OVER THEORETISCHE INFORMATICA

UITGESPROKEN BIJ DE AANVAARDING VAN HET AMBT VAN GEWOON HOOGLERAAR IN DE INFORMATICA AAN DE TECHNISCHE HOGESCHOOL TWENTE OP 2 OKTOBER 1975

DOOR

DR. IR. L. A. M. VERBEEK

Mijnheer de Rector Magnificus
Zeer gewaardeerde Toehoorders

Het is een goede gewoonte dat een nieuw benoemde hoogleraar zijn ambt in het openbaar aanvaardt. Dat gebeurt, enige tijd nadat hij zijn werkzaamheden feitelijk is begonnen, met het uitspreken van een rede over het vakgebied dat hij doceert en waarin hij wetenschappelijk werkzaam is. Het geeft hem de gelegenheid zijn visie en opinies in het openbaar te berde te brengen en het stelt zijn toehoorders in de gelegenheid die te aanhoren en te wegen.
Mijn waardering voor het houden van een inaugurele rede moet ik hier wel voorop stellen want in zekere zin heb ik een excuus nodig om nu, zes jaar nadat ik het aan de Technische Hogeschool Delft deed, hier weer een oratie te houden.

Toen ik in 1969 in Delft mijn inaugurele rede hield heb ik geprobeerd uit te leggen wat een bepaald onderwerp uit de theoretische studie van informatica behelst. Het ging daarbij over abstracte automaten en formele talen. Sindsdien hebben zich in hoog tempo diepgaande ontwikkelingen voorgedaan zowel in het toepassen van informatica als in de informatica zelf en in de grondslagen ervan. Daarom wil ik nu graag iets vertellen over theoretische informatica en proberen aan te geven wat de betekenis ervan is voor onderwijs en onderzoek in het vakgebied van de informatica.
Om dat te doen zal ik eerst in grote lijnen aangeven wat informatica is en wat theoretische informatica behelst. Daarna kom ik dan toe aan de zin en het belang van de theoretische informatica en wel vooral in het kader van het opleiden van informatica ingenieurs.
In mijn betoog zal ik geen volledig of evenwichtig beeld schetsen van de betekenis van theoretische informatica. Dat kan nog lang niet omdat het betreffende gebied volop in ontwikkeling is. Wel wil ik aangeven welke rol, naar mijn mening, de theoretische informatica nu dient te spelen bij het onderwijs in de informatica aan deze technische hogeschool.

Het woord informatica werd dertien jaar geleden door de Fransman Philippe Dreyfus bedacht als samentrekking van “information” en “automatique”. Sindsdien heeft het ingang gevonden als naam voor een vakgebied dat nu ongeveer dertig jaar oud is en zich steeds duidelijker aftekent. Oorspronkelijk betrof het ongeveer alles wat met computers te maken had, het ontwerpen en de bouw ervan, de gebruiksmethoden en ook de toepassingen op diverse gebieden. Geleidelijk aan heeft informatica zich ontwikkeld tot een apart vakgebied, los van zijn toepassingen bij technisch-wetenschappelijke of administratief-bestuurlijke werkzaamheden. Over die ontwikkeling wordt ook gesproken in de diesrede 1974 die Duijvestijn hier aan de T.H. Twente gehouden heeft. Het vakgebied informatica gaat over het met behulp van computers verwerken van informatie, of liever: van gegevens. (Het woord “gegevens” gebruik ik liever dan “informatie” omdat dit laatste zoveel meer bijbetekenissen heeft.) De gegevens die verwerkt worden zijn meestal afkomstig van zinnige menselijke bezigheden. Het zijn bijvoorbeeld administratieve gegevens, zoals namen, adressen en premiegegevens van klanten van een verzekeringsmaatschappij, of technische gegevens, zoals afmetingen en belastingen van een brug over een kanaal. Het verwerken van gegevens in de informatica wordt gedaan door digitale computers. Deze noem ik kortweg computers omdat ik analoge en hybride computers buiten beschouwing laat. Overigens zal ik computers als gegeven apparaten beschouwen en niet uitweiden over de organisatie en structuur ervan.

Ik ga nu enigszins in op enkele belangrijke aspecten van gegevensverwerking in de informatica. De te verwerken gegevens worden vaak als invoer aan de computer verschaft, bijvoorbeeld op ponskaarten of door intoetsen op een soort typemachine. Het komt ook vaak voor dat de te verwerken gegevens opgeslagen zijn in een deel van het geheugen van de computer in de vorm van een gegevensbestand. Zo’n bestand is een collectie gelijksoortige gegevens die volgens een bepaald stramien in het computergeheugen zijn opgeborgen. Elk gegeven uit zo’n bestand is ook weer een uit onderdelen samengesteld en gestructureerd geheel. We kunnen hierbij bijvoorbeeld nog even denken aan het

klantenbestand van de verzekeringsmaatschappij. Elk gegeven hierin betreft een klant en bestaat uit zijn naam, adres en andere personalia en verder ook uit nummers, premies en vervaldata van zijn polissen. Het resultaat van de gegevensverwerking kan door de computer als uitvoer worden afgegeven, bijvoorbeeld in de vorm van een ponsband, of als tekst afgeleverd door een regeldrukker of een typemachine. Vaak bestaat het resultaat ook uit veranderingen die in een gegevensbestand in het computergeheugen zijn aangebracht, zodat het bestand weer is bijgewerkt.
Zowel bij de invoer of uitvoer van gegevens als bij de interne opslag en de verwerking ervan is de structuur van de gegevens, dat wil zeggen de manier waarop de onderdelen ervan samenhangen, een belangrijke zaak. We zullen dit onderwerp straks nog tegenkomen onder de naam gegevensstructuur of informatiestructuur.

Maar nu eerst iets over het proces dat in de computer plaatsvindt om de gegevens te verwerken. Dat proces is vastgelegd door een computerprogramma, kortweg programma genoemd, waarin precies staat wat de procedure of algoritme is volgens welke het proces van gegevensverwerking verloopt. De algoritme geeft de collectie aan van de stappen of acties die de computer moet uitvoeren bij de verwerking en bevat ook de condities waaronder en de volgorde waarin dat moet gebeuren. Volgens die algoritme wordt de computer gestuurd in zijn acties. De algoritme is dus de beschrijving van het verwerkingsproces, hij geeft precies het verloop van dat proces aan. Daarom vormt het onderwerp algoritmen een tweede essentieel onderdeel van informatica en ik zal er straks op terugkomen.

Het programma waarin de algoritme is vastgelegd wordt geschreven in een programmeertaal, zoals bijvoorbeeld Algol of Fortran. De programmeur moet de algoritme opschrijven in de programmeertaal om hem een vorm te geven die voor de computer geschikt is. Maar de programmeur is ook maar een mens en daarom moet de programmeertaal voor hem bruikbaar zijn om er algoritmen in uit te drukken, de taal moet met een zeker gemak gehanteerd kunnen worden. Een programmeertaal moet dus zo zijn dat zowel de computer als ook de programmeur ermee kan werken en natuurlijk elk op een eigen en totaal andere wijze. Op deze vereisten bij programmeertalen kom ik dadelijk terug. De elementaire acties die een computer kan uitvoeren zijn erg simpel en daarom is er soms een hele reeks elementaire opdrachten nodig om een eenvoudige actie in een verwerkingsproces op te schrijven.
Om dit te verduidelijken neem ik als voorbeeld het delen van het getal 47 door 11 met als resultaat 4 en rest 3. Veronderstel dat delen geen elementaire actie is van de computer maar wel, onder andere, aftrekken. Het delingsproces kan dan gerealiseerd worden door herhaald aftrekken. Immers de computer kan net zolang 11 van 47 aftrekken tot er minder dan 11 overblijft en dan als resultaat geven het aantal keren dat het aftrekken lukte, dus 4, en wat het restant is, dus 3. In plaats van de opdracht: deel 47 door 11 moet de programmeur daarvoor een hele reeks opdrachten schrijven, bijvoorbeeld trek 11 van 47 af en als het resultaat groter dan 11 is trek daar dan weer 11 van af, blijf dit herhalen tot het resultaat kleiner is dan 11. Inderdaad, veel elementaire opdrachten om één minder elementaire actie te realiseren. Om nu het opschrijven van veel elementaire opdrachten en hun juiste volgorde te vermijden schrijft men liever een globale opdracht en laat die door de computer zelf eerst omzetten in de juiste elementaire opdrachten. Zodoende schrijft men een computerprogramma in wat men wel noemt een hogere programmeertaal, zoals Algol of Fortran, en laat de computer dat programma omzetten in de erdoor aangeduide rij elementaire computeropdrachten. De computer vertaalt het programma dus van de hogere programmeertaal in de machinetaal. Daarom kan de hogere programmeertaal meer gericht zijn op het gemak van hanteren door de programmeur; immers de machinetaal is geschikt voor interpretatie door de computer. Hiermee is aan de zoeven aangeduide eisen bij programmeertalen in principe voldaan.
Het is hiermee wel duidelijk geworden dat programmeertalen een derde belangrijk onderwerp in de informatica vormen, we zullen het ook straks weer tegenkomen.

Het omzetten of vertalen van een programma geschreven in een hogere programmeertaal in de machinetaal is op zichzelf ook een geval van gegevensverwerking. Daarbij vormt het programma het invoergegeven, het verwerken bestaat uit het vertalen, het vertaalde programma is het resultaat ervan.
De globale schets van het verwerken van gegevens door een computer volgens een algoritme zoals we dat tot nu toe hebben gezien is als volgt:
De algoritme voor het verwerken van gegevens wordt gegeven in een programma dat geschreven is in een programmeertaal. Het programma wordt aan de computer als invoer gegeven en door het al in de computer opgeslagen vertaalprogramma omgezet in machinetaal. Dit vertaalde programma regelt het verwerken van de gegevens door de computer.
Bij deze gang van zaken is het duidelijk dat de in- en uitvoerorganen, het geheugen en andere delen van de computer allemaal op hun beurt geactiveerd en stilgelegd moeten worden. Ook de gegevensstroom en de diverse verwerkingsprocessen moeten worden gecoördineerd. Dit besturings- en coördinatiewerk wordt door de computer zelf gedaan volgens een collectie algoritmen die vervat zijn in de programma's die samen het zgn. bedrijfssysteem vormen. Vertaalprogramma's en bedrijfssysteem samen noemt men systeemprogrammatuur.

Tot nu toe heb ik dus drie belangrijke zaken in de informatica aangestipt, te weten gegevensstructuren, algoritmen en programmeertalen. Er zijn natuurlijk ook andere belangrijke kernonderwerpen aan te wijzen, zoals de computer zelf en de bijbehorende systeemprogrammatuur. Omdat ik deze echter niet strikt nodig heb bij het praten over theoretische informatica zal ik ze verder buiten beschouwing laten.
Zoals de naam al zegt, omvat de theoretische informatica alle theoretische aspecten van het verwerken van gegevens door computers.
Theoretisch, dus er wordt afgezien van praktische uitvoeringsproblemen maar juist gemikt op fundamentele problemen en op inzicht in de samenhang van methoden en begrippen.
Beoefenen van het vak theoretische informatica houdt voornamelijk in het analyseren, het expliciet maken en beschrijven van fundamentele structuren, begrippen, activiteiten en methoden die in de informatica een rol spelen of kunnen gaan spelen. Daarbij wordt geabstraheerd van het concrete verwerken van gegevens. Bijvoorbeeld de technische uitvoering van de computer is niet van belang evenmin als de fysische karakteristieken van de manier waarop gegevens en programma's zowel binnen als buiten de computer worden vastgelegd. Dergelijke zaken zijn weliswaar erg belangrijk voor het praktisch werken met computers maar ze zijn onbelangrijk voor het inzicht in fundamentele begrippen, in mogelijke verwerkingsmethoden en in de samenhang van begrippen en methoden.
De meeste onderwerpen die tot de theoretische informatica worden gerekend, zijn gemakkelijk in te passen in de drie al eerder genoemde hoofdzaken van de informatica: gegevensstructuur, algoritme en programmeertaal. Ik zal ze achtereenvolgens wat nader bespreken om zo een indruk te geven van wat er omgaat in theoretische informatica, zonder daarbij het verband met de informatica zelf uit het oog te verliezen.

Wat betreft gegevensstructuren omvat theoretische informatica de wiskundige beschrijving van structuren in gegevens dat wil zeggen van verbanden tussen de elementen waaruit de gegevens zijn samengesteld. Het gaat dus over zaken zoals lijsten met als structuur hun lineaire ordening of zoals tabellen met twee of meer indices voor het bepalen van de plaats van elementen. Denk ook aan mathematische bomen, met hun hiërarchische en hun vertakkingsstructuur. Wiskundige manieren voor het beschrijven van dergelijke structuren en vooral ook voor het manipuleren ermee hebben hun weerslag in de representatie van gegevens in het computergeheugen en in de algoritmen volgens welke ze verwerkt worden. Het inzicht dat ontstaat door abstracte, in wiskundige vorm uitgedrukte, beschouwingen en door theorievorming heeft gevolgen voor de opbouw van bestanden van gestructureerde gegevens. Het verband van die opbouw met sorteer- en zoekalgoritmen op deze bestanden en met mogelijkheden voor het praktisch gebruik ervan vormt een belangrijk onderwerp in theoretische informatica. Vooral de laatste jaren zijn verschillende manieren in studie en in ontwikkeling voor het beschrijven van samengestelde, gestructureerde gegevens met behulp van elementaire wiskundige begrippen zoals verzamelingen en relaties. Deze theoretische activiteiten worden gestimuleerd door de vragen en problemen die voortkomen uit praktische gegevensverwerking. Het gaat daarbij bijvoorbeeld om de gegevens van de postgiro, of de gegevens van de inwoners in een gemeentelijke administratie, of de gegevens van de verkooporganisatie van een industriële onderneming.
Het gaat steeds om enorm grote hoeveelheden gegevens met een min of meer blijvend karakter die in een gegevensbank moeten worden opgeslagen.
Laten we als voorbeeld even denken aan de persoonsgegevens van alle inwoners van een gemeente. Daaraan moeten, bijvoorbeeld bij geboorte, nieuwe gegevens worden toegevoegd en ook hun verband met al bestaande gegevens. Verder moeten veranderingen worden aangebracht, denk maar aan huwelijk of verhuizen. Uiteraard moet er informatie uit worden opgezocht, bijvoorbeeld voor het maken van een lijst van alle stemgerechtigden. Ook verificatie van bestaande gegevens is vaak nodig, bijvoorbeeld wanneer een inwoner een paspoort aanvraagt.

De informatie in een gegevensbank is meestal gestructureerd o.a. door verwijzing van deelgegevens naast elkaar. Dergelijke verwijzingen zijn ook nodig voor het opzoeken van gegevens. Hierdoor en door de grote hoeveelheid informatie zijn gegevensbanken en het gebruik ervan voor het opslaan, het opzoeken, het verifiëren of het wijzigen van gegevens enorm ingewikkeld. Bovendien is het gebruik van de inhoud van gegevensbanken vaak van grote invloed op de belangen en rechten van individuen. Vandaar dat er dringend behoefte is aan theorievorming en fundamenteel inzicht in gegevensbanken om daardoor te verduidelijken wat de mogelijkheden van het gebruik ervan en de begrenzing van die mogelijkheden zijn.
Het tweede concentratiepunt van theoretische informatica wordt gevormd door algoritmen. Inherent aan het werken van computers is het op precies omschreven manier verwerken van gegevens. Vandaar dat algoritmen een kernbegrip van de informatica zijn. Er is dan ook steeds theoretisch onderzoek verricht aan het ontwerpen, het beschrijven en het analyseren van algoritmen. In de eerste tijd vooral van algoritmen voor numerieke berekeningen die vaak van mathematisch-fysische aard zijn, zoals berekeningen aan vliegtuigeigenschappen of aan civieltechnische constructies. Later kwamen ook algoritmen voor niet numerieke verwerking van gegevens op de voorgrond, vooral in verband met administratieve toepassingen van de computer.

Bij de studie naar de grondslagen van de wiskunde in de mathematische logica werd veertig jaar geleden al onderzoek aan effectieve procedures of algoritmen verricht. Dat was gericht op het formuleren van wat eigenlijk berekenbaar is en wat niet, en welke ja-nee problemen door een formeel logisch proces beslist kunnen worden en welke niet.
Bij dat onderzoek van berekenbaarheid en beslisbaarheid is veel theorie over deze onderwerpen opgebouwd en ook inzicht verkregen in de grenzen van de mogelijkheden van algoritmen om verwerkingsprocessen te bepalen. Al doende ontwikkelden de logici daarbij ook wiskundige modellen voor het verwerken van gegevens zoals Turingmachines en λ-calculus.
Zowel begrippen als methoden van aanpak als manieren van beschrijven zijn uit het grondslagenonderzoek overgegaan in de theoretische informatica bij het bestuderen van algoritmen die door een computer kunnen worden gerealiseerd. Verschillende wiskundige modellen van computers, of functionele delen ervan, zijn zo ontwikkeld en vormen de kern van de automatentheorie die een van de erkende hoofdstukken van theoretische informatica is.

Omdat algoritmen die invoergegevens omzetten in uitvoergegevens, betrokken zijn op praktische problemen werd en wordt natuurlijk ook onderzoek verricht naar hun efficiëntie. Voor de hand ligt de wens naar algoritmen die goedkoop werken, gemeten in termen van computertijd en benodigde geheugenruimte. In samenhang hiermee is er reeds en wordt er nog veel onderzoek gedaan aan complexiteit van algoritmen en is een hele complexitieitstheorie ontstaan.
Inzicht in verbanden tussen processen wordt verkregen door de studie van algoritmen die de coördinatie verzorgen van de activiteiten van verschillende delen van een computersysteem, dat is een computer samen met zijn systeemprogrammatuur.
In een computerinstallatie kunnen verscheidene invoerapparaten en uitvoerapparaten vaak gelijktijdig naast elkaar werken zonder elkaar te storen, elk met een eigen deel van het geheugen.
Dit parallel werken bevordert de efficiëntie van het gehele computersysteem en maakt het mogelijk dat verscheidene gebruikers elk voor zich over het systeem beschikken. Bij deze zogenaamde multi-programmering is natuurlijk coördinatie tussen de verschillende processen nodig en moeten wederzijdse verstoringen worden vermeden. Hiertoe zijn ingewikkelde besturingsalgoritmen nodig die in het bedrijfssysteem zijn ingebed. De studie en ontwikkeling hiervan is volop in gang en er zitten fascinerende theoretische problemen aan vast.

Een derde groep onderwerpen uit de theoretische informatica heeft als kernpunt programmeertalen. Deze kunstmatige, door de mens bewust ontworpen talen dienen als communicatiemiddel waarin de programmeur algoritmen uitdrukt die door een computersysteem kunnen worden geëxecuteerd. Daarnaast dienen vooral de hogere programmeertalen ook voor communicatie tussen mensen.
Net als een volzin in een natuurlijke taal heeft een programma in een programmeertaal een vorm, die bepaald wordt door de syntaxis, en een betekenis, die bepaald wordt door semantiek van de programmeertaal.
Hiermee zijn meteen twee belangrijke onderwerpen uit de theoretische informatica aangegeven, nl. de theorie van de syntaxis en die van de semantiek van programmeertalen. Deze twee aspecten, namelijk vorm en inhoud van programma's en hun verband ga ik nu wat nader bespreken.
Theoretisch onderzoek van syntaxis van programmeertalen is al twintig jaar aan de gang en er is een hele theorie ontstaan van formele talen, dat wil zeggen talen die alleen vormen, syntaxis, maar geen betekenis, semantiek, hebben. Deze theorie is trouwens niet alleen van belang voor informatica maar is uitgegroeid tot zinvolle beschrijvingswijze van velerlei structuren en processen. Als voorbeeld noem ik de beschrijving van biologische groei. Een toepassing van de theorie van formele talen in de informatica vinden we bij het ontleden van een programma in zijn delen en hun samenhang. Deze syntactische analyse wordt in elk vertaalprogramma geïncorporeerd omdat de betekenis van een programma is vastgelegd door die van zijn delen en hun samenstelling. Voor het interpreteren van de algoritme die door het programma is beschreven, moet de computer het programma dus eerst uiteenrafelen of ontleden.
Overigens is het onderzoek van syntactische structuur veelal gedaan in verband met en soms geïnspireerd door soortgelijk onderzoek aan de syntaxis van natuurlijke, dus menselijke talen. Vooral het initiatief tot formaliseringen van syntaxis van talen door de Amerikaanse filosoof-Linguïst Noam Chomsky is hierbij stimulerend geweest.

Ook nu nog wordt er volop theorie bedreven aan syntactische beschrijvingen van programmeertalen, structuren en processen. Daarbij worden af en toe resultaten bereikt die praktisch inzicht geven en hun nut afwerpen, bijvoorbeeld voor het maken van vertaalprogramma's.
De semantiek van programmeertalen is pas veel korter, zeg sinds tien jaar, echt onderwerp van theoretisch onderzoek in de informatica. Een moeilijkheid is vooral om een goede vorm te vinden voor het uitdrukken van de betekenis van delen van een computerprogramma, die samengesteld moeten worden in overeenstemming met de syntactische structuur om de betekenis van het hele programma te verkrijgen.
Er zijn twee, soms een beetje concurrerende, opvattingen over het formaliseren van semantiek. Volgens de operationele opvatting is de betekenis van een programmadeel dat wat erdoor aan operationele stappen of

acties van de computer wordt veroorzaakt. Deze opvatting is uiteraard sterk gericht op de samenhang van programma en computer. De andere opvatting noemt men wel de mathematische semantiek. Daarbij wordt de betekenis van een programmadeel uitgedrukt in wiskundige termen zoals functies en relaties. Hierbij wordt benadrukt dat de betekenis een aspect is van het programma zelf los van de computer die het programma interpreteert. De specifieke merites van beide methoden zijn volop in onderzoek en er wordt veel werk verricht aan het uitbreiden en uitdiepen van de theorie van semantiek van programmeertalen.
Een interessante kant van semantiek betreft de correctheid van programma's. Het is natuurlijk erg belangrijk, ook en vooral uit praktisch oogpunt, of een programma correct is, dat wil zeggen of de betekenis van het programma overeenstemt met de bedoeling ervan. Om dit bij grote en ingewikkelde programma's te kunnen verifiëren moeten we beschikken over een wijze van uitdrukken van de bedoeling van het programma die los staat van het programma zelf. Een of andere wiskundige vorm is daarvoor het aangewezen middel. Als nu de bedoeling van een programma en de betekenis ervan, beide in wiskundige vorm uitgedrukt, hetzelfde zijn weten we dat het programma correct is en is dat niet het geval dan is wellicht aan te geven wat er mis is.
In de tot nu toe gegeven opsomming van onderwerpen uit de theoretische informatica rondom de drie kernpunten, gegevensstructuur, algoritme en programmeertaal, is het onderlinge verband niet erg benadrukt.
Uiteraard is dat wel steeds aanwezig en het treedt sterk op de voorgrond bij een aantal onderwerpen die nog niet aan de orde kwamen, namelijk de onderwerpen die men rekent tot het gebied dat met de naam machinale of kunstmatige intelligentie wordt aangeduid. Veel onderzoek en ontwikkelingswerk hierin betreft deelaspecten van het menselijk denken of handelen en bestaat vaak uit het simuleren van uitwendig merkbare effecten ervan met behulp van computerprogramma's. Interessant daarbij en vaak ook hoofddoel van het onderzoek zijn de methoden van structurering van gegevens en de representatie ervan zodanig dat met adequaat in elkaar grijpende algoritmen de gewenste activiteiten door het computersysteem verricht worden. Het ontwerpen van geschikte kenmerken die programmeertalen moeten hebben voor het kunnen uitdrukken van die gegevensstructuren en algoritmen en van hun interactie vormt een belangrijk deel van dergelijk onderzoek.

Hiermee heb ik, naar ik meen, wel voldoende onderwerpen uit de theoretische informatica opgenoemd om u een indruk te geven van wat er in dat gebied omgaat. Meer is er over te vinden in de inaugurele oratie die de Bakker hield aan de Vrije Universiteit in Amsterdam in 1974.
Bij het bespreken van de betekenis van theoretische informatica wil ik er graag twee kanten aan onderscheiden. Eerst wil ik die betekenis voor de informatica zelf bezien. Daarna zal ik kijken naar de betekenis van theoretische informatica voor de opleiding in informatica en daarmee ook voor een deel van het onderwijs en onderzoek aan deze hogeschool.

Zoals we al zagen houdt men zich in informatica bezig met het verwerken van informatie, of gegevens, met behulp van computers. Daarbij bleek dat de begrippen gegevensstructuur, algoritme en programmeertaal kernonderwerpen zijn waarmee de informaticus in zijn beroepsuitoefening te maken heeft. Om over zijn activiteiten en over de resultaten ervan te kunnen denken, praten, schrijven en lezen is het natuurlijk nodig dat hij een daarvoor passende manier van uitdrukken en beschrijven heeft. Hij moet beschikken over een collectie begrippen en ook namen voor die begrippen en de verbanden ertussen, die geschikt is om met anderen van gedachten te wisselen over gegevensverwerking en alles wat daarbij hoort. De begrippen die bij die menselijke communicatie worden gebruikt moeten onafhankelijk zijn van bijkomstigheden en van praktische of technische details. Bovendien moeten ze precies en eenduidig zijn in hun betekenis. Anders gezegd: die begrippen behoren abstract te zijn en moeten in wiskundige vorm gegoten zijn. Hiermee is een belangrijke taak en een deel van de betekenis van theoretische informatica aangegeven, namelijk het leveren in wiskundige vorm van passende en bruikbare abstracte begrippen voor het beschrijven van zaken die in de informatica van fundamenteel belang zijn. Maar als communicatie eenmaal goed mogelijk is omdat juiste begrippen en zinvolle verbanden geformuleerd kunnen worden, gaan we verder.

Bij het zoeken naar en vorm geven aan belangrijke begrippen en verbanden ontstaat meestal beter, scherper inzicht in de betrokken onderwerpen. Dat is dus inzicht in wat er essentieel en wat er bijkomstig aan is, hoe de samenhang van de essentiële zaken is, wat de theoretische grenzen zijn van mogelijkheden bij het verwerken van gegevens. Dat inzicht richt en stimuleert weer het zoeken naar manieren om zo'n theoretische grens ook praktisch te benaderen. Kortom, de betekenis van theoretische informatica voor het beoefenen van de informatica zelf is gelegen in de begripsvorming, het leveren van beschrijvingsmethoden, het verdiepen van inzicht en het aangeven van mogelijkheden voor het oplossen van problemen bij het verwerken van gegevens. Ik wil dit adstrueren door te herinneren aan drie voorbeelden die straks terloops al even ter spraken kwamen. Dat was het gebruik maken van complexiteitstheorie bij de evaluatie van algoritmen. Verder het gebruik van theorie van formele talen bij het ontleden van programmeertalen, d.w.z. bij de syntactische analyse van programma's. Ten derde het gebruik van theorie van semantiek van programmeertalen bij het bewijzen van de correctheid van programma's. Bij deze drie voorbeelden hebben begrippen die in wisselwerking tussen theorie en praktijk werden ontwikkeld geleid tot een verdere

theorievorming. De zo ontstane abstracte begrippen en hun wiskundige beschrijvingswijze worden praktisch gehanteerd en het verkregen inzicht geeft aanleiding tot betere methoden bij het ontwikkelen van programmatuur.

Hierbij wil ik nog wel een paar opmerkingen maken om misvattingen te voorkomen.
Ook al is er dus geen sprake van éénrichtingsverkeer tussen theorie en praktijk, toch is er dikwijls behoefte aan meer of duidelijker stimulans of hulp van theorie aan praktijk en vice versa.
Een tweede opmerking over de rol van de theorie in de informatica houdt verband met de relatieve jeugd van het hele vakgebied. Immers informatica is pas dertig jaar oud en als technisch-wetenschappelijk vakgebied is het maar twintig jaar oud, schat ik. Samenhangend met die jeugd is een sterke groei en verdieping van het vakgebied te constateren die extra gestimuleerd wordt door de grote maatschappelijke relevantie ervan met duidelijke sociale zowel als economische kanten. Hierdoor zijn ook uitspraken over wat er belangrijk in is, aan verandering onderhevig. Ik ben er van overtuigd dat over tien jaar andere onderwerpen op de voorgrond treden dan nu. Ook de relatie theorie-praktijk zal dan zeker geëvolueerd zijn en er anders, wellicht nog boeiender, uitzien dan nu.
Ik kom nu toe aan de tweede kant van de betekenis van theoretische informatica die ik hier wilde bespreken. Het betreft de betekenis die zij heeft in de opleiding van informatica ingenieurs. Eerst moet ik dan opmerken dat er in Nederland geen volledige opleiding in de informatica bestaat op academisch niveau. Dat wil zeggen dat er aan de universiteiten en hogescholen geen zelfstandige studierichting informatica bestaat en dus ook geen diploma van ingenieur of doctorandus in de informatica wordt verleend. Er zijn wel plannen om een aparte studierichting voor informatica wettelijk mogelijk te maken en op landelijk gecoördineerde wijze op te zetten.

Aan de meeste universitaire instellingen wordt er natuurlijk wel, en vaak ook al vele jaren, onderzoek verricht en onderwijs gegeven in informatica. Op alle drie Nederlandse technische hogescholen wordt er duidelijk en expliciet aandacht aan besteed.
Dat is vooral geconcentreerd in de afdelingen waar toegepaste wiskunde en waar elektrotechniek zijn ingedeeld. Globaal gezien kunnen studenten, na een verplichte inleiding erin, naar keuze ook nog verder onderwijs volgen, studie maken en oefening opdoen in informatica en het ook als afstudeerrichting kiezen.
Bij het bespreken van fundamentele aspecten in de informatica heb ik straks al aangegeven dat de inhoud en de betekenis ervan de laatste jaren erg gegroeid zijn. Daarom lijkt mij nuttig aan te geven wat, naar mijn mening, de betekenis is van theoretische informatica in de opleiding van informatica ingenieurs. Met die term informatica ingenieur bedoel ik hier iemand die via een aparte studierichting voor informatica is opgeleid of een toegepast wiskundig of elektrotechnisch ingenieur met informatica als afstudeerrichting. Het onderscheid lijkt me namelijk op dit inhoudelijke punt niet essentieel maar meer van organisatorische aard en ik zal er nu niet verder op in gaan.
Een informatica ingenieur beoefent een technisch-wetenschappelijk vak. Een probleem dat zich in zijn vakgebied, de informatica, voordoet of dat gesteld wordt, moet hij kunnen analyseren, in zinvolle deelproblemen uiteenleggen om zodoende tot een ontwerp voor een oplossing en vandaar tot een oplossing te komen.

Het essentiële zit in het analyseren van het probleem en het ontwerpen en uitwerken van een oplossing. Het eerste is kenmerk van wetenschappelijk en het tweede van technisch bezig zijn.
De technisch-wetenschappelijke denk- en handelwijze in het beoefenen van het vakgebied is kenmerkend voor de informatica ingenieur ter onderscheiding van de universitair opgeleide wetenschappelijke informaticus en de man met een ambachtelijke beroepsopleiding. Bij deze plaatsbepaling van de informatica ingenieur moet ik wel benadrukken dat ik hier over het type spreek en niet over afzonderlijke individuen. Het karakteristieke van een individu in zijn werk wordt gelukkig niet alleen bepaald door zijn opleiding maar is ook erg afhankelijk van zijn aard, aanleg en inspanning.
Na deze opmerking van meer algemene aard kom ik nu terug op de betekenis van theoretische informatica in de opleiding tot informatica ingenieur. Uit de aard van zijn technisch-wetenschappelijk beroep moet de informatica ingenieur kunnen denken, praten, schrijven en lezen over zijn werk in duidelijke termen en met gebruik van algemeen aanvaarde begrippen. Daartoe moet hij kennis hebben van de terminologie en inzicht in de begrippen die in de informatica een rol spelen.
Zoals ik al eerder betoogde is het de theoretische informatica die fundamentele begrippen in de informatica opspoort en in passende vorm beschrijft. Om dit vaak formeel-wiskundige taalgebruik en het bijbehorende inzicht te verwerven, dient een informatica ingenieur in zijn opleiding theoretische informatica te bestuderen. Natuurlijk geldt deze redenering evenzeer voor de studie van de theorie van het vak bij de opleiding in elk ander technisch-wetenschappelijk gebied. Maar het extra benadrukken ervan voor de informatica is mijns inziens om twee redenen op zijn plaats.
De eerste is dat computersystemen die gebruikt worden voor het verwerken van gegevens, erg ingewikkeld zijn. Vaak wordt bijvoorbeeld systeemprogrammatuur gebruikt waarvoor enkele manjaren nodig zijn geweest om het te maken. Een intuïtief inzicht in de werkelijke gang van zaken bij het ontwerpen of bij het gebruiken van dergelijke middelen is nauwelijks mogelijk en ook moeilijk onderwijsbaar. Daarom is een

abstraheren nodig van details van gegevensverwerking om met globale begrippen op hoger niveau te werken en dit moet nog weer op verscheidene niveaus herhaald worden. Het is daartoe nodig dat een informatica ingenieur op verschillende niveaus van abstractie kan denken en werken. Daarbij zijn theoretische begrippen en methoden onontbeerlijk.

De tweede reden voor het benadrukken van de betekenis van de theoretische informatica in de opleiding tot informatica ingenieur is mijns inziens de volgende. Met de snelle uitbreiding en ontwikkeling van de informatica gaat een snelle veroudering van concrete vakkennis gepaard. Een informatica ingenieur zal dus, o.a. via publicaties, moeten studeren in zijn vak om bij te blijven. In zijn vakliteratuur wordt, overigens geleidelijk aan steeds meer, gebruik gemaakt van wiskundig geformuleerde abstracte begrippen. Theoretische informatica geeft voorbereiding en oefening in het hanteren ervan en is daarom nodig om vakliteratuur te kunnen bestuderen.
Bij dit hele betoog over de betekenis van theoretische informatica voor de informatica zelf en voor de opleiding tot informatica ingenieur moet ik nog wel deze kanttekening maken.
Er volgt niet strict uit welke onderwerpen het meest in aanmerking komen voor opname in een studieprogramma. Voor die inhoudelijke prioriteitenbepaling spelen ook factoren mee als de kennis en de verwachtingen van studenten en de zwaartepunten in het werk van de groep mensen die met het onderwijs is belast.

Na deze uiteenzettingen over onderwijs wordt het tijd nog iets te zeggen over onderzoek in theoretische informatica. Ik weet het, vaak is het andersom en wordt er wel veel over onderzoek en maar weinig over onderwijs gezegd bij het openbaar aanvaarden van het ambt van hoogleraar. Maar ik weet ook uit zeven jaar ervaring dat zowel in energie als in tijd gemeten het onderwijs van mij meer eist en krijgt dan het onderzoek. Dat komt omdat het vakgebied nieuw is en er nog geen vanzelfsprekende en in de praktijk getoetste plaats en inhoud is voor het onderwijs in theoretische informatica. Verder staat op mijn prioriteitenlijstje de mens vóór de zaak en komt de student vóór het vakgebied en daarmee onderwijs vóór onderzoek. Overigens is het niet zo zwart-wit als deze opmerkingen wellicht suggereren, ook is het onderscheid tussen onderzoek en onderwijs soms niet scherp aan te geven.
Het globale doel van theoretisch onderzoek, ook in de informatica, is het ontwikkelen van het vakgebied, het verdiepen van het inzicht erin en het beschikbaar maken van theoretische resultaten voor gebruik bij de beoefening van het vak. In relatie met een opleiding tot informatica ingenieur moeten bij het onderzoek in beginsel alle onderwerpen uit de theoretische informatica vertegenwoordigd zijn. In feite zal een expliciete volgorde moeten vaststaan van de door onderzoek te bewerken onderwerpen. Om tot een juiste afweging van prioriteiten te kunnen komen moeten alle belangrijke ontwikkelingen in het vakgebied voldoende worden bijgehouden. Alweer mede vanwege de snelle ontwikkeling en uitbreiding van de informatica betekent dit dat er veel literatuur bijgehouden en bestudeerd moet worden in de theoretische informatica. Dit behoort echter niet eigen creatief onderzoek te beletten. Ik kan u verzekeren dat het totaal van studie en onderzoek een boeiende bezigheid is.

Zeer gewaardeerde Toehoorders,

Hiermede ben ik gekomen aan het eind van wat ik wilde zeggen over theorie in de praktijk van onderzoek en onderwijs in de informatica. Het ging mij daarbij voornamelijk om het waarom ervan en het perspectief waarin die theorie betekenis heeft. Impliciet heb ik daarmee ook de motivering gegeven voor mijn werkzaamheden en plannen voor verder werk in de theoretische informatica hier aan de Technische Hogeschool Twente. Ik vind dat fijn werk en daarom wil ik deze gelegenheid graag gebruiken om mijn dank te uiten aan allen die betrokken waren bij het mogelijk maken van mijn overgaan van Delft naar Twente. Ik vat dit samen door mijn erkentelijkheid uit te spreken jegens Hare Majesteit de Koningin voor mijn benoeming hier en mijn ontslag in Delft.

Mijn hartelijke dank ook aan velen in Delft, collega's en medewerkers in wetenschappelijk, technisch of administratief opzicht, voor hun bijdrage aan mijn onderwijs- en onderzoekwerk en in mijn persoonlijke groei. Ik heb vele goede herinneringen aan plezierige ervaringen gedurende in totaal negen Delftse jaren. Nu ik al een jaar hier in Twente werk heb ik ook volop redenen om mijn waardering te uiten voor de manier waarop ik werd opgenomen in de Onderafdeling der Toegepaste Wiskunde en vooral natuurlijk door de leden van de vakgroep informatica. Ik zal graag mijn deel aan ons aller werk in de vakgroep en in de ontwikkeling ervan bijdragen.

Ik heb gezegd.