Technical Appendix

Step-by-Step Example of Learning One Production Rule

This section shows an example of learning a new production rule for `slice a/an <object>` task. In the interest of space, only relevant information is kept. The original complete prompt, along with the responses from the LLM, are provided in the code and data supplementary material.

System prompt for action selection and production description in English The system prompt mainly describes the robot's affordance model and explains the input of future user prompts. It is the same prompt for all action selections.

User prompt for action selection As mentioned in the methods section, the production generation is grounded to a specific instance, and the LLM is first asked to choose an action. The user prompt for action selection has a fixed template, where the information will be dynamically filled according to the actual knowledge of the agent. Below is an example of the information provided in the prompt. Unimportant information and static instructions are omitted in the interest of space. The user prompt below and the system prompt will be provided to the LLM in the same request for the action selection.

```
[Current Task] slice a/an lettuce

[Current Location] in front of SinkBasin_28084e25

[Spatial Knowledge]
 * (0.0 meters away) RobotGripper(Gripper) has Lettuce_895e9ec5(Lettuce), and
nothing  else
 * (0.0 meters away) SinkBasin_28084e25(SinkBasin) has been fully explored: it has
Cup_26e78d79(Cup), Egg_113844f2(Egg), and nothing else
 * (0.9 meters away) CounterTop4(CounterTop) has been explored: it is empty
 * <<OTHER RECEPTACLES OMITTED>>

[Object Knowledge]
 * Apple_969aaa5d(Apple) at Fridge_683c888d
 * Cabinet_2fee59ce(Cabinet): opened,
 * Faucet_63e88a3b(Faucet):turned off,
 * Lettuce_895e9ec5(Lettuce) at RobotGripper
 * <<OTHER OBJECTS OMITTED>>

[Previous Tasks]
 * go to explore a/an sinkbasin_28084e25: True
 * find a/an lettuce: True
 * <<OTHER TASKS OMITTED>>

[Action History]
 * <<OTHER HISTORY OMITTED>>
 * (time 20) motor action: pick up lettuce (purpose: IF the current task is to find
a/an <object> AND the <object> is located in a/an <receptacle> AND the robot is in
front of the <receptacle> AND the robot's gripper is empty THEN choose motor action:
pick up <object>.)
 * (time 21) special action: done (purpose: IF the current task is to find a/an
<object> AND the robot's gripper has <object> THEN choose special action: 'done'.)

[Possible Options]
 * attend to subtask: find a/an <object> (Apply anytime. End condition: the robot
has found the object and has it in its gripper.)
 * motor action: put Lettuce_895e9ec5 on CounterTop4
 * special action: 'done'
 * <<OTHER OPTIONS OMITTED>>

[Blacklisted Options]
 * attend to subtask: slice a/an lettuce
```

GPT4 response for action selection The response of GPT4 to the prompt above is recreated below.

```
[Current Task]
Slice a lettuce.

[Common Strategies for "slice a lettuce"]
 * Find a lettuce and a knife.
 * Pick up the lettuce.
 * Move to a countertop or a similar flat surface.
 * Put down the lettuce.
 * Pick up the knife.
 * Slice the lettuce.

[Goal Conditions]
 * The lettuce is sliced.

[Current Situation Analysis]
The robot is currently holding a lettuce and is located in front of the sink. The
robot has explored several areas of the kitchen, including the sink, countertop4,
countertop5, and a few cabinets. However, there are still several unexplored areas,
including countertop1, countertop2, countertop3, and several cabinets. The robot has
not found a knife yet, which is necessary for slicing the lettuce. The robot's
previous actions indicate that it has successfully found and picked up the lettuce,
which is part of the task. There is no existing plan provided.

[Task Analysis]
The current task is to slice a lettuce. The robot has already found and picked up
the lettuce, which is part of the task. However, the robot has not found a knife
yet, which is necessary for slicing the lettuce. The robot also needs to find a
suitable place to put down the lettuce for slicing, such as a countertop.

[Plan]
Step1: Put down the lettuce on a countertop.
Step2: Find and pick up a knife.
Step3: Slice the lettuce.

[Option Evaluation]
"attend to subtask: find a/an <object>": This option is not useful right now because
the robot already has the lettuce in its gripper.
"motor action: put Lettuce_895e9ec5 on CounterTop4": This option is useful because
it allows the robot to free its gripper so it can pick up a knife.
"special action: 'done'": This option is not appropriate because the task is not
completed yet.
<<OTHER EVALUATION OMITTED>>

[Option Suggestion]
"motor action: put Lettuce_895e9ec5 on CounterTop4"

[Purpose]
The purpose of the suggested option is to free the robot's gripper so it can pick up
a knife.

[End]
```

User prompt for generating English description of the production rule After parsing the suggested option from the previous response, we prompt the LLM again for the English description of the production rule. We combine all the previous queries and responses and send all of them to the LLM in a single request.

GPT4 response for production rule in English

```
[Relevant Information]
 * The current task is to slice a lettuce.
 * The robot is currently holding a lettuce in its gripper.
```

```
 * The robot is currently in front of the sink, which is not an ideal place for
slicing.
 * A countertop is a suitable place for slicing.
 * The robot has the option to put the lettuce on CounterTop4.

[Specific Rule]
IF the current task is to slice a lettuce AND the robot is holding a lettuce in its
gripper AND the robot is not at a suitable place for slicing AND there is a
countertop available THEN choose motor action: put Lettuce_895e9ec5 on CounterTop4.

[Generalizable Constants]
 * Lettuce_895e9ec5 could be any object that needs to be sliced.
 * CounterTop4 could be any countertop or similar flat surface.

[Generalized Rule]
IF the current task is to slice a/an <object> AND the robot is holding the <object>
in its gripper AND the robot is not at a suitable place for slicing AND there is a
countertop available THEN choose motor action: put <object> on <countertop>.

[Correspondence]
 * <object> is Lettuce_895e9ec5.
 * <countertop> is CounterTop4.

Notes: This rule is designed to ensure that the robot places the object to be sliced
on a suitable surface before attempting to slice it. It is applicable to any task
that involves slicing an object.
```

System prompt for production rule code generation The system prompt mainly describes the Python interfaces for the productions and information stored in the agent. It is the same prompt for all code generation.

User prompt for production rule code generation The response of the production rule in English is copied verbatim along with the instructions.

GPT4 response for production code generation The response generated by GPT4 is recreated below. The code has been reformatted to fit the page.

```
[Variable Bindings]
 * target_object: the object that needs to be sliced. Its value should be parsed
from the current task.
 * target_receptacle: a suitable place for slicing. Its value should be assigned by
iterating through all receptacles and finding the countertop that is closest to the
robot.

[Precondition Relation]
 * "the current task is to slice a/an <object>" is about the current task
 * "the robot is holding the <object> in its gripper" is about the object type of
target_object and the hosting attribute of the robot's gripper
 * "the robot is not at a suitable place for slicing" is about the current location
 * "there is a countertop available" is about the existence of target_receptacle

[Precondition Test]
 * The current task is to slice a/an <object> can be tested by checking if
`re.match(r"slice a/an (.+)", current_task)`
 * The robot is holding the <object> in its gripper can be tested by checking if
`spatial_knowledge["RobotGripper"].hosting and name_equal(
spatial_knowledge["RobotGripper"].hosting[0].object_type, target_object)`
 * The robot is not at a suitable place for slicing can be tested by checking if
`not OracleQuery(f"{current_location} is a suitable place for slicing")`
 * There is a countertop available can be tested by checking if `target_receptacle
is not None`
```

[Python Implementation]
```py
import re
from typing import Dict, Tuple
from .knowledge import SpatialKnowledge, ObjectKnowledge

class SliceObject:

    target_task: str = "slice a/an <sliceable>"

    def precondition(
        self,
        current_task: str,
        current_location: str,
        previous_tasks: Dict[str, bool],
        spatial_knowledge: SpatialKnowledge,
        object_knowledge: ObjectKnowledge,
    ) -> Tuple[bool, str]:
        match = re.match(r"slice a/an (.+)", current_task)
        if not match:
            return False, "The current task is not to slice an object"
        target_object = match.group(1)

        if not spatial_knowledge["RobotGripper"].hosting or not name_equal(
            spatial_knowledge["RobotGripper"].hosting[0].object_type,
            target_object,
        ):
            return (
                False,
                "The robot is not holding the target object in its gripper",
            )

        if OracleQuery(f"{current_location} is a suitable place for slicing"):
            return False, "The robot is already at a suitable place for slicing"

        target_receptacle = min(
            (
                r for r in spatial_knowledge.locations.values()
                if name_equal(r.object_type, "CounterTop")
            ),
            key=spatial_knowledge.get_distance
        )
        if target_receptacle is None:
            return False, "There is no countertop available"

        setattr(self, "target_object", target_object)
        setattr(self, "target_receptacle", target_receptacle.object_name)
        return True, ""

    def apply(self) -> str:
        return f"motor action: put {self.target_object} on {self.target_receptacle}"
```

User prompt for code revision As described in the production improvement section, if the code generated does not pass the current state knowledge, we query it again with the current knowledge and the error raised, which in this example is the world knowledge cannot determine whether "SinkBasin_28084e25 is a suitable place for slicing" is true or not.

```
<<INFORMATION FROM THE ACTION SELECTION STEP>>

[Expected Return of `precondition()`]
```

```
(True, "")

[Your Code Return]
ValueError: got unknown statement: SinkBasin_28084e25 is a suitable place for
slicing
```

GPT4 response for code revision

```
[Error Interpretation]
The message indicates that the OracleQuery function has failed.

[Error Attribution]
The error is because the OracleQuery function does not understand the statement
"SinkBasin_28084e25 is a suitable place for slicing". The OracleQuery function is
designed to answer general knowledge questions, not specific queries about the
current environment.

[Things to Change]
The OracleQuery test should be replaced with a check against a list of suitable
places for slicing.

[Python Implementation]
```py
import re
from typing import Dict, Tuple
from .knowledge import SpatialKnowledge, ObjectKnowledge

class SliceObject:

    target_task: str = "slice a/an <sliceable>"

    def precondition(
        self,
        current_task: str,
        current_location: str,
        previous_tasks: Dict[str, bool],
        spatial_knowledge: SpatialKnowledge,
        object_knowledge: ObjectKnowledge,
    ) -> Tuple[bool, str]:
        match = re.match(r"slice a/an (.+)", current_task)
        if not match:
            return False, "The current task is not to slice an object"
        target_object = match.group(1)

        if not spatial_knowledge["RobotGripper"].hosting or not name_equal(
            spatial_knowledge["RobotGripper"].hosting[0].object_type,
            target_object,
        ):
            return (
                False,
                "The robot is not holding the target object in its gripper",
            )

        suitable_places_for_slicing = ["CounterTop"]
        if not any(
            name_equal(current_location, place)
            for place in suitable_places_for_slicing
        ):
            return False, "The robot is not at a suitable place for slicing"
```
```

```
        target_receptacle = min(
            (
                r for r in spatial_knowledge.locations.values()
                if name_equal(r.object_type, "CounterTop")
            ),
            key=spatial_knowledge.get_distance,
        )
        if target_receptacle is None:
            return False, "There is no countertop available"

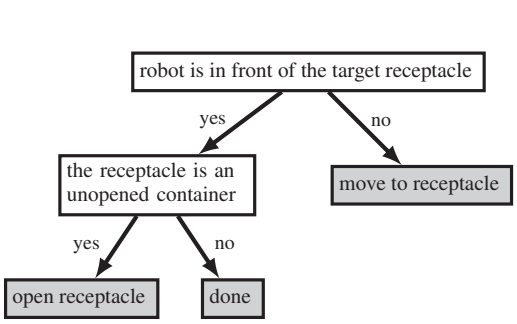
        setattr(self, "target_object", target_object)
        setattr(self, "target_receptacle", target_receptacle.object_name)
        return True, ""

    def apply(self) -> str:
        return f"motor action: put {self.target_object} on {self.target_receptacle}"
```

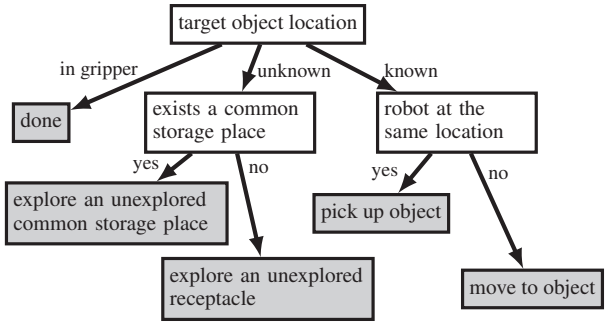
Production Rules Generated

Figures 5 and 6 illustrate the production rules learned for each task in the form of decision trees. These decision trees are for illustrations purpose and are not completely equivalent to the agent because of the sampling mechanism in the production selection process. The order of the nodes is chosen based on the utility of the production. That is, productions with higher utility (e.g., productions leading to the `done` action) will be closer to the root. This demonstrates how the production rules can be converted to verifiable decision trees as mentioned in the discussion section in the main paper.

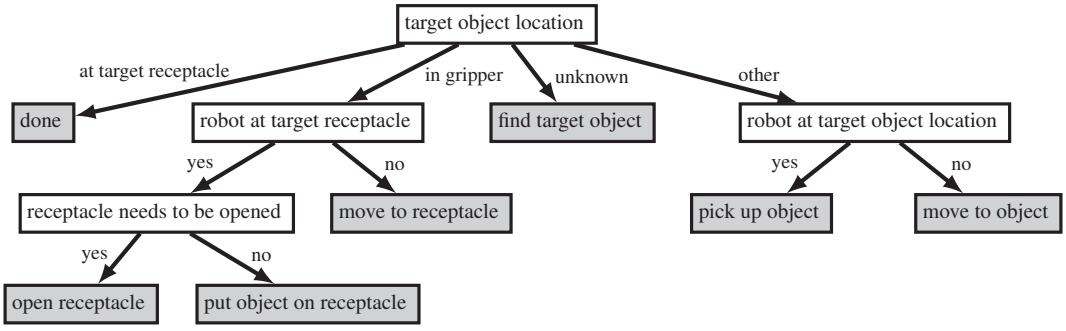
The actual production rules and their Python implementations can be found in the code and data supplementary material.



(a) explore

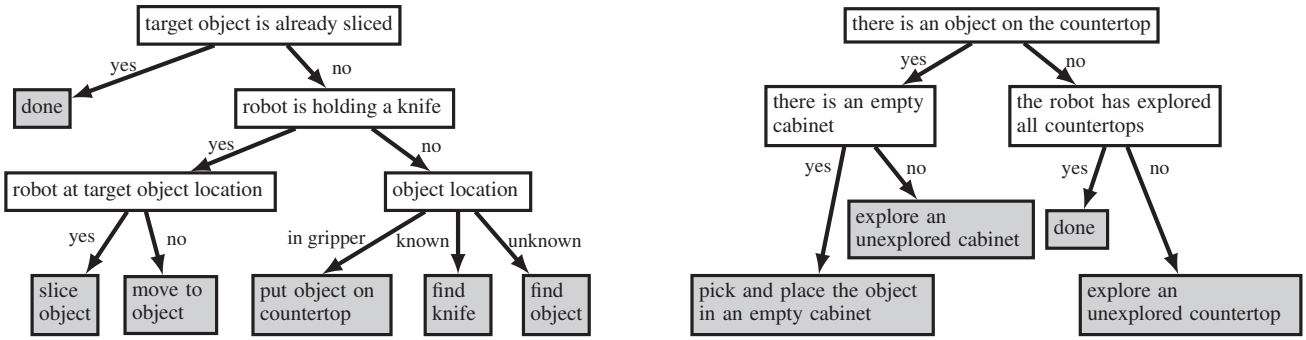


(b) find



(c) pick and place

Figure 5: Productions learned through bootstrapping for exploring, finding, and placing. Gray nodes are the effects, and white nodes are the features being conditioned on.



(a) slice

(b) put things away

Figure 6: Productions learned through bootstrapping for slicing and putting things away. Gray nodes are the effects, and white nodes are the features being conditioned on.

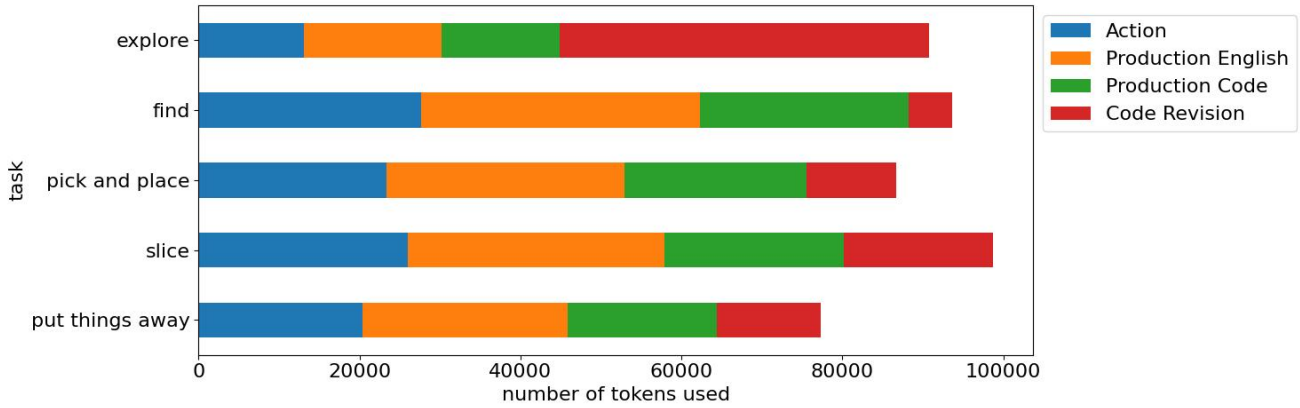


Figure 7: Number of tokens used during bootstrapping

Tokens Usage

As shown in Figure 7, the number of tokens needed to train each task is roughly the same. So as the curriculum expands, the number of tokens needed will only grow linearly. Additionally, the number of tokens needed to train one task is less than one single trial of slicing objects of the action-only agent as reflected in Table 1. This shows that our framework is much more cost-effective. The testing experiments on the baseline action-only agent cost around $120 in total while the bootstrapping of our framework costs less than $40.

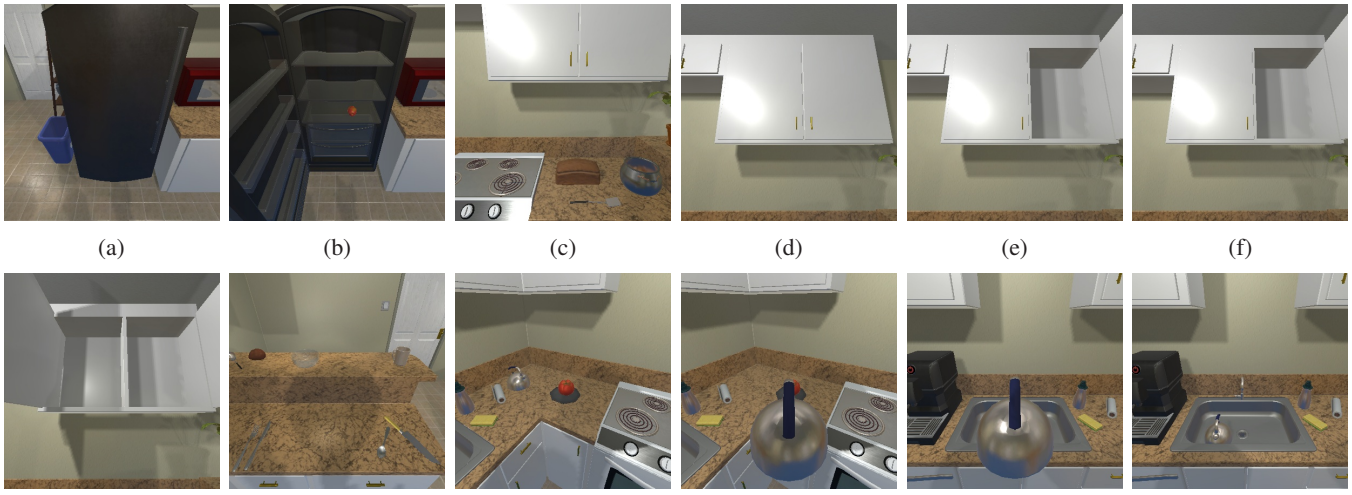
Step-by-Step Example of Completing One Task

Figure 8 and Table 2 shows the trajectory of the agent completing the task of "pick up and place a/an kettle in/on a/an sinkbasin" after bootstrapping. The agent first attends to the subtask of finding a kettle, during which process it also uses the explore subtasks, and finally moves to the sink basin and places the kettle as instructed. The main task column in the table reflects the management of the task stack in the agent: it attends to a single main task at a time and releases it when a production rule determines the current task is done.

Task End Conditions Generated

Here is a list of end conditions for the tasks families in our curriculum.

- `explore a/an <receptacle>`: "the robot has fully explored the receptacle."
- `find a/an <object>`: "the robot has found the object and has it in its gripper."
- `pick up and place a/an <object> in/on a/an <receptacle>`: "the robot has successfully picked up the specified object and placed it in/on the specified receptacle, and the robot's gripper is empty."
- `slice a/an <sliceable>`: "the sliceable object is already sliced and the robot's gripper is holding a knife."



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 8: Trajectory of the agent completing "pick up and place a/an kettle in/on a/an sinkbasin"

- `put things on the countertops away`: "all objects on the countertops have been put away in the cabinets and there are no more unexplored countertops or cabinets."

They might not be fully aligned with the human's intention (e.g., someone may think having an object in view already satisfies the goal of "find", but the agent believes the task is not done until it picks the object up), but they reflect what the agent would do if the subtask is chosen. This is very helpful for reusing previously learned tasks.

Reproducibility Checklist

We answer "partial" to the following question on the reproducibility checklist

> [Computational experiments] If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results.

This is because our experiments involve the use of GPT-4 and an Unity-based simulator, whose internal mechanism is not fully disclosed to the best of our knowledge. We have done our best to set the temperature of GPT-4 to 0, but empirical experiments and experiences from other users suggest that its behavior is still not deterministic when the temperature is set to 0. Additionally, there is physics simulation in the simulator (e.g., when slicing a lettuce, the slices will fall apart). These motions are not deterministic according to our observation, and the documentation of the simulator does not provide a way to make the result deterministic. Because the bootstrapping process contains multiple steps (~ 1400 steps), small discrepancies at the beginning may accumulate and result in very different production rules learned.

Despite we cannot guarantee whether anyone bootstrapping the agent from scratch will generate the same production rules as we do, we attach (in the code and data supplementary material) our bootstrapped production rules, the logs generated during the bootstrapping process, and the logs generated during the testing process such that one can use them to verify the results we reported in the experiments section of the main paper.

Main Task	Figure	Action	Production Preconditions
pick and place a/an kettle in/on a/an sinkbasin	8a	find a/an kettle	the robot's gripper is empty AND the <object> has not been located
find a/an kettle		open fridge	the robot is in front of a closed <receptacle> AND the robot's gripper is empty
	8b	move to countertop2	the robot's gripper is empty AND there is an unexplored <receptacle> that is commonly associated with the <object>
	8c	explore a/an cabinet1	the robot's gripper is empty AND there are unexplored receptacles in the kitchen
explore a/an cabinet1		move to cabinet1	the <receptacle> is not at the current location AND the robot's gripper is empty
	8d	open cabinet1	the robot is at the location of the <receptacle> AND the <receptacle> is closed and unexplored AND the robot's gripper is empty
	8e	done	the robot is in front of the <receptacle> AND the <receptacle> has been fully explored
find a/an kettle		explore a/an cabinet2	the robot's gripper is empty AND there are unexplored receptacles in the kitchen
explore a/an cabinet2		move to cabinet2	the <receptacle> is not at the current location AND the robot's gripper is empty
	8f	open cabinet2	the robot is at the location of the <receptacle> AND the <receptacle> is closed and unexplored AND the robot's gripper is empty
	8g	done	the robot is in front of the <receptacle> AND the <receptacle> has been fully explored
find a/an kettle		move to countertop1	the robot's gripper is empty AND there is an unexplored <receptacle> that is commonly associated with the <object>
	8h	move to countertop3	the robot's gripper is empty AND there is an unexplored <receptacle> that is commonly associated with the <object>
	8i	pick up kettle	the <object> is located in a/an <receptacle> AND the robot is in front of the <receptacle>
	8j	done	the robot's gripper has <object>
pick and place a/an kettle in/on a/an sinkbasin		move to sinkbasin	the robot's gripper has the <object> AND the robot is not at the <receptacle>
	8k	put on sinkbasin	the robot is holding the <object> AND the robot is in front of the <receptacle> AND the <receptacle> is empty
	8l	done	the <object> is already in the <receptacle> AND the robot's gripper is empty

Table 2: Action history of the agent completing "pick up and place a/an kettle in/on a/an sinkbasin". Due to space constraints, the object names are simplified and the task matching is omitted from the preconditions.