Hogeschool Rotterdam

Technische Informatica

Tinlab Advanced Algorithms

Gezamenlijk Eindverslag

Auteurs
Jorian Nakorikantodas 0969032
Matthijs Meijerink 0962038
Ebrar Eryigit 0961449
Aroena Almeida Mendes 0968262

School
Hogeschool Rotterdam Wijnhaven
3011 WN Rotterdam
Nederland

Docenten
Wessel Oele
Elvira van der Ven

Vakcode
TINLAA01

15 april 2021
versie 1.1

1 Opdracht

Onze opdracht is om een sluis te modelleren. We hebben enkele requirements gekregen waarmee we uiteindelijk onze eigen keuzes mogen maken om een sluis te modelleren. Het model moet uitbreidbaar zijn om in meerdere situaties toegepast te kunnen worden. We moeten technische besluiten gaan nemen over het type sluis dat we gaan modelleren, gebruik van sensoren, queues en integers in het model, het aantal boten, tijd en het waterniveau. We zullen deze keuzes ook gaande het verslag beargumenteren.

Lijst van figuren

Lijst van tabellen

Inhoudsopgave

2 Sluizen

2.1 Soorten Sluizen

Voor onze opdracht hadden we de vrijheid om zelf een type sluis uit te kiezen die we vervolgens moesten modelleren. Er waren verassend veel type sluizen waar we uit konden kiezen:

- Zeesluis
- De schutsluis
- De keersluis
- De uitwateringssluis
- De ontlastsluis
- De inlaatsluis
- De spuisluis
- De inundatie sluis
- De doksluis.
- Waaiersluis

De sluis die we uiteindelijk hebben gekozen is de schutsluis. We zullen hieronder kort uitleggen waarom deze sluis onze keuze is geweest.

- Eenvoudig: Dit type sluis heeft de minste bewegende onderdelen en sensoren. Het modelleren en verifiëren vergt hierdoor minder werk.
- Veiligheid: Doordat dit type sluis niet veel onderdelen heeft, is het makkelijker om rekening te houden met de veiligheid van de sluis. Er zijn namelijk minder onderdelen die onverwachts ander gedrag kunnen vertonen dan gewenst is.

- Veel voorkomend: Dit is de meest gebruikte sluis in Nederland. Hierdoor heb je veel referentiemateriaal. Zelfs de sluis in het Kralingse Bos is een schutsluis.

Een schutsluis heeft als doel scheepvaart mogelijk te maken tussen waterwegen met een ongelijk waterpeil. Daartoe bestaat een schutsluis uit minimaal twee sluishoofden, die onderling zijn verbonden met een schutkolk. Door het water in deze kolk omstebeurt het waterpeil te geven van het boven- of beneden water, kunnen de sluisdeuren omstebeurt worden geopend om de schepen in- en uit te laten varen[1].

2.2 Requirements en Specificaties

In deze paragraaf zullen we het hebben over onze requirements die bij het model horen. We kregen van A. M. B. ten Aar de volgende 'requirements':

- veiligheid
- efficiëntie
- capaciteit
- onderhoudskosten
- duurzaamheid

Met deze requirements konden we niet veel, want wat is „veiligheid, efficiëntie, capaciteit, onderhoudskosten en duurzaamheid?" In tabel 3.1 hebben we onze requirements en uitleg/specificaties opgenomen.

Requirements	Specificatie	Tijdseenheden
1) Sluisdeuren kunnen openen en sluiten.	De sluisdeur(en) moeten in staat zijn om open en dicht te gaan in een realistische tijd. Dit gebeurt wanneer er geen obstakels bij de deur zijn.	5 tijdseenheden.
2) Sensoren kunnen boten detecteren.	Er zijn sensoren aanwezig voor de sluisdeuren (IR). Deze sensoren kunnen detecteren wanneer een boot arriveert.	n.v.t.
3) Stoplichten kunnen aan en uit. (rood en groen)	Wanneer er een boot door de sluis heen gaat dan gaan de stoplichten op de juiste kleur.	n.v.t.
4) Water moet met de schuiven naar buiten/binnen kunnen stromen.	Om het waterniveau te kunnen reguleren, zijn er schuiven in de sluisdeuren aanwezig.	2 tijdseenheden per meter waterniveau.
5) De sluisdeuren kunnen niet tegelijkertijd open.	Door de hoogteverschillen in water kunnen de beide sluisdeuren niet tegelijk open.	n.v.t.
6) Waterniveau kan gemeten worden binnen en buiten de sluis.	Door te kijken naar het waterniveau buiten de sluis, kan bepaald worden welke schuiven open moeten om het waterniveau aan te passen binnen de sluisdeuren.	n.v.t.
7) De sluis heeft een binnenruimte waar boten in kunnen komen.	Er is een binnenruimte aanwezig, waar de boten van buitenaf naar binnen kunnen varen. In deze binnenruimte wordt het waterniveau gezakt of verhoogd.	n.v.t.

Tabel 2.1: Requirements en specificaties tabel

3 Verificatie

3.1 Soorten Verifiers

Door middel van verificatie queries zorgen we ervoor dat we aan de requirements en specificaties voldoen in het model. In Uppaal heb je ook de mogelijkheid om de eigenschappen van je model te verifiëren door middel van verschillende temporale operatoren. De belangrijkste operatoren die wij ook in onze verificatie gebruiken zijn:

- A[ ] p : Voor alle states in elk pad is p waar.
- A<> p : Voor alle paden zal p vroeg of laat waar zijn.
- E[ ] p : Er is een pad waar p in alle states waar is.
- E<> p : Er is een pad waar p vroeg of laat waar zal zijn.

3.2 Onze Verifiers

- A[ ] not deadlock

  Hiermee controleren we of in geen enkele state van het model, er een deadlock bestaat. Oftewel, een situatie waarin er in het model geen verdere transities meer mogelijk zijn.

- A[ ] !(Gate(0).open and Gate(1).open)

  Het is in geen enkele state mogelijk dat beide deuren tegelijkertijd open zijn. Het is de bedoeling dat dit altijd 'rood' moet zijn in Uppaal. Zo weten we dat er nooit een state bestaat waar beide gates open kunnen.

- A[ ] !((Gate(0).open or Gate(1).open) and (Pump(0).pomp_aan or Pump(1).pomp_aan))

  In alle mogelijke paden zal wanneer deur_check state is behaald, terug gegaan worden naar de idle state.

- A<> !(Pump(0).pomp_aan and Pump(1).pomp_aan)

  In alle paden is er geen mogelijkheid om beide pompen tegelijkertijd aan te hebben

- A<>(Controller.deur_check imply Controller.idle)

  Normaal gesproken laat je liveness zien door middel van A[ ], alleen omdat we in ons model ook een state hebben waar queue_legen naar queue_legen loopt, gebruiken we A<>. Hier leggen we uit dat we van de liveness van het model, er is altijd voortgang in het model.

- A[ ] (queues[0] + queues[1]) <= Request.max_boten.

  Er is nooit een state waar de queues bij elkaar groter zijn dan het maximale aantal boten dat we in de declarations hebben opgesteld. Hiermee zeggen we dat we nooit een verdubbeling van een boot krijgen.

- E<> (waterniveau==5)

  Hiermee willen we aantonen dat er een state bestaat waar het waterniveau: 5 is.

- E<> (waterniveau==0)

  Hiermee willen we aantonen dat er een state bestaat waar het waterniveau: 0 is.

4 Uppaal modellen

4.1 Onze keuzes

Aantal sluisdeuren
We hebben besloten om twee sluisdeuren te modelleren in ons model omdat dit al voldoende is om een werkende sluis te modelleren en het aan onze eisen voldoet.

Waterniveau reguleren
Het waterniveau reguleren we in de main controller. We hebben zelf twee waterpompen gemodelleerd die water in of uit de sluiskolk pompen.

Detectie van boten
De detectie van boten houden we bij in het request handler model, we kijken dan van welke kant de boot komt van de sluis. Dus de directie waar de boot vandaan komt.

Stoplichten
In ons model hebben we geen stoplichten gemodelleerd. Dit komt door de complexiteit van stoplichten in een sluis. Je zou dan in totaal vier stoplichten moeten bijhouden. Twee aan de binnenkant van de sluis en twee aan de buitenkant van de sluis.

Queues
We hebben besloten om 2 queues te maken: queue[0] en queue[1]. De ene queue wordt gebruikt als binnenkant van de sluis, waar de boten in aankomen. De andere queue wordt gebruikt als buitenkant van de sluis, daar waar de boten aan komen varen. Welke queue wordt gebruikt voor welke van de 2 functies hangt af van de directie van de boten. Wanneer de boten bijvoorbeeld omhoog moeten zal queue[0] gebruikt worden als binnenplaats en queue[1] als aankomstplaats. Wanneer de boten de andere kant op moeten, dus van hoog naar laag, zal dit precies andersom zijn.

Pomp
Onze pomp bestaat uit 2 states. Deze worden aangeroepen vanuit de Controller. De pomp is geparametriseerd, want we hebben 2 pompen. 1 voor elke deur van de sluis. Vanuit de Controller roepen we de juiste pomp aan middels de direction boolean. Deze boolean is 1 of 0. Dit ligt aan de richting van de boten.

4.2 Verloop van het model

1. De request handler wordt als eerste aangeroepen door de Controller. Hier kunnen we een direction kiezen die we willen. Deze direction bepaalt van waar de boten komen. Vervolgens wordt er een random waarde tussen 1-5. Dit wordt het aantal boten. Deze zetten we in de queue[direction]. Zo weet de controller hoeveel boten er zijn en waar deze boten staan. Zie 4.1.

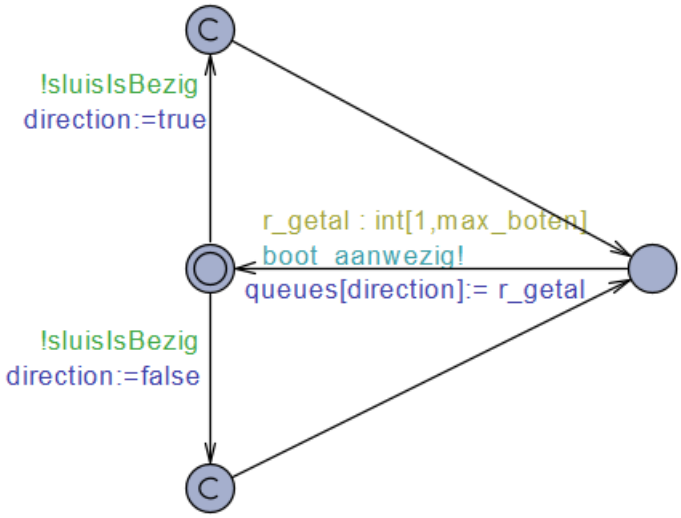
2. Wanneer de request handler de boten in de queue heeft gezet, gaat de Controller de gates checken. Als één van de Gates nog aan het sluiten is dan wacht de Controller hierop 4.2. Vervolgens wordt de juiste pomp aangeroepen op basis van de huidige direction 4.3. De pomp zorgt ervoor dat het waterniveau op de juiste hoogte is voordat de Gate waar de boten zijn geopend kan worden. Wanneer dit het geval is dan gaat de pomp weer uit en wordt de Gate geopend.

3. De gate wordt geopend. De state gaat van closed naar opening 4.4. Het duurt voor de Gate 5 tijdseenheden om te openen. De tijdseenheden hebben we lokaal gedefinieerd. Na deze 5 tijdseenheden gaat de gate naar Open. Wanneer de gate open is, kunnen de boten naar binnen varen. Dan wordt de queue[direction] overgezet naar queue[!direction]. Hierdoor wordt dus de sluis gevuld met de boten.

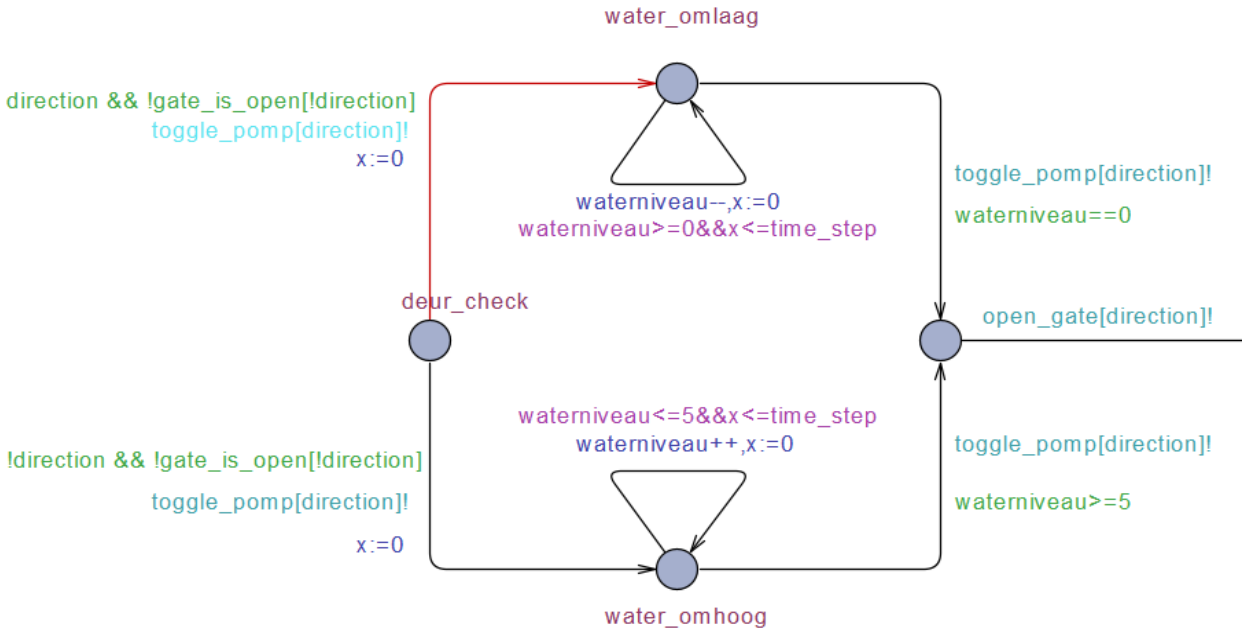
4. Wanneer de Gate dan open is, dan wordt de queue één voor één omgezet. We hebben een tijdseenheid gedefinieerd van 2 tijdseenheden van hoe lang het duurt voordat er een boot wordt omgezet naar de andere queue. Dit gebeurt net zo lang totdat de queue die voor de Gate stond te wachten helemaal leeg is. Zie 4.5.

5. Als de queue voor de eerste keer helemaal geleegd is, dan wordt de direction omgedraaid, de sluisToggle variabele op false gezet en de geopende Gate gesloten 4.6. De reden hiervoor is dat we nu de andere kant op gaan met de sluis. We gebruiken nu de andere pomp waardoor het water de andere kant op gaat.

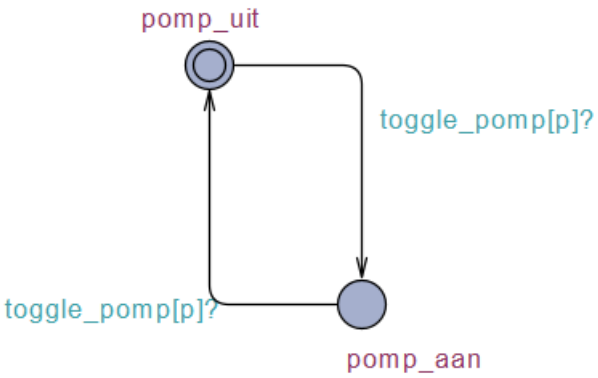
6. Nu wordt de queue opnieuw geleegd, dit maal van de binnenkant van de sluis naar de buitenkant van de Gate. De boten zijn nu dus naar de andere kant overgebracht. De sluis is klaar en kan nu weer naar de idle stand. Voor de volledige controller, zie 4.7.



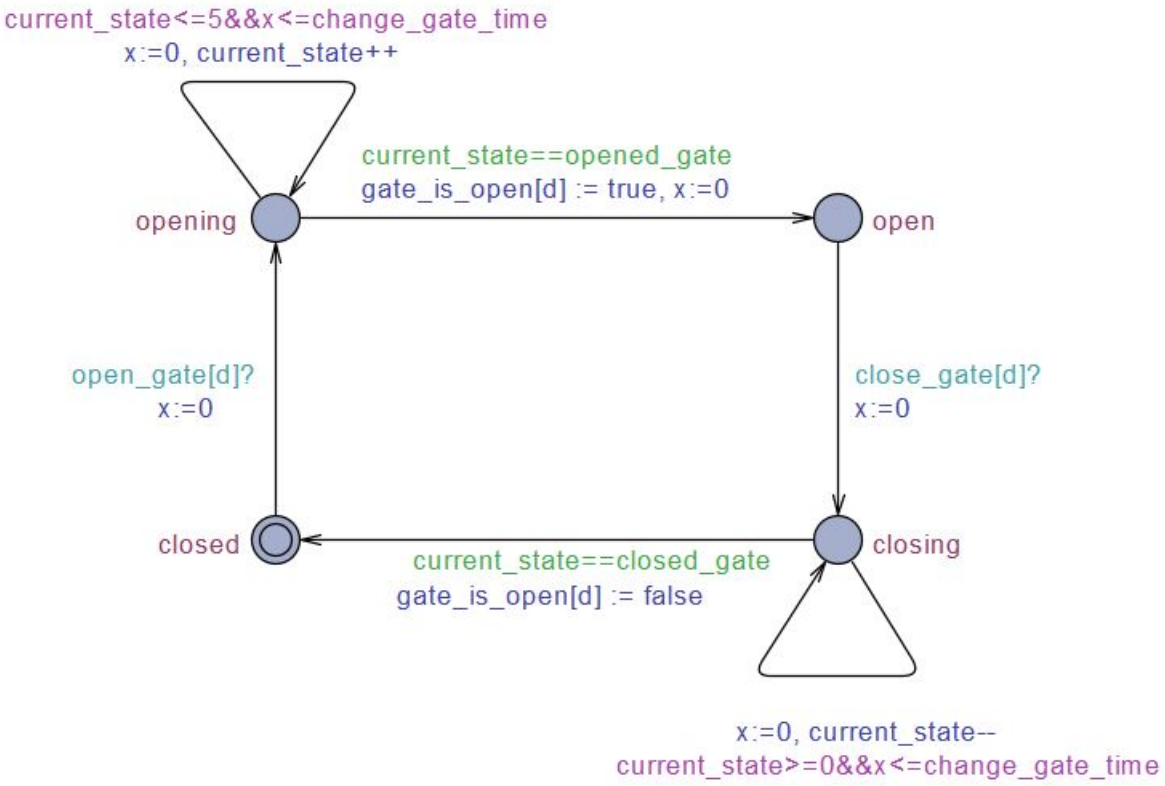
Figuur 4.1: Request handler



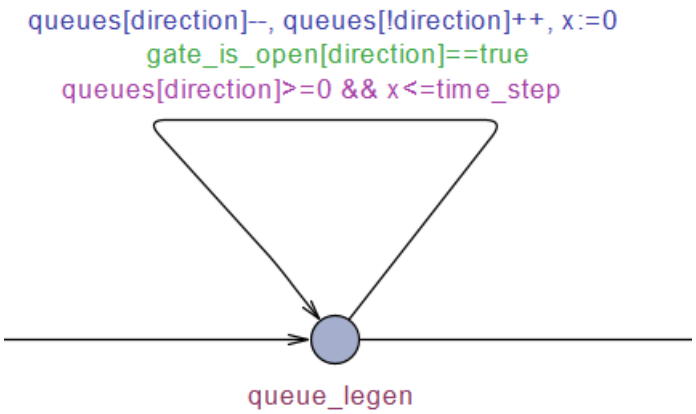
Figuur 4.2: Het waterniveau wordt gereguleerd, Pompen worden aangeroepen



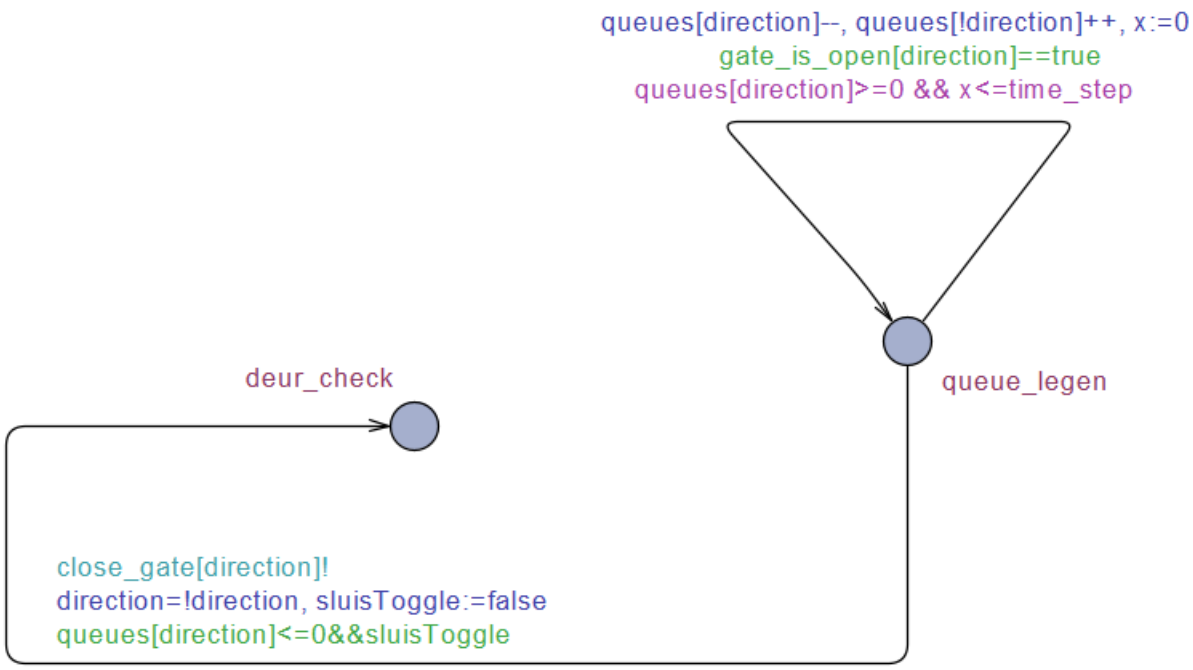
Figuur 4.3: Pomp in het systeem



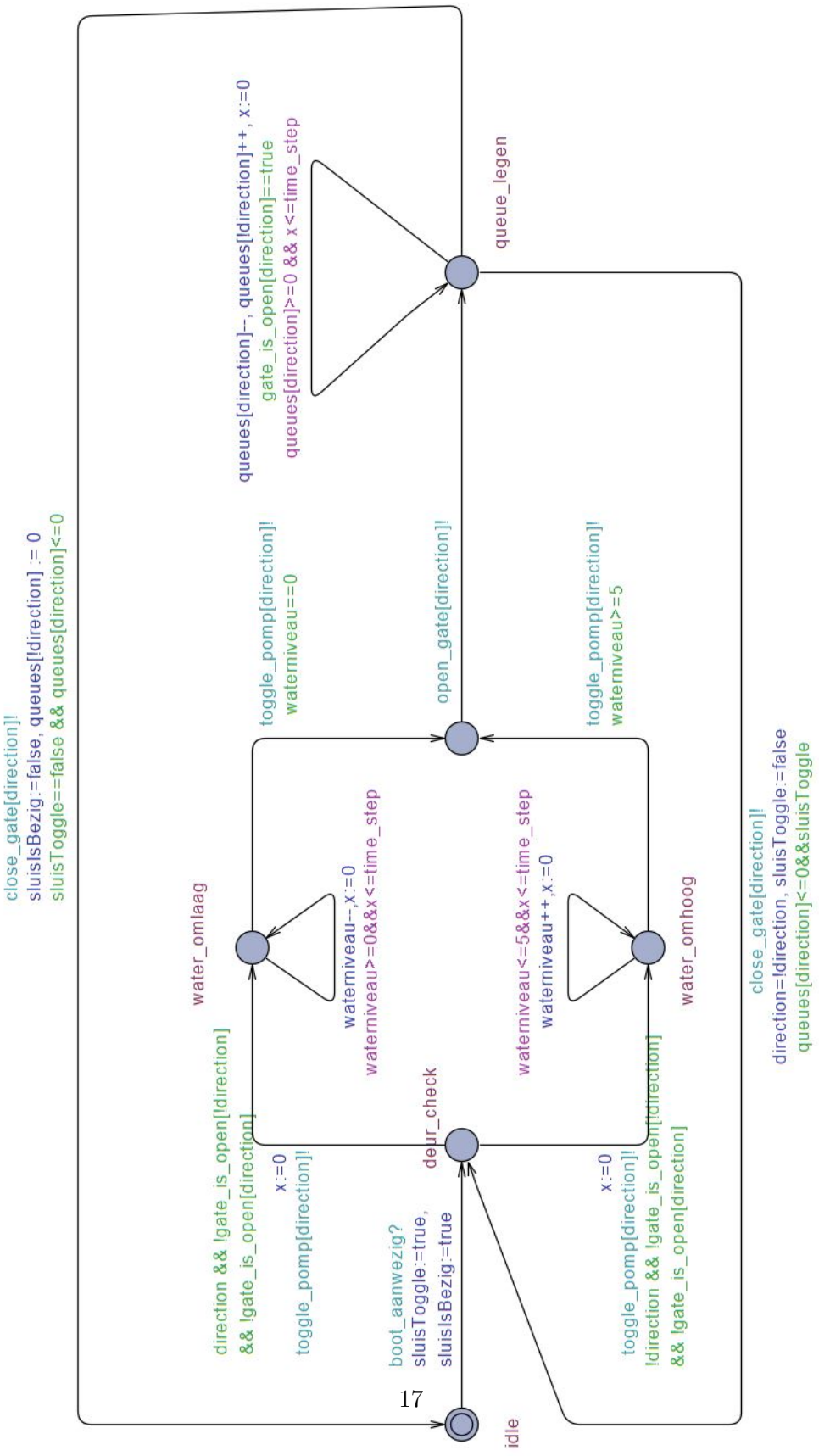
Figuur 4.4: Deur in het systeem



Figuur 4.5: Queue legen in het systeem



Figuur 4.6: De direction wordt omgedraaid, zodat de sluis de andere kant gaat



Figuur 4.7: Main controller van de sluis

Bibliografie

[1] Gerrit Jan Arends. Sluizen en stuwen: de ontwikkeling van de sluis-en stuwbouw in nederland tot 1940. *Bouwtechniek in Nederland 5*, 1994.