Verslag Tinlab Advanced Algorithms

A. de Ridder
0937558

15 april 2019

Inhoudsopgave

1 Inleiding **2**
- 1.1 Air France Flight 447 . . . 2
- 1.2 Automatiseringsparadox . . . 2
- 1.3 Therac . . . 2
- 1.4 Turkish Airlines-vlucht 1951 . . . 3

2 Requirements **4**
- 2.1 Requirements . . . 4
- 2.2 Specificaties . . . 4

3 Modellen **5**
- 3.1 De Kripke structuur . . . 5
- 3.2 Soorten modellen . . . 5
- 3.3 Guards en invarianten . . . 5
- 3.4 Tijd . . . 5
- 3.5 Deadlock . . . 5
- 3.6 Zeno gedrag . . . 5

4 Logica **6**
- 4.1 Propositielogica . . . 6
- 4.2 Predicatenlogica . . . 6
- 4.3 Kwantoren . . . 6
- 4.4 Dualiteiten . . . 6

5 Computation tree logic **7**
- 5.1 De computation tree . . . 7
- 5.2 Operator: AG . . . 7
- 5.3 Operator: EG . . . 7
- 5.4 Operator: AF . . . 7
- 5.5 Operator: EF . . . 7
- 5.6 Operator: AX . . . 7
- 5.7 Operator: EX . . . 8
- 5.8 Operator: p U q . . . 8
- 5.9 Operator: p R q . . . 8
- 5.10 Fairness . . . 8
- 5.11 Liveness . . . 8

1 Inleiding

Voor het Tinlab: Advanced Algorithms moet er informatie verkregen worden over bepaalde onderwerpen voordat er gewerkt kan worden aan het eindproduct: model van een sluis. In dit verslag worden alle onderwerpen besproken en uitgelegd, zodat hier op teruggekeken kan worden als iets onbekend is tijdens het maken van de sluis.

1.1 Air France Flight 447

Op 1 juni 2019 is een Airbus A33 neergestort in de atlantische oceaan, 228 mensen zijn omgekomen. Na langdurig onderzoek was gebleken dat de piloot de oorzaak was van het ongeluk. Hij misimpreteerde de luchtsnelheid indicatoren en reageerde door precies het tegenovergestelde te doen wat er verwacht wordt.

Dit is een voorbeeld waar 'mode confusion' de oorzaak is van een ongeluk. Modusfouten gebeuren als het actuele automatiseringssysteem verschilt van de verwachting van de gebruiker. Dit kan leiden tot acties die resulteren in onverwachte of ongewilde consequenties. De gebruiker weet niet meer wat die moet doen wanneer zijn actie leidt tot iets onverwachts. 'Mode confusion' heeft bijgedragen tot significanten luchvaatongevallen en -incidenten. [1]

1.2 Automatiseringsparadox

The Paradox of Automation zegt dat hoe efficinter het geautomatiseerde systeem is, hoe crucialer de menselijke bijdrage van de operators is. Mensen zijn minder betrokken, maar hun betrokkenheid wordt belangrijker.

Als een geautomatiseerd systeem een fout bevat, wordt die fout vermenigvuldigd totdat deze is hersteld of uitgeschakeld. Dit is waar menselijke operatoren binnenkomen. [?]

1.3 Therac

De Therac-25 was een radiatie therapie machine gemaakt door AECL in de jaren 90. Dit apparaat maakt revoluitionaire 'dual treatment mdoe'. Het was ontworpen vanaf het begin om software gebaseerde veiligheidssystemen te gebruiken in plaats van hardware gebaseerde veiligheidsysteem.

Het verwijderen van deze hardwarematige veiligheidsmaatregelen had tragische gevolgen. Er was een 'race condition' in de software geprogrammeerd wat geleidt heeft tot de dood van drie patinten en bij minstens drie andere patinten slopende verwondingen veroorzaakten. Een 'race conditition' vindt plaats wanneer meerdere treads dezelfde resource tergelijkertijd aanvragen.

Dit is een extreem voorbeeld wat er mis kan gaan met softwaresystemen en de gevolgen die bugs kunnen hebben voor gewone mensen. [3]

1.4 Turkish Airlines-vlucht 1951

Een bekende hardware defect van de Boeing 737-800 is een onderdeel geweest bij het neerstorten van dit vliegtuig. Hierbij kwamen 9 van de 135 inzittenden om het leven. Het vliegtuig kampte met een defecte linker radiohoogtemeter, hierdoor ging het automatische gashendelsysteem zich gedragen alsof het toestel zich in de laatste fase voor de landing bevond.

De signalen dat het vliegtuig te langzaam vloog werd niet door de bemanning niet opgemerkt. Andere piloten hebben dezelfde symptomen met het vliegtuig gehad maar losten dit zelf op. Daarnaast hebben ze deze fout niet gerapporteerd.

De fout was bekend bij zowel de vliegtuigmaker als maatschappij voor de crash. Echter hebben ze het risico van het defect verkeerd ingeschat.

2 Requirements

Bij het opzetten van een project wordt er lijst van eisen/requirements opgesteld. Met deze lijst van eisen kan het eindproduct gecontroleerd worden. Dit wordt vaak gedaan door specificaties op te stellen voor het eindproduct. In de volgende kopjes wordt er verder ingegaan op deze twee termen.

2.1 Requirements

Toen Tesla begon aan zijn elektrische auto's observeerden ze dat de elektrische automerkt erg veel kritiek kreeg. Vanuit deze observatie formuleerde ze het volgende doel: 'to accelerate the advent of sustainable transport by bringing compelling mass market electric cars to market as soon as possible.' . Dit doel geeft eigenlijk de eisen waar het bedrijf moet voldoen, echter ook specifiek de auto.

Een requirement is een uitspraak over fenomenen of doelen die bereikt moeten worden. Deze zijn met enige regelmaat informeel, niet precies geformuleerd. De bovenstaande eisen voldoen hier ook aan.

2.2 Specificaties

Een specificatie is een uitspraak over gedeelde fenomenen of doelen die de machine moet bereiken middels de onderdelen waar die machine uit bestaat of middels de fenomenen waar de machine controle over heeft. Deze zijn doorgaans preciezer, meetbaar, exact geformuleerd

Specificaties geformuleerd met Tesla's doel:

- Auto moet massa-geproduceerd kunnen worden
- Auto moet elektrisch zijn
- Auto moet duurzaam zijn
- etc..

3 Modellen

Een systeem wordt gebouwd door middel van een 'state transition diagram'. Uiteindelijk wordt door middel van temporele logica geverifiërd of uit een door requirements en specificatie voortkomende uitspraak geldig is voor het model.

Er bestaan veel varianten van state transition diagrams. 'E'en daarvan zal nu besproken worden.

3.1 De Kripke structuur

Een Kripke structuur bestaat uit states. Een state is een beschrijving van het systeem gedurende een bepaald tijdsinterval. Het bevat de waarden van alle variabelen van het systeem gedurende dat interval. En is niet een moment in de tijd. Een werkend systeem kan men modelleren middels het doorlopen van een aantal toestanden. Het overgaan van de ene toestant daar de andere, wordt ook wel een transitie genoemd.

3.2 Soorten modellen

3.3 Guards en invarianten

Invarianten en guards zijn condities. Een invariant is een conditie die geldt in een state en die altijd waar is wanneer de automaat zich in die state bevindt. Waar een guard is een conditie die geldt in een transitie. M.a.w. De transitie kan alleen genomen worden wanneer de guard geldig.

3.4 Tijd

Tijd is een continu verschrijnsel en wordt bijgehouden met klokken. Alle klokken in een model lopen even snel. Klokwaarden kunnen uitgelezen worden in invarianten en quards, echter kunnen ze alleen worden gereset in transities en worden altijd reset naar 0. Tijd verstrijkt uitsluitend in states.

3.5 Deadlock

Deadlock onstaat wanneer een combinatie van invarianten en queard het verstrijken van tijd verhindert.

3.6 Zeno gedrag

Zeno gedrag is een probleem dat de mogelijkheid geeft om in een eindige hoeveel tijd een ondereindig aantal handelingen kan worden verricht.

4 Logica

4.1 Propositielogica

Propositielogica is logica door middel van proposities. Proposities zijn uitspraken of beweringen die ofwel waar, ofwel onwaar zijn.

Connective	Symbol	meaning
Negation	$\neg$	not
Conjunction	$\wedge$	and
Disjunction	$\vee$	or
Implicit	$\Rightarrow$	if...then
double implicit	$\Leftrightarrow$	then and only then

4.2 Predicatenlogica

Een predicaat is een uitspraak met vrije variabelen, die een propositie wordt zodra alle vrije variabelen in dat predikaat gebonden zijn aan een waarde.

Voorbeeld predicaat:

Uitspraak	Symbolisch
ik ben Alex	A(ik)
Thomas is 12	T(12)

Vrije variabele kunnen gebonden worden aan waardes in een universum met kwantoren.

4.3 Kwantoren

Uitspraak	Symbolisch	Naam
voor alle x	$\forall$ y	Universele kwantor
er zijn x	$\exists$x	Existiëntele kwantor

4.4 Dualiteiten

Ontkenning van Meervoudige predikaten

$\neg\forall x \quad \Leftrightarrow \quad \exists\neg x$
$\neg\exists x \quad \Leftrightarrow \quad \forall\neg x$

Meervoudige predikaten

$$
\begin{array}{lcl}
\forall x \forall y\ \mathrm{P(x,y)} & \Leftrightarrow & \forall x \forall y\ \mathrm{P(x,y)} \\
\forall x \exists y\ \mathrm{P(x,y)} & \Leftrightarrow & \forall x \exists y\ \mathrm{P(x,y)} \\
\exists x \forall y\ \mathrm{P(x,y)} & \Leftrightarrow & \exists x \forall y\ \mathrm{P(x,y)} \\
\exists x \exists y\ \mathrm{P(x,y)} & \Leftrightarrow & \exists x \exists y\ \mathrm{P(x,y)} \\
\neg \forall x \forall y\ \mathrm{P(x,y)} & \Leftrightarrow & \exists x \exists y\ \neg \mathrm{P(x,y)} \\
\neg \forall x \exists y\ \mathrm{P(x,y)} & \Leftrightarrow & \exists x \forall y\ \neg \mathrm{P(x,y)} \\
\neg \exists x \forall y\ \mathrm{P(x,y)} & \Leftrightarrow & \forall x \exists y\ \neg \mathrm{P(x,y)} \\
\neg \exists x \exists y\ \mathrm{P(x,y)} & \Leftrightarrow & \forall x \forall y\ \neg \mathrm{P(x,y)}
\end{array}
$$

5 Computation tree logic

5.1 De computation tree

Een 'Compution tree' is een boom met nodes en edges. Elke 'node' in de boom representeerd een alleenstaande computionele toestand, waar elke 'edge' de overgang naar een volgende computionele toestand representeerd. [2]

'Computionation tree logic' beschrijft eigenschappen van een 'Compution tree': formules kunnen redeneren over veel executies tegelijkertijd. De semantiek van deze logica is gedefinieerd in termen van 'states'.

5.2 Operator: AG

AG p : p moet gelden in alle volgende nodes in alle paden vanaf de huidige node (Elke node moet waar zijn vanaf de huidige staat).

5.3 Operator: EG

EG p: Er bestaat minstens een pad startend van de huidige node waar p geld en blijft gelden voor een pad

5.4 Operator: AF

AF p : p moet vanzelf gelden in één node op elk pad vanaf de huidige node.

5.5 Operator: EF

EF p: Er bestaat minstens een pad startend van de huidige node waar p eventueel gaat gelden

5.6 Operator: AX

AX p : p moet gelden in alleen de volgende state van all paden vanaf de huidige node.

5.7 Operator: EX

EX p: Er bestaat minstens een pad startend van de huidige node waar p in de volgende state geld

5.8 Operator: p U q

p U q: In alle paden en volgende states geld p tot q geldt. Dan valt er niks meer te zeggen over p in dat specifieke pad.

5.9 Operator: p R q

p U q: In alle paden en volgende states geld p tot q geldt. Daarna geldt p niet meer in dat specifieke pad.

5.10 Fairness

AG(AF(q)):

- Op elk denkbaar pad moet in elke dekbare state gelden AF(p)
- In welke state de automaat zich ook bevindt, in alle richtingen kom je vroeg of laat een state tegen, waarin p geldig is.

5.11 Liveness

AG(p $\rightarrow$ AF(q)):

- Altijd en overal geldt: p $\rightarrow$ AF(q)
- Altijd en overal geldt: "Als p geldt dan geldt vroeg of laat q"

Referenties

[1] Jan Bredereke and Axel Lankenau. A rigorous view of mode confusion. 2002.

[2] Edward R Griffor. *Handbook of computability theory*, volume 140. Elsevier, 1999.

[3] Nancy G Leveson and Clark S Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.