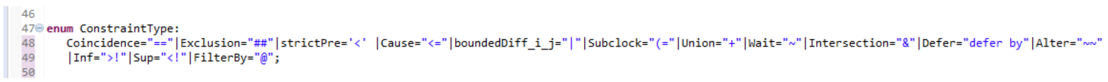
Step 0: timing constraints changes in PF-CCSL （snap from XText）

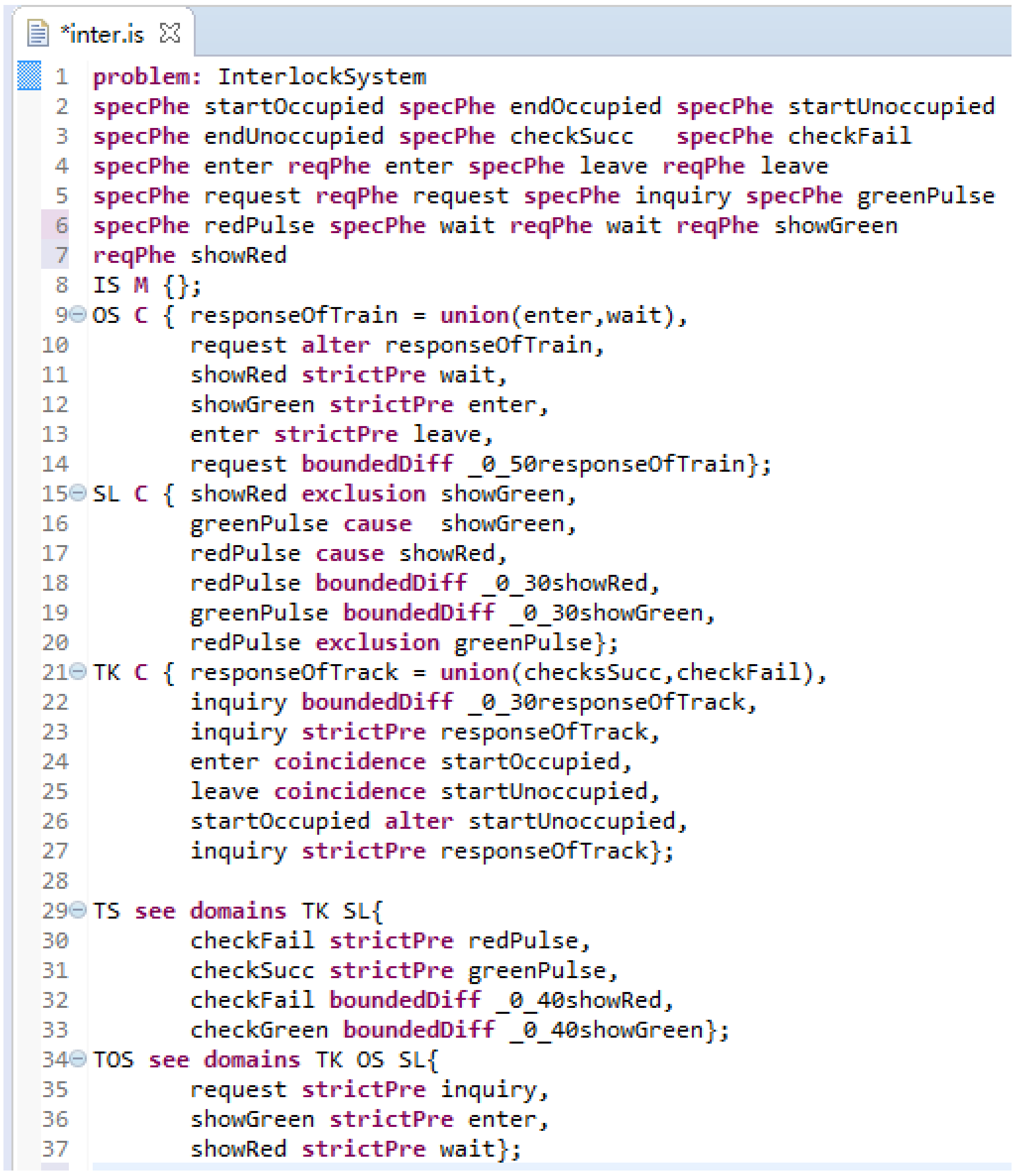


```
enum ConstraintType:
    Coincidence="=="|Exclusion="##"|strictPre='<' |Cause="<="|boundedDiff_i_j="|"|Subclock="(="|Union="+"|Wait="~"|Intersection="&"|Defer="defer by"|Alter="~~"
    |Inf=">!"|Sup="<!"|FilterBy="@";
```

# step 1: problem description of Interlocking System

1) When the train enters the track, it sends a request signal to the control center and waits for the traffic light signal.
2) After receiving the query result signal "result", the control center sends the detecting occupancy signal "checkoccupied" to the track to detect the occupation of the track.
3) After receiving the detecting occupancy signal, the track check whether the track is occupied. If the track is occupied, it sends the occupied signal "occupied" to the control center. If not occupied, it sends the unoccupied signal "unoccupied". When the train receiving the signal "trainEnter", the track is set to be occupied, and the track condition is not occupied when the train sends the train departure signal "trainLeave".
4) If the track is unoccupied, the control center sends the green pulse signal to the signal light. If the track is occupied, the control center sends the red pulse signal to the signal light.
5) The initial state of the signal Light is red light. After receiving the green pulse, it becomes a green light state. After receiving the red pulse, it becomes a red light state.
6) If the train sees the red light, it will wait. If the train sees the green light, it will enter, and finally leaves the station.
7) After the train sends the request, the train will wait or enter will be in 50 ms.
8) The response time of the light is 30 ms.
9) From checking the state of track to the set light, the time bound in 40 ms.

# step 2: Timing requirements and physical world description of Interlocking system in PF-CCSL



```
 1 problem: InterlockSystem
 2 specPhe startOccupied specPhe endOccupied specPhe startUnoccupied
 3 specPhe endUnoccupied specPhe checkSucc   specPhe checkFail
 4 specPhe enter reqPhe enter specPhe leave reqPhe leave
 5 specPhe request reqPhe request specPhe inquiry specPhe greenPulse
 6 specPhe redPulse specPhe wait reqPhe wait reqPhe showGreen
 7 reqPhe showRed
 8 IS M {};
 9 OS C { responseOfTrain = union(enter,wait),
10        request alter responseOfTrain,
11        showRed strictPre wait,
12        showGreen strictPre enter,
13        enter strictPre leave,
14        request boundedDiff _0_50responseOfTrain};
15 SL C { showRed exclusion showGreen,
16        greenPulse cause  showGreen,
17        redPulse cause showRed,
18        redPulse boundedDiff _0_30showRed,
19        greenPulse boundedDiff _0_30showGreen,
20        redPulse exclusion greenPulse};
21 TK C { responseOfTrack = union(checksSucc,checkFail),
22        inquiry boundedDiff _0_30responseOfTrack,
23        inquiry strictPre responseOfTrack,
24        enter coincidence startOccupied,
25        leave coincidence startUnoccupied,
26        startOccupied alter startUnoccupied,
27        inquiry strictPre responseOfTrack};
28
29 TS see domains TK SL{
30        checkFail strictPre redPulse,
31        checkSucc strictPre greenPulse,
32        checkFail boundedDiff _0_40showRed,
33        checkGreen boundedDiff _0_40showGreen};
34 TOS see domains TK OS SL{
35        request strictPre inquiry,
36        showGreen strictPre enter,
37        showRed strictPre wait};
```

# Step 3: change

Signal light changes:
The constraints in line 18, 19 changes to:

redPulse boundedDiff_0_20 showRed,
greenPulse boundedDiff_0_20 showGreen,

# step 4: model checking of satisfaction property

1) The model checking file .smv

MODULE main
VAR

```
startOccupied:boolean;
endOccupied:boolean;
startUnoccupied:boolean;
endUnoccupied:boolean;
checkSucc:boolean;
checkFail:boolean;
responseOfTrack:boolean;
enter:boolean;
leave:boolean;
request:boolean;
inquiry:boolean;
greenPulse:boolean;
redPulse:boolean;
wait:boolean;
startUnion:boolean;
responseOfTrain:boolean;
showGreen:boolean;
showRed:boolean;



ctr1:unionn(checkSucc,checkFail,responseOfTrack);
ctr2:boundeddiff(inquiry,responseOfTrack,0,30);
ctr3:strictpre(inquiry,responseOfTrack,5);
ctr4: coincidence(enter,startOccupied);
ctr5: coincidence(leave,startUnoccupied);
ctr6: Alter(startOccupied,startUnoccupied);

ctr7:unionn(enter,wait,responseOfTrain);
ctr8:Alter(request,responseOfTrain);
ctr9:strictpre(showRed,wait,5);
ctr10:strictpre(showGreen,enter,5);
ctr11:strictpre(enter,leave,5);
ctr12:boundeddiff(request,responseOfTrain,0,50);

ctr13:exclusion(showRed,showGreen);
ctr14:cause(greenPulse,showGreen,5);
ctr15:cause(redPulse,showRed,5);
ctr16:boundeddiff(redPulse,showRed,0,30);
ctr17:boundeddiff(greenPulse,showGreen,0,30);

ctr18:strictpre(checkFail,redPulse,5);
ctr19:strictpre(checkSucc,greenPulse,5);
ctr20:boundeddiff(checkFail,showRed,0,40);
```

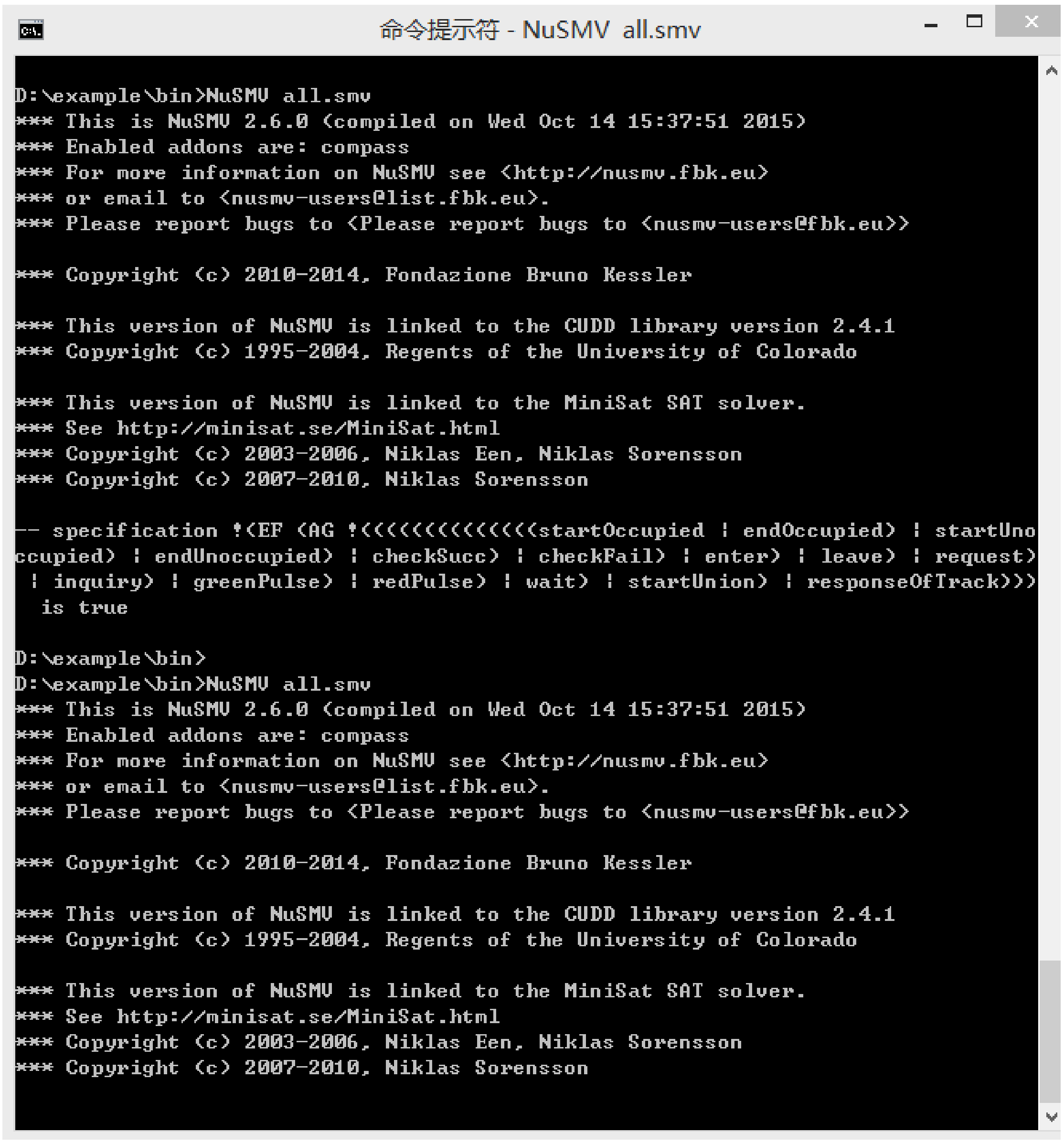
```
ctr21:boundeddiff(checkSucc,showGreen,0,40);
ctr22:strictpre(request,inquiry,5);
ctr23:strictpre(showGreen,enter,5);
ctr24:strictpre(showRed,wait,5);
ctr25:strictpre(inquiry,responseOfTrack,5);

ctr26:exclusion(redPulse,greenPulse);


ASSIGN
init(startOccupied):=FALSE;
init(endOccupied):=FALSE;
init(startUnoccupied):=FALSE;
init(endUnoccupied):=FALSE;
init(checkSucc):=FALSE;
init(checkFail):=FALSE;
init(enter):=FALSE;
init(leave):=FALSE;
init(request):=FALSE;
init(inquiry):=FALSE;
init(greenPulse):=FALSE;
init(redPulse):=FALSE;
init(wait):=FALSE;
init(responseOfTrain):=FALSE;
init(startUnion):=FALSE;
init(responseOfTrack):=FALSE;
init(showRed):=TRUE;
init(showGreen):=FALSE;

CTLSPEC                                    (!                                     EF
AG !(startOccupied|endOccupied|startUnoccupied|endUnoccupied|checkSucc|checkFail|enter|
leave|request|inquiry|greenPulse|redPulse|wait|startUnion|responseOfTrack))

MODULE strictpre(left,right,n)
VAR
    coun:0..n;
INIT
    coun=0;
TRANS
case
    coun=0: (next(left)=TRUE & next(right)=FALSE & next(coun)=coun+1)|(next(left)=FALSE &
next(right)=FALSE & next(coun)=coun);
    coun<n: (next(left)=TRUE & next(right)=FALSE & next(coun)=coun+1)|(next(left)=TRUE &
next(right)=TRUE & next(coun)=coun)|(next(left)=FALSE & next(right)=TRUE & next(coun)=coun -
```

```
1)|(next(left)=FALSE & next(right)=FALSE & next(coun)=coun);
    TRUE: (next(left)=TRUE & next(right)=TRUE & next(coun)=coun)|(next(left)=FALSE &
next(right)=TRUE & next(coun)=coun - 1)|(next(left)=FALSE & next(right)=FALSE &
next(coun)=coun);
esac


MODULE unionn(left,right,new)
TRANS
    (next(left)=TRUE & next(new)=TRUE)|(next(right)=TRUE &
next(new)=TRUE)|(next(left)=FALSE & next(right)=FALSE & next(new)=FALSE)

MODULE boundeddiff(left,right,i,j)
VAR
dif: i..j;
INIT
dif=0
TRANS
case
dif=j:((next(left)=FALSE&next(right)=TRUE&next(dif)=dif - 1) |
(next(left)=TRUE&next(right)=TRUE&next(dif)=dif) |
(next(left)=FALSE&next(right)=FALSE&next(dif)=dif));
dif=i:((next(left)=TRUE&next(right)=FALSE&next(dif)=dif+1) |
(next(left)=TRUE&next(right)=TRUE&next(dif)=dif) |
(next(left)=FALSE&next(right)=FALSE&next(dif)=dif));
TRUE:(next(left)=TRUE&next(right)=FALSE&next(dif)=dif+1) |
(next(left)=FALSE&next(right)=TRUE&next(dif)=dif - 1) |
(next(left)=TRUE&next(right)=TRUE&next(dif)=dif) |
(next(left)=FALSE&next(right)=FALSE&next(dif)=dif);
esac

MODULE cause(left,right,n)
VAR
scnt:0..n;
INIT
scnt=0
TRANS
case
scnt=0:((next(left)=TRUE&next(right)=FALSE&next(scnt)=scnt+1) |
(next(left)=FALSE&next(right)=FALSE&next(scnt)=scnt) |
(next(left)=TRUE&next(right)=TRUE&next(scnt)=scnt));
scnt>0&scnt<n:(next(left)=TRUE&next(right)=FALSE&next(scnt)=scnt+1) |
(next(left)=TRUE&next(right)=TRUE&next(scnt)=scnt) |
(next(left)=FALSE&next(right)=TRUE&next(scnt)=scnt - 1) | (next(left)=FALSE&next(right)=FALSE&
next(scnt)=scnt);
scnt=n:(next(left)=TRUE&next(right)=TRUE&next(scnt)=scnt)                              |
(next(left)=FALSE&next(right)=TRUE&next(scnt)=scnt              -              1)      |
(next(left)=FALSE&next(right)=FALSE&next(scnt)=scnt);

esac

MODULE coincidence(left,right)
TRANS
next(left)->next(right)&next(right)->next(left)

MODULE subclock(left,right)
TRANS
        next(left)->next(right)

MODULE delay(left,right,k)
VAR
     cnt:0..k;
INIT
     cnt=0;
TRANS
case
     cnt<k:  next(left)=FALSE  &  ((next(right)=TRUE  &  next(cnt)=cnt+1)|(next(right)=FALSE  &
next(cnt)=cnt));
     cnt=k|cnt>k:  (next(left)=TRUE  &  next(right)=TRUE  &  next(cnt)=cnt)|(next(left)=FALSE  &
next(right)=FALSE & next(cnt)=cnt);

esac

MODULE exclusion (left, right)
TRANS
(next   (left)=TRUE   &   next   (right)=FALSE)   |   (next(left)=FALSE   &   next   (right)=TRUE)|
(next(left)=FALSE & next (right)=FALSE);

MODULE Alter(left,right)
VAR
     state:boolean;
INIT
     state=FALSE;
TRANS
case
     state   =   FALSE   :     (next(right)=FALSE   &   next(left)=TRUE     &   next(state)   =
TRUE )|(next(right)=FALSE & next(left)=FALSE & next(state)=FALSE);
     state   =   TRUE   :     (next(left)=FALSE   &   next(right)=TRUE     &   next(state)   =
```

FALSE)|(next(left)=FALSE & next(right)=FALSE & next(state)=TRUE);

esac

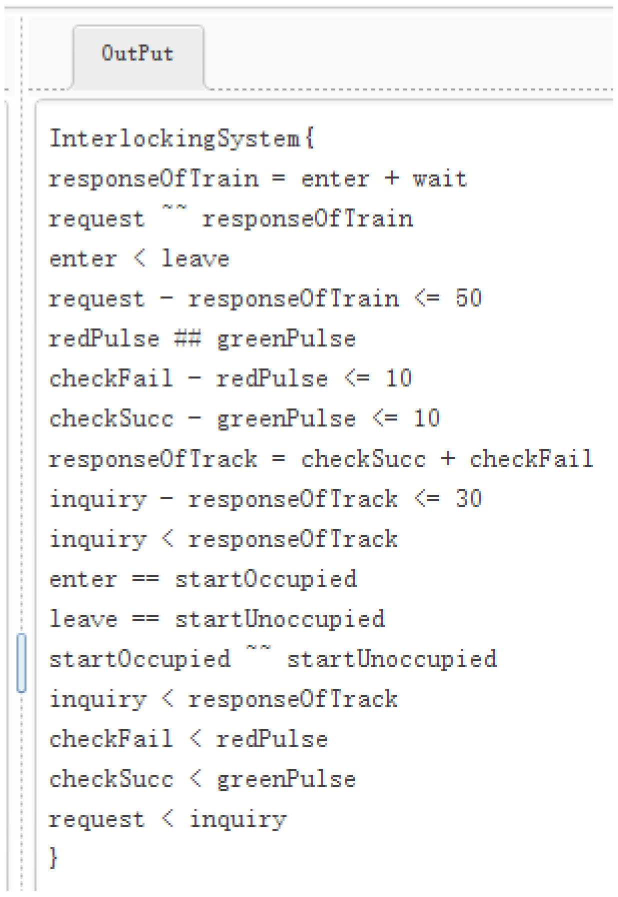
2) verification result



命令提示符 - NuSMV all.smv

```
D:\example\bin>NuSMV all.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification !(EF (AG !(((((((((((((startOccupied | endOccupied) | startUno
ccupied) | endUnoccupied) | checkSucc) | checkFail) | enter) | leave) | request)
 | inquiry) | greenPulse) | redPulse) | wait) | startUnion) | responseOfTrack)))
  is true

D:\example\bin>
D:\example\bin>NuSMV all.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson
```

# step 5: Specification derivation



```
InterlockingSystem{
responseOfTrain = enter + wait
request ~~ responseOfTrain
enter < leave
request - responseOfTrain <= 50
redPulse ## greenPulse
checkFail - redPulse <= 10
checkSucc - greenPulse <= 10
responseOfTrack = checkSucc + checkFail
inquiry - responseOfTrack <= 30
inquiry < responseOfTrack
enter == startOccupied
leave == startUnoccupied
startOccupied ~~ startUnoccupied
inquiry < responseOfTrack
checkFail < redPulse
checkSucc < greenPulse
request < inquiry
}
```