TU Twente onderzoekt geautomatiseerd modelgebaseerd testen van reactieve systemen

TorX genereert zelf

Veel embedded systemen zijn reactief; zij ontvangen en sturen signalen van en naar de buitenwereld. Hun gedrag kan dan formeel worden gespecificeerd als een model van toestanden en overgangen daartussen. Modelgebaseerd testen onderzoekt of het gedrag van de implementatie afwijkt van het gedrag van de specificatie. Door Nederlandse universiteiten en onderzoeksinstituten, met de TU Twente als centrum, is een tool ontwikkeld voor automatische generatie, uitvoering en analyse van dit soort tests. Het vrij beschikbare TorX kan zelf, tijdens de uitvoering van het programma, nieuwe tests opzetten.

HANS VAN THIEL

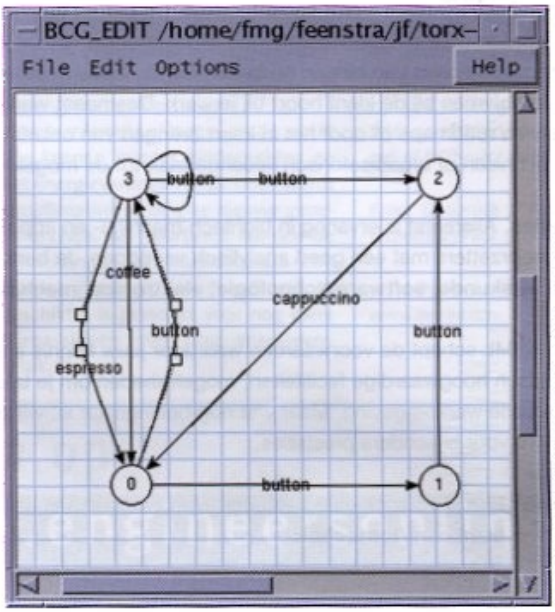
"Wij verwachten veel van toepassing in embedded systemen," zegt dr. Arend Rensink, verbonden aan de afdeling 'Formal Methods en Tools', faculteit 'Electrical Engineering, Mathematics and Computer Science' van de TU Twente. "De acceptatie van fouten bij embedded systemen is veel lager dan die voor andere software, en formele methoden verminderen de kans op fouten. Maar in de praktijk is er veel voorbereidend werk nodig. Ons onderzoek richt zich op het testen van reactieve systemen, systemen die impulsen ontvangen van de buitenwereld en zelf impulsen doen uitgaan naar buiten. Zulke systemen kun je specificeren als toestanden met overgangen daartussen en daarvoor bestaan talen als Lotos, Promela, SDL en Fsp. Dan heb je een model van het gedrag van je systeem, en modelgebaseerd testen houdt in dat je onderzoekt of je implementatie zich net zo gedraagt als je model. Dit is dus een vorm van 'black box testing', je observeert alleen het gedrag."

"Maar in de praktijk is er meestal geen specificatiemodel, dus dat moet je eerst zelf maken. Overigens kan dit op zich al ontwerpfouten aan het licht brengen, maar het gaat vooraf aan het eigenlijke testen. Een tweede praktisch probleem is dat de testsoftware moet kunnen communiceren met het SUT (System Under Test) en in de praktijk moet dan voor elk SUT een eigen 'adapter' module worden geschreven. De door ons ontwikkelde tool voor automatische testgeneratie, -uitvoering en -analyse, TorX, heeft dan ook vier modulen, Spec, Driver, Adapter en SUT. Het laatste kan dan C-code zijn, hardware, maar ook een simulator."

Testen

"Bij modelgebaseerd testen wil je controleren of een implementatie zich inderdaad zo gedraagt zoals je specificatie voorschrijft," vervolgt Rensink. "Daartoe maak je ook van je implementatie een formeel model en dan kan je gebruik maken van de zogenoemde ioco testtheorie om correctheid vast te stellen. Het punt is natuurlijk dat die twee toestandsmodellen, beschreven in een specificatietaal of -format, niet identiek zijn. In je specificatie kan, bijvoorbeeld, één actie gespecificeerd zijn met één pijl, terwijl je implementatie daar twee acties voor gebruikt, in het model twee pijlen met een extra toestand. Denk aan de implementatie van een communicatieprotocol over mogelijk verschillende routers."

"Met de ioco testtheorie kun je volgens een algoritme een testsuite genereren



Figuur 1. Een koffieautomaat is een reactief systeem. De figuur toont een grafische weergave in TorX van een modelspecificatie van toestanden en overgangen. Elke pijl representeert een actie van de gebruiker, dan wel een actie van het systeem.

tests

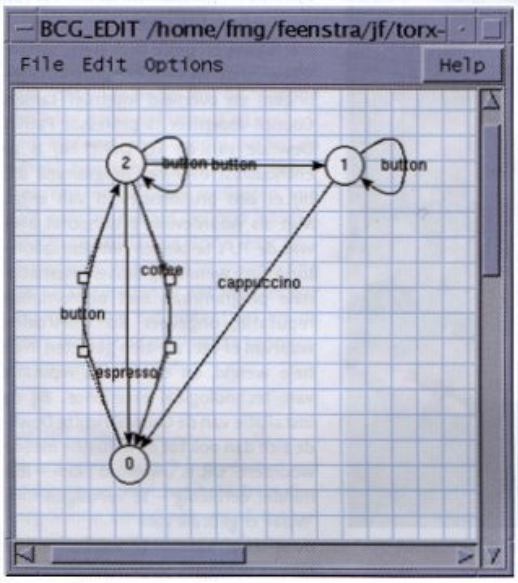
die je specificatiemodel als uitgangspunt neemt en elk implementatiemodel daaraan toetst. Het is formeel te bewijzen dat als een test faalt, de implementatie niet correct is, en dat als de implementatie niet correct is, er ook een test is die dat kan aantonen. Hiermee heb je dus een mathematisch-logische basis voor je testen, en dat is natuurlijk heel waardevol. Het aantal mogelijke tests is echter oneindig groot en dus moet je keuzes maken."
Rensink laat op een whiteboard zien hoe een model met slechts drie toestanden, met non-determinisme en 'quiescence', al enorm veel verschillende mogelijkheden heeft. Non-determinisme wordt aangegeven door meerdere overgangen met eenzelfde input of output vanuit een toestand, en 'quiescence' is de vakterm voor de afwezigheid van elke vorm van uitvoer van het systeem.
"Ons onderzoek naar modelgebaseerd testen is zo'n tien tot vijftien jaar geleden begonnen," vervolgt hij. "De achtergrond, toen en nu, is dat 30 ... 70% van een ontwikkeltraject bestaat uit testen. Dat moet minder kunnen."
"De basis werd gelegd met een theorie van puur reactieve beschrijvingen en die is in de loop der jaren uitgebreid naar systemen in context. Dat zijn systemen die je niet geïsoleerd kunt testen maar alleen als subsysteem binnen een geheel. Denk bijvoorbeeld aan een SUT bovenop het UDP communicatieprotocol. De volgorde van aankomst van UDP pakketten is niet gegarandeerd, dus daar moet je rekening mee houden.
Een volgend onderwerp is het modelleren van tijd. Hoe lang moet je wachten voordat je kunt concluderen dat er geen overgang zal plaatsvinden, hoe kun je timeouts in protocollen specificeren en testen, en alle andere tijdgerelateerde aspecten. Iemand van hier zal daar binnenkort op promoveren."

TorX

De TU Twente werkt niet alleen samen met de andere technische universiteiten in Nederland maar ook met de industrie. Zo is de TorX-tool ontwikkeld met de TU Eindhoven, Philips Research en Lucent Technologies binnen een project dat gefundeerd is door de Technologiestichting STW.
Axel Belinfante is degene die binnen de Twentse vakgroep verantwoordelijk is voor de implementatie van TorX en hij demonstreert op een laptop hoe een en ander er uitziet. In de loop der tijd heeft TorX een uitgebreide grafische interface gekregen (optioneel) en Belinfante laat zien hoe de gebruiker interactief kan werken en hoe de modellen met hun transities zijn gevisualiseerd. Daarnaast wordt de voortgang ook voortdurend weergegeven in een sequentiediagram.
"Maar," benadrukt Belinfante nog eens, "TorX kan tijdens het testen zelf nieuwe tests genereren. Het gaat om meer dan het uitvoeren van een van tevoren vastgelegde testsuite."
Een voorbeeld hiervan wordt gegeven door een simulatie van een in C geschreven conferentieprotocol, dat UDP gebruikt, en waar expliciete fouten zijn ingebracht. Op verschillende versies van het protocol worden, zonder bemoeienis van de menselijke tester, verschillende tests uitgevoerd.

De tool is het product van een uitgebreid onderzoeksproject onder de naam Côte de Resyste – Conformance Testing of Reactive Systems (zie referentie 1) en het is in dat kader toegepast op verschillende applicaties. Naast het genoemde conferentieprotocol zijn dit de 'Payment Box' die door Interpay is ontwikkeld voor het rekeningrijden, het EasyLink-protocol van Philips voor de communicatie tussen een videorecorder en een TV-toestel, en een component van het Cell Broadcast Centre van LogicaCMG. Het is ook gebruikt voor testen van een implementatie van Lucent van een ETSI-standaard, en voor besturingssoftware van de Oosterscheldekering.

Zoals gezegd vervult TorX een belangrijke rol in het verdere onderzoek van de TU Twente en samenwerkende partners. Een van die partners is ASML (onder andere) en de afgelopen jaren is binnen het Tangram-project het testen van een ASML waferstepper onderzocht.
De oorspronkelijke financiers van het onderzoek naar modelgebaseerd testen hebben ermee ingestemd de tool vrij te geven, en de huidige status van TorX (zie referenties) is open source volgens de Apache-licentie. ■



Figuur 2. Een model van de implementatie van een koffieautomaat. Vergeleken met de specificatie ontbreken een toestand en twee overgangen. Automatische testgeneratie, -uitvoering en -analyse door TorX kan vaststellen of de implementatie correct is. (Bron figuren 1 en 2: http://fmt.cs.utwente.nl/tools/torx/torx-examples.html

Referenties:
Tretmans, J., Brinksma, E., 'TorX: Automated Model Based Testing', First European Conference on Model Driven Software Engineering, December 11 ... 12, 2003, Neurenberg, Duitsland, p31-43
http://fmt.cs.utwente.nl/tools/torx/introduction.html
Zie voor andere embedded systems onderzoeksprojecten van de vakgroep 'Formal Methods and Tools': http://fmt.cs.utwente.nl/research.html