PlusCal, een taal voor algorithmen

Leslie Lamport: 'De interessantste problemen zijn geïnspireerd door de praktijk'

De laatste jaren houdt Leslie Lamport van Microsoft Research zich vooral bezig met formele specificatie van (gedistribueerde) systemen. Zijn boek 'Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers' stamt uit 2003. 'TLA' staat voor 'temporal logic of actions' en aansluitend heeft Lamport een taal ontworpen voor de specificatie van algorithmen, PlusCal, die geïntegreerd is in de TLA toolbox. Bij het CWI in Amsterdam gaf hij een toelichting.

HANS VAN THIEL

Lamport is auteur van meer dan 150 wetenschappelijke artikelen en ontving in de loop van zijn carrière talloze onderscheidingen. Zijn bekendste werk is misschien wel, 'LaTeX: A Document Preparation System' uit 1986. Dit standaardwerk over het bekende opmaakprogramma LaTeX was echter min of meer een nevenactiviteit, naast zijn werk aan concurrente systemen. Zijn latere boek uit 2003, 'Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers' beschrijft hoe (gedistribueerde) systemen gespecificeerd kunnen worden als 'state machines'. Zo'n toestandsmachine is een systeem van variabelen en waarden waarvan het gedrag beschreven kan worden in TLA (Temporal Logic of Actions). Het formalisme dat Lamport gebruikt is geïmplementeerd in een model checker, TLC, die vrij beschikbaar is als open source programma.

```
--algorithm EuclidSedgewick
variables m ∈ 1..K, n ∈ 1..K, u = m, v = n
begin while u ≠ 0 do
        if u < v then u := v || v := u end if;
        u := u − v
      end while;
      assert IsGCD(v, m, n)
end algorithm
```

Een zeer eenvoudig voorbeeld van een PlusCal beschrijving, het algorithme van Euclides om de grootste gemene deler van twee natuurlijke getallen te vinden. Het algorithme wordt automatisch vertaald naar TLA (Temporal Logic of Actions) en resultaten worden gecontroleerd door de TLC model checker (door middel van de 'assert'). Met '||' wordt aangegeven dat variabele u de waarde van v krijgt en omgekeerd. In TLA wordt dit een 'action', in dit geval een conjunctie van waardeveranderingen.

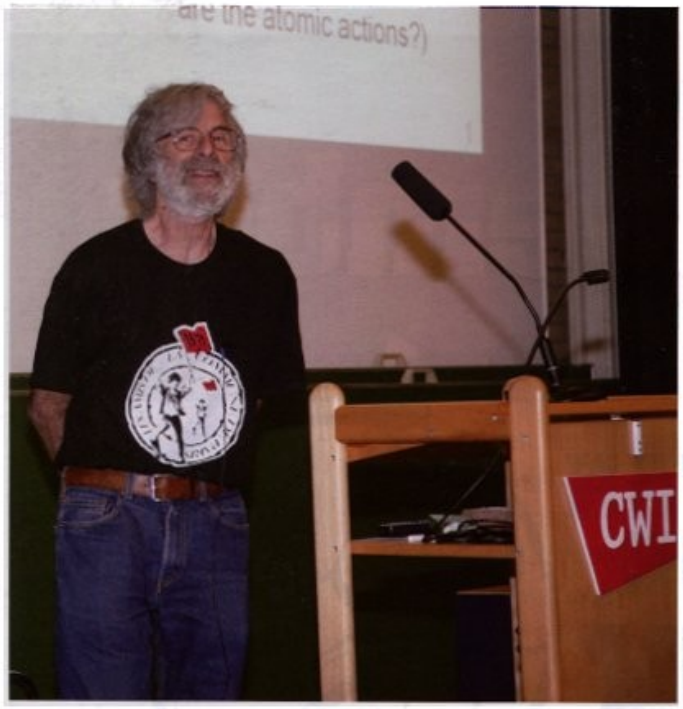
Aansluitend heeft Leslie Lamport, die tegenwoordig werkzaam is bij Microsoft Research, een specificatietaal voor algorithmen ontworpen die volledig geïntegreerd is met de TLA+ toolkit. Dit PlusCal is onder meer bedoeld om beschrijvingen in pseudocode te vervangen. Dergelijke ongeïmplementeerde 'talen' worden vaak gebruikt om een algorithme te schetsen, maar met PlusCal kunnen zulke beschrijvingen nu ook als TLA-modellen worden gechecked.
Leslie Lamport was op uitnodiging van het CWI (Centrum voor Wiskunde en Informatica) op het Science Park in Amsterdam om een toelichting te geven op PlusCal. Vervolgens beantwoordde hij per e-mail een aantal vragen van Elektronica + Embedded Systems.

Elektronica: In uw CWI lezing benadrukte u dat een algorithme geen implementatie is. In uw voorbeeld van het Quicksort algorithme: elke implementatie moet op de een of andere manier de scheidingswaarden kiezen. Het algorithme is onafhankelijk van elke keuze; dit verschil is fundamenteel. Een PlusCal vertaling naar TLA+ specificeert een algorithme.

Leslie Lamport: Het is beter om te zeggen dat een PlusCal algorithme vertaald wordt naar TLA+, en TLA+ heeft een welbepaalde betekenis. Dus de PlusCal code heeft een welbepaalde betekenis en specificeert een algorithme.
De modelchecker verifieert alle mogelijke executies, waarbij hij alle mogelijke toestanden genereert. Voor algorithmen met te veel mogelijke executies (misschien oneindig veel) instrueer je de modelchecker hoe die de verificaties moet beperken tot een eindige verzameling mogelijke executies. Dit blijkt in de praktijk effectiever te zijn om bugs te vinden dan het uitvoeren van een aantal door het toeval geselecteerde executies.

Dus de checker verifieert, onder meer natuurlijk, alle mogelijke keuzen van scheidingswaarden?

Lamport: De beschrijving van Quicksort die ik in PlusCal heb geschreven kan bij elke stap een willekeurige scheidingswaarde kiezen. Omdat de modelchecker alle mogelijke executies verifieert, verifieert hij alle mogelijke keuzen van scheidingswaarden in alle mogelijke executies (voor een zekere waarde van de lengte van het array).



Leslie Lamport: "Aangezien algorithmen en specificaties geen programma's zijn, is de vanzelfsprekende conclusie dat zij beschreven moeten worden door toepassing van wiskunde".

De correctheid van concurrent algorithmen kan sterk afhangen van het niveau van atomiciteit (granulariteit). Ik meen dat u in uw CWI lezing als voorbeeld noemde hoe een concurrent algorithme correct kan zijn als assignments atomair zijn, maar incorrect als diezelfde assignments worden gemodelleerd als samenstellingen van lees- en schrijfoperaties. Kunt u iets zeggen over hoe u een vertaling van PlusCal naar TLA+ granulariteit bewaart?

Lamport: In PlusCal geef je expliciet de granulariteit aan door het gebruik van labels. Alle code tussen twee labels wordt atomair geexecuteerd. (Zoals ik liet zien met het Quicksort voorbeeld kun je de labels ook weglaten en dan kiest de vertaler de grofst mogelijke granulariteit.)

Informatica via de achterdeur

Al vroeg in uw loopbaan introduceerde u een methode om gedetailleerd wiskundige bewijzen op te stellen, en vervolgens een systematiek om heel lange formules te schrijven. Uw latere werk aan specificaties in de taal van 'temporal logic actions' lijkt een voortzetting te zijn van die eerdere ideeën. In het bijzonder benadrukt u dat algorithmen beschreven horen te worden in mathematische termen en onafhankelijk van 'data typen'. Hoe ziet uzelf het verband tussen uw vroegere werk en uw huidige, in het bijzonder PlusCal?

Lamport: Ik ben opgeleid als wiskundige en ben de informatica binnengekomen door de 'achterdeur' van het programmeren. Toen ik mijn opleiding kreeg was er nog geen 'voordeur'. Na ongeveer 15 jaar net zoals andere 'computer scientists' te hebben gewerkt besefte ik twee dingen:

1) Als je geen programma schrijft hoef je ook geen programmeertaal te gebruiken.

2) Gewone wiskunde (zoals ik die in mijn opleiding heb geleerd) is veel eenvoudiger en oneindig veel expressiever dan enige programmeertaal.

Aangezien algorithmen en specificaties geen programma's zijn, is de vanzelfsprekende conclusie dat zij beschreven moeten worden door toepassing van wiskunde. TLA+ is mijn poging tot een praktische taal voor het gebruik van wiskunde om algorithmen en specificaties te (be)schrijven.

Keith Marzullo suggereerde dat het didactisch zinvol zou kunnen zijn om een kleine taal te ontwerpen voor algorithmen, die vertaald zou kunnen worden naar TLA+. Ik realiseerde me dat je de meeste expressiviteit van wiskunde in een programmeertaal zou kunnen krijgen door als expressies een willekeurige mathematische expressie toe te laten. Dat betekent in de praktijk elke expressie die in TLA+ geschreven kan worden. Het resultaat was PlusCal, een taal die slechts ietsje ingewikkelder is en ietsje minder krachtig dan TLA+, maar die er veel vriendelijker uitziet voor de meerderheid van programmeurs en 'computer scientists' die vrees hebben voor wiskunde.

Op uw website2 zegt u dat de meest interessante theoretische problemen voortkomen uit de implementatie van werkelijke systemen. Uzelf werkt voor Microsoft Research. Kunt u iets meer zeggen over hoe een industriële omgeving een inspiratie kan vormen voor academisch onderzoek?

Leslie Lamport: De meeste interessante problemen in 'computer science' zijn geïnspireerd door praktische problemen die de engineers, de programmeurs en systeemontwerpers, in hun werk tegenkomen. Dus, om interessante 'computer science' problemen te vinden, is het handig om engineers te helpen bij het oplossen van hun problemen. Het is altijd de moeite waard met goede engineers, omdat het je veel leert over wat zij doen, en die kennis is onbetaalbaar als je wilt dat je research ook nuttig is.

Zo nu en dan, als je goed doordenkt, kun je zien dat een bepaald probleem dat een engineer probeert op te lossen een speciaal geval is van een algemener informatica probleem waar niemand ooit aan had gedacht. ■

1) *In Quicksort wordt een sequentie gesorteerd door een scheidingswaarde te kiezen en vervolgens alle grotere elementen aan de ene kant te plaatsen en alle kleinere (eventueel inclusief gelijken) aan de andere kant. Dit proces wordt herhaald voor elk van de twee helften. Uiteindelijk blijft dan een aantal gesorteerde secties over die samen een gesorteerd geheel vormen.*

2) *Zie collected writings nr. 73: http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html*