TECHNISCHE HOGESCHOOL EINDHOVEN
Afdeling Algemene Wetenschappen
Onderafdeling der Wiskunde

TOEGEPASTE LOGICA I

van

Prof. Dr. W. Peremans

Najaarssemester 1980

Technische Hogeschool
Eindhoven

Dictaatnummer 2.213
Prijs f. 7,50

Onderafdeling der Wiskunde en Informatica

Toegepaste Logica I

Van prof. dr. W. Peremans

Inhoudsbeschrijving

TOEGEPASTE LOGICA I

W. Peremans

Najaarssemester 1980

JdG, 31 Juli 2005

TECHNISCHE HOGESCHOOL EINDHOVEN

Onderafdeling der Wiskunde

Toegepaste Logica I

van

Prof. Dr. W. Peremans

Najaarssemester 1980

VOORWOORD

Deze syllabus geeft een overzicht van hetgeen in het college Toegepaste Logica I wordt behandeld. Een uitzondering hierop vormt de appendix, waarvan de inhoud niet op het college wordt besproken. De aard van het college brengt met zich mee, dat verbale beschouwingen er een grotere plaats in bekleden dan dit in de meeste colleges over wiskunde het geval is. In deze syllabus is een poging gedaan deze beschouwingen, hoewel zeer beknopt, toch tot hun recht te laten komen, omdat ze voor het begrip van de behandelde onderwerpen essentieel zijn. Het ligt voor de hand, dat de lezer behoefte zal hebben aan een grotere uitvoerigheid; deze kan gevonden worden in het mondeling gegeven college, maar ook in de zeer uitgebreide literatuur over de besproken onderwerpen, waarvan hieronder bij wijze van voorbeeld enkele titels worden genoemd.

LITERATUUR

Met betrekking tot hoofdstuk I:

1. A. Church, Introduction to mathematical logic, Princeton, 1956.
2. H.B. Enderton, A mathematical introduction to logic, New York, 1972.
3. P. Lorenzen, Einführung in die operative Logik und Mathematik, Berlin, 1969.
4. D. Hilbert, W. Ackermann, Grundzüge der theoretischen Logik, Berlin, 1972.

Met betrekking tot hoofdstuk II:

5. H. Hermes, Enumerability, decidability, computability, Berlin, 1969.
6. M.L. Minsky, Computation: finite and infinite machines, Englewood Cliffs, 1967.
7. M. Davis, Computability and unsolvability, New York, 1958.
8. H. Rogers, Theory of recursive functions and effective computability, New York, 1967.

HOOFDSTUK I FORMELE SYSTEMEN

§ 1. Inleiding

De logica houdt zich bezig met de wetmatigheden van ons denkproces. De formele, strikte regels, die hierbij gelden, lenen zich voor een mathematische behandeling, die in de negentiende eeuw tot ontwikkeling gekomen is, beginnend met G. Boole, The mathematical analysis of logic, 1847. Aanvankelijk werd sterk de nadruk gelegd op de analogie met de gewone algebra (algebra der logica). Later heeft men deze analogie laten varen. In de twintigste eeuw is het vak tot grote bloei gekomen, onder verschillende namen, zoals: formele logica, symbolische logica, mathematische logica.

In de praktijk worden de woorden logica en logisch in een ruimere betekenis gebruikt. Voorbeelden zijn logische schakelingen en het gebruik van logica in de theorie van rekenautomaten en programmeertalen. De toevoeging van het bijvoeglijk naamwoord "toegepast" in de naam van dit college duidt aan, dat de keuze en de wijze van behandeling van de onderwerpen uit de mathematische logica gericht is op deze toepassingen.

De wijze van behandeling is wiskundig. Omdat onderwerpen worden besproken, die voor de beoefening der wiskunde fundamenteel zijn, zal een poging worden gedaan de vermogens van de menselijke geest, waarop een beroep gedaan wordt, zoveel mogelijk expliciet te maken. Op de wijsgerige vraag of de menselijke geest deze vermogens inderdaad bezit, zal evenwel niet worden ingegaan. Sommige van de behandelde onderwerpen zijn ook van belang voor het wiskundig grondslagenonderzoek. Een behandeling daarvan is echter niet het oogmerk van dit college.

We zullen ons onder meer bezig houden met aspecten van de wiskundige taal, zoals formules, symbolen, tekens. Over het gebruik daarvan is een grote hoeveelheid kennis, afspraken, gewoonten en vooroordelen vastgeroest. Om ons daarvan los te maken zullen we zoveel mogelijk met een schone lei beginnen en zal er bij de formele opzet geen beroep worden gedaan op wiskundekennis. Bij de toelichting over de grote lijn van het betoog en de doelen, die worden nagestreefd, zal echter wel een beroep moeten worden gedaan op wiskundekennis. Om hierbij verwarring te voorkomen zullen we twee spraakniveaus onderscheiden, één voor het formele werk en één voor de toelichting. Deze

moeten van elkaar onderscheiden kunnen worden. Dit zou bijvoorbeeld kunnen door de toelichting tussen haakjes te zetten; men denke aan de comment-passages in programmeertalen. Omdat in ons geval het commentaar het leeuwendeel van het betoog zal zijn, doen we het anders en zullen we de formele tekst markeren met een kantlijnbalk.

§ 2. Symbolen, symboolrijen, gelijkheid, concatenatie, formules

Om na te gaan wat symbolen zijn, schrijven we twee tekens op: O en | . Ze lijken op nul en één; om ongewenste associaties te voorkomen noemen we ze rondje en streepje. Het symbool is echter niet het geschreven of gedrukte ding; het is een abstractie, waarvan het teken, dat we opschrijven, een aanduiding is. We nemen aan, dat de menselijke geest in staat is dergelijke abstracties aan te nemen en op herkenbare wijze aan te geven. Wij moeten in staat zijn ze te stellen en te herkennen, verschillende realisaties van hetzelfde symbool van elkaar te onderscheiden, verschillende symbolen van elkaar te onderscheiden en een symbool onbepaald vaak te reproduceren. In de laatste veronderstelling zit een overschrijding opgesloten van hetgeen wij werkelijk kunnen, die met "potentieel oneindig" wordt aangeduid. Deze idealisatie van de werkelijkheid is essentieel voor de beoefening van wiskunde.

De volgende stap is, dat we symbolen in een rij achter elkaar schrijven en hetgeen zo ontstaat weer als een geheel beschouwen: een symboolrij. Intuitief kunnen we dit op allerlei manieren doen, links of rechts symbolen bijschrijven, twee verkregen symboolrijen aan elkaar plakken enz. Voor de formele definitie beperken we ons tot het van links naar rechts opschrijven, dat onbepaald vaak herhaald mag worden.

Definitie symboolrij.
O is een symboolrij.
| is een symboolrij.
Uit een symboolrij kan een nieuwe symboolrij worden gevormd door er een rondje (of een streepje) achter te voegen.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een symboolrij.

Dat een bepaling als hierboven gegeven inderdaad iets definieert is een nieuwe veronderstelling, die we maken over de vermogens van de menselijke geest. Het eigenaardige van de definitie is, dat bij de definitie het te definiëren begrip zelf gebruikt wordt, hetgeen de schijn heeft van een vicieuse cirkel. Een definitie van dit soort heet een recurrente definitie. Een symboolrij kan ontstaan zijn door toevoeging van een symbool aan een andere symboolrij, die op zijn beurt weer ontstaan is door toevoeging van een symbool aan een andere symboolrij enz., tot tenslotte een los symbool overblijft. Deze definitie loopt terug; zij doorloopt de symboolrijen in een volgorde tegengesteld aan die welke bij de constructie tot stand komt. Recurrente definities zijn onontkoombaar bij de beoefening van wiskunde.

Behalve symboolrijen opschrijven zullen we ook over symboolrijen praten, niet alleen in het commentaar, maar ook in het formele gedeelte. In het commentaar kunnen we makkelijk voorbeelden geven van beschouwingen over symboolrijen. De vraag kan worden gesteld of er een rondje in de symboolrij voorkomt en of de symboolrij vooraan een rondje heeft. We kunnen vragen naar het aantal symbolen of naar het aantal streepjes in de symboolrij. We kunnen ons afvragen of het aantal rondjes in de symboolrij even is, enz.

In een zin, waarin we iets over een symboolrij zeggen, hebben we een naam voor die symboolrij nodig; we spreken af de symboolrij als zijn eigen naam te gebruiken. De vraag rijst of deze afspraak niet tot moeilijkheden zal leiden.

In de zin

1. Piet is een jongen.

wordt het woord Piet gebruikt als naam van de persoon, waarover de zin gaat. In het algemeen wordt in een zin een object aangeduid met een naam voor dat object; de zin zegt iets over het object, dat door die naam wordt aangeduid. De zin

2. Piet heeft vier letters.

zegt iets over het woord Piet. In overeenstemming met de zojuist gemaakte afspraak moet een naam voor dit woord worden gebruikt. Men zou het woord als zijn eigen naam kunnen kiezen; dit heeft het nadeel, dat twee verschillende objecten (persoon en woord) dezelfde naam hebben. In de gewone omgangstaal komt het verschijnsel dat verschillende objecten dezelfde naam hebben vrij

veel voor. De betekenis van de zin en eventueel de contekst waarin hij geplaatst is moet misverstanden voorkomen. Wil men precieser zijn en geen gelijke namen voor verschillende objecten toelaten, dan moet men zin 2 verwerpen en een naam voor het woord Piet introduceren. Men gebruikt wel eens enkele aanhalingstekens voor "een naam van" en schrijft:

2. 'Piet' heeft vier letters.

Hierin is 'Piet' een naam voor het woord Piet en de zin zegt iets over het woord.

Een object kan verschillende namen hebben; 2 + 2 en 4 zijn verschillende namen voor hetzelfde getal. Deze dubbelzinnigheid is niet schadelijk voor de betekenis van de zinnen, waarin deze namen gebruikt worden. Let wel, dat in deze commentaar-alinea het gebruik van aanhalingstekens achterwege is gebleven.

In de zin

3. Piet is met zwarte letters geschreven.

heeft de betekenis betrekking op een bepaalde realisatie van de zin en het woord daarin. Hier stuiten we op de onderscheiding tussen de zin als abstractum en een bepaalde concrete realisatie van deze zin. Deze zelfde onderscheiding hebben we bij een symbool al opgemerkt. Zinnen van het type 3 verwerpen we volledig. Wil men over een realisatie spreken, dan dient in de zin zelf aangegeven te worden welke realisatie bedoeld is.

Terugkerend tot onze afspraak om als naam van een symboolrij die symboolrij zelf te gebruiken constateren we, dat dit geen kwaad kan, mits een symboolrij niet de naam is van iets anders. In het formele werk klopt dat ook; een symboolrij betekent niets. Dit neemt niet weg, dat er in de commentaarsfeer best associaties aan een symboolrij verbonden mogen zijn.

Het feit, dat symboolrijen op zichzelf niets betekenen, is echter niet voldoende om dubbelzinnigheid uit te sluiten. Er moet ook nog voor gezorgd worden, dat een symboolrij niet verward kan worden met een woord uit de taal, waarin over symboolrijen wordt gesproken. Hiertoe is het voldoende, dat de formele symbolen niet in die taal voorkomen, of iets nauwkeuriger uitgedrukt, dat ze in die taal uitsluitend worden gebruikt om zichzelf aan te duiden. In het bijzonder zou, als we later de voorraad symbolen gaan vergroten, het gebruik van gewone letters daarbij niet toegestaan zijn. Tegen dit voorschrift zullen we echter, zoals in de wiskunde gebruikelijk, wel zondigen. Door verschillende lettertypen voor de symbolen en de gewone letters te bezigen, kan men dan verwarring uitsluiten, mits men zich strikt aan het gemaakte onderscheid houdt.

Om bij het formele werk over symboolrijen te kunnen praten is een taal nodig; daarvoor kiezen we het nederlands. In definities hebben we dat al gedaan. Helaas is nergens precies beschreven, wat correct nederlands is. Toch nemen we de omgangstaal als gegeven aan. Men noemt een formeel systeem ook vaak een taal; wij zullen dat niet doen. Het systeem of de taal waarin uitspraken over het formele systeem worden gedaan, noemt men het metasysteem of de metataal van het systeem. We moeten het formele systeem en de metataal van elkaar onderscheiden. De metataal verschilt weer van het commentaar.

Men zou op de gedachte kunnen komen de metataal ook te formaliseren. Doet men dit, dan rijst de behoefte om ook over dit formele systeem te praten; daarvoor is een meta-metataal nodig enz. We gaan hier niet op in.

In de metataal schrijven we de definitie van symbool op:

Definitie symbool.
O is een symbool.
| is een symbool.
Er zijn geen andere symbolen.

We willen de definitie van symboolrij ook eenvoudiger opschrijven. In het commentaar zouden we "een symboolrij die met rondje begint" als volgt willen beschrijven: Oa, waarin a een willekeurige symboolrij is. Deze a is een variabele, het is een symbool in de metataal. We signaleren een nieuwe abstractie op het niveau van de metataal, nl. het begrip "willekeurige symboolrij". De invoering van deze abstractie staat los van de invoering van een symbool ervoor: ook in de eerste versie van de definitie van symboolrij speelde dit begrip al een rol.

Definitie symboolrij.
O is een symboolrij.
| is een symboolrij.
Als a een symboolrij is, dan is aO een symboolrij.
Als a een symboolrij is, dan is a| een symboolrij.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een symboolrij.

In deze definitie moet aO opgevat worden als de symboolrij die ontstaat door achter de symboolrij a een rondje toe te voegen. Analoog a| . Hiermee is nog niet verklaard, wat Oa zou moeten betekenen.

We willen nu gelijkheid van symboolrijen definiëren. Intuïtief constateren we of twee (lange) symboolrijen gelijk zijn door ze gelijktijdig van links naar rechts na te lopen. Dit komt er op neer, dat twee symboolrijen

slechts dan gelijk zijn, als het proces waarmee ze gemaakt zijn hetzelfde is. Iedere symboolrij die men maakt verschilt van degene, die tijdens het proces eerder tot stand gekomen zijn en ook van degene, die ontstaan door tijdens het proces een andere weg in te slaan. In de eerste versie van de definitie was dit gedeeltelijk in het woord "nieuwe" tot uitdrukking gebracht.

Gelijkheid wordt nu ook recurrent gedefinieerd. We gebruiken = als symbool voor de gelijkheid; dit is een symbool in de metataal.

Definitie gelijkheid van symboolrijen.
O = O.
| = | .
Als a en b symboolrijen zijn en a = b, dan aO = bO en a| = b|.
a = b geldt voor symboolrijen a en b slechts dan, als dit uit bovenstaande voorschriften volgt.

Ook dit is een recurrente definitie. maar niet van een object, zoals bij de symboolrij het geval was, maar van een relatie tussen objecten, waarbij de geldigheid van de relatie voor langere symboolrijen teruggebracht wordt op de geldigheid voor kortere. We noemen dit een definitie door recurrentie langs de opbouw van een symboolrij.

In de formulering van deze definitie, evenals in die van de vorige, treden herhalingen op, doordat voor elk van beide symbolen hetzelfde afzonderlijk moet worden geformuleerd. Dit kunnen we vermijden door in de metataal een variabele voor symbolen in te voeren; deze variabele kan de twee waarden O en | aannemen. We kiezen een griekse letter voor deze variabele. De definitie van gelijkheid ziet er dan als volgt uit:

Definitie gelijkheid van symboolrijen.
Als ε een symbool is, dan ε = ε.
Als a en b symboolrijen zijn, ε een symbool is en a = b, dan aε = bε.
a = b geldt voor symboolrijen a en b slechts dan, als dit uit bovenstaande voorschriften volgt.

De definitie van symboolrij kan op soortgelijke wijze worden aangepast.

We moeten ons goed rekenschap geven van de vèrreikende consequenties van de laatste zin van de gelijkheidsdefinitie. aO betekent de symboolrij, die is ontstaan uit a, door er O achter te plaatsen; een symboolrij is niets anders dan zijn constructie. Daaruit volgt, dat aO = b alleen ontstaan kan

zijn uit iets van de gedaante a = b' door over te gaan op aO = b'O, dus b is ontstaan door achter een symboolrij b', waarvoor geldt a = b', het symbool O te plaatsen. Dit geeft de volgende stelling.

Stelling 2.1. Als a en b symboolrijen zijn, ε een symbool is en aε = b, dan bestaat er een symboolrij b', zodat a = b' en b is ontstaan door achter b' het symbool ε te plaatsen.

We geven, zoals gebruikelijk, het niet gelden van de gelijkheid aan met ≠. De volgende stelling volgt direct uit stelling 2.1.

Stelling 2.2. Als a en b symboolrijen zijn, dan

$$aO \neq O,\ aO \neq |,\ aO \neq b|,\ a| \neq O,\ a| \neq |,\ a| \neq bO.$$

Stelling 2.3. Als a en b symboolrijen zijn, α en β symbolen en aα = bβ, dan a = b en α = β.

Bewijs. De keuzen O, | en |, O voor α, β zijn onmogelijk op grond van stelling 2.2. Er blijven alleen de keuzen O,O en |, | over, maar dan is α = β, want O = O en | = |.
Als aO = bO, dan volgt uit stelling 2.1, dat er een symboolrij b' bestaat, zodat a = b', en bO is ontstaan door achter b' het symbool O te plaatsen, maar bO is de symboolrij die is ontstaan door achter b het symbool O te plaatsen. Dus b en b' zijn dezelfde symboolrij en a = b.
Het geval a| = b| wordt op analoge wijze behandeld.

We tonen nu aan dat gelijkheid een equivalentierelatie is.

Stelling 2.4. Als a, b en c symboolrijen zijn, dan geldt:

a = a;
als a = b, dan b = a;
als a = b en b = c, dan a = c.

We bewijzen stelling 2.4 met een bewijs door recurrentie langs de definitie van gelijkheid. Bij een dergelijk bewijs wordt bij de vorming van een nieuwe gelijkheid uit een oude in de definitie aangenomen dat de te bewijzen eigenschap voor de oude gelijkheid geldt en afgeleid dat zij voor de nieuwe ge-

lijkheid ook geldt. De laatste zin van de gelijkheidsdefinitie garandeert, dat de eigenschap in alle gevallen geldt.

Bewijs. Om a = a te bewijzen passen we recurrentie langs de definitie van gelijkheid toe. ε = ε voldoet. Als a = a, dan ook aε = aε, hetgeen ook voldoet.
We bewijzen nu, dat uit a = b volgt b = a door recurrentie langs de definitie van a = b. We beginnen met ε = ε; dit levert ε = ε, hetgeen klopt. Stel nu a = b en ook b = a. Dan aε = bε en bε = aε, hetgeen ook klopt.
Tenslotte de eigenschap, dat uit a = b en b = c volgt a = c. We passen recurrentie toe langs de definitie van a = b. We beginnen met ε = ε, dan levert ε = c, dat ε = c, hetgeen klopt. We beschouwen nu a = b en nemen aan dat voor iedere c, waarvoor b = c, ook a = c. We hebben nu aε = bε en veronderstellen bε = c. Op grond van stelling 2.1 bestaat er een symboolrij c', zodat b = c' en c is ontstaan door achter c' het symbool ε te plaatsen. Volgens de recurrentieveronderstelling volgt uit b = c', dat a = c', dus aε = c'ε, aε = c.

Dat een bewijs door recurrentie inderdaad overtuigend de juistheid aantoont van hetgeen erdoor bewezen moet worden, wordt hier als één van de vermogens van de menselijke geest zonder nadere verklaring aanvaard.

We passen stelling 2.4 in de toekomst stilzwijgend toe. Een voorbeeld van zulk een toepassing is het opschrijven van een geschakelde gelijkheid van het type a = b = ... = d. Het is dan de bedoeling, dat ieder lid van deze geschakelde gelijkheid gelijk is aan ieder ander lid. Om dit aan te tonen is het voldoende om de gelijkheid van elk tweetal direct opeenvolgende leden te bewijzen.

Het volgende begrip, dat we willen beschouwen, is het achter elkaar schrijven van twee symboolrijen, genaamd concatenatie. Om de constructie van ab te verkrijgen moeten we eerst de constructie van a uitvoeren en na voltooiing daarvan doorgaan met de constructie van b. De schrijfwijze ab voor het resultaat heeft het bezwaar, dat als we abc opschrijven, we niet weten of daarmee bedoeld is om eerst ab te construeren en dan c erachter, dus (ab)c, dan wel eerst bc te construeren en dit achter a, dus a(bc). Verder

willen we ook het verschil tussen de concatenatie en het reeds ingevoerde achterplaatsen van een symbool in de notatie tot uitdrukking brengen. Daarom schrijven we voorlopig (a * b) voor de concatenatie van a en b. Verder voeren we om de definitie makkelijk op te schrijven een nieuw symbool in de metataal in, namelijk :=, hetgeen betekent "wordt gedefinieerd door". Links van dit symbool staat iets wat gedefinieerd moet worden, rechts staat hetgeen het per definitie is. De symbolen *, (,) zijn symbolen in de metataal.

Definitie concatenatie.
Als a en b symboolrijen zijn en ε een symbool is, dan

$$(a * \varepsilon) := a\varepsilon \ ,$$
$$(a * b\varepsilon) := (a * b)\varepsilon \ .$$

Dit is een definitie van (a * b) door recurrentie langs de opbouw van b.

Het is van belang om vast te stellen of vervanging van symboolrijen door gelijke tot resultaat heeft dat de concatenatie van deze symboolrijen door een gelijke wordt vervangen. Dit betekent, dat als a = b en c = d, dan (a * c) = (b * d). Dit volgt makkelijk uit de volgende stelling.

Stelling 2.5. Als a, b en c symboolrijen zijn en a = b, dan

$$(a * c) = (b * c) \quad \text{en} \quad (c * a) = (c * b) \ .$$

Bewijs. De eerste gelijkheid bewijzen we door recurrentie langs de opbouw van c.
(a * ε) = aε. Uit a = b volgt aε = bε. Verder bε = (b * ε). Als (a * c) = = (b * c), dan (a * cε) = (a * c)ε = (b * c)ε = (b * cε).
De tweede gelijkheid bewijzen we door recurrentie langs de definitie van gelijkheid.
(c * ε) = (c * ε). Als a = b en (c * a) = (c * b), dan aε = bε en
(c * aε) = (c * a)ε = (c * b)ε = (c * bε).

Gebruik makend van stelling 2.5 leiden we uit a = b en c = d direct af, dat (a * c) = (b * c) = (b * d). Het beginsel, dat begrippen die we voor symboolrijen definiëren bestand moeten zijn tegen het vervangen van symboolrijen door gelijke, noemen we het beginsel van de extensionaliteit. Het zal

ook voor in de toekomst te definiëren begrippen moeten gelden, al zullen we dat niet altijd verifiëren.

We tonen nu aan, dat concatenatie associatief is.

Stelling 2.6. Als a, b en c symboolrijen zijn, dan

$$((a * b) * c) = (a * (b * c)) .$$

Bewijs. We passen recurrentie toe naar de opbouw van c.

$$((a * b) * \varepsilon) = (a * b)\varepsilon = (a * b\varepsilon) = (a * (b * \varepsilon)) .$$

Stel $((a * b) * c) = (a * (b * c))$, dan

$$((a * b) * c)\varepsilon = (a * (b * c))\varepsilon$$

en

$$((a * b) * c\varepsilon) = ((a * b) * c)\varepsilon = (a * (b * c))\varepsilon =$$

$$= (a * (b * c)\varepsilon) = (a * (b * c\varepsilon)) .$$

Op grond van de associativiteit van de concatenatie laten we voortaan de haakjes weg en schrijven a * b voor de concatenatie van a en b; dit is een toepassing van het extensionaliteitsbeginsel, omdat het inhoudt, dat we in de notatie geen onderscheid maken tussen gelijke objecten, die op verschillende manieren zijn verkregen. Omdat $a * \varepsilon = a\varepsilon$, zouden we de ster ook nog kunnen weglaten; we doen dit voorlopig nog niet.

In het commentaar zeggen we, dat in de verzameling van de symbboolrijen door * een binaire operatie is gedefinieerd, die associatief is, zodat de verzameling ten opzichte van deze operatie een semigroép vormt. In semigroepen heeft men soms een neutraal element e, dat voldoet aan $a * e = e * a = a$ voor alle a. Het is echter duidelijk dat onze semigroep zulk een element niet bevat. Men zou het kunnen verkrijgen door introductie van de lege symboolrij, dat is een rij die geen symbolen bevat. In de literatuur wordt bij de definitie van symboolrij de lege rij vaak meegeteld. Wij hebben dit niet gedaan, omdat deze rij onzichtbaar is en er dus niet vastgesteld kan worden of hij wel of niet in de beschouwing wordt betrokken. Toch is het vaak prettig ook over de lege symboolrij te kunnen beschikken. Daartoe voeren we een symbool Λ in de metataal in, dat we de lege symboolrij noemen. Dit is geen symboolrij. Als we in het midden willen laten of we met een echte symboolrij te maken hebben of met de lege symboolrij, spreken we van een eventueellege symboolrij.

Definitie lege symboolrij, eventueel-lege symboolrij, concatenatie.
Λ is een symbool in de metataal, genaamd lege symboolrij.
Als a een symboolrij of de lege symboolrij is, noemen we a een eventueel-lege symboolrij.
Als a een eventueel-lege symboolrij is, dan

$$a * \Lambda := a ,$$
$$\Lambda * a := a .$$

Het is makkelijk te bewijzen, dat concatenatie van eventueel-lege symboolrijen associatief is, want als minstens één van de factoren Λ is, volgt het uit de definitie van concatenatie voor Λ, en als geen van de factoren Λ is, volgt het uit stelling 2.6.

De verzameling der eventueel-lege symboolrijen vormt ten opzichte van de concatenatie een semigroep met neutraal element. Dat noemen we een monoïde.

Een voorbeeld van het nut van de lege symboolrij vinden we, als we proberen te formuleren, dat een symboolrij a als bestanddeel in een symboolrij b voorkomt. Dat is zo, als er symboolrijen p en q bestaan, zodat $b = p * a * q$. Maar a mag ook vooraan staan: $b = a * q$, of achteraan: $b = p * a$, of zelfs gelijk zijn aan b: $b = a$. Deze gevalonderscheidingen zijn overbodig als we de lege symboolrij toelaten:

Definitie voorkomen als bestanddeel.
Als a en b symboolrijen zijn, dan komt a als bestanddeel in b voor, als er eventueel-lege symboolrijen p en q bestaan, zodat $b = p * a * q$.

We kunnen voor a en b zelf natuurlijk ook de lege symboolrij toelaten met een gelijkluidende definitie. Dan komt Λ in iedere eventueel-lege symboolrij als bestanddeel voor en bevat Λ alleen Λ als bestanddeel.

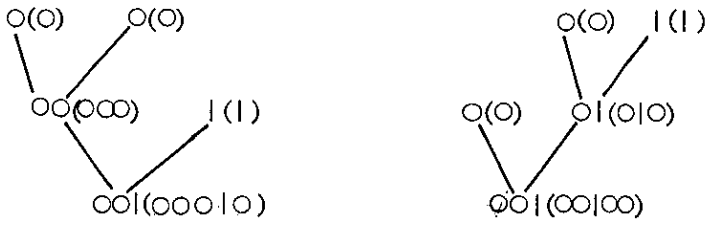
Men pleegt wel eens, in plaats van het achterplaatsen van één symbool, zoals wij gedaan hebben, de concatenatie als ongedefinieerd begrip in te voeren. De definitie van symboolrij berust dan daarop en eenzelfde symboolrij kan op verschillende wijzen ontstaan. Er komen dan moeilijkheden bij de definitie van gelijkheid, die dan niet teruggebracht kan worden op de identieke opbouw. Het aannemen van de gelijkheid als ongedefinieerd begrip is

bezwaarlijk, omdat symboolrijen willekeurig lang kunnen zijn en de verzameling van symboolrijen oneindig is. Definities door recurrentie langs de opbouw zijn ook niet zonder meer geoorloofd als hetzelfde object langs verschillende wegen kan worden opgebouwd. Het is dan de vraag of het gedefinieerde onafhankelijk is van de gekozen opbouw. We zullen dit met een voorbeeld verduidelijken. De volgende "definitie" door recurrentie langs concatenatie van een afbeelding φ, die aan iedere symboolrij een symboolrij moet toevoegen, is ondeugdelijk.

Als a en b symboolrijen zijn en α een symbool is, dan

$$\varphi(\alpha) := \alpha \ ,$$
$$\varphi(a * b) := \varphi(a) * \varphi(b)O \ .$$

We laten zien dat dit misloopt door twee verschillende wijzen van opbouw van OO| in boomvorm te demonsteren met achter iedere symboolrij de bijbehorende φ-waarden:



In formele systemen is het vaak gebruikelijk niet alle symboolrijen in de beschouwing te betrekken. In het commentaar stellen we vast, dat er systemen zijn met andere symbolen dan de door ons gebruikte O en |. Zo gebruikt men bij het rekenen cijfersymbolen, +-teken en andere operatietekens en ook = als formeel symbool. In de algebra komen daar nog letters bij. (Let wel, dat we = al als symbool in de metataal hadden ingevoerd. In § 4 zullen we dat ook met + doen. Pas op voor verwarring!) In andere systemen komen weer andere symbolen voor. Zo heeft men in logische systemen symbolen voor "en", "of", enz. In de algebra zal men (a+b)c als een zinvolle symboolrij opvatten, maar a)cb(+ niet. Zulke zinvolle symboolrijen noemt men vaak welgevormde formules (engels: well-formed formula met standaardafkorting wff). Wij zullen kortweg van formules spreken. In het door ons beschouwde systeem zullen we formules invoeren, die natuurlijke getallen zullen moeten voorstellen. Alvorens dit te doen houden we eerst een intuïtieve beschouwing over natuurlijke getallen.

§ 3. Intuïtieve behandeling der natuurlijke getallen

We onderscheiden hoofdtelwoorden of cardinaalgetallen (één, twee, drie,..) en rangtelwoorden of ordinaalgetallen (eerste, tweede, derde, ..). In de wiskunde wordt in de notatie geen onderscheid gemaakt (1,2,3,...). We beginnen met cardinaalgetallen (aantallen) en baseren deze op het begrip evenveel. Als voor twee (eindige) verzamelingen geldt, dat er een eeneenduidige afbeelding van de ene op de andere tot stand kan worden gebracht, zeggen we dat ze evenveel elementen hebben of dat de verzamelingen gelijkmachtig of equipotent zijn. Om dit vast te stellen hoeft men niet te kunnen tellen. Verder is de geldigheid van deze betrekking niet afhankelijk van de keuze van de afbeelding (althans voor eindige verzamelingen waartoe wij ons hier beperken) in die zin dat iedere andere eeneenduidige afbeelding van de ene verzameling in de andere ook een afbeelding op is.

De evenveel-relatie is een equivalentierelatie en leidt dus tot een indeling in klassen van de verzamelingen, waarbij alle verzamelingen in één klasse evenveel elementen hebben en verzamelingen uit verschillende klassen niet. Men zou op de gedachte kunnen komen om deze klassen als aantallen te gebruiken, maar het bezwaar hiertegen is de onbepaaldheid en oeverloosheid van zulk een klasse, omdat de aard van de erin zittende verzamelingen en hun elementen onbepaald is. Een ander idee zou zijn uit elke klasse een representant te kiezen, die als maatstafverzameling wordt gebruikt. Om na te gaan of een verzameling tot een zekere klasse behoort, zou men haar dan met de maatstafverzameling uit die klasse kunnen vergelijken.

Om tot maatstafverzamelingen te komen introduceren we ordinaalgetallen. Uitgangspunt hierbij is dat in de aantallen op natuurlijke wijze een ordening aan te brengen is. Als twee eindige verzamelingen A en B niet gelijkmachtig zijn, dan bestaat er hetzij een eeneenduidige afbeelding van B in A hetzij een eeneenduidige van A in B en niet beide. In het eerste geval zeggen we dat A meer elementen heeft dan B en in het tweede geval dat B meer elementen heeft dan A. Op deze wijze wordt een ordening in de aantallen aangebracht, die aan de eisen van een nette mathematische ordening voldoet.

Hoe men aan ordinaalgetallen komt is enigszins arbitrair. Wij zullen ze als formules in een formeel systeem introduceren, maar het kan ook anders. In deze intuïtieve inleiding nemen we aan, dat we ze hebben. Natuurlijke

getallen staan in een rij: 0,1,2,3,... . Omdat we deze getallen gebruiken om aantallen mee te meten wordt 0 ook als een natuurlijk getal beschouwd terwille van de lege verzameling. We gebruiken de natuurlijke getallen nu om verzamelingen te tellen. Bij de traditionele manier van tellen beginnen we met 1 en etiketteren de elementen van de verzameling achtereenvolgens met 1,2,..., totdat we ze alle gehad hebben. Het laatst gebruikte getal geeft dan het aantal aan. Het komt neer op het tot stand brengen van een eeneenduidige betrekking tussen de verzameling en een met 1 beginnend segment van de natuurlijke getallen. Het bezwaar van de methode is, dat zij niet werkt voor de lege verzameling: er is dan geen laatst gebruikt getal. Het is daarom een betere methode om het tellen met 0 te doen beginnen. Het aantal is dan het voorste in de rij van de niet gebruikte getallen, hetgeen ook klopt bij de lege verzameling. Op deze wijze fungeren de beginsegmenten van de geordende verzameling der natuurlijke getallen als maatstafverzamelingen.

Een ander voordeel, dat overigens voor onze beschouwingen niet van belang is, van de zojuist geïntroduceerde wijze van tellen, die met 0 begint, is dat zij beter dan de traditionele methode tot oneindige verzamelingen kan worden uitgebreid.

We merken ook nog op, dat in de verzamelingsleer een vrij natuurlijke introductie van ordinaalgetallen gebruikelijk is. Daar is 0 per definitie de lege verzameling, 1 de verzameling die 0 als enige element heeft, 2 de verzameling die 0 en 1 als enige elementen heeft, enz. Het aantrekkelijke van deze methode is, dat een ordinaalgetal een verzameling is, die zelf het goede aantal elementen heeft en dus als maatstafverzameling kan dienen. We gaan hier echter niet verder op in.

§ 4. Natuurlijke getallen als formeel systeem. Elementaire rekenkunde

We willen de natuurlijke getallen als formules introduceren. Men zou daarvoor een rij streepjes kunnen nemen; evenveel als het getal, dat men wil aanduiden. Dan zou 0 met de lege rij corresponderen, die we niet als symboolrij hebben erkend. Dit zou op te vangen zijn door één streepje te veel bij elk getal. We doen het anders en gebruiken O als signaal dat er een natuurlijk getal op komst is: O,O|,O||,O|||,... . Dit kan makkelijk in een recurrente definitie; O is uitgangssymbool, achterschrijven doen we alleen |.

De ordening is die van de constructie. Op deze wijze definiëren we een formeel systeem; in de definitie nemen we ook de gebruikte symbolen op.

Definitie formeel systeem der natuurlijke getallen.
O en | zijn symbolen en er zijn geen andere.
O is een formule.
Als de symboolrij a een formule is, dan is a| een formule.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een formule.

We gaan in de metataal van verzamelingstheoretische terminologie gebruik maken om tot eenvoudiger formuleringen te komen. Het gevaar is, dat men niet alleen terminologie maar ook kritiekloos begrippen uit de verzamelingsleer gaat gebruiken; men bedenke, dat verzamelingen als die der symboolrijen en die der natuurlijke getallen oneindig zijn. Wij hebben deze niet als een voltooid geheel beschouwd ("actueel oneindig") en willen dat voorlopig ook niet doen.

We gebruiken de letter N voor de verzameling der natuurlijke getallen. $a \in N$ betekent dan: a is een formule in het formele systeem der natuurlijke getallen. De definitie van formule kan dan als volgt worden vertaald.

$O \in N$.
Als $a \in N$, dan $a| \in N$.
Alleen hetgeen op grond van bovenstaande voorschriften tot N behoort, is element van N.

We maken een opmerking terzijde over een andere gebruikelijke manier om natuurlijke getallen te introduceren, die aanzienlijk van de hier gegeven methode verschilt. Men vat dan N wel als een gegeven verzameling op, die men met axioma's probeert te karakteriseren. Het achterschrijven van een | in ons formele systeem is op te vatten als een afbeelding, die aan ieder element van N een element van N toevoegt, genaamd S (successor). Omdat de | er achter staat, schrijven we het functiesymbool ook achter het argument van de functie: $aS := a|$. We nemen het getal 0 en de afbeelding S als uitgangspunt. Het feit dat het achtervoegen van | steeds nieuwe formules oplevert, wordt vertaald tot eigenschappen van S: eeneenduidig en 0 is geen beeld. Tenslotte

moet nog het feit vertaald worden, dat natuurlijke getallen alleen die zijn, die uit 0 door herhaald toepassen van S worden verkregen: een deelverzameling, die 0 bevat en met iedere a $\in$ N ook aS, is N.

Axioma's van Peano.
Verzameling N en afbeelding S.
$0 \in N$.
Als $a \in N$, dan $aS \in N$.
Als $a \in N$, dan $aS \neq 0$.
Als $a \in N$, $b \in N$, en $aS = bS$, dan $a = b$.
Als $D \subset N$, $0 \in D$ en uit $a \in D$ volgt $aS \in D$, dan $D = N$ (inductie-axioma).

Wat een willekeurige deelverzameling van N is, weten we niet. Deze abstractie accepteren we niet. Daarom is het inductie-axioma voor ons onaanvaardbaar. Dit sluit het feit niet uit, dat we wel deelverzamelingen van N kunnen maken, ook oneindige, zoals de verzameling E der even getallen.

Definitie even getal.
$0 \in E$.
Als $a \in E$, dan $a|| \in E$.
Alleen hetgeen op grond van bovenstaande voorschriften tot E behoort, is element van E.

We keren terug tot ons formele systeem der natuurlijke getallen. Hierin kunnen we nu elementaire rekenkunde gaan ontwikkelen. We behandelen daarvan slechts een aanzet.

We willen eerst a + b definiëren; hierin is + een symbool in de metataal. Eigenlijk zouden we (a + b) moeten schrijven, maar we zijn slordig en bezuinigen wat op haakjes. In ieder geval laten we de buitenste haakjes weg en schrijven a + b voor (a + b) en a + (b + c) voor (a + (b + c)).

Definitie optelling.
Als a en b natuurlijke getallen zijn, dan

$$a + 0 := a \ ,$$
$$a + b| := (a + b)| \ .$$

We vragen ons af, of dit inderdaad een definitie is. Het is kennelijk bedoeld als definitie van a + b door recurrentie langs de opbouw van het natuurlijke getal b. Dit is dus niet recurrentie langs de opbouw van een symboolrij, maar langs de opbouw van een formule. In het algemeen hoeft de eenduidigheid van de opbouw bij een recurrente opbouw van formules niet gewaarborgd te zijn. In dit geval, nu bij iedere constructiestap slechts één symbool wordt toegevoegd is dit wel het geval.

We stellen ook nog vast dat recurrentie langs de opbouw van een natuurlijk getal volledige inductie genoemd wordt. Er bestaan definities en bewijzen door volledige inductie.

Stelling 4.1. Voor natuurlijke getallen a, b en c geldt

$$(a + b) + c = a + (b + c) .$$

Het bewijs van stelling 4.1 slaan we over (volledige inductie naar c). Op grond van stelling 4.1 schrijven we a + b + c en langere sommen meestal zonder haakjes.

Stelling 4.2. Voor natuurlijke getallen a geldt

$$O + a = a ,$$
$$O| + a = a| .$$

Bewijs. Beide met volledige inductie naar a.
$O + O = O$. Als $O + a = a$, dan $O + a| = (O + a)| = a|$.
$O| + O = O|$. Als $O| + a = a|$, dan $O| + a| = (O| + a)| = a||$.

Stelling 4.3. Voor natuurlijke getallen a en b geldt

$$a + b = b + a .$$

Bewijs. Volledige inductie naar b.
$a + O = a = O + a$. Als $a + b = b + a$, dan

$$a + b| = (a + b)| = (b + a)| = O| + (b + a) = (O| + b) + a = b| + a .$$

In de metataal gebruiken we 0 als symbool voor O en 1 als symbool voor O|. We kunnen dan ook a + 1 schrijven voor a|.

We kunnen de ontwikkeling der rekenkunde voortzetten: invoering van vermenigvuldiging en ordening enz. We voeren dat niet uit. Het is in details te vinden in: E. Landau, Grundlagen der Analysis. Weliswaar verschilt zijn opzet van de onze, maar vertaling is niet al te moeilijk. Hij gebruikt de axioma's van Peano als uitgangspunt met 1 als kleinste natuurlijke getal, en aanvaardt dus meer verzamelingsleer dan wij, maar is in het gebruik van die grotere vrijheid zeer terughoudend. Aan de andere kant veronderstelt hij minder dan wij: voor ons zijn definities en bewijzen door volledige inductie zonder meer aanvaardbaar. Voor bewijzen door volledige inductie is dat bij hem ook zo, omdat deze uit het inductie-axioma volgen: neem voor D de verzameling der natuurlijke getallen waarvoor de te bewijzen eigenschap geldt. Definities door volledige inductie volgen niet uit de axioma's van Peano en moeten bij hem in ieder voorkomend geval met een existentiebewijs worden waargemaakt.

We zullen in het vervolg vrijelijk van de elementaire rekenkunde gebruik maken. Op één punt willen we hier nog wijzen en dat is een andere vorm van volledige inductie, die pas kan worden geformuleerd en gerechtvaardigd als de ordening der natuurlijke getallen is ingevoerd. Deze vorm van volledige inductie kan bij bewijzen en bij definities worden toegepast. We formuleren het hier voor bewijzen.

Een eigenschap van alle natuurlijke getallen is bewezen als we het volgende aantonen: als n een natuurlijk getal is en de eigenschap geldt voor alle natuurlijke getallen $< n$, dan geldt de eigenschap voor n.

We leggen nu een verband tussen concatenatie van symboolrijen en natuurlijke getallen via het begrip lengte van een symboolrij a, aangeduid met $\ell(a)$.

Definitie lengte van een symboolrij.
Als a een symboolrij is en ε een symbool, dan

$$\ell(\varepsilon) := 1 ,$$
$$\ell(a\varepsilon) := \ell(a) + 1 .$$

Stelling 4.4. Voor symboolrijen a en b geldt

$$\ell(a * b) = \ell(a) + \ell(b) ,$$
$$\ell(a) > 0 .$$

Het bewijs van stelling 4.4 slaan we over (recurrentie naar de opbouw van b, resp. a).

Stelling 4.5. Voor symboolrijen a geldt:
hetzij er bestaat een symbool α, zodat $a = \alpha$ en $\ell(a) = 1$,
hetzij er bestaat een symboolrij a' en een symbool α, zodat $a = a'\alpha$ en $\ell(a) > 1$.

Bewijs. Recurrentie langs de opbouw van a.
$\varepsilon = \varepsilon$ en $\ell(\varepsilon) = 1$.
$a\varepsilon = a\varepsilon$ en $\ell(a\varepsilon) = \ell(a) + 1 > 1$.

Stelling 4.6. Als a een symboolrij is, k een natuurlijk getal en $0 < k < \ell(a)$, dan bestaan er eenduidig bepaalde symboolrijen b en c, zodat $a = b * c$ en $\ell(b) = k$.

We merken op, dat "eenduidig bepaalde" betekent: als $a = b * c = b' * c'$, $\ell(b) = \ell(b') = k$, dan $b' = b$ en $c' = c$.

Bewijs. Recurrentie langs de opbouw van a.
$\ell(\varepsilon) = 1$; er is geen natuurlijk getal k, waarvoor $0 < k < \ell(\varepsilon)$.
Stel nu a zo dat voor alle natuurlijke getallen k, waarvoor $0 < k < \ell(a)$, eenduidig bepaalde symboolrijen b en c bestaan, zodat $a = b * c$ en $\ell(b) = k$.
$\ell(a\varepsilon) = \ell(a) + 1$ en stel $0 < k < \ell(a) + 1$.
Als $k < \ell(a)$, dan zijn er symboolrijen b en c met $a = b * c$ en $\ell(b) = k$, maar dan is $a\varepsilon = b * c\varepsilon$ en $\ell(b) = k$. Stel ook $a\varepsilon = d * e$ met $\ell(d) = k$, dan is $k + \ell(e) = \ell(d) + \ell(e) = \ell(d * e) = \ell(a\varepsilon) = \ell(a) + 1$, $\ell(e) = \ell(a) + 1 - k > 1$.
Volgens stelling 4.5 is er een symboolrij e' en een symbool ε' zodat $e = e'\varepsilon'$, $a\varepsilon = (d * e')\varepsilon'$. Nu levert stelling 2.3 dat $a = d * e'$ en $\varepsilon = \varepsilon'$. Volgens recurrentieveronderstelling is dan $d = b$ en $e' = c$, dus $e = e'\varepsilon' = c\varepsilon$.
Als $k = \ell(a)$, dan is $a\varepsilon = a * \varepsilon$ en $\ell(a) = k$. Stel ook $a\varepsilon = d * e$ met $\ell(d) = k = \ell(a)$. Dan is $\ell(e) = 1$. Volgens stelling 4.5 is er een symbool ε' zodat $e = \varepsilon'$, dus $a\varepsilon = d * \varepsilon' = d\varepsilon'$. Volgens stelling 2.3 is nu $a = d$ en $\varepsilon = \varepsilon'$, dus $e = \varepsilon$.

Stelling 4.7. Voor symboolrijen a, b en c geldt:

als $c * a = c * b$, dan $a = b$,
als $a * c = b * c$, dan $a = b$.

Eenvoudig gevolg van de vorige stellingen. De in stelling 4.7 uitgedrukte eigenschap van symboolrijen wordt wel de schrapeigenschap genoemd.

Stelling 4.8. Voor symboolrijen a, b, c en d geldt:
als a * b = c * d, dan bestaat er een eventueel-lege symboolrij p, zodat
hetzij a = c * p, d = p * b,
hetzij c = a * p, b = p * d.

Bewijs. We onderscheiden de gevallen $\ell(a) > \ell(c)$, $\ell(a) < \ell(c)$, $\ell(a) = \ell(c)$.
Als $\ell(a) > \ell(c)$, dan bestaan er op grond van stelling 4.6 symboolrijen a' en p zodat a = a' * p en $\ell(a') = \ell(c)$. Dan is a' * p * b = a * b = c * d. Wederom op grond van stelling 4.6 is dan d = p * b en c = a', dus a = c * p.
Het geval $\ell(a) < \ell(c)$ gaat analoog.
Als $\ell(a) = \ell(c)$, dan voldoet de keuze p = Λ op grond van stelling 4.6.

We kunnen het bovenstaande uitbreiden tot lege symboolrijen door te definiëren: $\ell(\Lambda) := 0$. Behoudens enkele voor de hand liggende wijzigingen zijn bovenstaande stellingen dan geldig voor eventueel-lege symboolrijen. We gaan dat niet na.

We laten van nu af het gebruik van de * voor concatenatie varen en schrijven ab in plaats van a * b.

§ 5. Propositielogica als formeel systeem

Tot nu toe hebben we één formeel systeem beschouwd. In de toekomst zullen we ook andere behandelen. Bestanddelen waren: keuze van symbolen, overgang op symboolrijen, keuze van sommige symboolrijen als formules. In § 2 zijn voorbeelden van andere formele systemen genoemd: algebra met cijfers, letters, operatiesymbolen en haakjes; propositielogica met letters voor uitspraken en verbindingssymbolen; predicatenlogica met predicaten (onderwerp en gezegde) en quantoren.

Bij elk volgend formeel systeem zouden we de hele opbouw, die in het voorafgaande gegeven is, moeten overdoen. Daarbij gaat veel op analoge wijze. Om dit te voorkomen voeren we een nieuwe abstractie in: willekeurig formeel systeem. Hierin beginnen we met een "willekeurige verzameling symbolen". Omdat symbolen onderscheidbare denkdingen moeten zijn, zal een dergelijke verzameling eindig moeten zijn. We doen een nieuw beroep op het menselijke

abstractievermogen, als we aannemen, dat we ons een zinvolle voorstelling kunnen vormen van een "willekeurig stel symbolen", dat niet een expliciet aangegeven stel is.

We behoren nu een groot gedeelte van de in het voorafgaande gegeven beschouwingen na te lopen om na te gaan wat daarvan ook voor willekeurige systemen goed is en welke wijzigingen daartoe moeten worden aangebracht. Wij zullen dat niet uitvoeren. We merken op, dat het betreft de invoering van symboolrijen, de gelijkheid met eigenschappen en de concatenatie met eigenschappen (ook die in § 4). Op enkele voor de hand liggende kleine wijzigingen na blijft alles geldig.

We voeren nog een notatie in. Als S de verzameling der symbolen is, dan stelt S^+ de verzameling der symboolrijen over S en S^* de verzameling der eventueel-lege symboolrijen over S voor.

Als nieuw voorbeeld van een formeel systeem kiezen we de propositielogica, waarvan we de intuïtieve betekenis bekend veronderstellen. In de propositielogica houden we ons bezig met mededelende volzinnen, die niet waar behoeven te zijn. Het moet echter wel zinvol zijn om naar de waarheid of onwaarheid ervan te vragen. Men gebruikt de volgende namen voor zulk een zin: volzin, bewering, uitspraak. Wij kiezen de naam propositie. In het engels zijn sentence, statement en proposition gebruikelijk. Proposities worden gekoppeld met logische connectieven. We gebruiken daarvoor de volgende tekens:

$\wedge$ voor en
$\vee$ voor of
$\rightarrow$ voor als ... dan
$\neg$ voor niet.

We gebruiken variabelen om een hele zin met één letter aan te geven; daarvoor gebruiken we latijnse letters a, b, ... Haakjes zijn nodig om verwarring te voorkomen: $(a \wedge b) \vee c$ moet kunnen worden onderscheiden van $a \wedge (b \vee c)$.

Intuïtief bestaat er wel verband tussen connectieven; zo betekenen $a \wedge b$ en $\neg(\neg a \vee \neg b)$ hetzelfde en eveneens $a \rightarrow b$ en $\neg a \vee b$. Hoe we dit "hetzelfde betekenen" formeel moeten preciseren zullen we binnenkort zien. Het blijkt, dat we bepaalde connectieven kunnen missen; aan de andere kant zijn ze wel geregeld in gebruik. Bij de opzet van het formele systeem willen we ons met niet te veel symbolen belasten. De overige zullen dan als symbolen in de metataal worden gedefinieerd.

Tenslotte bespreken we het uitsparen van haakjes. Hiermee zijn niet de uit de algebra bekende uitsparingsregels bedoeld, die berusten op voorrangsregels voor operaties, associatieve eigenschappen en dergelijke, maar een methode, die haakjes helemaal overbodig maakt.

Tot nu toe schreven we een connectief tussen de leden waar het bijhoort: $a \wedge b$ of $(a \to c) \wedge (b \to c)$. Als men het ervoor of erachter zet, zijn haakjes overbodig. Deze bewering moet natuurlijk nog waargemaakt worden.

Voorbeelden van voorplaatsing:

$(a \wedge b) \vee c$ wordt $\vee \wedge abc$,

$a \wedge (b \vee c)$ wordt $\wedge a \vee bc$,

$(a \to c) \wedge (b \to c)$ wordt $\wedge \to ac \to bc$.

In deze voorbeelden kan men ook makkelijk uit de rechtsgeplaatste formules de linksgeplaatste terugvinden.

Men noemt de notatie, waarbij de connectieven voorgeplaatst worden, prefixnotatie of poolse notatie (naar de uitvinder, de poolse logicus J. Lukasiewicz). De klassieke notatie met tussenplaatsing heet infixnotatie. Uiteraard kan men de connectieven ook achterplaatsen; in dat geval spreekt men van postfixnotatie (in het engels ook wel "reversed Polish" genoemd).

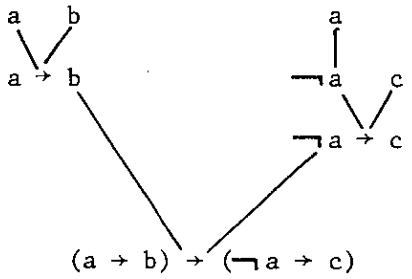
Wij zijn het meest aan infixnotatie gewend, niet alleen in de algebra, maar ook in de logica. Daarom zullen we ter bevordering van de leesbaarheid formules in prefixnotatie soms herschrijven in infixnotatie. Dit is op te vatten als een herbenoeming in de metataal.

De definitie van een formeel systeem der propositielogica zal voorlopig nog een onbepaaldheid bevatten in de keuze van de verzameling der propositievariabelen.

Symbolen zijn: een eindige verzameling propositievariabelen, $\neg$, $\to$. De symbolen $\neg$ en $\to$ zijn verschillend van alle propositievariabelen. We duiden propositievariabelen aan met kleine latijnse letters. Daarom zullen we, in afwijking van vroeger gebruik, symboolrijen met latijnse hoofdletters aanduiden. Voor een willekeurig symbool blijven we griekse letters gebruiken.

Definitie formule in propositielogica.
Iedere propositievariabele is een formule.
Als A een formule is, dan is ¬ A een formule.
Als A en B formules zijn, dan is → AB een formule.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd is een formule.

De vraag rijst of door deze recurrente definitie de opbouw van een formule uit eenvoudigere formules eenduidig bepaald is. Nu wordt bij → AB de opbouw van deze formule teruggebracht op die van A en B, waarbij de vraag of eerst A of eerst B wordt opgebouwd niet belangrijk is. Het gaat om de eenduidigheid van een boomstructuur. Voorbeeld in infixnotatie:



Als men deze constructie in een lijstje zet, kan dit op verschillende manieren. Twee voorbeelden:

a	a
b	¬a
a → b	c
a	¬a → c
¬a	b
c	a → b
¬a → c	(a → b) → (¬a → c)
(a → b) → (¬a → c)	

Een boom is een moeilijk hanteerbaar ding. Essentieel is echter alleen, dat bij iedere formule, die bij de constructie optreedt, vastligt hoe en uit welke andere formules zij gevormd is. Dit gaan we preciseren en aantonen. Daartoe voeren we het begrip designator in. Dit is een rij symboolrijen. Dit begrip rij zouden we in een nieuw formeel systeem kunnen formaliseren; we doen dit niet. De lengte van de rij is ≤ 3 en we schrijven de symboolrijen met komma's ertussen.

Definitie designator.
Iedere propositievariabele is een designator.
Als A een formule is, dan is $\neg$,A een designator.
Als A en B formules zijn, dan is $\rightarrow$,A,B een designator.

Definitie bijbehorende formule van een designator.
Als a een propositievariabele is, dan is a de bijbehorende formule van de designator a.
Van de designator $\neg$,A is $\neg$A de bijbehorende formule.
Van de designator $\rightarrow$,A,B is $\rightarrow$ AB de bijbehorende formule.

Stelling 5.1 (Designatorstelling). Bij iedere formule bestaat er één en slechts één designator, waarvan die formule de bijbehorende formule is.

Alvorens stelling 5.1 te bewijzen, bewijzen we eerst twee hulpstellingen.

Stelling 5.2. Als α een symbool is, X een eventueel-lege symboolrij en αX een formule, dan geldt één en slechts één van de volgende drie beweringen.

1. α is een propositievariabele, X = Λ.
2. $\alpha = \neg$, X is een formule.
3. $\alpha = \rightarrow$, er bestaan formules A en B, zodat X = AB.

Bewijs. Dat ten hoogste één van de drie beweringen kan gelden, ziet men aan α. Dat ten minste één van de drie beweringen geldt, volgt door recurrentie langs de opbouw van de formule αX met behulp van stelling 4.6.

Merk op dat we bij een bewijs door recurrentie langs de opbouw niet behoeven te weten of de opbouw eenduidig is; dit in contrast met een definitie langs de opbouw.

Stelling 5.3. Als A een formule is, X een eventueel-lege symboolrij en AX een formule, dan X = Λ.

Bewijs. We passen volledige inductie naar ℓ(A) toe. Op grond van stelling 5.2 zijn er drie mogelijkheden voor A.

1. Er is een propositievariabele a, zodat A = a, dus AX = aX. De formule aX bevindt zich in geval 1. van stelling 5.2, dus X = Λ.
2. Er is een formule B, zodat A = ¬B, dus AX = ¬BX. De formule ¬BX bevindt zich in geval 2. van stelling 5.2, dus BX is een formule. Omdat ℓ(B) < ℓ(A), mag op BX de inductieveronderstelling worden toegepast, op grond waarvan X = Λ.
3. Er zijn formules B en C, zodat A = → BC, dus AX = → BCX. De formule → BCX bevindt zich in geval 3. van stelling 5.2, dus er zijn formules D en E, zodat → BCX = → DE, BCX = DE. Op grond van stelling 4.8 bestaat er een eventueel-lege symboolrij R, zodat:
   hetzij D = BR, CX = RE; omdat ℓ(B) < ℓ(A), mag op BR de inductieveronderstelling worden toegepast, die oplevert, dat R = Λ, dus CX = E; omdat ℓ(C) < ℓ(A), mag de inductieveronderstelling ook op CX worden toegepast, hetgeen X = Λ oplevert;
   hetzij B = DR, E = RCX, R ≠ Λ; omdat ℓ(D) < ℓ(A), mag op DR de inductieveronderstelling worden toegepast, hetgeen R = Λ oplevert, wat een contradictie is.

Bewijs van stelling 5.1. We stellen eerst vast, dat van de drie soorten designatoren a en ¬, A en →, A, B een gegeven formule alleen bij twee designatoren van dezelfde soort kan behoren, omdat het voorste symbool van de bijbehorende formule in de drie gevallen resp. een propositievariabele, ¬, → is. Op grond van stelling 5.2 zijn er voor een formule A de volgende drie mogelijkheden.

1. Er is een propositievariabele a, zodat A = a. Dan behoort A bij de designator a en kan bij geen andere designator behoren.
2. Er is een formule B, zodat A = ¬B. Dan behoort A bij de designator ¬,B. Stel dat A ook bij de designator ¬,B' behoort, dan ¬B = ¬B', B = B' en de twee designatoren zijn gelijk.
3. Er zijn formules B en C, zodat A = → BC. Dan behoort A bij de designator →, B, C. Stel dat A ook behoort bij de designator →, B', C', dan → BC = = → B'C', BC = B'C'. Op grond van stelling 4.8 bestaat er een eventueel-lege symboolrij R, zodat:
   hetzij B = B'R, C' = RC; op grond van stelling 5.3 volgt uit B = B'R, dat R = Λ, B = B', C = C' en beide designatoren zijn gelijk;
   hetzij B' = BR, C = RC'; analoog als het vorige geval.

Stelling 5.1 rechtvaardigt het weglaten van haakjes in de prefixnotatie. Ondanks het ontbreken van haakjes is de opbouw van formules eenduidig vastgelegd.

De onbepaaldheid van een propositielogica is een onbevredigend aspect. Voor de toepassingen is het ongewenst een bovengrens voor het aantal propositievariabelen aan te nemen, zodat steeds behoefte aan een nieuwe propositielogica kan ontstaan. We zouden graag over oneindig veel propositievariabelen kunnen beschikken. De overgang op oneindig veel symbolen in een formeel systeem is echter ontoelaatbaar. De oplossing is, voor de propositievariabelen symboolrijen te kiezen. Intuïtief zouden we ze in een genummerde rij zetten: $p_0, p_1, p_2, \ldots$. Nemen we O en | in ons formele systeem op, dan is dit direct te vertalen tot pO, pO|, pO||, ... Daar O bij de natuurlijke getallen echter alleen als signaal diende en die rol nu door p kan worden overgenomen, kunnen we O ook weglaten.

We definiëren nu de propositielogica. Symbolen zijn: p, |, ¬, →.

Definitie propositievariabele.
p is een propositievariabele.
Als a een propositievariabele is, dan is a| een propositievariabele.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een propositievariabele.

De definitie van formule kan letterlijk worden herhaald. We zullen de verdere ontwikkeling van de propositielogica niet herhalen. De designatorstelling geldt ook hier, maar in het bewijs treden wel enkele extra complicaties op!

Tenslotte voeren we nog $\vee$ en $\wedge$ als symbolen in de metataal in door:

$\vee AB := \rightarrow\neg AB$

$\wedge AB := \neg\rightarrow A\neg B$.

§ 6. Uitgebreide symbolen.

De in § 5 ingevoerde mogelijkheid om propositievariabelen als symboolrijen op te vatten en niet als symbolen en er daarna toch mee te werken alsof het symbolen zijn levert een methode die ook in andere situaties nuttig is. We willen deze kwestie daarom nu in het algemeen beschouwen.

We gaan daarbij uit van een verzameling S van symbolen en van een verzameling E van symboolrijen. De elementen van E zullen de rol van symbolen in uitgebreide zin moeten gaan vervullen. Daartoe zullen we symboolrijen vormen door concatenatie van elementen van E. Om hiermee goed te kunnen werken zullen we de verkregen symboolrij echter weer op ondubbelzinnige wijze in zijn tot E behorende bestanddelen moeten kunnen splitsen. Zo zal het niet toelaatbaar zijn om || en ||| in E op te nemen, omdat dan |||||| zowel in ||·||·|| als in |||·||| gesplitst zou kunnen worden.

Definitie E^*.

$\Lambda \in E^*$.

Als $A \in E^*$ en $a \in E$, dan $Aa \in E^*$.

Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een element van E^*.

Definitie E^+.

$E^+ := E^* \setminus \{\Lambda\}$.

Definitie uitgebreide symbolen.

Een verzameling E van symboolrijen heet een verzameling uitgebreide symbolen als geldt:

als $A \in E^*$, $B \in E^*$, $a \in E$, $b \in E$, en $Aa = Bb$, dan $A = B$ en $a = b$.

Definitie uitgebreide symboolrijen.
Als E een verzameling uitgebreide symbolen is, dan heten de elementen van E^+ uitgebreide symboolrijen en de elementen van E^* eventueel-lege uitgebreide symboolrijen.

Deze definities komen niet in conflict met de vroeger gegeven definities van S^+ en S^*, omdat, bij de keuze E := S, de nieuwe definities van E^+ en E^* en de oude definities van S^+ en S^* hetzelfde resultaat opleveren.

Op grond van de voorwaarde waaraan E voldoet, is het geoorloofd om definities door recurrentie langs de opbouw van elementen van E^* of E^+ te geven. Dat dit inderdaad het geval is, zullen we niet in detail nagaan; er kan op eenvoudige wijze een designatorstelling (analoog stelling 5.1) worden bewezen.

Zo kunnen we ook een lengte van elementen van E^* invoeren, die wij, ter onderscheiding van de lengte van symboolrijen, met ℓ' zullen aanduiden.

Definitie lengte van een element van E^*.
Als $A \in E^*$ en $a \in E$, dan

$$\ell'(\Lambda) := 0 ,$$

$$\ell'(Aa) := \ell'(A) + 1 .$$

Hiervoor gelden stellingen analoog met de stellingen 4.4 t/m 4.8. We geven de formuleringen van de stellingen, maar slaan de bewijzen over, omdat ze analoog zijn met de vroegere bewijzen. Verder nemen we ook de lege symboolrij in de formuleringen op.

Stelling 6.1. Voor $A \in E^*$, $B \in E^*$ en $C \in E^+$ geldt

$$\ell'(AB) = \ell'(A) + \ell'(B) ,$$

$$\ell'(C) > 0 .$$

Stelling 6.2. Voor $A \in E^+$ geldt:
hetzij er bestaat een $a \in E$, zodat $A = a$ en $\ell'(A) = 1$,
hetzij er bestaat een $B \in E^+$ en een $a \in E$, zodat $A = Ba$ en $\ell'(A) > 1$.

Stelling 6.3. Als $A \in E^*$, k een natuurlijk getal en $0 \leq k \leq \ell'(A)$, dan bestaan er eenduidig bepaalde $B \in E^*$ en $C \in E^*$, zodat $A = BC$ en $\ell'(B) = k$.

Stelling 6.4. Voor $A \in E^*$, $B \in E^*$ en $C \in E^*$ geldt:
als $CA = CB$, dan $A = B$,
als $AC = BC$, dan $A = B$.

Stelling 6.5. Voor $A \in E^*$, $B \in E^*$, $C \in E^*$ en $D \in E^*$ geldt:
als $AB = CD$, dan bestaat er een $P \in E^*$, zodat
hetzij $A = CP$, $D = PB$,
hetzij $C = AP$, $B = PD$.

Deze stellingen preciseren de vage bewering, dat er met uitgebreide symbolen en symboolrijen kan worden gerekend alsof het gewone symbolen en symboolrijen zijn. Bovendien zijn definities door recurrentie langs de opbouw van E^+ (of E^*) voor een verzameling uitgebreide symbolen E geoorloofd. De verzameling E behoeft echter niet eindig te zijn. Dat het begrip "uitgebreid symbool" inderdaad een uitbreiding is van het begrip "symbool" volgt uit het feit dat de keuze $E := S$ een verzameling uitgebreide symbolen levert (zie de stellingen 2.2 en 2.3). Hetgeen men in vele leerboeken over logica symbolen noemt, zijn eigenlijk uitgebreide symbolen.

Bij de definitie van de propositielogica (eind § 5) kunnen we nu de propositievariabelen tezamen met $\neg$, $\rightarrow$ als verzameling uitgebreide symbolen nemen en vervolgens analoog te werk gaan als bij de eerder gegeven opbouw van een propositielogica. Het bewijs van de designatorstelling blijft ongewijzigd, mits overal symbolen en symboolrijen door uitgebreide symbolen en symboolrijen worden vervangen. Uiteraard moet wel worden aangetoond, dat propositievariabelen plus $\neg$ en $\rightarrow$ aan de voorwaarden van een verzameling uitgebreide symbolen voldoen. We doen dit in een iets algemenere situatie.

De propositievariabelen zouden we genummerde symbolen kunnen noemen. Voor het nummeren hebben we het symbool | gebruikt. In een algemenere situatie willen we echter de mogelijkheid niet uitsluiten, dat | reeds tot de gegeven symbolen behoort en voeren we daarom voor de nummering een nieuw symbool in.

Laat S een verzameling symbolen zijn en $\lfloor$ een symbool, dat niet tot S behoort, genaamd nummeringssymbool. Als $a \in S^+$, voeren we het begrip genummerde a in.

Definitie genummerde a.
a is een genummerde a.
Als A een genummerde a is, dan is A$\lfloor$ een genummerde a.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een genummerde a.

Laat nu een bij S behorende verzameling E van uitgebreide symbolen zijn gegeven, benevens een deelverzameling F van E. Laat nu E_1 de bij $S \cup \{\lfloor\}$ behorende verzameling symboolrijen zijn, bestaande uit alle genummerde a voor $a \in F$ en alle a in $E \setminus F$. We willen aantonen, dat E_1 een verzameling uitgebreide symbolen is.

Stelling 6.6. Als A een genummerde a is, dan is er een $A' \in \{\lfloor\}^*$, zodat $A = aA'$.

Bewijs door recurrentie naar de opbouw van de genummerde a.

We voeren een afbeelding $d: (S \cup \{\lfloor\})^* \to S^*$ in door:

Definitie d.
$d(\Lambda) = \Lambda$,
als $\alpha \in S$, dan $d(\alpha) = \alpha$,
$d(\lfloor) = \Lambda$,
als $A \in (S \cup \{\lfloor\})^*$ en $\alpha \in S \cup \{\lfloor\}$, dan $d(A\alpha) = d(A)d(\alpha)$.

Stelling 6.7. Als $A \in (S \cup \{\lfloor\})^*$ en $B \in (S \cup \{\lfloor\})^*$, dan $d(AB) = d(A)d(B)$.
Als $a \in S^*$, dan $d(a) = a$.
Als A een genummerde a is, dan $d(A) = a$.
Als $A \in E_1$, dan $d(A) \in E$.
Als $A \in E_1^+$, dan $d(A) \in E^+$.
Als $A \in (S \cup \{\lfloor\})^*$ en $d(A) = \Lambda$, dan $A \in \{\lfloor\}^*$.

Het bewijs van stelling 6.7 slaan we over.

Stelling 6.8. E_1 is een verzameling uitgebreide symbolen.

Bewijs. Stel $A \in E_1^*$, $B \in E_1^*$, $a \in E_1$, $b \in E_1$, $Aa = Bb$. Op grond van stelling 6.7 geldt $d(A)d(a) = d(Aa) = d(Bb) = d(B)d(b)$, $d(A) \in E^*$, $d(a) \in E$, $d(B) \in E^*$, $d(b) \in E$.
Omdat E een verzameling uitgebreide symbolen is, volgt hieruit $d(A) = d(B)$ en $d(a) = d(b)$.
Op grond van stelling 4.8 bestaat er een $R \in (S \cup \{\lfloor\})^*$, zodat hetzij $A = BR$, $b = Ra$; dan $d(b) = d(R)d(a)$, maar wegens $d(a) = d(b)$, $d(R) = \Lambda$; zou nu $R \neq \Lambda$ zijn, dan $R \in \{\lfloor\}^+$, dus op grond van stelling 4.6 is er een $R' \in \{\lfloor\}^*$, zodat $R = \lfloor R'$, $b = \lfloor R'a$, maar $b \in E_1$, dus is er een $c \in E$ en een $b' \in \{\lfloor\}^*$, zodat $b = cb'$ (stelling 6.6) en er is een $\alpha \in S$ en een $c' \in S^*$, zodat $c = \alpha c'$, dus $\alpha c'b' = b = \lfloor R'a$, dus $\alpha = \lfloor$ (stelling 4.6), hetgeen onmogelijk is omdat $\alpha \in S$ en $\lfloor \notin S$, dus $R = \Lambda$, $A = B$, $a = b$;
hetzij $B = AR$, $a = Rb$; dit leidt op analoge wijze tot $A = B$ en $a = b$.

We kunnen dit toepassen op de propositielogica door $S := \{p, \neg, \rightarrow\}$ te kiezen en | als nummeringssymbool. Vervolgens $E := S$ en $F := \{p\}$. Dan bestaat E_1 uit de propositievariabelen plus $\neg$ en $\rightarrow$.

Een nog eenvoudigere toepassing vormen de natuurlijke getallen met $S := \{O\}$, | als nummeringssymbool, $E := \{O\}$, $F := \{O\}$. Dan is E_1 de verzameling der natuurlijke getallen.

We zullen later nog andere toepassingen van uitgebreide symbolen behandelen.

§ 7. Intuïtieve behandeling van beslisbaarheid en opsombaarheid.

We komen terug op het begrip willekeurig formeel systeem en stellen vast, dat we nog niet hebben verklaard, hoe we daarin aan de verzameling der formules komen. In de voorbeelden waren dit oneindige verzamelingen, zodat ze niet door opsomming te definiëren zijn. Ze zijn in de voorbeelden door recurrente processen gedefinieerd, maar het is niet duidelijk wat een willekeurig recurrent proces is. We zien wel in de voorbeelden, dat het bestaat uit een opsomming van uitgangsformules, uit formatieregels, die uit formules andere formules maken, en uit een beperkende slotclausule. Dit is moeilijk algemeen vast te leggen; bovendien willen we ook niet vastleggen, dat de verzameling formules door een recurrent proces moeten worden gedefinieerd.

Het is redelijk om te eisen, dat we voor een willekeurige symboolrij effectief moeten kunnen uitmaken of het een formule is of niet. Er moet een procedure zijn, waarmee dat voor iedere symboolrij kan worden vastgesteld. We drukken dit uit door te zeggen, dat de verzameling der formules beslisbaar moet zijn; hetzelfde geldt voor een verzameling uitgebreide symbolen.

Wat in het algemeen een effectieve procedure is, is niet duidelijk. In concrete gevallen zal het misschien weinig moeite kosten gegeven procedures als effectief te aanvaarden; daarmee is de grens van wat als zodanig kan worden erkend niet bepaald. Op het preciseren van het begrip beslisbaar komen we in hoofdstuk II terug. We zullen er echter voordien in intuïtieve zin al mee werken. We merken wel al op, dat het feit dat een verzameling door een recurrent proces wordt gedefinieerd, niet garandeert, dat zij beslisbaar is. Deze bewering kan uiteraard pas na een precisering van de begrippen recurrent proces en beslisbaar bewezen worden.

Van een oneindige verzameling kan men niet alle elementen opschrijven. Men kan echter wel een lijst van die elementen gaan maken, die, als de verzameling oneindig is, nooit klaar komt. Als er een effectieve procedure bestaat, die een oneindig voortlopende lijst maakt van elementen van de verzameling, waarin ieder element van die verzameling op den duur aan de beurt komt, heet de verzameling opsombaar.

Een verzameling, die met een recurrent proces wordt gedefinieerd, is opsombaar. Immers, schrijf eerst de uitgangsformules in een eindige lijst; we nemen aan dat dat mogelijk is. Pas daarna de formatieregels voor het proces op alle mogelijke manieren op de uitgangsformules toe. Dit levert een eindige lijst formules op, die we bij de oorspronkelijke lijst voegen. Pas op de zo verkregen totale eindige lijst weer op alle mogelijke manieren de formatieregels toe en voeg de verkregen formules aan de lijst toe en herhaal dit proces onbepaald vaak. Dit levert een opsommingslijst van de verzameling, omdat iedere formule uit de verzameling ontstaat door geïtereerd een eindig aantal malen formatieregels toe te passen met uitgangsformules als uitgangspunt.

Een beslisbare verzameling is een opsombare verzameling. Dit is makkelijk in te zien, als we bedenken, dat de verzameling van alle symboolrijen opsombaar is. Als we daarvan een lijst maken, gaan we die lijst nalopen en stuk voor stuk onderzoeken, of de elementen van die lijst tot de verzameling behoren, hetgeen moet kunnen, omdat de verzameling beslisbaar is. Is het antwoord ja, dan nemen we de symboolrij op in de lijst voor de verzameling, is het antwoord nee, dan niet.

Later zullen we na precisering der begrippen bewijzen, dat een opsombare verzameling niet beslisbaar hoeft te zijn. Als de volgende nevenconditie vervuld is, is dat echter wel het geval.

Er is een effectieve procedure, die voor ieder natuurlijk getal n een plaats in de lijst aanwijst, zodat alle symboolrijen in de lijst voorbij die plaats lengte > n hebben.

Om aan te tonen, dat een opsombare verzameling, waarvan de opsommingslijst aan deze conditie voldoet, beslisbaar is, nemen we een willekeurige symboolrij a en bepalen de plaats in de lijst die volgens de nevenconditie bij $\ell(a)$ hoort. We controleren alle symboolrijen in de lijst tot aan die plaats, of ze gelijk zijn aan a. Als er één bij is, dan behoort a tot de verzameling; als dat echter niet zo is, behoort a niet tot de verzameling, want voorbij die plaats hebben alle symboolrijen in de lijst een lengte $> \ell(a)$ en zijn dus niet gelijk aan a.

Het is nu zo, dat alle recurrente processen, die we tot nu toe in voorbeelden hebben ontmoet, omgezet kunnen worden in opsommingslijsten, die voorzien kunnen worden van een effectieve procedure als boven beschreven. De

door deze recurrente processen gedefinieerde verzamelingen zijn dus alle beslisbaar, in het bijzonder de verzamelingen der formules in de formele systemen.

§ 8. Waarheid. Syntaxis en semantiek

In de intuïtieve toelichting hebben we al opgemerkt, dat proposities de eigenschap hebben waar of onwaar te zijn. Men gebruikt vaak de afkortingen t (true) en f (false); wij zullen 1 (waar) en 0 (onwaar) gebruiken. Van een propositievariabele is de betekenis onbepaald: zij kan waar of onwaar zijn. Bij het verbinden van proposities met connectieven wordt de waarheidswaarde van de samengestelde propositie echter vastgelegd door die van haar bestanddelen. Dit wordt uitgedrukt in de volgende waarheidstafels:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\rightarrow$	0	1
0	1	1
1	0	1

$\neg$	0	1
	1	0

In de tafels lezen we af, dat als A waarde 1 heeft en B waarde 0, dan heeft $A \rightarrow B$ waarde 0, $B \rightarrow A$ waarde 1, $A \vee B$ waarde 1, $A \wedge B$ waarde 0, $\neg A$ waarde 0.

Op deze wijze kunnen ook aan ingewikkelde formules waarheidswaarden worden toegekend, zodra de waarheidswaarden der propositievariabelen bekend zijn.

Een formele precisering vinden we in het begrip waardering of valuatie v, die de verzameling der formules afbeeldt in de verzameling {0,1}. Gegeven moet zijn v(a) voor alle propositievariabelen a; dan is de valuatie geheel vastgelegd.

Definitie valuatie.
Voor alle propositievariabelen a is v(a) gegeven; waarde 0 of 1.
$v(\neg A) = 1 - v(A)$.
$v(\rightarrow AB) = 1 - v(A) + v(A)v(B)$.

Deze definitie stemt overeen met de hierboven gegeven waarheidstafels.

Dit is een definitie door recurrentie langs de opbouw van een formule. Zulk een definitie is toegestaan op grond van de designatorstelling (5.1). In de definitie zijn alleen de waarheidstafels van $\rightarrow$ en $\neg$ gebruikt. Gebruikt men de definities van $\wedge$ en $\vee$, zoals gegeven in § 5, dan blijken deze te kloppen met bovenstaande waarheidstafels.

Er zijn formules, zoals $a \rightarrow (b \rightarrow a)$ (a en b propositievariabelen) die bij iedere valuatie de waarde 1 aannemen. Ook $A \rightarrow (B \rightarrow A)$ (A en B formules) is zulk een formule. Men spreekt dan van een tautologie.

Definitie tautologie.
Een formule uit de propositielogica heet een tautologie, als zij bij iedere valuatie de waarde 1 heeft.

Definitie semantisch equivalent.
De formules A en B uit de propositielogica heten semantisch equivalent als $v(A) = v(B)$ voor alle valuaties v.

Zo zijn $a \rightarrow (b \rightarrow c)$ en $\neg (a \rightarrow \neg b) \rightarrow c$ semantisch equivalent. Semantische equivalentie is een equivalentierelatie. Op grond daarvan kan de verzameling der formules in klassen van onderling semantisch equivalente worden verdeeld. De tautologieën vormen een klasse in deze verdeling. Formules die semantisch equivalent zijn, hebben dezelfde logische betekenis. Dit preciseert de in de intuïtieve inleiding gemaakte opmerking, dat sommige formules hetzelfde betekenen.

In het woord semantisch zit opgesloten, dat de beschouwing van waarheid en onwaarheid van formules behoort tot het gedeelte van de logica, dat we semantiek noemen. Het gedeelte van de logica, dat zich met de formele opbouw van formules en hun onderlinge samenhang bezighoudt, heet syntaxis. Als we aan formules betekenissen buiten het formele systeem toekennen, komen we op het terrein van de semantiek. Formule is een begrip uit de syntaxis, waarheid is een begrip uit de semantiek.

Omdat er oneindig veel propositievariabelen zijn, zijn er ook oneindig veel valuaties. In een gegeven formule komen echter slechts eindig veel propositievariabelen voor en alleen de waarheidswaarden, die aan de werkelijk in de formule voorkomende propositievariabelen worden toegekend, zijn relevant, als men bijv. wil nagaan of de formule een tautologie is of niet. Als

in de formule n propositievariabelen voorkomen, moeten er 2^n gevallen gecontroleerd worden, hetgeen effectief uitvoerbaar is. Hieruit volgt:

De verzameling der tautologieën is beslisbaar.

Hoewel hierdoor de verzameling der tautologieën effectief vastgelegd is, is ze nog niet erg overzichtelijk en heeft men andere wegen gezocht om ze voort te brengen.

§ 9. Axioma's en afleidingsregels. Zwakke volledigheid

We zullen de tautologieën van de propositielogica voortbrengen op een manier, die wat op een recurrent proces lijkt. Uitgangspunt zal een overzichtelijke collectie tautologieën moeten zijn. Verder moeten er één of enkele formatieregels zijn, die uit tautologieën andere tautologieën maken. De bedoeling is dan, dat op deze wijze alle tautologieën worden verkregen.

Een voorbeeld van een formatieregel is modus ponens. Uit de formules A en $A \to B$ wordt tot B geconcludeerd, geheel in overeenstemming met de in de wiskunde gebruikelijke redeneerwijze. Het is de gewoonte een dergelijke regel als volgt te noteren:

$$\frac{A,\ A \to B}{B}\ .$$

De formules, die in zulk een regel boven de streep staan, heten de premissen en de formule, die onder de streep staat, de conclusie van de regel. De formules, die als uitgangspunt worden gekozen, heten axioma's.

Neemt men een eindige verzameling axioma's, dan komt men er met modus ponens alleen als regel niet. Maar men kan de axioma's makkelijk uitbreiden. Neemt men de tautologie $a \to (b \to a)$, dan is ook $A \to (B \to A)$ een tautologie. Als men werkelijk één formule wil opschrijven, is $p_0 \to (p_1 \to p_0)$ correcter, of helemaal formeel $\to p \to p|p$.

$A \to (B \to A)$ is een "formule" in de metataal, met A en B als variabelen in de metataal, die formules voorstellen. Pas na invullen van echte formules voor A en B ontstaat een formule; bijv. voor A: p_2 en voor B: $p_1 \to p_0$, dan komt er

$$p_2 \to ((p_1 \to p_0) \to p_2) \quad \text{(formeel: } \to p|| \to \to p|pp||) \ .$$

We noemen een "formule" van de gedaante $A \to (B \to A)$ een axiomaschema. Een axiomaschema staat voor oneindig veel axioma's. Formatieregels noemen we afleidingsregels. Toevoeging van axioma's en afleidingsregels aan een formeel systeem maakt dit tot een deductief systeem. In § 10 zullen we deze begrippen nader preciseren.

In de keuze van axioma's en afleidingsregels voor de tautologieën van de propositielogica is nog veel vrijheid. In leerboeken over mathematische logica wordt dit onderwerp uitvoerig behandeld. Voor ons is het principe wel belangrijk maar de technische uitwerking veel minder.

We eisen uiteraard, dat de verzameling der axioma's beslisbaar is. Heeft men een eindig aantal axiomaschema's, dan is dat wel zo. Ook de verzameling der afleidingsregels moet beslisbaar zijn. Daarmee is bedoeld, dat, als men een rijtje formules met daarbij een formule als conclusie opschrijft, het uit te maken moet zijn, of dat een legitieme toepassing van een der afleidingsregels voorstelt. Een afleidingsregel als modus ponens is namelijk ook een schema, omdat er variabelen voor formules in voorkomen. Het zou dan ook correcter, zij het ongebruikelijk, zijn om van een schema van een afleidingsregel te spreken en de regels, die ontstaan door voor de variabelen concrete formules in te vullen, afleidingsregels te noemen.

We vermelden drie voorbeelden van deductieve systemen voor de tautologieën van de propositielogica.

I Axioma's: alle tautologieën.
Afleidingsregels: geen.

II Axioma's (P, Q, R zijn variabelen voor formules):
$P \to (Q \to P)$,
$(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$,
$(\neg P \to \neg Q) \to (Q \to P)$.
Afleidingsregels: modus ponens.

III Axioma's: $p_0 \to (p_1 \to p_0)$,
$(p_0 \to (p_1 \to p_2)) \to ((p_0 \to p_1) \to (p_0 \to p_2))$,
$(\neg p_0 \to \neg p_1) \to (p_1 \to p_0)$.
Afleidingsregels: modus ponens,
substitutieregel.

Geval I is flauw en voldoet niet aan de praktische eisen van overzichtelijkheid, die we aan een deductief systeem hebben gesteld. Geval II is een klassiek stelsel met een eindig aantal axiomaschema's en modus ponens als regel. Geval III heeft slechts eindig veel axioma's. In ruil daarvoor hebben we de regels met een substitutieregel moeten uitbreiden. We zullen geen exacte formulering van deze regel geven, maar volstaan met de opmerking dat toepassing van deze regel erop neerkomt, dat men in een formule voor de erin voorkomende propositievariabelen formules substitueert.

Voorbeeld: $a \to (a \to b)$ gaat door de substitutie a wordt $c \to b$, b wordt a over in $(c \to b) \to ((c \to b) \to a)$.

De vraag is nu of afleidbare formules inderdaad tautologieën zijn. Daartoe moet nagegaan worden, of de axioma's tautologieën zijn, hetgeen eenvoudig is. Voor modus ponens bekijken we de mogelijke combinaties van v(A), v(B) en $v(A \to B)$:

A	B	$A \to B$
0	0	1
0	1	1
1	0	0
1	1	1

Als nu A en $A \to B$ tautologieën zijn, komen we bij iedere valuatie in de laatste regel terecht, dus ook B is een tautologie.

Ook bij toepassing van de substitutieregel blijken tautologieën in tautologieën over te gaan.

De omgekeerde vraag is nu of iedere tautologie afleidbaar is. Voor I is dat triviaal. Voor II en III is het waar, maar niet eenvoudig te bewijzen. We behandelen dat niet en verwijzen naar leerboeken.

De stelling dat iedere tautologie afleidbaar is, heet de zwakke volledigheidsstelling. Het is een bewering over het gekozen deductieve systeem.

Als de verzameling axioma's oneindig is, valt het stelsel van axioma's en afleidingsregels niet onder het begrip recurrent proces, zoals dat in § 7 is beschreven. Toch kan men ook voor een deductief systeem aantonen, dat de verzameling der afleidbare formules opsombaar is. In de hierboven gegeven voorbeelden, waarvoor de zwakke volledigheidsstelling geldt, is dat niet van

belang, omdat we al weten, dat de verzameling der tautologieën beslisbaar is. Voor andere deductieve systemen kan het echter wel degelijk belangrijk zijn.

We geven nog een voorbeeld van een afleiding in stelsel II voor de propositielogica en wel van $P \to P$. Voor het gemak nummeren we de axiomaschema's met: ax 0, ax 1 en ax 2.

ax 0	$Q := P \to P$	$P \to ((P \to P) \to P)$	0
ax 1	$Q := P \to P$	$(P \to ((P \to P) \to P)) \to ((P \to (P \to P)) \to (P \to P))$	1
	$R := P$		
modus ponens 0,1		$(P \to (P \to P)) \to (P \to P)$	2
ax 0	$Q := P$	$P \to (P \to P)$	3
modus ponens 3,2		$P \to P$	4

We merken allereerst op, dat dit eigenlijk een schema van een afleiding is; hetgeen afgeleid moet worden $(P \to P)$ is trouwens een schema van een formule. De afleiding is een lijst van formules, die in regels opgeschreven is; voor het gemak hebben we rechts deze regels genummerd. Bij het opschrijven van zo'n lijst mogen we in een regel altijd een axioma invullen, maar ook de conclusie van een afleidingsregel, mits alle premissen als eerdere regels in de lijst voorkomen. Links is aangegeven, welk axioma gebruikt is met daarachter eventueel nog in het axiomaschema aangebrachte substituties, dan wel welke regel is toegepast en in welke voorgaande regels de premissen van die regel staan.

Voordat we de discussie van deductieve systemen der propositielogica voortzetten behandelen we het begrip deductief systeem in het algemeen.

§ 10. Deductieve systemen

Een afleiding hebben we in § 9 beschreven als een lijst van formules. Om zulk een lijst in een formeel systeem te interpreteren zou men de formules achter elkaar in plaats van onder elkaar moeten schrijven. Schrijft men ze gewoon achter elkaar, dan ontstaat er concatenatie en dreigt het gevaar, dat we de afzonderlijke formules niet meer kunnen terugvinden. Weliswaar is dat bij formules uit de propositielogica wel mogelijk, maar in andere geval-

len niet. We ondervangen dit bezwaar door het tussenplaatsen van een scheidingssymbool. In de taal wordt daarvoor vaak een komma gebruikt. Wij geven de voorkeur aan een nieuw symbool, dat nergens anders voor wordt gebruikt.

In afleidingsregels hadden we formules boven en onder de streep. We kunnen ze zonder bezwaar in een lijst achter elkaar zetten; de laatste formule is de conclusie. Een axioma is op te vatten als een afleidingsregel zonder premissen; dit klopt met het gebruik dat we van axioma's in een afleiding maken. Deze afspraak verklaart waarom we niet zoveel waarde hechten aan een eindige verzameling axioma's en waarom we ook bij afleidingsregels van een schema willen spreken.

We gaan uit van een formeel systeem F met een, hier niet gespecificeerde, verzameling symbolen en een beslisbare verzameling W van formules. We maken een nieuw formeel systeem F' met als symbolen de symbolen van F plus één nieuw symbool, dat scheidingssymbool heet. We schrijven Γ voor het scheidingssymbool. We definiëren het begrip lijst van formules; een lijst van formules is een symboolrij in F' en de formules uit de lijst zijn formules in F. We vatten daartoe de formules gevolgd door een Γ op als uitgebreide symbolen.

Definitie E.
E bestaat uit de symboolrijen AΓ met $A \in W$.

Definitie lijst van formules.
Een lijst van formules is een element van E^+.

Stelling 10.1. E is een verzameling uitgebreide symbolen.

Het bewijs van stelling 10.1 slaan we over.

De verzameling van lijsten van formules is beslisbaar. We nemen nu aan, dat er een beslisbare verzameling van lijsten van formules gegeven is, waarvan de elementen afleidingsregels heten. Een afleidingsregel van de gedaante AΓ met $A \in W$ heet een axioma.

Definitie voorkomen in, slotformule.
Als $A \in W$ en L een lijst van formules is, dan zeggen we dat A in L voorkomt, als er eventueel-lege lijsten van formules M en N bestaan, zodat $L = MA\Gamma N$.
Als N leeg is, dan heet A de slotformule van L.

Definitie conclusie, premisse.
Als L een afleidingsregel is en A een formule, dan heet A de conclusie van L, als A de slotformule van L is, en heet A een premisse van L als A in L voorkomt met een N uit de definitie van voorkomen in, die niet leeg is.

Let wel, dat in een afleidingsregel dezelfde formule A zowel premisse als conclusie kan zijn.

Voor het begrip afleiding geven we een voorlopige, voor de hand liggende definitie, die tot bezwaren blijkt te leiden. Om aan die bezwaren tegemoet te komen moet dan een andere, ingewikkeldere definitie worden gegeven.

Definitie compatibel.
Een lijst van formules L en een afleidingsregel S heten compatibel als alle premissen van S in L voorkomen.

Definitie afleiding (voorlopig).
Als $A \in W$ en $A\lceil$ een afleidingsregel is, dan is $A\lceil$ een afleiding.
Als L een afleiding is, S een afleidingsregel met conclusie A, en L en S zijn compatibel, dan is $LA\lceil$ een afleiding.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een afleiding.

Definitie afleidbare formule.
Als $A \in W$, dan heet A een afleidbare formule, als A slotformule van een afleiding is.

Het bezwaar van bovenstaande definitie is, dat de verzameling der afleidingen niet beslisbaar hoeft te zijn. We voeren op een voor de hand liggende manier het begrip lijstlengte van een lijst van formules in als de lengte ℓ' van de rij uitgebreide symbolen in E^+ (het is het aantal malen dat het scheidingssymbool in de lijst voorkomt). De lijstlengte hoeft in de verzameling der afleidingsregels niet begrensd te zijn; als dat wel het geval is, is de verzameling der afleidingen wel beslisbaar. Hoewel in praktische voorbeelden deze begrensdheid veelal vervuld is, willen we deze beperking toch niet opleggen, omdat door wijziging van de definitie van afleiding de verzameling der afleidingen in alle gevallen beslisbaar is, zonder dat het begrip afleidbare formule erdoor verandert. De grondgedachte is om bij het maken van de lijst, die de afleiding is, niet alleen de conclusie van de

gebruikte afleidingsregel op te schrijven, maar de hele afleidingsregel. Een afleiding wordt dan in plaats van een lijst van formules een lijst van afleidingsregels, dat is een lijst van lijsten van formules. Dit betekent dat het proces van het vormen van een lijst een keer geïtereerd moet worden, en dat we een tweede scheidingssymbool nodig hebben.

We geven de formele uitvoering van bovenstaande gedachte niet. In een afleiding mag telkens een nieuwe afleidingsregel bijgeschreven worden, als al zijn premissen al eerder in een afleidingsregel van de lijst hebben gestaan. Een afleidbare formule is de conclusie van de slotafleidingsregel van een afleiding.

Met deze definitie is de verzameling der afleidingen beslisbaar. De verzameling der afleidbare formules hoeft echter niet beslisbaar te zijn. Wel geldt:

De verzameling der afleidbare formules is opsombaar.

Zo is de verzameling der afleidbare formules van de predicatenlogica van de eerste orde (zie appendix) opsombaar. Deze verzameling is echter niet beslisbaar; dit is door Church in 1936 bewezen.

Het met afleidingsregels uitgebreide systeem F heet een deductief systeem. Bouwstenen zijn: symbolen, formules, afleidingsregels.

§ 11. Theorieën. Sterke volledigheid

Logische systemen worden onder meer gebruikt om mathematische theorieën te beschrijven. De propositielogica is echter te arm aan uitdrukkingsmiddelen om daarvan interessante voorbeelden te geven. De predicatenlogica van de eerste orde is daarvoor beter geschikt; formalisering daarvan is echter veel ingewikkelder dan van de propositielogica en kan daarom hier niet worden behandeld. In de appendix zal een beknopte behandeling van een formalisering van de predicatenlogica van de eerste orde worden gegeven.

We beschouwen een aantal concrete beweringen uit een wetenschapsgebied. Daartussen kan verband bestaan. Bij logische analyse is de inhoud van de beweringen niet in het geding; we duiden ze daarom aan met letters. Er komt dan zoiets als: $(a \wedge b) \rightarrow c$. Of dit juist is, hangt af van de interpretatie. We abstraheren nu van de inhoud en nemen beweringen als $(a \wedge b) \rightarrow c$ als axioma. Dit zijn andere axioma's dan die in § 9; het zijn geen tautologieën en de geldigheid hangt af van hetgeen voor de letters wordt ingevuld. We noemen ze niet-logische axioma's.

We nemen bijvoorbeeld $(a \wedge b) \rightarrow c$ en $(b \wedge c) \rightarrow d$ als axioma's en zeggen dat daar $a \rightarrow (\neg b \vee d)$ uit volgt. Om in te zien, wat dat betekent, vatten we a, b, c en d op als propositievariabelen, maar laten slechts die valuaties toe, waarvoor de axioma's waarde 1 hebben. Omdat $a \rightarrow (\neg b \vee d)$ voor al die valuaties ook waarde 1 heeft, is het een logische consequentie van de axioma's.

Formeel vertaalt men elke bewering, die bestanddeel van een axioma is, met een propositievariabele (d.w.z. men nummert ze). De axioma's gaan dan over in een verzameling T van formules uit de propositielogica (een theorie).

Definitie theorie.
Een beslisbare verzameling T van formules uit de propositielogica heet een theorie.

Definitie logische consequentie.
Als T een theorie is en A een formule, dan heet A een logische consequentie van T, als voor iedere valuatie v, waarvoor $v(X) = 1$ voor alle $X \in T$, $v(A) = 1$ (notatie: $T \models A$).

Naast het semantische begrip logische consequentie hebben we het syntactische begrip afleidbaar uit.

Wij zullen in het volgende terugkeren tot de gangbare terminologie, waarbij we afwijken van de in §10 ingevoerde benamingen. Zo zullen we de axioma's weer apart nemen en niet tellen als afleidingsregels zonder premisse. Verder zal een schema als modus ponens weer een afleidingsregel worden genoemd (in §10 was dit een verzameling afleidingsregels). Tenslotte zal een afleiding een lijst van formules zijn en niet een lijst van afleidingsregels.

Definitie afleidbaar.
Als T een theorie is en A een formule, dan heet A afleidbaar uit T als in het deductieve systeem, dat ontstaat door alle formules van T als axioma's toe te voegen, A afleidbaar is (notatie: $T \vdash A$).

Het begrip "afleidbaar uit" is afhankelijk van de keuze van het deductieve systeem voor de tautologieën van de propositielogica. We beginnen met een hulpstelling.

Stelling 10.1. Als T_0 en T_1 theorieën zijn, A een formule is, $T_1 \vdash A$ en $T_0 \vdash X$ voor alle $X \in T_1$, dan $T_0 \vdash A$.

Bewijsschets. In een afleiding van A uit T_1 vervangen we iedere voorkomende formule van T_1 door haar afleiding uit T_0. Dit levert een afleiding van A uit T_0. Men kan hiervan een formeel bewijs maken door achtereenvolgens de volgende beweringen aan te tonen.

1. Als L en M afleidingen zijn, dan is ook LM een afleiding.
2. Als T_0 een theorie en L een lijst van formules is, zodat voor alle formules X, die in L voorkomen, geldt $T_0 \vdash X$, dan is er een afleiding uit T_0, waarin alle formules van L voorkomen.
3. Als T_0 een theorie is, S een afleidingsregel met conclusie A en als voor alle premissen X van S geldt, dat $T_0 \vdash X$, dan $T_0 \vdash A$.
4. Als T_0 en T_1 theorieën zijn, $T_0 \vdash X$ voor alle $X \in T_1$ en L een afleiding uit T_1, dan geldt voor alle formules A, die in L voorkomen, dat $T_0 \vdash A$.

Als we voor T de lege verzameling nemen, schrijven we $\models A$ en $\dot{\vdash} A$. We krijgen dan eerder gedefinieerde begrippen terug:

$\models A$: A is een tautologie,

$\vdash A$: A is afleidbaar.

De vraag rijst of $T \models A$ en $T \vdash A$ op hetzelfde neerkomen. Het antwoord is afhankelijk van de keuze van het deductieve systeem, maar er zijn deductieve systemen, waarvoor het geldt. We geven nodige en voldoende voorwaarden voor een deductief systeem, opdat de gelijkwaardigheid van $T \models A$ en $T \vdash A$ geldt, maar beperken ons daarbij tot eindige T.

Stelling 11.2. Nodige en voldoende voorwaarden, opdat voor alle eindige theorieën T en alle formules A geldt, dat $T \models A$ dan en slechts dan als $T \vdash A$, zijn:

1. Alle logische axioma's zijn tautologieën.
2. Voor iedere afleidingsregel $\dfrac{A_0,\ldots,A_{n-1}}{B}$ en iedere valuatie v geldt, dat als $v(A_0) = \ldots = v(A_{n-1}) = 1$, dan $v(B) = 1$.
3. Alle tautologieën zijn afleidbaar.
4. Als A en B formules zijn, dan $\{A, A \to B\} \vdash B$.

Bewijs. We bewijzen eerst, dat de voorwaarden noodzakelijk zijn. We nemen dus de gelijkwaardigheid van $T \models A$ en $T \vdash A$ aan. Nu volgen 1 en 3 direct uit de keuze van de lege verzameling voor T. Voor 2 nemen we $T = \{A_0,\ldots,A_{n-1}\}$, dan is $T \vdash B$, dus $T \models B$, waaruit 2 volgt. Tenslotte volgt 4 uit $\{A, A \to B\} \models B$. Nu bewijzen we, dat de voorwaarden voldoende zijn, dus nemen we aan dat 1, 2, 3, 4 vervuld zijn.

Stel $T \vdash A$. Dan bestaat er een afleiding van A uit T. We bewijzen, dat iedere formule die in deze afleiding voorkomt een logische consequentie van T is. Laat v een valuatie zijn waarin alle formules van T de waarde 1 hebben; we moeten aantonen, dat iedere formule, die in de afleiding als conclusie voorkomt, de waarde 1 heeft. Dit bewijzen we door recurrentie langs de afleiding. Voor logische axioma's (zie 1) en formules van T klopt het. Als de formule conclusie is van een afleidingsregel, waarvan de premissen eerder in de afleiding voorkomen, dan hebben die premissen op grond van de recurrentieveronderstelling waarde 1. Op grond van 2 heeft de conclusie ook waarde 1. Daarmee is $T \models A$ bewezen.

Stel nu $T \models A$. Verondersteld is, dat T eindig is; we passen volledige inductie toe naar het aantal n van de elementen van T. Als $n = 0$, dan is A een tautologie en dus, op grond van 3., afleidbaar, dus $T \vdash A$. Stel, dat de te bewijzen bewering juist is voor theorieën met n elementen. Stel nu dat T $n + 1$ elementen heeft en $T \models A$. Nu is T niet leeg, dus we kunnen een $B \in T$ kiezen. Nu geldt $T\backslash\{B\} \models B \to A$. Immers, laat v een valuatie zijn, zodat voor alle $X \in T\backslash\{B\}$ geldt $v(X) = 1$. Als dan $v(B) = 1$, dan geldt $v(X) = 1$ voor alle $X \in T$, zodat $v(A) = 1$, dus $v(B \to A) = 1$. Als $v(B) = 0$, dan geldt ook $v(B \to A) = 1$. Omdat $T\backslash\{B\}$ n elementen heeft, levert de inductieveronderstelling, dat $T\backslash\{B\} \vdash B \to A$, dus zeker $T \vdash B \to A$. Verder uiteraard $T \vdash B$. Hieruit, uit 4. en stelling 11.1 volgt dan, dat $T \vdash A$.

In de formulering en het bewijs van stelling 11.2 hebben we in de metataal gebruik gemaakt van een in de wiskunde gebruikelijke notatie met ... voor eindige rijen, zoals $A_0,\ldots,A_{n-1}$ of $v(A_0) = \ldots = v(A_{n-1})$ of $\to A_0 \to A_1 \to \ldots \to A_{n-1}$ A. Dit is een korte en suggestieve notatie, die echter onderstelt, dat er een afbeelding tot stand gebracht is tussen een beginsegment van de verzameling der natuurlijke getallen en de geïndiceerde grootheden. Deze notatie wordt ook gebruikt als $n = 1$, waarbij A_0 en A_{n-1} samenvallen en zelfs als $n = 0$, waarbij er niets staat en $n - 1$ niet zinvol is. Bovendien wordt wordt wel eens een ander segment dan het beginsegment van N gebruikt, zoals in $\to A_k \to A_{k+1} \to \ldots \to A_{n-1}$ A.

Voor de in § 9 gegeven voorbeelden van deductieve systemen geldt het volgende:

I voldoet aan 1, 2, 3, niet aan 4.

II voldoet aan alle vier.

III voldoet aan 1, 3, 4, niet aan 2.

Dat 1 en 3 gelden volgt uit de zwakke volledigheidsstelling, die voor I, II en III geldt. Voor II en III hebben we dat niet bewezen. Dat 2 voor modus ponens geldt, is makkelijk na te gaan. Voor de substitutieregel geldt 2 echter niet. Voorbeeld: neem $a \to (a \to b)$ en een valuatie, waarvoor $v(a) = 0$, $v(b) = 1$. Dan is $v(a \to (a \to b)) = 1$. Substitueert men a wordt $\neg a$, b wordt $b \to a$, dan is $v(\neg a \to (\neg a \to (b \to a))) = 0$. Tenslotte geldt 4 voor II en III, omdat modus ponens daar een afleidingsregel is. Voor I geldt $T \vdash A$ dan en slechts dan als A een tautologie is of $A \in T$. Neemt men in 4 A wordt p_0, B wordt p_1, dan geldt 4 niet.

Van de drie is systeem II dus het enige waarvoor de vier voorwaarden van stelling 11.2 gelden. We geven nu nog een voorbeeld van een systeem, waarvoor dat het geval is:

IV Axioma's: alle tautologieën.
Afleidingsregels: modus ponens.

Van het standpunt van het afleiden van tautologieën is dit een dwaas systeem, omdat bij de gegeven axioma's de afleidingsregels overbodig zijn. Van het standpunt van het afleiden uit een theorie is dat niet het geval. Het is eenvoudig in te zien, dat het systeem aan de eisen 1, 2, 3, 4 voldoet, zodat we nu over een bewijs zonder lacunes beschikken, dat een dergelijk systeem inderdaad bestaat.

Stelling 11.3. Er bestaat een beslisbaar deductief systeem voor de propositielogica, zodat $T \models A$ dan en slechts dan als $T \vdash A$ voor alle eindige theorieën T en alle formules A geldt.

Als men in deze stelling het woord "eindige" weglaat krijgt men een stelling, die sterke volledigheidsstelling heet. Deze laatste stelling hebben we niet bewezen; het bewijs is moeilijk. Het blijkt, dat een systeem, dat voor eindige theorieën voldoet, ook voor oneindige theorieën bruikbaar is. We nemen nu verder aan, dat het deductieve systeem zo gekozen is, dat het aan de eisen 1, 2, 3, 4 voldoet.

Stelling 11.4. Als het deductieve systeem voldoet aan de eisen 1,2,3,4 van stelling 11.2, T een theorie is, A en B formules zijn en $T \cup \{A\} \vdash B$, dan $T \vdash A \to B$ (deductiestelling).

Bewijs. Beschouw een afleiding van B uit $T \cup \{A\}$; daarin komen maar eindig veel formules voor. Er is dus een eindige deelverzameling T' van T, zodat $T' \cup \{A\} \vdash B$. Op grond van stelling 11.2 geldt dan $T' \cup \{A\} \models B$. Daaruit volgt $T' \models A \to B$. Immers, laat v een valuatie zijn waarvoor alle formules van T' waarde 1 hebben. Als dan $v(A) = 0$, dan $v(A \to B) = 1$; als $v(A) = 1$, dan volgt uit $T' \cup \{A\} \models B$, dat $v(B) = 1$, dus $v(A \to B) = 1$. Uit $T' \models A \to B$ volgt met stelling 11.2, dat $T' \vdash A \to B$, dus a fortiori $T \vdash A \to B$.

We merken op, dat de omkering van stelling 11.4, dat $T \cup \{A\} \vdash B$ uit $T \vdash A \to B$ volgt, een triviaal gevolg is van modus ponens.

De volgende stelling volgt uit de sterke volledigheidsstelling:

Als T een theorie is, A een formule en $T \models A$, dan is er een eindige deelverzameling T' van T, waarvoor $T' \models A$ (compactheidsstelling).

Voor afleidbaarheid in plaats van logische consequentie volgt de stelling uit het feit, dat in een afleiding maar eindig veel formules voorkomen. Op grond van de sterke volledigheidsstelling komen afleidbaarheid en logische consequentie echter op hetzelfde neer.

HOOFDSTUK II BESLISBAARHEID

§ 1. Inleiding

In dit hoofdstuk geven we een precisering van de begrippen beslisbaar en opsombaar, waarover in hoofdstuk I al in intuïtieve zin gesproken is. De precisering van deze begrippen dateert van de jaren tussen 1930 en 1940, waarin deze langs een aantal verschillende wegen is geschied. Ook is toen bewezen, dat de verschillende definities equivalent zijn in dien zin dat ze dezelfde begrippen opleveren. Daar deze definities van mathematische aard zijn, kan deze equivalentie met wiskundige hulpmiddelen worden aangetoond.

De bewering, dat de op deze wijze mathematisch gepreciseerde begrippen overeenstemmen met de intuïtieve begrippen beslisbaar en opsombaar wordt gewoonlijk de these van Church genoemd, omdat Church dit in 1936 voor één van de definities gesteld heeft. De these van Church kan niet exact bewezen worden. Een poging om dit toch te doen kan slechts tot een vicieuze cirkel leiden. Om namelijk een exact bewijs te kunnen geven moeten de vage begrippen beslisbaar en opsombaar eerst gepreciseerd worden. Kiest men daarvoor een van de bestaande preciseringen, dan valt er niets te bewijzen, maar is men ook niets opgeschoten; kiest men een nieuwe precisering, dan kan men misschien daarvan wel bewijzen, dat zij equivalent is met een der bestaande preciseringen, maar verplaatst het probleem zich naar de vraag of de nieuwe precisering een getrouwe weergave van de intuïtieve begrippen geeft.

Hetgeen voor de these van Church pleit is de praktische bruikbaarheid: alle effectieve methoden, die men heeft kunnen verzinnen, zijn gebleken door de mathematische preciseringen te worden gedekt.

De verschillende methoden om de begrippen te definiëren vallen in drie categorieën uiteen.

1. Precisering van het begrip recurrent proces. Daartoe moet het verengd worden tot iets wat wel exact geformuleerd kan worden, maar toch algemeen genoeg is om er alle effectief uitvoerbare processen mee te beschrijven. Bekende voorbeelden zijn de producties van Post en de algoritmen van Markov.

2. Aansluiting bij methoden, die in de gewone wiskunde gebruikelijk zijn. Men beschouwt verzamelingen en functies van natuurlijke getallen. Via de karakteristieke functie kan een verzameling ook op een functie worden teruggebracht, die de waarden 0 en 1 aanneemt. Men behoeft zich echter niet tot zulke functies te beperken. Men wil tot een begrip "effectief berekenbare functie" komen. Dit wordt weer op een recurrente wijze gedaan. Men begint met enkele eenvoudige functies en heeft enkele processen (zoals bijvoorbeeld het vormen van samengestelde functies) om uit functies nieuwe functies te maken. Op deze wijze komt men tot de zogenaamde recursieve functies (Herbrand, Gödel, Kleene).

3. Hetgeen effectief uitvoerbaar is moet door een machine kunnen worden gedaan. Men moet echter wel een abstracte machine hebben, die alles kan wat door machines kan worden gedaan. Dit is verwezenlijkt in de Turing-machine.

Het lijkt efficiënt één van bovengenoemde methoden te behandelen en daarbij te blijven. Omdat ze equivalent zijn verliest men daar niets bij. Het is echter zo, dat hetgeen met de ene methode makkelijk gaat, bij de andere methode heel ingewikkeld kan zijn en omgekeerd; dit pleit ervoor niet strikt aan één methode vast te houden.

De recursieve functies sluiten het meest aan bij de wijze waarop wij wiskunde bedrijven. Er wordt met getallen gewerkt en allerlei vertrouwde functies en notaties kunnen worden gebruikt. Wil men echter een begrip als "afleidbaar in een logisch systeem" behandelen, dan leidt vertaling daarvan tot moeilijk hanteerbare getallenfuncties. De begrippen productie en algoritme liggen dan veel meer voor de hand.

Wij zullen met de Turing-machine beginnen. Als aanloop daartoe zullen we echter eerst eenvoudigere machines bespreken en inzien, dat deze niet adequaat zijn om het probleem der beslisbaarheid te behandelen, omdat verzamelingen, die duidelijk intuïtief beslisbaar zijn, ontoegankelijk zijn voor de machine. Methoden om dit te verhelpen zullen dan tot de Turing-machine leiden. Dat nog verdergaande complicering van de machine daarna niet meer leidt tot uitbreiding van hetgeen de machine kan, zullen we niet aantonen.

§ 2. Eindige machines

We beginnen met het beeld van een doos, waar iets kan worden ingestopt en iets kan worden uitgehaald. Hetgeen er ingestopt wordt heet de invoer (engels: input) en hetgeen er uitgehaald wordt heet de uitvoer (engels: output). De invoer kan bestaan uit symbolen uit de verzameling I van invoersymbolen en evenzo de uitvoer uit symbolen uit de verzameling O van uitvoersymbolen. Hetgeen zich binnen de doos afspeelt laten we zoveel mogelijk buiten beschouwing, hetgeen tot uitdrukking gebracht wordt door over een zwarte doos (engels: black box) te spreken. Met de invoer van een invoersymbool correspondeert de uitvoer van een uitvoersymbool. Hierbij wordt het begrip symbool evenwel in een wat ruimere betekenis gebruikt, dan we tot nu toe gewend waren. Zo mogen de elementen van O wel symboolrijen zijn. Wel worden de verzamelingen I en O beide eindig verondersteld. Verder is de machine deterministisch: onder dezelfde omstandigheden is het gedrag van de machine altijd hetzelfde.

We breiden de mogelijkheden van de machine nu verder uit dan het reageren op het invoeren van een invoersymbool door het uitvoeren van een vast uitvoersymbool. We gaan een rij invoersymbolen nu symbool voor symbool invoeren. Als we eerst het symbool s invoeren, waarop de machine reageert met de uitvoer σ, en we voeren daarna een symbool t in, dan herinnert de machine zich, dat aan die t een s voorafgegaan is en hangt hetgeen hij dan uitvoert behalve van t ook van s af. Algemener: als we achtereenvolgens hebben ingevoerd $s_0\ s_1 \ldots s_{n-1}$ met corresponderende uitvoer $\tau_0\ \tau_1 \ldots \tau_{n-1}$ en we voeren daarna s_n in, dan zal de corresponderende uitvoer τ_n afhangen van $s_0,\ldots,s_{n-1},s_n$. Op deze wijze komt er een correspondentie tot stand tussen I^* en O^*. Hetgeen de machine heeft onthouden noemen we een toestand (engels: state) van de machine. We splitsen $s_0,\ldots,s_n$ in $q_n = s_0,\ldots,s_{n-1}$ en s_n, dan is q_n de toestand voor de invoer van s_n en τ_n is een functie van q_n en s_n: $\tau_n = \varphi(q_n,s_n)$, maar $q_{n+1} = s_0,\ldots,s_n$ hangt ook af van q_n en s_n: $q_{n+1} = \psi(q_n,s_n)$. Verder zij q_0 de lege rij.

Na deze inleidende beschouwingen voeren we voor de definitie van een machine een eindige verzameling S van toestanden in en functies φ en ψ, zodat

$$(\tau_k,q_{k+1}) = (\varphi(q_k,s_k),\psi(q_k,s_k)) \ .$$

De vraag, waarom we slechts eindig veel toestanden invoeren, hoewel het ingevoerde verleden in de vorm van een ingevoerde symboolrij steeds langer wordt, wordt beantwoord door de opmerking, dat in een doos niet meer dan eindig veel verschillende informatietoestanden kunnen worden geborgen. Een goede discussie van dit vraagstuk en van andere vraagstukken betreffende de definitie van machines is te vinden in M.L. Minsky, Computation: finite and infinite machines, 1967. We zullen geen strikt formele definitie geven. Voor de definitie van een eindige machine (engels: finite-state machine of finite sequential machine) zijn nodig eindige verzamelingen I, O en S, een begintoestand $q_0 \in S$, en een afbeelding $S \times I \to O \times S$. Daarna kan men iteratief aan een rij invoersymbolen een rij toestanden en een rij uitvoersymbolen toevoegen, hetgeen resulteert in een afbeelding $I^* \to O^*$.

Omdat de verzameling $S \times I$ eindig is, kan de afbeelding $S \times I \to O \times S$ worden vastgelegd door een opsomming van functiewaarden. Omdat $S \times I$ een Cartesisch product is kan men dit in een matrix doen, waarin op elke plaats een uitvoersymbool en een toestand worden geschreven.

Een uiterst eenvoudig voorbeeld is een "parity chequer", die nagaat of een symboolrij een even of een oneven aantal symbolen bevat, door als laatste symbool een O te produceren als het aantal even is en een | als het aantal oneven is. Stel $I = \{|\}$, $O = \{O, |\}$, $S = \{q_0, q_1\}$, q_0 is begintoestand en de matrix is

S \ I	|
q_0	| , q_1
q_1	O, q_0

Om de beslisbaarheid van een verzameling V van symboolrijen met een eindige machine vast te stellen, gebruiken we de verzameling symbolen, waaruit die verzameling is gevormd, als invoerverzameling I en $O = \{O, |\}$. We gebruiken de laatste uitvoer als criterium; de machine is nu bruikbaar als beslisser voor V als de symboolrijen van V als laatste uitvoer | geven en de symboolrijen buiten V als laatste uitvoer O. Een verzameling, waarbij een eindige machine bestaat die dit doet, heet beslisbaar met een eindige machine.

Als men de machine op de boven beschreven wijze gebruikt, zegt men dat men de machine als acceptor gebruikt. Men kan dit ook op een andere manier beschrijven. Men laat dan de uitvoer geheel weg en beoordeelt het resultaat van de invoer naar de bereikte toestand. Men kiest een deelverzameling F van S als verzameling van toegelaten toestanden. De geaccepteerde verzameling is de verzameling van de symboolrijen, die de machine na invoer van die symboolrij in een toegelaten toestand brengen. Een zo beschreven machine heet een eindige automaat (engels: finite automaton).

Het blijkt dat beide methoden van accepteren equivalent zijn: een verzameling van symboolrijen, die volgens de ene methode beslisbaar is, is het ook volgens de andere methode. Wij tonen dit niet aan.

Om aan te tonen, dat eindige machines inadequaat zijn om het probleem van de beslisbaarheid te behandelen, beschouwen we het probleem van de gelijkheid van symboolrijen. Laat U een niet-lege verzameling symbolen zijn en stel $U' := U \cup \{=\}$, waarin = een symbool is, dat niet tot U behoort. Laat V de verzameling zijn van de symboolrijen over U' van de gedaante $A = A$ met $A \in U^*$. Bewering: er bestaat geen eindige machine, die V accepteert.

Om dit aan te tonen veronderstellen we, dat we een eindige machine hebben die V wel accepteert. Omdat U niet leeg is, kunnen we een symbool a in U kiezen. We voeren de volgende notatie in. Als α een symbool is, dan schrijven we α^k voor de symboolrij bestaande uit k maal het symbool α. Formele definitie:

Definitie α^n.
Als α een symbool is en n een natuurlijk getal, dan

$$\alpha^0 := \Lambda ,$$
$$\alpha^{n+1} := \alpha^n \alpha .$$

Als we de machine in de begintoestand de invoer a^k geven, dan zal de machine na afloop in een zekere van k afhankelijke toestand zijn. Omdat er echter maar eindig veel toestanden zijn, geldt het volgende:

Er bestaan natuurlijke getallen k en ℓ met $k \neq \ell$, zodat de invoer a^k en de invoer a^ℓ dezelfde eindtoestand q opleveren.

We voeren nu $a^k = a^k$ in. De machine verwerkt eerst a^k en is dan in toestand q. Wat daarna gebeurt hangt alleen af van de toestand waarin de machine dan is en van de invoer die daarna plaats vindt, dat is q en = a^k. Omdat $a^k = a^k \in V$ is de laatste uitvoer |.

Nu voeren we $a^\ell = a^k$ in. De machine verwerkt eerst a^ℓ en is dan weer in toestand q. Vanuit die toestand verwerkt de machine = a^k, hetgeen hetzelfde resultaat oplevert als in het vorige geval, namelijk een laatste uitvoer |. Dit is echter onjuist, want $a^\ell = a^k \notin V$, omdat $\ell \neq k$. Hieruit volgt, dat een eindige machine, die V accepteert, niet bestaat.

Men zou kunnen stellen, dat het bovenstaande geen goede formalisering is van het probleem van de gelijkheid van symboolrijen, omdat ook allerlei irrelevante symboolrijen over U', zoals bijvoorbeeld symboolrijen, waar meer dan één = in voorkomt, in de beschouwing worden betrokken. Het zou beter zijn van de machine slechts het volgende te eisen:

symboolrijen A = A met $A \in U^+$ worden geaccepteerd,
symboolrijen A = B met $A \in U^+$, $B \in U^+$ en $A \neq B$ worden verworpen,
alle andere symboolrijen uit $(U')^+$: onverschillig.

Het is echter zo, dat er ook geen eindige machine bestaat, die dit verwezenlijkt, hetgeen op dezelfde wijze als hierboven kan worden ingezien.

Het lijkt alsof de moeilijkheid veroorzaakt wordt door het feit, dat de machine op een eenmaal gelezen deel van de invoersymboolrij niet meer kan terugkomen. Men zou de mogelijkheden van de machine willen uitbreiden met het vermogen om terug te lopen langs de invoerrij en de symbolen daarvan nogmaals te lezen; de toestanden van de machine worden dan niet belast met het onthouden van deze informatie. We krijgen dan een tweerichtingmachine in tegenstelling tot de vroegere éénrichtingmachine (engels: two-way machine, one-way machine). Bij de eenrichtingmachine hoeven de invoersymbolen slechts stuk voor stuk beschikbaar te komen en kunnen ze na lezing worden weggegooid. Dit laatste is bij de tweerichtingmachine niet meer het geval, omdat de machine erop terug kan komen. We gebruiken het fysische beeld van een band (engels: tape), waar de symbolen opstaan. De machine heeft een leeskop, die op een bepaalde plaats van de band staat en het daar aanwezige symbool kan lezen; beweging naar links en naar rechts is mogelijk. Wat onder links en rechts moet worden verstaan hangt van de beschouwingswijze af. Men kan de

band als vast beschouwen en de leeskop heen en weer bewegen of men kan de leeskop vast kiezen en de band er onderdoor bewegen. Hoewel het laatste meer overeenkomt met hetgeen in werkelijke rekenautomaten gebeurt, is het in de theorie der abstracte machines gebruikelijk om de eerste methode te gebruiken, bij welk gebruik wij ons zullen aansluiten.

We zullen geen exacte definitie van een tweerichtingmachine geven, maar volstaan met een beschrijving. Er zijn eindige verzamelingen I, O en S. De uitvoer geschiedt als vroeger; wil men ook daarvoor een band gebruiken, dan moet na iedere productie van een symbool één stap naar rechts worden gedaan. De machine is in een toestand en leest een invoersymbool. Afhankelijk daarvan voert de machine een actie uit, die uit drie delen bestaat:

1. Er wordt een uitvoersymbool geproduceerd,
2. de machine gaat over in een nieuwe toestand,
3. de machine gaat op de invoerband één plaats naar rechts of één plaats naar links of blijft op zijn plaats staan.

Voor het begin wordt afgesproken, dat er een begintoestand is en dat de leeskop op het meest linkse symbool van de band staat. De werking van de machine eindigt als de machine hetzij rechts hetzij links van de symboolrij op de band afloopt. Bij deze machine kan een verschijnsel optreden dat bij de eenrichtingmachine niet voorkomt, namelijk dat de actie van de machine nooit eindigt.

De tweerichtingmachine kan ook als acceptor worden gebruikt, hetzij op grond van het laatste invoersymbool, hetzij op grond van de eindtoestand. De verwachting, dat de verzameling der geaccepteerde verzamelingen wordt uitgebreid, gaat echter niet in vervulling. Er geldt namelijk de volgende stelling:

Bij iedere verzameling symboolrijen, waarvoor een tweerichtingmachine bestaat, die de verzameling accepteert, bestaat ook een eenrichtingmachine die de verzameling accepteert (Rabin en Scott, 1959; eenvoudiger bewijs bij Shepherdson, 1959).

We bewijzen deze stelling niet. We merken op, dat het feit dat de eenrichtingmachine en de tweerichtingmachine als acceptor gelijkwaardig zijn, niet betekent dat ze als sequentiële machine (voortbrenger van een functie $I^* \rightarrow O^*$) ook gelijkwaardig zijn, hetgeen niet het geval is.

We komen terug op de mogelijkheid, dat de machine niet stopt. Als men de machine als acceptor wil gebruiken, is men geneigd te eisen, dat de machine op alle invoerbanden stopt, omdat bij niet-stoppen geen uitsluitsel wordt verkregen. Het is niet noodzakelijk dit te doen, omdat de vraag of de machine zal stoppen beslisbaar is. Wat bij gegeven machine en gegeven invoerband in de toekomst in de machine gebeurt hangt op een bepaald ogenblik alleen af van de geldende toestand en van de plaats van de leeskop op de band. Noem de combinatie van deze twee gegevens een configuratie. Als het aantal toestanden in de machine n is en de lengte van de bandinscriptie ℓ, dan zijn er $n\ell$ configuraties. Als de machine in de loop van zijn actie twee keer in dezelfde configuratie komt, stopt hij nooit. Als de machine dus $n\ell + 1$ stappen na het begin nog niet gestopt is, dan stopt hij nooit meer. De vraag of de machine op een bepaalde band stopt is dus beslisbaar.

Alvorens de discussie van mogelijke machines voort te zetten geven we nog een voorbeeld van een taak, die een eindige eenrichtingmachine kan vervullen.

We willen een binaire opteller maken, die uit twee natuurlijke getallen m en n, beide geschreven in het tweetallig stelsel, de som m + n maakt, eveneens geschreven in het tweetallig stelsel. We voeren telkens een paar overeenkomstige cijfers van m en n in en wel om praktische redenen van rechts naar links. Als we dus 101001 en 1101 willen optellen, voeren we achtereenvolgens de paren 11, 00, 01, 11, 00, 10 in, nog gevolgd door één keer 00. Uitvoer gaat met één cijfer tegelijk. Bij het optellen hebben we een overdracht (engels: carry): 0 + 0 = 0, 0 + 1 = 1 + 0 = 1, 1 + 1 = 10; in het laatste geval moet de voorste 1 onthouden worden en bij de volgende optelling worden meegeteld. Meer dan één cijfer vasthouden hoeft niet, want bij de volgende optellingen is het ergste wat kan gebeuren 1 + 1 + 1 = 11, met toch maar één cijfer overdracht. Er zijn twee toestanden nodig: q_0 (geen overdracht, tevens begintoestand), q_1 (wel overdracht).

$$I = \{00,01,10,11\} \ , \quad O = \{0,1\} \ , \quad S = \{q_0,q_1\} \ .$$

S \ I	00	01	10	11
q_0	$0,q_0$	$1,q_0$	$1,q_0$	$0,q_1$
q_1	$1,q_0$	$0,q_1$	$0,q_1$	$1,q_1$

De optelling 101001 + 1101 gaat als volgt:

q_0 11 → 0
q_1 00 → 1
q_0 01 → 1
q_0 11 → 0
q_1 00 → 1
q_0 10 → 1
q_0 00 → 0

Resultaat 110110. De laatste toevoeging van 00 is soms nodig, omdat anders aan het eind nog een overdracht kan overblijven. Met 00 erbij eindigen we altijd in de toestand q_0. Voorbeeld 11 + 1111:

q_0 11 → 0
q_1 11 → 1
q_1 01 → 0
q_1 01 → 0
q_1 00 → 1 .

Resultaat 10010. Zonder de laatste 00 zou de voorste 1 in het resultaat ontbroken hebben.

Ondanks het feit, dat een machine maar eindig veel toestanden heeft, kan een vaste eindige machine willekeurig lange getallen optellen. Dit komt, omdat men tijdens het lezen van de invoer al met rekenen begint. Een gewone rekenautomaat doet dit anders: eerst worden beide getallen gelezen en in het geheugen gezet en pas daarna wordt de som uitgerekend.

Een binaire vermenigvuldiger is niet als eindige eenrichtingmachine te realiseren, zelfs niet als men al tijdens het lezen met rekenen begint. Dit bewijzen we niet.

§ 3. Uitbreiding tot Turing-machines.

Om beslissingsproblemen op te kunnen lossen moeten de vermogens van de machine worden uitgebreid. Bij het vergelijken van symboolrijen bestaat de behoefte om een symbool te markeren of te accentueren om bij terugkeer te kunnen vaststellen waar men gebleven is; het oude symbool moet daarna weer kunnen worden hersteld. Men kan het ook elders noteren, als men beschikt over kladpapier. We nemen nu aan, dat er naast de invoerband, waarop men kan lezen, en de uitvoerband, waarop men kan schrijven, ook nog een kladband is, waarop men kan lezen en schrijven. Op invoerband en kladband kan men in beide richtingen bewegen. Neemt men een kladband van eindige lengte, dan komt men niet verder; men kan hetgeen daarop bewaard kan worden ook in toestanden vastleggen. We nemen daarom een naar beide zijden oneindige kladband. Dit is niet ernstig, omdat dit toch slechts een potentieel oneindig introduceert. Immers, als men per actie van de machine slechts één symbool op de band schrijft, is, als men met een lege band begint, op ieder ogenblik slechts een eindig stuk van de band beschreven. Omdat we niet van tevoren weten, hoe lang de machine zal werken, weten we echter niet hoe groot het gedeelte van de band is, dat voor de berekening nodig is. Men kan dit ook met de volgende beeldspraak beschrijven. Als we ons voorstellen, dat de band van papier is, is op geen enkel ogenblik een oneindige hoeveelheid papier nodig, maar wel moet er ten behoeve van de machine een papierfabriek zijn, die tijdens de werking van de machine naar behoefte nieuw papier kan leveren.

Het is niet zo verwonderlijk, als men bedenkt, dat bij de introductie van symboolrijen een potentieel oneindig is gebruikt, dat men voor het oplossen van beslissingsproblemen voor verzamelingen van symboolrijen ook een vorm van potentieel oneindig nodig heeft.

We hebben nu een machine met drie banden, één voor invoer, één voor uitvoer en één voor klad. Verder is er een eindig aantal toestanden. De machine is in een toestand, leest een invoersymbool en een kladsymbool. De actie van de machine bestaat dan uit schrijven in de uitvoer, schrijven in het klad (wat er stond is dan verdwenen; eventueel kan hetzelfde geschreven worden wat er stond), op de kladband één plaats naar links gaan of één plaats naar rechts gaan of op dezelfde plaats blijven en op de invoerband evenzo en tenslotte overgaan in een nieuwe toestand. De kladband behoort tot de interne configuratie van de machine, de invoer- en uitvoerbanden verzorgen de communicatie met de buitenwereld.

Het blijkt nu, dat zonder verlies van vermogen de machine kan worden vereenvoudigd, doordat men slechts één band gebruikt, die voor invoer, uitvoer en klad dient. Er is dan ook maar één soort bandsymbolen; dat sluit de mogelijkheid niet uit, dat sommige van deze symbolen specifiek voor invoer, resp. uitvoer of klad worden gebezigd.

We geven nu een beschrijving van de Turing-machine, waarbij ook fysische termen worden gebruikt. In § 4 volgt een exacte definitie van de Turing-machine als een formeel systeem.

De machine heeft een eindige verzameling S, waarvan de elementen toestanden heten, en een eindige verzameling I, waarvan de elementen symbolen heten. Verder is er een naar twee zijden oneindige band, verdeeld in hokjes (engels: squares). Eén van de hokjes is het aangewezen hokje (engels: scanned square). In ieder hokje kan een symbool staan; als dat zo is, heet het hokje beschreven. De verzameling der beschreven hokjes is eindig. Terwille van een eenvoudigere beschrijving voegen we een oneigenlijk symbool b toe, dat we loos symbool (engels: blank) noemen en dat in alle hokjes staat, waar geen echt symbool in staat. Het voordeel hiervan is, dat iedere wijziging van hetgeen op de band staat als schrijven kan worden betiteld; hierbij geldt de afspraak, dat schrijven in een hokje ten gevolge heeft, dat hetgeen in het hokje stond uitgewist wordt. Uitwissen van een symbool wordt nu als schrijven van b gedefinieerd. Uiteraard verschilt b van alle elementen van I. Een van de toestanden heet begintoestand. Wat de machine doet, hangt af van de toestand q, waarin hij is en van het symbool s, dat op het aangewezen hokje staat (dit kan ook b zijn, we noemen s een eventueel-oneigenlijk symbool). De machine schrijft in het hokje een eventueel-oneigenlijk symbool s'; de machine kan het aangewezen hokje één plaats naar rechts of naar links verschuiven of ongewijzigd laten; de machine gaat over in toestand q'.

Het blijkt nu geen verlies van algemeenheid op te leveren om de machine bij elke stap van de twee handelingen schrijven en schuiven er slechts één te laten uitvoeren. We kunnen de handeling dan met vier letters beschrijven, een quadrupel:

qss'q' (schrijfopdracht),
qsrq' (rechtsschuifopdracht),
qslq' (linksschuifopdracht).

In alle gevallen is overgang naar een nieuwe toestand begrepen; uiteraard kan q' = q zijn.

Men zou nu voor ieder paar q,s een quadrupel kunnen vastleggen, maar dan zou de machine nooit met werken ophouden. Voor sommige q,s is er geen quadrupel: de machine stopt.

Bij de machine hoort een verzameling quadrupels; omdat de machine deterministisch is, geldt, dat als qss'q' en qss"q" quadrupels uit die verzameling zijn, dan s" = s' en q" = q'. Hierin zijn s' en s" eventueel-oneigenlijke symbolen of ℓ of r. Wat de machine doet hangt verder af van hetgeen op de band staat: de bandinscriptie.

De werking van de machine op een gegeven ogenblik hangt af van de bandinscriptie, het aangewezen hokje en de geldende toestand. Deze gegevens samen heten de momentane beschrijving (engels: instantaneous description) van de machine.

We beginnen in de begintoestand en met een bandinscriptie, die als invoer fungeert; over de plaats van het aangewezen hokje bij het begin maken we geen vaste afspraak, zodat ook dit gegeven tot de invoer behoort. Soms neemt men voor het aangewezen hokje het meest linkse hokje dat met een eigenlijk symbool beschreven is; wij leggen ons daarop niet vast.

Er vindt nu volgens bovenstaande beschrijving een rij handelingen plaats, die een Turing-berekening vormt. Deze kan onbepaald doorlopen of stoppen. Als de machine stopt, is de bandinscriptie plus het aangewezen hokje de uitvoer van de machine. Om van deze uitvoer een symboolrij te maken, die als resultaat van de berekening kan gelden, zijn er verschillende conventies. We noemen er twee.

1. Laat in de bandinscriptie loze symbolen weg, ook degene die tussen eigenlijke symbolen staan. Plak hetgeen overblijft aan elkaar tot één symboolrij.
2. Neem het zo groot mogelijke aaneengesloten stuk van de bandinscriptie, waar geen loos symbool in staat en dat het aangewezen hokje bevat, en verwaarloos de rest.

Wij zullen met geval 1 werken.

§ 4. Turing-machine als formeel systeem

Als we de Turing-machine als formeel systeem invoeren, zijn de toestanden ook symbolen.

We gaan uit van twee eindige, niet-lege verzamelingen symbolen S en I, waarvan de elementen resp. toestandssymbolen en bandsymbolen heten. Verder zijn er nog drie verschillende losse symbolen b, r, ℓ. We veronderstellen deze alle verschillend:

$$S \cap I = \emptyset \ , \quad (S \cup I) \cap \{b,r,\ell\} = \emptyset \ .$$

We voeren in $I_0 := I \cup \{b\}$. We beschouwen een formeel systeem met $S \cup I \cup \{b,r,\ell\}$ als verzameling symbolen. Een der elementen van S is als begintoestand gekarakteriseerd; we noemen deze vaak q_0.

Het blijkt voor latere toepassingen, waarin verscheidene Turing-machines worden beschouwd, die samengesmolten moeten worden tot één Turing-machine, noodzakelijk het bovenstaande te verruimen door het gebruik van uitgebreide symbolen (zie § 6 van Hoofdstuk I). We gaan dan uit van een verzameling symbolen en kiezen $S \cup I \cup \{b,r,\ell\}$ als eindige verzameling uitgebreide symbolen. Desondanks zullen we de elementen van deze verzameling de symbolen van de Turing-machine blijven noemen en niet de uitgebreide symbolen.

Definitie quadrupel.
Een quadrupel is een symboolrij qstq', waarin $q \in S$, $s \in I_0$, $t \in I_0 \cup \{r,\ell\}$, $q' \in S$.

Voor een rij quadrupels is geen scheidingssymbool nodig, omdat alle quadrupels uit vier symbolen bestaan.

Definitie rij quadrupels.
Een quadrupel is een rij quadrupels.
Als R een rij quadrupels is en Q een quadrupel, dan is RQ een rij quadrupels.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een rij quadrupels.

Definitie voorkomen van quadrupel in een rij quadrupels.
Als Q een quadrupel is en R een rij quadrupels, dan komt Q in R voor, als er eventueel-lege rijen quadrupels S en T bestaan, zodat R = SQT.

Definitie rij quadrupels zonder herhalingen.
Als R een rij quadrupels is en voor alle quadrupels Q en alle eventueel-lege rijen quadrupels S, T, S', T', waarvoor R = SQT en R = S'QT', ook S = S', dan heet R een rij quadrupels zonder herhalingen.

Definitie compatibel.
De quadrupels $q_0 s_0 t_0 q_0'$ en $q_1 s_1 t_1 q_1'$ heten compatibel als uit $q_0 = q_1$ en $s_0 = s_1$ volgt $t_0 = t_1$ en $q_0' = q_1'$.

Definitie machinerij.
Een rij quadrupels R zonder herhalingen en zodat alle quadrupels Q en Q', die in R voorkomen, compatibel zijn, heet een machinerij.

Van nu af aan nemen we aan dat een machinerij M gegeven is; hetgeen volgt hangt af van de keuze van M. We willen nu een momentane beschrijving formeel met een symboolrij weergeven. De bandinscriptie is na weglating van loze symbolen voor en achter een symboolrij, eventueel doorschoten met loze symbolen. We moeten nu nog het aangewezen hokje en de geldende toestand aangeven. We doen dit door het toestandssymbool in de rij in te lassen onmiddellijk links van het aangewezen hokje. Om bij gegeven bandinscriptie, toestand en aangewezen hokje de symboolrij geheel vast te leggen vegen we links en rechts zoveel mogelijk loze symbolen weg; het kan zijn dat er een aantal moeten blijven staan om het aangewezen hokje te kunnen markeren.

Voor de formele definities bedenken we, dat de symbolen van een Turingmachine uitgebreide symbolen kunnen zijn. Als A een uitgebreide symboolrij is, dan is A op eenduidige wijze te schrijven in de gedaante $A = aA'$, waarin a een uitgebreid symbool is en A' een eventueel-lege symboolrij en analoog met het uitgebreide symbool aan de achterzijde (zie Hoofdstuk I, stelling 6.3). Dit rechtvaardigt het gebruik van het bepaald lidwoord in de volgende definitie.

Definitie beginsymbool en eindsymbool.
Als A een eventueel-lege uitgebreide symboolrij is en a een uitgebreid symbool, dan heet a het beginsymbool van aA en het eindsymbool van Aa, behorende bij de gegeven verzameling van uitgebreide symbolen.

Definitie momentane beschrijving en bijbehorende bandinscriptie.
Een momentane beschrijving is een symboolrij AqC, waarin $q \in S$, $A \in I_0^*$, $C \in I_0^*$, zodat als $A \neq \Lambda$, het beginsymbool van A niet b is en als $C \neq \Lambda$, het eindsymbool van C niet b is. De bij deze momentane beschrijving behorende bandinscriptie is AC.

Bedenk dat als $A = \Lambda$ het beginsymbool van AC wel b kan zijn en analoog voor het eindsymbool van AC als $C = \Lambda$.

Definitie direct gevolg.

De momentane beschrijving A'q'C' ($q' \in S$, $A' \in I_0^*$, $C' \in I_0^*$) is een direct gevolg van de momentane beschrijving AqC ($q \in S$, $A \in I_0^*$, $C \in I_0^*$) ten opzichte van de machinerij M in de volgende gevallen:

We voeren de uitgebreide symbolen a en c en de eventueel-lege uitgebreide symboolrijen A" en C" in door

$$A = A''a \quad \text{als} \quad A \neq \Lambda ,$$

$$A'' = \Lambda, \; a = b \quad \text{als} \quad A = \Lambda ,$$

$$C = cC'' \quad \text{als} \quad C \neq \Lambda ,$$

$$C'' = \Lambda, \; c = b \quad \text{als} \quad C = \Lambda .$$

Het quadrupel qctq' komt voor in M.

1. $t \in I_0$. Dan is

$$A' = A$$

en

$$C' = \begin{cases} tC'' & \text{als} \quad C'' \neq \Lambda \quad \text{of} \quad t \neq b , \\ \Lambda & \text{als} \quad C'' = \Lambda \quad \text{en} \quad t = b . \end{cases}$$

2. $t = r$. Dan is

$$A' = \begin{cases} Ac & \text{als} \quad A \neq \Lambda \quad \text{of} \quad c \neq b , \\ \Lambda & \text{als} \quad A = \Lambda \quad \text{en} \; c = b \end{cases}$$

en

$$C' = C'' .$$

3. $t = \ell$. Dan is

$$A' = A''$$

en

$$C' = \begin{cases} aC & \text{als} \quad C \neq \Lambda \quad \text{of} \quad a \neq b , \\ \Lambda & \text{als} \quad C = \Lambda \quad \text{en} \; a = b . \end{cases}$$

Stelling 4.1. Bij iedere momentane beschrijving bestaat ten hoogste één momentane beschrijving, die er een direct gevolg van is.

Definitie finaal.

Een momentane beschrijving heet finaal, als zij geen directe gevolgen bezit.

Stelling 4.2. Een momentane beschrijving is dan en slechts dan finaal, als er bij de keuze van de notatie AqC voor de momentane beschrijving en de keuze van c als in de definitie van direct gevolg geen $t \in I_0 \cup \{r, \ell\}$ en $q' \in S$ bestaat, zodat het quadrupel qctq' in M voorkomt.

De bewijzen van de stellingen 4.1 en 4.2 zijn eenvoudig.

We voeren nu een lijst van momentane beschrijvingen in zoals dit voor lijsten in § 10 van hoofdstuk I is geschied. Daartoe voegen we een scheidingssymbool ⌈ toe, dat verschilt van de eerder ingevoerde symbolen.

Definitie toegelaten lijst van momentane beschrijvingen.
Als P een momentane beschrijving is, dan is P⌈ een toegelaten lijst van momentane beschrijvingen.
Als R een eventueel-lege lijst van momentane beschrijvingen is, P en P' momentane beschrijvingen zijn, RP⌈ een toegelaten lijst van momentane beschrijvingen is en P' een direct gevolg van P is, dan is RP⌈P'⌈ een toegelaten lijst van momentane beschrijvingen.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een toegelaten lijst van momentane beschrijvingen.

We willen nu begin en slot van een toegelaten lijst van momentane beschrijvingen definiëren. We geven geen formele definitie, maar stellen vast:
Het begin van P⌈ is P en het slot van P⌈ is P.
Het begin van RP⌈P'⌈ is het begin van RP⌈ en het slot van RP⌈P'⌈ is P'.

Om dit tot een definitie door recurrentie langs de opbouw van een toegelaten lijst van momentane beschrijvingen te maken zou eerst moeten worden aangetoond, dat een dergelijke definitie gegeven mag worden.

Definitie machineberekening.
Een machineberekening is een toegelaten lijst van momentane beschrijvingen, waarvan het begin de gedaante Aq_0C heeft, waarin q_0 de begintoestand is, $A \in I_0^*$, $C \in I_0^*$ en waarvan het slot finaal is.

Definitie invoer, uitvoer.
Het begin (resp. slot) van een machineberekening heet de invoer (resp. uitvoer) van die machineberekening.

Stelling 4.3. Machineberekeningen met gelijke invoer zijn gelijk.

Stelling 4.4. Als er bij gegeven q_0 = begintoestand, $A \in I_0^*$, $C \in I_0^*$ geen machineberekening bestaat met invoer Aq_0C, dan is de lengte van toegelaten lijsten van momentane beschrijvingen met begin Aq_0C onbegrensd.

De bewijzen van stellingen 4.3 en 4.4 zijn eenvoudig. Onder de omstandigheden van stelling 4.4 zeggen we, dat de machine op de invoer Aq_0C niet stopt.

We willen nu trachten tot een afbeelding $I^* \to I^*$ te komen. Daartoe moeten we de loze symbolen uit de uitvoer verwijderen, benevens het toestandssymbool.

Definitie d.
De afbeelding d: $(I_0 \cup S)^* \to I^*$ wordt gedefinieerd door

$$d(\Lambda) := \Lambda ,$$

$$d(A\alpha) := \begin{cases} d(A) & \text{als } \alpha = b \text{ of } \alpha \in S , \\ d(A)\alpha & \text{als } \alpha \neq b \text{ en } \alpha \notin S ; \end{cases}$$

hierin: $A \in (I_0 \cup S)^*$, $\alpha \in I_0 \cup S$.

Als nu $C \in I^*$, dan voegen we daaraan toe d(U), waarin U de uitvoer is van de machineberekening met invoer q_0C. Het kan echter zijn, dat de machine op de invoer q_0C niet stopt, zodat de afbeelding dan niet gedefinieerd is. We drukken dit uit door te zeggen, dat de machine een partiële afbeelding $I^* \to I^*$ definieert.

Men kan nu proberen voor de uitvoering van bepaalde taken Turing-machines te maken. Dit doen we door eerst enkele Turing-machines te construeren voor zeer eenvoudige taken en deze vervolgens te assembleren tot grotere Turing-machines voor ingewikkeldere taken.

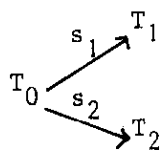
§ 5. Koppeling van Turing-machines. Elementaire machines

We beginnen met een voorbeeld. Stel dat T_0 en T_1 Turing-machines zijn. We maken een nieuwe Turing-machine, die we met $T_0 \rightarrow T_1$ aanduiden. De toestanden van een machine zijn een interne aangelegenheid. Of ze bij de machines T_0 en T_1 al dan niet gelijk zijn beschouwen we als niet relevant. De bandsymbolen echter zijn een externe aangelegenheid. We veronderstellen niet, dat T_0 en T_1 dezelfde verzamelingen bandsymbolen hebben; noem ze respectievelijk $I^{(0)}$ en $I^{(1)}$. De machine $T_0 \rightarrow T_1$ zal $I^{(0)} \cup I^{(1)}$ als verzameling bandsymbolen hebben. We kunnen echter ook T_0 en T_1 opvatten als machines, die elk $I^{(0)} \cup I^{(1)}$ als verzameling bandsymbolen hebben; formeel betekent dit een overgang op een andere machine. We nemen de grotere verzameling bandsymbolen, maar dezelfde machinerij. Deze machine kunnen we een invoer geven, die ook toegevoegde symbolen bevat. Zolang het aangewezen hokje één der oude symbolen bevat, werkt de machine evenals de oorspronkelijke. Zodra echter in het aangewezen hokje een toegevoegd symbool staat, stopt de machine. Op grond van bovenstaande afspraak nemen we aan, dat de machines, die we bij het koppelen zullen beschouwen, alle dezelfde verzameling bandsymbolen bezitten. Verder spreken we af, dat de symbolen b, ℓ, r in alle machines dezelfde zijn.

Het is de bedoeling, dat $T_0 \rightarrow T_1$ het volgende doet. De invoer gaat in de machine T_0; als echter T_0 stopt, gaat de uitvoer met de begintoestand van T_1 in T_1. We moeten aantonen, dat er een Turing-machine bestaat, die dit resultaat realiseert. Om dit te doen moeten we over een voldoende groot aantal toestanden kunnen beschikken. Over de vraag of de toestanden van T_0 en T_1 verschillend zijn, willen we geen veronderstellingen maken; het is zelfs toegestaan dat T_0 en T_1 dezelfde machine zijn. We maken gebruik van het feit dat we met uitgebreide symbolen mogen werken en voegen aan de verzameling symbolen een nieuw nummeringssymbool $\llcorner$ toe, om daarmee de toestanden van T_0 en T_1 (en in latere gevallen van verdere optredende machines) van verschillende nummers te voorzien.

Laat voor i = 0 of 1 in T_i nu S_i de verzameling toestanden zijn, M_i de machinerij en q_{i0} de begintoestand. I is de verzameling bandsymbolen van beide machines. We plaatsen nu achter ieder toestandssymbool van T_1 een $\sqcup$ en vormen dienovereenkomstig een gewijzigde machinerij M_1' (de verzameling toestanden noemen we S_1'). Nu nemen we voor alle $q \in S_0$ en $s \in I_0$, waarvoor er geen quadrupel in M_0 voorkomt, dat met qs begint, een quadrupel $qssq_{10}\sqcup$. We rijgen de zo verkregen quadrupels aaneen tot een rij M_{01}. Om deze te bepalen moet een volgorde gekozen worden; dit kan bijvoorbeeld gedaan worden door de verzamelingen S_0 en I_0 te nummeren. De machine $T_0 \to T_1$ heeft nu $S_0 \cup S_1'$ als verzameling toestandssymbolen, $M_0M_1'M_{01}$ als machinerij en q_{00} als begintoestand.

Een ander voorbeeld is $T_0 \overset{s}{\to} T_1$, waarin $s \in I_0$. Als nu de machine T_0 stopt met s in het aangewezen hokje, dan gaat T_1 verder; als er iets anders in het aangewezen hokje staat, stopt de machine. Op analoge wijze kan



geïnterpreteerd worden.

We kunnen nu ook ingewikkelder schema's met machines en pijlen maken; een dergelijk schema heet een <u>diagram</u>. We zullen geen algemene definitie van een diagram geven en ook niet bewijzen, dat er bij ieder diagram een Turingmachine bestaat, die de werking van dat diagram realiseert; daarbij moet voor het nummeren van de toestanden en het maken van een machinerij een volgorde gekozen worden van de machines die in het diagram voorkomen. We maken wel enkele opmerkingen over diagrammen.

Van ieder symbool voor een machine in een diagram mag voor iedere $s \in I_0$ ten hoogste één pijl uitgaan, waar s bij staat. In het diagram moet aangegeven worden, waar met de werking wordt begonnen; we maken de afspraak, dat we, als er niets anders aangegeven is, linksboven beginnen. Als tussen T_0 en T_1 in een diagram meer pijlen lopen behorende bij $s_0, s_1, \ldots$, dan kunnen we een naam V voor de verzameling van die symbolen invoeren en $T_0 \overset{V}{\to} T_1$ schrijven. Hetgeen we vroeger $T_0 \to T_1$ genoemd hebben, is dan hetzelfde als $T_0 \overset{I_0}{\to} T_1$. Tenslotte laten we bij naast elkaar staande T_0 en T_1 in $T_0 \to T_1$ de pijl wel weg: T_0T_1.

Als een machine die met een diagram gemaakt is, waarbij een nummeringssymbool gebruikt is, als bestanddeel van een nieuw diagram wordt gekozen, zal voor dit nieuwe diagram weer een nieuw nummeringssymbool moeten worden gebruikt.

Alvorens met diagrammen machines voor bepaalde taken te maken definiëren we eerst enkele elementaire machines. Daarvoor hebben we twee toestandssymbolen nodig, die we hier niet expliciet zullen maken. We noemen ze q_0 en q_1. De machines behoren bij een gegeven verzameling bandsymbolen I. We zullen de afhankelijkheid van I niet in de notatie tot uitdrukking brengen. Voor de nu volgende machines is een I gegeven. Voor het opstellen van de machinerij moet een volgorde in I_0 gekozen worden.

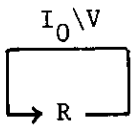
De rechtsmachine R gaat één plaats naar rechts en stopt. $S = \{q_0, q_1\}$, q_0 begintoestand. De quadrupels zijn $q_0 s r q_1$ voor alle $s \in I_0$.

De linksmachine L analoog.

De drukmachine (engels: printing machine). Deze hangt af van de keuze van het gedrukte symbool: D_s met $s \in I_0$. Als $s = b$, dan hebben we een uitveegmachine. $S = \{q_0\}$, q_0 begintoestand. De quadrupels zijn $q_0 s' s q_0$ voor alle $s' \in I_0 \backslash \{s\}$.

Met behulp van deze elementaire machines gaan we andere machines maken.

De rechtszoekmachine R_V, waarin $V \subset I_0$, gaat stapsgewijs naar rechts zoeken tot hij een symbool gevonden heeft, dat in V ligt. Diagram:



Als op het beginhokje een symbool uit V staat, gaat de machine toch naar rechts om te zoeken. Als er nergens rechts van het beginhokje een symbool uit V staat, stopt de machine niet. Als de machine stopt, is de bandinscriptie ongewijzigd gebleven; alleen het aangewezen hokje is verplaatst.

De linkszoekmachine L_V analoog.

De zoekmachine Z_V zoekt ergens op de band naar een symbool, dat tot V behoort. Alleen als er nergens op de band een symbool uit V staat, is het geoorloofd, dat de machine niet stopt. Als de machine stopt, moet de bandinscriptie ongewijzigd zijn gebleven. Op grond van deze voorschriften mag de machine niet uitsluitend naar rechts of uitsluitend naar links gaan zoeken. De machine moet op de band heen en weer lopen en moet daarom op de band kunnen markeren, waar hij met inspecteren gebleven is; omdat er geen a priori

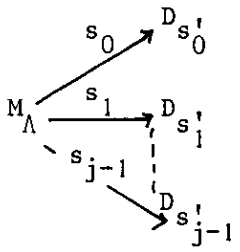
bovengrens is voor de afstand, waarover hij zich over de band verplaatst, kan dit niet via de toestanden worden geregeld. Verder moet na afloop datgene wat op de band gestaan heeft worden hersteld. Er zijn twee verschillende methoden om dit te verwezenlijken.

1. We kiezen een markeersymbool zo dat daaraan te zien is voor welk symbool het in de plaats gekomen is: een "geaccentueerde" van dat symbool. Het bezwaar hiervan is, dat er veel nieuwe symbolen nodig zijn, evenveel als er verschillende symbolen kunnen staan op plaatsen waar zulk een markeersymbool wordt geplaatst. De verzameling bandsymbolen van de machine wordt met niet-essentiële symbolen uitgebreid. Dit vormt een belasting als de machine als bestanddeel in een diagram wordt gebruikt.
2. We kiezen een vast markeersymbool en onthouden hetgeen op die plaats op de band heeft gestaan met behulp van toestanden, hetgeen kan, omdat er maar eindig veel bandsymbolen zijn. Het bezwaar hiervan is, dat de toestanden hierdoor op oneigenlijke wijze worden gebruikt, omdat het natuurlijker lijkt te zijn de band te gebruiken als geheugen en de toestanden voor besturing.

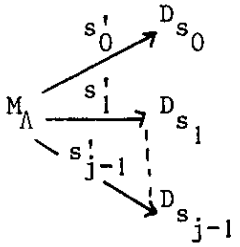
De nadelen van de ene methode weerspiegelen zich als voordelen van de andere methode. De geringere belasting van de toestanden in geval 1 leidt tot een eenvoudigere beschrijving van de machine, dikwijls in de vorm van een eenvoudiger diagram. In geval 2 is ten hoogste één extra symbool nodig en in vele gevallen zelfs helemaal geen, omdat één van de bestaande symbolen kan worden gebruikt. Omdat wij de toestanden als een interne zaak van de machine beschouwen, die we, zodra de machine is gedefinieerd, verder zoveel mogelijk buiten beschouwing laten, zullen we aan methode 2 de voorkeur geven. Bij wijze van voorbeeld zullen we bij de zoekmachine echter nog beide methoden behandelen.

Om methode 1 te kunnen toepassen definiëren we eerst een <u>accentueermachine</u> A_W, waarin $W \subset I_0$. Voor elke $s \in W$ voeren we een nieuw symbool s' in. Laat W' de verzameling van die symbolen zijn. De verzameling bandsymbolen van A_W is nu $I \cup W'$; de in de definitie van A_W voorkomende machines moeten ook als machines met $I \cup W'$ als verzameling bandsymbolen worden opgevat. Omdat in A_W de eerste actie al afhangt van hetgeen op de band staat, voeren we als hulpmiddel nog in de <u>lege machine</u> M_Λ. Omdat de lege symboolrij niet is

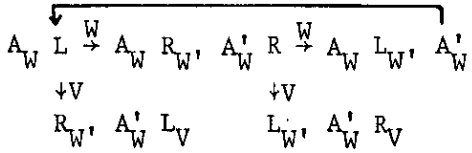
toegestaan als machinerij, kiezen we voor M_Λ: $S = \{q_0, q_1\}$ en één quadrupel $q_0 b b q_1$. Laat nu $s_0, s_1, \ldots, s_{j-1}$ de elementen van W zijn, dan is het diagram van A_W:



Naast de accentueermachine hebben we ook een <u>desaccentueermachine</u> A'_W nodig met diagram:



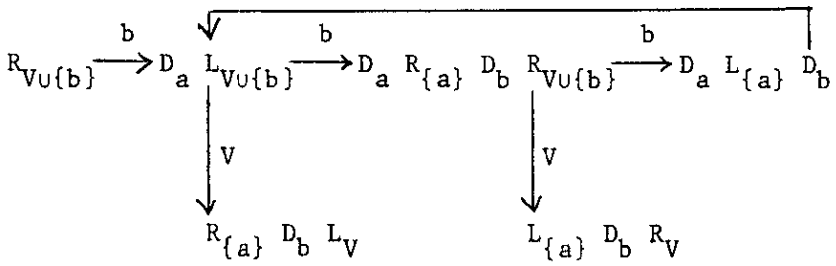
Voor het diagram van Z_V definiëren we $W := I_0 \backslash V$ en stellen vast, dat alle optredende machines $I \cup W'$ als verzameling bandsymbolen hebben. Diagram:



Als men deze machine een invoer geeft, waarin geen symbolen uit W' voorkomen, hetgeen de bedoeling is, dan bevat, als de machine stopt, de uitvoer ook geen symbolen uit W'.

Bij het toepassen van methode 2 maken we gebruik van het feit, dat we op de ogenblikken, dat we markeersymbolen nodig hebben, nog geen symbool van V ontmoet hebben, zodat we een vastgekozen symbool van V als markeersymbool kunnen kiezen. Dit kan natuurlijk niet als V leeg is, maar dan heeft het zoekprobleem uiteraard geen oplossing, zodat we dat geval verder buiten beschouwing laten. Verder maken we gebruik van de bijzondere rol die het loze

symbool b vervult en die maakt dat als we bijvoorbeeld naar rechts gaan zoeken, we er zeker van zijn op den duur b tegen te komen. Dit heeft wel ten gevolge dat het verschil maakt of b zelf wel of niet in V zit. Als echter $b \in V$, dan lost de machine R_V het zoekprobleem op. Stel dus $b \notin V$ en kies een $a \in V$. We nemen als diagram voor Z_V:



Doordat we het markeersymbool a uitsluitend op een plaats zetten waar een b heeft gestaan, hoeft er in dit geval niets onthouden te worden over het symbool dat er gestaan heeft.

Een zoekmachine wordt veel eenvoudiger, als van tevoren gegeven is, dat slechts een begrensd stuk van de band moet worden afgezocht. Dit stuk wordt vastgelegd door twee plaatsen op de band, waar een markeersymbool m staat. Het af te zoeken gedeelte is het gedeelte tussen die twee plaatsen. Gegeven is, dat op dat gedeelte het symbool m niet voorkomt en verder $m \notin V$. Diagram:

$$R_{V\cup\{m\}} \overset{m}{\rightarrow} L_{V\cup\{m\}} \overset{m}{\rightarrow} R \ .$$

Als het onderzochte deel geen symbolen uit V bevat, stopt de machine op het meest linkse hokje van het onderzochte deel.

Als we met een machine een zekere invoer in een zekere uitvoer om willen zetten, interesseren we ons vaak wel voor de plaats van het aangewezen hokje, maar niet voor de toestand. In dat geval schrijven we in plaats van het toestandssymbool een $\downarrow$.

Een <u>opteller</u> heeft I = {O, |} en moet $\downarrow O|^m O|^n$ omzetten in $\downarrow O|^{m+n}$ voor alle m ∈ N en n ∈ N. Wat de machine met een invoer van andere gedaante doet, interesseert ons niet. Diagram:

$$R_{\{O\}}\ D_{|}\ R_{\{b\}}\ L\ D_b\ L_{\{O\}}\ .$$

Een <u>vermenigvuldiger</u> moet $\downarrow O|^m O|^n$ omzetten in $\downarrow O|^{mn}$. Men wil vermenigvuldigen uitvoeren als herhaald optellen. Men moet dan echter zorgen, dat men na uitvoering van de optelling ook de opgetelde getallen nog bewaard heeft. Daartoe is er behoefte aan een copieermachine, die we nu eerst bespreken. We laten daaraan nog een notatiekwestie voorafgaan.

Bij de beschrijving van een omzetting van een invoer in een uitvoer is het vaak hinderlijk, dat aan voor- en achterzijde van invoer en uitvoer geen loze symbolen mogen staan. We laten deze eis daarom nu vallen. Dat betekent echter wel, dat om de invoer en uitvoer te verkrijgen, die in werkelijkheid bij de machine behoren, de overtollige loze symbolen voor en achter eerst moeten worden verwijderd. Dit kan als volgt formeel worden beschreven.

<u>Stelling 5.1</u>. Als U een verzameling uitgebreide symbolen is, b een symboolrij, b ∉ U, U_0 := U ∪ {b} een verzameling uitgebreide symbolen, en A ∈ U_0^*, dan geldt voor A één en slechts één van de volgende drie mogelijkheden:

1. Er bestaat een k ∈ N: $A = b^k$;
2. er bestaan k ∈ N, ℓ ∈ N, a' ∈ U: $A = b^k a' b^\ell$;
3. er bestaan k ∈ N, ℓ ∈ N, a' ∈ U, a'' ∈ U, A' ∈ U_0^*: $A = b^k a' A' a'' b^\ell$.

Bovendien zijn de k, ℓ, a', a'' en A' door A eenduidig vastgelegd.

Het bewijs van stelling 5.1 is eenvoudig. Dat A in de gedaante geschreven kan worden, die in de stelling is aangegeven, bewijst men door recurrentie langs de opbouw van de eventueel-lege uitgebreide symboolrij A. Bij de eenduidigheid gebruikt men stelling 6.5 uit hoofdstuk I.

<u>Definitie ω</u>.
U, b en U_0 als in stelling 5.1. We definiëren ω: $U_0^* \to U_0^*$ als volgt: Als A ∈ U_0^*, dan

$$\omega(A) := \begin{cases} \Lambda & \text{in geval 1 van stelling 5.1,} \\ a' & \text{in geval 2 van stelling 5.1,} \\ a'A'a'' & \text{in geval 3 van stelling 5.1.} \end{cases}$$

Stelling 5.2. U, b en U_0 als in stelling 5.1. Als $A \in U_0^*$, dan

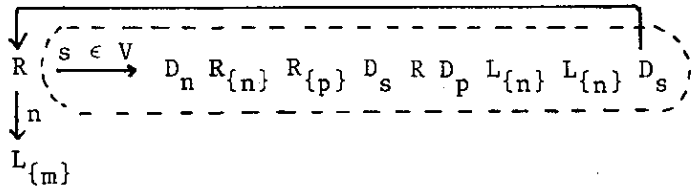
$$\omega(\omega(A)) = \omega(A) ,$$
$$\omega(Ab) = \omega(bA) = \omega(A) .$$

Het bewijs van stelling 5.2 is eenvoudig.

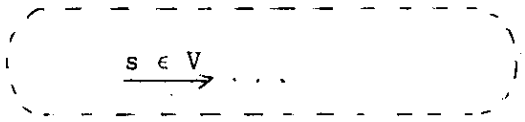
We passen het bovenstaande toe met $U = I \cup \{\downarrow\}$. Als we in het vervolg spreken over een machine die $A \downarrow C$ omzet in $A' \downarrow C'$ (waarin $A \in I_0^*$, $C \in I_0^*$, $A' \in I_0^*$, $C' \in I_0^*$), dan bedoelen we een machine, waarin bij de invoer $\omega(A \downarrow C)$ de uitvoer $\omega(A' \downarrow C')$ behoort. Uiteraard moet daarin de $\downarrow$ nog door een toestandssymbool worden vervangen.

Bij een copieermachine is een te copiëren symboolrij op de band links en rechts gemarkeerd met symbolen m resp. n. We nemen aan, dat het gedeelte van de band waar gecopieerd moet worden rechts ligt van het gedeelte van de band waar het te copiëren stuk staat. Met een symbool p wordt de plaats gemarkeerd, waar het meest linkse symbool van de te copiëren symboolrij moet komen. Verder is er een $V \subset I_0$ gegeven, zodat de te copiëren symboolrij slechts symbolen uit V bevat. We veronderstellen $m \notin V$, $n \notin V$, $p \notin V$; de symbolen m, n en p behoeven niet verschillend te zijn. Verder komen n en p niet voor op het gedeelte van de band, gelegen tussen bovengenoemde markeerplaatsen n en p. Om technische redenen drukken we achter de gecopieerde symboolrij de p nog eens af.

Als $X \in V^*$, $Y \in (I_0 \backslash \{n,p\})^*$, $T \in I_0^*$, $U \in I_0^*$, $Z \in I_0^*$, $\ell(X) = \ell(Z)$, dan moet $T\downarrow mXnYpZU$ overgaan in $T\downarrow mXnYXpU$. Een copieermachine $C^V_{m,n,p}$, die dit realiseert, is beschreven met het volgende diagram:

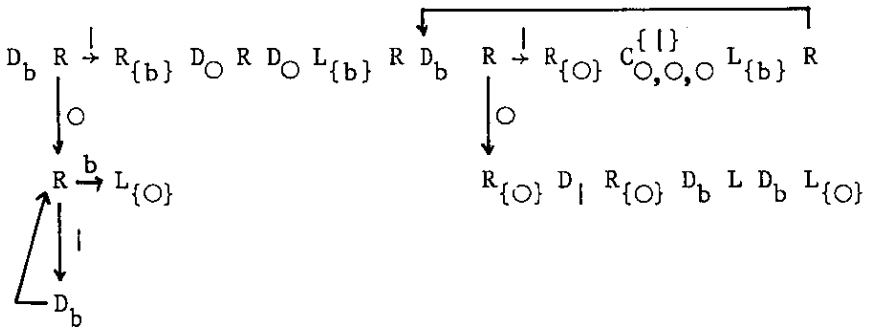


In dit diagram is als notatie gebruik gemaakt van een schema van de vorm



Als iets dergelijks in een diagram voorkomt, moet in het diagram voor elk element s van V het binnen de kring staande stuk worden herhaald en met een pijl $\overset{s}{\rightarrow}$ gehecht aan het symbool waar de pijl $\xrightarrow{s \in V}$ aan gehecht was.

Een vermenigvuldiger wordt nu gerealiseerd met het volgende diagram ($I = \{0,1\}$):



Tenslotte vermelden we nog enkele machines, waarvan we het bestaan niet zullen aantonen.

Een <u>rechtsverschuiver</u> $T^r_{m,n}$ verschuift een met markeersymbolen m en n links resp. rechts gemarkeerde symboolrij één plaats naar rechts. De markeersymbolen worden meeverschoven. Het symbool rechts naast n, dat door het verschuiven wordt uitgewist, wordt links van de verschoven m weer afgedrukt.

Als $X \in (I_0 \setminus \{m,n\})^*$, $T \in I_0^*$, $s \in I_0$, $U \in I_0^*$, dan moet TmX↓nsU overgaan in TsmX↓nU.

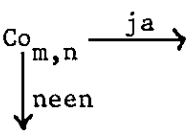
Een linksverschuiver $T^{\ell}_{m,n}$ wordt op analoge wijze gedefinieerd.

Bij een bewaarmachine gaat het om het volgende probleem. Laat een Turing-machine T gegeven zijn. We willen een machine hebben, die hetzelfde doet als T, maar tevens een op de band gegeven symboolrij ongeschonden bewaart. Het kan zijn, dat dit onschendbare gedeelte rechts ligt van het gedeelte van de band, waar de machine werkt of dat het links ligt of zelfs dat er aan beide zijden zo'n gedeelte ligt. Wij beperken ons hier tot het geval, dat het rechts ligt. Daar het a priori onbekend is, hoeveel bandruimte de machine T bij zijn werk nodig heeft, zal het onschendbare gedeelte af en toe naar rechts moeten worden verschoven. Daartoe schermen we het aan de linkerkant af met een symbool m, dat geen bandsymbool van T is. De bewaarmachine zal de werking van T nabootsen, maar, zodra het symbool m gelezen wordt, deze werking onderbreken en het onschendbare gedeelte één plaats naar rechts verschuiven, waarna de werking van T in de toestand van voor de onderbreking wordt voortgezet.

Als T een Turing-machine is met I als verzameling bandsymbolen, m een symbool, $m \notin I_0$, $n \in I_0 \cup \{m\}$, dan wordt een machine met $I \cup \{m\}$ als verzameling bandsymbolen gezocht, die het volgende doet. Laat P een invoer zijn voor T, $k \in N$ en $Y \in (I_0\backslash\{n\})^*$. Als T op de invoer P niet stopt, dan stopt de machine niet op de invoer Pb^kmYn. Als T op de invoer P de uitvoer P' levert, dan is er een $k' \in N$, zodat de machine op de invoer Pb^kmYn de uitvoer $P'b^{k'}mYn$ levert.

Tenslotte willen we dat een machine kan vaststellen of twee op de band aanwezige symboolrijen gelijk zijn of niet. We beperken ons tot het geval, dat deze symboolrijen natuurlijke getallen voorstellen in de gedaante van rijen streepjes. Wat we zoeken is eigenlijk geen machine, maar een diagram met twee uitgangen, gemarkeerd met ja en neen, dat ingebouwd kan worden in een diagram van een grotere machine.

Een getalvergelijkingsdiagram



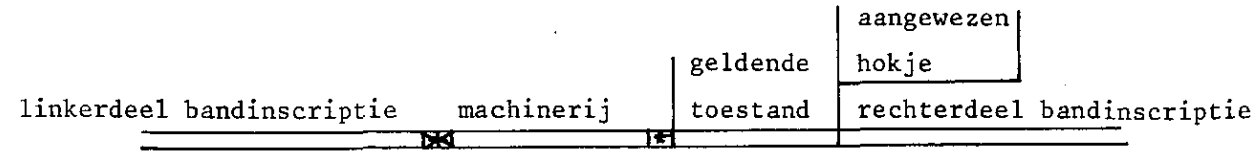
doet het volgende. Gegeven een verzameling bandsymbolen I met $| \in I$, $m \in I_0\backslash\{|\}$, $n \in I_0\backslash\{|\}$. Als $T \in I_0^*$, $s \in I_0\backslash\{|\}$, $k \in N$, $X \in (I_0\backslash\{m,n\})^*$, $\ell \in N$, $t \in I_0\backslash\{|\}$, $U \in I_0^*$, dan gaat de invoer $Ts|^k{\downarrow}mXn|^{\ell}tU$ in zichzelf als uitvoer over met de uitgang "ja" als $k = \ell$ en de uitgang "neen" als $k \neq \ell$.

§ 6. Universele Turing-machine. Stopprobleem

Tot nu toe moest voor elk probleem een aparte Turing-machine worden gemaakt. We kunnen nu echter een universele Turing-machine U maken, die de werking van alle Turing-machines kan nabootsen. We beginnen met een grove schets.

De werking van een Turing-machine T wordt vastgelegd door zijn machinerij, dat is een symboolrij, die we op de band van U zetten. De actie van T wordt beschreven met opeenvolgende momentane beschrijvingen. Een momentane beschrijving is ook een symboolrij, die we op de band van U zetten. U moet nu het volgende doen. In de momentane beschrijving van T zoekt hij de geldende toestand en het symbool in het aangewezen hokje. Daarna zoekt hij in de machinerij van T naar een quadrupel, dat met het overeenkomstige tweetal begint. Als dat niet gevonden wordt, stopt de machine. Als het wel gevonden wordt, gaat hij de momentane beschrijving van T wijzigen in overeenstemming met het voorschrift, dat door het laatste tweetal symbolen van het quadrupel wordt aangegeven.

Nu is U een vaste machine met een vaste verzameling bandsymbolen en een vaste verzameling toestandssymbolen. Om machinerij en momentane beschrijving van T weer te geven moeten bandsymbolen en toestandssymbolen van T op de band van U kunnen worden weergegeven voor alle Turing-machines T. Dit is alleen mogelijk als we symbolen van T weergeven met symboolrijen van U. We doen dit door te nummeren. We nemen aan, dat O en | tot de bandsymbolen van U behoren (naast eventueel andere symbolen). Het symbool O dient tevens als scheidingssymbool: natuurlijke getallen, elk van de vorm $O|^k$, kunnen gewoon achter elkaar in een rij gezet worden zonder gevaar voor verwarring. We maken nu een vertaling door aan elk toestandssymbool en elk bandsymbool van T een verschillend natuurlijk getal toe te voegen. We spreken af, dat we ook aan de symbolen b, & en r getallen toevoegen en wel vaste, die onafhankelijk zijn van de keuze van T. Ook wordt aan de begintoestand bij alle T hetzelfde getal toegevoegd. Zo krijgt ook iedere symboolrij een vertaling, die een rij achter elkaar geplaatste natuurlijke getallen is. Op de band van U zetten we nu de vertaling van de machinerij van T, links en rechts geflankeerd door een beschermend markeersymbool. Van een momentane beschrijving AqC zetten we de vertaling van A links naast de machinerij en de vertaling van qC rechts naast de machinerij. Het aangewezen hokje is dat waar de O van de geldende toestand in staat. Schematisch weergegeven:



Het eerste gedeelte van de actie van U is het zoeken van een passend quadrupel in de machinerij. Dit is niet het zoeken van goede symbolen, maar van goede symboolrijen (nummers). De lengte van die symboolrijen is niet a priori begrensd, het aantal toestanden van U wel. Het is dus alleen uitvoerbaar met heen en weer lopen en markeren; dat is hetgeen in een getalvergelijkingsdiagram geschiedt. Er moet verder mod 4 met de getallen van de machinerij worden gewerkt. Als geen goed quadrupel wordt gevonden stopt de machine. Als wel een goed quadrupel wordt gevonden, begint de machine aan het tweede gedeelte van zijn actie.

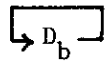
Het tweede gedeelte van de actie van U bestaat in het tot uitvoering brengen van de in het derde en vierde symboolnummer van het gevonden quadrupel besloten opdracht. Hierbij kan een links- of rechtsschuifopdracht zijn. Deze komt neer op het verwisselen van twee symboolrijen op de band van U; dit kan door herhaaldelijk toepassen van een verschuiver worden verwezenlijkt. Een schrijfopdracht en de opdracht tot wijzigen van de geldende toestand komen beide neer op het vervangen van een getal op de band door een elders op de band vermeld ander getal. Dit zou met een copieermachine kunnen worden uitgevoerd, ware het niet, dat het te copiëren getal als symboolrij in het algemeen een andere lengte heeft dan het getal, waarvoor het in de plaats moet worden gesteld. Dit wordt opgelost door zonodig hetgeen rechts van de copieerplaats staat naar rechts of naar links te schuiven om dat stuk ongeschonden te bewaren en een aaneengesloten symboolrij te behouden. Zonodig moeten links of rechts vertalingen van b worden toegevoegd of weggelaten. Aan het eind van de actie wordt het aangewezen hokje weer het hokje dat de O van de geldende toestand bevat en begint de machine opnieuw.

We bewijzen niet, dat hetgeen hierboven is beschreven, inderdaad met een Turing-machine kan worden verwezenlijkt.

Men vat de universele Turing-machine vaak op als idealisering van bestaande rekenautomaten exclusief invoer- en uitvoervoorzieningen. Een specifieke Turing-machine T correspondeert dan met een programma en wel is de machinerij van T het programma en bevat de invoer van T de bij dit programma

behorende invoergegevens. Beide worden, vertaald in een voor U bruikbare code, op de band van U gezet, die correspondeert met het geheugen van de rekenautomaat. De toestanden van U corresponderen met het besturend orgaan. Dit handelt in overeenstemming met de op de band staande gegevens en instructies en levert een nieuwe bandbeschrijving af, die rijp is voor uitvoer. De onbegrensde geheugencapaciteit van U is een idealisatie van de werkelijke situatie in een rekenautomaat.

Het soms niet stoppen van een Turing-machine is een hinderlijke omstandigheid. Of deze optreedt of niet kan niet door proberen worden vastgesteld. We zouden prijs stellen op een effectieve methode om voor iedere combinatie van Turing-machine en invoer uit te maken of de machineberekening zal stoppen of niet. Een effectieve methode betekent hier de constructie van een Turing-machine, die het doet. We proberen een machine T_h te maken, op welks band de machinerij en de invoer van alle Turing-machines T kunnen worden gezet op dezelfde wijze als dit bij U geschiedt. We willen nu, dat als T op een zekere invoer stopt, T_h op de vertaling van T en van die invoer stopt met | in het aangewezen hokje, en als T op een zekere invoer niet stopt, T_h stopt met O in het aangewezen hokje. Een dergelijke machine T_h zou de oplossing leveren van het stopprobleem (engels: halting problem) voor Turing-machines. We bewijzen echter, dat een dergelijke machine T_h niet bestaat. Daartoe nemen we aan, dat T_h wel bestaat. We maken de afspraak, dat bij alle vertalingen van Turing-machines, die O en | als bandsymbool hebben, voor deze symbolen dezelfde vertalingen worden gebruikt. Verder hebben we als hulpmiddel ook nog een nooit-stoppende machine O nodig, die op alle invoeren een niet-stoppende berekening produceert. Zulk een machine kan bijvoorbeeld gerealiseerd worden met het diagram



We maken een machine T_h' en beschrijven, wat deze doet met een invoer die bestaat uit q_0 gevolgd door een symboolrij, die slechts de symbolen O en | bevat. T_h' gaat eerst voor en achter deze rij het markeersymbool zetten, dat op U en T_h de machinerij markeert, vervolgens plaatst hij rechts van de zo verkregen rij achtereenvolgens de vertaling van de begintoestand en de vertaling van de symboolrij, die aanvankelijk op de band van T_h' was geplaatst;

daarna gaat hij naar het hokje waar de O van de begintoestand in staat en geeft uitvoering aan het diagram

$$T_h \overset{|}{\rightarrow} O \ .$$

We bewijzen niet, dat er inderdaad een machine T_h' bestaat, die dit realiseert. Als T_h' is geconstrueerd, maken we een vertaling van de symbolen van T_h'. Laat R de vertaling zijn van de machinerij van T_h'. We geven nu aan T_h' de invoer q_0R. Het eerste deel van de berekening van T_h' tot aan de introductie van het diagram construeert op de band precies de invoer die op T_h moet worden gezet om het stopprobleem van de machine T_h' met de invoer q_0R te behandelen.

Veronderstel nu, dat T_h' op de invoer q_0R stopt. Dan zal bij intrede in het diagram de machine T_h stoppen met | in het aangewezen hokje, waarna het diagram verder gaat met de nooit-stoppende machine O en T_h' dus op de invoer q_0R niet stopt. Deze tegenstrijdigheid toont aan, dat de veronderstelling, dat T_h' op de invoer q_0R stopt, onjuist is.

Dus T_h' stopt niet op de invoer q_0R. Dan zal bij intrede in het diagram de machine T_h stoppen met O in het aangewezen hokje, waarna ook het diagram stopt en T_h' dus op de invoer q_0R stopt. Deze tegenstrijdigheid kan alleen zijn ontstaan doordat de veronderstelling, dat T_h bestaat, onjuist is. Men drukt het verkregen resultaat wel als volgt uit:

Het stopprobleem voor Turing-machines is niet beslisbaar.

We hebben echter nog geen definitie van beslisbaar gegeven en men kan zich afvragen of er geen andere methode bestaat om het probleem aan te pakken, die wel tot een beslissing leidt. Dit blijkt echter niet het geval te zijn; daarbij wordt uiteraard gebruik gemaakt van de these van Church.

Men trekt uit de bewering, dat het stopprobleem voor Turing-machines niet beslisbaar is, wel eens de conclusie, dat het niet mogelijk is om voor een rekenautomaat een programma te maken, dat in staat is voor willekeurige programma's en invoergegevens te testen of de berekening stopt of niet. Er zijn twee omstandigheden, die het twijfelachtig maken of deze conclusie gerechtvaardigd is. Beide hebben iets te maken met de idealisatie van het potentieel oneindige.

1. De geheugencapaciteit van een rekenautomaat is niet onbegrensd. In werkelijkheid is er een bovengrens, waardoor de machine meer overeenkomt met een eindige machine dan met een Turing-machine. Voor een eindige machine is het stopprobleem echter wel beslisbaar.
2. In het begrip beslisbaar zelf zit ook een potentieel oneindig opgesloten. Dat een probleem beslisbaar is zegt, dat er een effectieve procedure is, die in een eindig aantal stappen een beslissing levert. Over het aantal stappen wordt niet gesproken; dit hangt uiteraard van de gegevens af. Bij het stopprobleem zijn dat de omvang van het geheugen, het programma en de invoergegevens. Het zou kunnen, dat als deze gegevens een hanteerbare omvang hebben, het aantal stappen van een corresponderende beslissingsprocedure toch onhanteerbaar groot is.

Voor een machine met een onbegrensde geheugencapaciteit is het stopprobleem zelfs theoretisch onbeslisbaar. Voor een machine met een begrensde, doch zeer grote geheugencapaciteit, is het stopprobleem weliswaar theoretisch beslisbaar, maar zal de omvang (en dus de tijdsduur) van een beslissingsprocedure onhanteerbaar groot zijn. In hoeverre dit in de praktijk zo is, hangt ervan af, in hoeverre mag worden aangenomen, dat de geheugenomvang groot is vergeleken bij hanteerbare aantallen realiseerbare machine-acties. Klaarblijkelijk mag worden aangenomen, dat dit inderdaad het geval is; dit rechtvaardigt de keuze van de Turing-machine als geïdealiseerd model voor de rekenautomaat.

Uit de onbeslisbaarheid van het stopprobleem voor Turing-machines kan de onbeslisbaarheid van allerlei analoge problemen worden afgeleid. We gaan daarop hier niet in.

§ 7. Turing-berekenbare functies

In § 4 hebben we gezien dat een Turing-machine een partiële afbeelding $I^* \to I^*$ oplevert. Om een partiële functie $N \to N$ uit te rekenen nemen we een Turing-machine, waarvoor O en | tot de bandsymbolen behoren (een dergelijke machine moemen we een (O,|)-Turing-machine). We kunnen deze een natuurlijk getal als invoer geven: $O|^n$. Ook als de machine stopt, hoeft de uitvoer U geen natuurlijk getal als d(U) op te leveren.

Definitie partieel Turing-berekenbaar.
Een partiële functie f: N → N heet partieel Turing-berekenbaar, als er een (O,|)-Turing-machine T bestaat zodat voor iedere n ∈ N geldt, dat als f(n) niet gedefinieerd is, T niet stopt op de invoer ↓O|^n en als f(n) gedefinieerd is, T op de invoer ↓O|^n de uitvoer ↓O|$^{f(n)}$ levert.

Het is uiteraard toegestaan, dat de functie voor alle n ∈ N gedefinieerd is. In dat geval spreekt men van een totale functie.

Aan een willekeurige (O,|)-Turing-machine kan men ook ↓O|^n als invoer geven. Als de machine stopt, geeft hij een uitvoer. Om van die uitvoer een natuurlijk getal te maken tellen we het aantal streepjes, dat erin voorkomt. Dat gaat formeel met de volgende afbeelding d': $(I_0 \cup S)^* \to N$:

$$d'(\Lambda) := 0 ,$$

$$d'(A\alpha) := \begin{cases} d'(A) & \text{als } \alpha \neq | , \\ d'(A) + 1 & \text{als } \alpha = | , \end{cases}$$

hierin $A \in (I_0 \cup S)^*$, $\alpha \in I_0 \cup S$.

Definitie bijbehorende partiële functie.
Als T een (O,|)-Turing-machine is, dan is de bijbehorende partiële functie N → N voor n ∈ N als volgt gedefinieerd:
als T op de invoer ↓O|^n niet stopt: functie is niet gedefinieerd,
als T op de invoer ↓O|^n stopt met uitvoer U: functie is gedefinieerd met waarde d'(U).

Uiteraard geldt voor de Turing-machine T uit de definitie van partieel berekenbaar, dat f de bij T behorende partiële functie is. Omgerekend is echter ook de bij een Turing-machine behorende partiële functie partieel Turing-berekenbaar:

Stelling 7.1. Als T een (O,|)-Turing-machine is, dan is de bij T behorende partiële functie partieel Turing-berekenbaar.

We geven een schets van het bewijs van stelling 7.1. Kies eerst nieuwe markeersymbolen m_ℓ en m_r. We willen een Turing-machine T' maken, die het volgende doet. Eerst wordt de invoer $\dagger O|^n$ omgezet in $m_\ell \dagger O|^n m_r$. Daarna wordt hierop T toegepast, echter met bewaring links en rechts van m_ℓ resp. m_r. Omdat aan beide zijden slechts één symbool bewaard moet worden, is daarvoor geen volledige bewaarmachine nodig. Dit levert, als U de uitvoer van T is, een uitvoer van de gedaante $m_\ell b^k U b^{k'} m_r$ met $k \in N$, $k' \in N$. Daarna moeten tussen m_ℓ en m_r alle streepjes opgezocht worden en in een blok bij elkaar gezet worden met een rondje ervoor, waarna al het overige moet worden uitgeveegd. Dit is uitvoerbaar, omdat slechts tussen twee gegeven markeersymbolen behoeft te worden gezocht. We bewijzen niet, dat een dergelijke Turing-machine T' bestaat.

We kunnen ook functies van meer variabelen beschouwen. In § 9 bespreken we een methode om deze tot functies van één variabele terug te brengen. Men kan ze echter ook rechtstreeks met een (O,|)-Turing-machine berekenen. Om $f(n_0,\ldots,n_{k-1})$ uit te rekenen voor $n_0 \in N,\ldots,n_{k-1} \in N$ geven we aan de machine de invoer $\dagger O|^{n_0} O|^{n_1} \ldots O|^{n_{k-1}}$. Evenals boven kunnen we dan weer eisen, dat de uitvoer de gedaante $\dagger O|^{f(n_0,\ldots,n_{k-1})}$ heeft, maar we kunnen ook in een willekeurige uitvoer de streepjes tellen. Dit gaat analoog als in het geval dat $k = 1$.

Ook $k = 0$ is mogelijk. Dan is er alleen de invoer $\dagger$. Een totale functie van nul variabelen is een constante. Er is echter ook een niet-gedefinieerde partiële functie van nul variabelen corresponderend met een machine, die op de invoer $\dagger$ niet stopt.

Tenslotte beschouwt men ook predicaten. Een <u>predicaat</u> P van k variabelen wordt geschreven $P(x_0,\ldots,x_{k-1})$. Voor sommige k-tupels van natuurlijke getallen is het predicaat waar, voor de andere onwaar. Bij een predicaat P behoort een <u>karakteristieke functie</u> χ_P, gedefinieerd door

$$\chi_P(n_0,\ldots,n_{k-1}) := \begin{cases} 1 & \text{als } P(n_0,\ldots,n_{k-1}) \text{ waar is,} \\ 0 & \text{als } P(n_0,\ldots,n_{k-1}) \text{ onwaar is.} \end{cases}$$

Dit is een totale functie. Bij een predicaat hoort ook een verzameling, namelijk de verzameling van de k-tupels waarvoor het waar is. De bijbehorende karakteristieke functie noemen we ook de karakteristieke functie van de verzameling. Als $k = 1$, is de verzameling een verzameling natuurlijke getallen.

Definitie Turing-beslisbaar.
Een verzameling natuurlijke getallen heet Turing-beslisbaar, als haar karakteristieke functie Turing-berekenbaar is.

§ 8. Gödel-nummering

Het definiëren van beslisbaarheid uitsluitend voor verzamelingen natuurlijke getallen is te rechtvaardigen, als we verzamelingen symboolrijen omzetten in verzamelingen natuurlijke getallen. Dit doen we door symboolrijen te nummeren.

Laat I een eindige verzameling symbolen zijn. De verzameling van alle symboolrijen gevormd uit symbolen van I is aftelbaar. We willen dat aftellen effectief doen; dit behandelen we eerst intuïtief. We proberen eerst een lijst van de symboolrijen te maken en gaan deze lijst vervolgens nummeren. We maken gebruik van een lexicografische ordening. Daartoe bepalen we eerst een volgorde voor de symbolen (alfabet). Om de volgorde van twee symboolrijen te bepalen zoekt men van links naar rechts naar het eerste verschillende symbool en ordent volgens de ordening van die symbolen ("aandeel" komt voor "aap", omdat n voor p komt in het alfabet). Het systeem werkt niet, als de ene symboolrij een beginstuk is van de andere: "aan" en "aandeel". We spreken af dat in dat geval de kortere voor de langere gaat: "aan" voor "aandeel". Samenvatting:

1. lexicografisch,
2. korter voor langer.

Hierbij wordt in eerste instantie volgens 1 geordend en alleen als dat geen uitsluitsel geeft volgens 2.

Deze ordening is voor ons niet bruikbaar, omdat de verzameling van alle symboolrijen op grond van deze ordening niet in een lijst kan worden gezet. Bij het gewone alfabet zou een woord, dat met een b begint, voorafgegaan moeten worden door alle oneindig veel woorden, die met een a beginnen. We lossen de moeilijkheid op door de prioriteit te veranderen:

1. korter voor langer,
2. lexicografisch.

We ordenen eerst volgens 1 en als dat niet werkt volgens 2. Als k het aantal symbolen is, komen in de lijst de opeenvolgende waarden ℓ van de lengte van

de symboolrij in lexicografisch geordende blokken van k^{ℓ} symboolrijen aan de orde. Hieruit blijkt, dat iedere symboolrij aan de beurt komt. Deze lijst gaan we nummeren; de lege symboolrij wordt ook meegeteld.

Laat I een verzameling van k symbolen zijn ($k \geq 1$). We ordenen deze door ze te nummeren; we noemen ze $a_0,\ldots,a_{k-1}$. Maakt men een lijst op de bovenbeschreven wijze van de elementen van I^* en nummert men deze lijst, dan noemen we $\varphi(W)$ het nummer, dat $W \in I^*$ krijgt. Dan is $\varphi(W)$ het aantal eventueel-lege symboolrijen, dat in de lijst aan W voorafgaat. Hiervoor geldt de volgende bewering.

Als $W \in I^*$, $W' \in I^*$, $j \in N$, $j < k$, dan gaat $W'a_j$ in de lijst vooraf aan Wa_0 dan en slechts dan als W' in de lijst voorafgaat aan W.

Om deze bewering te bewijzen onderscheiden we twee gevallen.

1. $\ell(W) \neq \ell(W')$. Dan ook $\ell(Wa_0) \neq \ell(W'a_j)$. In beide gevallen wordt de volgorde dus op grond van de lengte bepaald. Beide uitspraken over de volgorde gelden dan en slechts dan als $\ell(W') < \ell(W)$.
2. $\ell(W) = \ell(W')$. Dan ook $\ell(Wa_0) = \ell(W'a_j)$. In beide gevallen wordt de volgorde dus lexicografisch bepaald. Daar a_j niet voorafgaat aan a_0, gaat $W'a_j$ vooraf aan Wa_0 dan en slechts dan als W' voorafgaat aan W.

Uit bovenstaande bewering volgt makkelijk, dat $\varphi(Wa_0) = k\varphi(W) + 1$, en daaruit dat $\varphi(Wa_j) = k\varphi(W) + j + 1$.

Het bovenstaande beschouwen we als een intuïtieve inleiding. We geven nu een formele definitie van φ door recurrentie langs de opbouw van een symboolrij; daarbij is verondersteld, dat de nummering $a_0,\ldots,a_{k-1}$ der symbolen gegeven is.

Definitie φ.
I is een niet-lege verzameling symbolen; $a_0,\ldots,a_{k-1}$ is een nummering van de elementen van I.
De afbeelding $\varphi: I^* \to N$ wordt als volgt gedefinieerd:
Als $W \in I^*$, $j \in N$ en $j < k$, dan

$$\varphi(\Lambda) := 0 ,$$
$$\varphi(Wa_j) := k\varphi(W) + j + 1 .$$

Stelling 8.1. Als $W \in I^*$ en $X \in I^*$, dan

$$\varphi(WX) = k^{\ell(X)}\varphi(W) + \varphi(X) .$$

Het bewijs wordt geleverd door recurrentie langs de opbouw van X.

Uit de intuïtieve interpretatie van φ zien we, dat de volgende eigenschappen juist zijn:

1. Als $W \in I^*$, dan $\varphi(W) \in N$.
2. Als $W \in I^*$, $W' \in I^*$ en $\varphi(W) = \varphi(W')$, dan $W = W'$.
3. Als $n \in N$, dan bestaat er een $W \in I^*$, zodat $\varphi(W) = n$.

Eigenschap 1 is ook makkelijk formeel te bewijzen (recurrentie langs de opbouw van W). Voor het bewijs van 2 en 3 bepalen we de inverse functie van φ, die we ψ noemen. Eerst bekijken we intuïtief hoe deze er moet uitzien.

Stel $\varphi(Wa_j) = x$ en $\varphi(W) = y$, dan $\psi(x) = Wa_j$ en $\psi(y) = W$, $\psi(x) = \psi(y)a_j$, $x = ky + j + 1$. Dit wijst in de richting van delen door k met rest. Omdat k vast is, nemen we de afhankelijkheid van k niet in de notatie op. Uit de elementaire rekenkunde is het volgende bekend (q is quotiënt, r is rest):

Stelling 8.2. Gegeven $k \in N$, $k \neq 0$. Er bestaan functies $q: N \to N$ en $r: N \to N$, zodat voor alle $n \in N$:

$$n = kq(n) + r(n) \quad \text{en} \quad r(n) < k .$$

Verder als $n \in N$, $\ell \in N$, $m \in N$, $n = k\ell + m$, $m < k$, dan $\ell = q(n)$, $m = r(n)$.

De functie q heeft nog de eigenschap, dat uit $n \in N$ volgt $q(n) \leq n$, immers $n = kq(n) + r(n) \geq kq(n) \geq q(n)$.

In het bovenstaande wordt $x - 1$ door k met rest gedeeld: als $x \neq 0$, dan $y = q(x - 1)$, $j = r(x - 1)$. *Dit leidt tot de volgende formele definitie.*

Definitie ψ.
I is een niet-lege verzameling symbolen; $a_0, \ldots, a_{k-1}$ is een nummering van de elementen van I.
De afbeelding $\psi: N \to I^*$ wordt als volgt gedefinieerd:
Als $x \in N$, $x \neq 0$, dan

$$\psi(0) := \Lambda ,$$
$$\psi(x) := \psi(q(x - 1))a_{r(x-1)} .$$

Dit is een definitie door volledige inductie, want $q(x - 1) \leq x - 1 < x$.

Stelling 8.3. Als $n \in N$, dan $\varphi(\psi(n)) = n$, als $W \in I^*$, dan $\psi(\varphi(W)) = W$.

Het bewijs van stelling 8.3 wordt geleverd door volledige inductie naar n en door recurrentie langs de opbouw van W.

Uit stelling 8.3 volgen nu makkelijk bovenstaande eigenschappen 2 en 3. Uit $\varphi(W) = \varphi(W')$ volgt $W = \psi(\varphi(W)) = \psi(\varphi(W')) = W'$. Als $n \in N$, dan kiezen we $W = \psi(n)$, dan $\varphi(W) = \varphi(\psi(n)) = n$.

Een bezwaar van bovenstaande nummering is dat, als we voor een verzameling I van symbolen een nummering van I^* hebben ingevoerd en we breiden daarna I uit tot een grotere verzameling symbolen, er een geheel nieuwe nummering ontstaat, waarin ook de elementen van I^* andere nummers krijgen dan ze eerst hadden. We zouden liever een nummering hebben, die bij uitbreiding van de verzameling symbolen kan worden voortgezet, waarbij de oude symboolrijen dezelfde nummers houden. Dit is echter onmogelijk, als aan bovengenoemde eigenschappen 1, 2 en 3 voldaan moet zijn. Immers, als de nummering aan deze eigenschappen voldoet en we zouden deze willen voortzetten tot rijen met nieuwe symbolen, zodat ook aan deze eigenschappen voldaan is, dan komen we tot tegenstrijdige eisen. Nemen we namelijk een symboolrij, waar een nieuw symbool in voorkomt, dan krijgt deze volgens 1 een natuurlijk getal als nummer. Op grond van 3 is er echter ook een oude symboolrij met datzelfde nummer, in strijd met 2. Terwille van de bruikbaarheid willen we nu 1 en 2 handhaven. We zullen dan 3 moeten laten vallen.

Als we aan de symbolen natuurlijke getallen toekennen, dan correspondeert met een symboolrij een rij natuurlijke getallen. Er is nu behoefte aan een afbeelding van de verzameling der eindige rijen natuurlijke getallen in de verzameling der natuurlijke getallen. Dit is een andere afbeelding dan die voor symboolrijen, omdat er oneindig veel natuurlijke getallen zijn en eindig veel symbolen. Zo is het aantal symboolrijen van vaste lengte ℓ eindig (k^ℓ) en het aantal rijen natuurlijke getallen van vaste lengte oneindig. Nemen we nu $I = \{O, |\}$, dan kunnen in I^* niet alleen natuurlijke getallen, maar zelfs rijen natuurlijke getallen weergegeven worden door ze gewoon achter elkaar te schrijven; de rij 3,2,0,5 wordt O|||O||O|||||. We kunnen I^* op de tevoren behandelde wijze met φ nummeren, hetgeen ook een nummering voor rijen natuurlijke getallen oplevert. Om technische redenen brengen we

nog een wijziging aan. Een symboolrij, die met een rij natuurlijke getallen correspondeert, begint altijd met een O; we knippen dit overbodige symbool eraf. Dan is $\Lambda \in I^*$ beeld van een rij, die alleen uit 0 bestaat. Omgekeerd geeft een rij van ℓ natuurlijke getallen een element van I^*, waarin $\ell - 1$ rondjes voorkomen, dus $\ell \geq 1$. Er komt een eeneenduidige correspondentie tussen de verzameling der niet-lege rijen natuurlijke getallen en I^*.

We gebruiken de functie φ voor $I = \{O, I\}$; $k = 2$. We kiezen $a_0 := I$ en $a_1 := O$. We willen nu rechtstreeks de afbeelding voor rijen natuurlijke getallen beschrijven.
Als $n \in N$, dan $\varphi(I^n) = 2^n - 1$ (volledige inductie naar n).
Als $n \in N$, dan $\varphi(OI^n) = 3.2^n - 1$ (stelling 8.1).
Als r een rij natuurlijke getallen is (geschreven met komma's tussen de getallen), dan duiden we de corresponderende symboolrij aan met $\bar{r}$.
Als r een rij natuurlijke getallen is en n een natuurlijk getal, dan

$$\varphi(\overline{r,n}) = \varphi(\bar{r}OI^n) = 2^{n+1}\,\varphi(\bar{r}) + 3.2^n - 1 ,$$

$$\varphi(\bar{n}) = \varphi(I^n) = 2^n - 1 .$$

De afbeelding voor rijen natuurlijke getallen zullen we gn noemen. Omdat we ook graag de lege rij een nummer willen geven, stellen we $gn(r) := \varphi(\bar{r}) + 1$. Dan geldt

$$gn(\Lambda) := 0 ,$$

$$gn(r,n) = \varphi(\overline{r,n}) + 1 = 2^{n+1}\,\varphi(\bar{r}) + 3.2^n = (2gn(r) + 1)2^n .$$

Dit laatste klopt ook als $r = \Lambda$, want $gn(n) = \varphi(\bar{n}) + 1 = 2^n = (2gn(\Lambda) + 1)2^n$.

Na deze intuïtieve inleiding geven we nu een formele definitie. We schrijven rijen natuurlijke getallen met komma's ertussen, we schrijven N^* voor de verzameling der eventueel-lege rijen natuurlijke getallen en we gebruiken Λ ook voor de lege rij natuurlijke getallen.

Definitie gn.
De afbeelding gn: $N^* \to N$ wordt als volgt gedefinieerd:
Als $r \in N^*$, $n \in N$, dan

$$gn(\Lambda) := 0 ,$$

$$gn(r,n) := (2gn(r) + 1)2^n .$$

We willen nu ook de inverse functie van gn, die we rj noemen, expliciet maken. We doen dit eerst intuïtief.

Stel $gn(r,n) = x$ en $gn(r) = y$, dan $rj(x) = r,n$ en $rj(y) = r$, $rj(x) = rj(y),n$ en $x = (2y + 1)2^n$. Hierbij treedt op de schrijfwijze van een natuurlijk getal als een product van een oneven getal en een macht van 2. Dit kan echter alleen voor getallen $\neq 0$, maar levert dan een correspondentie tussen getallen x en paren getallen (y,n), die eeneenduidig op is. Om ook het getal 0 erbij te brengen verhogen we het getal eerst met 1 alvorens het als product te schrijven. Het voorste element van het paar noemen we hd (engels: head) en het achterste element $t\ell$ (engels: tail). Uit de elementaire rekenkunde is het volgende bekend:

Stelling 8.4. Er bestaan functies $hd: N \to N$ en $t\ell: N \to N$, zodat voor alle $n \in N$:

$$n = (2\ hd(n) + 1)2^{t\ell(n)} - 1 .$$

Verder als $n \in N$, $k \in N$, $\ell \in N$, $n = (2k + 1)2^{\ell} - 1$, dan $k = hd(n)$, $\ell = t\ell(n)$.

De functie hd heeft nog de eigenschap, dat uit $n \in N$ volgt $hd(n) \leq n$, immers $n = (2\ hd(n) + 1)2^{t\ell(n)} - 1 \geq 2\ hd(n) \geq hd(n)$.

In het bovenstaande wordt $y = hd(x - 1)$, $n = t\ell(x - 1)$ voor $x \neq 0$. Dit leidt tot de volgende formele definitie.

Definitie rj.
De afbeelding $rj: N \to N^*$ wordt als volgt gedefinieerd:
Als $x \in N$, $x \neq 0$, dan

$$rj(0) := \Lambda ,$$

$$rj(x) := rj(hd(x - 1)),t\ell(x - 1) .$$

Dit is een definitie door volledige inductie, want $hd(x - 1) \leq x - 1 < x$.

Stelling 8.5. Als $n \in N$, dan $gn(rj(n)) = n$, als $r \in N^*$, dan $rj(gn(r)) = r$.

Men bewijst dit door volledige inductie naar n en naar het aantal getallen in de rij r.

Als I een eindige verzameling symbolen is, kan men aan elk symbool een natuurlijk getal toekennen, zodat met verschillende symbolen verschillende getallen corresponderen. Met $W \in I^*$ correspondeert dan een $r \in N^*$. Het getal gn(r) heet dan het Gödel-nummer van W; dit hangt uiteraard af van de getallen, die aan de symbolen zijn toegekend. Men kan bij een natuurlijk getal n ook terugzoeken. Daartoe bepaalt men rj(n), dit is een rij natuurlijke getallen. Als in die rij een getal voorkomt, dat niet met een symbool correspondeert, dan is n geen Gödel-nummer. Als alle getallen in de rij wel met symbolen corresponderen, dan heeft de bijbehorende symboolrij n als Gödelnummer.

Uitbreiding van I tot I_1 kan geen kwaad, mits men aan de elementen van I dezelfde getallen blijft toekennen: het Gödel-nummer van een symboolrij hangt alleen af van de getallen, die zijn toegekend aan symbolen, die werkelijk in de rij voorkomen.

We maken nog drie opmerkingen.

1. Aan een symbool is een natuurlijk getal toegekend. Het Gödel-nummer van de symboolrij van lengte 1, die uit dat symbool bestaat, is een ander getal. Als het eerste getal j is, dan is het tweede getal 2^j.
2. Men zou zich kunnen afvragen, waarom een zo elementaire aangelegenheid als deze nummering de naam van zulk een beroemd man draagt. De verdienste van Gödel is niet het bedenken van een nummering geweest, maar het vernuftige gebruik, dat hij ervan gemaakt heeft bij het bewijzen van allerminst elementaire stellingen.
3. De hierboven gedefinieerde nummering is een andere dan de originele van Gödel, die berust op de ontbinding van natuurlijke getallen in priemfactoren.

We passen het bovenstaande toe op Turing-machines. Bij de formele definitie maken we gebruik van verzamelingen I en S van symbolen plus de losse symbolen b, ℓ, r. Over de toekenning van getallen aan deze symbolen maken we de volgende afspraken.

Aan de elementen van S worden oneven getallen toegekend, aan de begintoestand het getal 1. Verder krijgt ℓ het getal 2, r het getal 4, b het getal 6 en de elementen van I even getallen ≥ 8. Voor de symbolen O en | maken we de afspraak, dat O het getal 8 en | het getal 10 krijgt. De machinerij krijgt dan een Gödel-nummer.

Een Turing-machine tezamen met een toekenning van getallen aan zijn symbolen noemen we een genummerde Turing-machine. We spreken dan van het Gödel-nummer van een genummerde Turing-machine.

Aan een momentane beschrijving kan nu ook een Gödel-nummer worden toegekend. Om een berekening als een symboolrij op te vatten hebben we een scheidingssymbool nodig. Aan dit symbool kennen we het getal 0 toe. Dan heeft ook een toegelaten lijst van momentane beschrijvingen en dus ook een machineberekening een Gödel-nummer.

§ 9. Recursieve functies. Normaalvorm van Kleene

De partieel recursieve functies zijn functies van k variabelen ($k \in N$), die natuurlijke getallen als waarden kunnen aannemen en die, als de functie voor die waarden van de variabelen gedefinieerd is, een natuurlijk getal als functiewaarde hebben. Ze worden met een recurrente opzet gekarakteriseerd: een verzameling uitgangsfuncties, een stel formatieregels, die uit functies nieuwe functies maken, en een beperkende bepaling. We maken dit nu nog niet expliciet.

Een functie wordt vaak met één symbool aangeduid. Wij willen echter graag zien van hoeveel variabelen de functie afhangt en bovendien kunnen manipuleren met die variabelen. We voeren daartoe loze variabelen $x_0, x_1, \ldots$ in en schrijven $f(x_0, x_1, x_2)$ voor een functie van drie variabelen. We gebruiken ook wel de letters x, y, z.

We drukken ons wel eens slordig uit en spreken over de functie y^x. We moeten eigenlijk de volgorde der variabelen vaststellen: $f(x,y) := y^x$. Soms hebben we met drie variabelen x, y, z te maken en vatten dan y^x op als functie van x, y, z; dat is een andere functie $g(x,y,z) := f(x,y)$. Ook is y^x als functie van x, y een andere dan als functie van y, x; $g(x,y) := f(y,x)$. Men kan ook variabelen gelijk stellen: als $f(x,y)$ gegeven is, $g(x) := f(x,x)$.

Bovenstaande overgangen, die van recursieve functies weer recursieve functies maken, zijn formeel uit te drukken met samengestelde functies en enkele eenvoudige functies.

Definitie samengestelde functie.

Als $g(x_0,\ldots,x_{k-1})$ een partiële functie van k variabelen is en $h_0(x_0,\ldots,x_{n-1}),\ldots,h_{k-1}(x_0,\ldots,x_{n-1})$ zijn k partiële functies van n variabelen, dan heet de partiële functie $f(x_0,\ldots,x_{n-1})$ van n variabelen gedefinieerd door

$$f(x_0,\ldots,x_{n-1}) := g(h_0(x_0,\ldots,x_{n-1}),\ldots,h_{k-1}(x_0,\ldots,x_{n-1}))$$

de samengestelde functie van $g(x_0,\ldots,x_{k-1})$ en $h_0(x_0,\ldots,x_{n-1}),\ldots,h_{k-1}(x_0,\ldots,x_{n-1})$. Dit betekent, dat, voor $a_0 \in N,\ldots,a_{n-1} \in N$, $f(a_0,\ldots,a_{n-1})$ dan en slechts dan gedefinieerd is als $h_0(a_0,\ldots,a_{n-1}),\ldots,h_{k-1}(a_0,\ldots,a_{n-1})$ en $g(h_0(a_0,\ldots,a_{n-1}),\ldots,h_{k-1}(a_0,\ldots,a_{n-1}))$ alle gedefinieerd zijn; het laatste van deze getallen is dan de functiewaarde.

Definitie u_n^j.

Als $n \in N$, $j \in N$ en $j < n$, dan

$$u_n^j(x_0,\ldots,x_{n-1}) := x_j .$$

Men noemt de u_n^j identieke functies.

Als f(x,y) een functie van twee variabelen is, dan is g(x,y,z) := f(x,y) de samengestelde functie $f(u_3^0(x,y,z),u_3^1(x,y,z))$. Evenzo is g(x,y) := f(y,x) te schrijven als $f(u_2^1(x,y),u_2^0(x,y))$ en g(x) := f(x,x) als $f(u_1^0(x),u_1^0(x))$. Hierin is u_1^0 de gangbare identieke functie van één variabele: $u_1^0(x) := x$.

We laten ook functies van 0 variabelen toe. Totale functies zijn dan constanten. Bij de samengestelde functie kan ook k = 0 of n = 0. Het geval k = 0 geeft de overgang van een constante naar een constante functie van n variabelen. Het geval n = 0 geeft de overgang van een functie van k variabelen naar de constante functiewaarde, die ontstaat als we voor de variabelen constanten invullen.

Als we partieel Turing-berekenbare functies en partieel recursieve functies met elkaar willen vergelijken, moeten we ook functies van meer variabelen beschouwen. In § 7 is het begrip Turing-berekenbaar behandeld door voor $f(a_0,\ldots,a_{k-1})$ de invoer $\text{↓O|}^{a_0} \ldots \text{O|}^{a_{k-1}}$ te gebruiken. We kunnen nu echter ook proberen met de functies gn en rj op functies van één variabele terug te vallen. We gebruiken als hulpmiddel de functie ℓ', die aan rijen natuurlijke getallen het aantal getallen in de rij toevoegt:

Definitie ℓ'.
De afbeelding ℓ': $N^* \to N$ wordt als volgt gedefinieerd:
Als $r \in N^*$ en $n \in N$, dan

$$\ell'(\Lambda) := 0 ,$$
$$\ell'(r,n) := \ell'(r) + 1 .$$

Als $f(x_0,\ldots,x_{k-1})$ gegeven is, dan beschouwen we $g(x) := f(rj(x))$. Hiermee is bedoeld, dat $g(a)$ voor $a \in N$ dan en slechts dan gedefinieerd is, als $\ell'(rj(a)) = k$ en $f(rj(a))$ gedefinieerd is. Als we nu $f(x_0,\ldots,x_{k-1})$ partieel Turing-berekenbaar noemen, als $g(x)$ partieel Turing-berekenbaar is, dan komt deze definitie op hetzelfde neer als de definitie van § 7. Dat is makkelijk te bewijzen, als we bedenken, dat $f(x_0,\ldots,x_{k-1}) = g(gn(x_0,\ldots,x_{k-1}))$ en aantonen, dat de volgende Turing-machines bestaan:

Rj : als $n \in N$ en $rj(n) = a_0,\ldots,a_{k-1}$, dan gaat ↓O|^n over in $\text{↓O|}^{a_0} \ldots \text{O|}^{a_{k-1}}$.

Gn : als $k \in N$, $a_0 \in N,\ldots,a_{k-1} \in N$, dan gaat $\text{↓O|}^{a_0} \ldots \text{O|}^{a_{k-1}}$ over in $\text{↓O|}^{gn(a_0,\ldots,a_{k-1})}$.

Rt_k voor $k \in N$: als $\ell \in N$, $a_0 \in N,\ldots,a_{\ell-1} \in N$, dan stopt de machine niet op de invoer $\text{↓O|}^{a_0} \ldots \text{O|}^{a_{\ell-1}}$ als $\ell \neq k$ en gaat op deze invoer in zichzelf over als $\ell = k$.

Om te bewijzen, dat partieel recursieve functies partieel Turing-berekenbaar zijn, moet men laten zien, dat de uitgangsfuncties dat zijn en dat de formatieregels, toegepast op partieel Turing-berekenbare functies, weer partieel Turing-berekenbare functies opleveren. We voeren dat niet uit; het

is niet moeilijk. Het bewijs, dat partieel Turing-berekenbare functies partieel recursief zijn is veel moeilijker. We geven er een schets van en beginnen met functies van één variabele.

We beschouwen het predicaat T(z,x,y) van drie variabelen, dat als volgt is gedefinieerd. Als $c \in N$, $a \in N$, $b \in N$, dan is T(c,a,b) dan en slechts dan waar, als c het Gödel-nummer van een genummerde (O,|)-Turing-machine is en b het Gödel-nummer van een berekening op die machine met invoer $\vdash O|^a$. Intuïtief is dit predicaat beslisbaar. We vormen rj(c) en kijken of die rij getallen kan horen bij de machinerij van een Turing-machine; dit is hoofdzakelijk een kwestie van pariteit. Als dat niet zo is, is T(c,a,b) onwaar. Als het wel zo is, dan is er een genummerde (O,|)-Turing-machine met c als Gödel-nummer, die op de benoeming der symbolen en het toevoegen van irrelevante band- en toestandssymbolen na vastligt. We vormen nu rj(b) en controleren of die getallenrij correspondeert met een machineberekening met invoer $\vdash O|^a$ op die machine. Als dat niet zo is, is T(c,a,b) onwaar; als het wel zo is, is T(c,a,b) waar.

Bij iedere keuze van c en a is er ten hoogste één b, waarvoor T(c,a,b) waar is. Deze b als functie van c en a levert een partiële functie van twee variabelen, die we met μ_y T(z,x,y) aanduiden. We willen de partiële functie echter nog wat veranderen. Bij gegeven Turing-machine levert zij bij gegeven x het Gödel-nummer van de berekening met invoer $\vdash O|^x$. We zouden liever de functiewaarde van de bij de Turing-machine behorende functie hebben. Daarom voeren we de functie U(t) van één variabele in, die aan $n \in N$ toevoegt het aantal plaatsen in de getallenrij rj(n), waar het getal 10 staat en waarvan rechts in deze rij nog precies éénmaal het getal 0 staat. We voeren nu $\Psi(z,x)$ in door:

$$\Psi(z,x) := U(\mu_y\ T(z,x,y)) \qquad \text{(normaalvorm van Kleene)}.$$

Als $c \in N$, $a \in N$ en c is geen Gödel-nummer van een genummerde (O,|)-Turing-machine, dan is $\Psi(c,a)$ niet gedefinieerd; als c dat wel is met bijbehorende functie f(x) en f(a) is niet gedefinieerd, dan is $\Psi(c,a)$ niet gedefinieerd, en als f(a) wel gedefinieerd is, is $\Psi(c,a) = f(a)$.

Als f(x) een partieel Turing-berekenbare functie is, dan is er een (O,|)-Turing-machine, waar f bijhoort. Deze machine kunnen we tot een genummerde machine maken; dan heeft de machine een Gödel-nummer c. Dan is

$\psi(c,x) = f(x)$; d.w.z. beide leden van de gelijkheid zijn niet gedefinieerd of beide leden van de gelijkheid zijn wel gedefinieerd en gelijk.

De partiële functie $\psi(z,x)$ van twee variabelen geeft een opsomming van alle partieel Turing-berekenbare functies van één variabele. Daarbij fungeert de z als opsommingsnummer en de x als variabele in de functie. Dat is ook goed als we voor z een getal c kiezen, dat geen Gödel-nummer van een machine is. Dan is $\psi(c,x)$ voor geen enkele waarde van x gedefinieerd; deze functie is partieel Turing-berekenbaar (nooit-stoppende machine). De opsomming is er één met herhalingen.

Met de operator μ_y hebben we van een predicaat van drie variabelen z, x, y een partiële functie van twee variabelen z, x gemaakt. Dit berustte op de omstandigheid, dat er bij iedere keuze van z, x ten hoogste één y bestond waarvoor het predicaat waar was. We willen de operator uitbreiden tot willekeurige predicaten P(z,x,y). Als P intuïtief beslisbaar is, zouden we y proberen te vinden door in de rij P(c,a,0),P(c,a,1),... achtereenvolgens de waarheid van de predicaten te onderzoeken en te stoppen als er één gevonden wordt, die waar is. Bij een willekeurig predicaat levert dit de kleinste y, waarvoor P(c,a,y) waar is, mits er een y bestaat waarvoor P(c,a,y) waar is. Als dat niet zo is, hebben we een niet-stoppende procedure. We laten nu ook nog de beperking tot drie variabelen vervallen.

Definitie μ-operator.
Als $P(x_0,\ldots,x_k)$ een predicaat van k + 1 variabelen is, $j \in N$, $j \leq k$, dan is $\mu_{x_j} P(x_0,\ldots,x_k)$ de partiële functie van de k variabelen $x_0,\ldots,x_k$ met daaruit x_j weggelaten, die de kleinste x_j levert, waarvoor $P(x_0,\ldots,x_k)$ waar is, zo deze bestaat en anders ongedefinieerd is.

Als $f(x_0,\ldots,x_{k-1})$ een totale functie is, dan kan een predicaat $P(x_0,\ldots,x_{k-1})$ verkregen worden door nul stellen van deze functie:

$P(x_0,\ldots,x_{k-1})$ is waar dan en slechts dan als $f(x_0,\ldots,x_{k-1}) = 0$.

Voeren we de functie $\alpha(x)$ in door:

$$\alpha(0) := 1$$
$$\alpha(x) := 0 \quad \text{voor } x \neq 0 ,$$

dan geldt

$$\chi_P(x_0,\ldots,x_{k-1}) = \alpha(f(x_0,\ldots,x_{k-1})) \ .$$

Nu is $\alpha(x)$ een recursieve functie, dus bij een recursieve functie f hoort op deze wijze een recursief predicaat P (d.w.z. een predicaat met een recursieve karakteristieke functie). Omgekeerd is ook ieder recursief predicaat zo te verkrijgen. Voert men een functie f in door

$$f(x_0,\ldots,x_{k-1}) := \alpha(\chi_P(x_0,\ldots,x_{k-1})) \ ,$$

dan is f recursief en P is waar dan en slechts dan als f = 0.

Men kan nu bewijzen, dat T(z,x,y) een recursief predicaat is en U(t) een recursieve functie. Verder behoort de μ-operator en de vorming van een samengestelde functie tot de formatieregels. Dan is $\Psi(z,x)$ een partieel recursieve functie. Als nu f(x) een partieel Turing-berekenbare functie is, dan bestaat er een $c \in N$, zodat $f(x) = \Psi(c,x)$, dus f(x) is partieel recursief, omdat alle constanten recursief zijn. Daarmee zijn functies van één variabele afgehandeld. Als $f(x_0,\ldots,x_{k-1})$ een partieel Turing-berekenbare functie van k variabelen is, dan is $g(x) := f(rj(x))$ partieel Turing-berekenbaar en dus partieel recursief. Om te bewijzen dat $f(x_0,\ldots,x_{k-1}) = g(gn(x_0,\ldots,x_{k-1}))$ partieel recursief is, is het voldoende om aan te tonen dat de functie $gn_k(x_0,\ldots,x_{k-1})$ van k variabelen, gedefinieerd door $gn_k(x_0,\ldots,x_{k-1}) := gn(x_0,\ldots,x_{k-1})$, recursief is.

Er bestaan verschillende definities van recursieve functies. Er is een gangbare definitie, waarbij het vormen van samengestelde functies en de μ-operator de enige formatieregels zijn. Wij behandelen een andere definitie, waarbij aan deze formatieregels ook nog de primitieve recursie is toegevoegd. We behandelen enkele voorbeelden ter inleiding tot dat begrip.

Als f(x) een functie is, dan kunnen we de n maal geïtereerde functie f^n voor $n \in N$ door volledige inductie als volgt definiëren:

$$f^0(x) := x \ ,$$

$$f^{n+1}(x) := f(f^n(x)) \ .$$

Als we alle geïtereerde functies tegelijk beschouwen, hebben we een functie van twee variabelen, omdat dan n ook een variabele is. Noemen we deze functie g(x,y), dan krijgt deze de volgende definitie:

$$g(x,0) := x ,$$
$$g(x,y+1) := f(g(x,y)) .$$

We schrijven in deze definitie in plaats van y+1 liever s(y), de successorfunctie. Dan krijgt ook de optelling een definitie van hetzelfde type:

$f(x,y) = x + y$ wordt gedefinieerd door $f(x,0) := x$,
$f(x,s(y)) := s(f(x,y))$.

We geven nog enkele voorbeelden van soortgelijke definities:

$f(x,y) = xy$ wordt gedefinieerd door $f(x,0) := 0$,
$f(x,s(y)) := f(x,y) + x$.

$f(x) = x!$ wordt gedefinieerd door $f(0) := 1$,
$f(s(x)) := f(x)s(x)$.

De definities worden geïtereerd toegepast: bij xy wordt de optelling gebruikt, bij x! de vermenigvuldiging.

Definitie primitieve recursie.
Als $g(x_0,\ldots,x_{k-1})$ een totale functie van k variabelen is, $h(x_0,\ldots,x_{k+1})$ een totale functie van k + 2 variabelen en de totale functie $f(x_0,\ldots,x_k)$ van k + 1 variabelen is gedefinieerd door

$$f(x_0,\ldots,x_{k-1},0) := g(x_0,\ldots,x_{k-1}) ,$$
$$f(x_0,\ldots,x_{k-1},s(y)) := h(x_0,\ldots,x_{k-1},y,f(x_0,\ldots,x_{k-1},y)) ,$$

dan is $f(x_0,\ldots,x_k)$ door primitieve recursie ontstaan uit $g(x_0,\ldots,x_{k-1})$ en $h(x_0,\ldots,x_{k+1})$.

Definitie partieel recursieve functie.
Partieel recursieve functies worden gedefinieerd met een recurrente definitie met
uitgangsfuncties: constante 0,
successorfunctie s(x),
de identieke functies $u_k^j(x_0,\ldots,x_{k-1})$ voor $k \in N$, $j \in N$, $j < k$,

formatieregels: vorming van samengestelde functies,
primitieve recursie,
μ-operator.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd is een partieel recursieve functie.

Indien men de μ-operator weglaat, krijgt men de primitief-recursieve functies, die alle totaal zijn.

Voortaan beschouwen we de begrippen partieel Turing-berekenbare functie en partieel recursieve functie als gelijkwaardig.

§ 10. Recursieve en recursief opsombare verzamelingen.

Beslisbaarheid van verzamelingen symboolrijen zou op twee manieren kunnen worden behandeld:

1. Men kan een Turing-machine, die de gegeven symbolen als bandsymbolen heeft, als acceptor gebruiken.
2. Men kan de verzameling door Gödel-nummering terugbrengen op een verzameling natuurlijke getallen en van deze laatste eisen dat ze recursief is.

Beide methoden blijken op hetzelfde neer te komen. Bovendien blijkt in geval 2 de uitkomst niet af te hangen van de keuze van de toekenning van getallen aan symbolen, waar de Gödel-nummering van afhangt. Op grond van 2 beperken we ons tot verzamelingen natuurlijke getallen. Het intuïtieve begrip beslisbaar is dan geformaliseerd tot recursief, hetgeen betekent, dat de karakteristieke functie recursief (of Turing-berekenbaar) is.

We willen nu het begrip opsombaar formaliseren voor verzamelingen natuurlijke getallen. Intuïtief betekent het, dat men een lijst kan maken, die onbepaald doorloopt, en waar alle getallen van de verzameling in voorkomen. Men kan de plaatsen van die lijst nummeren. Het maken van de lijst komt dan neer op het definiëren van een totale functie van één variabele, die de verzameling als beeld heeft. Omdat het effectief uitvoerbaar moet zijn, eisen we dat de functie recursief is.

In een opsomming is men geneigd te denken aan een opsomming zonder herhalingen. We eisen dat evenwel niet. Dit heeft het voordeel, dat ook eindige verzamelingen aan bod komen met uitzondering van de lege verzameling. Het blijkt gewenst de lege verzameling mee te tellen. We noemen zo'n verzameling recursief opsombaar (engels: recursively enumerable).

Definitie recursief opsombaar.
Een verzameling natuurlijke getallen heet recursief opsombaar als zij leeg is of beeld van een totaal recursieve functie van één variabele.

Deze definitie sluit nauw aan bij de intuïtieve betekenis van het begrip. Een andere karakterisering is handiger om eigenschappen van recursief opsombare verzamelingen mee af te leiden. Deze wordt in de volgende stelling geformuleerd.

Stelling 10.1. Een verzameling natuurlijke getallen is recursief opsombaar dan en slechts dan als zij de definitieverzameling is van een partieel recursieve functie van één variabele.

We geven een schets van een bewijs. Stel eerst, dat de verzameling V recursief opsombaar is. Als V leeg is, dan is V definitieverzameling van de nergens gedefinieerde functie, die partieel recursief is (nooit-stoppende machine). Als V niet-leeg is, is er een totaal recursieve functie f(x), die V als beeld heeft. Laat T een Turing-machine zijn, die f(x) berekent. We zoeken een Turing-machine T', die het volgende doet. Als de invoer $\vdash O|^n$ is, bewaart hij dit getal en gaat daarna de berekening van T achtereenvolgens uitvoeren voor de getallen 0,1,2,... . Na elk van deze berekeningen vergelijkt hij de gevonden functiewaarde f(j) met n; als $f(j) \neq n$ gaat hij door met de berekening voor j + 1; als f(j) = n stopt de machine. We bewijzen niet, dat er een Turing-machine T' bestaat, die dit doet. Het is eenvoudig in te zien, dat V de definitieverzameling is van de bij T' behorende partieel Turing-berekenbare functie.

Stel nu, dat de verzameling V definitieverzameling is van de partieel recursieve functie f(x). Laat T een Turing-machine zijn, die f(x) berekent. Als V leeg is, is V recursief opsombaar. Als V niet-leeg is, nemen we een $a \in V$. In het laatste geval zoeken we een Turing-machine T', die het volgende

doet. Als de invoer $\vdash\bigcirc|^n$ is, berekent de machine eerst hd(n) en $t\ell(n)$ en probeert vervolgens f(hd(n)) met de machine T uit te rekenen, maar doet daarbij ten hoogste $t\ell(n)$ berekeningsstappen. Als dit leidt tot een berekening van f(hd(n)), dan produceert de machine het getal hd(n) en stopt; als dit niet leidt tot een berekening van f(hd(n)), dan produceert de machine het getal a en stopt. We bewijzen niet, dat er een Turing-machine T' bestaat, die dit doet. Het is duidelijk, dat de bij T' behorende functie totaal is. Verder liggen de berekende functiewaarden in V. Immers zulk een functiewaarde is hetzij hd(n), maar dan is f(hd(n)) gedefinieerd, dus $hd(n) \in V$, hetzij a, maar ook $a \in V$. Als omgekeerd $k \in V$, dan is f(k) gedefinieerd. Laat ℓ het aantal berekeningsstappen zijn van de berekening van f(k) op de machine T. Geven we nu de invoer $\vdash\bigcirc|^n$ met $n = (2k + 1)2^{\ell} - 1$ aan de machine T', dan gaat deze ten hoogste $t\ell(n) = \ell$ stappen doen van de berekening van f(hd(n)) = f(k) op de machine T. Daarmee is f(hd(n)) dan inderdaad berekend, zodat T' het getal hd(n) = k produceert en stopt. Dus zit k in het beeld van de bij T' behorende functie. Dus V is recursief opsombaar.

We noemen de karakterisering in de definitie van recursief opsombaar karakterisering 1 en de in stelling 10.1 vermelde karakterisering 2. Zonder bewijs vermelden we nog twee karakteriseringen:

3. beeld van een partieel recursieve functie van één variabele,
4. leeg of beeld van een primitief-recursieve functie van één variabele.

Definitie beslisbaar, opsombaar.
Een verzameling symboolrijen heet beslisbaar (resp. opsombaar), als de bij een Gödel-nummering met de verzameling corresponderende verzameling natuurlijke getallen recursief (resp. recursief opsombaar) is.

Op het hierboven gegeven tweede deel van het bewijs van stelling 10.1 kan fundamentele kritiek worden uitgeoefend. Eerst wordt het geval, dat V leeg is, afgehandeld. Daarna komt het geval, dat V niet-leeg is; men neemt dan een $a \in V$. De vraag is, hoe men aan zulk een a komt. De machine T' kan alleen gemaakt worden als a bekend is. Gegeven is een verzameling, die definitieverzameling is van een partieel recursieve functie; men mag dan aannemen, dat een Turing-machine, die die functie berekent, gegeven is. Daarmee

weet men nog niet of de verzameling leeg is of niet en heeft men geen constructief middel om een element van die verzameling te vinden. Proberen hoeft niet te helpen.

De in dit hoofdstuk gegeven discussie beoogt een analyse van constructiviteit met behulp van Turing-machines, recursieve functies en dergelijke. Over deze begrippen wordt in de metataal gesproken en worden stellingen afgeleid. Men is geneigd daaraan ook constructiviteitseisen te stellen, hetgeen moeilijk is, omdat de metataal informeel is. Doet men het niet, dan dreigt het gevaar van een onconstructieve theorie van het constructieve. Als men met een niet-constructief bewijs aantoont, dat een verzameling recursief is, dan is daarmee geen uitvoerbaar beslissingsproces gevonden.

In het bijzonder zijn uitspraken over existentie gevaarlijk. Zo is op grond van de definitie een niet-lege verzameling recursief opsombaar, als er een (O,|)-Turing-machine bestaat, waarbij een totale functie behoort, die de verzameling als beeld heeft. De vraag rijst, welke constructieve eisen aan dit bestaan gesteld worden; bovenstaand bewijs van stelling 10.1 illustreert dit duidelijk. We geven nog een ander voorbeeld.

In de theorie van de recursieve verzamelingen bewijst men, dat iedere eindige verzameling recursief is. De vraag is echter wat een eindige verzameling is. Met gebruikelijke wiskundige methoden kan men eindige verzamelingen definiëren, waarvan men niet expliciet kan vaststellen of bepaalde natuurlijke getallen er element van zijn of niet. Zulk een verzameling zou dan toch recursief zijn. Om dit te verkrijgen kan men een onopgelost probleem uit de getaltheorie gebruiken, zoals de stelling van Fermat. Men heeft dan een eigenschap, die natuurlijke getallen kunnen hebben en waarbij er natuurlijke getallen zijn waarvoor men geen procedure kent om vast te stellen of de eigenschap voor dat getal geldt of niet. Neemt men nu een getal m, dat zo groot is dat er zulke onbeslisbare getallen onder liggen en neemt men als verzameling V de verzameling der natuurlijke getallen $< m$, die de eigenschap hebben, dan is V zulk een eindige verzameling. We gaan op de hier aangeroerde aangelegenheid, die ons op het terrein van de grondslagen der wiskunde heeft gebracht, niet verder in.

Stelling 10.2. Iedere recursieve verzameling is recursief opsombaar.

Bewijs. Als V recursief is, dan is χ_V een recursieve functie. Laat T een Turing-machine zijn, die χ_V berekent. De machine, die eerst T uitvoert en als dit het getal 0 als functiewaarde oplevert in een niet-stoppende machine overgaat, heeft V als definitieverzameling van de bijbehorende functie. Karakterisering 2 leert dat V recursief opsombaar is.

Stelling 10.3. Als een verzameling en haar complement recursief opsombaar zijn, dan is de verzameling recursief.

Bewijs. Laat S en haar complement $\bar{S}$ recursief opsombaar zijn. Op grond van karakterisering 2 bestaat er dan een Turing-machine T_0, die S als definitieverzameling van de bijbehorende functie heeft en een Turing-machine T_1, waarvoor hetzelfde voor $\bar{S}$ geldt. De Turing-machine T' doet het volgende met een invoer $\downarrow 0|^n$. Hij copieert het getal n en gaat nu beurtelings een berekeningsstap van de berekening van T_0 en T_1 doen, uitgaande van het ene en het andere getal n op de band, waarbij de resultaten van de wederzijdse berekeningen bewaard worden. Als de berekening stopt bij een handeling van T_0, dan produceert T' het getal 1 en stopt, en als de berekening stopt bij een handeling van T_1, dan produceert hij het getal 0 en stopt. De bij deze machine behorende functie is χ_S, dus S is recursief.

Stelling 10.4. Als een verzameling recursief is, dan is haar complement recursief.

Het bewijs van stelling 10.4 is eenvoudig.

We willen een voorbeeld geven van een verzameling, die recursief opsombaar maar niet recursief is. Dan is op grond van stelling 10.3 het complement van die verzameling niet recursief opsombaar.

Bij iedere recursief opsombare verzameling S bestaat er een $k \in N$, zodat

$$S = \{x \mid \exists_y T(k,x,y)\} .$$

Immers als S recursief opsombaar is, dan is, op grond van karakterisering 2, S definitieverzameling van een partieel recursieve functie f(x); in § 9 hebben we gezien, dat er een $k \in N$ bestaat, zodat

$$f(x) = \Psi(k,x) = U(\mu_y\ T(k,x,y)) \quad .$$

Hieruit volgt direct dat de definitieverzameling van f(x) de verzameling $\{x \mid \exists_y\ T(k,x,y)\}$ is.

We nemen nu de verzameling $V := \{x \mid \exists_y\ T(x,x,y)\}$; deze is recursief opsombaar, want het is de definitieverzameling van de partieel recursieve functie $\mu_y\ T(x,x,y)$.

Veronderstel nu, dat V recursief is, dan is ook $\bar{V}$ recursief (stelling 10.4) en dus recursief opsombaar (stelling 10.2). Dus is er een $k \in N$, zodat

$$\bar{V} = \{x \mid \exists_y\ T(k,x,y)\} \ .$$

Veronderstel dat $k \in \bar{V}$, dan $\exists_y\ T(k,k,y)$, dus $k \in V$, hetgeen een tegenspraak oplevert. Dus $k \notin \bar{V}$, maar dan $k \in V$, $\exists_y\ T(k,k,y)$, $k \in \bar{V}$. Dit levert een tegenspraak, dus V is niet recursief.

Er bestaan ook verzamelingen S, zodat S en $\bar{S}$ beide niet recursief opsombaar zijn. We gaan dat niet na. Het blijkt dat de verzameling van de Gödel-nummers van de genummerde (O,|)-Turing-machines, waarvan de bijbehorende functie een totale functie is, een voorbeeld is van zulk een verzameling.

Interessanter dan de bewering, dat bovenstaande verzameling V niet recursief is, zijn soortgelijke beweringen over relevante logische of mathematische verzamelingen, zoals bijvoorbeeld de verzameling van de geldige formules in de predicatenlogica van de eerste orde. Bewijzen van dergelijke beweringen worden vaak gevoerd door terugbrenging op de verzameling V.

Ook het stopprobleem kan nu worden vertaald. Een Turing-machine T_0, die de partieel recursieve functie $\mu_y\ T(x,x,y)$ berekent, heeft een onoplosbaar stopprobleem. Zou dit stopprobleem namelijk wel oplosbaar zijn, dan zou de verzameling $\{x \mid \exists_y\ T(x,x,y)\}$ een beslisbare en dus recursieve verzameling zijn, terwijl we hierboven hebben gezien, dat dit niet het geval is. De hier gegeven oplossing levert een sterkere versie van de onoplosbaarheid van het stopprobleem dan de vroeger behandelde. Toen lieten we de onmogelijkheid zien van een beslissingsprocedure, die voor een willekeurige Turing-machine plus bijbehorende invoer uitmaakt of de machine stopt of niet. Nu beschikken we over een vaste en expliciet construeerbare Turing-machine T_0, waarvoor geen beslissingsprocedure bestaat, die voor een willekeurige invoer uitmaakt of de machine stopt of niet.

APPENDIX PREDICATENLOGICA VAN DE EERSTE ORDE

In de predicatenlogica wordt de structuur van mededelende volzinnen nader onderzocht; de uit de redekundige ontleding bekende begrippen onderwerp (subject) en gezegde (predicaat) zijn daarbij uitgangspunt.

Zo geeft de zin "Piet is een jongen" het predicaat J(x): "een jongen zijn" met subject Piet: J(Piet).

Evenzo "5 is een priemgetal"; predicaat P(x): "een priemgetal zijn" met subject 5: P(5).

Men kan P(x) lezen als "x is een priemgetal"; hierin is x een objectvariabele. Predicaten kunnen ook van meer variabelen afhangen.

"Jan is de vader van Piet": V(x,y) is "vader zijn van": V(Jan,Piet).

"Jan en Marie zijn ouders van Piet": O(Jan,Marie,Piet).

Een nieuw aspect van de predicatenlogica is de quantificatie:

$\exists_x$: er bestaat een x zodat ...

$\forall_x$: voor alle x geldt ...

Voorbeelden: $\exists_x$ P(x) of $\forall_x$ (V(x,Piet) $\rightarrow$ J(x)).

De zin $\exists_x$ P(x) hangt niet van de variabele x af. Door de quantificatie is x een gebonden variabele geworden, net als de x in $\int_0^{2\pi} \sin x \, dx$.

Als we het bovenstaande gaan formaliseren krijgen we objectvariabelen en predicaatvariabelen; bij deze laatste behoort een orde, dat is het aantal objectvariabelen waarvan zij afhangt. Wij kiezen bij de formalisering een (ongebruikelijke!) opzet, waarbij de orde van een predicaatvariabele niet expliciet gemaakt wordt; het aantal objecten, dat erachter staat, geeft aan wat de orde is en hetzelfde symbool kan voor predicaatvariabelen van verschillende orde worden gebruikt.

Bij een concrete interpretatie worden de objectvariabelen vervangen door concrete objecten en de predicaatvariabelen door concrete predicaten. Over concrete objecten kan men echter niet quantificeren. We lossen deze moeilijkheid op door nog een type variabelen in te voeren, waarover gequantificeerd mag worden en die we loze variabelen (engels: dummy variables)

noemen. In sommige boeken gebruikt men "variabelen" voor wat wij "loze variabelen" noemen en "constanten" voor wat wij "objectvariabelen" noemen.

In de formele opzet kan $\forall$ gemist worden, omdat $\forall_x$ hetzelfde betekent als $\neg\exists_x\neg$. Propositievariabelen komen in het systeem voor als predicaatvariabelen van orde nul.

Als F een formule is, dan is $\exists_x$ F een formule. De vraag rijst, of we daarbij moeten eisen, dat x vrij in F voorkomt. Dit is lastig, want om het begrip formule te definiëren moeten we dan weten, wat het betekent dat x vrij in F voorkomt, maar om dat te definiëren moeten we weten, wat een formule is. Wij kiezen een andere methode, waarbij $\exists_x$ F altijd mag worden opgeschreven ook als x gebonden in F voorkomt. Als x uitsluitend gebonden in F voorkomt, heeft het voorplaatsen van $\exists_x$ voor de betekenis van de formule geen effect. We kunnen dan formules krijgen als $\exists_x$ (($\exists_x$ F) $\wedge$ G). De voorste $\exists_x$ werkt dan alleen op G. De hier gekozen opzet correspondeert met het toelaten in de wiskunde van formules van de gedaante

$$\tan x + \int_0^{\frac{1}{2}\pi} (\sin x + \int_0^{\frac{1}{2}\pi} \cos x \, dx)dx \ .$$

In de analyse wordt het gebruik van zulke formules gewoonlijk verboden.

We definiëren de predicatenlogica van de eerste orde als formeel systeem. De symbolen zijn: x, a, p, |, $\neg$, $\rightarrow$, $\exists$. Het symbool | wordt gebruikt als nummeringssymbool, waarmee genummerde symboolrijen kunnen worden gevormd (zie § 6 van hoofdstuk I).

Definitie loze variabele.
Een loze variabele is een genummerde x.

Definitie objectvariabele.
Een objectvariabele is een genummerde a.

Definitie object.
Een object is een loze variabele of een objectvariabele.

Definitie predicaatvariabele.
Een predicaatvariabele is een genummerde p.

Definitie predicaat.
Een predicaatvariabele is een predicaat.
Als Q een predicaat is en V een object, dan is QV een predicaat.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een predicaat.

Definitie formule.
Een predicaat is een formule.
Als A een formule is, dan is ¬A een formule.
Als A en B formules zijn, dan is → AB een formule.
Als A een formule is en X een loze variabele, dan is ∃XA een formule.
Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een formule.

Op grond van stelling 6.8 van hoofdstuk I is de verzameling E bestaande uit de objecten, de predicaatvariabelen, ¬, → en ∃ een verzameling uitgebreide symbolen. Uiteraard zijn predicaten en formules dan elementen van E^+, dus uitgebreide symboolrijen. We willen recurrente definities geven zowel langs de opbouw van een predicaat als langs de opbouw van een formule. Voor predicaten is dit eenvoudig te rechtvaardigen, omdat bij de opbouw van een predicaat telkens één uitgebreid symbool wordt toegevoegd. Voor de rechtvaardiging bij formules zullen we een designatorstelling afleiden.

Definitie formuledesignator.
Ieder predicaat is een formuledesignator.
Als A een formule is, dan is ¬, A een formuledesignator.
Als A en B formules zijn, dan is →,A,B een formuledesignator.
Als A een formule is en X een loze variabele, dan is ∃,X,A een formuledesignator.

Definitie bijbehorende formule van een formuledesignator.
Als Q een predicaat is, dan is Q de bijbehorende formule van de formuledesignator Q.
Van de formuledesignator ¬,A is ¬A de bijbehorende formule.
Van de formuledesignator →,A,B is →AB de bijbehorende formule.
Van de formuledesignator ∃,X,A is ∃XA de bijbehorende formule.

Stelling A1. Als Q een predicaat is, dan bestaat er een $Q' \in E^*$ en een predicaatvariabele P, zodat $Q = PQ'$; als $C \in E^+$, $D \in E^+$ en $Q = CD$, dan bestaat er een object V en een $D' \in E^*$, zodat $D = VD'$.

Bewijs. We passen recurrentie toe langs de opbouw van het predicaat Q. Als Q een predicaatvariabele is, kiezen we $P = Q$ en $Q' = \Lambda$. Verder is $Q = CD$ dan onmogelijk, omdat $\ell'(Q) = 1$, $\ell'(C) \geq 1$, $\ell'(D) \geq 1$.
Stel dat de stelling geldt voor het predicaat Q en dat V een object is, dan bestaat er een $Q' \in E^*$ en een predicaatvariabele P, zodat $Q = PQ'$, $QV = PQ'V$. Stel $QV = CD$. Op grond van stelling 6.5 bestaat er een $R \in E^*$, zodat:
hetzij $Q = CR$, $D = RV$; als $R = \Lambda$, dan $D = V\Lambda$; als $R \neq \Lambda$, dan bestaat er een object V' en een $R' \in E^*$, zodat $R = V'R'$, dus $D = V'R'V$;
hetzij $C = QR$, $V = RD$, $R \neq \Lambda$; dit is onmogelijk, omdat $\ell'(V) = 1$, $\ell'(R) \geq 1$, $\ell'(D) \geq 1$.

Stelling A2. Als $\alpha \in E$, $T \in E^*$ en αT een formule, dan geldt één en slechts één van de volgende vier beweringen.

1. α is een predicaatvariabele, αT is een predicaat.
2. $\alpha = \neg$, T is een formule.
3. $\alpha = \rightarrow$, er bestaan formules A en B, zodat $T = AB$.
4. $\alpha = \exists$, er bestaat een loze variabele X en een formule A, zodat $T = XA$.

Bewijs. Dat ten hoogste één van de vier beweringen kan gelden, ziet men aan α. Dat ten minste één van de vier beweringen geldt bewijzen we door recurren ie langs de opbouw van de formule αT. Als αT een predicaat is, bestaat er op grond van stelling A1 een predicaatvariabele P en een $Q' \in E^*$, zodat $\alpha T = PQ'$. Omdat $\ell'(\alpha) = \ell'(P) = 1$, geldt dan $\alpha = P$, hetgeen geval 1. oplevert. Stel, dat A een formule is, waarvoor de stelling geldt en $\alpha T = \neg A$. Omdat $\ell'(\alpha) = \ell'(\neg) = 1$, geldt $\alpha = \neg$, $T = A$, hetgeen tot geval 2. leidt. Stel nu dat de stelling voor de formules A en B geldt en $\alpha T = \rightarrow AB$. Wederom geldt $\alpha = \rightarrow$, $T = AB$: geval 3. Stel tenslotte, dat A een formule is, waarvoor de stelling geldt, X een loze variabele en $\alpha T = \exists XA$. Dan $\alpha = \exists$, $T = XA$: geval 4.

Stelling A3. Als C een formule is, $D \in E^+$ en CD een formule, dan bestaat er een object V en een $D' \in E^*$, zodat $D = VD'$.

Bewijs. We passen volledige inductie naar $\ell'(C)$ toe. Op grond van stelling A2 zijn er vier mogelijkheden voor C.

1. C is een predicaat en er bestaat een predicaatvariabele P en een $Q' \in E^*$, zodat $C = PQ'$. De formule $PQ'D$ bevindt zich in geval 1. van stelling A2 en is dus een predicaat. Op grond van stelling A1 bestaat er een object V en een $D' \in E^*$, zodat $D = VD'$.
2. Er bestaat een formule T, zodat $C = \neg T$. De formule $\neg TD$ bevindt zich in geval 2., dus TD is een formule. Omdat $\ell'(T) < \ell'(C)$ levert de inductieveronderstelling, dat er een object V en een $D' \in E^*$ bestaan, zodat $D = VD'$.
3. Er bestaan formules A en B, zodat $C = \rightarrow AB$. De formule $\rightarrow ABD$ bevindt zich in geval 3., dus er bestaan formules A' en B', zodat $ABD = A'B'$. Op grond van stelling 6.5 bestaat er een $R \in E^*$, zodat:
   hetzij $A' = AR$, $BD = RB'$; als $R = \Lambda$, dan $B' = BD$ en verder $\ell'(B) < \ell'(C)$, zodat er op grond van de inductieveronderstelling een object V en een $D' \in E^*$ bestaat, zodat $D = VD'$; als $R \neq \Lambda$, dan levert de inductieveronderstelling, die toegepast mag worden omdat $\ell'(A) < \ell'(C)$, dat er een object V en een $R' \in E^*$ bestaan, zodat $R = VR'$, dus $BD = VR'B'$, hetgeen onmogelijk is op grond van stelling A2, toegepast op B (het voorste uitgebreide symbool van B kan niet V zijn);
   hetzij $A = A'R$, $B' = RBD$, $R \neq \Lambda$; omdat $\ell'(A') < \ell'(A) < \ell'(C)$, passen we de inductieveronderstelling toe, hetgeen oplevert, dat er een object V en een $R' \in E^*$ bestaan, zodat $R = VR'$, $B' = VR'BD$, hetgeen onmogelijk is op grond van stelling A2, toegepast op B'.
4. Er bestaat een loze variabele X en een formule A, zodat $C = \exists XA$. De formule $\exists XAD$ bevindt zich in geval 4., dus AD is een formule. Omdat $\ell'(A) < \ell'(C)$, levert de inductieveronderstelling, dat er een object V en een $D' \in E^*$ bestaan, zodat $D = VD'$.

Stelling A4. Bij iedere formule bestaat er één en slechts één formuledesignator, waarvan die formule de bijbehorende formule is.

Bewijs. Als een formule bij twee designatoren hoort, dan zijn deze of beide predicaten of beide van het type ¬, A of beide van het type →, A, B of beide van het type ∃, X, A. Dit volgt direct uit stelling A1 en stelling A2.
We bewijzen de stelling nu door recurrentie langs de opbouw van de formule.
Als A een predicaat is, dan behoort A bij de designator A en dat is de enige designator, waar A bij kan horen.
Als A een formule is, waarvoor de stelling geldt, dan behoort ¬A bij de designator ¬,A. Stel dat ¬A ook bij de designator ¬,A' behoort, dan ¬A = ¬A', A = A' en de beide designatoren zijn gelijk.
Als A en B formules zijn, waarvoor de stelling geldt, dan behoort → AB bij de designator →, A, B. Stel dat → AB ook behoort bij de designator →,A',B', dan → AB = → A'B', AB = A'B'. Op grond van stelling 6.5 bestaat er een $R \in E^*$, zodat:
hetzij A = A'R, B' = RB; als $R \neq \Lambda$, dan bestaat er op grond van stelling A3 een object V en een $R' \in E^*$, zodat R = VR', B' = VR'B, hetgeen onmogelijk is op grond van stelling A2, toegepast op B'; dus $R = \Lambda$, A = A', B = B' en beide designatoren zijn gelijk;
hetzij A' = AR, B = RB'; analoog als het vorige geval.
Als A een formule is, waarvoor de stelling geldt en X een loze variabele, dan behoort ∃XA bij de designator ∃,X,A. Stel dat ∃XA ook behoort bij de designator ∃,X',A', dan ∃XA = ∃X'A', XA = X'A'. Omdat $\ell'(X) = \ell'(X') = 1$, volgt hieruit X = X' en A = A', dus beide designatoren zijn gelijk.

Definitie vrij voorkomen in predicaat.
Laat X een loze variabele zijn.
Als P een predicaatvariabele is, dan komt X niet vrij voor in P.
Als Q een predicaat is en V een object, dan komt X vrij voor in QV dan en slechts dan als X vrij voorkomt in Q of X = V.

Definitie vrij voorkomen in formule.
Laat X een loze variabele zijn.
Als P een predicaat is, dan komt X vrij voor in P als formule dan en slechts dan als X vrij voorkomt in P als predicaat.
Als A een formule is, dan komt X vrij voor in ¬A dan en slechts dan als X vrij voorkomt in A.
Als A en B formules zijn, dan komt X vrij voor in →AB dan en slechts dan als X vrij voorkomt in A of X vrij voorkomt in B.
Als A een formule is en Y een loze variabele, dan komt X vrij voor in ∃YA dan en slechts dan als X vrij voorkomt in A en $X \neq Y$.

Definitie gesloten formule.
Een formule A heet een gesloten formule als voor alle loze variabelen X geldt, dat X niet vrij voorkomt in A.

We behandelen nu de semantiek van de predicatenlogica. Nu moeten de objectvariabelen en predicaatvariabelen worden geïnterpreteerd met behulp van objecten en predicaten. De objecten zijn elementen van een verzameling M en de predicaten zijn op M gedefinieerd. Bij de semantiek van de predicatenlogica zien we af van ons constructieve standpunt en maken een vrij gebruik van het verzamelingsbegrip. Omdat de objectvariabelen een genummerde rij vormen, moet er een bijbehorende rij objecten gekozen worden. Een predicaat van orde n interpreteren we als een verzameling n-tupels van elementen van M. Aan een predicaat hebben we echter geen vaste orde toegekend, zodat alle eindige rijen van elementen van M in aanmerking komen.

Als M een verzameling is, noemen we M^* de verzameling van alle eindige rijen van elementen van M, inclusief de lege rij Λ. We schrijven de rijen met komma's tussen de elementen van de rij. Verder schrijven we P(V) voor de verzameling van alle deelverzamelingen van de verzameling V.

Definitie structuur.
Een structuur is een tripel $< M,O,R >$, waarin M een verzameling is, O een afbeelding $N \to M$ en R een afbeelding $N \to P(M^*)$.

De intuïtieve interpretatie hiervan is:
O(n) is het element van M, dat toegevoegd is aan de objectvariabele a_n,
R(n) is het predicaat van M, dat toegevoegd is aan het predicaat p_n.

Let wel, dat het bestaan van een afbeelding $N \to M$ impliceert, dat M niet leeg is.

Bij de interpretatie van formules spelen echter ook de loze variabelen, die vrij in de formules voorkomen, een rol. Men zou dit kunnen uitschakelen door zich tot gesloten formules te beperken. Dit is echter moeilijk uitvoerbaar, omdat bij de recurrente opbouw van een gesloten formule ook formules optreden, die niet gesloten zijn. We voegen daarom als hulpmiddel ook een toevoeging van objecten van M aan loze variabelen toe, die we, in navolging van Van Dalen, een bedeling noemen.

Definitie bedeling.
Als M een verzameling is, dan is een bedeling in M een afbeelding $N \to M$.

De intuïtieve interpretatie van een bedeling α is:
$\alpha(n)$ is het element van M, dat toegevoegd is aan de loze variabele x_n.

Als α een symbool is, dan behoort bij een genummerde α ook een nummer $j(\alpha)$. We passen daarbij recurrentie langs de opbouw van de genummerde α toe, hetgeen geoorloofd is, omdat bij die opbouw telkens slechts één symbool toegevoegd wordt.

Definitie nummer j van genummerde α.
Als A een genummerde α is, dan

$$j(\alpha) := 0 ,$$

$$j(A|) := j(A) + 1 .$$

Stelling A5. Als A een genummerde α is, dan geldt:
hetzij $A = \alpha$ en $j(A) = 0$;
hetzij er bestaat een symboolrij A', die een genummerde α is, $A = A'|$ en $j(A) > 0$.

Bewijs door recurrentie langs de opbouw van de genummerde α.

Stelling A6. Als A en B genummerde α zijn en $j(A) = j(B)$, dan $A = B$.

Bewijs met volledige inductie naar $j(A)$; hierbij kan stelling A5 gebruikt worden.

Als S een structuur is, α een bedeling en Q een predicaat, dan is er een bij Q behorend element van M^*, dat we $e_{S,\alpha}(Q)$ noemen.

Definitie $e_{S,\alpha}(Q)$.

Als $S = < M,O,R >$ een structuur is, α een bedeling in M, P een predicaatvariabele, Q een predicaat en V een object, dan

$$e_{S,\alpha}(P) := \Lambda ,$$

$$e_{S,\alpha}(QV) := \begin{cases} e_{S,\alpha}(Q),\ O(j(V)) & \text{als V een objectvariabele is,} \\ e_{S,\alpha}(Q),\ \alpha(j(V)) & \text{als V een loze variabele is.} \end{cases}$$

Definitie nummer i van predicaat.

Als P een predicaatvariabele is, Q een predicaat en V een object, dan

$$i(P) := j(P) ,$$

$$i(QV) := i(Q) .$$

We voeren nu in de waarheidswaarde van een formule A, behorend bij een structuur S en een bedeling α, genaamd $v_{S,\alpha}(A)$.

Definitie $v_{S,\alpha}(A)$.

Laat $S = < M,O,R >$ een structuur zijn en α een bedeling in M. Als Q een predicaat is, dan

$$v_{S,\alpha}(Q) := \begin{cases} 1, \text{ als } e_{S,\alpha}(Q) \in R(i(Q)) , \\ 0, \text{ als } e_{S,\alpha}(Q) \notin R(i(Q)) . \end{cases}$$

Als A en B formules zijn, dan worden $v_{S,\alpha}(\neg A)$ en $v_{S,\alpha}(\rightarrow AB)$ bepaald uit $v_{S,\alpha}(A)$ en $v_{S,\alpha}(B)$ met behulp van de waarheidstafels voor $\neg$ en $\rightarrow$.
Als A een formule is en X een loze variabele, dan $v_{S,\alpha}(\exists XA) = 1$, als er een bedeling β in M bestaat, zodat als $n \in N$ en $n \neq j(X)$, dan $\beta(n) = \alpha(n)$ en $v_{S,\beta}(A) = 1$; anders $v_{S,\alpha}(\exists XA) = 0$.

Het blijkt nu, dat de waarheidswaarde van een formule A niet afhangt van de elementen, die in de bedeling zijn toegekend aan variabelen, die niet vrij voorkomen in A. Dit wordt uitgedrukt in de volgende stelling.

Stelling A7. Als $S = < M,O,R >$ een structuur is, α en β bedelingen in M zijn en A een formule is, zodat voor iedere loze variabele X, die vrij voorkomt in A, geldt $\alpha(j(X)) = \beta(j(X))$, dan $v_{S,\alpha}(A) = v_{S,\beta}(A)$.

Bewijs. We passen recurrentie toe langs de opbouw van de formule A. Voor een predicaat Q is het voldoende om $e_{S,\alpha}(Q) = e_{S,\beta}(Q)$ te bewijzen, hetgeen we met recurrentie langs de opbouw van Q doen.
$e_{S,\alpha}(P) = \Lambda = e_{S,\beta}(P)$.
Stel de bewering is goed voor het predicaat Q, V is een object en α en β voldoen aan de voorwaarden voor het predicaat QV. Omdat iedere loze variabele, die vrij voorkomt in Q, ook vrij voorkomt in QV, voldoen α en β ook aan de voorwaarden voor het predicaat Q. Als V een objectvariabele is, dan

$$e_{S,\alpha}(QV) = e_{S,\alpha}(Q) \ , \ O(j(V)) = e_{S,\beta}(Q) \ , \ O(j(V)) = e_{S,\beta}(QV) \ .$$

Als V een loze variabele is, dan komt V vrij voor in QV, dus $\alpha(j(V)) = \beta(j(V))$, $e_{S,\alpha}(QV) = e_{S,\alpha}(Q)$, $\alpha(j(V)) = e_{S,\beta}(Q)$, $\beta(j(V)) = e_{S,\beta}(QV)$.
De overgang van formules A en B op $\neg A$ en $\rightarrow AB$ is eenvoudig uit te voeren, omdat iedere variabele, die vrij voorkomt in A, ook vrij voorkomt in $\neg A$ en $\rightarrow AB$ en evenzo voor B en $\rightarrow AB$.
Stel nu dat de stelling juist is voor de formule A en dat α en β voldoen aan de voorwaarden voor de formule $\exists XA$. Stel dat $v_{S,\alpha}(\exists XA) = 1$. Dan bestaat er een bedeling γ in M, zodat als $n \in N$ en $n \neq j(X)$, dan $\gamma(n) = \alpha(n)$ en $v_{S,\gamma}(A) = 1$. We definiëren een bedeling δ in M als volgt: $\delta(n) := \beta(n)$ als $n \neq j(X)$, $\delta(j(X)) := \gamma(j(X))$. Stel Y is een loze variabele, die vrij voorkomt in A. Als $Y \neq X$, dan $j(Y) \neq j(X)$ op grond van stelling A6 en Y komt vrij voor in $\exists XA$, dus $\gamma(j(Y)) = \alpha(j(Y)) = \beta(j(Y)) = \delta(j(Y))$. Als $Y = X$, dan ook $\gamma(j(Y)) = \delta(j(Y))$. Op grond van de recurrentieveronderstelling geldt $v_{S,\gamma}(A) = v_{S,\delta}(A)$, dus $v_{S,\delta}(A) = 1$, dus $v_{S,\beta}(\exists XA) = 1$. Op analoge wijze bewijst men: als $v_{S,\beta}(\exists XA) = 1$, dan $v_{S,\alpha}(\exists XA) = 1$. Dus $v_{S,\alpha}(\exists XA) = v_{S,\beta}(\exists XA)$.

Stelling A8. Als $S = < M,O,R >$ een structuur is, α en β bedelingen zijn in M en A een gesloten formule is, dan is $v_{S,\alpha}(A) = v_{S,\beta}(A)$.

Dit is een direct gevolg van stelling A7.

Op grond van stelling A8 hangt voor een gesloten formule de waarheidswaarde helemaal niet af van de gekozen bedeling. We kunnen in dat geval $v_S(A)$ schrijven in plaats van $v_{S,\alpha}(A)$. Voor het gemak beperken we ons voortaan tot gesloten formules.

In de predicatenlogica is het niet gebruikelijk formules die altijd waar zijn tautologieën te noemen. We noemen ze geldige formules.

Definitie geldige formule.
Een gesloten formule A heet een geldige formule als voor alle structuren S geldt $v_S(A) = 1$.

Men kan een deductief systeem voor de predicatenlogica van de eerste orde definiëren, zodat de afleidbare gesloten formules samenvallen met de geldige formules.

Stelling A9. Er bestaat een beslisbaar deductief systeem voor de predicatenlogica van de eerste orde, zodat een gesloten formule afleidbaar is dan en slechts dan als ze een geldige formule is.

Het bewijs van deze zwakke volledigheidsstelling is zeer gecompliceerd. Gödel, die deze stelling in 1930 heeft bewezen, toonde aan dat een gebruikelijk klassiek deductief systeem voldoet.

Uit stelling A9 volgt:

Stelling A10. De verzameling van de geldige formules is opsombaar.

Church heeft in 1936 bewezen, dat de verzameling van de geldige formules niet beslisbaar is.

Men kan nu ook het begrip theorie invoeren, dat is een beslisbare verzameling gesloten formules.

Definitie model.
Als T een theorie is en S een structuur, dan heet S een model van T, als voor alle $A \in T$ geldt $v_S(A) = 1$.

Definitie logische consequentie.
Als T een theorie is en A een gesloten formule, dan heet A een logische consequentie van T als voor ieder model S van T geldt $v_S(A) = 1$.

Stelling A11. Er bestaat een beslisbaar deductief systeem voor de predicatenlogica van de eerste orde, zodat voor iedere theorie T en iedere gesloten formule A geldt, dat A een logische consequentie is van T dan en slechts dan als A afleidbaar is uit T.

Ook deze sterke volledigheidsstelling is door Gödel in 1930 bewezen voor hetzelfde deductieve systeem als bij stelling A9. Uit de sterke volledigheidsstelling volgt de compactheidsstelling, die zegt, dat als de gesloten formule A logische consequentie is van een theorie T, er een eindige $T' \subset T$ bestaat, zodat A logische consequentie is van T'. We nemen aan, dat een vast deductief systeem gekozen is, waarvoor de sterke volledigheidsstelling geldt.

Definitie consistente theorie.
Een theorie T heet consistent, als $\neg\rightarrow pp$ niet afleidbaar is uit T.

Men kan dan bewijzen, dat een theorie dan en slechts dan consistent is, als er een gesloten formule bestaat, die niet afleidbaar is uit T. Het is eenvoudig in te zien, dat een theorie, die een model heeft, consistent is. Het omgekeerde is echter ook waar:

Stelling A12. Als een theorie T consistent is, dan bestaat er een structuur, die model van T is.

Deze stelling volgt gemakkelijk uit de sterke volledigheidsstelling.

Werkcollege toegepaste Logica, najaar 1978.

1-ste serie: boole-tralie en boole-algebra.

Definities

Een tralie is een partieel geordende verzameling waarin ieder tweetal elementen een infimum en een supremum heeft.
Een boole-tralie is een distributief tralie met een grootste en een kleinste element en met complementen.
Notatie: $\langle B, \leq \rangle$ of kortweg B.
Een boole-algebra is een algebraische structuur $\langle B, \vee, \wedge, ^*, 0, 1 \rangle$ die voldoet aan:
voor alle x, y en z uit B geldt

1	$x \vee y = y \vee x$	$x \wedge y = y \wedge x$
2	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
3	$(x \vee y) \wedge y = y$	$(x \wedge y) \vee y = y$
4	$x \vee x^* = 1$	$x \wedge x^* = 0$
5	$(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$	$(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z)$.

Zie ook: Ackermans en Van Lint, Algebra en Analyse § 2.8 en § 3.12.

Opgaven

1. Een totaal geordende verzameling B is dan en slechts dan een boole-tralie wanneer B ten hoogste twee elementen heeft. Bewijs.
2. We definiëren de relatie $\sim$ op de machtsverzameling van de natuurlijke getallen (notatie $P(\mathbb{N})$) door

   $$A \sim B := (A \setminus B) \cup (B \setminus A) \text{ is eindig.}$$

   Laat $\tilde{A}$ de equivalentie-klasse zijn waartoe A behoort en $X := \{\tilde{A} \mid A \subset \mathbb{N}\}$.
   We definiëren de relatie $\leq$ op X door

   $$\tilde{A} \leq \tilde{B} := A \setminus B \text{ is eindig.}$$

   Toon aan dat

a) $\sim$ een equivalentie-relatie is.

b) $\leq$ correct gedefinieerd is; d.w.z. de definitie is onafhankelijk van de keuze van de representant uit de equivalentie-klasse.

c) $\langle X,\leq\rangle$ een boole-tralie is.

3. Een boole-tralie induceert een boole-algebra (en omgekeerd).
   Laat het boole-tralie $\langle B,\leq\rangle$ gegeven zijn. Geef definities voor $\vee,\wedge,^*,0$ en 1 zó dat $\langle B,\vee,\wedge,^*,0,1\rangle$ een boole-algebra wordt.

4. Als $\langle B,\vee,\wedge,^*,0,1\rangle$ een boole-algebra is, dan is $\langle B,\wedge,\vee,^*,1,0\rangle$ ook een boole-algebra. Bewijs.

Aanvullingen en verbeteringen behorende bij het dictaat nr. 2.213
Toegepaste Logica I van prof.dr. W. Peremans, uitgave najaarssemester 1974

1. Van de op blz. 24 als stelling 5.1 vermelde designatorstelling kan een ander bewijs worden gegeven, dat geen gebruik maakt van de hulpafbeelding w. De nu volgende tekst komt in de plaats van het tekstgedeelte beginnend op blz. 24 midden en eindigend na de eerste alinea van blz. 26.

Alvorens stelling 5.1 te bewijzen, bewijzen we eerst twee hulpstellingen.

Stelling 5.2. Als α een symbool is, X een eventueel-lege symboolrij en αX een formule, dan geldt één en slechts één van de volgende drie beweringen.

1. α is een propositievariabele, $X = \Lambda$.
2. $\alpha = \neg$, X is een formule.
3. $\alpha = \rightarrow$, er bestaan formules A en B, zodat $X = AB$.

Bewijs. Dat ten hoogste één van de drie beweringen kan gelden, ziet men aan α. Dat ten minste één van de drie beweringen geldt, volgt door recurrentie langs de opbouw van de formule αX met behulp van stelling 4.6.

Stelling 5.3. Als A een formule is, X een eventueel-lege symboolrij en AX een formule, dan $X = \Lambda$.

Bewijs. We passen volledige inductie naar $\ell(A)$ toe. Op grond van stelling 5.2 zijn er drie mogelijkheden voor A.

1. Er is een propositievariabele a, zodat $A = a$, dus $AX = aX$. De formule aX bevindt zich in geval 1. van stelling 5.2, dus $X = \Lambda$.
2. Er is een formule B, zodat $A = \neg B$, dus $AX = \neg BX$. De formule $\neg BX$ bevindt zich in geval 2. van stelling 5.2, dus BX is een formule. Omdat $\ell(B) < \ell(A)$, mag op BX de inductieveronderstelling worden toegepast, op grond waarvan $X = \Lambda$.
3. Er zijn formules B en C, zodat $A = \rightarrow BC$, dus $AX = \rightarrow BCX$. De formule $\rightarrow BCX$ bevindt zich in geval 3. van stelling 5.2, dus er zijn formules D en E, zodat $\rightarrow BCX = \rightarrow DE$, $BCX = DE$. Op grond van stelling 4.8 bestaat er een eventueel-lege symboolrij R, zodat:
   hetzij $D = BR$, $CX = RE$; omdat $\ell(B) < \ell(A)$, mag op BR de inductieveronderstelling worden toegepast, die oplevert, dat $R = \Lambda$, dus $CX = E$; omdat $\ell(C) < \ell(A)$, mag de inductieveronderstelling ook op CX worden toegepast, hetgeen $X = \Lambda$ oplevert;

hetzij B = DR, E = RCX, R ≠ Λ; omdat ℓ(D) < ℓ(A), mag op DR de inductieveronderstelling worden toegepast, hetgeen R = Λ oplevert, wat een contradictie is.

Bewijs van stelling 5.1. We stellen eerst vast, dat van de drie soorten designatoren a en ¬, A en →, A, B een gegeven formule alleen bij twee designatoren van dezelfde soort kan behoren, omdat het voorste symbool van de bijbehorende formule in de drie gevallen resp. een propositievariabele, ¬, → is. Op grond van stelling 5.2 zijn er voor een formule A de volgende drie mogelijkheden.

1. Er is een propositievariabele a, zodat A = a. Dan behoort A bij de designator a en kan bij geen andere designator behoren.
2. Er is een formule B, zodat A = ¬B. Dan behoort A bij de designator ¬,B. Stel dat A ook bij de designator ¬,B' behoort, dan ¬B = ¬B', B = B' en de twee designatoren zijn gelijk.
3. Er zijn formules B en C, zodat A = → BC. Dan behoort A bij de designator →, B, C. Stel dat A ook behoort bij de designator →, B', C', dan → BC = = → B'C', BC = B'C'. Op grond van stelling 4.8 bestaat er een eventueellege symboolrij R, zodat:
   hetzij B = B'R, C' = RC; op grond van stelling 5.3 volgt uit B = B'R, dat R = Λ, B = B', C = C' en beide designatoren zijn gelijk;
   hetzij B' = BR, C = RC'; analoog als het vorige geval.

2. In de appendix wordt ook een designatorstelling bewezen met behulp van een hulpafbeelding w, die gemist kan worden. In plaats van het tekstgedeelte, dat begint op blz. 103, regel 11 van onder met "Daartoe voeren we ..." en dat eindigt met het bewijs van stelling A4 op blz. 106, beginnen we nu met de definities van formuledesignator en van bijbehorende formule van een formuledesignator zoals geformuleerd op blz. 105 en vervolgen dan met de volgende tekst.

Stelling A1. Als Q een predicaat is, dan bestaat er een Q' ∈ E^* en een predicaatvariabele P, zodat Q = PQ'; als C ∈ E^+, D ∈ E^+ en Q = CD, dan bestaat er een object V en een D' ∈ E^*, zodat D = VD'.

Bewijs. We passen recurrentie toe langs de opbouw van het predicaat Q. Als Q een predicaatvariabele is, kiezen we P = Q en Q' = Λ. Verder is Q = CD dan onmogelijk, omdat $\ell'(Q) = 1$, $\ell'(C) \geq 1$, $\ell'(D) \geq 1$.
Stel dat de stelling geldt voor het predicaat Q en dat V een object is, dan bestaat er een $Q' \in E^*$ en een predicaatvariabele P, zodat Q = PQ', QV = PQ'V. Stel QV = CD. Op grond van stelling 6.5 bestaat er een $R \in E^*$, zodat:
hetzij Q = CR, D = RV; als R = Λ, dan D = VΛ; als R ≠ Λ, dan bestaat er een object V' en een $R' \in E^*$, zodat R = V'R', dus D = V'R'V;
hetzij C = QR, V = RD, R ≠ Λ; dit is onmogelijk, omdat $\ell'(V) = 1$, $\ell'(R) \geq 1$, $\ell'(D) \geq 1$.

Stelling A2. Als $\alpha \in E$, $T \in E^*$ en αT een formule, dan geldt één en slechts één van de volgende vier beweringen.

1. α is een predicaatvariabele, αT is een predicaat.
2. α = ¬, T is een formule.
3. α = →, er bestaan formules A en B, zodat T = AB.
4. α = ∃, er bestaat een loze variabele X en een formule A, zodat T = XA.

Bewijs. Dat ten hoogste één van de vier beweringen kan gelden, ziet men aan α. Dat ten minste één van de vier beweringen geldt bewijzen we door recurrentie langs de opbouw van de formule αT. Als αT een predicaat is, bestaat er op grond van stelling A1 een predicaatvariabele P en een $Q' \in E^*$, zodat αT = PQ'. Omdat $\ell'(\alpha) = \ell'(P) = 1$, geldt dan α = P, hetgeen geval 1. oplevert. Stel, dat A een formule is, waarvoor de stelling geldt en αT = ¬A. Omdat $\ell'(\alpha) = \ell'(\neg) = 1$, geldt α = ¬, T = A, hetgeen tot geval 2. leidt. Stel nu dat de stelling voor de formules A en B geldt en αT = → AB. Wederom geldt α = →, T = AB: geval 3. Stel tenslotte, dat A een formule is, waarvoor de stelling geldt, X een loze variabele en αT = ∃XA. Dan α = ∃, T = XA: geval 4.

Stelling A3. Als C een formule is, $D \in E^+$ en CD een formule, dan bestaat er een object V en een $D' \in E^*$, zodat D = VD'.

Bewijs. We passen volledige inductie naar $\ell'(C)$ toe. Op grond van stelling A2 zijn er vier mogelijkheden voor C.

1. C is een predicaat en er bestaat een predicaatvariabele P en een $Q' \in E^*$, zodat C = PQ'. De formule PQ'D bevindt zich in geval 1. van stelling A2 en is dus een predicaat. Op grond van stelling A1 bestaat er een object V en een $D' \in E^*$, zodat D = VD'.

2. Er bestaat een formule T, zodat C = ¬T. De formule ¬TD bevindt zich in geval 2., dus TD is een formule. Omdat $\ell'(T) < \ell'(C)$ levert de inductieveronderstelling, dat er een object V en een $D' \in E^*$ bestaan, zodat $D = VD'$.
3. Er bestaan formules A en B, zodat C = → AB. De formule → ABD bevindt zich in geval 3., dus er bestaan formules A' en B', zodat ABD = A'B'. Op grond van stelling 6.5 bestaat er een $R \in E^*$, zodat:
   hetzij A' = AR, BD = RB'; als R = Λ, dan B' = BD en verder $\ell'(B) < \ell'(C)$, zodat er op grond van de inductieveronderstelling een object V en een $D' \in E^*$ bestaat, zodat D = VD'; als $R \neq \Lambda$, dan levert de inductieveronderstelling, die toegepast mag worden omdat $\ell'(A) < \ell'(C)$, dat er een object V en een $R' \in E^*$ bestaan, zodat R = VR', dus BD = VR'B', hetgeen onmogelijk is op grond van stelling A2, toegepast op B (het voorste uitgebreide symbool van B kan niet V zijn);
   hetzij A = A'R, B' = RBD, $R \neq \Lambda$; omdat $\ell'(A') < \ell'(A) < \ell'(C)$, passen we de inductieveronderstelling toe, hetgeen oplevert, dat er een object V en een $R' \in E^*$ bestaan, zodat R = VR', B' = VR'BD, hetgeen onmogelijk is op grond van stelling A2, toegepast op B'.
4. Er bestaat een loze variabele X en een formule A, zodat C = ∃XA. De formule ∃XAD bevindt zich in geval 4., dus AD is een formule. Omdat $\ell'(A) < \ell'(C)$, levert de inductieveronderstelling, dat er een object V en een $D' \in E^*$ bestaan, zodat D = VD'.

Stelling A4. Bij iedere formule bestaat er één en slechts één formuledesignator, waarvan die formule de bijbehorende formule is.

Bewijs. Als een formule bij twee designatoren hoort, dan zijn deze of beide predicaten of beide van het type ¬, A of beide van het type →, A, B of beide van het type ∃, X, A. Dit volgt direct uit stelling A1 en stelling A2.
We bewijzen de stelling nu door recurrentie langs de opbouw van de formule.
Als A een predicaat is, dan behoort A bij de designator A en dat is de enige designator, waar A bij kan horen.
Als A een formule is, waarvoor de stelling geldt, dan behoort ¬A bij de designator ¬,A. Stel dat ¬A ook bij de designator ¬,A' behoort, dan ¬A = ¬A', A = A' en de beide designatoren zijn gelijk.

Als A en B formules zijn, waarvoor de stelling geldt, dan behoort → AB bij de designator →, A, B. Stel dat → AB ook behoort bij de designator →,A',B', dan → AB = → A'B', AB = A'B'. Op grond van stelling 6.5 bestaat er een $R \in E^*$, zodat:
hetzij A = A'R, B' = RB; als $R \neq \Lambda$, dan bestaat er op grond van stelling A3 een object V en een $R' \in E^*$, zodat R = VR', B' = VR'B, hetgeen onmogelijk is op grond van stelling A2, toegepast op B'; dus $R = \Lambda$, A = A', B = B' en beide designatoren zijn gelijk;
hetzij A' = AR, B = RB'; analoog als het vorige geval.
Als A een formule is, waarvoor de stelling geldt en X een loze variabele, dan behoort ∃XA bij de designator ∃,X,A. Stel dat ∃XA ook behoort bij de designator ∃,X',A', dan ∃XA = ∃X'A', XA = X'A'. Omdat $\ell'(X) = \ell'(X') = 1$, volgt hieruit X = X' en A = A', dus beide designatoren zijn gelijk.

3. Op blz. 43 midden is het dienstig de opmerking in te lassen, dat in het volgende, in afwijking van hetgeen in § 10 is geschied, teruggekeerd zal worden tot de gangbare terminologie, waarbij de axioma's apart worden genomen en niet als afleidingsregels zonder premisse worden geteld en waarbij een afleiding een lijst van formules is en niet een lijst van afleidingsregels.

4. Onderaan blz. 43 kan de bewijsschets van stelling 10.1 iets worden uitgebreid, door op te merken, dat het bewijs kan worden geleverd door achtereenvolgens de volgende beweringen aan te tonen.

   1. Als L en M afleidingen zijn, dan is ook LM een afleiding.
   2. Als T_0 een theorie en L een lijst van formules is, zodat voor alle formules X, die in L voorkomen, geldt $T_0 \vdash X$, dan is er een afleiding uit T_0, waarin alle formules van L voorkomen.
   3. Als T_0 een theorie is, S een afleidingsregel met conclusie A en als voor alle premissen X van S geldt, dat $T_0 \vdash X$, dan $T_0 \vdash A$.
   4. Als T_0 en T_1 theorieën zijn, $T_0 \vdash X$ voor alle $X \in T_1$ en L een afleiding uit T_1, dan geldt voor alle formules A, die in L voorkomen, dat $T_0 \vdash A$.

5. Op blz. 45 kan het laatste gedeelte van het bewijs van stelling 11.2 als volgt worden gewijzigd.

Stel nu $T \models A$. Verondersteld is, dat T eindig is; we passen volledige inductie toe naar het aantal n van de elementen van T. Als $n = 0$, dan is A een tautologie en dus, op grond van 3., afleidbaar, dus $T \vdash A$. Stel, dat de te bewijzen bewering juist is voor theorieën met n elementen. Stel nu dat T $n+1$ elementen heeft en $T \models A$. Nu is T niet leeg, dus we kunnen een $B \in T$ kiezen. Nu geldt $T\backslash\{B\} \models B \to A$. Immers, laat v een valuatie zijn, zodat voor alle $X \in T\backslash\{B\}$ geldt $v(X) = 1$. Als dan $v(B) = 1$, dan geldt $v(X) = 1$ voor alle $X \in T$, zodat $v(A) = 1$, dus $v(B \to A) = 1$. Als $v(B) = 0$, dan geldt ook $v(B \to A) = 1$. Omdat $T\backslash\{B\}$ n elementen heeft, levert de inductieveronderstelling, dat $T\backslash\{B\} \vdash B \to A$, dus zeker $T \vdash B \to A$. Verder uiteraard $T \vdash B$. Hieruit, uit 4. en stelling 11.1 volgt dan, dat $T \vdash A$.

6. Op blz. 59 midden kan de opmerking worden ingelast, dat, hoewel een Turing-machine gedefinieerd wordt met behulp van uitgebreide symbolen, we toch over de symbolen van de Turing-machine blijven spreken, dat zijn de elementen van $S \cup I \cup \{b,r,\ell\}$.

7. De op blz. 91 regel 12 van boven genoemde Turing-machine ligt op de benoeming der symbolen en het toevoegen van irrelevante band- en toestandssymbolen na, vast.

8. Tenslotte nog enkele drukfouten.

	er staat	er moet staan
blz. 30, regel 10 van boven	benavens	benevens
blz. 51, regel 15 van onder	$A \in U$	$A \in U^*$
blz. 68, regel 10 van onder, diagram, geheel links	M	M_Λ
blz. 90, regel 6 van onder	$\downarrow O\overline{\vert}^{a_0} \ldots O\vert^{a_{\ell-1}}$	$\downarrow O\vert^{a_0} \ldots O\vert^{a_{\ell-1}}$