Planning and Scheduling at home project

Ángela Patricia Enríquez Gómez, Ethan Oswald Massey, Maximilian Mensing

January 15, 2019

BRSU

Introduction

Task

Storing Groceries

- **Case 1**: Everything is known. One object is on the table and has to be placed at any shelf after the cupboard has been opened.
- **Case 2**: The amount of Objects, the table and the cupboard have to be located. The objects have to be placed on the shelves.
- **Case 3**: As case 2 - In addition to not knowing the number of objects, the objects themself are also unknown.
- **Case 4**: In addition to case 3 the cupboard is unknown and has to be explored. The items have to be put in different categories and sorted by category on the shelf.

Selection of the Planner

NOAH

```
(puton
  (qlambda
    (on <-X <-Y)
     (pand
          (pgoal (clear $x) (cleartop $x)
               apply (clear))
          (pgoal (clear $y) (cleartop $y)
               apply (clear))
      )
     (pgoal (put $x on top of $y)
            (on $X $y) apply nil)
     (pdeny (cleartop $y))
  )
)
```

Nonlin

```
actschema puton
  pattern <<put $*x on top of $*y>>
  conditions
    holds <<cleartop $*x>> at self
    holds <<cleartop $*y>> at self
    holds <<on $*x $*z>> at self
  effects
    + <<on $*x $*y>>
    - <<cleartop $*y>>
    - <<on $*x $*z>>
    + <<cleartop $*z>>
  vars x undef y undef z undef;
end;
```

SIPE-2

```
operator: puton
arguments: block1, object1 is not block1;
purpose: (on block1 object1);
plot:
parallel
  branch 1: goals: (clear object1);
  branch 2: goals: (clear block1);
end parallel
process
action: puton.primitive;
arguments: block1, object1;
resources: block1;
effects: (on block1 object1);
end plot end operator
```

O-Plan2

```
schema puton;
 vars ?x=undef, ?y=undef, ?z=undef;
 expands {put ?x on top of ?y};
 only_use_for_effects
    {on ?x ?y}=true,
    {cleartop ?y}=false,
    {on ?x ?z}=false,
    {cleartop ?z}=true;
 conditions
    only_use_for_query {on ?x ?y}=true,
    achievable {cleartop ?y}=true,
    achievable {cleartop ?x}=true;
endschema;
```

UMCP

```
(operator puton (x y)
   :pre ((clear x)(on x table)(clear y))
   :post ((~on x table)(on x y)(~clear y))
)
```

SHOP2

```
(:operator (!puton ?x ?y)
  ((clear ?x) (on-table ?x) (clear ?y))
  ((clear ?y) (on-table ?x))
  ((on ?x ?y))
)
```

SIADEX

```
(:action puton
 :parameters (?x ?y - block)
 :precondition (and (grasping ?x)(clear ?y))
 :effect (and (not (grasping ?x))
         (not (clear ?y))(clear ?x)
         (on ?x ?y)(handempty))
)
```

Using JSHOP2

- Get jshop from `https://github.com/mas-group/jshop2`
- Set environment **export CLASSPATH="`pwd`/bin.build/JSHOP2.jar:`pwd`/antlr.jar:."**
- Compile using **make c**
- Run by calling **make problem1/2/3/4**

Limitations and Issues

Limitations

- Without sufficient prior information the planner is not able to classify objects
- In problem 4, if the shelves don't have example objects the planner has problems putting categories for them.
- Executing with java -ra generates all possible plans
  - High space and time complexity
  - Maximum of 4 objects to avoid outOfMemoryError

Diskussion

Offene Punkte