Citius, Vilius, Melius

Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems

Ansgar Fehnker

Samenvatting

Et is better öm met 'n laaien te werken,
Äs met 'n dummen
Nedersaksische gezegde

Om een gevaarte met ruim 290 ton vloeibaar ruw ijzer van de hoogovens naar de gieterij te vervoeren is op zich al een hele taak. Temeer als niet één maar meerdere charges tegelijk verwerkt moeten worden, welke op weg naar de gieterij een aantal behandelingen moeten ondergaan die van charge tot charge en afhankelijk van de beoogde kwaliteit staal kunnen verschillen. Het wordt er niet makkelijker op als de machines die het ijzer behandelen hooguit één charge tegelijk kunnen verwerken, en als de twee kranen die men voor elk transport moet gebruiken hetzelfde spoor moeten delen; de ene kraan kan de andere de weg versperren, charges kunnen elkaar hinderen. En bovendien mag het ruwe ijzer eer het bij de gieterij arriveert niet te ver afkoelen, en moet de toevoer zodanig zijn, dat de gieterij een continue stroom staal naar de warmwalserij kan garanderen.

Dit planningsprobleem was een van de zes casestudy's van the EU onderzoeksproject *Verificatie van Hybride Systemen*. Naast de groep Informatica voor Technische Toepassingen van de Katholieke Universiteit Nijmegen namen een tiental onderzoeksinstellingen en bedrijven uit Nederland, België, Duitsland, Frankrijk, Zwitserland, Denemarken, Zweden en Israël deel aan dit project, dat het onderzoek op het gebied van Formele Methodes moest stimuleren. De bedoeling was om inzicht te krijgen hoe met name verificatietechnieken voor hybride systemen kunnen bijdragen aan de oplossing van enkele uitdagende problemen. Het boven beschreven probleem werd voorgesteld door Sidmar, een vlakstaal-producent te Gent, België.

Een systeem is *hybride* als het gedrag van het systeem essentieel bepaald wordt door de interactie van discrete componenten zoals microcontrollers met continue processen waarin fysische grootheden en tijd een rol spelen. Met *Formele Methoden* worden in de informatica methoden en technieken bedoeld om de correctheid van een systeemontwerp mathematisch aan te tonen. Men gebruikt op wiskunde gebaseerde talen om een hard- of softwaresysteem te beschrijven, die ons dan in staat stellen om te bewijzen of een ontwerp aan gewenste eigenschappen voldoet. Dit proces wordt *verificatie* genoemd. Typerend is te laten zien dat een systeem nooit in een onveilige toestand kan verkeren, waarbij onveilig bijvoorbeeld op een rekenfout, op fout ontvangen informatie of op het overschrijden van de maximaal toelaatbare fysische grootheid kan slaan. Een voorbeeld waarbij een afrondfout tot een verkeerd gemeten snelheid, en uiteindelijk tot een explosie heeft geleid is het mislukken van de eerste vlucht van de Ariane 5 raket.

Het met de hand bewijzen van de correctheid van een ontwerp kan zelfs vooreen klein model een nogal tijdrovende en vooral fout gevoeligebezigheid zijn. *Model checking* is een van de meest gebruikte computerondersteunde technieken voor verificatie; men begint bij de beginsituatie van het systeem, en loopt dan automatisch alle mogelijke paden die het systeem kan nemen af. Zodoende wordt bijvoorbeeld geverifieerd dat het systeem nooit in een onveilige toestand kan belanden. Als traditionele algoritmen voor model checking een onveilige toestand en dus een fout vinden, kunnen deze ook informatie verstrekken hoe deze fout bereikt kan worden. Deze informatie is belangrijk om na te trekken waar de fout in het ontwerp zit. Voor bepaalde klassen modellen is gegarandeerd dat het model-checkingalgoritme altijd stopt. Men zegt dan dat het model-checkingprobleem voor deze klasse *beslisbaar* is.

Met een *automaat* wordt een formeel model bedoeld dat het gedrag van een hard- en softwaresysteem beschrijft als overgangen van één toestand naar de andere. Getimede automaten zijn een variant die toestaat om het continue verstrijken van de tijd te modelleren Ondanks dat continue tijd tot een oneindig en zelf overaftelbaar aantal toestanden leidt, is het model-checkingprobleem voor getimede automaten beslisbaar als men met verzamelingen toestanden werkt. Model-checkingtools voor getimede automaten, zoals Uppaal en KRONOS, hebben in de afgelopen jaren aangetoond dat zij een belangrijke bijdrage kunnen leveren aan het correcte ontwerp van soft- en hardwaresystemen.

Om terug te keren tot de staalfabriek van Sidmar. Hier gaat het er niet om om een fout in het ontwerp te vinden, maar om een oplossing voor een planningsprobleem te bepalen. We zijn niet geïnteresseerd om uit te sluiten dat ooit een onveilige toestand bereikt kan worden, maar we willen aantonen dat een gewenste toestand bereikbaar is, namelijk een toestand waarin alle charges in de gewenste volgorde en op tijd bij de gieterij gearriveerd zijn. Het is mogelijk het planningsprobleem als getimede automaat te modelleren en om een model-checkingalgoritme naar een gewenste toestand te laten zoeken. Het “tegenvoorbeeld” dat aantoont hoe deze toestand te bereiken is, kan vervolgens als oplossing van het planningsprobleem worden beschouwd.

Er zijn echter ook verschillen tussen traditionele algoritmes die optimisatieproblemen oplossen en het model-checkingalgoritme. Ten eerste maakt het model-checkingalgoritme geen verschil tussen goede en slechte oplossingen. De kosten die aan een oplossing worden verbonden kunnen bijvoorbeeld afhangen van de totale tijd die nodig is om het bijbehorende rooster uit te voeren. Model checking wordt gebruikt om fouten te vinden, en zal stoppen zodra het de eerste fout tegenkomt. Als we hetzelfde algoritme gebruiken om planningsproblemen op te lossen, zal het stoppen zodra het een oplossing vindt, ongeacht de kosten die eraan verbonden zijn. Er is geen garantie dat dit de optimale of zelfs maar een goede oplossing is.

Een ander verschil is dat men bij het model checken de optimistische aanname maakt, dat men geen ongewenste toestand tegen zal komen, en dat men dus alle toestanden moet gaan doorzoeken. Het algoritme is derhalve zo opgezet dat het de gehele toestandsruimte efficiënt exploreert. Als we een optimisatieprobleem oplossen is daarentegen bekend dat de toestandsruimte een gewenste toestand bevat. Vaak heeft men enig idee waar een (goede) oplossing te vinden valt, en als het mogelijk zou zijn om gericht naar een oplossing te zoeken, zou dit de tijd die het algoritme nodig heeft drastisch kunnen verkorten.

Dit proefschrift onderzoekt hoe het model-checkingalgoritme voor getimede automaten uitgebreid kan worden, opdat het geschikt kan worden gemaakt voor het oplossen van planningsproblemen. Het proefschrift laat eerst zien hoe planningsproblemen als getimede automaat gemodelleerd kunnen worden. Vervolgens wordt een notie van kosten geïntroduceerd. We introduceren een gemodificeerd model-checkingalgoritme, en tonen aan dat dit desondanks gegarandeerd ooit zal stoppen, en dat het, mits een oplossing bestaat, de optimale oplossing vindt. Verder wordt aandacht besteed aan efficiënte datastructuren, en aan hoe het algoritme aangepast kan worden, zodat gericht zoeken naar goede oplossingen mogelijk wordt. Deze aanpassingen kunnen echter ook een positief effect hebben op model checking ter verificatie. Het proefschrift introduceert een model-checkingalgoritme voor een algemenere klasse hybride systemen, en onderzoekt of gericht zoeken ook voor deze klasse een positieve invloed op de performance kan hebben.