Samenvatting

> Het begrip toeval is niet tegenstrijdig, het wordt pas tegenstrijdig als ik probeer het te formaliseren.
> H. Freudenthal

Motivatie

Dit proefschrift gaat over het correct bewijzen van software en hardware systemen. In het bijzonder leggen we ons toe op het correct bewijzen van *reactieve systemen* waarin kansen, tijdseisen en/of parameters een belangrijke rol spelen.

Reactieve systemen zijn hardware– of softwaresystemen die in continue wisselwerking met hun omgeving staan. Zij reageren op signalen uit hun omgeving en sturen op hun beurt ook weer signalen die hun omgeving beïnvloeden. Voorbeelden van zulke systemen zijn besturingssoftware voor kerncentrales en protocollen voor het oversturen van informatie over het Internet.

Formele Methoden en Formele Verificatie

Door hun complexe interactie met de omgeving is het ontwerp en de realisatie van reactieve systemen een lastig en foutgevoelig proces. Diverse voorbeelden, zoals de softwarefout die de crash van de Ariane 5 raket veroorzaakte en de hardwarefout in de Pentium V, onderstrepen dit.

Correctheid van soft– en hardware vormt traditioneel een belangrijk onderwerp van studie binnen het informatica–onderzoek. Dit onderzoek heeft een veelheid aan talen, technieken en tools (computergereedschappen) voortgebracht om hardware en software systemen te beschrijven en te analyseren. Technieken, talen en tools die een wiskundige onderbouwing hebben worden ook wel *formele methoden* genoemd.

Deze wiskundige aanpak heeft als voordeel dat ze heel precies is. Wanneer de systeemvereisten van een product formeel gespecificeerd worden, ligt de interpretatie vast en kunnen er in latere fases van de productontwikkeling minder gemakkelijk misverstanden over ontstaan. Bovendien kunnen formele redeneringen en analyses over (formele) systeemmodellen op hun correctheid gecontroleerd worden.

Formele methoden kunnen worden ingezet binnen het gehele ontwikkeltraject van software en hardware, van de inventarisatie van de wensen van de klant, het ontwerp, de implementatie, de testfase tot en met het systeemonderhoud. Dit proefschrift richt zich in het bijzonder op *formele verificatie.* Dit houdt in dat men van een systeem wiskundig bewijst dat het voldoet aan de gewenste systeemvereisten, ook wel *specificatie* genoemd. Ook hier zorgt de wiskundige aanpak voor een grote mate van zekerheid. Een systeem dat formeel correct bewezen is, heeft een gerede kans dat het ook doet wat het moet doen.

Dit laatste geldt in mindere mate voor correctheidsargumenten die gebaseerd zijn op informele of semi–formele methoden. Indien de correctheid van een systeem met informele redeneringen aangetoond wordt, kunnen er gemakkelijk onduidelijkheden, impliciete — en mogelijk onjuiste aannamen — en onvolkomenheden ontstaan. Informele en semi–formele methoden hebben het voordeel dat zij sneller en makkelijker toe te passen zijn en complexere systemen aankunnen. Hoewel zij tot nu toe de voorkeur van de industrie hebben, worden formele methoden steeds vaker toegepast.

Tijd, Parameters en Kansen

Dit proefschrift gebruikt formele methoden om fundamenteel inzicht te krijgen in systemen waarin kansen, parameters en tijd een belangrijke rol spelen. Kansen, tijd en parameters zijn alle drie belangrijke mechanismen om de werkelijkheid te beschrijven. Zij spelen ook een belangrijke rol in het modelleren en construeren van software en hardware.

Tijd speelt in veel reactieve systemen een belangrijke rol, omdat veel reactieve systemen tijdkritisch zijn, dat wil zeggen zij moeten aan bepaalde tijdseisen voldoen om correct te zijn. Zo is de timing van het openen en sluiten van de bomen van een spoorwegovergang kritisch, omdat de bomen gesloten moeten zijn voordat de trein bij de overgang is.

Parameters maken het mogelijk om een klasse van systemen met dezelfde structuur te beschrijven. Men kan zich dan afvragen voor welke parameterwaarden het systeem correct is en voor welke niet. Inzicht in de voorwaarden op de parameters die nodig zijn voor de correctheid van het systeem, leveren belangrijke informatie over het functioneren ervan. Zij geven bijvoorbeeld inzicht in beperkingen op de toepassing van het systeem in toekomstige applicaties of binnen een andere context.

Een belangrijke plaats in dit proefschrift wordt ingenomen door systemen waarin *kansen* een rol spelen, de zogeheten probabilistische systemen. Kansen zijn, niet alleen in de informatica, een belangrijk hulpmiddel om systemen te beschrijven en analyseren. Zo vormen kansen een belangrijk hulpmiddel om de prestatie (performance) van reactieve systemen te analyseren. Ook worden zij gebruikt om onbetrouwbare systeemcomponenten te modelleren.

In dit proefschrift zijn we echter vooral geïnteresseerd in de toepassing van kansen in gedistribueerde algoritmen en communicatieprotocollen. In deze computersystemen — een speciale klasse van reactieve systemen — worden bepaalde beslissingen genomen op basis van een kansmechanisme, bijvoorbeeld het gooien van een dobbelsteen.

Het gebruik van kansmechanismen in deze systemen heeft drie belangrijke voordelen. Probabilistische methoden zijn vaak efficiënter in tijd– of geheugengebruik dan niet–probabilistische. Bovendien kan het gebruik van probabilistische keuzen leiden tot uniformere oplossingen. Tenslotte zijn er problemen, zoals het Byzantijnse generaalsprobleem, waarvoor bewezen is dat klassieke oplossingen niet bestaan, maar probabilistische wel. Het gebruik van kansmechanismen heeft echter ook een belangrijk nadeel. Het ontwerp en de verificatie van op kansen gebaseerde reactive systemen zijn meestal zeer complex. Zelfs specialisten op dit gebied hebben oplossingen bedacht die later niet correct bleken. Dit onderstreept nogmaals de noodzaak van rigoureuze wiskundige analyse.

Alea jacta est

Dit proefschrift presenteert een aantal wiskundige methoden voor het formeel specificeren en verifiëren van systemen met kansen, tijdseisen en parameters, waarbij probabilistische systemen de hoofdmoot vormen. We gebruiken Formele methoden om fundamenteel inzicht te krijgen in de eigenschappen en het gedrag van dergelijke systemen. We houden ons bezig met vragen als: *Hoe kunnen we systemen met kansen, tijd en/of parameters het best beschrijven? Hoe kan het gedrag van dergelijke systemen geformaliseerd worden? Wat zijn goede methoden om zulke systemen correct te bewijzen?* Tenslotte bekijkt dit proefschrift een kleine industriële toepassing waarin tijd, kansen en parameters voorkomen, te weten het Root Contention–protocol uit de IEEE 1394 'FireWire' standaard.

Resultaten

De belangrijkste resultaten van dit proefschrift worden uiteengezet in de Hoofdstukken 4 tot en met 7. Zij kunnen als volgt worden samengevat.

- Hoofdstuk 4 presenteert een testing scenario dat een rechtvaardiging geeft voor de bestaande definitie van het externe gedrag van probabilistische systemen uit de literatuur.
- In Hoofdstuk 5 ontwikkelen we een bewijstechniek, namelijk een bepaald type probabilistische (bi–)simulatie. Deze (bi–)simulatie staat het toe om, in zekere mate, te abstraheren van interne berekeningsstappen, maar kan toch efficiënt geautomatiseerd worden.
- Hoofdstuk 6 presenteert een zogenaamde *model checking* techniek die het mogelijk maakt om voor getimede systemen de exacte parametervergelijkingen te synthetiseren die nodig zijn, wil het systeem een bepaalde eigenschap bezitten.
- In Hoofdstuk 7 worden methoden en technieken uit de voorgaande hoofdstukken gebruikt om het Root Contention–protocol uit de IEEE 1394 standaard correct te bewijzen. We analyseren zowel het probabilistische, als het parametrische en het tijdsgedrag van dit protocol.

Overzicht

Hoofdstuk 1 leidt het proefschrift in. Het bespreekt de modellen die in de rest van het proefschrift gebuikt worden voor het analyseren van reactieve systemen. Deze modellen zijn gebaseerd op toestandovergangssystemen. In dit hoofdstuk wordt uitgelegd hoe het basismodel kan worden uitgebreid met tijd, parameters en kansen. Voorts bespreekt Hoofdstuk 1 de basisideeën achter de verificatiemethoden die we verder ontwikkelen en toepassen in latere hoofdstukken. De modellerings– en verificatiemethoden worden geïllustreerd aan de hand van een klassiek voorbeeld uit de literatuur, namelijk een sterk vereenvoudigd model van een spoorwegovergang.

Hoofdstuk 2 van dit proefschrift bespreekt het probabilistische–automatenmodel (PA–model). Dit raamwerk is ontwikkeld door Roberto Segala voor het beschrijven en analyseren van reactieve systemen met discrete kansen. Het PA–model vormt de basis voor volgende hoofdstukken in het proefschrift en Hoofdstuk 2 legt de achterliggende gedachten en theoretische concepten uit. Binnen het PA–model worden systemen gemodelleerd als toestandsovergangsmodellen. Dat wil zeggen, op ieder moment bevindt het systeem zich in een bepaalde toestand en veranderingen in het systeem worden gemodelleerd met behulp van toestandsovergangen, van de oude toestand naar de nieuwe toestand van het systeem. Naast probabilistische keuze, is ook nondeterministische keuze essentieel in het PA–model. De combinatie van beide maakt dit model vrij lastig. In het bijzonder wordt uitgelegd hoe deze combinatie leidt tot de notie van *trace distributies* van een PA, die het externe gedrag van een probabilistisch systeem beschrijven.

In Hoofdstuk 4 wordt een testing scenario gepresenteerd voor probabilistische automaten. Dit wil zeggen dat we op een intuïtieve manier aangeven hoe het externe gedrag van een PA geobserveerd kan worden. Dit doen we door de uitvoer een aantal onafhankelijke executies van de PA te analyseren met behulp van statistische methoden. Iedere executie van de PA levert een rijtje acties. Bij een willekeurige verzameling rijtjes toetsen we de hypothese "Deze verzameling rijtjes is gegenereerd door de gegeven PA." De klassieke wiskundige theorie over hypothese toetsen zal de hypothese aannemen dan wel verwerpen. Zo kunnen we van iedere verzameling rijtjes bepalen of het redelijk is om aan te nemen dat deze van een gegeven PA afkomstig is. Het observeerbare gedrag van een PA wordt daarmee gedefinieerd als precies die verzamelingen rijtjes die redelijkerwijs afkomstig zijn van de PA, waarvoor de hypothese dus wordt aangenomen.

Dit testing scenario, dat wil zeggen deze definitie van observeerbaar gedrag, levert precies de trace distributies als observaties van een PA op. Preciezer gezegd, twee PAs hebben dezelfde trace distributies precies dan als zij dezelfde observaties hebben. Dit is een belangrijke aanwijzing dat we met de trace distributies inderdaad een geschikte definitie van het concept 'extern gedrag' te pakken hebben.

Een belangrijke technische bijdrage van dit hoofdstuk is het zogenaamde *approximatie–inductie principe*. Deze stelling zegt dat als twee eindig vertakkende PAs hetzelfde eindige gedrag hebben, zij ook hetzelfde oneindige gedrag hebben. Met andere woorden, om in te zien of twee PAs precies hetzelfde gedrag hebben, volstaat het te kijken naar de eindige gedragingen van de PAs.

Hoofdstuk 5 is gewijd aan bisimulatie– en simulatierelaties voor PAs. Bisimulaties zijn relaties op de toestandsruimte van een PA die toestanden met hetzelfde stapsgewijze gedrag aan elkaar relateren. Men kan bewijzen dat toestanden met hetzelfde stapsgewijze gedrag ook hetzelfde globale gedrag hebben, dat wil zeggen, dezelfde verzameling trace distributies. Bisimulaties vormen dus een methode om gelijkheid van trace distributies aan te tonen. Deze methode is veel eenvoudiger en gemakkelijker te automatiseren dan een direct bewijs van trace distributie–gelijkheid. Zoals uitgelegd in Hoofdstuk 2, komen correctheidsbewijzen vaak neer op het bewijzen van trace distributie–gelijkheid of trace distributie–inclusie. Daarmee zijn bisimulaties een nuttig hulpmiddel voor systeemverificatie.

Simulatierelaties kunnen worden gezien als asymmetrische versies van bisimulatierelaties. Het verschil is dat als een toestand s is gerelateerd aan een toestand t, t al het stapsgewijze gedrag van s heeft, maar misschien kan t meer. Een gevolg hiervan is dat alle trace distributies van s ook trace distributies zijn van t; andersom hoeft dat niet het geval te zijn. Simulaties zijn daarmee een (relatief eenvoudige) methode om trace distributie–inclusie te bewijzen. Aangezien correctheidsbewijzen vaak neerkomen op het aantonen van trace distributie–inclusie, zijn simulatierelaties, net als bisimulatierelaties, nuttig voor systeemverificatie.

Dit hoofdstuk geeft een overzicht van verschillende probabilistische (bi–)simulatierelaties uit de literatuur. Vervolgens wordt een nieuw type (bi–)simulatie geïntroduceerd, de zogenaamde *vertraagde (bi–)simulatie*, waarvan er vier varianten besproken worden. Deze (bi–)simulaties hebben de eigenschap dat zij op een bepaalde manier abstraheren van interne berekeningsstappen. Dit wil zeggen dat interne berekeningsstappen, die immers voor de buitenwereld onzichtbaar zijn, altijd gebruikt mogen worden om aan te tonen dat bepaalde toestanden hetzelfde stapsgewijze gedrag hebben. We geven een alternatieve definitie van vertraagde (bi–)simulaties in termen van *normfuncties.* Een belangrijk resultaat is dat vertraagde (bi–)simulaties effiënt door een computer berekend kunnen worden. In technische termen, de besproken relaties zijn beslisbaar in polynomiale tijd en ruimte.

Hoofdstuk 6 behandelt tijdparamterische systemen. We breiden het klassieke model van Alur & Dill [AD94], de zgn *getimede automaten*, uit met timing parameters. Dit levert het *parametrische getimede automaten model (PTA model)* op. In het model van Alur & Dill kunnen tijdseisen geformuleerd worden door middel van expressies met klokvariabelen. In het PTA model kunnen deze expressies ook tijdsparameters bevatten. Gegeven een uitspraak over het model, kunnen we bekijken voor welke waarden van de parameters deze uitspraak geldt, en voor welke parameterwaarden deze niet geldt. Het doel is om de precieze parametervergelijkingen af te leiden die nodig zijn om een uitspraak waar te maken.

De belangrijkste bijdrage van dit hoofdstuk is een precieze, wiskundige semantiek van het PTA model en een methode om bij een uitspraak de parametervergelijkingen te synthetiseren die nodig zijn om de uitspraak waar te maken. Deze methode hoeft in het algemeen niet te termineren. Het is echter bewezen [AHV93] dat geen enkele methode om parametervergelijkingen te vinden in alle gevallen termineert. In technische bewoordingen: parameter synthese is onberekenbaar. Het is uiteraard wel zo dat, als de beschreven methode termineert, deze ook het correcte resultaat oplevert. Deze methode om parametervergelijkingen te synthetiseren is geïmplementeerd in een prototype, dat we gebouwd hebben uitgaande van de bestaande tool Uppaal.

Een andere belangrijke bijdrage van het hoofdstuk is de definitie van een speciale klasse van PTAs waarvoor veel parametervragen wel termineren. In deze PTAs, *ondergrens/bovengrens automaten* genoemd, mag iedere parameter òf een bovengrens òf een ondergrens op de tijdsgrenzen in de PTA afdwingen, maar niet de ene keer een bovengrens en de andere keer een ondergrens.

We sluiten het hoofdstuk af met een aantal kleine, industriële case studies die aantonen dat de prototype implementatie op voorbeelden uit de praktijk goed en relatief snel werkt.

Hoofdstuk 7 betreft een case studie. We beschrijven en analyseren het Root Contention–protocol (RCP) uit de IEEE 1394 standaard. Deze standaard wordt onder de populaire benamingen FireWire en iLink verkocht in onder andere laptops en PC's. De standaard definieert een manier om multimedia–apparatuur, zoals TV's, video recorders en PC's, met elkaar te verbinden tot een netwerk dat hen in staat stel data, bijvoorbeeld video beelden, uit te wisselen. Het Root Contention–gedeelte uit deze standaard specificeert een algoritme (i.e. een methode) om een leider te kiezen uit twee processen. Deze leider regelt het dataverkeer over het netwerk in latere fasen van het 1394 protocol. Hiertoe maakt het algoritme gebruikt van zowel random bits ("muntwerpen") als timing delays. Bovendien zijn timing parameters essentieel voor het correct functioneren van het protocol. Daarmee is dit protocol bijzonder geschikt om de praktische bruikbaarheid te onderzoeken van de analysemethoden uit eerdere hoofdstukken.

Het hoofdstuk begint met een overzicht van verscheidene analysemethoden uit de literatuur die zijn toegepast om RCP te verifiëren. Vervolgens bespreken we drie analyses van RCP die we zelf hebben uitgevoerd.

Eerst geven we een handmatige verificatie, waarbij de nadruk ligt op de probabilistische en timing aspecten van het protocol. We beschrijven het protocol en zijn specificatie in termen van het PA model uit Hoofdstuk 2. Vervolgens gebruiken we varianten van de simulatie relaties uit Hoofdstuk 5 om te bewijzen dat het protocol aan zijn specificatie voldoet. Hiertoe introduceren we een aantal hulpautomaten die tussen de implementatie en de specificatie-automaat inliggen, zodat we het verificatieproces kunnen opdelen in kleine, overzichtelijke deelstappen. Dit heeft in het bijzonder de probabilistische analyse sterk vereenvoudigd: de belangrijkste probabilistische stap wordt gemaakt in een simulatierelatie tussen twee heel kleine automaten (10 toestanden).

Vervolgens geven we een geautomatiseerde verificatie van het protocol met behulp van de model checker Uppaal. We maken een verbeterde versie van protocolmodel dat de communicatie tussen de processen op een realistischere manier weergeeft. Van dit model analyseren we vooral het timing en parametrische gedrag. We gebruiken Uppaal om (aangepaste versie van) de simulatierelaties uit het voorgaande verificatieproces en onderzoeken voor welke parameterwaarden de simulatierelaties geldig zijn. Dit is enigszins omslachtig omdat Uppaal niet ontworpen is voor dit soort taken. Toch slagen we erin de gewenste eigenschappen te onderzoeken – een uitbreiding van Uppaal met enkele eenvoudige operaties zou dit proces echter vergemakkelijken. Omdat de standaardversie van Uppaal geen parametrische analyse kan uitvoeren en omdat de parametrische versie van Uppaal nog niet beschikbaar was toen we dit onderzoek uitvoerden, checken we de parametervergelijkingen op een experimentele manier. We controleren voor een groot aantal parameterwaarden of de gewenste correctheidseigenschappen gelden. Hieruit leiden we de algemene parametervergelijkingen af. Dit geeft weliswaar geen wiskundige zekerheid over deze juistheid van deze vergelijkingen, maar naar ons idee toch betrouwbare resultaten. De parametervergelijkingen die we afleiden zijn eerder, informeel, afgeleid in een webdocument van de ontwerpers van het Root Contention–protocol.

Tenslotte gebruiken we de prototype parametrische model checker uit Hoofdstuk 6 om de parametervergelijkingen te vinden. We analyseren de modellen uit beide voorgaande verificaties. De resultaten die we met de parametrische model checker verkrijgen komen overeen met de resultaten uit de eerdere analyses.