## Planning-Notes

Nitin Yadav

July 2020

## Number of variables (e (dropped a, b,d) PDDL Translation to SAS using FD parser begin\_variable -1: Not a derived variable of News 2: takes 2 values; 0 and 1 Note 1 (define (domain test) (:requirements :adl) Atom c() $VarD = O \rightarrow C$ NegatedAtom c() $VarD = 1 \rightarrow TC$ (a) (c) end\_variable (d) begin\_variable varl denotes predicate e (e) 2 Atom e() -> First value is e true. (:action action :parameters () NegatedAtom e() :precondition (a) vend\_variable :effect (and initial state has varo=1 and (when (b)(c)) begin\_state 70,70 Var1=1 (when (c) (e)) end\_state begin\_goal 1: Number of goals 1: Number of goals 1: (e) ) (define (problem problem\_test) (:domain test) end\_goal 1 (:init begin\_operator 0:0 preconditions (a) is alway true 2: Number of effects (a) (b) (:goal 1 0 0 1 -1 0 (e) 0: operator cost end\_dperator effects var D ) Listing 1: SAS encoding OL next value of var D prev. value of varo Number of effect conditions. Vav0=0 CHECKS Var 1 and sets it

```
nowegset initial condition
                                                           3 variables in SAS
                                                                       varo var) var 2
          to not (a).
                                                    egin_variable
                                                                       a c
    (2) add action to make
                                                   var0
                                                                                          6
                                                   -1
                (a) frue.
                                                  2
                                                  Atom a()
                                                   NegatedAtom a()
(define (domain test)
                                                  fend_variable
 (:requirements :adl)
                                                   begin_variable
  (:predicates
                                                   var1
   (a)
                                                   -1
    (b)
    (C)
                                                  Atom c()
    (d)
                                                  NegatedAtom c()
    (e)
                                                   end_variable
                                                   begin_variable
                                                   var2
  (:action action
                                                   -1
   :parameters ()
   :precondition (a)
                                                  Atom e()
   :effect (and
                                                   NegatedAtom e()
     (when (b)(c))
                                                   end_variable
     (when (c)(e))
                                                  -0
     )
                                                   begin_state
 )
                                                   1
                                                   1
                                                   1
  (:action actionA
                                                   end_state
   :parameters ()
                                                   begin_goal
   :precondition (not (a))
                                                   1
   :effect (a)
                                                   2 0
 )
                                                   end_goal
)
                                                   action now has I precondition
(define (problem problem_test)
 (:domain test)
                                                   00 : var 0 = 0
                              2 effects
                                                                    rest of action is same.
  (:init
  (not a)
                                                   0 1 -1 0
    (b)-
                                                   1 1 0 2 -1 0
                                                   end_operator
  (:goal
                                                   begin_operator
   (e)
                                                   actiona
                                                   0
                                                   1
                                                   0 0 1 0
                                                   end_operator
```

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)
)
```

```
begin_variable
var0
-1
2
Atom a()
NegatedAtom a()
end_variable
begin_variable
var1
-1
2
Atom e()
NegatedAtom e()
end_variable
begin_state
1
1
end_state
begin_goal
1 0
end_goal
begin_operator
action
       pre condition exists because initial state is 7 a
            removing second effect

(C) will be the
end_operator
begin_operator
actiona
1
                   regrandless.
0 0 1 0
end_operator
```

Listing 3: SAS encoding

```
Replace and with "one of"
                                                         begin_variable
                                                         var0
                                                         -1
                                                         Atom a()
                                                         NegatedAtom a()
                                                         end_variable
                                                         begin_variable
(define (domain test)
                                                         var1
                                                         -1
    (:requirements :adl)
                                                         2
                                                         Atom c()
    (:predicates
                                                         NegatedAtom c()
        (a)
                                                         end_variable
        (b)
                                                         begin_variable
        (c)
                                                         var2
        (d)
                                                         -1
        (e)
                                                         2
                                                         Atom e()
                                                         NegatedAtom e()
    (:action action
                                                         end_variable
        :parameters ()
        :precondition (a)
                                                         begin_state
        :effect (oneof
                                                         1
            (when (b)(c))
                                                         1
            (when (c) (e))
                                                         1
                                                         end_state
    )
                                                         begin_goal
                                                         1
    (:action actionA
                                                         2 0
        :parameters ()
                                                         end_goal
        :precondition (not (a))
                                                                              operator copies
each with unique
conditional effect
                                                         3
        :effect (a)
                                                         begin_operator
                                                         action_DETDUP_0
)
                                                         0 0
(define (problem problem_test)
                                                         1
                                                         0 1 -1 0
    (:domain test)
                                                         0
                                                         end_operator
    (:init
                                                         begin_operator
        (not a)
                                                         action_DETDUP_1
        (b)
                                                         1
    )
                                                         0 0
                                                         1
    (:goal
                                                         1 1 0 2 -1 0
       (e)
                                                         end_operator
)
                                                         begin_operator
                                                         actiona
                                                         0
                                                         1
                                                         0 0 1 0
                                                         end_operator
```

Listing 4: SAS encoding