Dr. Wolfgang Emmerich
Dept. of Computer Science
University College London
Pearson Building Room 402

8-Okt-99

Z01 Communication Software Engineering

Tutorial Session 3 - Work Sheet

This tutorial session will give you practical experience with progress property specification. Consider the followin specification for the single lane bridge:

```
const N=3
range T=0..N
range ID=1..N
CAR = (enter->exit->CAR).

NOPASS1=C[1],
C[i:ID]=([i].enter->C[i%N+1]).
NOPASS2=C[1],
C[i:ID]=([i].exit->C[i%N+1]).

||CONVOY=([ID]:CAR||NOPASS1||NOPASS2).
||CARS=(red:CONVOY||blue:CONVOY).

BRIDGE=BRIDGE[0][0],
BRIDGE[nr:T][nb:T]=(when (nb==0) red[ID].enter->BRIDGE[nr+1][nb]
                   |when (nr==0) blue[ID].enter->BRIDGE[nr][nb+1]
                   |when (nr>0) red[ID].exit -> BRIDGE[nr-1][nb]
                   |when (nb>0) blue[ID].exit-> BRIDGE[nr][nb-1]
                   ).

||SINGLELANEBRIDGE = (CARS || BRIDGE).
```

Exercise 1:
Specify progress properties in the single lane bridge example we used in Tutorial Session 2 so that you can verify livenes of cars that are waiting to be admitted onto the bridge. Which results will you obtain from progress analysis of the labelle transition system that is equivalent to your LTS?

Exercise 2:
Use priorities to model a congested bridge. In particular consider a situation where the rate of arrival of new blue or re cars is higher than the rate of departure from the bridge.

Exercise 3:
How do you modify the single lane bridge specification above so that there is progress on both sides of the bridge?

Dr. Wolfgang Emmerich
Dept. of Computer Science
University College London
Pearson Building Room 402

8-Okt-99

Z01 Communications Software Engineering

Tutorial3 - Answer Sheet

Exercise 1:

```
progress BLUE_CARS = {blue[ID].enter}
progress RED_CARS = {red[ID].enter}
```

Exercise 2:

```
HEAVY_TRAFFIC = (SINGLELANEBRIDGE)>>{red[ID].exit, blue[ID].exit}
```

Exercise 3:

```
const N=3
const True=1
const False=0
range B=False..True
range T=0..N
range ID=1..N
CAR = (request->enter->exit->CAR).

NOPASS1=C[1],
C[i:ID]=([i].enter->C[i%N+1]).
NOPASS2=C[1],
C[i:ID]=([i].exit->C[i%N+1]).

||CONVOY=([ID]:CAR||NOPASS1||NOPASS2).
||CARS=(red:CONVOY||blue:CONVOY).

BRIDGE=BRIDGE[0][0][0][0][True],
BRIDGE[nr:T][nb:T][wr:T][wb:T][bt:B]=
 (when(wr<N) red[ID].request->BRIDGE[nr][nb][wr+1][wb][bt]
 |when(wb<N) blue[ID].request->BRIDGE[nr][nb][wr][wb+1][bt]
 |when(nr<N&&wr>0&&nb==0&&(wb==0||!bt))red[ID].enter->BRIDGE[nr+1][nb][wr-1][wb][bt]
 |when(nb<N&&wb>0&&nr==0&&(wr==0||bt)) blue[ID].enter->BRIDGE[nr][nb+1][wr][wb-1][bt]
 |when(nr>0) red[ID].exit -> BRIDGE[nr-1][nb][wr][wb][True]
 |when(nb>0) blue[ID].exit-> BRIDGE[nr][nb-1][wr][wb][False]
 ).

||SINGLELANEBRIDGE = (CARS || BRIDGE)>>{red[ID].exit,blue[ID].exit}.

progress BLUE_CARS = {blue[ID].enter}
progress RED_CARS = {red[ID].enter}
```