Model-checkers als slimme schroevendraaiers

In een aantal praktijkcases hebben studenten van de Radboud Universiteit met succes model-checking toegepast. Dit suggereert dat de techniek een eenvoudig en breed inzetbaar gereedschap aan het worden is, betoogt de Nijmeegse hoogleraar Frits Vaandrager.

Frits Vaandrager

Dit weekeinde heb ik de batterij van mijn tomtom vervangen. Volgens Tomtom is het absoluut niet de bedoeling dat je dat zelf doet, maar ik wilde dit stukje Nederlandse hightechtrots wel eens van binnen zien. Met behulp van instructies die ik van internet had geplukt en een minuscule schroevendraaier was het klusje zo gepiept. Goed gereedschap is het halve werk. Zo vaak gebruik ik die schroevendraaier niet, maar in dit geval bleek hij onmisbaar.

Een *model checker* is ook een stuk gereedschap, een soort slimme schroevendraaier, waarmee embedded-softwareontwikkelaars hun systemen kunnen doormeten en verbeteren. Je hebt dit gereedschap niet dagelijks nodig, maar soms is het verdraaid handig voor het vinden van fouten of verbeterpunten. En het gebruik is eenvoudiger dan velen denken.

Op de Radboud Universiteit Nijmegen hebben we studenten met model-checking laten stoeien in een aantal praktijkcases. Daarbij zijn ze er bij herhaling in geslaagd om binnen enkele dagen modellen te bouwen van realistische toepassingen en interessante resultaten te verkrijgen. En dat zonder voorafgaande kennis van het domein en met slechts beperkte ervaring in model-checking. Dit suggereert dat de aanpak zich ontwikkelt tot een gereedschap dat, net als de schroevendraaier, eenvoudig en breed inzetbaar is.

Aansprekende toepassingen

Model-checking is een techniek om te bewijzen dat een systeemontwerp voldoet aan de eisen die eraan worden gesteld. Een model-checker is een computerprogramma met als invoer dat ontwerp (het 'model') en een gewenste eigenschap (de 'specificatie'). Het gereedschap berekent vervolgens of het model voldoet aan de specificatie. Zo niet, dan geeft het een tegenvoorbeeld. Door dit tegenvoorbeeld te bestuderen, kun je erachter komen wat de oorzaak is van het probleem. Na model en/of specificatie te hebben verbeterd, kun je het dan nog eens proberen. Het idee is dat door een groot aantal eigenschappen te checken, het vertrouwen in de correctheid van het systeemontwerp toeneemt.

Model-checkers richten zich op modellen die dynamisch systeemgedrag beschrijven. Ze analyseren in welke toestanden een systeem zich kan bevinden en welke overgangen daartussen mogelijk zijn. Eigenlijk doen ze niet veel meer dan razendsnel grote toestandsruimtes doorzoeken.

Inmiddels zijn er duizenden aansprekende toepassingen. In de hardware-industrie is het gebruik van model-checking en vergelijkbare aanpakken uitgegroeid tot een standaard praktijk sinds Intel in 1994 475 miljoen dollar verloor door een fout met de drijvendekommadeling in zijn Pentium-processor. Zelf heeft de chipgigant bijvoorbeeld het ontwerp van de Pentium IV voor twintig procent gecontroleerd met deze technieken.

Bij Nasa hebben ze de Spin-model-checker gebruikt om de multithreaded *plan execution*-module voor de Deep Space 1-missie correct te bewijzen. Dit heeft vijf fouten aan het licht gebracht die nog niet eerder naar boven waren gekomen. Gerard Holzmann, de van oorsprong Nederlandse maker van Spin, werkt tegenwoordig bij het Jet Propulsion Laboratory van de Amerikaanse ruimtevaartorganisatie. Hij geeft daar leiding aan het team dat verantwoordelijk is voor de kwaliteit van de software die Nasa gebruikt bij missies, zoals in de Curiosity-rover op Mars.

In 2009 heeft pionier in model-checking Ed Clarke van de Amerikaanse regering een subsidie van tien miljoen dollar gekregen voor een prestigieus Expeditions in Computing-project. Hierin past hij de techniek nu toe om fouten te vinden in de besturingssoftware van vliegtuigen en auto's, maar ook om de oorzaken van alvleesklierkanker en atriumfibrilleren op te sporen. Volgens Clarke zijn dezelfde klassen van modellen en analysetechnieken inzetbaar in zowel het embedded- als het medische domein.

'Echte' problemen

Ondanks de zware wiskundige artikelen die erover zijn geschreven, vereist model-checking geen promotie in de theoretische informatica. Zo heb ik een lesmodule over de aanpak ontwikkeld voor middelbare scholieren, als onderdeel van de methode Informatica-Actief. Na een training van één les analyseren 5-vwo-leerlingen hun eerste modellen met de model-checker Uppaal en na tien lessen hebben ze gerekend aan flappentappers, frisdrankautomaten, spoorwegovergangen en algoritmes om een leider in een ringvormig netwerk te kiezen.

Sinds een aantal jaren gebruiken eerstejaarsstudenten informatica in Nijmegen model-checkers bij de cursus 'Processen en concurrency'. De ervaring is dat zij hierdoor sneller en beter inzicht krijgen in concurrencyproblemen en de werking van synchronisatieprimitieven zoals locks, monitoren en semaforen. Dit jaar heb ik hun gevraagd een bekend of minder bekend synchronisatieprobleem te kiezen uit 'The little book of semaphores' van Allen Downey en zijn semafoorgebaseerde oplossing te modelleren en te analyseren met Uppaal. Hierbij hebben we serieuze fouten gevonden in vijf oplossingen. En dat terwijl Downey een gerespecteerd onderzoeker is en specialist op het gebied van concurrency. Er is maar één conclusie mogelijk: deze vraagstukken zijn zo complex dat de oplossingen alleen correct zijn te krijgen met ondersteuning van (bijvoorbeeld) een model-checker.

In de Nijmeegse masteropleiding informatica verzorg ik een cursus waarbij de deelnemers model-checkers gebruiken om embedded systemen uit de praktijk te analyseren. Behalve dat het motiverend is om te worstelen met 'echte' problemen, leren studenten wat er allemaal komt kijken bij het modelleren van complexe applicaties en krijgen ze inzicht in de sterke en zwakke kanten van de tooling. Zo hebben ze onder meer gewerkt aan de 8b/10b-codering in het Sata-protocol (gebruikt voor communicatie tussen CPU en harddisk), het ontdekken van nieuwe apparaten in het Bluetooth-protocol, scheduling van *tilt tray*-sorteermachines bij Vanderlande, scheduling van het datapad in printers van Océ en kloksynchronisatie in het draadloze Myrianed-sensornetwerk van Chess/Devlab. Ook al leidde het gebruik van model-checkers niet in alle gevallen tot nieuwe inzichten, toch blijkt het construeren van een model een uitstekende manier voor studenten om te begrijpen hoe een systeem werkt.

Een zeer recente case komt van NXP's Industrial Technology and Engineering Centre (Itec) in Nijmegen. Deze groep ontwikkelt assemblagemachines voor discrete halfgeleiders. Een van de machines, de Die-Bonder Strip Glue (DBSG), bestaat uit verschillende modules die ieder een aantal toestanden hebben en die moeten samenwerken om strips te laden, er druppels lijm op te laten vallen, er vervolgens chips (dies) op te plaatsen en het geheel ten slotte af te voeren. Een centrale besturingseenheid stuurt periodiek opdrachten naar de modules, op basis van hun toestanden op dat moment. De kern van de besturingssoftware bestaat uit een lijst van 93 regels die, gegeven de toestand, de volgende opdracht voor iedere module specificeren.

Itec had twee vragen voor de studenten. Ten eerste: zijn er globale configuraties bereikbaar van de modules waarvoor er geen regel is gespecificeerd voor de besturingseenheid? En ten tweede: zijn er regels die nooit kunnen vuren omdat de globale configuratie waarin ze worden getriggerd niet bereikbaar is? De studenten hebben deze vragen bestudeerd met behulp van de NuSMV2-tool, een state-of-the-art checker voor modellen met een eindig aantal toestanden. Hierbij hebben ze gewerkt in twee onafhankelijke groepen, die elk veertig uur hadden om met een oplossing te komen. De studenten hadden wel ervaring met het gebruik van model-checkers, maar niet met NuSMV2.

Na een paar iteraties (met goede feedback vanuit Itec) hebben beide teams een werkend model weten te construeren van de DBSG. Hierbij zijn een paar fouten boven water gekomen in de tabel met toestandsovergangen. Een van de teams heeft vastgesteld dat er geen regels ontbreken. Vanwege de complexiteit van de DBSG was er echter onvoldoende tijd om ook het gedrag van de (menselijke) machine-operator te modelleren, waardoor deze conclusie voorlopig is. Het andere team heeft geconstateerd dat in het model 21 van de 93 regels nooit worden gebruikt. Na bestudering van deze 21 regels heeft NXP geconcludeerd dat het inderdaad onwaarschijnlijk is dat de machine ze zal gebruiken. De Itec-groep overweegt om model-checking standaard te gaan gebruiken bij de systeemontwikkeling.

Frits Vaandrager is hoogleraar informatica voor technische toepassingen aan de Radboud Universiteit Nijmegen. Tijdens Bits&Chips 2012 Embedded Systems op 8 november zal hij, samen met Jorg van Daelen van NXP, dieper ingaan op de model-checkingcase bij Itec.

Redactie Nieke Roos