lumen lamp

// Place global declarations here.

// knop

chan klik;

// alle lampen tegelijk

//broadcast chan klik;

// max aantal lampen

const int MAX=4;

//declarate voor een lamp

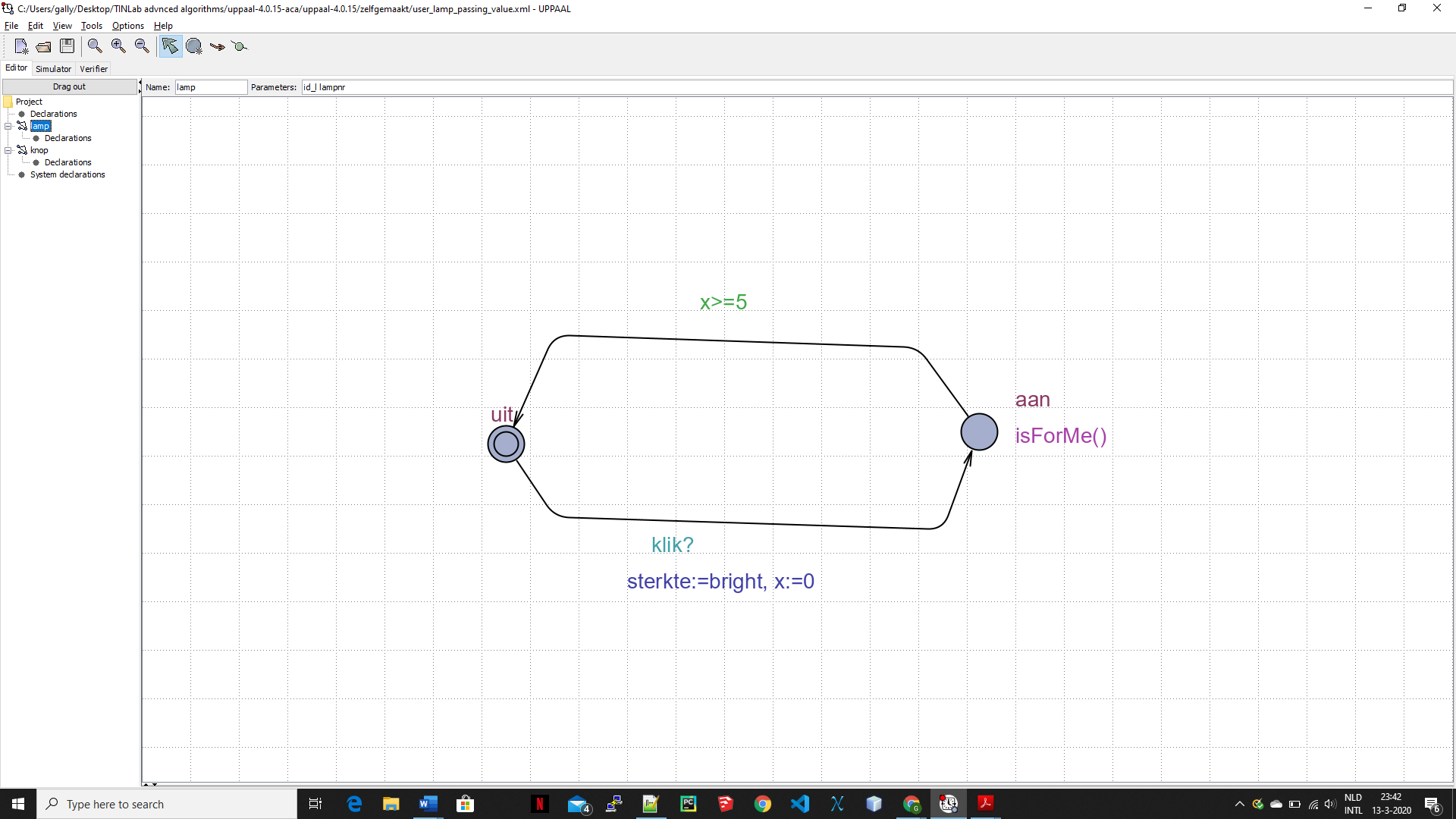
typedef int[0,MAX-1] id\_l;

// lichtintensiteit in lu

typedef int[0,10] lumen;

lumen bright;

template lamp:



parameters: id\_l lampnr

// Place local declarations here.

void functie()

{

}

lumen sterkte;

// wacht 5 tijdseenheden

clock x;

// een constraint op een bepaalde variabele

bool isForMe()

{

// return false;

if(lampnr < 2 && bright<6) return true;

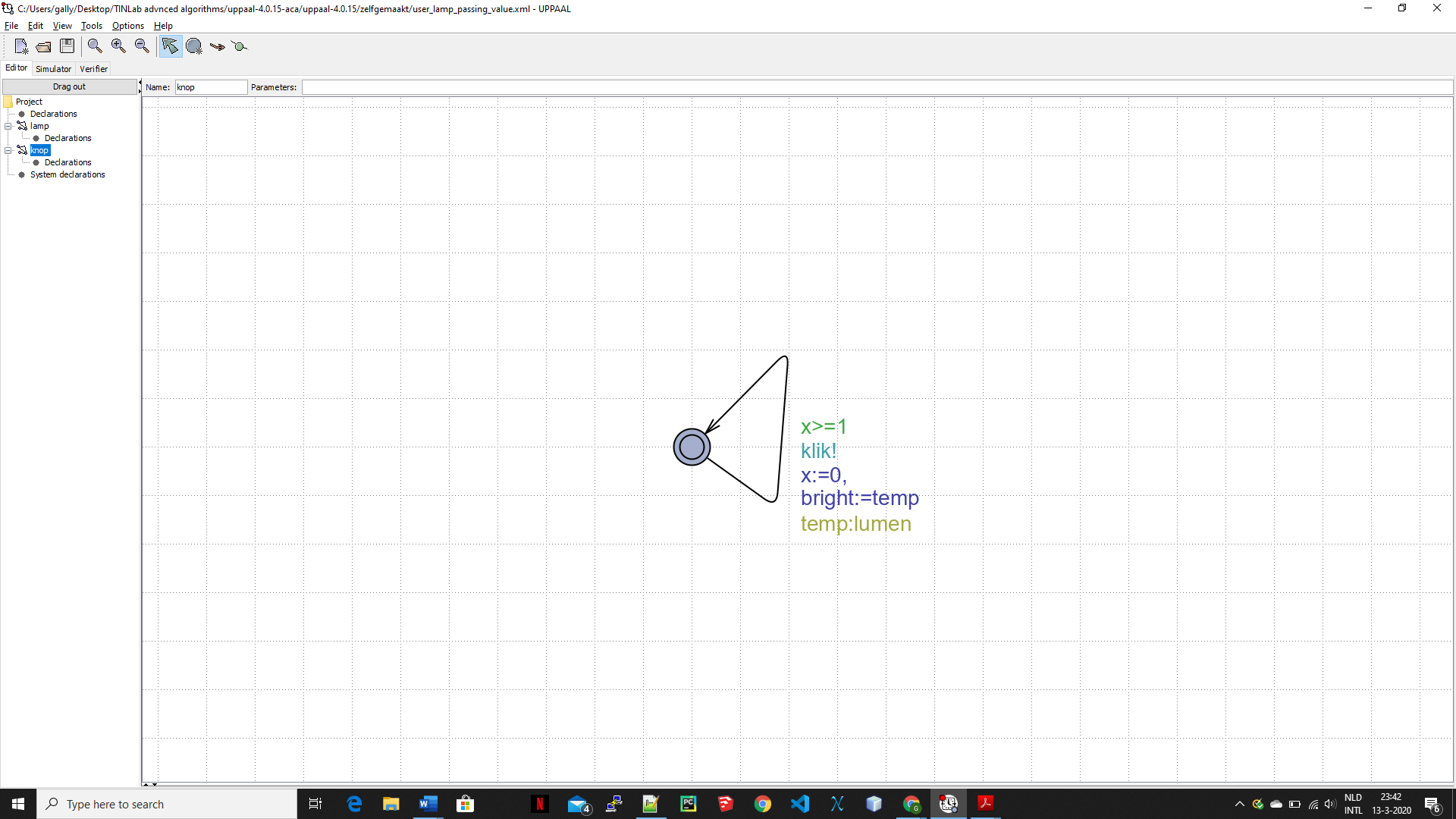
else if(lampnr>1 && bright>=6) return true;

else return false;

}

//verschillende tijdseenheden voor even en oneven lampnummers

template knop:



clock x;

system declarations:

// Place template instantiations here.

// List one or more processes to be composed into a system.

system knop, lamp;

Volgende opdracht:

A.s. vrijdag op het menu:

-Uit de losse pols oefeningen met CTL in Uppaal.

-Waar nodig wat herhaling/extra uitleg van de stof.

-Toelichting eindopdracht

-Verslagen check.

Bij deze enkele studeeraanwijzingen n.a.v. de les van gisteren:

Lees uit het boek model checking:

Hfst. 3

-------

3.1: Hierin wordt de ctl\* semantiek geintroduceerd. Ctl\* is een combinatie van ltl en ctl. Wij bestuderen in deze tinlab alleen de ctl, dus bekijk in dit hoofdstuk vooral figuur 3.1, waarin getoond wordt hoe van een Kripke structuur een computation tree kan worden gemaakt.

3.2: Hierin worden de verschillen tussen ltl en ctl uitgelegd. Belangrijk is hier vooral dat je de ctl fomules AG, EG, AF en EF goed begrijpt.

Hfst. 4

-------

Dit hoofdstuk beschrijft hoe het model checken algoritmisch in elkaar zit, d.w.z. Als we een ctl uitspraak doen over een gegeven Kripke structuur, hoe dan te controleren of de uitspraak geldig is voor de betreffende structuur?

Staar je niet blind op de algoritmen die worden beschreven. Deze zijn moeilijk te lezen.

Verderop in het hoofdstuk (na theorem 1) worden enkele voorbeelden gegeven van het model checking proces, losgelaten op een magnetron. Probeer die voorbeelden te volgen.

Uppaal

------

Uppaal begrijpt een subset van ctl. Derhalve is niet elke ctl formule in Uppaal te checken.

-Lees in de tutorial van Uppaal hoofdstuk 2.2 (the query language). Je zult de ctl herkennen.

-In Uppaal zit een help functie. Lees uit het gedeelte "Language reference" vooral de requirement specifications en semantics. Dat geeeft een goed beeld van wat in Uppaal met ctl kan en niet kan.

Oefeningen:

-Bouw in Uppaal een eenvoudig model met enkele vertakkingen en wat loops.

-Zet enkele booleans in je model en gebruik de updates in de transities om ze waarden te geven.

-Check je model op de formules A[], A<>, E[], en E<>...Experimenteer!

-Checken op liveness is in Uppaal wat moeilijker daar Uppaal met tijd werkt. Je zult met combinaties van klokken en invarianten het model systematisch uit states moeten dwingen om een check op liveness te laten slagen.

Ten slotte een klein breinbrekertje:

Geef (op papier) een Kripke structuur, waarin:

-AG(p) geldig is.

-AGAF(p) niet geldig is.

Tot vrijdag!

Requirements:

5 brightness levels plus Strobe, SOS:

Turbo - 900 lumens - 1 hr. 55 min.

High - 400 lumens - 4 hr.

Mid - 150 lumens - 10 hr. 25 min.

Low - 50 lumens - 37 hr.

Eco - 5 lumens - 260 hr.

Tactical tail switch with momentary-on function

Intelligent memory - automatically memorizes the last brightness level when switched off

Digitally regulated output to maintain constant brightness

Reverse polarity protection to protect from improper battery installation

Over heat protection to avoid high-temperature of the surface

Side switch on the head

Uses a Cree XP-L HI LED with a lifespan of 50,000 hours

https://nl.knivesandtools.eu/nl/pt/-fenix-pd35.htm?gclid=CjwKCAiAzJLzBRAZEiwAmZb0ajEM1oPjbKFpCUT8Dtk9RvISkKoFdOVemxj-Nbz9TTq\_9DnvxqwP\_BoCidMQAvD\_BwE

https://www.fenix-store.com/fenix-pd32-led-flashlight-2016-edt/