CORRESPONDENTIES IN SEMANTIEK

J.-J.Ch. Meyer
Vrije Universiteit, Amsterdam /
Katholieke Universiteit, Nijmegen

0. Samenvatting

In deze korte bijdrage wil ik trachten het hedendaagse werk aan semantiek van programmeertalen in een historisch/filosofisch perspectief te plaatsen, zodanig dat het duidelijk wordt dat de 'queeste' naar compositionele (denotationele) semantieken van moderne talen als POOL in feite direct voortvloeit uit de ideeën van Tarski (en in zekere zin al Frege) met betrekking tot de logica.

1. Semantiek

Semantiek (< σημαινω (Gr.) = met een teken aanduiden, betekenen) is *betekenisleer*: het houdt zich bezig met het betekenis geven aan syntactische objecten zoals woorden, zinnen, (logische) formules, en de laatste 20 jaren ook *programma's*. Ofschoon (praktisch) informatici vaak de indruk wekken geen behoefte te hebben aan een aparte discipline die de semantiek van programmeertalen bestudeert ("Ik weet gewoon wat een programma doet, en anders laat ik het wel uitvoeren om te zien wat het effect is"), is het wel degelijk zeer natuurlijk om een semantische theorie te ontwikkelen als een formeel model, waarin een programma geschreven in een bepaalde programmeertaal kan worden geïnterpreteerd.

Eigenlijk is dit niet anders dan in de linguistiek: de betekenis van een zin zou ook als "vanzelfsprekend" kunnen worden beschouwd. Linguisten hebben zich echter de moeite getroost om semantische theorieën voor natuurlijke talen op te stellen, teneinde grip te krijgen op de betekenis van de overweldigende hoeveelheid zinnen in de diverse talen van de wereld. We noemen hier de Franse taalkundige M. Bréal die de term "diachronische semantiek" in de taalkunde invoerde voor het probleemgebied van de oorsprong en verandering van betekenissen der woorden (Bréal [1964], Kuypers [1977]) en Richard Montague die een (compositionele) semantiek heeft ontworpen voor natuurlijke taal, gebaseerd op intensionele logica (cf. D.R. Dowty [1979]).

In feite is de behoefte om zich met betekenissen van syntactische objecten bezig te houden al veel ouder.

Van oudsher houdt de filosofie zich bezig met het waarheidsbegrip. Grofweg zijn er twee soorten waarheidsdefinities: gebaseerd op de *coherentietheorie* en op de *correspondentietheorie* (zie Haack [1978], Martin [1987]). De eerste theorie zegt dat een verzameling zinnen waar is als deze op de juiste manier *samenhangt*. Dit is natuurlijk nogal vaag. Van een samenhangende verzameling zinnen wordt op zijn minst vereist dat deze intern consistent is, maar verder ook dat deze als het ware een totaal wereldbeeld geeft.

Correspondentietheorie, anderszijds, beweert dat een zin waar is als deze op de een of andere

manier *correspondeert* met (entiteiten in) de wereld. Volgens een middeleeuwse formulering is een zin waar als wat deze betekent (dat wil zeggen, hetgeen waarmee deze correspondeert in de wereld), inderdaad zo is, en ook meer recent in Wittgenstein's *Tractatus* zien we een dergelijke correspondentie (cf. Wittgenstein [1922], Grayling [1988]). Uiteraard is dit in zijn algemeenheid ook erg vaag. De verdienste van *Tarski* is, dat deze correspondenties in een wiskundig kader heeft gegoten, zodat men precies kan vaststellen wat de correspondentie inhoudt.

Dat syntactische expressies corresponderen met entiteiten in de wereld, krijgt bij Tarski de gedaante van een wiskundige functie $[\cdot]$ van deze syntactische expressies $\mathcal{E}$ naar een semantisch *domein* $\mathcal{D}$ *van denotaties*. Dit domein $\mathcal{D}$ van denotaties (betekenissen) is een geformaliseerd model van de wereld (voor zover relevant met betrekking tot de beschouwde expressies). Bijvoorbeeld, arithmetische expressies (inclusief cijfers) worden afgebeeld op een domein van (mathematische) getallen; propositie-logische expressies worden afgebeeld op het domein van de (doorgaans twee) waarheidswaarden.

2. Compositionaliteit

Een belangrijk punt bij Tarski is de wijze waarop de semantische functie $[\cdot]$ wordt geëvalueerd. Laat e bijvoorbeeld een propositioneel-logische expressie zijn, dus $[e] \in \mathcal{D} = \{t, f\}$. Wil er sprake zijn van een echte correspondentietheorie van waarheid, dan moet er volgens Tarski een zin e* in de metataal zijn die precies aangeeft wanneer e waar is, dat wil zeggen, wanneer $[e] = t$:

$$[e] = t \text{ desda } e^*.$$

e* wordt de *waarheidsconditie* van e genoemd, die conditie die nodig en voldoende is voor de waarheid van expressie e. De waarheidsconditie e* voor expressie e moet volgens Tarski ook voldoen aan het (ook al bij Frege voorkomende) *principe van compositionaliteit*: de denotatie van expressie e wordt volledig bepaald door de denotatie (cq. waarheid) van de subexpressies van e. Als bijvoorbeeld $e = e_1 \wedge e_2$, dan moet gelden dat

$$e^* \text{ d.e.s.d.a. } e_1^* \text{ en } e_2^*.$$

In het algemeen kunnen we dit principe van compositionaliteit als volgt formuleren:
Zij $e = e_1$ **op** $e_2 \in \mathcal{E}$ voor een of ander syntactische connectief **op**, dan moet gelden voor $[\cdot]: \mathcal{E} \rightarrow \mathcal{D}$:

$$[e] = op\ ([e_1], [e_2]),$$

waarbij *op* een operator (functie) is van type $\mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$ die correspondeert (met de werking / betekenis van) het connectief **op**. In het geval van $e = e_1 \wedge e_2$, leidt dit inderdaad tot

$e^* \Leftrightarrow [\![e]\!] = t \Leftrightarrow$
AND $([\![e_1]\!], [\![e_2]\!]) = t \Leftrightarrow$
$[\![e_1]\!] = t$ en $[\![e_2]\!] = t \Leftrightarrow e_1^*$ en e_2^*,

waarbij AND de boolean functie is, die correspondeert met het connectief $\wedge$, gedefiniëerd zoals gebruikelijk (bijvoorbeeld met een waarheidstabel).

3. Semantiek van programmeertalen

In de semantiek van programmeertalen, als onderzoeksterrein geïnitieerd in de eind 60-er jaren door pioniers zoals J.W. de Bakker, D. Scott en C. Strachey, zien we Tarski's principes weer terug in de context van programma's (zie De Bakker & Scott [1969][1], Scott & Strachey [1971], De Bakker [1980]).

Nu correspondeert onze verzameling $\mathcal{E}$ van expressies met een verzameling programma's in een bepaalde programmeertaal. (Aanvankelijk ALGOL-achtig, maar later werden ook andere talen beschouwd zoals logische programmeertalen (PROLOG), functionele talen (LISP, Miranda) en object-georiënteerde talen (POOL).) Het semantisch domein is in deze gevallen veel ingewikkelder geworden, met name voor talen met de mogelijkheid om recursieve en parallelle programma's te schrijven (zie bijvoorbeeld America & De Bakker [1988], De Bakker & Meyer [1988]). De grondgedachte is echter dezelfde: interpreteer expressies cq. programmastatements uit een klasse $\mathcal{E}$ in een (wiskundig) domein met behulp van een semantische functie $[\![\cdot]\!]: \mathcal{E} \rightarrow \mathcal{D}$, die (bij voorkeur) *compositioneel* is. $\mathcal{D}$ bestaat over het algemeen uit functies die de uitvoering van statements denoteren, bijvoorbeeld functies van inputwaarden van variabelen naar outputwaarden. Bijvoorbeeld, zij $e = e_1 ; e_2$, waarbij ; staat voor de sequentiële compositie (het achter elkaar uitvoeren), dan is

$$[\![e]\!] = [\![e_1]\!] \circ [\![e_2]\!],$$

waarbij de (mathematische) functie $\circ: \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$ ruwweg werkt als volgt: $d_1 \circ d_2$ is het resultaat dat men krijgt door eerst d_1 toe te passen en dan d_2 op het resultaat hiervan.

Deze wijze van semantiek geven aan programma's heet *denotationele semantiek*: Ze werkt met behulp van denoterende semantische functies en is aldus een directe *specialisatie* van Tarskiaanse semantiek gebaseerd op *correspondentietheorie* voor programmeertalen, zoals Montague-semantiek dit is voor natuurlijke talen.

Met een beetje goede wil kunnen we in de context van programmeertalen ook de *coherentietheorie* terugvinden. Behalve denotationele semantiek is er ook nog zoiets als *operationele semantiek* van programma's. Deze geeft weer, hoe een programma zich operationeel (stapje voor stapje) gedraagt. Zo vaag gesteld, is het verschil met denotationele semantiek niet altijd even duidelijk, hetgeen vaak aanleiding geeft tot oeverloze discussies of een bepaalde semantiek nu operationeel dan wel denotationeel is. Laat ik omwille van de duidelijkheid hier stellen dat een denotationele semantiek wordt gedefiniëerd aan de hand van een semantische functie $[\![\cdot]\!]: \mathcal{E} \rightarrow \mathcal{D}$, die compositioneel is. Een operationeel semantiek wordt tegenwoordig vaak gedefiniëerd met

behulp van zogenaamde transitiesystemen en is dus *a priori* niet-compositioneel. (Dit neemt niet weg, dat er bij een operationeel gedefiniëerde semantiek vaak een semantische functie is te definiëren die de interpretatie (of liever evaluatie) van een statement m.b.v. het gegeven transitiesysteem weergeeft. Deze functie f kan zelfs *a posteriori* compositioneel zijn in de zin, dat men kan *bewijzen* dat f (e) = g(f(e_1), f(e_2)) voor e = e_1 **op** e_2 en een functie g: $\mathcal{D} \rightarrow \mathcal{D}$. Echter, een operationele semantiek wordt zo NIET *gedefiniëerd*!)

Op zichzelf staand kan men een operationele semantiek opvatten als een *coherentietheorie*: ze moet een samenhangend geheel zijn, intern consistent en een totale afspiegeling van de betreffende (programma)wereld.

Echter, hiermee is het verhaal niet uit. Juist omdat een operationele semantiek als primitiever wordt ervaren dan een denotationele semantiek–ze is/levert immers een directe afspiegeling van de werking van programma's–worden denotationele semantieken vaak gerechtvaardigd door een relatie vast te stellen met een operationele semantiek. In eenvoudige gevallen (sequentiële programmeertalen) is deze rechtvaardigende relatie vaak (*extensionele*) *equivalentie*: het opleveren van hetzelfde resultaat (dat wil zeggen hoewel '*intensioneel*' verschillend opgezet, hebben denotationele en operationele semantiek dezelfde *extensie*; cf. De Bakker [1980]). In moeilijker gevallen heeft men vaak een minder dwingende eis: de denotationele semantiek moet *volledig abstract* zijn ten opzichte van de operationele (zie bijv. Rutten [1988]). Dit komt dan neer op de eis dat de denotationele semantiek zoveel ingewikkelder uitkomsten mag geven dan de operationele, dat de denotationele semantiek compositioneel is en precies die statements identificeert die operationeel in elke context hetzelfde opleveren. Zodoende ontstaat er weer een *correspondentie* tussen (de twee) semantieken.

Noot

1. In feite ontstond dit werk uit een correspondentie tussen De Bakker en Scott met als doel het vinden van een complete calculus voor equivalenties tussen while-statements (Bron: persoonlijke 'correspondentie' (cq. gesprekken) met De Bakker, 1989). Dit werk leverde tevens de kern van een wiskundige theorie van programmacorrectheid, later verder ontwikkeld door De Bakker en De Roever (De Roever [1974], De Bakker [1980]), die een verbindend element zou vormen tussen de semantiek van programmeertalen in strikte zin en logica's voor programmacorrectheid, die in die tijd door onder meer Hoare werden voorgesteld onder de enigszins logisch contradictoire term "*axiomatische semantiek*" (zie Hoare [1969]).

Literatuur

1. P. America & J.W. de Bakker, Designing Equivalent Semantic Models for Process Creation, TCS 60, 1988, pp. 109 - 176.

2. J.W. de Bakker, Mathematical Theory of Program Correctness, Prentice-Hall Int., Englewood Cliffs, New Jersey, 1980.

3. J.W. de Bakker & J.-J.Ch. Meyer, Metric Semantics for Concurrency, BIT 28, 1988, pp. 504 - 529.

4. J.W. de Bakker & D. Scott, A Theory of Programs, Unpublished Notes, IBM Seminar, Vienna, 1969.

5. M. Bréal, Semantics: Studies in the Science of Meaning, 1964.

6. D.R. Dowty, Word Meaning and Montague Grammar, Reidel, Dordrecht, 1979.

7. A.C. Grayling, Wittgenstein, Oxford University Press, Oxford, 1988.

8. S. Haack, Philosophy of Logics, Cambridge University Press, Cambridge, 1978.

9. C.A.R. Hoare, An Axiomatic Basis for Computer Programming, Communications of the ACM 12, 1969, pp. 576-580.

10. K. Kuypers e.a., Encyclopedie van de filosofie, Winkler Prins Bibliotheek, Elsevier, Amsterdam/ Brussel, 1977.

11. J.N. Martin, Elements of Formal Semantics, Academie Press, Orlando, 1987.

12. W.P. de Roever, Recursion and Parameter Mechanisms: An Axiomatic Approach, in: Proceedings 2nd ICALP (J. Loeckx, ed.) LNCS 14, Springer, 1974, pp. 34-65.

13. J.J.M.M. Rutten, Correctness and Full Abstraction of Metric Semantics of Concurrency, CWI Rapport CS - R8831, 1988.

14. D.S. Scott & C. Strachey, Toward a Mathematical Semantics for Computer Languages, in: Proceedings Symposium on Computers and Automata (J. Fox, ed.), Polytechnic Institute of Brooklyn Press, 1971, pp. 19-46.

15. L. Wittgenstein, Tractatus Logico-philosophicus, F.R.S., London, 1922.