   (lisp-fun obj-int-get-container-object-transform ?container-name ?btr-environment
34   ?container-type ?container-transform))
35   (lisp-fun obj-int-get-object-grasping-poses ?container-name
36   :container-prismatic :left :open ?container-transform ?left-poses)
37
38   (lisp-fun obj-int-get-object-grasping-poses ?container-name
39   :container-prismatic :right :open ?container-transform ?right-poses)
40   (lisp-fun cram-mobile-pick-place-plans :extract-pick-up-manipulation-poses
41   ?arm ?left-poses ?right-poses
42   (?left-reach-poses ?right-reach-poses
43   ?left-grasp-poses ?right-grasp-poses
44
45   ?left-poses ?right-lift-poses))
46
47   (-> (lisp-pred identity ?left-lift-poses)
48   (equal ?left-lift-poses (?left-lift-poses ?left-2nd-lift-poses))
49   (equal (NIL NIL) (?left-lift-poses ?left-2nd-lift-poses)))
50
51   (-> (lisp-pred identity ?right-lift-poses)
52   (equal ?right-lift-poses (?right-lift-poses ?right-2nd-lift-poses))
53   (equal (NIL NIL) (?right-lift-poses ?right-2nd-lift-poses))))
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
567
568
569
569
570
571
572
573
574
575
576
577
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
626
627
628
628
629
629
630
631
632
633
634
635
635
636
637
637
638
638
639
639
640
641
642
643
644
645
645
646
647
647
648
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
726
727
728
728
729
729
730
731
732
733
734
735
735
736
737
737
738
738
739
739
740
741
742
743
744
745
745
746
747
747
748
748
749
749
750
751
752
753
754
755
756
757
757
758
759
759
760
761
762
763
764
765
766
766
767
768
768
769
769
770
771
772
773
774
775
775
776
777
777
778
778
779
779
780
781
782
783
784
785
785
786
787
787
788
788
789
789
790
791
792
793
794
795
795
796
797
797
798
798
799
799
800
801
802
803
804
805
805
806
807
807
808
808
809
809
810
811
812
813
814
815
815
816
817
817
818
818
819
819
820
821
822
823
824
825
825
826
827
827
828
828
829
829
830
831
832
833
834
835
835
836
837
837
838
838
839
839
840
841
842
843
844
845
845
846
847
847
848
848
849
849
850
851
852
853
854
855
855
856
857
857
858
858
859
859
860
861
862
863
864
865
865
866
867
867
868
868
869
869
870
871
872
873
874
875
875
876
877
877
878
878
879
879
880
881
882
883
884
885
885
886
887
887
888
888
889
889
890
891
892
893
894
894
895
896
896
897
897
898
898
899
899
900
901
902
903
904
904
905
906
906
907
907
908
908
909
909
910
911
912
913
914
914
915
916
916
917
917
918
918
919
919
920
921
922
923
924
925
925
926
927
927
928
928
929
929
930
931
932
933
934
935
935
936
937
937
938
938
939
939
940
941
942
943
944
944
945
946
946
947
947
948
948
949
949
950
951
952
953
954
955
955
956
957
957
958
958
959
959
960
961
962
963
964
964
965
966
966
967
967
968
968
969
969
970
971
972
973
974
974
975
976
976
977
977
978
978
979
979
980
981
982
983
984
984
985
986
986
987
987
988
988
989
989
990
991
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1710
1711
1711
1712
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720

```

Figure 31: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Slicing* (47 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
36	35	8	3

Findings:

- designator definition contains only one variable instead of 13
 - some minor renaming (e.g. *object* instead of *container*)
 - designator calculates trajectories and poses in a different manner
 - designator (falsely) ends by mapping the variables to designator "parts"

```

1  <- (desig.action-grounding ?action-designator (open-container ?arm
2    ?gripper-opening
3    ?distance
4    ?left-reach-poses
5    ?right-reach-poses
6    ?left-grasp-poses
7    ?right-grasp-poses
8    ?left-lift-poses
9    ?right-lift-poses
10   ?left-2nd-lift-poses
11   ?right-2nd-lift-poses
12   ?joint-name ?environment-obj)
13   (spec:property ?action-designator (:type :opening))
14   (spec:property ?action-designator (:object ?container-designator))
15   (spec:property ?container-designator (:type ?container-type))
16   (obj-int:object-type-subtype ?container ?container-type)
17   (spec:property ?container-designator (:urdf-name ?container-name))
18   (spec:property ?container-designator (:part-of ?bitr-environment))
19   (-> (spec:property ?action-designator (:arm ?arm))
20     (true)
21     (and
22       (cram-robot-interfaces:robot ?robot)
23       (cram-robot-interfaces:arm ?robot ?arm)))
24   (spec:property ?action-designator (:distance ?distance))
25   (lisp-fun get-contain-link ?container-name ?bitr-environment ?container-link)
26   (lisp-fun get-connecting-joint ?container-link :connecting-joint)
27   (lisp-fun get-joint ?joint-name ?joint)
28   (lisp-fun bullet-world ?world)
29   (lisp-fun bitr:object ?world ?bitr-environment ?environment-obj)
30   (lisp-fun obj-int:get-object-type-gripper-opening ?container-type ?gripper-opening)
31   (lisp-fun get-container-pose-and-transform ?container-name ?bitr-environment
32     ?container-pose ?container-transform)
33   (lisp-fun obj-int:get-object-grasping-poses ?container-name
34     :container-prismatic ?left :open ?container-transform ?left-poses)
35   (lisp-fun obj-int:get-object-prismatic-robot :open ?container-transform ?right-poses)
36   (lisp-fun cram-motor-pick-place-poses :extract-pick-up-manipulation-poses
37     ?arm ?left-poses ?right-poses
38     ?left-reach-poses ?right-reach-poses
39     ?left-grasp-poses ?right-grasp-poses
40     ?left-lift-poses ?right-lift-poses)
41   (-> (lisp-pred identity ?left-lift-poses)
42     (equal ?left-lift-poses (?left-lift-poses ?left-2nd-lift-poses))
43     (equal (nil nil) (?left-lift-poses ?left-2nd-lift-poses)))
44   (-> (lisp-pred identity ?right-lift-poses)
45     (equal ?right-lift-poses (?right-lift-poses ?right-2nd-lift-poses))
46     (equal (nil nil) (?right-lift-poses ?right-2nd-lift-poses)))

```



```

1  <- (desig.action-grounding ?action-designator (open-container ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13   (spec:property ?action-designator (:type :opening-container))
14   (spec:property ?action-designator (:grasp ?grasp))
15   (spec:property ?action-designator (:arm ?arm))
16   (desig.current-designator ?container-designator ?current-container-designator)
17
18   (spec:property ?current-container-designator (:type ?container-type))
19   (-> (spec:property ?current-container-designator
20     (lisp-fun get-trajectory :opening-container ?arm ?grasp ?current-container-designator ?lists)
21     (lisp-fun man-int:get-traj-poses-by-label ?lists :opening-container ?poses))
22   (design designator :action (:type :opening-container)
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37   (spec:property ?arm ?arm)
38
39   (spec:property ?grasp ?grasp)
40   (spec:property ?poses ?poses)
41   (spec:property ?resolved-action-designator))
42
43
44
45
46

```

Figure 32: Left: Manual designator for the action *Opening* (46 lines). Right: Generated designator based on *Wiping* (14 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
6	38	8	0

Findings:

- designator definition contains only one variable instead of 13
- designator contains only definitions, no assignments or calculations
- designator (falsely) ends by mapping the variables to designator "parts"

```

1  <- (design.action-grounding ?action-designator (pick-up ?current-object-desig ?arm
2   ?gripper-opening ?effort ?grasp
3
4
5
6   ?left-reach-poses ?right-reach-poses
7
8   ?left-grasp-poses ?right-grasp-poses
9   ?left-lift-poses ?right-lift-poses)))
10
11
12
13 (spec:property ?action-designator (type :picking-up))
14 (spec:property ?action-designator (object ?object-designator))
15 (design:current-designator ?object-designator ?current-object-designator)
16 (spec:property ?current-object-designator (type ?object-type))
17 (spec:property ?current-object-designator (name ?object-name))
18 (> (spec:property ?action-designator (arm ?arm))
19   (true)
20   (and (cram-robot-interfaces:robot ?robot)
21         (cram-robot-interfaces:arm ?robot ?arm)))
22   (lisp-fun obj-int:get-object-transform ?current-object-designator ?object-transform)
23   (lisp-fun obj-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
24   (-> (obj-int:object-rotationally-symmetric ?object-type)
25     (equal ?rotationally-symmetric t)
26     (equal ?rotationally-symmetric nil))
27   (spec:property ?action-designator (grasp ?grasp))
28   (true)
29   (and (lisp-fun obj-int:get-object-type-grasps
30         ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
31         ?grasps)
32     (member ?grasp ?grasps)))
33   (lisp-fun obj-int:get-object-type-gripping-effort ?object-type ?effort)
34   (lisp-fun obj-int:get-object-type-gripper-opening ?object-type ?gripper-opening)
35
36   (lisp-fun obj-int:get-object-grasping-poses
37     ?object-name ?object-type ?left ?grasp ?object-transform
38     ?left-poses)
39   (lisp-fun obj-int:get-object-grasping-poses
40     ?object-name ?object-type ?right ?grasp ?object-transform
41     ?right-poses)
42   (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses
43
44   (left-reach-poses ?right-reach-poses
45   ?left-grasp-poses ?right-grasp-poses
46   ?left-lift-poses ?right-lift-poses)))
47
48
49
50
51
52

```



```

1  <- (design.action-grounding ?action-designator (pick-up ?arm
2   ?gripper-opening
3   ?distance
4   ?object-designator
5   ?left-reach-poses
6   ?right-reach-poses
7   ?left-grasp-poses
8   ?right-grasp-poses
9   ?left-lift-poses
10  ?right-lift-poses
11  ?left-2nd-lift-poses
12  ?right-2nd-lift-poses)))
13 (spec:property ?action-designator (type :picking))
14 (spec:property ?action-designator (object ?object-designator))
15 (spec:property ?object-designator (type ?object-type))
16 (spec:property ?object-type-subtype ?object-type)
17 (spec:property ?object-designator (curlf-name ?object-name))
18 (> (spec:property ?action-designator (arm ?arm))
19   (true)
20   (and (cram-robot-interfaces:robot ?robot)
21         (cram-robot-interfaces:arm ?robot ?arm)))
22
23
24
25
26
27 (spec:property ?action-designator (distance ?distance))
28
29
30
31
32
33 (lisp-fun obj-int:get-object-type-gripper-opening ?object-type ?gripper-opening)
34 (lisp-fun obj-int:get-object-pose-and-transform ?object-name ?object-transform)
35 (lisp-fun obj-int:get-object-grasping-poses ?object-name
36   ?object-prismatic:left ?close ?object-transform ?left-poses)
37
38 (lisp-fun obj-int:get-object-grasping-poses ?object-name
39   ?object-prismatic:right ?close ?object-transform ?right-poses)
40
41 (lisp-fun cram-mobile:pick-place-plans: extract-pick-up-manipulation-poses
42   ?arm ?left-poses ?right-poses
43   ?left-reach-poses ?right-reach-poses
44   ?left-grasp-poses ?right-grasp-poses
45   ?left-lift-poses ?right-lift-poses)
46
47 (> (lisp-pred identity ?left-lift-poses)
48   (equal ?left-lift-poses (?left-lift-poses ?left-2nd-lift-poses)))
49   (equal (NIL NIL) (?left-lift-poses ?left-2nd-lift-poses)))
50
51 (> (lisp-pred identity ?right-lift-poses)
52   (equal ?right-lift-poses (?right-lift-poses ?right-2nd-lift-poses)))
53   (equal (NIL NIL) (?right-lift-poses ?right-2nd-lift-poses)))

```

Figure 33: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Closing* (39 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
16	14	15	8

Findings:

- tba

```

1  |<- (desig:action-grounding ?action-designator [pick-up ?current-object-desig ?arm
2  |  | 2gripper-opening ?effort ?grasp
3  |  |  ?left-reach-poses ?right-reach-poses
4  |  |  ?left-grasp-poses ?right-grasp-poses
5  |  |  ?left-lift-poses ?right-lift-poses)
6  |
7  |  | (spec:property ?action-designator (:type :picking-up))
8  |  | (spec:property ?action-designator (?object ?object-designator))
9  |  | (design:current-designator ?object-designator ?current-object-desig)
10 |  | (spec:property ?current-object-desig (:type ?object-type))
11 |  | (spec:property ?current-object-desig (:name ?object-name))
12 |  | (> (spec:property ?action-designator (:arm ?arm))
13 |  | (true)
14 |  | (and (cram-robot-interfaces:robot ?robot)
15 |  | (cram-robot-interfaces:arm ?robot ?arm))
16 |  | (lisp-fun obj-int:get-object-transform ?current-object-desig ?object-transform)
17 |  | (lisp-fun obj-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
18 |  | (> (obj-int:object-rotationally-symmetric ?object-type)
19 |  | (equal ?rotationally-symmetric t)
20 |  | (equal ?rotationally-symmetric nil))
21 |  | (> (spec:property ?action-designator (:grasp ?grasp))
22 |  | (true)
23 |  | (and (lisp-fun obj-int:get-object-type-grasps
24 |  | ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
25 |  | ?grasps)
26 |  | (member ?grasp ?grasps))
27 |  | (lisp-fun obj-int:get-object-type-gripping-effort ?object-type ?effort)
28 |  | (lisp-fun obj-int:get-object-type-gripper-opening ?object-type ?gripper-opening)
29 |  | (lisp-fun obj-int:get-object-grasping-poses
30 |  | ?object-name ?object-type :left ?grasp ?object-transform
31 |  | ?left-poses)
32 |  | (lisp-fun obj-int:get-object-grasping-poses
33 |  | ?object-name ?object-type :right ?grasp ?object-transform
34 |  | ?right-poses)
35 |  | (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses
36 |  | ?left-reach-poses ?right-reach-poses
37 |  | ?left-grasp-poses ?right-grasp-poses
38 |  | ?left-lift-poses ?right-lift-poses)))
39 |
40 |
41 |
42 |
43 |
44 |
45 |

```



```

1  |<- (desig:action-grounding ?action-designator [pick-up ?resolved-action-designator])
2  |
3  |
4  |
5  |
6  |
7  |  | (spec:property ?action-designator (:object ?object-designator))
8  |  | (design:current-designator ?object-designator ?current-object-desig)
9  |  | (spec:property ?current-object-desig (:type ?object-type))
10 |  | (spec:property ?current-object-desig (:name ?object-name))
11 |  | (> (spec:property ?action-designator (:arm ?arm))
12 |  | (true)
13 |  | (man-int:robot-free-hand ?_ ?arm))
14 |
15 |  | (lisp-fun man-int:get-object-old-transform ?current-object-desig ?object-transform)
16 |  | (lisp-fun man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
17 |
18 |
19 |
20 |  | (> (spec:property ?action-designator (:grasp ?grasp))
21 |  | (true)
22 |  | (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
23 |
24 |  | (member ?grasp ?grasps))
25 |  | (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
26 |  | (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
27 |  | (equal ?objects (?current-object-desig))
28 |  | (> (lisp-fun man-int:get-action-trajectory:picking ?arm ?grasp T ?objects
29 |  | ?picking-poses)
30 |  | (true)
31 |  | (lisp-fun man-int:get-traj-poses-by-label ?picking-pose :picking-up ?picking-up-poses))
32 |  | (> (desig:des-prop ?action-designator (:collision-mode ?collision-mode))
33 |  | (true)
34 |  | (equal ?collision-mode nil))
35 |  | (desig:designator action (:type :picking)
36 |  | (:object ?current-object-desig)
37 |  | (:object-name ?object-name)
38 |  | (:arm ?arm)
39 |  | (:gripper-opening ?gripper-opening)
40 |  | (:effort ?effort)
41 |  | (:grasp ?grasp)
42 |  | (:picking-up-poses ?picking-up-poses)
43 |  | (:collision-mode ?collision-mode))
44 |  | ?resolved-action-designator)
45 |

```

Figure 34: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Halving* (35 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
17	19	9	9

Findings:

- tba

The figure shows two side-by-side code snippets. The left snippet is a manual designator for the action *Picking Up*, consisting of 37 lines of Lisp-like code. The right snippet is a generated designator based on *Holding*, consisting of 51 lines. Both snippets use color coding to highlight different parts of the code: pink for action-grounding, green for properties, blue for objects, yellow for object types, and orange for specific actions or conditions. The generated designator on the right is significantly longer and more complex than the manual one on the left, reflecting the underlying logic of the generated action.

```

1  <- (desig:action-grounding ?action-designator :pick-up ?current-object-desig ?arm
2    ?gripper-opening ?efort ?grasp
3    ?left-reach-poses ?right-reach-poses
4    ?left-grasp-poses ?right-grasp-poses
5    ?left-lift-poses ?right-lift-poses)
6
7  (spec:property ?action-designator (:type :picking-up))
8  (spec:property ?action-designator (:object ?object-designator))
9  (design:current-designator ?object-designator ?current-object-desig)
10 (spec:property ?current-object-desig (:type ?object-type))
11 (spec:property ?current-object-desig (:name ?object-name))
12 (> (spec:property ?action-designator (:arm ?am))
13   (true)
14   (and (cram-robot-interfaces:robot ?robot)
15     (cram-robot-interfaces:arm ?robot ?am)))
16   (lisp-fun obj-int:get-object-transform ?current-object-desig ?object-transform)
17   (lisp-fun obj-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
18   (> (obj-int:object-relationally-symmetric ?object-type)
19     (equal ?rotationally-symmetric t)
20     (equal ?rotationally-symmetric nil))
21   (> (spec:property ?action-designator (:grasp ?grasp))
22     (true)
23   (and (lisp-fun obj-int:get-object-type-poses
24     ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
25     ?grasps)
26     (member ?grasp ?grasps))
27     (lisp-fun obj-int:get-object-type-gripping-effort ?object-type ?effort)
28     (lisp-fun obj-int:get-object-type-gripper-opening ?object-type ?gripper-opening)
29
30   (lisp-fun obj-int:get-object-grasping-poses
31     ?object-name ?object-type :left ?grasp ?object-transform
32     ?left-poses)
33
34   (lisp-fun obj-int:get-object-grasping-poses
35     ?object-name ?object-type :right ?grasp ?object-transform
36     ?right-poses)
37
38   (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses)
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```



```

1  <- (desig:action-grounding ?action-designator (pp-plans:pick-up ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

Figure 35: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Holding* (51 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
25	11	14	12

Findings:

- tba

The diagram illustrates two parallel state-space representations for a robotic arm's manipulation task. The left side shows a tree search space with various nodes representing actions like 'pick-up', 'grasp', and 'open'. The right side shows a grid-based representation where each cell corresponds to a specific pose and object type combination. Colored regions in both parts indicate different states or constraints, such as gripper opening/closing or object placement.

Figure 36: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Opening* (36 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
15	16	13	8

Findings:

- tba

```

1  <- (design:action-grounding ?action-designator (pick-up ?current-object-designator ?arm
2    ?grasper-opening ?effort ?grasp
3    ?left-reach-poses ?right-reach-poses
4    ?left-grasp-poses ?right-grasp-poses
5    ?left-lift-poses ?right-lift-poses))
6  (spec:property ?action-designator (type :picking-up))
7  (spec:property ?action-designator (object ?object-designator))
8  (desig current-designator ?object-designator ?current-object-design)
9  (spec:property ?current-object-design (type ?object-type))
10 (spec:property ?current-object-design (name ?object-name))
11 (spec:property ?action-designator (arm ?arm))
12 (true)
13 (and (cram-robot-interfaces robot ?robot)
14   (cram-robot-interfaces:arm ?robot ?arm))
15   (lisp-fun obj-int:get-object-transform ?current-object-design ?object-transform)
16   (lisp-fun obj-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
17   (-> (obj-int:object-rotationally-symmetric ?object-type)
18     (equal ?rotationally-symmetric t)
19     (equal ?rotationally-symmetric nil))
20   (-> (spec:property ?action-designator (grasp ?grasp)))
21   (true)
22   (and (lisp-fun obj-int:get-object-type-grasps
23     ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
24     ?grasps)
25     (member ?grasp ?grasps))
26   (lisp-fun obj-int:get-object-type-gripping-effort ?object-type ?effort)
27   (lisp-fun obj-int:get-object-type-gripper-opening ?object-type ?gripper-opening)
28   (lisp-fun obj-int:get-object-grasping-poses
29     ?object-name ?object-type :left ?grasp ?object-transform
30     ?left-poses)
31   (lisp-fun obj-int:get-object-grasping-poses
32     ?object-name ?object-type :right ?grasp ?object-transform
33     ?right-poses)
34   (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses
35     (?left-reach-poses ?right-reach-poses
36       ?left-grasp-poses ?right-grasp-poses
37       ?left-lift-poses ?right-lift-poses)))

```



```

1  <- (design:action-grounding ?action-designator (pick ?object-designator ?arm
2    ?left-reach-poses ?right-reach-poses
3    ?left-grasp-poses ?right-grasp-poses
4    ?left-lift-poses ?right-lift-poses))
5  (spec:property ?action-designator (type :picking))
6
7
8
9
10
11  (-> (spec:property ?action-designator (arm ?arm)))
12
13  (and (cram-robot-interfaces robot ?robot)
14    (cram-robot-interfaces:arm ?robot ?arm)))
15
16
17
18
19  (-> (spec:property ?action-designator (object ?object-designator)))
20
21
22
23
24
25
26  (cpoe:object-in-world ?object-designator)
27
28  (lisp-fun obj-int:get-object-grasping-poses
29    ?object-designator :left ?left-grasp-poses)
30
31  (lisp-fun obj-int:get-object-grasping-poses
32    ?object-designator :right ?right-grasp-poses)
33
34  (lisp-fun extract-pick-manipulation-poses ?arm ?left-grasp-poses ?right-grasp-poses
35    (?left-reach-poses ?right-reach-poses ?left-grasp-poses ?right-grasp-poses
36      ?left-lift-poses ?right-lift-poses)))
37

```

Figure 37: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Placing Down* (17 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
0	20	10	7

Findings:

- tba

```

1  :- (<- (design-action-grounding ?action-designator (pick-up ?current-object-design ?arm
2   ?gripper-opening ?effort ?grasp
3   ?left-reach-poses ?right-reach-poses
4   ?left-grasp-poses ?right-grasp-poses
5   ?left-lift-poses ?right-lift-poses))
6   (spec-property ?action-designator (type ?picking-up))
7   (spec-property ?action-designator ?object ?object-designator)
8   (design-current-designator ?object-designator ?current-object-design)
9   (spec-property ?current-object-design (type ?object-type))
10  (spec-property ?current-object-design (name ?object-name))
11  (-> (spec-property ?action-designator ?arm ?arm)
12   (true)
13   (and (cram-robot-interfaces robot ?robot)
14   (cram-robot-interfaces arm ?robot ?arm)))
15   (lisp-fun obj-int-get-object-transform ?current-object-design ?object-transform)
16   (lisp-fun obj-int-calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face))
17   (-> (obj-int-object-rotationally-symmetric ?object-type)
18   (equal ?rotationally-symmetric t)
19   (equal ?rotationally-symmetric nil))
20   (-> (spec-property ?action-designator (?grasp ?grasp))
21   (true)
22   (and (lisp-fun obj-int-get-object-type-grasps
23   ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
24   ?grasps)
25   (member ?grasp ?grasps))
26   (lisp-fun obj-int-get-object-type-gripping-effort ?object-type ?effort)
27   (lisp-fun obj-int-get-object-type-gripper-opening ?object-type ?gripper-opening)
28
29
30   (lisp-fun obj-int-get-object-grasping-poses
31   ?object-name ?object-type :left ?grasp ?object-transform
32   ?left-poses)
33   (lisp-fun obj-int-get-object-grasping-poses
34   ?object-name ?object-type :right ?grasp ?object-transform
35   ?right-poses)
36   (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses
37   ?left-reach-poses ?right-reach-poses
38   ?left-grasp-poses ?right-grasp-poses
39   ?left-lift-poses ?right-lift-poses)))
40
41
42
43
44
45

```



```

1  :- (<- (design-action-grounding ?action-designator (pick-up ?resolved-action-designator))
2
3
4
5
6   (spec-property ?action-designator (type ?picking))
7   (spec-property ?action-designator ?object ?object-designator)
8   (design-current-designator ?object-designator ?current-object-design)
9   (spec-property ?current-object-design (type ?object-type))
10  (spec-property ?current-object-design (name ?object-name))
11  (-> (spec-property ?action-designator (?arms ?arms))
12   (true)
13   (and (man-int:robot-free-hand ?_ ?arm)
14   (equal ?arms ?arm)))
15   (lisp-fun man-int-get-object-transform ?current-object-design ?object-transform)
16   (lisp-fun man-int-calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face))
17   (-> (man-int-object-rotationally-symmetric ?object-type)
18   (equal ?rotationally-symmetric t)
19   (equal ?rotationally-symmetric nil))
20   (-> (spec-property ?action-designator (?grasp ?grasp))
21   (true)
22   (and (lisp-fun man-int-get-action-grasps ?object-type ?arm ?object-transform ?grasps)
23   (member ?grasp ?grasps))
24   (lisp-fun man-int-get-grip-effort ?object-type ?effort)
25   (lisp-fun man-int-get-gripper-opening ?object-type ?gripper-opening)
26   (equal ?objects ?current-object-design)
27   (-> (member ?arm ?arms)
28   (lisp-fun man-int-get-action-trajectory :picking ?arm ?grasp T ?objects ?approach-pose)
29
30
31
32
33   (lisp-fun man-int-get-traj-poses-by-label ?approach-pose :approach ?approach-poses)
34   (-> (design-design-prop ?action-designator (collision-mode ?collision-mode)))
35   (true)
36   (equal ?collision-mode nil)
37   (design-designator :action (type ?picking)
38   (object ?current-object-design)
39   (object-type ?object-type)
40   (object-name ?object-name)
41   (arms ?arms)
42   (grasp ?grasp)
43   (approach-poses ?approach-poses)
44   (collision-mode ?collision-mode)))
45   ?resolved-action-designator)

```

Figure 38: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Pouring* (38 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
18	17	10	10

Findings:

- tba

```

1  #-< (desig:action-grounding ?action-designator (pick-up ?current-object-designator ?arm)
2   |  ?gripper-opening ?effort ?grasp
3   |  ?left-reach-poses ?right-reach-poses
4   |  ?left-grasp-poses ?right-grasp-poses
5   |  ?left-lift-poses ?right-lift-poses))
6
7  (+ (spec:property ?action-designator (type :picking-up))
8   + (spec:property ?action-designator (object ?object-designator))
9   + (design:current-designator ?object-designator ?current-object-design)
10  + (spec:property ?current-object-desig (type ?object-type))
11  + (spec:property ?current-object-desig (name ?object-name))
12  (-> (spec:property ?action-designator (?arm ?arm))
13   (true)
14  (-> (and (cram-robot-interfaces:robot ?robot)
15   (cram-robot-interfaces:arm ?robot ?arm)))
16  (-> (lisp-fun obj-int:get-object-transform ?current-object-desig ?object-transform)
17   (lisp-fun obj-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face)))
18  (-> (obj-int:object-rotationally-symmetric ?object-type)
19   (equal ?rotationally-symmetric t)
20   (equal ?rotationally-symmetric nil))
21  (-> (spec:property ?action-designator (?grasp ?grasp))
22   (true)
23  (-> (and (lisp-fun obj-int:get-object-type-grasps
24   ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
25   ?grasps)
26   (member ?grasp ?grasps)))
27  (-> (lisp-fun obj-int:get-object-type ?gripping-effort ?object-type ?effort)
28   (lisp-fun obj-int:get-object-type ?gripper-opening ?object-type ?gripper-opening)
29
30  (-> (lisp-fun obj-int:get-object-grasping-poses
31   ?object-name ?object-type ?left ?grasp ?object-transform
32   ?left-poses))
33  (-> (lisp-fun obj-int:get-object-grasping-poses
34   ?object-name ?object-type ?right ?grasp ?object-transform
35   ?right-poses))
36  (-> (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses
37   ?left-reach-poses ?right-reach-poses
38
39
40
41  ?left-grasp-poses ?right-grasp-poses
42  ?left-lift-poses ?right-lift-poses)))
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
10
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
25
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
788
789
789
790
791
792
793
794
795
796
796
797
797
798
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
838
839
839
840
841
842
843
844
845
846
847
847
848
848
849
849
850
851
852
853
854
855
856
857
858
858
859
859
860
861
862
863
864
865
866
867
868
868
869
869
870
871
872
873
874
875
876
877
877
878
878
879
879
880
881
882
883
884
885
886
886
887
887
888
888
889
889
890
891
892
893
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
928
928
929
929
930
931
932
933
934
935
936
937
938
938
939
939
940
941
942
943
944
945
946
947
947
948
948
949
949
950
951
952
953
954
955
956
957
958
958
959
959
960
961
962
963
964
965
966
967
967
968
968
969
969
970
971
972
973
974
975
976
976
977
977
978
978
979
979
980
981
982
983
984
985
985
986
986
987
987
988
988
989
989
990
991
992
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1710
1711
1711
1712
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1720
1721
1721
1722
1722
1723
1723
1724
1724
1725
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1730
1731
1731
1732
1732
1733
1733
1734
1734
1735
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1740
1741
174
```

Figure 39: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Slicing* (33 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
15	19	9	9

Findings:

- tba

```

1  |-< (desig action-grounding ?action-designator (pick-up ?current-object-desig ?arm
2   |  ?gripper-opening ?effort ?grasp
3   |  ?left-reach-poses ?right-reach-poses
4   |  ?left-grasp-poses ?right-grasp-poses
5   |  ?left-lift-poses ?right-lift-poses))
6
7  | (spec property ?action-designator (type :picking-up))
8  | (spec property ?action-designator (object ?object-designator))
9  | (design current-designator ?object-designator ?current-object-designator)
10 | (spec property ?current-object-desig (type ?object-type))
11 | (spec property ?current-object-desig (name ?object-name))
12 | (spec property ?action-designator (arm ?arm ?true))
13 | (and (cram-robot-interfaces:robot ?robot)
14 |       (cram-robot-interfaces:arm ?arm ?robot)))
15 | (lisp-fun ob:int get-object-transform ?current-object-designig ?object-transform)
16 | (lisp-fun ob:int calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
17 | (> (obj-int:object-rotationally-symmetric ?object-type)
18 |      (equal ?rotationally-symmetric t))
19
20
21 | (equal ?rotationally-symmetric nil)
22 | (> (spec property ?action-designator (:grasp ?grasp))
23 |      (true))
24 | (and (lisp-fun ob:int get-object-type-poses
25 |           ?object-type ?facing-robot-face ?bottom-face ?rotationally-symmetric ?arm
26 |           ?grasps)
27 |      (member ?grasp ?grasps))
28 | (lisp-fun ob:int get-object-type-gripping-effort ?object-type ?effort)
29 | (lisp-fun ob:int get-object-type-gripper-opening ?object-type ?gripper-opening)
30 | (lisp-fun ob:int get-object-grasping-poses
31 |      ?object-name ?object-type :left ?grasp ?object-transform
32 |      ?left-poses)
33 | (lisp-fun ob:int get-object-grasping-poses
34 |      ?object-name ?object-type :right ?grasp ?object-transform
35 |      ?right-poses)
36 | (lisp-fun extract-pick-up-manipulation-poses ?arm ?left-poses ?right-poses
37 |      ?left-reach-poses ?right-reach-poses
38 |      ?left-grasp-poses ?right-grasp-poses
39 |      ?left-lift-poses ?right-lift-poses))
40

1  |-< (desig action-grounding ?action-designator (pick-up ?resolved-action-designator))
2   | (spec property ?action-designator (type :picking))
3   | (spec property ?action-designator (grasp ?grasp))
4
5
6  | (spec property ?action-designator (arm ?arm))
7  | (design desig-prop ?action-designator (object ?object-designator))
8  | (design current-designator ?object-designator ?current-object-designator)
9  | (spec property ?current-object-designator (type ?object-type))
10 | (spec property ?current-object-designator (name ?object-name))
11 | (design desig-prop ?action-designator (?collision-mode ?collision-mode))
12 | (> (equal ?arm :left)
13 |      (and (lisp-fun get-trajectory :picking ?arm ?current-object-designator T NIL ?lists)
14 |            (lisp-fun man:int get-traj-poses-by-label ?lists :picking
15 |                  ?left-pick-up-poses)
16 |            (lisp-fun man:int get-traj-poses-by-label ?lists :initial
17 |                  ?left-initial-poses))
18 |      (and (equal ?left-pick-up-poses NIL)
19 |            (equal ?left-initial-poses NIL)))
20 |      (> (equal ?arm :right)
21 |            (and (lisp-fun get-trajectory :picking ?arm ?current-object-designator T NIL ?lists)
22 |                  (lisp-fun man:int get-traj-poses-by-label ?lists :picking
23 |                      ?right-pick-up-poses)
24 |                  (lisp-fun man:int get-traj-poses-by-label ?lists :initial
25 |                      ?right-initial-poses))
26 |          (and (equal ?right-pick-up-poses NIL)
27 |                (equal ?right-initial-poses NIL)))
28 |          (design:designator action (:type :picking)
29 |                            (:collision-mode ?collision-mode)
30 |                            (left-pick-up-poses ?left-pick-up-poses)
31 |                            (right-pick-up-poses ?right-pick-up-poses)
32 |                            (left-initial-poses ?left-initial-poses)
33 |                            (right-initial-poses ?right-initial-poses))
34 |          ?resolved-action-designator)))
35
36
37
38
39
40

```

Figure 40: Left: Manual designator for the action *Picking Up* (37 lines). Right: Generated designator based on *Wiping* (32 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
19	24	13	0

Findings:

- tba

```

1  | 1  | 
2  | 2  | 
3  | 3  | 
4  | 4  | 
5  | 5  | 
6  | 6  | 
7  | 7  | 
8  | 8  | 
9  | 9  | 
10 | 10 | 
11 | 11 | 
12 | 12 | 
13 | 13 | 
14 | 14 | 
15 | 15 | 
16 | 16 | 
17 | 17 | 
18 | 18 | 
19 | 19 | 
20 | 20 | 
21 | 21 | 
22 | 22 | 
23 | 23 | 
24 | 24 | 
25 | 25 | 
26 | 26 | 
27 | 27 | 
28 | 28 | 
29 | 29 | 
30 | 30 | 
31 | 31 | 
32 | 32 | 
33 | 33 | 
34 | 34 | 
35 | 35 | 
36 | 36 | 
37 | 37 | 
38 | 38 | 
39 | 39 | 
40 | 40 | 
41 | 41 | 
42 | 42 | 
43 | 43 | 
44 | 44 | 
45 | 45 | 
46 | 46 | 
47 | 47 | 
48 | 48 | 
49 | 49 | 
50 | 50 | 
51 | 51 | 
52 | 52 | 
53 | 53 | 
54 | 54 | 

```

Figure 41: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Closing* (31 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
14	29	14	3

Findings:

- designator definition contains eight instead of nine variables (2 overlap)
- some minor renaming (e.g. *held-object-name* instead of *object-name*)
- trajectories and poses are only defined and not calculated / used

```

1  ;<- (design:action-grounding ?action-designator (place ?current-object-designator ?arm
2    ?left-reach-poses ?right-reach-poses
3    ?left-put-poses ?right-put-poses
4    ?left-retract-poses ?right-retract-poses
5    ?location))
6  (spec:property ?action-designator (type :placing))
7
8  ;> (spec:property ?position-designator (arm ?arm))
9  ;> (spec:property ?action-designator (object ?object-designator))
10 ;> (or (cpe:object-in-hand ?object-designator ?arm)
11 ;> (and (format "WARNING: Wanted to place an object ~a with a arm ~a, ~
12 ;> but it's not in the arm.~%" ?object-designator ?arm)
13 ;> ))
14
15 ;> (spec:property ?object-in-hand ?object-designator ?arm)
16 ;> (cpe:object-in-hand ?object-designator ?arm)
17 ;> (and (cram-robot-interfaces robot ?robot)
18 ;> (cram-robot-interfaces:arm ?robot ?arm)
19 ;> (cpe:object-in-hand ?object-designator ?arm))
20 ;> (once (or (cpe:object-in-hand ?object-designator ?arm)
21 ;> (spec:property ?object-in-hand ?object-designator ?arm)))
22 ;> (design:current-designator ?object-designator ?current-object-designator)
23 ;> (spec:property ?current-object-designator (type ?object-type))
24 ;> (spec:property ?current-object-designator (name ?object-name))
25 ;> (spec:object-type-grasp ?object-type ?grasp)
26 ;> (spec:property ?action-designator (target ?location))
27 ;> (and (design:current-designator ?location ?current-location-designator)
28 ;> (design:designator-groundings ?current-location-designator ?poses)
29 ;> (member ?pose ?poses)
30 ;> (symbol-value :cram-if-not-base-frame* ?base-frame)
31 ;> (lisp-fun cram-if-ensure-pose-in-frame ?target-pose ?base-frame :use-zero-time t
32 ;> ?target-pose-in-base)
33 ;> (lisp-fun roslib-utilities:rosify-underscores-lisp-name ?object-name ?if-name)
34 ;> (lisp-fun cram-if:pose-stamped>transform-stamped ?target-pose-in-base ?if-name
35 ;> ?target-transform)
36 ;> (and (lisp-fun obj-int:object-transform ?current-object-designator ?target-transform)
37 ;> (design:designator ?location (pose ?target-pose) ?location))
38 ;> (lisp-fun obj-int:object-grasping-poses
39 ;> ?object-name ?object-type :left ?grasp ?target-transform
40 ;> ?left-poses)
41 ;> (lisp-fun obj-int:object-grasping-poses
42 ;> ?object-name ?object-type :right ?grasp ?target-transform
43 ;> ?right-poses)
44 ;> (lisp-fun extract-place-manipulation-poses ?arm ?left-poses ?right-poses
45 ;> ?left-reach-poses ?right-reach-poses ?left-put-poses ?right-put-poses
46 ;> ?left-retract-poses ?right-retract-poses)))
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

```



```

1  ;<- (design:action-grounding ?action-designator (place-down ?resolved-action-designator))
2
3
4
5
6
7  (spec:property ?action-designator (type :placing))
8  (spec:current-object-designator ?object ?object-designator)
9  (spec:property ?current-object-designer (type ?object-type))
10 (spec:property ?current-object-designer (name ?object-name))
11 ;> (spec:property ?action-designator (location ?location-designator))
12 ;> (spec:property ?current-location-designer ?current-location-desig)
13 ;> (spec:property ?current-location-design (name ?location-name))
14 ;> (lisp-fun man-int:man-int-get-location-pose ?current-location-design ?location-pose)
15 ;> (spec:property ?action-designator (arm ?arm))
16 ;> (true)
17 ;> (man-int:robot-holding-object ?arm ?current-object-design)
18 ;> (lisp-fun man-int:man-int-get-object-old-transform ?current-object-design ?object-transform)
19
20
21 ;> (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
22 ;> (spec:property ?action-designator (grasp ?grasp))
23 ;> (true)
24 ;> (and (lisp-fun man-int:man-int-get-action-grasps ?object-type ?arm ?object-transform ?grasps
25 ;> (member ?grasp ?grasps)))
26 ;> (lisp-fun man-int:man-int-get-action-gripping-effort ?object-type ?effort)
27
28
29
30
31
32 ;> (lisp-fun man-int:man-int-get-action-gripper-opening ?object-type ?gripper-opening)
33 ;> (equal ?objects ?current-object-design)
34 ;> (equal ?arm :left)
35 ;> (and (lisp-fun man-int:man-int-get-action-trajectory :placing ?arm ?grasp T ?objects
36 ;> ?left-placing-down-poses)
37 ;> (lisp-fun man-int:man-int-get-drag-poses-by-label ?left-placing-pose :placing-down
38 ;> ?left-placing-down-poses)
39 ;> (equal ?left-placing-down-poses NIL))
40 ;> (equal ?arm :right)
41 ;> (and (lisp-fun man-int:man-int-get-action-trajectory :placing ?arm ?grasp T ?objects
42 ;> ?right-placing-pose ?location-pose)
43
44 ;> (lisp-fun man-int:man-int-get-trip-poses-by-label ?right-placing-pose :placing-down
45 ;> ?right-placing-down-poses)
46 ;> (equal ?right-placing-down-poses NIL))
47 ;> (design:design-prop ?action-designator (collision-mode ?collision-mode))
48 ;> (true)
49 ;> (equal ?collision-mode NIL)
50 ;> (design:designator :action (type :placing)
51 ;> (object ?current-object-design)
52 ;> (name ?current-object-name)
53 ;> (arm ?arm)
54 ;> (gripper-opening ?gripper-opening)
55 ;> (effort ?effort)
56 ;> (grasp ?grasp)
57 ;> (location ?current-location-design)
58 ;> (location ?current-location-design)
59 ;> (lisp-fun man-int:man-int-get-left-placing-down-poses ?left-placing-down-poses)
60 ;> (lisp-fun man-int:man-int-get-right-placing-down-poses ?right-placing-down-poses)
61 ;> (collision-mode ?collision-mode))
62 ;> (resolved-action-designator)
63

```

Figure 42: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Halving* (51 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
43	38	6	2

Findings:

- designator definition contains only one instead of nine variables
- designator calculates trajectories and poses in a different manner
- designator (falsely) ends by mapping the variables to designator "parts"

```

1 1 <- (design-action-grounding ?action-designator (place ?current-object-designator ?arm
2   ?left-reach-poses ?right-reach-poses
3   ?left-put-poses ?right-put-poses
4   ?left-retract-poses ?right-retract-poses
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

(spec:property ?action-designator (type :placing-down))
 (spec:property ?action-designator (object ?object-designator))
 (and (format "WARNING: Wanted to place an object ~a with an arm ~a, ~a but it's not in the arm.-%~a" ?object-designator ?arm))
 (cram:object-in-hand ?object-designator ?arm))
 (or (cram:object-in-hand ?object-designator ?arm))
 (and (cram:robot-interfaces:robot ?robot)
 (cram:robot-interfaces:arm ?robot ?arm))
 (cram:object-in-hand ?object-designator ?arm)))
 (once (or (cram:object-in-hand ?object-designator ?arm))
 (design-current-designator ?current-object-designator ?object-type ?object-name))
 (spec:property ?current-object-designator (type ?object-type))
 (spec:property ?current-object-designator (name ?object-name))
 (obj-int:object-type-grasp ?object-type ?grasp))
 (> (spec:property ?action-designator (target ?location))
 (and (design-current-designator (location ?current-location-designator)
 (design-current-designator ?current-location-designator ?poses)
 (symbol-value cram-if::"robot-base-frame" ?base-frame)
 (lisp-fun cram-if:ensure-pose-in-frame ?target-pose ?base-frame :use-zero-time t
 ?target-pose-in-base))
 (lisp-fun roslib-utilities:rosify-underscores-lisp-name ?object-name ?t1-name)
 (lisp-fun cram-if:pose-stamped->transform-stamped ?target-pose-in-base ?t1-name
 ?target-transform))
 (and (lisp-fun obj-int:get-object-transform ?current-object-designator ?target-transform)
 (lisp-fun obj-int:get-object-?pose ?current-object-designator ?target-?pose)
 (design-designator location (?pose ?target-?pose) ?location)))
 (lisp-fun obj-int:get-object-grasping-poses
 ?object-name ?object-type ?left ?grasp ?target-transform
 ?right-poses)
 (lisp-fun extract-place-manipulation-poses ?arm ?left-poses ?right-poses
 ?left-reach-poses ?right-reach-poses ?left-put-poses ?right-put-poses
 ?left-retract-poses ?right-retract-poses))))

(lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
 (equal ?objects (?current-object-designator))
 (design-designator :action ((type :placing-down)
 (object ?current-object-designator)
 (object-name ?object-name)
 (location ?current-location-designator)
 (location-name ?location-name)
 (arm ?arm)
 (gripper-opening ?gripper-opening)
 (effort ?effort)
 (reach-poses ?reach-poses)
 (place-poses ?place-poses)))
 ?resolved-action-designator)

Figure 43: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Holding* (35 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
26	37	8	1

Findings:

- designator definition contains only one instead of nine variables
- designator sets high focus on definitions relevant for calculating trajectories / poses instead of using them
- designator (falsely) ends by mapping the variables to designator "parts"

```

1  :-> (design.action-grounding ?action-designator (place ?current-object-designator ?arm
2      ?left-reach-poses ?right-reach-poses
3
4      ?left-put-poses ?right-put-poses
5      ?left-retract-poses ?right-retract-poses
6
7      ?location))
8      (spec.property ?action-designator (:type :placing))
9
10
11
12
13
14
15  :-> (spec.property ?action-designator (:arm ?arm))
16      (-> (spec.property ?action-designator (:object ?object ?object-designator))
17          (cpose:object-in-hand ?object ?object-designator ?arm)
18          (and) (format "WARNING: Wanted to place an object ~a with arm ~a, ~
19          but it's not the arm - %~a" ?object-designator ?arm)
20
21      ) (cpose:object-in-hand ?object-designator ?arm)
22      (-> (spec.property ?action-designator (:object ?object ?object-designator))
23          (cpose:object-in-hand ?object ?object-designator ?arm)
24          (and) (cram:robot-interfaces:arm ?robot ?arm)
25          (cpose:object-in-hand ?object-designator ?arm)))
26
27  :-> (once (or (cpose:object-in-hand ?object ?object-designator ?arm)
28      (spec.property ?action-designator (:object ?object ?object-designator)))
29      (desig.current-designator ?object-designator ?current-object-designator)
30      (spec.property ?current-object-designator (:type ?object-type))
31      (spec.property ?current-object-designator (:name ?object-name))
32
33  :-> (spec.property ?action-designator (?target ?location))
34      (and) (desig.current-designator ?location ?current-location-designator)
35      (design.designator-groundings ?current-location-designator ?poses)
36      (member ?target-pose ?poses)
37      (symbol-value cram:if:robot-base-frame *base-frame*)
38      (lisp-fun cram:tf:ensure-pose-in-frame ?target-pose ?base-frame :use-zero-time t
39      ?target-pose-in-base)
40      (lisp-fun ros-lisp:utilities:rosify-underscores-lisp-name ?object-name ?tf-name)
41      (lisp-fun cram:tf:pose-stamped->transform-stamped ?target-pose-in-base ?tf-name
42      ?target-transform)
43
44  :-> (and (lisp-fun obj:int-get-object-transform ?current-object-designator ?target-transform)
45      (lisp-fun obj:int-get-object-base-pose ?current-object-designator ?target-pose)
46      (lisp-fun obj:int-get-object-location (:pose ?target-pose) ?location))
47      (lisp-fun obj:int-get-object-grasping-poses
48          ?object-name ?object-type ?left ?grasp ?target-transform
49          ?left-poses)
50      (lisp-fun obj:int-get-object-grasping-poses
51          ?object-name ?object-type ?right ?grasp ?target-transform
52          ?right-poses)
53
54  :-> (lisp-fun extract-place-manipulation-poses ?arm ?left-poses ?right-poses
55      (?left-reach-poses ?right-reach-poses ?left-put-poses ?right-put-poses
56      ?left-retract-poses ?right-retract-poses)))

```



```

1  :-> (design.action-grounding ?action-designator (place-down ?arm
2      ?gripper-opening
3      ?distance
4      ?left-reach-poses
5      ?right-reach-poses
6      ?left-release-poses
7      ?right-release-poses
8      ?location-designator))
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

Figure 44: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Opening* (33 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
19	32	11	3

Findings:

- designator definition contains eight instead of nine variables (2 overlap)
- designator sets high focus on definitions relevant for calculating trajectories / poses instead of using them

```

1  :-> (design.action-grounding ?action-designator [place ?current-object-designator ?arm
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

```

1  :-> (design.action-grounding ?action-designator [place-down ?current-object-desig ?arm
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

Figure 45: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Picking Up* (29 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
10	27	17	2

Findings:

- tba

```

1  | (spec property ?action-designator (place ?current-object-designator ?arm
2  |   ?left-reach-poses ?right-reach-poses
3  |   ?left-put-poses ?right-put-poses
4  |   ?left-retract-poses ?right-retract-poses
5  |   ?location))
6  | 
7  | (spec property ?action-designator (type :placing))
8  | (spec property ?action-designator (arm ?arm))
9  | (> (spec property ?action-designator (object ?object-designator))
10 |  (or (cpe:object-in-hand ?object-designator ?arm)
11 |      (and (format "WARNING: Wanted to place an object ~a with arm ~a, ~
12 |      but it's not in the arm. ~c" ?object-designator ?arm)
13 |          ))
14 | 
15 | (cpe:object-in-hand ?object-designator ?arm))
16 | (spec property ?action-designator (object ?object-designator))
17 | (cpe:object-in-hand ?object-designator ?arm)
18 | (and (cpe:object-in-hand ?object-designator ?arm)
19 |       (cram-robot-interfaces:arm ?robot ?arm)
20 |       (cpe:object-in-hand ?object-designator ?arm)))
21 | 
22 | (once (or (cpe:object-in-hand ?object-designator ?arm)
23 |            (spec property ?action-designator (object ?object-designator))
24 |            (spec property ?current-object-designator (type ?object-type))
25 |            (spec property ?current-object-designator (name ?object-name))
26 |            (spec property ?current-object-designator (pose ?poses))
27 |            (spec property ?action-designator (?target ?location))
28 |            (and (desig:current-designator ?location ?current-location-designator)
29 |                  (desig:designator-groundings ?current-location-designator ?poses)
30 |                  (member ?target-poses ?poses)
31 |                  (symbol-value cram-if::"robot-base-frame" ?base-frame)
32 |                  (lisp-fun cram-if::"base-frame-in-frame" ?target-poses ?base-frame :use-zero-time t
33 |                      ?target-poses-in-base)
34 |                  (lisp-fun ros-lisp::utilities::rosly-underscores-lisp-name ?object-name ?if-name)
35 |                  (lisp-fun cram-if::"pose-stamped->transform-stamped" ?target-poses-in-base ?if-name
36 |                      ?target-transform)
37 |                  (and (lisp-fun obj-int::get-object-transform ?current-object-designator ?target-transform)
38 |                      (lisp-fun obj-int::get-object-pose ?current-object-designator ?target-pose)
39 |                      (lisp-fun obj-int::get-object-location ((?pose ?target-pose) ?location)))
40 |                      (lisp-fun obj-int::get-object-grasping-poses
41 |                          ?object-name ?object-type :left "grasp" ?target-transform
42 |                          ?left-poses)
43 |                      (lisp-fun obj-int::get-object-grasping-poses
44 |                          ?object-name ?object-type :right "grasp" ?target-transform
45 |                          ?right-poses)
46 |                      (lisp-fun extract-place-manipulation-poses ?arm ?left-poses ?right-poses
47 | 
48 | 
49 | 
50 | 
51 | 
52 | 
53 | 
54 | 
55 | 
56 | 
57 | 
58 | 
59 | 
60 | 
61 | 
62 | 
63 | 
64 | 
65 | 
66 | 
67 | 
68 | 
69 | 
70 | 
71 | 
72 | 
73 | 
74 | 
75 | 
76 | 
77 | 
78 | 
79 | 
80 | 
81 | 
82 | 
83 | 
84 | 
85 | 
86 | 
87 | 
88 | 
89 | 
90 | 
91 | 
92 | 
93 | 
94 | 
95 | 
96 | 
97 | 
98 | 
99 | 
100 | 
101 | 
102 | 
103 | 
104 | 
105 | 
106 | 
107 | 
108 | 
109 | 
110 | 
111 | 
112 | 
113 | 
114 | 
115 | 
116 | 
117 | 
118 | 
119 | 
120 | 
121 | 
122 | 
123 | 
124 | 
125 | 
126 | 
127 | 
128 | 
129 | 
130 | 
131 | 
132 | 
133 | 
134 | 
135 | 
136 | 
137 | 
138 | 
139 | 
140 | 
141 | 
142 | 
143 | 
144 | 
145 | 
146 | 
147 | 
148 | 
149 | 
150 | 
151 | 
152 | 
153 | 
154 | 
155 | 
156 | 
157 | 
158 | 
159 | 
160 | 
161 | 
162 | 
163 | 
164 | 
165 | 
166 | 
167 | 
168 | 
169 | 
170 | 
171 | 
172 | 
173 | 
174 | 
175 | 
176 | 
177 | 
178 | 
179 | 
180 | 
181 | 
182 | 
183 | 
184 | 
185 | 
186 | 
187 | 
188 | 
189 | 
190 | 
191 | 
192 | 
193 | 
194 | 
195 | 
196 | 
197 | 
198 | 
199 | 
200 | 
201 | 
202 | 
203 | 
204 | 
205 | 
206 | 
207 | 
208 | 
209 | 
210 | 
211 | 
212 | 
213 | 
214 | 
215 | 
216 | 
217 | 
218 | 
219 | 
220 | 
221 | 
222 | 
223 | 
224 | 
225 | 
226 | 
227 | 
228 | 
229 | 
230 | 
231 | 
232 | 
233 | 
234 | 
235 | 
236 | 
237 | 
238 | 
239 | 
240 | 
241 | 
242 | 
243 | 
244 | 
245 | 
246 | 
247 | 
248 | 
249 | 
250 | 
251 | 
252 | 
253 | 
254 | 
255 | 
256 | 
257 | 
258 | 
259 | 
260 | 
261 | 
262 | 
263 | 
264 | 
265 | 
266 | 
267 | 
267 | 
268 | 
269 | 
270 | 
271 | 
272 | 
273 | 
274 | 
275 | 
276 | 
277 | 
278 | 
279 | 
280 | 
281 | 
282 | 
283 | 
284 | 
285 | 
286 | 
287 | 
287 | 
288 | 
289 | 
290 | 
291 | 
292 | 
293 | 
294 | 
295 | 
296 | 
297 | 
298 | 
299 | 
300 | 
301 | 
302 | 
303 | 
304 | 
305 | 
306 | 
307 | 
308 | 
309 | 
310 | 
311 | 
312 | 
313 | 
314 | 
315 | 
316 | 
317 | 
317 | 
318 | 
319 | 
320 | 
321 | 
322 | 
323 | 
324 | 
325 | 
326 | 
327 | 
327 | 
328 | 
329 | 
330 | 
331 | 
332 | 
333 | 
334 | 
335 | 
336 | 
337 | 
337 | 
338 | 
339 | 
340 | 
341 | 
342 | 
343 | 
344 | 
345 | 
346 | 
347 | 
347 | 
348 | 
349 | 
350 | 
351 | 
352 | 
353 | 
354 | 
355 | 
356 | 
357 | 
358 | 
359 | 
360 | 
361 | 
362 | 
363 | 
364 | 
365 | 
366 | 
367 | 
367 | 
368 | 
369 | 
370 | 
371 | 
372 | 
373 | 
374 | 
375 | 
376 | 
377 | 
378 | 
379 | 
380 | 
381 | 
382 | 
383 | 
384 | 
385 | 
386 | 
387 | 
387 | 
388 | 
389 | 
390 | 
391 | 
392 | 
393 | 
394 | 
395 | 
396 | 
397 | 
398 | 
399 | 
400 | 
401 | 
402 | 
403 | 
404 | 
405 | 
406 | 
407 | 
408 | 
409 | 
410 | 
411 | 
412 | 
413 | 
414 | 
415 | 
416 | 
417 | 
418 | 
419 | 
420 | 
421 | 
422 | 
423 | 
424 | 
425 | 
426 | 
427 | 
428 | 
429 | 
430 | 
431 | 
432 | 
433 | 
434 | 
435 | 
436 | 
437 | 
438 | 
439 | 
440 | 
441 | 
442 | 
443 | 
444 | 
445 | 
446 | 
447 | 
448 | 
449 | 
450 | 
451 | 
452 | 
453 | 
454 | 
455 | 
456 | 
457 | 
458 | 
459 | 
460 | 
461 | 
462 | 
463 | 
464 | 
465 | 
466 | 
467 | 
468 | 
469 | 
470 | 
471 | 
472 | 
473 | 
474 | 
475 | 
476 | 
477 | 
478 | 
479 | 
480 | 
481 | 
482 | 
483 | 
484 | 
485 | 
486 | 
487 | 
488 | 
489 | 
490 | 
491 | 
492 | 
493 | 
494 | 
495 | 
496 | 
497 | 
498 | 
499 | 
500 | 
501 | 
502 | 
503 | 
504 | 
505 | 
506 | 
507 | 
508 | 
509 | 
510 | 
511 | 
512 | 
513 | 
514 | 
515 | 
516 | 
517 | 
518 | 
519 | 
519 | 
520 | 
521 | 
522 | 
523 | 
524 | 
525 | 
526 | 
527 | 
528 | 
529 | 
529 | 
530 | 
531 | 
532 | 
533 | 
534 | 
535 | 
536 | 
537 | 
538 | 
539 | 
540 | 
541 | 
542 | 
543 | 
544 | 
545 | 
546 | 
547 | 
548 | 
549 | 
550 | 
551 | 
552 | 
553 | 
554 | 
555 | 
556 | 
557 | 
558 | 
559 | 
560 | 
561 | 
562 | 
563 | 
564 | 
565 | 
566 | 
567 | 
568 | 
569 | 
570 | 
571 | 
572 | 
573 | 
574 | 
575 | 
576 | 
577 | 
578 | 
579 | 
580 | 
581 | 
582 | 
583 | 
584 | 
585 | 
586 | 
587 | 
588 | 
589 | 
589 | 
590 | 
591 | 
592 | 
593 | 
594 | 
595 | 
596 | 
597 | 
598 | 
599 | 
599 | 
600 | 
601 | 
602 | 
603 | 
604 | 
605 | 
606 | 
607 | 
608 | 
609 | 
610 | 
611 | 
612 | 
613 | 
614 | 
615 | 
616 | 
617 | 
618 | 
619 | 
619 | 
620 | 
621 | 
622 | 
623 | 
624 | 
625 | 
626 | 
627 | 
628 | 
629 | 
629 | 
630 | 
631 | 
632 | 
633 | 
634 | 
635 | 
636 | 
637 | 
638 | 
639 | 
640 | 
641 | 
642 | 
643 | 
644 | 
645 | 
646 | 
647 | 
648 | 
649 | 
649 | 
650 | 
651 | 
652 | 
653 | 
654 | 
655 | 
656 | 
657 | 
658 | 
659 | 
660 | 
661 | 
662 | 
663 | 
664 | 
665 | 
666 | 
667 | 
668 | 
669 | 
669 | 
670 | 
671 | 
672 | 
673 | 
674 | 
675 | 
676 | 
677 | 
678 | 
679 | 
679 | 
680 | 
681 | 
682 | 
683 | 
684 | 
685 | 
686 | 
687 | 
688 | 
689 | 
689 | 
690 | 
691 | 
692 | 
693 | 
694 | 
695 | 
696 | 
697 | 
698 | 
699 | 
699 | 
700 | 
701 | 
702 | 
703 | 
704 | 
705 | 
706 | 
707 | 
708 | 
709 | 
709 | 
710 | 
711 | 
712 | 
713 | 
714 | 
715 | 
716 | 
717 | 
718 | 
719 | 
719 | 
720 | 
721 | 
722 | 
723 | 
724 | 
725 | 
726 | 
727 | 
728 | 
729 | 
729 | 
730 | 
731 | 
732 | 
733 | 
734 | 
735 | 
736 | 
737 | 
738 | 
739 | 
739 | 
740 | 
741 | 
742 | 
743 | 
744 | 
745 | 
746 | 
747 | 
748 | 
749 | 
749 | 
750 | 
751 | 
752 | 
753 | 
754 | 
755 | 
756 | 
757 | 
758 | 
759 | 
759 | 
760 | 
761 | 
762 | 
763 | 
764 | 
765 | 
766 | 
767 | 
768 | 
769 | 
769 | 
770 | 
771 | 
772 | 
773 | 
774 | 
775 | 
776 | 
777 | 
778 | 
779 | 
779 | 
780 | 
781 | 
782 | 
783 | 
784 | 
785 | 
786 | 
787 | 
788 | 
789 | 
789 | 
790 | 
791 | 
792 | 
793 | 
794 | 
795 | 
796 | 
797 | 
798 | 
799 | 
799 | 
800 | 
801 | 
802 | 
803 | 
804 | 
805 | 
806 | 
807 | 
808 | 
809 | 
809 | 
810 | 
811 | 
812 | 
813 | 
814 | 
815 | 
816 | 
817 | 
818 | 
819 | 
819 | 
820 | 
821 | 
822 | 
823 | 
824 | 
825 | 
826 | 
827 | 
828 | 
829 | 
829 | 
830 | 
831 | 
832 | 
833 | 
834 | 
835 | 
836 | 
837 | 
838 | 
839 | 
839 | 
840 | 
841 | 
842 | 
843 | 
844 | 
845 | 
845 | 
846 | 
847 | 
848 | 
849 | 
850 | 
851 | 
852 | 
853 | 
854 | 
855 | 
856 | 
857 | 
858 | 
859 | 
859 | 
860 | 
861 | 
862 | 
863 | 
864 | 
865 | 
866 | 
867 | 
868 | 
869 | 
869 | 
870 | 
871 | 
872 | 
873 | 
874 | 
875 | 
876 | 
877 | 
878 | 
879 | 
879 | 
880 | 
881 | 
882 | 
883 | 
884 | 
885 | 
886 | 
887 | 
888 | 
889 | 
889 | 
890 | 
891 | 
892 | 
893 | 
894 | 
895 | 
896 | 
897 | 
898 | 
899 | 
899 | 
900 | 
901 | 
902 | 
903 | 
904 | 
905 | 
906 | 
907 | 
908 | 
909 | 
909 | 
910 | 
911 | 
912 | 
913 | 
914 | 
915 | 
916 | 
917 | 
918 | 
919 | 
919 | 
920 | 
921 | 
922 | 
923 | 
924 | 
925 | 
926 | 
927 | 
928 | 
929 | 
929 | 
930 | 
931 | 
932 | 
933 | 
934 | 
935 | 
936 | 
937 | 
938 | 
939 | 
939 | 
940 | 
941 | 
942 | 
943 | 
944 | 
945 | 
946 | 
947 | 
948 | 
949 | 
949 | 
950 | 
951 | 
952 | 
953 | 
954 | 
955 | 
956 | 
957 | 
958 | 
959 | 
959 | 
960 | 
961 | 
962 | 
963 | 
964 | 
965 | 
966 | 
967 | 
968 | 
969 | 
969 | 
970 | 
971 | 
972 | 
973 | 
974 | 
975 | 
976 | 
977 | 
978 | 
979 | 
979 | 
980 | 
981 | 
982 | 
983 | 
984 | 
985 | 
986 | 
987 | 
988 | 
989 | 
989 | 
990 | 
991 | 
992 | 
993 | 
994 | 
995 | 
996 | 
997 | 
998 | 
999 | 
999 | 
1000 | 
1001 | 
1002 | 
1003 | 
1004 | 
1005 | 
1006 | 
1007 | 
1008 | 
1009 | 
1009 | 
1010 | 
1011 | 
1012 | 
1013 | 
1014 | 
1015 | 
1016 | 
1017 | 
1018 | 
1019 | 
1019 | 
1020 | 
1021 | 
1022 | 
1023 | 
1024 | 
1025 | 
1026 | 
1027 | 
1028 | 
1029 | 
1029 | 
1030 | 
1031 | 
1032 | 
1033 | 
1034 | 
1035 | 
1036 | 
1037 | 
1038 | 
1039 | 
1039 | 
1040 | 
1041 | 
1042 | 
1043 | 
1044 | 
1045 | 
1046 | 
1047 | 
1047 | 
1048 | 
1049 | 
1050 | 
1051 | 
1052 | 
1053 | 
1054 | 
1055 | 
1056 | 
1057 | 
1058 | 
1059 | 
1059 | 
1060 | 
1061 | 
1062 | 
1063 | 
1064 | 
1065 | 
1066 | 
1067 | 
1068 | 
1069 | 
1069 | 
1070 | 
1071 | 
1072 | 
1073 | 
1074 | 
1075 | 
1076 | 
1077 | 
1078 | 
1078 | 
1079 | 
1080 | 
1081 | 
1082 | 
1083 | 
1084 | 
1085 | 
1086 | 
1087 | 
1088 | 
1088 | 
1089 | 
1090 | 
1091 | 
1092 | 
1093 | 
1094 | 
1095 | 
1096 | 
1097 | 
1097 | 
1098 | 
1099 | 
1099 | 
1100 | 
1101 | 
1102 | 
1103 | 
1104 | 
1105 | 
1106 | 
1107 | 
1108 | 
1109 | 
1109 | 
1110 | 
1111 | 
1112 | 
1113 | 
1114 | 
1115 | 
1116 | 
1117 | 
1118 | 
1119 | 
1119 | 
1120 | 
1121 | 
1122 | 
1123 | 
1124 | 
1125 | 
1126 | 
1127 | 
1128 | 
1129 | 
1129 | 
1130 | 
1131 | 
1132 | 
1133 | 
1134 | 
1135 | 
1136 | 
1137 | 
1138 | 
1139 | 
1139 | 
1140 | 
1141 | 
1142 | 
1143 | 
1144 | 
1145 | 
1146 | 
1147 | 
1148 | 
1148 | 
1149 | 
1150 | 
1151 | 
1152 | 
1153 | 
1154 | 
1155 | 
1156 | 
1157 | 
1158 | 
1158 | 
1159 | 
1160 | 
1161 | 
1162 | 
1163 | 
1164 | 
1165 | 
1166 | 
1167 | 
1168 | 
1169 | 
1169 | 
1170 | 
1171 | 
1172 | 
1173 | 
1174 | 
1175 | 
1176 | 
1177 | 
1177 | 
1178 | 
1179 | 
1180 | 
1181 | 
1182 | 
1183 | 
1184 | 
1185 | 
1186 | 
1187 | 
1187 | 
1188 | 
1189 | 
1190 | 
1191 | 
1192 | 
1193 | 
1194 | 
1195 | 
1195 | 
1196 | 
1197 | 
1198 | 
1199 | 
1199 | 
1200 | 
1201 | 
1202 | 
1203 | 
1204 | 
1205 | 
1206 | 
1207 | 
1208 | 
1209 | 
1209 | 
1210 | 
1211 | 
1212 | 
1213 | 
1214 | 
1215 | 
1216 | 
1217 | 
1218 | 
1219 | 
1219 | 
1220 | 
1221 | 
1222 | 
1223 | 
1224 | 
1225 | 
1226 | 
1227 | 
1228 | 
1229 | 
1229 | 
1230 | 
1231 | 
1232 | 
1233 | 
1234 | 
1235 | 
1236 | 
1237 | 
1238 | 
1239 | 
1239 | 
1240 | 
1241 | 
1242 | 
1243 | 
1244 | 
1245 | 
1246 | 
1247 | 
1247 | 
1248 | 
1249 | 
1250 | 
1251 | 
1252 | 
1253 | 
1254 | 
1255 | 
1256 | 
1257 | 
1258 | 
1258 | 
1259 | 
1260 | 
1261 | 
1262 | 
1263 | 
1264 | 
1265 | 
1266 | 
1267 | 
1268 | 
1269 | 
1269 | 
1270 | 
1271 | 
1272 | 
1273 | 
1274 | 
1275 | 
1276 | 
1277 | 
1277 | 
1278 | 
1279 | 
1280 | 
1281 | 
1282 | 
1283 | 
1284 | 
1285 | 
1286 | 
1287 | 
1287 | 
1288 | 
1289 | 
1290 | 
1291 | 
1292 | 
1293 | 
1294 | 
1295 | 
1295 | 
1296 | 
1297 | 
1298 | 
1299 | 
1299 | 
1300 | 
1301 | 
1302 | 
1303 | 
1304 | 
1305 | 
1306 | 
1307 | 
1308 | 
1309 | 
1309 | 
1310 | 
1311 | 
1312 | 
1313 | 
1314 | 
1315 | 
1316 | 
1317 | 
1318 | 
1319 | 
1319 | 
1320 | 
1321 | 
1322 | 
1323 | 
1324 | 
1325 | 
1326 | 
1327 | 
1328 | 
1329 | 
1329 | 
1330 | 
1331 | 
1332 | 
1333 | 
1334 | 
1335 | 
1336 | 
1337 | 
1338 | 
1339 | 
1339 | 
1340 | 
1341 | 
1342 | 
1343 | 
1344 | 
1345 | 
1346 | 
1347 | 
1348 | 
1348 | 
1349 | 
1350 | 
1351 | 
1352 | 
1353 | 
1354 | 
1355 | 
1356 | 
1357 | 
1358 | 
1358 | 
1359 | 
1360 | 
1361 | 
1362 | 
1363 | 
1364 | 
1365 | 
1366 | 
1367 | 
1368 | 
1369 | 
1369 | 
1370 | 
1371 | 
1372 | 
1373 | 
1374 | 
1375 | 
1376 | 
1377 | 
1377 | 
1378 | 
1379 | 
1380 | 
1381 | 
1382 | 
1383 | 
1384 | 
1385 | 
1386 | 
1387 | 
1387 | 
1388 | 
1389 | 
1390 | 
1391 | 
1392 | 
1393 | 
1394 | 
1394 | 
1395 | 
1396 | 
1397 | 
1398 | 
1399 | 
1399 | 
1400 | 
1401 | 
1402 | 
1403 | 
1404 | 
1405 | 
1406 | 
1407 | 
1408 | 
1409 | 
1409 | 
1410 | 
1411 | 
1412 | 
1413 | 
1414 | 
1415 | 
1416 | 
1417 | 
1418 | 
1418 | 
1419 | 
1420 | 
1421 | 
1422 | 
1423 | 
1424 | 
1425 | 
1426 | 
1427 | 
1428 | 
1428 | 
1429 | 
1430 | 
1431 | 
1432 | 
1433 | 
1434 | 
1435 | 
1436 | 
1437 | 
1438 | 
1438 | 
1439 | 
1440 | 
1441 | 
1442 | 
1443 | 
1444 | 
1445 | 
1446 | 
1447 | 
1448 | 
1448 | 
1449 | 
1450 | 
1451 | 
1452 | 
1453 | 
1454 | 
1455 | 
1456 | 
1457 | 
1458 | 
1458 | 
1459 | 
1460 | 
1461 | 
1462 | 
1463 | 
1464 | 
1465 | 
1466 | 
1467 | 
1468 | 
1469 | 
1469 | 
1470 | 
1471 | 
1472 | 
1473 | 
1474 | 
1475 | 
1476 | 
1477 | 
1477 | 
1478 | 
1479 | 
1480 | 
1481 | 
1482 | 
1483 | 
1484 | 
1485 | 
1486 | 
1487 | 
1487 | 
1488 | 
1489 | 
1490 | 
1491 | 
1492 | 
1493 | 
1494 | 
1494 | 
1495 | 
1496 | 
1497 | 
1498 | 
1499 | 
1499 | 
1500 | 
1501 | 
1502 | 
1503 | 
1504 | 
1505 | 
1506 | 
1507 | 
1508 | 
1509 | 
1509 | 
1510 | 
1511 | 
1512 | 
1513 | 
1514 | 
1515 | 
1516 | 
1517 | 
1518 | 
1518 | 
1519 | 
1520 | 
1521 | 
1522 | 
1523 | 
1524 | 
1525 | 
1526 | 
1527 | 
1528 | 
1528 | 
1529 | 
1530 | 
1531 | 
1532 | 
1533 | 
1534 | 
1535 | 
1536 | 
1537 | 
1538 | 
1538 | 
1539 | 
1540 | 
1541 | 
1542 | 
1543 | 
1544 | 
1545 | 
1546 | 
1547 | 
1548 | 
1548 | 
1549 | 
1550 | 
1551 | 
1552 | 
1553 | 
1554 | 
1555 | 
1556 | 
1557 | 
1558 | 
1558 | 
1559 | 
1560 | 
1561 | 
1562 | 
1563 | 
1564 | 
1565 | 
1566 | 
1567 | 
1568 | 
1569 | 
1569 | 
1570 | 
1571 | 
1572 | 
1573 | 
1574 | 
1575 | 
1576 | 
1577 | 
1577 | 
1578 | 
1579 | 
1580 | 
1581 | 
1582 | 
1583 | 
1584 | 
1585 | 
1586 | 
1587 | 
1587 | 
1588 | 
1589 | 
1590 | 
1591 | 
1592 | 
1593 | 
1594 | 
1594 | 
1595 | 
1596 | 
1597 | 
1598 | 
1599 | 
1599 | 
1600 | 
1601 | 
1602 | 
1603 | 
1604 | 
1605 | 
1606 | 
1607 | 
1608 | 
1609 | 
1609 | 
1610 | 
1611 | 
1612 | 
1613 | 
1614 | 
1615 | 
1616 | 
1617 | 
1618 | 
1618 | 
1619 | 
1620 | 
1621 | 
1622 | 
1623 | 
1624 | 
1625 | 
1626 | 
1627 | 
1628 | 
1628 | 
1629 | 
1630 | 
1631 | 
1632 | 
1633 | 
1634 | 
1635 | 
1636 | 
1637 | 
1638 | 
1638 | 
1639 | 
1640 | 
1641 | 
1642 | 
1643 | 
1644 | 
1645 | 
1646 | 
1647 | 
1648 | 
1648 | 
1649 | 
1650 | 
1651 | 
1652 | 
1653 | 
1654 | 
1655 | 
1656 | 
1657 | 
1658 | 
1658 | 
1659 | 
1660 | 
1661 | 
1662 | 
1663 | 
1664 | 
1665 | 
1666 | 
1667 | 
1668 | 
1669 | 
1669 | 
1670 | 
1671 | 
1672 | 
1673 | 
1674 | 
1675 | 
1676 | 
1677 | 
1677 | 
1678 | 
1679 | 
1680 | 
1681 | 
1682 | 
1683 | 
1684 | 
1685 | 
1686 | 
1687 | 
1687 | 
1688 | 
1689 | 
1690 | 
1691 | 
1692 | 
1693 | 
1694 | 
1694 | 
1695 | 
1696 | 
1697 | 
1698 | 
1699 | 
1699 | 
1700 | 
1701 | 
1702 | 
1703 | 
1704 | 
1705 | 
1706 | 
1707 | 
1708 | 
1709 | 
1709 | 
1710 | 
1711 | 
1712 | 
1713 | 
1714 | 
1715 | 
1716 | 
1717 | 
1718 | 
1718 | 
1719 | 
1720 | 
1721 | 
1722 | 
1723 | 
1724 | 
1725 | 
1726 | 
1727 | 
1728 | 
1728 | 
1729 | 
1730 | 
1731 | 
1732 | 
1733 | 
1734 | 
1735 | 
1736 | 
1737 | 
1738 | 
1738 | 
1739 | 
1740 | 
1741 | 
1742 | 
1743 | 
1744 | 
1745 | 
1746 | 
1747 | 
1748 | 
1748 | 
1749 | 
1750 | 
1751 | 
1752 | 
1753 | 
1754 | 
1755 | 
1756 | 
1757 | 
1758 | 
1758 | 
1759 | 
1760 | 
1761 | 
1762 | 
1763 | 
1764 | 
1765 | 
1766 | 
1767 | 
1768 | 
1769 | 
1769 | 
1770 | 
1771 | 
1772 | 
1773 | 
1774 | 
1775 | 
1776 | 
1777 | 
1778 | 
1779 | 
1779 | 
1780 | 
1781 | 
1782 | 
1783 | 
1784 | 
1785 | 
1786 | 
1787 | 
1787 | 
1788 | 
1789 | 
1790 | 
1791 | 
1792 | 
1793 | 
1794 | 
1794 | 
1795 | 
1796 | 
1797 | 
1798 | 
1799 | 
1799 | 
1800 | 
1801 | 
1802 | 
1803 | 
1804 | 
1805 | 
1806 | 
1807 | 
1808 | 
1809 | 
1809 | 
1810 | 
1811 | 
1812 | 
1813 | 
1814 | 
1815 | 
1816 | 
1817 | 
1818 | 
1818 | 
1819 | 
1820 | 
1821 | 
1822 | 
1823 | 
1824 | 
1825 | 
1826 | 
1827 | 
1
```

```

1  <- (design.action-grounding ?action-designator (place ?current-object-designator ?arm
2    ?left-reach-poses ?right-reach-poses
3    ?left-put-poses ?right-put-poses
4    ?left-retract-poses ?right-retract-poses
5    ?location))
6  (spec property ?action-designator (type :placing))
7
8
9
10
11  (> (spec property ?action-designator (arm ?arm))
12    (> (spec property ?action-designator (?object ?object-designator))
13      (or (cpoe object-in-hand ?object-designator ?arm)
14        (and (format "WARNING: Wanted to place an object ~a with arm ~a, ~
15          but it's not in the arm.~%" ?object-designator ?arm)
16        ))
17      )
18    )
19    (> (spec property ?action-designator (?object ?object-designator))
20      (spec property ?action-designator (?object ?object-designator ?arm)
21        (and (cram-robot-interfaces:robot ?robot)
22          (cram-robot-interfaces:arm ?robot ?arm)
23          (cpoe object-in-hand ?object-designator ?arm)
24          (cram-robot-interfaces:arm ?robot ?arm)))
25        )
26      )
27      (once (or (cpoe object-in-hand ?object-designator ?arm)
28        (spec property ?action-designator (?object ?object-designator)))
29        (spec property ?current-object-designator (type ?object-type))
30        (spec property ?current-object-designator (name ?object-name))
31        (obj-int:object-type-grasp ?object-type ?grasp)
32        (> (spec property ?action-designator (?target ?location))
33          (and (cram-robot-interfaces:location ?location ?current-location-designator)
34            (design-designator-grounding ?current-object-designator ?target ?location)
35            (member ?target ?poses)
36            (symbol-value cram-if:robot-base-frame ?base-frame)
37            (lisp-fun cram-if:ensure-pose-in-frame ?target-pose ?base-frame :use-zero-time t
38              ?target-pose-in-base)
39            (lisp-fun roslib-unities:rosify-underscores-lisp-name ?object-name ?if-name)
40            (lisp-fun cram-if:pose-stamped->transform-stamped ?target-pose-in-base ?if-name
41              ?target-transform)
42            (and (lisp-fun obj-int:get-object-transform ?current-object-designator ?target-transform)
43              (lisp-fun obj-int:get-object-pose ?current-object-designator ?target-pose)
44              (design-designator :location ((cpose ?target-pose) ?location)))
45              (lisp-fun obj-int:get-object-grasping-poses
46                ?object-name ?object-type :left ?grasp ?target-transform
47                ?left-poses)
48              (lisp-fun extract:place-manipulation-poses ?arm ?left-poses ?right-poses
49                ?left-reach-poses ?right-reach-poses ?left-put-poses ?right-put-poses
50                ?left-retract-poses ?right-retract-poses)))
51
52
53

```



```

1  <- (design.action-grounding ?action-designator (place-down ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53

```

Figure 47: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Slicing* (27 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
21	40	4	2

Findings:

- designator definition contains only one instead of nine variables
- designator sets high focus on definitions relevant for calculating trajectories / poses instead of using them
- designator (falsely) ends by mapping the variables to designator "parts"

```

1  | <-(design-action-grounding ?action-designator (place ?current-object-designator ?arm
2  |   ?left-reach-poses ?right-reach-poses
3  |   ?left-put-poses ?right-put-poses
4  |   ?left-retract-poses ?right-retract-poses
5  |   ?location))
6  | 
7  |   (spec:property ?action-designator (:type :placing))
8  | 
9  |   (spec:property ?action-designator (:arm ?arm))
10 | 
11 |   (spec:property ?action-designator (:object ?object-designator))
12 | 
13 |   (or (cpeo:object-in-hand ?object-designator ?arm)
14 |       (and (format WARNING: Wanted to place an object ~a with arm ~a, ~
15 |             but it's not in the arm. ~% ?object-designator ?arm)
16 |           ))
17 | 
18 |   (cpeo:object-in-hand ?object-designator ?object ?object-designator)
19 | 
20 |   (cpeo:object-in-hand ?object-designator ?object)
21 | 
22 |   (and (cram-robot-interfaces:robot ?robot))
23 | 
24 |   (cram-robot-interfaces:arm ?robot ?arm)
25 | 
26 |   (cpeo:object-in-hand ?object-designator ?arm))
27 | 
28 |   (once (spec:property ?action-designator (:type ?object-type)))
29 | 
30 |   (spec:property ?current-object-designator ?current-object-designator)
31 | 
32 |   (spec:property ?current-object-designator (:name ?object-name))
33 | 
34 |   (obj-int:object-type:grasp ?object-type ?grasp)
35 | 
36 |   (spec:property ?action-designator (:target ?location))
37 | 
38 |   (and (desig:current-designator ?location ?current-location-designator)
39 |         (desig:current-designator ?location ?current-location-designator)
40 |         (member ?target-poses ?poses)
41 |         (symbol-value cram-tf::"robot-base-frame" ?base-frame)
42 |         (lisp-fun cram-tf:ensure-pose-in-frame ?target-poses ?base-frame :use-zero-time t
43 |             ?target-pose-in-base)
44 |         (lisp-fun roslib-utilities:rosify-underscores-lisp-name ?object-name ?tf-name)
45 |         (lisp-fun cram-tf:pose-stamped->transform-stamped ?target-pose-in-base ?tf-name
46 |             ?target-transform)
47 |         (and (lisp-fun obj-int:get-object-transform ?current-object-designator ?target-transform)
48 |             (lisp-fun obj-int:get-object-pose ?current-object-designator ?target-pose)
49 |             (desig:current-designator ?location ((?poses ?target-poses) ?location)))
50 |             (lisp-fun obj-int:get-object-grasping-poses
51 |                 ?object-name ?object-type ?left ?grasp ?target-transform
52 |                 ?left-poses)
53 |             (lisp-fun obj-int:get-object-grasping-poses
54 |                 ?object-name ?object-type ?right ?grasp ?target-transform
55 |                 ?right-poses)
56 |             (lisp-fun extract-place-manipulation-poses ?arm ?left-poses ?right-poses
57 |                 (?left-reach-poses ?right-reach-poses ?left-put-poses ?right-put-poses
58 |                  ?left-retract-poses ?right-retract-poses)))))
59 | 
60 | 
61 | 
62 | 
63 | 
64 | 
65 | 
66 | 
67 | 
68 | 
69 | 
70 | 
71 | 
72 | 
73 | 
74 | 
75 | 
76 | 
77 | 
78 | 
79 | 
80 | 
81 | 
82 | 
83 | 
84 | 
85 | 
86 | 
87 | 
88 | 
89 | 
90 | 
91 | 
92 | 
93 | 
94 | 
95 | 
96 | 
97 | 
98 | 
99 | 
100 | 
101 | 
102 | 
103 | 
104 | 
105 | 
106 | 
107 | 
108 | 
109 | 
110 | 
111 | 
112 | 
113 | 
114 | 
115 | 
116 | 
117 | 
118 | 
119 | 
120 | 
121 | 
122 | 
123 | 
124 | 
125 | 
126 | 
127 | 
128 | 
129 | 
130 | 
131 | 
132 | 
133 | 
134 | 
135 | 
136 | 
137 | 
138 | 
139 | 
140 | 
141 | 
142 | 
143 | 
144 | 
145 | 
146 | 

```

Figure 48: Left: Manual designator for the action *Placing Down* (46 lines). Right: Generated designator based on *Wiping* (13 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
1	34	11	1

Findings:

- designator definition contains only one instead of nine variables
- designator contains only (some) definitions and no usage / calculations
- designator (falsely) ends by mapping the variables to designator "parts"

```

1  (-> (design.action-grounding ?action-designator (pour ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85

```

(spec:property ?action-designator (:type ?pouring))
(spec:property ?action-designator ?object ?source-object-designator)
(spec:current-designator ?object-designator ?current-object-desig)
(spec:property ?current-object-desig (:type ?object-type))
(-> (spec:property ?action-designator (:arms ?arms))
(when
 (and (man-intro-trust-free-hand ?_ ?arm)
 (equal ?arms (?arm)))
 (lisp-fun man-int-get-object-transform ?current-object-desig ?object-transform)
 (lisp-fun man-int-calculate-object-faces ?object-transform (facing-robot-face ?bottom-face))
 (-> (spec:property ?current-object-desig (:symmetric ?object-type))
 (equal ?rotationally-symmetric nil))
 (-> (spec:property ?action-designator (:grasp ?grasp))
 (true))
 (and (member ?arm ?arms)
 (lisp-fun man-int-get-action-grasps ?object-type ?arm ?object-transform ?grasps)
 (member ?grasp ?grasps))
 (lisp-fun man-int-get-action-gripping-effort ?object-type ?effort)
 (lisp-fun man-int-get-action-gripper-opening ?object-type ?gripper-opening)
 (equal ?object ?current-object-desig))
 (-> (spec:property ?target ?arm))
 (and (lisp-fun man-int-get-action-trajectory :pouring :left ?grasp T ?objects
 ?left-pouring-pose)
 (lisp-fun man-int-get-traj-poses-by-label ?left-pouring-pose :approach
 ?right-approach-poses)
 (lisp-fun man-int-get-traj-poses-by-label ?left-pouring-pose :tilting
 ?right-tilt-poses))
 (and (equal ?left-approach-poses NIL))
 (equal ?left-tilt-poses NIL))
 (-> (spec:property ?target ?right))
 (and (lisp-fun man-int-get-action-trajectory :pouring :right ?grasp T ?objects
 ?right-pouring-pose)
 (lisp-fun man-int-get-traj-poses-by-label ?right-pouring-pose :approach
 ?right-approach-poses)
 (lisp-fun man-int-get-traj-poses-by-label ?right-pouring-pose :tilting
 ?right-tilt-poses))
 (and (equal ?right-approach-poses NIL))
 (equal ?right-tilt-poses NIL))
 (-> (spec:desig-prop ?action-designator (:collision-mode ?collision-mode))
 (true))
 (equal ?collision-mode nil))
 (design:designator ?action :pouring)
 (:object ?current-object-desig)
 (:object-type ?object-type))
 (-> (spec:property ?object-name ?object-name)
 (?object-name ?object-name)
 (?arms ?arms)
 (?grasp ?grasp)
 (?left-approach-poses ?left-approach-poses)
 (?right-approach-poses ?right-approach-poses)
 (?left-tilt-poses ?left-tilt-poses)
 (?right-tilt-poses ?right-tilt-poses)
 (?collision-mode ?collision-mode))
 (?resolved-action-designator))


```

1  (-> (design.action-grounding ?action-designator (pour ?arm))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85

```

(spec:property ?action-designator (:type ?pouring))
(spec:property ?action-designator ?target ?target-container-designator)
(spec:property ?source-container-designator (:type ?source-container-type))
(spec:property ?object-type-subtype ?container ?source-container-type)
(spec:property ?source-container-designator (:urdf-name ?source-container-name))
(spec:property ?source-container-designator (:part-of ?btr-environment))
(spec:property ?target-container-designator (:type ?target-container-type))
(spec:property ?target-container-designator (?name ?target-container-name))
(spec:property ?target-container-designator (:part-of ?btr-environment))
(-> (spec:property ?action-designator (:arm ?arm))
(true))
(and (cram-robot-interfaces robot ?robot))
(cram-robot-interfaces arm ?robot ?arm))
(lisp-fun obj-int-get-object-type-gripper-opening ?source-container-type ?gripper-opening)
(spec:property ?action-designator (:distance ?distance))
(lisp-fun get-container-link ?source-container-name ?btr-environment ?source-container-link)
(lisp-fun get-container-joint ?source-container-link ?connecting-joint)
(lisp-fun el-strict-name ?connecting-joint ?part-name))
(btr-bullet-world ?world)
(lisp-fun bl-object ?world ?btr-environment ?environment-obj)
(lisp-fun get-container-pose-and-transform ?target-container-name ?btr-environment
?target-container-pose-and-transform ?target-container-transform)
(lisp-fun get-container-pose-and-transform ?target-container-name ?btr-environment
?target-container-pose ?target-container-transform)
(lisp-fun obj-int-get-object-grasping-pose ?source-container-name
:container-prismatic :left :close ?source-container-transform ?left-poses)
(lisp-fun obj-int-get-object-grasping-pose ?source-container-name
:container-prismatic :right :close ?source-container-transform ?right-poses)
(lisp-fun cram-mobile-pick-place-plans :extract-pick-up-manipulation-poses
?arm ?left-poses ?right-poses
?left-reach-poses ?right-reach-poses
?left-grasp-poses ?right-grasp-poses
?left-tilt-poses ?right-tilt-poses))
(-> (lisp-pred identity ?left-tilt-poses)
(equal ?left-tilt-poses ?left-tilt-poses ?left-2nd-lft-tilt-poses))
(equal (NIL NIL) ?left-tilt-poses ?left-2nd-lft-tilt-poses))
(-> (lisp-pred identity ?right-tilt-poses)
(equal ?right-tilt-poses ?right-tilt-poses ?right-2nd-lft-tilt-poses))
(equal (NIL NIL) ?right-tilt-poses ?right-2nd-lft-tilt-poses)))

Figure 49: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Closing* (56 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
48	48	6	2

Findings:

- designator definition contains 15 instead of only one variable
- designator calculates trajectories and poses in a different manner
- generation adds block about **cram-robot-interfaces** (2 lines)
- designator does not end by mapping the variables to designator "parts"

```

1  <- (desig:action-grounding ?action-designator (pour ?resolved-action-designator))
2   (spec:property ?action-designator :type ?pouring)
3   (spec:property ?action-designator :object ?object-designator)
4   (spec:current-designator ?object-designator ?current-object-desig)
5   (spec:property ?current-object-desig :type ?object-type)
6   (spec:property ?current-object-desig :name ?object-name)
7
8   (> (spec:property ?action-designator :arms ?arms)
9    (true)
10   (and (man:int:robot-free-hand ?_ ?arm)
11     (equal ?arms ?arm)))
12
13   (lisp-fun-man-int-set-object-transform ?current-object-designator ?object-transform)
14   (lisp-fun-man-int-set-object-faces ?object-transform ?facing-robot-face ?bottom-face)
15
16   (> (man:int:object-rotationally-symmetric ?object-type)
17     (equal ?rotationally-symmetric nil))
18
19   (> (spec:property ?action-designator (grasp ?grasp)
20    (true)
21   (and (member ?arm ?arms)
22     (lisp-fun-man-int-get-action-grasps ?object-type ?arm ?object-transform ?grasps)
23     (member ?grasp ?grasps)))
24
25
26
27
28   (lisp-fun-man-int-get-action-gripping-effort ?object-type ?effort)
29   (lisp-fun-man-int-get-action-gripper-opening ?object-type ?gripper-opening)
30   (equal ?object-type ?current-object-designator)
31
32   (> (member ?left ?arms)
33     (and (lisp-fun-man-int-get-action-trajectory :pouring ?left ?grasp T ?objects
34       ?left-pouring-pose)
35       (lisp-fun-man-int-get-traj-poses-by-label ?left-pouring-pose :approach
36         ?left-approach-poses)
37       (lisp-fun-man-int-get-traj-poses-by-label ?left-pouring-pose :tilting
38         ?left-tilt-poses)
39       (and (equal ?left-tilt-poses NIL)
40         (equal ?left-tilt-poses NIL)))
41
42   (> (member ?right ?arms)
43     (and (lisp-fun-man-int-get-action-trajectory :pouring ?right ?grasp T ?objects
44       ?right-pouring-pose)
45       (lisp-fun-man-int-get-traj-poses-by-label ?right-pouring-pose :approach
46         ?right-approach-poses)
47       (lisp-fun-man-int-get-traj-poses-by-label ?right-pouring-pose :tilting
48         ?right-tilt-poses)
49
50   (> (desig:design-prop ?action-designator (:collision-mode ?collision-mode))
51     (true)
52     (equal ?collision-mode nil))
53
54   (desig:designator ?action :type ?pouring)
55   (desig:designator ?action :object ?current-object-designator)
56   (desig:designator ?action :object-type ?object-type)
57
58   (desig:designator ?action :object-name ?object-name)
59   (desig:designator ?action :arms ?arms)
60
61   (desig:designator ?action :grasp ?grasp)
62   (desig:designator ?action :left-approach-poses ?left-approach-poses)
63   (desig:designator ?action :right-approach-poses ?right-approach-poses)
64   (desig:designator ?action :left-tilt-poses ?left-tilt-poses)
65   (desig:designator ?action :right-tilt-poses ?right-tilt-poses)
66
67   (:collision-mode ?collision-mode)
68
69   ?resolved-action-designator]

```



```

1  <- (desig:action-grounding ?action-designator (pour ?resolved-action-designator))
2   (spec:property ?action-designator :type ?pouring)
3   (spec:property ?action-designator :source ?source-designator)
4   (spec:property ?action-designator :destination ?destination-designator)
5   (spec:current-designator ?source-designator ?current-source-desig)
6   (spec:current-designator ?destination-designator ?current-destination-desig)
7
8   (spec:property ?current-source-desig :type ?source-type)
9   (spec:property ?current-source-desig :name ?source-name)
10  (spec:property ?current-destination-desig :type ?destination-type)
11  (spec:property ?current-destination-desig :name ?destination-name)
12
13  (> (spec:property ?action-designator (?arm ?arm))
14    (true)
15    (and (man:int:robot-free-hand ?_ ?arm)
16      (lisp-fun-man-int-get-object-old-transform ?current-source-designator ?source-transform)
17      (lisp-fun-man-int-get-object-old-transform ?current-destination-designator ?destination-transform)
18      (lisp-fun-man-int-calculate-object-faces ?source-transform ?source-top-face ?source-bottom-face)
19      (lisp-fun-man-int-calculate-object-faces ?destination-transform ?destination-top-face ?destination-bottom-face)))
20
21
22  (> (spec:property ?action-designator (grasp ?grasp)
23    (true)
24    (and (lisp-fun-man-int-get-action-grasps ?source-type ?arm ?source-transform ?grasps)
25      (member ?grasp ?grasps)))
26
27  (> (spec:property ?action-designator (:pouring-pose ?pouring-pose))
28    (true)
29    (formal "Please infer the pouring pose, or use the query system to infer it here")
30    (lisp-fun-man-int-get-action-gripping-effort ?source-type ?effort)
31    (lisp-fun-man-int-get-action-gripper-opening ?source-type ?gripper-opening)
32    (equal ?objects ?current-source-designator ?current-destination-designator)
33
34  (> (equal ?left ?arm)
35    (and (lisp-fun-man-int-get-traj-poses-by-label ?left-pouring-pose :pouring-up
36      ?left-pouring-up-poses)
37      (lisp-fun-man-int-get-traj-poses-by-label ?left-pouring-pose :pouring-down
38        ?left-pouring-down-poses)
39      (and (equal ?left-pouring-up-poses NIL)
40        (equal ?left-pouring-down-poses NIL)))
41
42  (> (equal ?right ?arm)
43    (and (lisp-fun-man-int-get-traj-poses-by-label ?right-pouring-pose :pouring-up
44      ?right-pouring-up-poses)
45      (lisp-fun-man-int-get-traj-poses-by-label ?right-pouring-pose :pouring-down
46        ?right-pouring-down-poses)
47      (and (equal ?right-pouring-up-poses NIL)
48        (equal ?right-pouring-down-poses NIL)))
49
50  (> (desig:design-prop ?action-designator (:collision-mode ?collision-mode))
51    (true)
52    (equal ?collision-mode nil))
53
54  (desig:designator ?action :type ?pouring)
55  (desig:designator ?action :object ?current-object-designator)
56  (desig:designator ?action :source-name ?source-name)
57  (desig:designator ?action :destination-name ?destination-name)
58
59  (desig:designator ?action :arm ?arm)
60  (desig:designator ?action :gripper-opening ?gripper-opening)
61  (desig:designator ?action :effort ?effort)
62  (desig:designator ?action :grasp ?grasp)
63
64  (desig:designator ?action :left-pouring-up-poses ?left-pouring-up-poses)
65  (desig:designator ?action :right-pouring-up-poses ?right-pouring-up-poses)
66  (desig:designator ?action :left-pouring-down-poses ?left-pouring-down-poses)
67  (desig:designator ?action :right-pouring-down-poses ?right-pouring-down-poses)
68
69  (:collision-mode ?collision-mode)
70
71  ?resolved-action-designator]

```

Figure 50: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Halving* (63 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
14	7	34	15

Findings:

- some minor renaming (e.g. *source* instead of *object*)
- designator calculates a specific *pouring-pose*
- generated designator extends the mapping of variables to designator "parts" (misses *:object-type* and *:arms* but adds *:source-name*, *:destination*, *:gripper-opening*, *:effort* and *:arm*)

```
(design-action-grounding ?action-designator ?our ?resolved-action-designator)
(spec-property ?action-designator :type ?pouring)
(spec-property ?action-designator :object ?object-designator)
(spec-current-designator ?object-designator ?current-object-design)
(spec-property ?current-object-desig :type ?object-type)
(spec-property ?current-object-desig :name ?object-name)

(> (spec-property ?action-designator :arms ?arms))
(true)
(and man-int-robot-free-hand ?_ ?arm)
(equal ?arms ?arm)))
(lisp-fun man-int-get-object-transform ?current-object-desig ?object-transform)
(lisp-fun man-int-calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
(> (man-int-object-rotationally-symmetric ?object-type)
(equal ?rotationally-symmetric t)
(equal ?rotationally-symmetric NIL)))
(> (spec-property ?action-designator :grasp ?grasp))
(true)
(and (member ?arm ?arms)
(lisp-fun man-int-action-grasps ?object-type ?arm ?object-transform ?grasps)
(equal ?grasp ?grasp)))
(lisp-fun man-int-action-grapping-exp? ?object-type ?effector)
(lisp-fun man-int-get-action-gripper-opening ?object-type ?gripper-opening)
(equal ?objects (?current-object-design))
(> (member ?left ?arms)
(and (lisp-fun man-int-get-action-trajectory :pouring :left) ?grasp T ?objects
(left-pouring-pose)
(lisp-fun man-int-get-traj-poses-by-label ?left-pouring-pose :approach
:left-approach-poses)
(lisp-fun man-int-get-traj-poses-by-label ?left-pouring-pose :tilting
:left-tilt-poses)
(and (equal ?left-approach-poses NIL)
(equal ?left-tilt-poses NIL)))
(> (member ?right ?arms)
(and (lisp-fun man-int-get-action-trajectory :pouring :right) ?grasp T ?objects
(right-pouring-pose)
(lisp-fun man-int-get-traj-poses-by-label ?right-pouring-pose :approach
:right-approach-poses)
(lisp-fun man-int-get-traj-poses-by-label ?right-pouring-pose :tilting
:right-tilt-poses))
(and (equal ?right-approach-poses NIL)
(equal ?right-tilt-poses NIL)))
(> (design-prop ?action-designator :collision-mode ?collision-mode))
(true)
(equal ?collision-mode NIL))

(design-designator :action ((:type :pouring)
(:object ?current-object-design)
(:object-type ?object-type)
(:object-name ?object-name)
(:arms ?arms)
(:grasp ?grasp)
(:left-approach-poses ?left-approach-poses)
(:right-approach-poses ?right-approach-poses)
(:left-tilt-poses ?left-tilt-poses)
(:right-tilt-poses ?right-tilt-poses)
(:collision-mode ?collision-mode)
?resolved-action-designator))

1 (< (design-action-grounding ?np-plans ?our ?resolved-action-designator))
(spec-property ?action-designator :type ?pouring)
(spec-property ?action-designator :source ?source-designator)
(spec-property ?action-designator :target ?target-designator)
(spec-current-designator ?source-designator ?current-source-design)
(spec-current-designator ?target-designator ?current-target-design)
(spec-property ?current-source-desig :name ?source-name)
(spec-property ?current-target-desig :type ?target-type)
(spec-property ?current-target-desig :name ?target-name)
(lisp-fun man-int-get-object-transform ?current-source-desig ?source-transform)
(lisp-fun man-int-get-object-transform ?current-target-desig ?target-transform)
(lisp-fun man-int-calculate-object-faces ?source-transform ?source-top-face ?source-bottom-face)
(lisp-fun man-int-calculate-object-faces ?target-transform ?target-top-face ?target-bottom-face))
(> (equal ?source-type ?target-type)
(equal ?source-name ?target-name))
(> (equal ?source-top-face ?target-top-face)
(equal ?source-bottom-face ?target-bottom-face)))
(> (spec-property ?action-designator :arm ?arm))
(true)
(man-int-robot-free-hand ?_ ?arm))

(> (spec-property ?action-designator :pour-direction ?pour-direction))
(member ?pour-direction (:into :out-of))
(lisp-fun man-int-get-action-pour-trajectory ?source-type ?pour-direction ?arm ?source-transform ?target-transform ?trajectory)
(spec-current-designator ?source-designator ?current-source-design)
(spec-current-designator ?target-designator ?current-target-design))

40 (> (lisp-fun man-int-get-traj-poses-by-label ?trajectory :reaching ?reach-poses)
41 (lisp-fun man-int-get-traj-poses-by-label ?trajectory :pouring ?pour-poses)
42 (equal ?objects (?current-source-design ?current-target-design)))

57 (design-designator :action ((:type :pouring)
58 (:source ?current-source-design)
59 (:target ?current-target-design)
60 (:target-name ?target-name)
61 (:arm ?arm)
62 (:pour-direction ?pour-direction)
63 (:reach-poses ?reach-poses)
64 (:pour-poses ?pour-poses)
65 (:collision-mode ?collision-mode)
66 ?resolved-action-designator))
```

Figure 51: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Holding* (37 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
18	37	15	4

Findings:

- designator sets high focus on definitions relevant for calculating trajectories / poses instead of using them
 - generated designator shortens the mapping of variables to designator "parts" (misses `:object-type`, `:grasp`, `:collision-mode`, `:right-approach-poses` and `:arms` but adds `:target`, `:target-name` and `:pour-direction`)

```

1  :-(< (design:action-grounding ?action-designator (pour ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15   (spec:property ?action-designator (:type :pouring))
16   (spec:property ?action-designator (:object ?object-designator))
17   (design:current-designator ?object-designator ?current-object-desig)
18   (spec:property ?current-object-desig (:type ?object-type))
19
20
21
22
23
24   (spec:property ?current-object-desig (:name ?object-name))
25
26   :-(> (spec:property ?action-designator (:arms ?arms))
27
28     (true)
29     (and (man-int:robot-free-hand ?_ ?arm)
30       (equal ?arms ?arm)))
31     (lisp-fun man-int:get-object-transform ?current-object-desig ?object-transform)
32     (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
33     (> (man-int:object-rotationally-symmetric ?object-type)
34       (equal ?rotationally-symmetric nil))
35     (> (spec:property ?action-designator (:grasp ?grasp))
36       (true)
37       (and (member ?arm ?arms)
38         (+ (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
39             (member ?grasp ?grasps)))
40         (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
41         (+ (lisp-fun obj-int:get-object-gripper-opening ?object-type ?gripper-opening)
42             (equal ?objects ?current-object-desig))
43         (> (member :left ?arms)
44           (and (lisp-fun man-int:get-action-trajectory :pouring :left ?grasp T ?objects
45             ?left-pouring-pose)
46             (lisp-fun man-int:get-traj-poses-by-label ?left-pouring-pose :approach
47               ?left-approach-poses)
48             (lisp-fun man-int:get-traj-poses-by-label ?left-pouring-pose :tilting
49               ?left-tilt-poses)
50             (and (equal ?left-approach-poses NIL)
51               (equal ?left-tilt-poses NIL)))
52         (> (member :right ?arms)
53           (and (lisp-fun man-int:get-action-trajectory :pouring :right ?grasp T ?objects
54             ?right-pouring-pose)
55             (lisp-fun man-int:get-traj-poses-by-label ?right-pouring-pose :approach
56               ?right-approach-poses)
57             (lisp-fun man-int:get-traj-poses-by-label ?right-pouring-pose :tilting
58               ?right-tilt-poses)
59             (and (equal ?right-approach-poses NIL)
60               (equal ?right-tilt-poses NIL)))
61         (> (design:desig-prop ?action-designator (:collision-mode ?collision-mode))
62           (true)
63           (equal ?collision-mode nil))
64         (design:designator ?action (:type :pouring)
65           (:object ?current-object-desig)
66           (:object-type ?object-type)
67           (:object-name ?object-name)
68           (:arms ?arms)
69           (:grasp ?grasp)
70           (:left-approach-poses ?left-approach-poses)
71           (:right-approach-poses ?right-approach-poses)
72           (:left-tilt-poses ?left-tilt-poses)
73           (:right-tilt-poses ?right-tilt-poses)
74           (:collision-mode ?collision-mode))
75           ?resolved-action-designator)))
1  :-(< (design:action-grounding ?action-designator [pour ?arm]
2
3
4
5
6
7
8
9
10
11
12
13
14
15   (spec:property ?action-designator (:type :pouring))
16   (spec:property ?action-designator (:source ?source-container-designator))
17   (spec:property ?action-designator (:target ?target-container-designator))
18   (spec:property ?source-container-designator (:type ?source-container-type))
19   (spec:property ?source-container-designator (?part-of ?btb-environment))
20   (spec:property ?source-container-designator (?part-of ?btb-environment))
21   (spec:property ?target-container-designator (?part-of ?btb-environment))
22   (spec:property ?target-container-designator (?part-of ?btb-environment))
23   (+ (spec:property ?action-designator (:arm ?arm))
24     (true)
25     (and (cram-robot-interfaces:robot ?robot)
26       (cram-robot-interfaces:arm ?robot ?arm)))
27     (spec:property ?action-designator (:distance ?distance))
28     (lisp-fun get-container-link ?source-container-name ?btb-environment ?source-container-link)
29     (lisp-fun get-connecting-joint ?source-container-link ?connecting-joint)
30     (lisp-fun cl-urdf:link-name ?connecting-joint ?joint-name)
31     (bt:bullet-world ?world)
32
33
34
35
36
37
38   (+ (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
39
40
41   (+ (lisp-fun obj-int:get-object-type-gripper-opening ?source-container-type ?gripper-opening)
42     (lisp-fun get-container-pose-and-transform ?source-container-name ?btr-environment
43       (?source-container-pose ?source-container-transform)))
44
45
46
47
48   (+ (lisp-fun obj-int:get-object-grasping-poses ?source-container-name
49     (:container-prismatic :left open ?source-container-transform ?left-poses)
50     (lisp-fun obj-int:get-object-grasping-poses ?source-container-name
51       (:container-prismatic :right open ?source-container-transform ?right-poses)
52       (lisp-fun cram-mobile:pick-place-plans::extract-pick-up-manipulation-poses
53
54
55
56   (+ ?arm ?left-poses ?right-poses
57     (?left-reach-poses ?right-reach-poses
58     ?left-grasp-poses ?right-grasp-poses)
59
60
61   (+ (lisp-pred identity ?left-lift-poses ?right-lift-poses))
62     (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose))
63     (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
64
65
66
67
68
69
70
71   (+ (lisp-pred identity ?right-lift-poses)
72     (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose))
73     (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose)))
74
75

```

Figure 52: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Opening* (53 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
40	43	11	2

Findings:

- designator definition contains 15 instead of only one variable
- designator calculates trajectories and poses in a different manner
- some minor renaming (e.g. *source* instead of *object*)
- generation adds block about **cram-robot-interfaces** (2 lines)
- designator does not end by mapping the variables to designator "parts"

```

1  :- (desig:action-grounding ?action-designator (pour ?resolved-action-designator))
2   (spec:property ?action-designator (:type :pouring))
3   (spec:property ?action-designator (?object ?object-designator))
4   (desig:current-designator ?object-designator ?current-object-desig)
5   (spec:property ?current-object-designator (:type ?object-type))
6   (spec:property ?current-object-designator (:name ?object-name))
7   (spec:property ?action-designator (arms ?arms))
8
9
10  (lisp-fun man-int:robot-free-hand ?_ ?arm)
11  (and (man-int:robot-free-hand ?_ ?arm)
12    (equal ?arms (2arm)))
13  (lisp-fun man-int:get-object-transform ?current-object-designator ?object-transform)
14  (lisp-fun man-int:calculate-object-faces ?object-transform ?facing:robot-face ?bottom-face)
15  (lisp-fun man-int:object-rotationally-symmetric ?object-face)
16  (equal ?rotationally-symmetric t)
17  (spec:property ?action-designator (grasp ?grasp))
18  (true)
19  (and (member ?arm ?arms)
20    (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
21    (member ?grasp ?grasps)))
22  (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
23  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
24  (equal ?objects ?current-object-designator)
25  (lisp-fun man-int:member ?left ?arms)
26  (and (lisp-fun man-int:get-action-trajectory :pouring :left ?grasp T ?objects
27    ?left-pouring-poses)
28    (lisp-fun man-int:get-traj-poses-by-label ?left-pouring-poses :approach
29    ?left-approach-poses)
30    (lisp-fun man-int:get-traj-poses-by-label ?left-pouring-poses :tilting
31    ?left-tilt-poses))
32  (and (equal ?left-approach-poses NIL)
33    (equal ?left-tilt-poses NIL)))
34  (lisp-fun man-int:member ?right ?arms)
35  (and (lisp-fun man-int:get-action-trajectory :pouring :right ?grasp T ?objects
36    ?right-pouring-poses)
37    (lisp-fun man-int:get-traj-poses-by-label ?right-pouring-poses :approach
38    ?right-approach-poses)
39    (lisp-fun man-int:get-traj-poses-by-label ?right-pouring-poses :tilting
40    ?right-tilt-poses))
41  (and (equal ?right-approach-poses NIL)
42    (equal ?right-tilt-poses NIL)))
43  (-> (desig:design-prop action-designator (:collision-mode ?collision-mode)))
44  (true)
45  (equal ?collision-mode nil)
46  (desig:designator action ((:type :pouring)
47    (:object ?current-object-designator)
48    (:object-type ?object-type)
49    (:object-name ?object-name)
50    (:arms ?arms)
51    (:grasp ?grasp)
52    (:left-approach-poses ?left-approach-poses)
53    (:right-approach-poses ?right-approach-poses)
54    (:left-tilt-poses ?left-tilt-poses)
55    (:right-tilt-poses ?right-tilt-poses)
56    (:collision-mode ?collision-mode)))
57
58  ?resolved-action-designator)

```



```

1  :- (desig:action-grounding ?action-designator (pour ?source-container ?target-container))
2   (spec:property ?action-designator (:type :pouring))
3   (spec:property ?action-designator (?source ?source-container))
4
5
6
7
8
9
10
11  (spec:property ?action-designator (?target ?target-container))
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

Figure 53: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Picking Up* (16 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
8	48	7	1

Findings:

- tba

```

1  | <- (desig.action-grounding ?action-designator (pour ?resolved-action-designator))
2  |   (spec property ?action-designator (:type :pouring))
3  |   (spec property ?action-designator (object ?object-designator))
4  |   (desig current-designator ?object-designator ?current-object-desig)
5  |   (spec property ?current-object-desig (:type ?object-type))
6  |   (spec property ?current-object-desig (:name ?object-name))
7  |   (-> (spec:property ?action-designator (:arms ?arms)))
8  |     (true)
9  |     (and (man-int:robot-free-hand ?_ ?arm)
10 |       (equal ?armc ?arm)))
11 |     (lisp-fun man-int:get-object-transform ?current-object-desig ?object-transform)
12 |     (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
13 |     (-> (man-int:object-rotationally-symmetric ?object-type)
14 |       (equal ?rotationally-symmetric nil))
15 |     (-> (spec:property ?action-designator (:grasp ?grasp)))
16 |       (true)
17 |       (and (member ?armc ?arms)
18 |         (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
19 |           (member ?grasp ?grasps)))
20 |         (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
21 |         (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
22 |         (equal ?objects ?current-object-desig)
23 |         (-> (member ?left ?arms)
24 |           (and (lisp-fun man-int:get-action-trajectory :pouring :left) ?grasp T ?objects
25 |             ?left-pouring-pose)
26 |             (lisp-fun man-int:get-traj-poses-by-label ?left-pouring-pose :approach
27 |               ?left-approach-poses)
28 |               (lisp-fun man-int:get-traj-poses-by-label ?left-pouring-pose :tilting
29 |                 ?left-tilt-poses))
30 |               (and (equal ?left-approach-poses NIL)
31 |                 (equal ?left-tilt-poses NIL)))
32 |               (-> (member ?right ?arms)
33 |                 (and (lisp-fun man-int:get-action-trajectory :pouring :right) ?grasp T ?objects
34 |                   ?right-pouring-pose)
35 |                     (lisp-fun man-int:get-traj-poses-by-label ?right-pouring-pose :approach
36 |                       ?right-approach-poses)
37 |                       (lisp-fun man-int:get-traj-poses-by-label ?right-pouring-pose :tilting
38 |                         ?right-tilt-poses))
39 |                         (and (equal ?right-approach-poses NIL)
40 |                           (equal ?right-tilt-poses NIL)))
41 |                           (-> (desig:design-prop ?action-designator (:collision-mode ?collision-mode)))
42 |                             (true)
43 |                             (equal ?collision-mode nil))
44 |                             (desig:designator :action ((:type :pouring)
45 |                               (object ?current-object-desig)
46 |                               (object-type ?object-type)
47 |                               (object-name ?object-name)
48 |                               (arms ?arms)
49 |                               (grasp ?grasp)
50 |                               (left-approach-poses ?left-approach-poses)
51 |                               (right-approach-poses ?right-approach-poses)
52 |                               (left-tilt-poses ?left-tilt-poses)
53 |                               (right-tilt-poses ?right-tilt-poses)
54 |                               (collision-mode ?collision-mode)))
55 |                               ?resolved-action-designator))
56 |

```



```

1  | <- (desig.action-grounding ?action-designator (pour ?source-container ?target-container))
2  |   (spec property ?action-designator (:type :pouring))
3  |   (spec property ?current-target-container (type ?target-type))
4  |   (spec property ?current-target-container (:name ?target-name))
5  |   (not (cpoe:object-in-hand ?current-target-container ?arm)))
6  |   (lisp-fun obj-int:get-container-contents ?source-container ?contents)
7  |   (lisp-fun obj-int:empty-container ?source-container)
8  |   (lisp-fun obj-int:fill-container ?target-container ?contents)
9  |
10 |   (-> (spec property ?action-designator (target ?target-container))
11 |     (desig current-designator ?target-container ?current-target-container)
12 |     (spec property ?current-target-container (type ?target-type))
13 |     (and (member ?target-type (:container :bottle :glass))
14 |       (not (cpoe:object-in-hand ?current-target-container ?arm))))
15 |     (lisp-fun obj-int:get-container-contents ?source-container ?contents)
16 |     (lisp-fun obj-int:empty-container ?source-container)
17 |     (lisp-fun obj-int:fill-container ?target-container ?contents)))
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
50 |
51 |
52 |
53 |
54 |
55 |
56 |

```

Figure 54: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Placing Down* (12 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
8	52	3	1

Findings:

- designator definition contains two instead of only one variable
- some minor renaming (e.g. *source-container* instead of *object*)
- designator contains only some definitions, no assignments or calculations
- designator does not end by mapping the variables to designator "parts"

```

1  <- (design-action-grounding ?action-designator (pour ?resolved-action-designator))
2   (spec:property ?action-designator (:type :pouring))
3   (spec:property ?action-designator (?object ?object-designator))
4   (spec:current-designator ?object-designator ?current-object-design)
5   (spec:property ?current-object-design (:type ?object-type))
6   (spec:property ?current-object-design (:name ?object-name))
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

```

1  <- (design-action-grounding ?action-designator (pour ?resolved-action-designator))
2   (spec:property ?action-designator (:type :pouring))
3   (spec:property ?action-designator (?source ?source-designator))
4   (spec:current-designator ?source-designator ?current-source-design)
5   (spec:current-designator ?target-designator ?current-target-design)
6   (spec:property ?current-source-design (:type ?source-type))
7   (spec:property ?current-source-design (:name ?source-name))
8   (spec:property ?current-target-design (:type ?target-type))
9   (spec:property ?current-target-design (:name ?target-name))
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

Figure 55: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Slicing* (55 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
13	14	27	15

Findings:

- some minor renaming (e.g. *source* instead of *object*)
- designator calculates trajectories and poses in a different manner
- generated designator extends the mapping of variables to designator "parts" (misses *:object-type*, *:left-tilt-poses*, *:right-tilt-poses* and *:arms* but adds *:target*, *:source-name*, *:arm*, *:gripper-opening* and *:effort*)

```

1  <- (design-action-grounding ?action-designator (pour ?resolved-action-designator))
2    (spec-property ?action-designator :type :pouring)
3    (spec-property ?action-designator :object ?object-designator)
4
5    (spec-current-designator ?object-designator ?current-object-desig)
6    (spec-property ?current-object-desig :type ?object-type)
7    (spec-property ?current-object-desig :name ?object-name)
8    (spec-property ?action-designator :arms ?arms)
9      (true)
10   (and (man-int-robot-free-hand ?_?arm)
11     (equal ?arms ?arm)))
12   (lisp-fun man-int-get-object-transform ?current-object-desig ?object-transform)
13   (lisp-fun man-int-calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
14   (> (man-int-object-rotationally-symmetric ?object-type)
15     (equal ?rotationally-symmetric t))
16   (equal ?rotationally-symmetric nil)
17
18   (> (spec-property ?action-designator :grasp ?grasp))
19   (true)
20   (and (member ?arm ?arms)
21     (lisp-fun man-int-get-action-grasps ?object-type ?arm ?object-transform ?grasps)
22     (member ?grasp ?grasps)))
23
24   (lisp-fun man-int-get-action-gripping-effort ?object-type ?effort)
25   (lisp-fun man-int-get-action-gripper-opening ?object-type ?gripper-opening)
26   (equal ?objects (?current-object-desig))
27   (> (member ?left ?arms)
28     (and (lisp-fun man-int-get-action-trajectory :pouring ?left ?grasp T ?objects
29       ?left-pouring-pose)
30       (lisp-fun man-int-get-traj-poses-by-label ?left-pouring-pose :approach
31         ?left-approach-poses)
32       (lisp-fun man-int-get-traj-poses-by-label ?left-pouring-pose :tilting
33         ?left-tilt-poses)))
34     (and (equal ?left-approach-poses NIL)
35       (equal ?left-tilt-poses NIL)))
36   (> (member ?right ?arms)
37     (and (lisp-fun man-int-get-action-trajectory :pouring ?right ?grasp T ?objects
38       ?right-pouring-pose)
39       (lisp-fun man-int-get-traj-poses-by-label ?right-pouring-pose :approach
40         ?right-approach-poses)
41       (lisp-fun man-int-get-traj-poses-by-label ?right-pouring-pose :tilting
42         ?right-tilt-poses)))
43     (and (equal ?right-approach-poses NIL)
44       (equal ?right-tilt-poses NIL)))
45   (> (design-design-prop ?action-designator :collision-mode ?collision-mode))
46   (true)
47   (equal ?collision-mode nil))
48
49   (design-designator :action ((type :pouring)
50     (:object ?current-object-desig)
51     (:object-type ?object-type)
52     (:object-name ?object-name)
53     (:arms ?arms)
54     (:grasp ?grasp)
55     (:left-approach-poses ?left-approach-poses)
56     (:right-approach-poses ?right-approach-poses)
57     (:left-tilt-poses ?left-tilt-poses)
58     (:right-tilt-poses ?right-tilt-poses)
59     (:collision-mode ?collision-mode)))
60     ?resolved-action-designator))
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
844
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
```

Figure 56: Left: Manual designator for the action *Pouring* (56 lines). Right: Generated designator based on *Wiping* (39 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
9	26	26	4

Findings:

- some minor renaming (e.g. *source* instead of *object*)
 - designator calculates trajectories and poses in a different manner
 - generated designator shortens the mapping of variables to designator "parts" (misses *:object*, *:object-type*, *:object-name*, *:grasp* and *:arms*

```

1  :- (<- (desig:action-grounding ?action-designator (slice ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

```

1  :- (<- (desig:action-grounding ?action-designator (slice ?arm
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

Figure 57: Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Closing* (48 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
28	35	17	3

Findings:

- designator definition contains 14 instead of only one variable
- some minor renaming (e.g. *cutting* instead of *slicing*)
- designator calculates trajectories and poses in a different manner
- generation adds block about `cram-robot-interfaces` (2 lines)
- designator does not end by mapping the variables to designator "parts"

```

1  :-> (desig-action-grounding ?action-designator (slice ?resolved-action-designator))
2   | spec:property ?action-designator (:type :slicing)
3   | spec:property ?action-designator (?object ?object-designator)
4   | (design-current-designator ?object-designator ?current-object-design)
5   | (spec:property ?current-object-desig (?type ?object-type))
6   | (spec:property ?current-object-desig (?name ?object-name))
7   | (spec:property ?action-designator (:arm ?arm))
8   | (true)
9   | (man-int:robot-free-hand ?, ?arm)
10  | (lisp-fun:man-int:get-object-old-transform ?current-object-desig ?object-transform)
11  | (lisp-fun:man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
12  | (-> (man-int:object-rotationally-symmetric ?object-type)
13  | (equal ?rotationally-symmetric t)
14  | (equal ?rotationally-symmetric nil))
15  | (-> (spec:property ?action-designator (:grasp ?grasp))
16  | (true)
17  | (and (lisp-fun:man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
18  | (member ?grasp ?grasps)))
19
20
21
22  | (lisp-fun:man-int:get-action-gripping-effort ?object-type ?effort)
23  | (lisp-fun:man-int:get-action-gripper-opening ?object-type ?gripper-opening)
24  | (-> (equal ?objects ?current-object-design))
25  | (-> (equal ?arm ?left))
26  | (and (lisp-fun:man-int:get-action-trajectory :slicing) ?arm ?grasp T ?objects
27  | ?left-slicing-poses)
28  | (lisp-fun:man-int:get-traj-poses-by-label ?left-slicing-poses :slice-up)
29  | ?left-slice-up-poses)
30  | (lisp-fun:man-int:get-traj-poses-by-label ?left-slicing-poses :slice-down)
31  | ?left-slice-down-poses)
32  | (and (equal ?left-slice-up-poses NIL)
33  | (equal ?left-slice-down-poses NIL))
34  | (-> (equal ?arm ?right))
35  | (and (lisp-fun:man-int:get-action-trajectory :slicing) ?arm ?grasp T ?objects
36  | ?right-slicing-poses)
37  | (lisp-fun:man-int:get-traj-poses-by-label ?right-slicing-poses :slice-up)
38  | ?right-slice-up-poses)
39  | (lisp-fun:man-int:get-traj-poses-by-label ?right-slicing-poses :slice-down)
40  | ?right-slice-down-poses)
41  | (and (equal ?right-slice-up-poses NIL)
42  | (equal ?right-slice-down-poses NIL))
43  | (-> (desig:design-prop ?action-designator (:collision-mode ?collision-mode)))
44  | (true)
45  | (equal ?collision-mode nil))
46  | (desig:designator :action ((?type :slicing)
47  | (?object ?current-object-design)
48  | (?object-name ?object-name)
49  | (?arm ?arm)
50  | (?gripper-opening ?gripper-opening)
51  | (?effort ?effort)
52  | (?grasp ?grasp)
53  | (?left-slice-up-poses ?left-slice-up-poses)
54  | (?right-slice-up-poses ?right-slice-up-poses)
55  | (?left-slice-down-poses ?left-slice-down-poses)
56  | (?right-slice-down-poses ?right-slice-down-poses)
57  | (?collision-mode ?collision-mode)))
58  | (?resolved-action-designator))

1  :-> (desig-action-grounding ?action-designator (slice-into-two ?resolved-action-designator))
2   | spec:property ?action-designator (:type :slicing)
3   | spec:property ?action-designator (?object ?object-designator)
4   | (design-current-designator ?object-designator ?current-object-design)
5   | (spec:property ?current-object-desig (?type ?object-type))
6   | (spec:property ?current-object-desig (?name ?object-name))
7   | (spec:property ?action-designator (:arm ?arm))
8   | (true)
9   | (man-int:robot-free-hand ?, ?arm)
10  | (lisp-fun:man-int:get-object-old-transform ?current-object-design ?object-transform)
11  | (lisp-fun:man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
12  | (-> (man-int:object-rotationally-symmetric ?object-type)
13  | (equal ?rotationally-symmetric t)
14  | (equal ?rotationally-symmetric nil))
15  | (-> (spec:property ?action-designator (:grasp ?grasp))
16  | (true)
17  | (and (lisp-fun:man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
18  | (member ?grasp ?grasps)))
19
20
21  | (-> (spec:property ?action-designator (:object-half-pose ?object-half-pose)))
22  | (true)
23  | (format "Please infer where to cut the object, or use the query system to infer it here")
24  | (lisp-fun:man-int:get-action-gripping-effort ?object-type ?effort)
25  | (lisp-fun:man-int:get-action-gripper-opening ?object-type ?gripper-opening)
26  | (-> (equal ?arm ?left))
27  | (and (lisp-fun:man-int:get-action-trajectory :slicing) ?arm ?grasp T ?objects
28  | ?left-slice-poses)
29  | (lisp-fun:man-int:get-traj-poses-by-label ?left-slice-poses :slice-up)
30  | ?left-slice-up-poses)
31  | (lisp-fun:man-int:get-traj-poses-by-label ?left-slice-poses :slice-down)
32  | ?left-slice-down-poses)
33  | (and (equal ?left-slice-up-poses NIL)
34  | (equal ?left-slice-down-poses NIL))
35  | (-> (equal ?arm ?right))
36  | (and (lisp-fun:man-int:get-action-trajectory :slicing) ?arm ?grasp T ?objects
37  | ?right-slice-poses)
38  | (lisp-fun:man-int:get-traj-poses-by-label ?right-slice-poses :slice-up)
39  | ?right-slice-up-poses)
40  | (lisp-fun:man-int:get-traj-poses-by-label ?right-slice-poses :slice-down)
41  | ?right-slice-down-poses)
42  | (and (equal ?right-slice-up-poses NIL)
43  | (equal ?right-slice-down-poses NIL))
44  | (-> (desig:design-prop ?action-designator (:collision-mode ?collision-mode)))
45  | (true)
46  | (equal ?collision-mode nil))
47  | (desig:designator :action ((?type :slicing)
48  | (?object ?current-object-design)
49  | (?object-name ?object-name)
50  | (?arm ?arm)
51  | (?gripper-opening ?gripper-opening)
52  | (?effort ?effort)
53  | (?grasp ?grasp)
54  | (?left-slice-up-poses ?left-slice-up-poses)
55  | (?right-slice-up-poses ?right-slice-up-poses)
56  | (?left-slice-down-poses ?left-slice-down-poses)
57  | (?right-slice-down-poses ?right-slice-down-poses)
58  | (?collision-mode ?collision-mode)))
59  | (?resolved-action-designator))

```

Figure 58: Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Halving* (58 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
3	0	7	48

Findings:

- added lines describe block that is used for inferring the position / pose necessary for meeting the exact halve (not necessary for "normal" slicing)
- added lines are the same that are deleted when the reference and generated action are swapped (Fig. 15)
- remaining changes are places where the action verb is used in the "wrong" form (generated *slice* instead of *slicing*)

```

1  (-> (design-action-grounding ?action-designator (slice ?resolved-action-designator))
2    (spec:property ?action-designator :type :slicing))
3    (spec:property ?action-designator :object ?object-designator)
4      (desig:current-designator ?object-designator ?current-object-desig)
5        (spec:property ?current-object-desig :type ?object-type)
6          (spec:property ?current-object-desig :name ?object-name))
7
8    (-> (spec:property ?action-designator :arm ?arm))
9      (true)
10     (man-int:robot-free-hand ?_ ?arm))
11       (lisp-fun man-int:get-object-old-transform ?current-object-desig ?object-transform)
12         (lisp-fun man-int:calculate-object-face ?object-transform ?facing-robot-face ?bottom-face))
13           (-> (man-int:object-rotationally-symmetric ?object-type)
14             (equal ?rotationally-symmetric t)
15               (equal ?rotationally-symmetric nil)))
16
17   (-> (spec:property ?action-designator :grasp ?grasp))
18     (true)
19       (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
20         (member ?grasp ?grasps)))
21         (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
22           (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
23             (equal ?objects ?current-object-desig))
24               (-> (equal ?arm :left)
25                 (and (lisp-fun man-int:get-action-trajectory :slicing ?arm ?grasp T ?objects
26                   ?left-slicing-poses)
27                     (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-poses :slice-up
28                       ?left-slice-up-poses)
29                         (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-poses :slice-down
30                           ?left-slice-down-poses)))
31               (and (equal ?left-slice-up-poses NIL)
32                 (equal ?left-slice-down-poses NIL)))
33                 (-> (equal ?arm :right)
34                   (and (lisp-fun man-int:get-action-trajectory :slicing ?arm ?grasp T ?objects
35                     ?right-slicing-poses)
36                       (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-poses :slice-up
37                         ?right-slice-up-poses)
38                           (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-poses :slice-down
39                             ?right-slice-down-poses)))
40               (and (equal ?right-slice-up-poses NIL)
41                 (equal ?right-slice-down-poses NIL)))
42                 (-> (design:desig-prop ?action-designator :collision-mode ?collision-mode)
43                   (true)
44                     (equal ?collision-mode nil))
45                     (desig:designator :action ((type :slicing)
46                       (object ?current-object-desig)
47                         (object-name ?object-name)
48                           (arm ?arm)
49                             (gripper-opening ?gripper-opening)
50                               (effort ?effort)
51                                 (grasp ?grasp)
52                                   (left-slice-up-poses ?left-slice-up-poses)
53                                     (right-slice-up-poses ?right-slice-up-poses)
54                                       (left-slice-down-poses ?left-slice-down-poses)
55                                         (right-slice-down-poses ?right-slice-down-poses)
56                                           (collision-mode ?collision-mode)
57                                             ?resolved-action-designator)))
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
267
268
269
270
271
272
273
274
275
276
277
278
278
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
320
321
322
323
324
325
326
327
327
328
329
330
331
332
333
334
335
336
337
337
338
339
340
341
342
343
344
345
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
367
368
369
370
371
372
373
374
375
376
377
377
378
379
380
381
382
383
384
385
386
387
387
388
389
390
391
392
393
394
395
396
397
397
398
399
399
400
401
402
403
404
405
406
407
407
408
409
410
411
412
413
414
415
415
416
417
418
419
420
421
422
423
424
425
426
427
427
428
429
430
431
432
433
434
435
436
437
437
438
439
440
441
442
443
444
445
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
467
468
469
470
471
472
473
474
475
476
477
477
478
479
480
481
482
483
484
485
486
487
487
488
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
520
521
522
523
524
525
526
527
527
528
529
530
531
532
533
534
535
536
537
537
538
539
540
541
542
543
544
545
546
547
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
567
568
569
569
570
571
572
573
574
575
576
577
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
607
608
609
609
610
611
612
613
614
615
615
616
617
618
619
619
620
621
622
623
624
625
625
626
627
628
629
629
630
631
632
633
634
635
635
636
637
638
639
639
640
641
642
643
644
644
645
646
647
647
648
649
649
650
651
652
653
653
654
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1
```

```

1  (-> (design:action-grounding ?action-designator (slice ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

```

1  (-> (design:action-grounding ?action-designator (slice ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

Figure 60: Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Opening* (47 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
32	40	13	2

Findings:

- designator definition contains 13 instead of only one variable
- some minor renaming (e.g. *cutting* instead of *slicing*)
- generation adds block about `cram-robot-interfaces` (2 lines)
- designator calculates trajectories and poses in a different manner
- designator does not end by mapping the variables to designator "parts"

```

1 1 <- (design-action-grounding ?action-designator (slice ?resolved-action-designator))
2 2 (spec:property ?action-designator (:type :slicing))
3 3 (spec:property ?action-designator (?object ?object-designator))
4 4 (design-current-designator ?object-designator ?current-object-desig)
5 5 (spec:property ?current-object-desig (:type ?object-type))
6 6 (spec:property ?current-object-desig (:name ?object-name))
7 7 (> (spec:property ?action-designator (?arm ?arm))
8 8 (true)
9 9 (man-int:robot-free-hand ? ?arm)
10 10 (lisp-fun man-int:get-object-old-transform ?current-object-desig ?object-transform)
11 11 (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
12 12 (> (man-int:object-rotationally-symmetric ?object-type)
13 13 (equal ?rotationally-symmetric t)
14 14 (equal ?rotationally-symmetric nil))
15 15 (> (spec:property ?action-designator (?grasp ?grasp))
16 16 (true)
17 17 (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
18 18 (member ?grasp ?grasps))
19 19 (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
20 20 (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
21 21 (equal ?objects (?current-object-desig))
22 22 (> (equal ?arm ?left)
23 23 (and (lisp-fun man-int:get-action-trajectory :slicing ?arm ?grasp T ?objects
24 24 ?left-slicing-poses)
25 25 (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-poses :slice-up
26 26 ?left-slice-up-poses)
27 27 (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-poses :slice-down
28 28 ?left-slice-down-poses))
29 29 (and (equal ?left-slice-up-poses NIL)
30 30 (equal ?left-slice-down-poses NIL)))
31 31 (> (equal ?right-slicing-poses)
32 32 (and (lisp-fun man-int:get-action-trajectory :slicing ?arm ?grasp T ?objects
33 33 ?right-slicing-poses)
34 34 (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-poses :slice-up
35 35 ?right-slice-up-poses)
36 36 (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-poses :slice-down
37 37 ?right-slice-down-poses))
38 38 (and (equal ?right-slice-up-poses NIL)
39 39 (equal ?right-slice-down-poses NIL)))
40 40 (> (desig:desig-prop ?action-designator (:collision-mode ?collision-mode))
41 41 (true)
42 42 (equal ?collision-mode nil))
43 43 (design:designator :action ((?type :slicing)
44 44 (?object ?current-object-desig)
45 45 (?object-name ?object-name)
46 46 (?arm ?arm)
47 47 (?gripper-opening ?gripper-opening)
48 48 (?effort ?effort)
49 49 (?grasp ?grasp)
50 50 (?left-slice-up-poses ?left-slice-up-poses)
51 51 (?right-slice-up-poses ?right-slice-up-poses)
52 52 (?left-slice-down-poses ?left-slice-down-poses)
53 53 (?right-slice-down-poses ?right-slice-down-poses)
54 54 (?collision-mode ?collision-mode)))
55 55 (?resolved-action-designator))
56 56

```



```

1 1 <- (design-action-grounding ?action-designator (slice ?current-object-desig ?knife ?cutting-board
2 2 ?small-slice ?big-slice))
3 3 (spec:property ?action-designator (:type :slicing))
4 4 (spec:property ?action-designator (?object ?object-designator))
5 5 (design-current-designator ?object-designator ?current-object-desig)
6 6 (spec:property ?current-object-desig (:type ?object-type))
7 7 (spec:property ?current-object-desig (:name ?object-name))
8 8 (> (spec:property ?action-designator (?knife ?knife))
9 9 (and (cram-robot-interfaces:robot ?robot)
10 10 (cram-robot-interfaces:arm ?robot ?knife)))
11 11
12 12
13 13
14 14
15 15 (> (spec:property ?action-designator (:cutting-board ?cutting-board))
16 16 (true)
17 17 (and (cram-robot-interfaces:robot ?robot)
18 18 (cram-robot-interfaces:table ?robot ?cutting-board))
19 19 (lisp-fun obj-int:get-object-transform ?current-object-desig ?object-transform)
20 20 (lisp-fun obj-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
21 21 (lisp-fun obj-int:get-object-type-cutting-poses ?object-type ?facing-robot-face ?bottom-face)
22 22
23 23
24 24
25 25
26 26
27 27
28 28
29 29
30 30
31 31
32 32
33 33
34 34 ?cutting-poses)
35 35 (lisp-fun extract-slicing-poses ?knife ?cutting-board ?cutting-poses ?small-slice ?big-slice)
36 36
37 37
38 38
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
54 54
55 55
56 56

```

Figure 61: Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Picking Up* (20 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
9	44	4	7

Findings:

- tba

```

1  (+< (design-action-grounding :action-designator (slice ?resolved-action-designator))
2    (spec:property ?action-designator :type :slicing))
3    (spec:property ?action-designator :object ?object-designator)
4      (design-current-designator ?object-designator ?current-object-design)
5        (spec:property ?current-object-desig :type ?object-type)
6          (spec:property ?current-object-desig :name ?object-name)
7            (> (spec:property ?non-ion-designate (:arm ?am))
8              (true))
9                (impl-int:robot-free-hand ?_2 ?am))
10               (lisp-fun:man-int:get-object-old-transform ?current-object-design :object-transform)
11                 (lisp-fun:man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
12                   (> (impl-int:object-rotationally-symmetric ?object-type)
13                     (equal ?rotationally-symmetric t))
14                       (equal ?rotationally-symmetric nil))
15                         (> (spec:property ?ion-designate :grasp ?grasp))
16                           (true)
17                             (and (lisp-fun:man-int:get-action-grasps ?object-type ?am ?object-transform ?grasps)
18                               (member ?grasp ?grasps)))
19                                 (lisp-fun:man-int:get-action-grasping-effort ?object-type ?effort)
20                                   (lisp-fun:man-int:get-action-gripper-opening ?object-type ?gripper-opening)
21                                     (equal ?objects ?current-object-design))
22                                       (> (equal ?am: :left)
23                                         (and (lisp-fun:man-int:get-action-trajectory :slicing ?am ?grasp T ?objects
24                                           ?left-slicing-poses)
25                                             (lisp-fun:man-int:get-traj-poses-by-label ?left-slicing-poses :slice-up
26                                               ?left-slice-up-poses)
27                                                 (lisp-fun:man-int:get-traj-poses-by-label ?left-slicing-poses :slice-down
28                                                   ?left-slice-down-poses)))
29                                       (and (equal ?left-slice-up-poses NIL)
30                                         (equal ?left-slice-down-poses NIL)))
31                                         (> (equal ?am: :right)
32                                           (and (lisp-fun:man-int:get-action-trajectory :slicing ?am ?grasp T ?objects
33                                             ?right-slicing-poses)
34                                               (lisp-fun:man-int:get-traj-poses-by-label ?right-slicing-poses :slice-up
35                                                 ?right-slice-up-poses)
36                                                 (lisp-fun:man-int:get-traj-poses-by-label ?right-slicing-poses :slice-down
37                                                   ?right-slice-down-poses)))
38                                       (and (equal ?right-slice-up-poses NIL)
39                                         (equal ?right-slice-down-poses NIL)))
40                                         (> (ddrg:design-prop ?action-designator :collision-mode ?collision-mode)
41                                           (true))
42                                           (equal ?collision-mode nil))
43                                         (design-designator :action ((?type :slicing)
44                                           (:object ?current-object-design)
45                                           (:object-name ?object-name)
46                                           (:arm ?am)
47                                           (:gripper-opening ?gripper-opening)
48                                           (:effort ?effort)
49                                           (:grasp ?grasp)
50                                             (left-slice-up-poses ?left-slice-up-poses)
51                                               (right-slice-up-poses ?right-slice-up-poses)
52                                                 (left-slice-down-poses ?left-slice-down-poses)
53                                                   (right-slice-down-poses ?right-slice-down-poses)
54                                                     (collision-mode ?collision-mode))
55                                                       ?resolved-action-designator)))

```



```

1  (+< (design-action-grounding :action-designator (slice ?object-designator ?knife-designator ?small-slice ?big-slice))
2    (spec:property ?action-designator :type :slicing))
3    (spec:property ?action-designator :object ?object-designator)
4      (cpose:object-in-hand ?object-designator ?hand))
5        (> (spec:property ?action-designator :tool ?knife-designator)
6          (cpose:object-in-hand ?knife-designator ?hand))
7            (design-current-designator ?object-designator ?current-object-designator)
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```

Figure 62: Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Placing Down* (11 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
5	49	5	1

Findings:

- designator definition contains four instead of only one variable
- designator does not calculate trajectories and poses
- designator does not end by mapping the variables to designator "parts"

```
1  (< design-action-grounding ?action-designator (?slice ?resolved-action-designator))
2  (spec:property ?action-designator (:type :slicing))
3  (spec:property ?action-designator (?object ?object-designator))
4  (design-current-designator ?object-designator ?current-object-desig)
5  (spec:property ?current-object-desig (:type ?object-type))
6  (spec:property ?current-object-desig (:name ?object-name))
7  (spec:property ?action-designator (:arm ?arm))
8  (true)
9  (man-int:robot-free-hand ?_ ?arm))
10 
11  (lisp-fun man-int:get-object-old-transform ?current-object-desig ?object-transform)
12  (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
13  (man-int:object-rotationally-symmetric ?object-type)
14  (equal ?rotationally-symmetric t)
15  (equal ?rotationally-symmetric nil))
16  (spec:property ?action-designator (:grasp ?grasp))
17  (true)
18 
19  (and (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
20  (member ?grasp ?grasps)))
21  (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
22  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
23  (equal ?objects ?current-object-designator)
24  (equal ?left ?arm-left)
25  (and (lisp-fun man-int:get-action-trajectory :slicing ?arm) ?grasp T ?objects
26  ?left-slicing)
27  (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-pose :slice-up)
28  (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-pose :slice-down)
29  (lisp-fun man-int:get-traj-poses-by-label ?left-slicing-pose :slice-down-poses))
30  (and (equal ?left-slice-up-poses NIL)
31  (equal ?left-slice-down-poses NIL)))
32  (lisp-fun man-int:get-action-trajectory :slicing ?arm) ?grasp T ?objects
33  ?right-slicing)
34  (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-pose :slice-up)
35  (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-pose :slice-down)
36  (lisp-fun man-int:get-traj-poses-by-label ?right-slicing-pose :slice-down-poses))
37  (and (equal ?right-slice-up-poses NIL)
38  (equal ?right-slice-down-poses NIL)))
39  (spec:design-prop ?action-designator (:collision-mode ?collision-mode))
40  (true)
41  (equal ?collision-mode nil))
42  (design-designator :action ((:type :slicing)
43  (:object ?current-object-designig)
44  (:object-name ?object-name)
45  (:arm ?arm)
46  (:gripper-opening ?gripper-opening)
47  (:effort ?effort)
48  (:grasp ?grasp)
49  (:left-slice-up-poses)
50  (:right-slice-up-poses)
51  (:left-slice-down-poses)
52  (:right-slice-down-poses)
53  (:right-slice-down-poses)
54  (:left-slice-down-poses)
55  (:right-slice-down-poses)
56  (:left-slice-down-poses)
57  (:right-slice-down-poses)
58  (:collision-mode ?collision-mode)))
59 
60  |?resolved-action-designator|)
```



```
1  (< design-action-grounding ?action-designator (?slice ?resolved-action-designator))
2  (spec:property ?action-designator (:type :slicing))
3  (spec:property ?action-designator (?object ?object-designator))
4  (design-current-designator ?object-designator ?current-object-desig)
5  (spec:property ?current-object-desig (:type ?object-type))
6  (spec:property ?current-object-desig (:name ?object-name))
7  (spec:property ?action-designator (:arms ?arms))
8  (true)
9  (and (man-int:robot-free-hand ?_ ?arm)
10  (equal ?arms ?arm)))
11  (lisp-fun man-int:get-object-transform ?current-object-designator ?object-transform)
12  (lisp-fun man-int:calculate-object-faces ?object-transform (?facing-robot-face ?bottom-face))
13 
14 
15  (spec:property ?action-designator (:grasp ?grasp))
16  (true)
17  (and (member ?arm ?arms)
18  (lisp-fun man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
19  (member ?grasp ?grasps)))
20  (lisp-fun man-int:get-action-gripping-effort ?object-type ?effort)
21  (lisp-fun man-int:get-action-gripper-opening ?object-type ?gripper-opening)
22  (equal ?objects ?current-object-designator)
23  (member ?left ?arms)
24  (and (lisp-fun man-int:get-action-trajectory :cutting ?left ?grasp T ?objects
25  ?left-slice-pose)
26  (lisp-fun man-int:get-traj-poses-by-label ?left-slice-pose :approach)
27  (lisp-fun man-int:get-traj-poses-by-label ?left-slice-pose :cutting)
28  (lisp-fun man-int:get-traj-poses-by-label ?left-slice-pose :cutting-poses))
29  (and (equal ?left-approach-poses NIL)
30  (equal ?left-cutting-poses NIL)))
31  (member ?right ?arms)
32  (and (lisp-fun man-int:get-action-trajectory :cutting ?right ?grasp T ?objects
33  ?right-slice-pose)
34  (lisp-fun man-int:get-traj-poses-by-label ?right-slice-pose :approach)
35  (lisp-fun man-int:get-traj-poses-by-label ?right-slice-pose :cutting)
36  (lisp-fun man-int:get-traj-poses-by-label ?right-slice-pose :cutting-poses))
37  (and (equal ?right-approach-poses NIL)
38  (equal ?right-cutting-poses NIL)))
39  (spec:design-prop ?action-designator (:collision-mode ?collision-mode))
40  (true)
41  (equal ?collision-mode nil))
42  (design-designator :action ((:type :cutting)
43  (:object ?current-object-designig)
44  (:object-type ?object-type)
45  (:object-name ?object-name)
46  (:arms ?arms))
47  (:grasp ?grasp)
48  (:left-approach-poses)
49  (:right-approach-poses)
50  (:left-cutting-poses)
51  (:right-cutting-poses)
52  (:collision-mode ?collision-mode))
53  (num-slices 2)
54  (:slice-sizes (1 2)))
55  |?resolved-action-designator|)
```

Figure 63: Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Pouring* (55 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
6	6	29	20

Findings:

- some minor renaming (e.g. *cutting* instead of *slicing*)
 - generated designator extends the mapping of variables to designator "parts" (misses *:arm*, *:gripper-opening* and *:effort* but adds *:object-type*, *:arms*, *:num-slices* and *:slice-sizes*)

```

1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
27 27
28 28
29 29
30 30
31 31
32 32
33 33
34 34
35 35
36 36
37 37
38 38
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
54 54
55 55

```

Left: Manual designator for the action *Slicing* (55 lines). Right: Generated designator based on *Wiping* (6 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
0	49	5	1

Findings:

- designator contains no definition and no calculations but just the mapping part
- generated designator shortens the mapping of variables to designator "parts" (misses :object, :arm, :gripper-opening, :grasp, :left-slice-down-poses, :right-slice-down-poses and :effort)

```

1  <- (desig:action-grounding ?action-designator (wipe ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16  (spec:property ?action-designator (:type :wiping))
17  (spec:property ?action-designator (:grasp ?grasp))
18  (spec:property ?action-designator (?arm ?arm))
19  (desig:desig-prop ?action-designator (?surface ?surface-designator))
20  (desig:current-designator ?surface-designator ?current-surface-designator)
21  (spec:property ?current-surface-designator (type ?surface-type))
22  (equal ?surface ?current-surface-designator))
23  (> (equal ?grasp :scrubbing)
24  (lisp-fun different-surface-types ?grasp ?surface ?current-grasp)
25  (equal ?grasp ?current-grasp))
26  (desig:desig-prop ?action-designator (:collision-mode ?collision-mode))
27  (> (equal ?arm :left)
28  (and (lisp-fun get-trajectory :wiping) ?arm ?current-grasp T ?surface ?lists)
29
30
31
32  (lisp-fun man-int:get-traj-poses-by-label ?lists :wiping
33  ?left-wipe-poses)
34  (lisp-fun man-int:get-traj-poses-by-label ?lists :initial
35  ?left-initial-poses))
36  (and (equal ?left-wipe-poses NIL)
37  (equal ?left-initial-poses NIL))
38  (> (equal ?arm :right)
39  (and (lisp-fun get-trajectory :wiping) ?arm ?current-grasp T ?surface ?lists)
40  (lisp-fun man-int:get-traj-poses-by-label ?lists :wiping
41  ?right-wipe-poses)
42  (lisp-fun man-int:get-traj-poses-by-label ?lists :initial
43  ?right-initial-poses))
44  (and (equal ?right-wipe-poses NIL)
45  (equal ?right-initial-poses NIL))
46  (desig:designator :action ((:type :wiping)
47  (:collision-mode ?collision-mode)
48
49  (lisp-fun (left-wipe-poses ?left-wipe-poses)
50  (right-wipe-poses ?right-wipe-poses)
51  (left-initial-poses ?left-initial-poses)
52
53  (right-initial-poses ?right-initial-poses)
54
55  ?resolved-action-designator)))

```



```

1  <- (desig:action-grounding ?action-designator (wipe ?arm)
2  ?gripper-opening
3  ?distance
4  ?surface-designator
5  ?towel-designator
6  ?left-reach-poses
7  ?right-reach-poses
8  ?left-grasp-poses
9  ?right-grasp-poses
10  (?left-lift-pose)
11  (?right-lift-pose)
12  (?left-2nd-lift-pose)
13  (?right-2nd-lift-pose)
14  ?joint-name
15  ?environment-obj)
16  (spec:property ?action-designator (:type :cleaning))
17  (spec:property ?action-designator (:object ?surface-designator))
18  (spec:property ?surface-designator (type ?surface-type))
19  (obj-int:object-type-subtype ?surface ?surface-type)
20  (spec:property ?surface-designator (urdf-name ?surface-name))
21  (spec:property ?surface-designator (:part-of ?btr-environment))
22  (spec:property ?towel-designator (tool ?towel-designator))
23  (spec:property ?towel-designator (:type :towel))
24  (spec:property ?towel-designator (:part-of ?btr-environment))
25
26
27  (> (spec:property ?action-designator (:arm ?arm))
28  (true)
29  (and (cram-robot-interfaces:robot ?robot)
30  (cram-robot-interfaces:arm ?robot ?arm)))
31  (spec:property ?action-designator (:distance ?distance))
32  (lisp-fun obj-int:get-object-gripper-opening ?towel-designator ?gripper-opening)
33  (lisp-fun get-surface-link ?surface-name ?btr-environment ?surface-link)
34  (lisp-fun get-connecting-joint ?surface-link ?connecting-joint)
35  (lisp-fun cl-urdf:name ?connecting-joint ?joint-name)
36  (btr:bullet-world ?world)
37  (lisp-fun btr:object ?world ?btr-environment ?environment-obj)
38  (lisp-fun get-surface-pose-and-transform ?surface-name ?btr-environment
39  ?surface-poses ?surface-transform)
40  (lisp-fun obj-int:get-object-grasping-poses ?towel-designator
41  :towel :left :close ?surface-transform ?left-poses)
42  (lisp-fun obj-int:get-object-grasping-poses ?towel-designator
43
44  :towel :right :close ?surface-transform ?right-poses)
45  (lisp-fun cram-mobile:pick-place-plans::extract-pick-up-manipulation-poses
46  ?arm ?left-poses ?right-poses
47  ?left-reach-poses ?right-reach-poses
48  ?left-grasp-poses ?right-grasp-poses
49  ?left-lift-poses ?right-lift-poses))
50  (> (lisp-pred identity ?left-lift-poses)
51  (equal ?left-lift-poses (?left-lift-pose ?left-2nd-lift-pose)))
52  (equal (NIL NIL) (?left-lift-pose ?left-2nd-lift-pose)))
53  (> (lisp-pred identity ?right-lift-poses)
54  (equal ?right-lift-poses (?right-lift-pose ?right-2nd-lift-pose)))
55  (equal (NIL NIL) (?right-lift-pose ?right-2nd-lift-pose))))|)

```

Figure 65: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Closing* (52 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
36	19	16	0

Findings:

- designator definition contains 15 instead of only one variable
- some minor renaming (e.g. *cleaning* instead of *wiping*)
- designator calculates trajectories and poses in a different manner
- generation adds block about **cram-robot-interfaces** (2 lines)
- designator does not end by mapping the variables to designator "parts"

```

1  <- (design-action-grounding ?action-designator (wipe ?resolved-action-designator))
2   (spec:property ?action-designator (:type :wiping))
3   (spec:property ?action-designator (:grasp ?grasp))
4
5
6
7   (spec:property ?action-designator (:arm ?arm))
8   (design:desig-prop ?action-designator (:surface ?surface-designator))
9
10
11
12
13
14
15   (design:current-designator ?surface-designator ?current-surface-designator)
16   (spec:property ?current-surface-designator (:type ?surface-type))
17
18   (-> (equal ?surface (?current-surface-designator))
19     (lisp-fun:differentiate-surface-types ?grasp ?surface ?current-grasp)
20
21   (design:desig-prop ?action-designator (:collision-mode ?collision-mode))
22
23   (-> (equal ?arm :left)
24     (and (lisp-fun:get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
25       (lisp-fun:man-int:get-traj-poses-by-label ?lists :wiping
26         ?left-wipe-poses)
27       (lisp-fun:man-int:get-traj-poses-by-label ?lists :initial
28         ?left-initial-poses)
29       (and (equal ?left-wipe-poses NIL)
30         (equal ?left-initial-poses NIL)))
31
32   (-> (equal ?arm :right)
33     (and (lisp-fun:get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
34
35       (lisp-fun:man-int:get-traj-poses-by-label ?lists :wiping
36         ?right-wipe-poses)
37       (lisp-fun:man-int:get-traj-poses-by-label ?lists :initial
38         ?right-initial-poses)
39       (and (equal ?right-wipe-poses NIL)
40         (equal ?right-initial-poses NIL)))
41
42
43   (design:designator :action ((:type :wiping)
44     (:collision-mode ?collision-mode)
45
46
47
48
49
50
51
52     (:left-wipe-poses ?left-wipe-poses)
53     (:right-wipe-poses ?right-wipe-poses)
54     (:left-initial-poses ?left-initial-poses)
55     (:right-initial-poses ?right-initial-poses)))
56
57   ?resolved-action-designator)

```



```

1  <- (design-action-grounding ?action-designator (wipe-surface ?resolved-action-designator))
2   (spec:property ?action-designator (:type :cleaning))
3   (spec:property ?action-designator (:object ?object-designator))
4   (design:current-designator ?object-designator ?current-object-desig)
5   (spec:property ?current-object-desig (:type ?object-type))
6   (spec:property ?current-object-desig (:name ?object-name))
7
8   (-> (spec:property ?action-designator (:arm ?arm)))
9
10  (true)
11  (man:int:robot-free-hand ?_ ?arm))
12  (-> (spec:property ?action-designator (:grasp ?grasp)))
13
14  (-> (and (lisp-fun:man-int:get-action-grasps ?object-type ?arm ?grasps)
15    (member ?grasp ?grasps)))
16
17  (-> (spec:property ?action-designator (:towel ?towel-designator)))
18  (design:current-designator ?towel-designator ?current-towel-desig)
19  (spec:property ?current-towel-desig (:type ?towel-type))
20
21  (lisp-fun:man-int:get-action-gripping-effort ?object-type ?effort)
22  (lisp-fun:man-int:get-action-gripper-opening ?object-type ?gripper-opening)
23  (equal ?objects (?current-object-desig ?current-towel-desig))
24
25
26  (-> (equal ?arm :left)
27    (and (lisp-fun:man-int:get-action-trajectory :wiping ?arm ?grasp T ?objects)
28      (lisp-fun:man-int:get-traj-poses-by-label ?left-wiping-poses :wiping-up)
29      (lisp-fun:man-int:get-traj-poses-by-label ?left-wiping-poses :wiping-down)
30      (and (equal ?left-wiping-poses NIL)
31        (equal ?left-wiping-down-poses NIL)))
32
33  (-> (equal ?arm :right)
34    (and (lisp-fun:man-int:get-action-trajectory :wiping ?arm ?grasp T ?objects)
35      (lisp-fun:man-int:get-traj-poses-by-label ?right-wiping-poses :wiping-up)
36      (lisp-fun:man-int:get-traj-poses-by-label ?right-wiping-poses :wiping-down)
37      (and (equal ?right-wiping-poses NIL)
38        (equal ?right-wiping-down-poses NIL)))
39
40  (-> (design:desig-prop ?action-designator (:collision-mode ?collision-mode)))
41
42  (true)
43  (equal ?collision-mode NIL))
44  (design:designator :action ((:type :cleaning)
45    (:object ?current-object-desig)
46    (:name ?object-name)
47    (:arm ?arm)
48    (:gripper-opening ?gripper-opening)
49    (:effort ?effort)
50    (:grasp ?grasp)
51    (:towel ?current-towel-desig)
52    (:towel-name ?towel-name)
53    (:left-wipe-up-poses ?left-wipe-up-poses)
54    (:right-wipe-up-poses ?right-wipe-up-poses)
55    (:left-wipe-down-poses ?left-wipe-down-poses)
56    (:right-wipe-down-poses ?right-wipe-down-poses)
57
58   ?resolved-action-designator)

```

Figure 66: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Halving* (56 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
25	4	28	3

Findings:

- some minor renaming (e.g. *cleaning* instead of *wiping*)
- designator calculates trajectories and poses in a different manner
- generated designator extends the mapping of variables to designator "parts" (adds *:object*, *:object-name*, *:arm*, *:gripper-opening*, *:grasp*, *:towel*, *:towel-name* and *:effort*)

```

(<- (desig-action-grounding :action-designator (wipe ?resolved-action-designator))
  (spec:property ?action-designator (:type :wiping))
  (spec:property ?action-designator (:grasp ?grasp))
  (spec:property ?action-designator (:arm ?arm))

  (spec:design-prop :action-designator (:surface ?surface-designator))
  (design-current-designator ?surface-designator ?current-surface-designator)
  (spec:property ?current-surface-designator (:type ?surface-type))
  (equal ?surface ?current-surface-designator)
  (>= (equal ?grasp :scrubbing)
    (lisp-fun-differentiate-surface-types ?grasp ?surface ?current-grasp)
    (equal ?grasp ?current-grasp))
  (design desig-prop :action-designator (:collision-mode ?collision-mode))

  (>= (equal ?arm :left)
    (and (lisp-fun-get-trajectory :wiping) ?arm ?current-grasp T ?surface ?lists)
    (lisp-fun man-int-get-traj-poses-by-label ?lists :wiping
      ?left-wipe-poses)
    (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
      ?left-initial-poses))
  (and (equal ?left-wipe-poses NIL)
    (equal ?left-initial-poses NIL)))
  (>= (equal ?arm :right)
    (and (lisp-fun-get-trajectory :wiping) ?arm ?current-grasp T ?surface ?lists)
    (lisp-fun man-int-get-traj-poses-by-label ?lists :wiping
      ?right-wipe-poses)
    (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
      ?right-initial-poses))
  (and (equal ?right-wipe-poses NIL)
    (equal ?right-initial-poses NIL)))

  (desig:designator :action ((:type :wiping)
    (:collision-mode ?collision-mode))

  (<- (desig-action-grounding :action-designator (:plans :wipe ?resolved-action-designator))
  (spec:property ?action-designator (:type :cleaning))
  (spec:property ?action-designator (:object ?object?designator))
  (spec:property ?current-object-designator (:type ?object-type))
  (spec:property ?current-object-designator (:name ?object-name))
  (design-current-designator ?object-designator (:tool ?tool-designator))
  (spec:property ?current-tool-designator (:type ?tool-type))
  (spec:property ?current-tool-designator (:name ?tool-name))
  (>= (spec:property ?action-designator (:arm ?arm))
    (man-int:robot-free-hand ?arm)
    (lisp-fun-man-int-get-object-transform ?current-object-designator ?object-transform)
    (lisp-fun-man-int:calculate-object-faces ?object-transform ?facing-robot-face ?bottom-face)
    (>= (spec:property ?action-designator (:grasp ?grasp))
      (true)
      (and (lisp-fun-man-int:get-action-grasps ?object-type ?arm ?object-transform ?grasps)
        (member ?grasp ?grasps))
      (lisp-fun-man-int:get-action-gripping-effort ?object-type ?effort)
      (lisp-fun-man-int:get-action-gripper-opening ?object-type ?gripper-opening)
      (equal ?object ?current-object-designator ?current-tool-designator)
      (>= (equal ?arm :left)
        (and (lisp-fun-man-int:get-action-trajectory :wiping) ?arm ?grasp T ?objects
          ?left-trajectory)
        (lisp-fun-man-int:get-traj-poses-by-label ?left-trajectory :reaching
          ?left-reach-poses)
        (lisp-fun-man-int:get-traj-poses-by-label ?left-trajectory :grasping
          ?left-grasp-poses))
      (lisp-fun-man-int-get-traj-poses-by-label ?left-trajectory :wiping
        ?left-wipe-poses)
      (and (equal ?left-reach-poses NIL)
        (equal ?left-grasp-poses NIL)
        (equal ?left-wipe-poses NIL)))
      (>= (equal ?arm :right)
        (and (lisp-fun-man-int:get-action-trajectory :wiping) ?arm ?grasp T ?objects
          ?right-trajectory)
        (lisp-fun-man-int:get-traj-poses-by-label ?right-trajectory :reaching
          ?right-reach-poses)
        (lisp-fun-man-int:get-traj-poses-by-label ?right-trajectory :grasping
          ?right-grasp-poses)
        (lisp-fun-man-int:get-traj-poses-by-label ?right-trajectory :wiping
          ?right-wipe-poses)
        (and (equal ?right-reach-poses NIL)
          (equal ?right-grasp-poses NIL)
          (equal ?right-wipe-poses NIL)))
        (desig:designator :action ((:type :wiping)
          (object ?current-object-designator)
          (object-name ?object-name)
          (tool ?current-tool-designator)
          (tool-name ?tool-name)
          (arm ?arm)
          (gripper-opening ?gripper-opening)
          (effort ?effort)
          (grasp ?grasp)
          (left-reach-poses ?left-reach-poses)
          (right-reach-poses ?right-reach-poses)
          (left-grasp-poses ?left-grasp-poses)
          (right-grasp-poses ?right-grasp-poses)
          (left-wipe-poses ?left-wipe-poses)
          (right-wipe-poses ?right-wipe-poses)
          (:cleaning :cleaning))))))))

```

Figure 67: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Holding* (63 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
36	8	21	6

Findings:

- some minor renaming (e.g. *cleaning* instead of *wiping*)
 - designator calculates trajectories and poses in a different manner
 - generated designator extends the mapping of variables to designator "parts" (misses *:collision-mode*, *:left-initial-pose* and *:right-initial-pose* but adds *:object*, *:object-name*, *:tool*, *:tool-name*, *:arm*, *:gripper-opening*, *:grasp*, *:cleaning*, *:left-reach-poses*, *:right-reach-poses*, *:left-grasp-poses*, *:right-grasp-poses* and *:effort*)

```

1  <- (design:action-grounding ?action-designator (wipe ?resolved-action-designator))
2
3
4
5
6
7
8
9
10
11
12
13
14
15  (spec:property ?action-designator (:type :wiping))
16  (spec:property ?action-designator (:grasp ?grasp))
17  (spec:property ?action-designator (?arm ?arm))
18  (design:desig-prop ?action-designator (?surface ?surface-designator))
19  (design:current-designator ?surface-designator ?current-surface-designator)
20  (spec:property ?current-surface-designator (:type ?surface-type))
21  (spec:property ?surface (?current-surface-designator))
22  (> (equal ?grasp ?scrubbing)
23  (lisp-fun:differentiate-surface-types ?grasp ?surface ?current-grasp)
24  (equal ?grasp ?current-grasp))
25  (design:desig-prop ?action-designator (:collision-mode ?collision-mode))
26  (> (equal ?arm :left)
27
28
29
30
31  (and (lisp-fun:get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
32  (lisp-fun:man-int:get-traj-poses-by-label ?lists :wiping
33  ?left-wipe-poses)
34
35
36  (lisp-fun:man-int:get-traj-poses-by-label ?lists :initial
37  ?left-initial-poses))
38  (and (equal ?left-wipe-poses NIL)
39  (equal ?left-initial-poses NIL)))
40  (> (equal ?arm :right)
41  (and (lisp-fun:get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
42  (lisp-fun:man-int:get-traj-poses-by-label ?lists :wiping
43  ?right-wipe-poses)
44  (lisp-fun:man-int:get-traj-poses-by-label ?lists :initial
45  ?right-initial-poses))
46  (and (equal ?right-wipe-poses NIL)
47  (equal ?right-initial-poses NIL)))
48  (design:designator :action (:type :wiping)
49  (:collision-mode ?collision-mode)
50  (:left-wipe-poses ?left-wipe-poses)
51  (:right-wipe-poses ?right-wipe-poses)
52
53  (:left-initial-poses ?left-initial-poses)
54
55  (:right-initial-poses ?right-initial-poses)
56  ?resolved-action-designator))

```



```

1  <- (design:action-grounding ?action-designator (wipe ?arm
2  ?gripper-opening
3  ?distance
4  ?surface-designator
5  ?towel-designator
6  ?left-reach-poses
7  ?right-reach-poses
8  ?left-grasp-poses
9  ?right-grasp-poses
10  (?left-lift-poses)
11  (?left-2nd-lift-poses)
12  (?right-lift-poses)
13  (?right-2nd-lift-poses)
14  ?joint-name ?environment-object))
15  (spec:property ?action-designator (:type :opening))
16  (spec:property ?action-designator (:grasp ?surface-designator))
17  (spec:property ?surface-designator (:type ?surface-type))
18  (obj-int-object-type-subtype ?surface ?surface-type))
19  (spec:property ?surface-designator (:url-name ?surface-name))
20  (spec:property ?surface-designator (:part-of ?btr-environment))
21  (spec:property ?action-designator (:tool ?towel-designator))
22  (spec:property ?towel-designator (:type :towel))
23  (spec:property ?towel-designator (:url-name ?towel-name))
24  (spec:property ?towel-designator (:part-of ?btr-environment))
25
26  (> (spec:property ?action-designator (:arm ?arm))
27  (true)
28  (and (cram-robot-interfaces:robot ?robot)
29  (cram-robot-interfaces:arm ?robot ?arm)))
30  (spec:property ?action-designator (:distance ?distance))
31  (lisp-fun:get-surface-link ?surface-name ?btr-environment ?surface-link)
32  (lisp-fun:get-connecting-joint ?surface-link ?connecting-joint)
33  (lisp-fun:c-lurl-name ?connecting-joint ?joint-name)
34  (btr:bullet-world ?world)
35  (lisp-fun:btr-object ?world ?btr-environment ?environment-object)
36  (lisp-fun:obj-int:get-object-type:gripper-opening ?towel-designator ?gripper-opening)
37  (lisp-fun:get-surface-poses-and-transform ?surface-name ?btr-environment
38  ?surface-poses ?surface-transform))
39  (lisp-fun:obj-int:get-object-grasping-poses ?towel-name
40  ?towel ?left :open ?surface-transform ?left-poses)
41
42  (lisp-fun:obj-int:get-object-grasping-poses ?towel-name
43  ?towel ?right :open ?surface-transform ?right-poses)
44  (lisp-fun:cram-mobile-pick-place-plans:extract-pick-up-manipulation-poses
45  ?arm ?left-poses ?right-poses
46  ?left-reach-poses ?right-reach-poses
47  ?left-grasp-poses ?right-grasp-poses
48
49  ?left-lift-poses ?right-lift-poses))
50  (> (lisp-pred:identity ?left-lift-poses)
51  (equal ?left-lift-poses (?left-lift-poses ?left-2nd-lift-poses)))
52  (equal (NIL NIL) (?left-lift-poses ?left-2nd-lift-poses)))
53
54  (> (lisp-pred:identity ?right-lift-poses)
55  (equal ?right-lift-poses (?right-lift-poses ?right-2nd-lift-poses)))
56  (equal (NIL NIL) (?right-lift-poses ?right-2nd-lift-poses)))

```

Figure 68: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Opening* (52 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
37	20	15	0

Findings:

- designator definition contains 15 instead of only one variable
- some minor renaming (e.g. *cleaning* instead of *wiping*)
- designator calculates trajectories and poses in a different manner
- designator does not end by mapping the variables to designator "parts"

```

1  (c- (desig action-grounding ?action-designator (wipe ?surface-desig ?towel-desig ?arm
2    (spec.property ?action-designator (:type .wiping))
3    (spec.property ?action-designator (:grasp ?grasp))
4
5    (spec.property ?action-designator (:arm ?arm))
6    (desig desig-prop ?action-designator (?surface ?surface-designator))
7    (spec.current-designator ?surface-designator ?current-surface-designator)
8    (spec.property ?current-surface-designator (?type ?surface-type))
9    (equal ?surface ?current-surface-designator))
10   (spec.property ?action-designator (?surface ?surface))
11   (desig desig-prop ?action-designator (?surface ?surface))
12   (spec.property ?surface (?type ?surface-type))
13   (equal ?surface ?surface))
14   (spec.property ?action-designator (?collision-mode ?collision-mode))
15   (and (equal ?arm :left)
16     (and (lisp-fun get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
17       (lisp-fun man-int-get-traj-poses-by-label ?lists :swiping
18         ?left-wipe-poses)
19       (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
20         ?left-initial-poses)))
21     (and (equal ?left-wipe-poses NIL)
22       (equal ?left-initial-poses NIL))
23     (and (equal ?arm :right)
24       (and (lisp-fun get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
25         (lisp-fun man-int-get-traj-poses-by-label ?lists :swiping
26           ?right-wipe-poses)
27         (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
28           ?right-initial-poses)))
29     (and (equal ?right-wipe-poses NIL)
30       (equal ?right-initial-poses NIL))
31     (desig.designator.action (:type :wiping)
32       (?collision-mode ?collision-mode)
33       (?left-wipe-poses ?left-wipe-poses)
34       (?right-wipe-poses ?right-wipe-poses)
35       (?left-initial-poses ?left-initial-poses)
36       (?right-initial-poses ?right-initial-poses))
37     (?resolved-action-designator)))
38
39
40
41

```



```

1  (c- (desig action-grounding ?action-designator (wipe ?surface-desig ?towel-desig ?arm
2    (spec.property ?action-designator (?upper-opening ?effort ?grasp
3      ?left-reach-poses ?right-reach-poses
4      ?left-grasp-poses ?right-grasp-poses
5      ?left-wipe-poses ?right-wipe-poses))
6    (spec.property ?action-designator (?surface ?surface-designator))
7    (spec.current-designator ?surface-designator ?surface-desig)
8    (spec.property ?surface-desig (?type ?surface-type))
9    (spec.property ?surface-desig (?name ?surface-name))
10   (spec.property ?action-designator (?tool ?towel-designator))
11   (desig.current-designator ?towel-designator ?towel-desig)
12   (spec.property ?towel-desig (?type ?towel))
13   (spec.property ?action-designator (?arm ?arm))
14   (true)
15   (and (cram-robot-interfaces:robot ?robot)
16     (cram-robot-interfaces:arm ?robot ?arm)))
17   (lisp-fun obj-int:get-surface-transform ?surface-desig ?surface-transform)
18
19   (lisp-fun obj-int:get-surface-faces ?surface-transform (?facing-robot-face ?bottom-face))
20   (> (spec.property ?action-designator (?grasp ?grasp))
21     (true)
22     (and (lisp-fun obj-int:get-towel-type-grasps
23       ?towel-desig ?facing-robot-face ?bottom-face ?arm
24       ?grasps)
25       (member ?grasp ?grasps)))
26   (lisp-fun obj-int:get-towel-type-gripping-effort ?towel-desig ?effort)
27   (lisp-fun obj-int:get-towel-type-gripper-opening ?towel-desig ?gripper-opening)
28
29   (lisp-fun obj-int:get-towel-wiping-poses
30     ?towel-desig ?grasp ?surface-transform
31     :left ?left-wipe-poses
32     :right ?right-wipe-poses)
33
34
35
36
37   (lisp-fun extract-wipe-manipulation-poses ?arm ?left-wipe-poses ?right-wipe-poses
38     (?left-reach-poses ?right-reach-poses
39       ?left-grasp-poses ?right-grasp-poses
40       ?left-wipe-poses ?right-wipe-poses)))
41

```

Figure 69: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Picking Up* (35 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
21	21	14	0

Findings:

- tba

```

1  <- (desig-action-grounding ?action-designator wipe ?resolved-action-designator)
2   (spec-property ?action-designator :type :wiping)
3   (spec-property ?action-designator :?grasp)
4   (spec-property ?action-designator :arm ?arm)
5   (design-desig-prop ?action-designator :surface ?surface-designator)
6   (design-current-designator ?surface-designator ?current-surface-designator)
7   (spec-property ?current-surface-designator :type ?surface-type)
8   (equal ?surface (?current-surface-designator))
9   (-> ?grasp :scrubbing)
10  (lisp-fun differentiate-surface-type ?grasp ?surface ?current-grasp)
11  (equal ?grasp ?current-grasp)
12  (design-desig-prop ?action-designator :collision-mode ?collision-mode)
13  (-> (equal ?arm :left)
14   (and lisp-fun get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
15   (lisp-fun man-int-get-traj-poses-by-label ?lists :wiping
16     ?left-wipe-poses)
17   (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
18     ?left-initial-poses))
19   (and (equal ?left-wipe-poses NIL)
20     (equal ?left-initial-poses NIL)))
21  (-> (equal ?arm :right)
22   (and lisp-fun get-trajectory :wiping ?arm ?current-grasp T ?surface ?lists)
23   (lisp-fun man-int-get-traj-poses-by-label ?lists :wiping
24     ?right-wipe-poses)
25   (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
26     ?right-initial-poses))
27   (and (equal ?right-wipe-poses NIL)
28     (equal ?right-initial-poses NIL)))
29  (design-designator :action ((type :wiping)
30    (collision-mode ?collision-mode)
31    (left-wipe-poses ?left-wipe-poses)
32    (right-wipe-poses ?right-wipe-poses)
33    (left-initial-poses ?left-initial-poses)
34    (right-initial-poses ?right-initial-poses)))
35    ?resolved-action-designator)

```



```

1  <- (desig-action-grounding ?action-designator (wipe ?surface-designator ?towel-designator))
2   (spec-property ?action-designator :type :cleaning)
3   (spec-property ?surface-designator ?surface ?surface-designator)
4   (spec-property ?towel-designator ?tool ?towel-designator)
5
6   (design-current-designator ?surface-designator ?current-surface-designator)
7   (spec-property ?current-surface-designator :type ?surface-type)
8   (spec-property ?current-surface-designator :name ?surface-name)
9   (design-current-designator ?towel-designator ?current-towel-designator)
10  (spec-property ?current-towel-designator :type ?towel-type)
11
12  (spec-property ?current-towel-designator :name ?towel-name)
13  (and (cram-robot-interfaces :robot ?robot)
14    (cram-robot-interfaces :arm ?robot ?arm)
15    (cram-robot-interfaces :gripper ?robot ?gripper)
16    (cram-object-in-hand ?towel-designator ?gripper)
17    (cram-object-in-hand ?towel-designator ?arm))
18  (lisp-fun obj-int-get-surface-transform ?current-surface-designator ?surface-transform)
19  (lisp-fun obj-int-get-object-transform ?current-towel-designator ?towel-transform)
20  (lisp-fun obj-int-get-object-pose ?current-towel-designator ?towel-pose)
21  (lisp-fun cram-robot-interfaces :arm-k ?arm ?towel-poses ?towel-transform ?ik-solution)
22  (lisp-fun cram-robot-interfaces :arm-move ?arm ?ik-solution)
23  (lisp-fun cram-robot-interfaces :gripper-close ?gripper)
24  (lisp-fun cram-robot-interfaces :gripper-move ?gripper :position 0.1)
25  (lisp-fun cram-robot-interfaces :gripper-move ?gripper :position 0.0)
26  (lisp-fun cram-robot-interfaces :gripper-open ?gripper)
27
28
29
30
31
32
33
34
35

```

Figure 70: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Placing Down* (25 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
17	27	6	2

Findings:

- designator definition contains two instead of only one variable
- some minor renaming (e.g. *cleaning* instead of *wiping*)
- generation adds block about **cram-robot-interfaces** (3 lines)
- designator contains only definitions but no calculations
- designator does not end by mapping the variables to designator "parts"

Figure 71: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Pouring* (53 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
24	6	25	2

Findings:

- some minor renaming (e.g. *cleaning* instead of *wiping*)
 - designator calculates trajectories and poses in a different manner
 - generated designator extends the mapping of variables to designator "parts" (adds *:object*, *:object-type*, *:object-name*, *:arms* and *:grasp*)

```

1  (<- (desig.action-grounding ?action-designator (wipe ?resolved-action-designator))
2    (spec.property ?action-designator (:type :wiping))
3    (spec.property ?action-designator (:grasp ?grasp))
4    (spec.property ?action-designator (:arm ?arm))
5    (spec.property ?surface ?surface-designator)
6    (spec.property ?surface-designator ?surface ?current-surface-designator)
7    (spec.property ?current-surface-designator (:type ?surface-type))
8    (equal ?surface (?current-surface-designator))
9
10
11  (> (equal ?grasp :scrubbing)
12    (lisp-fun differentiate-surface-types ?grasp ?surface ?current-grasp)
13    (equal ?grasp ?current-grasp)))
14  (design desig-prop ?action-designator (:collision-mode ?collision-mode))
15
16
17
18
19  (> (equal ?arm :left)
20    (and (lisp-fun get-trajectory :swiping ?arm ?current-grasp T ?surface ?lists)
21      (lisp-fun man-int-get-traj-poses-by-label ?lists :swiping
22        ?left-wipe-poses)
23        (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
24          ?left-initial-poses))
25        (and (equal ?left-wipe-poses NIL)
26          (equal ?left-initial-poses NIL)))
27
28  (> (equal ?arm :right)
29    (and (lisp-fun get-trajectory :swiping ?arm ?current-grasp T ?surface ?lists)
30      (lisp-fun man-int-get-traj-poses-by-label ?lists :swiping
31        ?right-wipe-poses)
32        (lisp-fun man-int-get-traj-poses-by-label ?lists :initial
33          ?right-initial-poses))
34        (and (equal ?right-wipe-poses NIL)
35          (equal ?right-initial-poses NIL)))
36
37  (desig designator :action ((:type :swiping)
38    (:collision-mode ?collision-mode))
39
40
41
42
43
44    (:left-wipe-poses ?left-wipe-poses)
45    (:right-wipe-poses ?right-wipe-poses)
46    (:left-initial-poses ?left-initial-poses)
47    (:right-initial-poses ?right-initial-poses))
48  ?resolved-action-designator)

```



```

1  (<- (desig.action-grounding ?action-designator (wipe ?resolved-action-designator))
2    (spec.property ?action-designator (:object ?object-designator))
3    (spec.property ?object-designator ?object-designator ?current-object-design)
4    (spec.property ?current-object-design (:type ?object-type))
5    (spec.property ?current-object-design (:name ?object-name))
6    (true)
7    (man-int:robot-free-hand ?_ ?arm))
8
9  (> (spec.property ?action-designator (:arm ?arm))
10    (true)
11    (and (lisp-fun man-int-get-action-grasps ?object-type ?arm ?grasps)
12      (member ?grasp ?grasps)))
13    (lisp-fun man-int-get-action-gripping-effort ?object-type ?effort)
14    (lisp-fun man-int-get-action-gripper-opening ?object-type ?gripper-opening)
15    (equal ?objects (?current-object-design))
16    (true)
17    (and (lisp-fun man-int-get-action-trajectory :cleaning ?arm ?grasp T ?objects
18      ?left-wiping-poses)
19        (lisp-fun man-int-get-traj-poses-by-label ?left-wiping-poses :wipe
20          ?left-wipe-poses))
21        (lisp-fun man-int-get-traj-poses-by-label ?right-wiping-poses :wipe
22          ?right-wipe-poses))
23        (equal ?left-wipe-poses NIL)
24        (equal ?right-wipe-poses NIL))
25  (> (equal ?arm :right)
26    (and (lisp-fun man-int-get-action-trajectory :cleaning ?arm ?grasp T ?objects
27      ?right-wiping-poses)
28        (lisp-fun man-int-get-traj-poses-by-label ?right-wiping-poses :wipe
29          ?right-wipe-poses)))
30        (true)
31    (desig designator :action ((:type :cleaning)
32      (:object ?current-object-design)
33      (:object-name ?object-name)
34      (:arm ?arm)
35      (:gripper-opening ?gripper-opening)
36      (:effort ?effort)
37      (:grasp ?grasp)
38      (:left-wipe-poses ?left-wipe-poses)
39      (:right-wipe-poses ?right-wipe-poses)
40      (:collision-mode ?collision-mode)))
41
42
43
44
45
46
47
48  ?resolved-action-designator)

```

Figure 72: Left: Manual designator for the action *Wiping* (35 lines). Right: Generated designator based on *Slicing* (42 lines).

Added Lines	Deleted Lines	Changed Lines	Unchanged Lines
19	12	17	6

Findings:

- some minor renaming (e.g. *cleaning* instead of *wiping*)
- designator calculates trajectories and poses in a different manner
- generated designator extends the mapping of variables to designator "parts" (misses *:left-initial-poses* and *:right-initial-poses* but adds *:object*, *:object-name*, *:arm*, *:gripper-opening*, *:effort* and *:grasp*)