Planning-Notes

Nitin Yadav

July 2020

1 PDDL Translation to SAS using FD parser

```
(define (domain test)
  (:requirements :adl)
  (:predicates
    (a)
    (b)
    (c)
    (d)
    (e)
  )

  (:action action
    :parameters ()
    :precondition (a)
    :effect (and
      (when (b)(c))
      (when (c)(e))
    )
  )
)

(define (problem problem_test)
  (:domain test)

  (:init
    (a)
    (b)
  )

  (:goal
    (e)
  )
)
```

5 predicates mapped to 2 variables

Number of variables (c, e)

(dropped a, b, d)

```
2
begin_variable
var0
-1    : Not a derived variable
2     : takes 2 values; 0 and 1
Atom c()         var0 = 0 → c
NegatedAtom c()  var0 = 1 → ¬c
end_variable
begin_variable
var1             var1 denotes predicate e
-1
2
Atom e()         → First value is e true.
NegatedAtom e()
end_variable
0
begin_state      initial state has var0 = 1 and var1 = 1
1                ¬c, ¬e
1
end_state
begin_goal
1 : Number of goals
1 0    var1 = 0   (i.e. (e))
end_goal
1
begin_operator
action
0 : 0 preconditions   (a) is alway true
2 : Number of effects
0 0 -1 0
1 0 0 1 -1 0
0 : operator cost
end_operator
0
```

note ∅ means 'true'.

c

e

init state

goal

Listing 1: SAS encoding

effect 1

c

effect 2

effects var0

$\boxed{0 \quad 0 \quad -1 \quad 0}$ ← next value of var0

Number of effect conditions.

prev. value of var0 (-1 means any value) "don't care"

$\boxed{1 \quad \underbrace{0 \quad 0}_{} \quad \underbrace{1 \quad -1 \quad 0}_{}}$

1 effect condition

var0 = 0

effects var1 and sets it to 0

now (1) set initial condition to not (a).
(2) add action to make (a) true.

```
(define (domain test)
  (:requirements :adl)
  (:predicates
    (a)
    (b)
    (c)
    (d)
    (e)
  )

  (:action action
    :parameters ()
    :precondition (a)
    :effect (and
      (when (b)(c))
      (when (c)(e))
      )
  )


  (:action actionA
    :parameters ()
    :precondition (not (a))
    :effect (a)
  )
)

(define (problem problem_test)
  (:domain test)

  (:init
    (not a)
    (b)
  )

  (:goal
    (e)
  )
)
```

3 variables in SAS

var0 var1 var2
a c e

```
3
begin_variable
var0
-1
2
Atom a()
NegatedAtom a()
end_variable
begin_variable
var1
-1
2
Atom c()
NegatedAtom c()
end_variable
begin_variable
var2
-1
2
Atom e()
NegatedAtom e()
end_variable
0
begin_state
1
1
1
end_state
begin_goal
1
2 0
end_goal
2
begin_operator
action
1
0 0
2
0 1 -1 0
1 1 0 2 -1 0
0
end_operator
begin_operator
actiona
0
1
0 0 1 0
0
end_operator
0
```

a

c

e

action now has 1 precondition
: var0 = 0

2 effects

rest of action is same.

Listing 2: SAS encoding

change one conditional effect

```
(define (domain test)

    (:requirements :adl)

    (:predicates
        (a)
        (b)
        (c)
        (d)
        (e)
    )

    (:action action
        :parameters ()
        :precondition (a)
        :effect (and
            (when (not (e))(e))
            (when (c)(e))
        )
    )

    (:action actionA
        :parameters ()
        :precondition (not (a))
        :effect (a)
    )
)

(define (problem problem_test)

    (:domain test)

    (:init
        (not a)
        (b)
    )

    (:goal
        (e)
    )
)
```

```
2
begin_variable
var0
-1
2
Atom a()
NegatedAtom a()
end_variable
begin_variable
var1
-1
2
Atom e()
NegatedAtom e()
end_variable
0
begin_state
1
1
end_state
begin_goal
1
1 0
end_goal
2
begin_operator
action
1
0 0
1
0 1 -1 0
0
end_operator
begin_operator
actiona
0
1
0 0 1 0
0
end_operator
0
```

Listing 3: SAS encoding

¬ pre condition exists because initial state is ¬ a

2 effects collapsed to 1 by removing second effect

(e) will be true regardless.

Replace "and" with "oneof"

```
(define (domain test)

    (:requirements :adl)

    (:predicates
        (a)
        (b)
        (c)
        (d)
        (e)
    )

    (:action action
        :parameters ()
        :precondition (a)
        :effect (oneof
            (when (b)(c))
            (when (c)(e))
        )
    )

    (:action actionA
        :parameters ()
        :precondition (not (a))
        :effect (a)
    )
)

(define (problem problem_test)

    (:domain test)

    (:init
        (not a)
        (b)
    )

    (:goal
        (e)
    )
)
```

```
3
begin_variable
var0
-1
2
Atom a()
NegatedAtom a()
end_variable
begin_variable
var1
-1
2
Atom c()
NegatedAtom c()
end_variable
begin_variable
var2
-1
2
Atom e()
NegatedAtom e()
end_variable
0
begin_state
1
1
1
end_state
begin_goal
1
2 0
end_goal
3
begin_operator
action_DETDUP_0
1
0 0
1
0 1 -1 0
0
end_operator
begin_operator
action_DETDUP_1
1
0 0
1
1 1 0 2 -1 0
0
end_operator
begin_operator
actiona
0
1
0 0 1 0
0
end_operator
0
```

Listing 4: SAS encoding

operator copies each with unique conditional effect