'Met Promela kun je samenwerkende en compact specificeren'

Gerard Holzmann

'Auto in interacties abstract

In 1983 is de ACM Software System Award voor het eerst toegekend aan Ken Thompson en Dennis Ritchie voor hun werk aan Unix. Latere ontvangers waren onder meer Vincent Cerf en Robert Kahn (1991) voor TCP/IP, John Ousterhout (1997) voor Tcl/Tk en de 'Apache Group' (1999).

Dr. Holzmann werkt sinds 1980 in de VS bij Bell Labs, dat thans onderdeel is van Lucent. De Spin-software, die overigens vrij verkrijgbaar is en wereldwijd door duizenden ontwikkelaars wordt gebruikt, is ook toegepast op de PathStar telefoonswitch van Lucent. Over methoden en toepassingen rondom Spin worden regelmatig conferenties gehouden en er is ook een nieuwsbrief.

Holzmann heeft de achterliggende theorie al in 1990 beschreven in zijn boek, 'Design and Validation of Computer Protocols'. Een op zijn website beschikbaar artikel, 'The Model Checker Spin' (IEEE TSE 23, 5, (1997)), geeft een beknopte beschrijving met voorbeelden.

Spin werkt niet rechtstreeks op code, maar op een model daarvan in de formele taal Promela (process meta language). De laatste jaren hebben Holzmann en zijn team echter een methode ontwikkeld om zo'n model automatisch uit ANSI-C te kunnen extraheren. Met de hierop gebaseerde tool Feaver kan code automatisch worden geverifieerd en dat is ook uitgevoerd op de bovengenoemde PathStar access server.

'Feaver is niet vrij beschikbaar', vertelt Holzmann, 'maar ook andere researchgroepen werken aan automatische omzetting van code naar Promela. Bandera, bijvoorbeeld, is een vrij beschikbaar

krijgt ACM Software System Award

matische verificatie fractie van een seconde'

Director Computing Principles Research van Bell Labs, Gerard Holzmann, heeft de ACM Software System Award verkregen voor de model checker Spin, een generiek verificatiesysteem dat ontwerp en verificatie van 'concurrent process' systemen ondersteunt. Met Spin is formele verificatie van software praktisch toepasbaar. Zo is met Spin besturingssoftware van NASA voor ruimtevluchten op correctheid getoetst.

HANS VAN THIEL

tool van Kansas State University die dit doet voor Java'. Zelf werkt Holzmann aan de uitbouw van de Feaver tool naar C++, dat als programmeertaal veel complexer is dan C.
'Het uiteindelijke doel is om automatische verificatie een standaardonderdeel te maken van de compiler,' zegt Gerard Holzmann. 'Dat kan over vijf tot tien jaar al mogelijk zijn, als we de verificatietijd omlaag kunnen brengen tot een fractie van een seconde. Ik denk dat dat ons gaat lukken!'

PT Embedded Systems: Mijnheer Holzmann, allereerst gefeliciteerd met de ACM Software System Award. Voor alle duidelijkheid, Spin is toch een modelchecker voor gedistribueerde, concurrent software, en geen automatische debugger die op verkeerde pointers en dergelijke test?
Gerard Holzmann: Dank u! Wel, in 1990 zijn we bij Bell Labs begonnen met handgebouwde modellen van gedistribueerde controlsystemen, zoals van onder andere ruimtevaartuigen. Later zijn we erin geslaagd om die modellen ook rechtstreeks uit applicatiesoftware te extraheren. De verificatie bestrijkt dan ook NULL pointers, overschrijding van arrays, ongeïnitialiseerde variabelen e.d. Maar gedistribueerde systemen, zoals een telefoonsysteem met heel veel threads en functies, bijvoorbeeld zesvoudige conferentie en 'call waiting', zijn veel moeilijker correct te ontwerpen dan sequentiële software. We wilden voor dergelijke non-deterministische systemen ontwerptools bouwen en verificatiealgoritmen implementeren, die we dan later konden toepassen op programmeertalen als C, C++, Java enz.

PT Embedded Systems: Kunt u het principe van Spin heel algemeen toelichten?
Gerard Holzmann: Er zijn twee aspecten, het gedrag van een systeem en de correctheidseigenschappen, die je onafhankelijk van elkaar kunt specificeren. Met Promela kun je, abstract en vrij compact, samenwerkende interacties specificeren. Denk aan processen die berichten uitwisselen via buffers, gedeelde objecten of door synchrone hand-shaking (rendez-vous).
Onafhankelijk hiervan kun je correctheidsrelaties uitdrukken in een ander formalisme, LTL (Linear Temporal Logic). Dat is een propositielogica met operaties 'always, eventually en until' waarmee je een correctheidseis kunt uitdrukken, zoals 'Als je de hoorn van de telefoon oppakt moet je altijd een kiestoon horen'. Dat is een eis die onafhankelijk is van de implementatie van de telefoonsoftware. Het is een fout als de kiestoon niet gaat, en de verificator kan dat uitzoeken.
In Spin gebeurt dat door de negatie te formuleren, dus: het is niet geval, dat... Ofwel, er is een situatie te vinden dat, als je de telefoon oppakt, de kiestoon niet overgaat. Als de model checker, die alle mogelijke toestanden afgaat, dus zo'n situatie vindt, dan heb je een fout gevonden.
Dit is ook een voorbeeld dat echt is opgetreden tijdens het verifiëren van een commerciële telefoonswitch, de PathStar access server van Lucent. Die heeft een 'call forwarding' feature, en het bleek nu dat er een klein 'window' was, een 'tijdsduur', dat wanneer de hoorn werd opgepakt net op het moment dat een binnenkomend gesprek werd doorverbonden, de kiestoon dan niet overging. Dat 'window' is zo klein, dat vind je nooit met standaard softwaretests, maar de verificator loopt alle mogelijkheden af.
Normale foutafhandeling is niet het probleem bij softwareontwerp, daar wordt heus wel aan gedacht. Het gaat juist om zeldzame combinaties van onverwachte gebeurtenissen, fout 1 wordt gevolgd door fout 2 en fout 3... Het nut van formele verificatie is dat je de requirements ook in uiterst onwaarschijnlijke situaties toetst.

PT Embedded Systems: U gebruikt dus Promela voor het algoritme en LTL voor de correctheidsclaims. Hoe is dan de relatie tussen die twee?
Gerard Holzmann: Dat is de basis van Spin. Al in de jaren '60 had je een uitbreiding van standaard automatentheorie naar non-deterministische software. Sommige applicaties beginnen, doen iets en stoppen dan, maar gedistribueerde systemen, zoals de processen van een OS of een telefooncentrale blijven altijd bestaan. Oneindige executies zijn nu te beschrijven als zogenoemde omega-automaten, en Promela is een formalisme daarvoor. Het was A. Pnueli die LTL in computer science in- ›

troduceerde als het juiste mechanisme om over de executies in gedistribueerde systemen te redeneren. P. Wolper en A. Vardi toonden in de jaren '80 het verband aan met omega-automata.
In technische zin: elk omega-automaton heeft een eigen taal die door dat automaton wordt geaccepteerd en die taal correspondeert met elke executie in LTL die correct is.
Als je dus de negatie neemt van je claims in LTL, die transformeert in de corresponderende automatentalen en de verzameling (automaten)talen neemt van je Promela-model, dan moet de doorsnede van die twee verzamelingen leeg zijn. Is die doorsnede niet leeg, dan heb je een fout gevonden, en het mooie is nu dat je meteen ook een exact tegenvoorbeeld hebt gevonden. Je weet dus niet alleen dat er een bug is, maar ook welke!

PT Embedded Systems: U heeft voor Spin een grafisch front-end geschreven, XSpin, en uit bestaande C programmatuur kunnen automatisch modellen worden gegenereerd met Feaver, maar is het noodzakelijk om de theorie te beheersen om met Spin te valideren?
Gerard Holzmann: In het begin dacht ik inderdaad dat gebruikers eerst mijn boek zouden moeten lezen, maar in de praktijk blijkt dat veel mensen gewoon de tool downloaden en Spin uitproberen. Voor gevorderde toepassingen is het wel nodig dat je de theorie kent, maar standaard asserties of de afwezigheid van deadlock verifiëren is vrij gemakkelijk te doen, ook zonder theoretische achtergrond. Overigens is het aantal downloads wel flink toegenomen nadat ik XSpin heb toegevoegd. Feaver produceert automatisch een Spin model, maar die tool is vrij recent en ook nog niet vrijgegeven. Lucent vindt het daarvoor, in elk geval voorlopig, nog te belangrijk.

PT Embedded Systems: U heeft ook een toolset geschreven voor het specificeren van requirements en het gedrag van gedistribueerde systemen. Dit UBET werkt op Message Sequence Charts, een familie van notaties die door de ITU (International Telecommunication Union) is gestandaardiseerd. Met UBET kun je onder meer raceconditities en timingfouten opsporen.
Gerard Holzmann: Door conversie van MSCs naar een Spin-model in Promela kun je de consistentie van requirements verifiëren, en UBET wordt dan ook in de praktijk gebruikt bij Lucent. Je ziet steeds meer dat Spin een black box functie krijgt en wordt toegepast in samenwerking met andere functionaliteit.

'De laatste vijf jaar is formele verificatie praktisch en efficiënt geworden'

PT Embedded Systems: U heeft een artikel gepubliceerd over de verificatie van SDL-modellen. Die Specification and Description Language, ook een ITU-standaard, heeft een vergelijkbaar doel als de Unified Modeling Language, die sterk in opkomst is. Er is ook al executeerbaar UML en de Object Management Group (OMG) heeft hiervoor een Precise Action Semantics ontwikkeld. Is of wordt het mogelijk om UML-modellen formeel te verifiëren?
Gerard Holzmann: SDL-specificaties kunnen inderdaad automatisch worden omgezet naar Promela. We hebben ook wel naar UML gekeken maar onze conclusie luidde toen, dat de UML standaard te vaag is. Maar de ontwikkelingen die u noemt gaan zeker de goede kant op, en dan zal dat zeker opgepakt worden door research-groepen. Wij zijn natuurlijk niet de enigen die zich hiermee bezig houden.

PT Embedded Systems: Formele verificatie van software is tot nu toe altijd toekomstmuziek geweest...
Gerard Holzmann: Toen ik in 1980 begon bij Bell Labs werd mij gevraagd om de correctheid te onderzoeken van de broncode van een telefoonswitch. Ik kwam toen tot de conclusie dat dat zeven dagen zou kosten op een machine van 700 MB. Die was er toen niet. Nu zou ditzelfde probleem 2 seconde vergen op een machine van 60 MB. Er is een revolutie geweest in zowel de kracht van de algoritmen als van de computersystemen. Voor de PathStar broncode zijn in 1999 zo'n 200 verschillende eigenschappen geverifieerd in 40 minuten. Nu, drie jaar later, kan dat in 20 minuten en nog twee jaar later zal dat 10 minuten zijn.
In de laatste vijf jaar is formele verificatie praktisch en efficiënt geworden, en dat wordt alleen maar beter. Zelfs als er niets verandert aan de algoritmen, dan zorgt de wet van Moore daar wel voor. Ons uiteindelijke doel is om automatische verificatie zo snel te maken dat het een functie kan worden van elke standaard compiler. Dan mag de verificatietijd niet meer zijn dan een fractie van een seconde en ik denk we over vijf tot tien jaar zo ver kunnen zijn. Dan kan de betrouwbaarheid van industriële software met enkele orden van grootte verbeteren.

PT Embedded Systems: U heeft behalve 'Design and Validation of Computer Protocols' nog twee boeken gepubliceerd, 'Beyond Photography – The Digital Darkroom' en 'The Early History of Data Networks'. U houdt zich nu niet meer bezig met beeldbewerking, maar in dat andere boek behandelt u onder meer de geschiedenis van de 'optische telegraaf'. Is dat een hobby van u?
Gerard Holzmann: Ik ben in de geschiedenis van datanetwerken geïnteresseerd geraakt door het eerste hoofdstuk van het boek over Spin, waarin ik een historische inleiding wilde geven. Hoe meer ik erover te weten kwam, hoe meer ik geïnteresseerd raakte. Communicatienetwerken bestonden al lang voor de invoering van elektriciteit en vooral in Frankrijk en Zweden is veel oorspronkelijk historisch materiaal te vinden. Dat kun je wel een hobby van me noemen... het houdt voor mij op in 1835, toen de elektrische telegraaf werd ingevoerd. Daarna is er eigenlijk niet zo veel meer veranderd... ■

Website:
http://cm.bell-labs.com/cm/cs/who/gerard/