TECHNISCHE HOGESCHOOL TWENTE

ONDERAFDELING DER TOEGEPASTE WISKUNDE

BEREKENBARE FUNKTIES

door

M.M. Fokkinga

L.A.M. Verbeek

Enschede, juli 1975.

Prijs f. 5,-

Collegedictaat

BEREKENBARE FUNKTIES

door
M.M. Fokkinga.
L.A.M. Verbeek.

Vakcode 59115 Enschede, juli 1975.

Voorwoord.

Dit dictaat Berekenbare Funkties geeft de inhoud van het gelijknamige college.
De bedoeling van het college is de fundementele begrippen berekenbare funktie en beslisbaarheid en opsombaarheid duidelijk uiteen te zetten en de betekenis van deze begrippen voor de wiskunde en vooral voor de informatica aan te geven.
Zodoende wordt enige theoretische ondergrond voor het verwerken van gegevens in de informatica gegeven. Daarnaast wordt hierdoor het lezen mogelijk gemaakt van theoretisch geöriënteerde literatuur in de informatica waarin vaak begrippen voorkomen die samenhangen met berekenbaarheid.
We houden ons aanbevolen voor commentaar op dit dictaat.

Het aanhangsel (A,B,C en D) is afzonderlijk gebonden.

Inhoud

Het Griekse alfabet

Kleine letter	Hoofdletter	Naam
α	A	alfa
β	B	bèta
γ	Γ	gamma
δ	Δ	delta
ε	E	epsilon
ζ	Z	zèta
η	H	èta
θ	Θ	tèta
ι	I	jota
κ	K	kappa
λ	Λ	lambda
μ	M	mu
ν	N	nu
ξ	Ξ	ksi
ο	O	omikron
π	Π	pi
ρ	P	ro
σ	Σ	sigma
τ	T	tau
υ	T	upsilon
φ of ϕ	Φ	fi
χ	X	gi
ψ	Ψ	psi
ω	Ω	omega

1. Inleiding

Er zijn twee manieren waarop in de wiskunde het begrip funktie wordt gedefinieerd en beide zijn van belang in de informatica.

1.1. De eerste, de zgn. extensionele, manier is gebaseerd op verzamelingen en zegt dat een funktie f van verzameling A naar verzameling B, aangeduid met $f: A \to B$, een deelverzameling van het cartesisch product $A \times B$ is, dus $f \subseteq A \times B$, en wel zó dat er voor elke $a \in A$ precies één paar (a,b) in f is met linkerdeel a.
Een voorbeeld vormt de telefoongids die bij elke abonné het bijbehorende nummer geeft. Merk op dat sommige nummers er meer dan eenmaal in voorkomen (Kapper Jansen staat bij de K en bij de J). Bij een digitale computer vormt het interne geheugen een voorbeeld, elk adres heeft een inhoud, en verschillende adressen kunnen een zelfde inhoud hebben. De verzameling A en B van een funktie $f: A \to B$ heten respectievelijk domein en codomein van f. De waardenverzameling, of het bereik van f is $B_f = \{b \mid \exists a \in A[(a,b) \in f]\} \subseteq B$. Vaak worden ook partiële funkties beschouwd waarbij niet elke $a \in A$ als linkerdeel van een paar in f hoeft voor te komen. De definitieverzameling, d.w.z. $D_f = \{a \mid \exists b \in B[(a,b) \in f]\}$, is een deelverzameling van het domein; beide vallen samen bij totale funkties.
Het essentiële van de extensionele opvatting van funkties is: als een argument $a \in A$ gegeven is dan wordt de funktiewaarde f(a) gevonden door het paar (a,b) met linkerdeel a te beschouwen en van dat paar het rechterdeel $b = f(a)$ te nemen. Het kost dus geen moeite om bij gegeven a de bijbehorende b te vinden of te constateren dat er niet zo'n b is bij een partiële funktie.

1.2. De tweede, de zgn. intensionele, manier om een funktie f te definiëren is door middel van een voorschrift waarmee bij een gegeven argument a de funktiewaarde f(a) bepaald wordt. Het domein en ook het codomein van f zijn hierbij vaak, maar niet altijd, impliciet door het voorschrift gegeven. Voorbeelden zijn algebraische funkties zoals $f(x) = 2x^2 + 3x + 6$ waarbij de funktiewaarde, voor bijvoorbeeld $x = 3$, wordt verkregen met behulp van vermenigvuldigen en optellen van natuurlijke getallen, terwijl voor $x = e + i\pi$ deze bewerkingen op complexe getallen moeten worden uitgevoerd.

Een voorbeeld uit de informatica is een sorteerfunktie die een lijst namen alfabetisch ordent door telkens twee opeenvolgende namen te verwisselen als hun volgorde niet alfabetisch is.
Het is plezierig als het voorschrift waarmee een funktie is bepaald ook resultaat oplevert, maar dat hoeft niet altijd het geval te zijn zodat ook weer partiële funkties voorkomen. Denk bijvoorbeeld aan de funktie $f(x) = \sqrt{x^2 - 3}$.

1.3. Bij funkties die gegeven zijn door een voorschrift is het essentieel om precies te weten wat voor soort "voorschriften" gehanteerd mogen of kunnen worden. Dit hangt samen met de bedoeling die we hebben met de te beschouwen funkties. In de rekenkunde laten we bijvoorbeeld voorschriften toe die samengesteld zijn uit optellen, aftrekken, vermenigvuldigen en delen van rationale getallen. In de analyse laten we bovendien ook integreren en differentieren van funkties van reële getallen toe. Bij het bestuderen van de grondslagen van de wiskunde is het criterium voor het toelaten van voorschriften kun "berekenbaarheid". De vraag luidt dan: welke funkties zijn berekenbaar, d.w.z. van welke funkties is de bepaling van de waarde bij willekeurig gegeven argument effectief uit te voeren, en welke niet ? Hierover bestaat een uitgebreide literatuur waarnaar we in het vervolg nog nader zullen verwijzen.
De berekenbaarheid van funkties wordt geformuleerd en ook bestudeerd met behulp van het begrip algoritme of effectieve procedure (we zullen deze woorden als synoniem beschouwen). Het gaat daarbij altijd om een procedure of handelwijze waardoor een symboolrij stap voor stap in een andere symboolrij wordt omgezet. Bij een effectieve procedure zijn altijd twee dingen van belang, nl. een collectie regels of voorschriften voor het handelen en een manier om die regels slaafs en mechanisch toe te passen. In termen van digitale computers gesteld is zowel het programma als de computer die het programma uitvoert of interpreteert van belang. De geprogrammeerde computer verwerkt invoergegevens volgens de algoritme die in het programma is vervat. Globaal kunnen we zeggen dat een algoritme of effectieve procedure een precies aangegeven manier van bewerking van symboolrijen is die in opeenvolgende stappen verloopt. De preciese beschrijving van zo'n manier bestaat uit een eindige collectie regels of voorschriften die op elk moment in het proces, d.w.z. de bewerking van de symboolrij, eenduidig voorschrijven wat de volgende

stap is, tesamen met alle benodigde details van een mechanisme dat de procedure kan uitvoeren, dus de regels interpreteert. Voor meer uitgewerkte beschouwingen over het begrip algoritme zie o.a. § 1 - § 4 in Hermes (1971), hoofdstuk 5 in Minsky (1967) en § 1.1 in Rogers (1967).

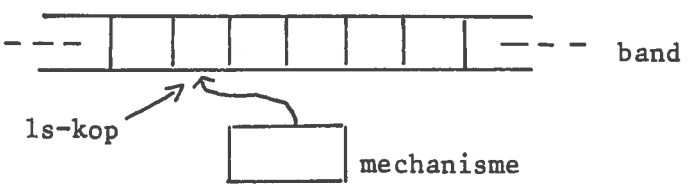
1.4. Voor de informatica is de studie van berekenbare funkties van fundamenteel belang omdat het inzicht geeft in welke funkties principieel wel of niet te realiseren zijn met geidealiseerde digitale computers. De idealisering bestaat uit het aannemen dat zowel het geheugen van de computer als de voor een berekening benodigde tijd onbegrensd, maar eindig, zijn. Daarnaast zijn bij de studie van berekenbare funkties een aantal beschrijvingswijzen ontwikkeld die nuttig of nodig zijn bij de behandeling van beslissingsvragen als: welke funkties zijn (onderling) gelijkwaardig en in welke zin, waarin verschillen ze ? Hierdoor is er enig gereedschap om vragen naar goed en zinnig gebruik van digitale computers te behandelen (maar lang niet altijd te beantwoorden!). Bovendien worden een aantal resultaten betreffende berekenbaarheid en beslisbaarheid gebruikt in de theorie van formele talen. Deze theorie vormt een grondslag bij onderwerpen uit de informatica, met name bij het definiëren van programmeertalen en het bouwen van vertaalprogramma's zoals compilatoren en interpretatoren.

1.5. Van de verschillende manieren waarop door logici de hierboven aangeduide begrippen effectieve procedure, berekenbaarheid en beslisbaarheid zijn omgezet in formele, dus ook wiskundig-preciese, definities bekijken we eerst de zgn. Turingmachine. Daarna worden recursieve funkties behandeld en tenslotte komt nog een formalisering met zgn. Markovalgoritmen aan de orde. De keuze van de onderwerpen en de manier van behandelen die hier gevolgd wordt is mede bedoeld om het lezen van vak-literatuur in de informatica te vergemakkelijken.

2. Turingmachines en berekenbare funkties.

In 1936 publiceerde de Engelse logicus Alan M. Turing (1912 - 1954) een artikel waarin hij voor het intuitieve begrip berekenbare funktie een bepaalde preciese formulering gaf. Deze formele definitie berust op een denkbeeldige machine en haar werking; zo'n machine noemt men Turingmachine (kortweg: Tm). Hoewel een Tm een puur wiskundige constructie is, is de definitie ervan gemakkelijker te begrijpen als we haar eerst als een fysisch apparaat beschrijven.

2.1. Een Tm kunnen we ons voorstellen, zie onderstaande figuur, te bestaan uit een mechanisme samen met een naar links en rechts onbegrensde horizontale band die in onderling gelijke hokjes is verdeeld. In elk hokje van de band kan één letter uit het alfabet van de Tm staan.



Beeld van een Turingmachine.

Op elk tijdstip van een discrete tijdschaal is het mechanisme in één van een eindig aantal mogelijke toestanden. Het mechanisme is aan de band gekoppeld met de lees-schrijfkop (ls-kop) die op elk tijdstip op een hokje staat. De toestand van het mechanisme, samen met de letter in het hokje waarop de ls-kop staat, bepaalt wat er gebeurt. Dat kan zijn: òfwel helemaal ophouden met werken, dus stoppen, òfwel het mechanisme gaat over in een nieuwe toestand terwijl de ls-kop in het hokje waarop hij staat een nieuwe letter schrijft in plaats van de daar gelezen letter en dan één hokje hetzij naar links, hetzij naar rechts, opschuift. Het onbegrensd zijn van de band van de Tm vatten we als volgt op: op elk tijdstip is de band eindig, maar als de ls-kop er aan de linker- of rechterkant dreigt af te lopen wordt daar vanzelf een blanco hokje aan de band toegevoegd en er is geen beperking aan het aantal keren dat dit kan gebeuren.

Bij de wiskundige behandeling van Tms zullen we voor intuitieve uitleg vaak teruggrijpen naar het hierboven gegeven beeld van een Tm als een fysisch apparaat.

2.2. Definitie Een Turingmachine T is een drietal $T = (A,Q,\tau)$, waarvoor geldt:

(i) A is een nietlege eindige verzameling; A heet het alfabet van T, de elementen van A heten letters.

(ii) Q is een nietlege eindige verzameling; Q heet de toestandverzameling van T, de elementen van Q heten toestanden.

(iii) τ is een afbeelding van een deelverzameling van $Q \times (A \cup \{\perp\})$ naar $Q \times A \times \{L,R\}$; hierbij is $\perp$ het zgn. blanco symbool van T, $\perp$ komt niet in het alfabet voor, $\perp \notin A$. τ heet de overgangs- of transitie funktie van T.

In elk hokje dat aan de band van T wordt toegevoegd staat het blanco symbool $\perp$. Hierdoor wordt gesignaleerd dat er een hokje is toegevoegd waarop nog geen letter staat, de Tm kan $\perp$ wel lezen maar niet schrijven. De tekens L en R in het rechterdeel van de beeldverzameling van τ duiden aan dat de ls-kop van de Tm een hokje naar links, respectievelijk naar rechts, opschuift.

2.3. Voorbeeld $T = (\{a,b,\#\},\{q_1,q_2,q_3,q_4\},\tau)$ waarin τ als volgt is:

	a	b	#	$\perp$
q_1	q_1aL	q_1bL	-	$q_2\#R$
q_2	q_2aR	q_3bR	$q_2\#R$	-
q_3	q_4bL	q_3bR	-	-
q_4	-	q_2aL	-	-

ofwel, op een andere manier gegeven,

$\tau(q_1,a) = (q_1,a,L)$	ofwel	q_1aq_1aL
$\tau(q_1,b) = (q_1,b,L)$		q_1bq_1bL
$\tau(q_1,\perp) = (q_2,\#,R)$		$q_1\perp q_2\#R$
$\tau(q_2,a) = (q_2,a,R)$		q_2aq_2aR
$\tau(q_2,b) = (q_3,b,R)$		q_2bq_3bR
$\tau(q_2,\#) = (q_2,\#,R)$		$q_2\#q_2\#R$
$\tau(q_3,a) = (q_4,b,L)$		q_3aq_4bL
$\tau(q_3,b) = (q_3,b,R)$		q_3bq_3bR
$\tau(q_4,b) = (q_2,a,L)$		q_4bq_2aL

2.4. Omdat τ een deelverzameling van $Q \times (A \cup \{\perp\})$ afbeeldt kan het voorkomen dat voor een $q \in Q$ in een $a \in A$ het beeld $\tau(q,a)$ of $\tau(q,\perp)$ niet gedefinieerd is, in dat geval stopt de Tm. Een Tm is daarom een incomplete machine. In voorbeeld (2.3) zijn onder andere $\tau(q_1,\#)$, $\tau(q_2,\perp)$ en $\tau(q_4,a)$ ongedefinieerd.
Zoals in voorbeeld (2.3) al is aangegeven kan de transitiefunktie van een Tm worden gegeven in de vorm van een transitietabel, daarin vallen de situaties waarin de Tm stopt goed op omdat daarbij de tabel leeg is.
In voorbeeld (2.3) is ook nog aangegeven hoe τ kan worden weergegeven met een eindig aantal vijftallen van de vorm

$$q_ia_jq_ka_1d$$

met $q_i,q_k \in Q$ en $a_j \in A\cup\{\perp\}$ en $a_1 \in A$ en $d \in \{L,R\}$. De letter d komt van "direction", Engels voor "richting".
Hierbij mogen geen twee vijftallen met hetzelfde tweetal q_ia_j beginnen want een Tm is een deterministische machine omdat τ een (partiële) funktie is.
Elk van de drie manieren om τ weer te geven is ook voldoende om de hele Tm te bepalen, immers A en Q bestaan precies uit de letters en toestanden die in τ voorkomen als we aannemen dat er geen overbodige elementen in A en Q zijn.

2.5. Voor het bekijken van de werking van een Tm T moeten we de situatie van T op elk tijdstip kunnen aangeven. We maken hiervoor gebruik van de zgn. bandexpressie van T, d.w.z. de rij letters uit A die op de band staat. Verder moeten we nog aangeven wat de toestand q van T is en op welke hokje de ls-kop staat. Dit doen we door in de bandexpressie het symbool q te zetten links van de letter in het betreffende hokje. De zo verkregen symboolrij heet de configuratie van T. Bij voorbeeld (2.3) geeft de configuratie aaq_2ba dus aan dat de ls-kop op het hokje staat waarin de letter b is geschreven, dat de toestand q_2 is en dat de bandexpressie aaba is.

Definitie Zij $T = (A,Q,\tau)$ een Turingmachine. Een configuratie C van T is een eindige symboolrij waarin op één plaats een element q van Q voorkomt en verder alleen elementen van $A \cup \{\perp\}$ voorkomen terwijl q niet het rechtersymbool van C is. De bandexpressie van een configuratie C van T is het woord over A dat uit C ontstaat door het symbool q en de blanco-symbolen $\perp$ er uit weg te laten.

Voor de Tm T van voorbeeld (2.3) zijn q_2aaba, $ab\#q_3b$, $babq_4\perp$ en baq_1ba configuraties en de bijbehorende bandexpressies zijn:

aaba, ab#b, bab en baba.

De rijen abq_1 baq_2a en $ab\#q_3$ zijn geen configuraties. Ga na waarom niet!

2.6 We gaan nu kijken hoe een Tm werkt.
Als in voorbeeld (2.3) de configuratie $C_1 = q_1$ aba is dan volgt uit $\tau(q_1,a) = (q_1,a,L)$ dat de volgende configuratie $C_2 = q_1\perp$ aba zal zijn. Immers de ls-kop zal links van de band af dreigen te lopen zodat er een hokje met blanco symbool $\perp$ aan de band wordt toegevoegd in overeenstemming met de opvatting over het onbegrensd zijn van de band die in (2.1) is gegeven. Uit $\tau(q_1,\perp) = (q_2,\#,R)$ volgt dat de daarna volgende configuratie is : $C_3 = \# q_2aba$. De letter # markeert nu de linkerkant van de bandexpressie. Vervolgen we deze beschouwing dan ontstaat

de volgende rij configuraties:

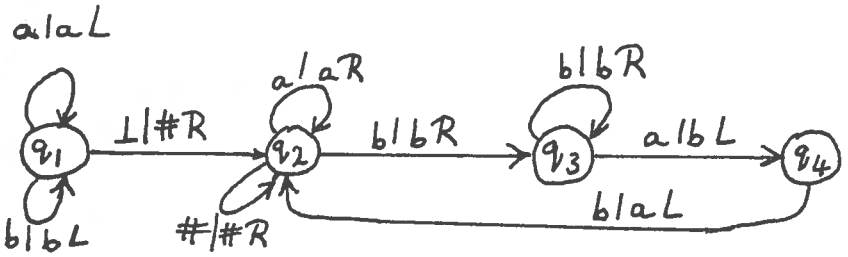
$C_1 = q_1aba$ $\quad C_5 = \#abq_3a$ $\quad C_9 = \#aaq_2b$

$C_2 = q_1\perp aba$ $\quad C_6 = \#aq_4bb$ $\quad C_{10} = \#aabq_3\perp$

$C_3 = \#q_2aba$ $\quad C_7 = \#q_2aab$

$C_4 = \#aq_2ba$ $\quad C_8 = \#aq_2ab$

Bij de aanvang van C_9 naar C_{10} is de band van Tm T aan de rechterkant verlengd met een hokje waarop $\perp$ geschreven is, weer in overeenstemming met de opvatting over het onbegrensd zijn van de band die in (2.1) is gegeven. Aangezien $\tau(q_3,\perp)$ niet gedefinieerd is stopt de Tm in de configuratie C_{10} en de bandexpressie is dan $\#aab$. Voor het bekijken van de werking van een Tm is het vaak handig deze weer te geven met een transitiediagram. Dat is een gerichte en benoemde graaf waarin de toestanden zijn gegeven als de benoemde knopen en voor elk vijftal een benoemde pijl voorkomt. Van een knoop met naam q_1 gaat een pijl met naam $a_1|a_2d$ naar een knoop met naam q_2 als en alleen als $\tau(q_1,a_1) = (q_2,a_2,d)$.
Het transitiediagram van de Tm van voorbeeld (2.3) is als volgt:



Transitiediagram van Tm T van voorbeeld (2.3).

Aan de hand van dit diagram is het gemakkelijk te verifiëren dat uit de configuratie $C_1 = q_1aba$ stap voor stap de configuraties $C_2, C_3, \ldots, C_{10}$ ontstaan en Tm T stopt in configuratie $C_{10} = \#aabq_3\perp$ met $\#aab$ als bandexpressie.

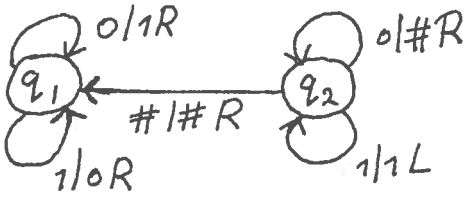
2.7. Definitie Zij $T = (A,Q,\tau)$ een Turingmachine en zijn C en C' twee configuraties van T. We zeggen C gaat over in C', of T stapt van C naar C', notatie: $C \vdash C'$, als en alleen als één van de volgende vier gevallen zich voordoet:
er zijn $w_1, w_2 \in A^*$ en $a \in A \cup \{\perp\}$ en a', $b \in A$ en $q,q' \in Q$ zó dat

1. $C = w_1qabw_2$ en $C' = w_1a'q'bw_2$ en $\tau(q,a) = (q',a',R)$
2. $C = w_1qa$ en $C' = w_1a'q'\perp$ en $\tau(q,a) = (q',a',R)$
3. $C = w_1bqaw_2$ en $C' = w_1q'ba'w_2$ en $\tau(q,a) = (q',a',L)$
4. $C = qaw_2$ en $C' = q'\perp a'w_2$ en $\tau(q,a) = (q',a',L)$

N.B. Voor de betekenis van A^* zie aanhangsel A.

In deze definitie volgen de stappen van T in de gevallen 1 en 3 gewoon uit de transitiefunctie τ van T. In de gevallen 2 en 4 dreigt de ls-kop van de band af te lopen aan de rechter- respectievelijk de linkerkant en wordt de band verlengd met een hokje waarop het blanco symbool $\perp$ is geschreven. De werking van een Tm met gegeven configuratie bestaat uit het stappen van configuratie naar configuratie tot zij niet meer in een volgende kan overgaan. De ontstane rij configuraties is het resultaat van de werking.
Een voorbeeld van het resultaat van de werking van Tm T van voorbeeld (2.3) is de al eerder gegeven rij configuraties $C_1, C_2, \ldots, C_{10}$. Andere voorbeelden zijn de rijen:
baq_3ba, $babq_3a$, baq_4bb, bq_2aab, baq_2ab, $baaq_2b$, $baabq_3\perp$ en
q_1ba, $q_1\perp ba$, $\#q_2ba$, $\#bq_3a$, $\#q_4bb$, $q_2\#ab$, $\#q_2ab$, $\#aq_2b$, $\#abq_3\perp$
Verifieer dit aan de hand van het transitiediagram (2.6). Het volgende voorbeeld geeft ook nog enkele resultaten van de werking van een Tm.

2.8. <u>Voorbeeld</u> Een Tm is gegeven door het volgende transitiediagram:



Verifieer zelf: deze Tm gaat vanuit
configuratie q_101001 in 5 stappen over in $10110q_1\perp$ en stopt,
" $010q_101$ " 2 " " " $01010q_1\perp$ " "
" q_201001 " 7 " " " $\#0110q_1\perp$ " "
" $010q_201$ " 4 " " $010\#0q_1\perp$ " "

Uit dit voorbeeld blijkt dat een bandexpressie tot verschillende resultaten leidt afhankelijk van de toestand en van de positie van de ls-kop bij het begin van de werking. Vandaar de volgende definitie

2.9. Definitie Zij $T = (A,Q,\tau)$ een Turingmachine en zij $q_o \in Q$ een vast element van Q, de zgn. begintoestand van T.

a) Een configuratie C van T is een beginconfiguratie als het linkersymbool van C juist q_o is; C is een eindconfiguratie als er geen configuratie van T is waarin C kan overgaan.

b) Een berekening van T is een eindige rij configuraties $(C_1,C_2,\ldots,C_n)$ van T zó dat C_1 een begin- en C_n een eindconfiguratie is en C_i overgaat in C_{i+1} voor $1 \leq i \leq n-1$.

Opmerkingen. 1. Van nu af zullen we bij elke Tm T een begintoestand $q_o \in Q$ geven en deze opnemen in de definitie van T zodat een Tm in het vervolg een viertal van de vorm (A,Q,τ,q_o) is.

2. De bandexpressies van een begin- of eindconfiguratie zullen we vaak beginbandexpressie of eindbandexpressie noemen.

Bekijk de Tm T van voorbeeld (2.3) met begintoestand q_1, dus $T = (\{a,b,\#\},\{q_1,q_2,q_3,q_4\},\tau,q_1)$. Een berekening van T is de al eerder gegeven rij $(C_1,C_2,\ldots,C_{10})$. Andere berekeningen van T zijn $(q_1\perp,\#q_2\perp)$ en $(q_1ab,q_1\perp ab,\#q_2ab,\#aq_2b,\#abq_3\perp)$.

2.10. Voor we nader ingaan op berekeningen van Tms geven we eerst enkele oefeningen:

1. Ga na dat bij de Tm van voorbeeld (2.3) met begintoestand q_1 elke berekening er op neerkomt dat een bandexpressie $w \in \{a,b\}^*$ wordt omgezet in een bandexpressie waarbij de letter # links van w wordt gezet en alle letters a die in w voorkomen links van de letters b die in w voorkomen worden geschreven. Dus $\varepsilon \in \{a,b\}^*$ wordt omgezet in #!
2. Wat is de eindconfiguratie van de Tm T van voorbeeld (2.3) als de beginconfiguratie $q_1$baab#ba is ? En als de beginconfiguratie $q_1w_1\#w_2$ is waarbij $w_1 \in \{a,b\}^*$ en $w_2 \in A^* = \{a,b,\#\}^*$.
3. Wat is de eindbandexpressie van de Tm van voorbeeld (2.8) met

begintoestand q_2 als de beginconfiguratie q_2w is met $w \in \{0,1\}^*$?
Beantwoord dezelfde vraag met q_1 in plaats van q_2 als begintoestand.

4. Zij $T = (\{0,1\},\{q_o,q_1,q_2\},\tau\ q_o)$ een Tm waarbij τ bepaald is door de volgende vijftallen:

$q_o 0 q_o 0 R$ $\quad$ $q_o \perp q_2 0 L$ $\quad$ $q_1 1 q_o 0 R$

$q_o 1 q_1 0 R$ $\quad$ $q_1 0 q_1 0 R$ $\quad$ $q_1 \perp q_2 1 L$

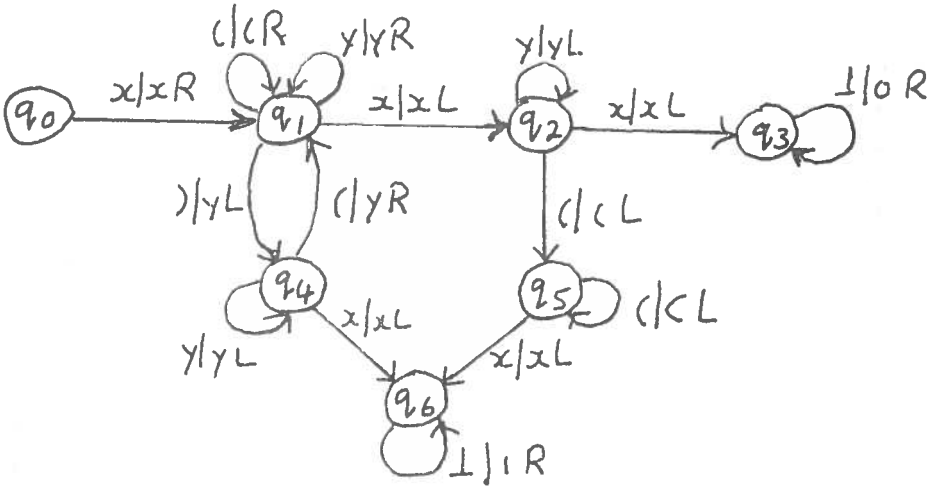
Geef het transitiediagram van T,
Wat is de eindbandexpressie van T als de beginbandexpressie ε of 101 of 10011 of 0100101 is ?
Verifieer dat T elk woord $w \in \{0,1\}^*$ omzet in een rij 0-en van dezelfde lengte als w met een 0 of 1 erachter als het aantal 1-en in w even respectievelijk oneven is.

5. Zij $T = (\{a,b,0,1,\#\},\{q_o,q_1,q_2\}, \tau, q_o)$ een Tm waarbij τ bepaald is door de volgende tabel:

	a	b	$\perp$
q_o	$q_o \# R$	$q_1 \# R$	$q_2 0 R$
q_1	$q_2 1 R$	$q_1 \# R$	$q_2 0 R$
q_2	$q_2 \# R$	$q_2 \# R$	-

terwijl de kolommen onder 0, 1 en # leeg zijn.
Geef het transitiediagram van T.
Wat is de eindbandexpressie van T als de beginbandexpressie aabb of bbaa of abbab is ?
Omschrijf in het algemeen wat T doet met een woord over {a,b}.

6. Tm T wordt gegeven door het volgende transitiediagram:



Geef T in de vorm van een transitietabel en van vijftallen. Verifieer dat $q_0x(()x$ door T wordt omgezet in $1q_6x(yyx$ en dat $q_0x(()())x$ wordt omgezet in $0q_3$ xyyyyyyx. Zij een haakjesvorm w gegeven, dus $w \in \{(,)\}^*$, verifieer dat de beginconfiguratie q_0xwx van T wordt omgezet in $0q_3xwx$ of $1q_6xw''x$ als w een wel, respectievelijk een niet, gebalanceerde haakjesvorm is; hierbij zijn w', $w'' \in \{(,),y\}^*$.

2.11. Na deze oefeningen bekijken we berekeningen van Tms nog wat nader. Het is mogelijk dat een Tm T vanuit een beginconfiguratie C geen eindconfiguratie bereikt maar steeds nieuwe configuraties of een bepaalde cyclus van configuraties doorloopt. In dat geval stopt T niet en er is geen berekening die met C begint.

Voorbeeld Zij $T = (\{a,b,c,\#\},\{q_0,q_1\},\tau,q_0)$ en Tm waarbij τ bepaald is door de volgende vijftallen:

q_0aq_0aL	$q_0\perp q_0\#R$	$q_0\#q_0\#R$
q_0bq_1bR	$q_0cq_0\#R$	q_1aq_0cR

Verifieer de volgende uitspraken, eventueel met behulp van het transitiediagram van T:
De beginconfiguratie q_0abc levert de volgende configuraties op:

q_0abc, $q_0\perp abc$, $\#q_0abc$, $q_0\#abc$, $\#q_0abc$,. . .

De beginconfiguratie q_0bac levert:
q_0bac, bq_1ac, bcq_0c, $bc\#q_0\perp$, $bc\#\#q_0\perp$, . . .

De beginconfiguratie q_0bca levert de berekening:
(q_0bca, bq_1ca).

2.12. Teruggrijpend naar de intuitieve beschrijving van een algoritme in de inleiding, zie (1.3), is het niet moeilijk om aan te nemen dat elke Tm een effectieve procedure realiseert. De regels zijn gegeven in de vorm van de transitiefunktie en de Tm werkt precies volgens deze regels. Elke berekening is het door de Tm uitgevoerde proces of de reeks stappen waardoor de beginbandexpressie wordt omgezet in de eindbandexpressie, terwijl het proces niets oplevert als de Tm niet stopt. Elke Tm realiseert dan ook een partiële funktie van bandexpressies naar bandexpressies. Enigszins misleidend zegt men gewoonlijk dat de Tm die funktie "berekent", beter zou zijn: de Tm realiseert die funktie.

Definitie. Zij $T = (A,Q,\tau,q_o)$ een Turingmachine. De partiële funktie f_T berekend door T is de partiële funktie van A^* naar A^* die voor $w \in A^*$ de waarde $f_T(w) = w'$ heeft, waarbij w' de eindbandexpressie is van de berekening van T die begint met $q_o w$, terwijl $f_T(w)$ ongedefinieerd is als er geen berekening van T is die begint met $q_o w$

2.13. De funktie f_T die een Tm T berekent heeft als domein en codomein de vrije monoide A^* gegenereerd door het alfabet A van T (zie aanhangsel A voor uitleg van deze begrippen en notaties). Om van deze beperking af te komen en funkties met willekeurig domein V en willekeurig codomein W te kunnen bekijken gebruiken we een codering van V naar A^* en een decodering van A^* naar W. Onder codering verstaan we hier een totale afbeelding $\gamma : V \to A^*$ die injectief is, d.w.z. zodat voor alle $v_1, v_2 \in V$ uit $\gamma(v_1) = \gamma(v_2)$ volgt $v_1 = v_2$. Onder een decodering verstaan we een afbeelding $\delta : A^* \to W$. Zowel codering γ als decodering δ moet een intuitief berekenbare funktie zijn.
Met behulp hiervan definiëren we Turing berekenbaarheid.

Definitie. Zij $V \to W$ een partiële funktie van V naar W. f is Turing berekenbaar (kortweg: T- berekenbaar) als er een Turingmachine T met alfabet A bestaat en een codering $\gamma : V \to A^*$ en een decodering $\delta: A^* \to W$ zodanig dat voor elke $v \in V$ geldt: $\delta(f_T(\gamma(v))) = f(v)$.
(dus de funkties $\delta f_T \gamma$ en f hebben dezelfde definitieverzameling)
m.a.w. zodanig dat het volgende diagram commuteert:

$$\begin{array}{ccc} V & \xrightarrow{f} & W \\ {\scriptstyle\gamma}\downarrow & & \uparrow{\scriptstyle\delta} \\ A^* & \xrightarrow{f_T} & A^* \end{array}$$

Voorbeelden van T-berekenbare funkties zijn bijvoorbeeld de funktie in opgave 1 van (2.10) en de rekenkundige funkties in hoofdstuk 3. Voorbeelden van niet T-berekenbare funkties zullen we tegenkomen in hoofdstuk 6.

In de definitie staat niet dat voor elke T-berekenbare funktie f een Tm T en een codering γ en een decodering δ geconstrueerd kunnen worden zodat $\delta f_T \gamma = f$. Er staat alleen maar dat zo'n T en γ en δ bestaan. Het verschil wordt wellicht duidelijk door het volgende voorbeeld:
In een brief aan Euler in 1742 schreef C. Goldbach (1690-1764) een

getaltheoretische opmerking die later, onder de naam "vermoeden van Goldbach", de volgende formulering heeft gekregen: (en nog steeds niet als waar of onwaar is bewezen)

(G) Elk even getal groter dan twee is de som van twee priemgetallen. Bekijk nu de funktie f : N → N (N is de verzameling van alle natuurlijke getallen) gegeven door:

$$f(x) = \begin{cases} 0 \text{ als (G) waar is} \\ 1 \text{ als (G) onwaar is} \end{cases}$$

Het is evident dat f een constante funktie is en uiteraard ook T-berekenbaar is. Maar zolang de waarheid of onwaarheid van het vermoeden van Goldbach niet bewezen is kunnen we de betreffende Tm niet aanwijzen hoewel die zeker bestaat.

2.14. Het is duidelijk dat elke totale of niet-totale funktie die T-berekenbaar is inderdaad zonder bezwaar intuitief berekenbaar kan worden genoemd. Immers de Tm, samen met de codering en decodering geeft een volledige en effectieve beschrijving van de procedure volgens welke de funktiewaarde wordt berekend voor elk argument waarvoor de funktie gedefinieerd is.
Het omgekeerde luidt: voor elke in intuitieve zin berekenbare funktie f : V → W bestaat er een Tm T met alfabet A en codering $\gamma : V \to A^*$ en decodering $\delta : A^* \to W$ zódat $f = \delta f_T \gamma$ Dat wil zeggen: voor elke waarde a in het domein V van f is f(a), als f(a) bestaat, gelijk aan $\delta f_T \gamma(a)$ en ook omgekeerd als $\delta f_T \gamma(a)$ bestaat, dan bestaat f(a) en is eraan gelijk. Met andere woorden: als $a \in V$ volgens γ gecodeerd is als beginbandexpressie $\gamma(a)$ voor T, dan stopt Tm T na een eindig aantal stoppen als en alleen als f(a) bestaat en als T stopt staat f(a), na decodering volgens δ, op de band van T.
Kortweg: elke intuitief berekenbare funktie is Turingberekenbaar.
Bovenstaande uitspraken vormen de zgn. <u>These van Turing</u>. Deze these van Turing is een variant van de zgn. these van Church. Het is eigenlijk een veronderstelling omdat het intuitieve begrip "berekenbaar" er door gelijk gesteld wordt aan "Turingberekenbaar" Dit kan uiteraard niet in wiskundige zin bewezen worden. De these

van Turing is gemakkelijk te aanvaarden omdat alle andere manieren om het intuitieve begrip berekenbare funktie op een formele manier te beschrijven (zoals o.a. met recursieve funkties en met Markov algoritmen) gelijkwaardig zijn gebleken. Ook varianten van Tms, zoals die met meer banden en lees- schrijfkoppen, of met een twee-dimensionale band, of die niet deterministisch werken, blijken alle dezelfde klasse van partiële funkties te realiseren als de Tms zoals wij ze hier gedefinieerd hebben. Zie bijvoorbeeld Fischer (1965).

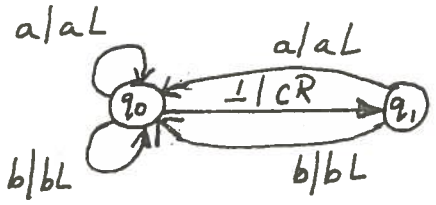
2.15. We zullen nu twee varianten van Tms wat nader bekijken. Vaak, o.a. in Davis (1958), worden Tms niet door vijftallen bepaald maar door viertallen en wel voor het eerst in Post (1947). We zullen zo'n mechanisme hier, ter onderscheiding, pseudo-Turingmachine noemen. Elk viertal is van de vorm $q_i a_j q_k a_l$ of van de vorm $q_i a_j q_k d$ met $d \in \{L,R\}$. Elke stap van een pseudo-Tm wordt bepaald door zo'n viertal en bestaat ofwel in het schrijven van a_l in plaats van a_j in het hokje onder de ls-kop en het overgaan van toestand q_i naar q_k, ofwel in het opschuiven van een hokje naar links of naar rechts, als $d = L$ of $d = R$, en het overgaan van toestand q_i naar q_k. Bij een pseudo-Tm zijn het veranderen van de letter in een hokje en het opschuiven van de ls-kop dus gescheiden. Uiteraard is voor elke Tm een gelijkwaardige pseudo-Tm te construeren en ook omgekeerd is voor elke pseudo-Tm een gelijkwaardige Tm te construeren. Verifieer dit zelf, eventueel naar aanleiding van het volgende voorbeeld.

Voorbeeld Hieronder geven we de vijftallen van de Tm T van voorbeeld (2.3) met ernaast de viertallen van een gelijkwaardige pseudo-Tm T'.

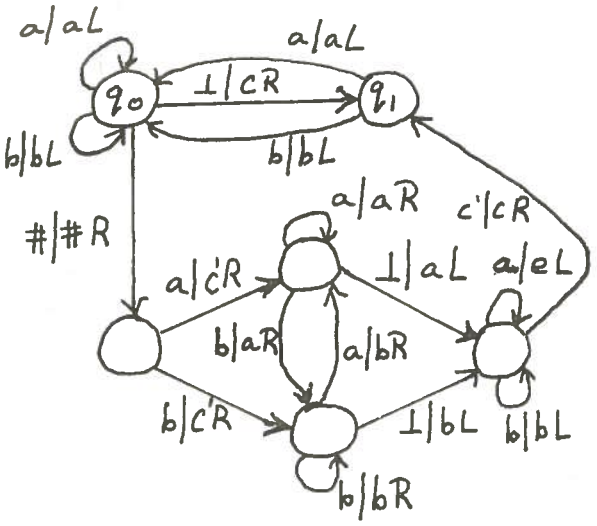
T :	T' :
q_1aq_1aL	q_1aq_1L
q_1bq_1bL	q_1bq_1L
$q_1\sqcup q_2\#R$	$q_1\sqcup q_1'\#$
	$q_1'\#q_2R$
q_2aq_2aR	q_2aq_2R
q_2bq_3bR	q_2bq_3R
$q_2\#q_2\#R$	$q_2\#q_2R$
q_3aq_4bL	$q_3aq_3'b$
	$q_3'bq_4L$
q_3bq_3bR	q_3bq_3R
q_4bq_2aL	$q_4bq_4'a$
	q'_4aq_2L

2.16. Een andere variant van een Tm is de zgn. eenzijdige Turingmachine. Daarbij is de band slechts aan één kant, bijvoorbeeld de rechterkant, onbegrensd terwijl aan de linkerkant een merkteken staat, bijvoorbeeld #, dat nooit overschreden wordt door de ls-kop van de Tm. We zullen zien dat elke Tm vervangen kan worden door een ermee gelijkwaardige eenzijdige Tm.

Voorbeeld De Tm T is gegeven door onderstaand transitiediagram:



Verifieer dat Tm T elk woord w ∈ {a,b}* onsvormt door er een c voor te zetten; beginconfiguratie q_0w levert eindconfiguratie q_0cw zodat $f_T(w) = cw$.
Bekijk nu de Tm T' gegeven door onderstaand diagram:



Verifieer dat Tm T' elke configuratie $\#q_0w$, met w ∈ {a,b}*, omvormt in $\#q_0cw$ waarbij het symbool # op hetzelfde hokje van de band van T' is blijven staan. Als we nu definitie (2.12) aanpassen door het merkteken # bij eenzijdige Tms buiten beschouwing te laten is $f_{T'}(w) = cw$.
De eenzijdige Tm T' is een uitbreiding van de gegeven Tm T en

werkt precies als T tenzij de ls-kop van T links van de band af dreigt te lopen, dus de ls-kop van T' op het hokje komt waarin het symbool # staat dat het linkereind van de band van T' markeert. In dat geval wordt de hele bandexpressie van T' één hokje naar rechts opgeschoven en de betreffende stap van T uitgevoerd. Daarna werkt T' weer verder net zoals T.

De manier waarop in bovenstaand voorbeeld de eenzijdige Tm T' de werking van de Tm T simuleert is algemeen en kan expliciet worden gegeven.

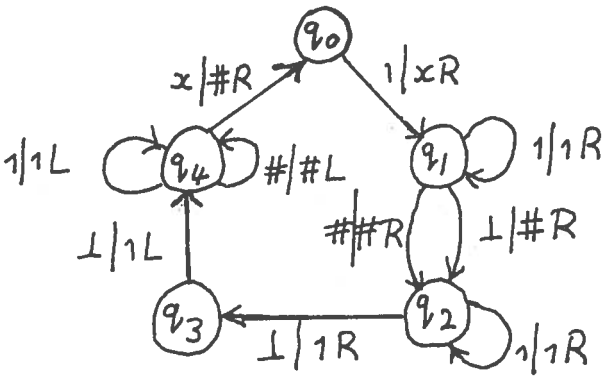
Oefening. Doe dit.

Het resultaat is: Voor elke Tm T kan een eenzijdige Tm T' uit T worden geconstrueerd zódat $f_{T'} = f_T$.

3. Turingmachines en rekenkundige funkties.

3.1. Om een funktie $f : V \to W$ te realiseren met een Tm, dus om f te berekenen, moeten we de argumenten van f, d.w.z. de elementen van V, als bandexpressies van de Tm kunnen schrijven en de resulterende bandexpressies kunnen lezen als elementen van W. We zullen dit coderen en decoderen in concreto doen voor enkele funkties die natuurlijke getallen omzetten in natuurlijke getallen, dus voor rekenkundige funkties.
Het is de gewoonte hierbij ook 0 als natuurlijk getal op te vatten en $n \in N = \{0,1,2,\ldots\}$ te coderen als n maal de letter 1. Met n associeren we dus de bandexpressie $\overline{n} = \underbrace{1\ 1\ \ldots\ 1}_{n}$ zodat $\overline{0} = \varepsilon$, $\overline{1} = 1$, $\overline{2} = 11$, $\overline{3} = 111$ enz. Bij het decoderen wordt ook een simpele regel gevolgd, m maal de letter 1 staat voor het getal m terwijl eventueel voorkomende andere letters in de bandexpressie (die bij het berekeningsproces gebruikt zijn) worden genegeerd.
Als beginbandexpressie $\overline{n} = \underbrace{11\ldots1}_{n}$ door Tm T wordt omgezet in een eind-bandexpressie waarin m maal de letter 1 voorkomt, dan zullen we schrijven $f_T(\overline{n}) = \overline{m}$. Hoewel het nogal slordig is maken we dus geen onderscheid tussen de bandexpressie die het gecodeerde getal n is en een bandexpressie die na decodering het getal n is, beide schrijven we als $\overline{n}$.

3.2. Voorbeeld. Zij $T = (\{1,x,\#\},\{q_0,q_1,q_2,q_3,q_4\}\ \tau,\ q_0)$ een Tm waarbij τ gegeven is door het volgende diagram:



Verifieer dat T de beginconfiguratie

$q_0\perp$ omzet in $q_0\perp$, dus dat $f_T(\overline{0}) = \overline{0}$

$q_0 1$ omzet in $\#q_0\#11$, dus dat $f_T(\overline{1}) = \overline{2}$

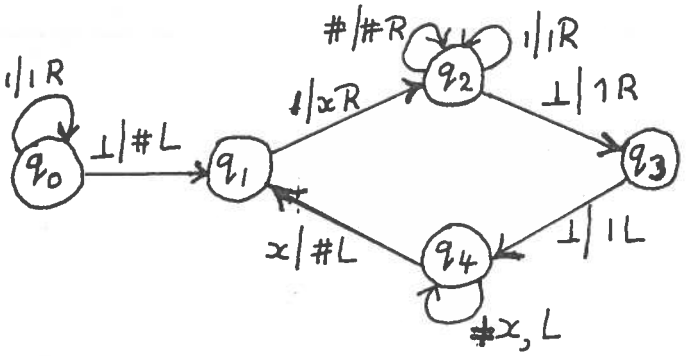
$q_0 11$ omzet in $\#\#q_0\#1111$, dus dat $f_T(\overline{2}) = \overline{4}$.

Verifieer in het algemeen dat $f_T(\overline{n}) = \overline{2n}$ voor elke $n \in N$, dus dat T werkt als vermenigvuldiger met 2.

Geef de vijftallen en de transitietabel van T.

Geef de berekening van 2×3 m.b.v. deze representaties van T.

3.3. Voorbeeld. De Tm $T' = (\{1,x,\#\},\{q_0,q_1,q_2,q_3,q_4\}, \tau, q_0)$ is gegeven door het volgende diagram:



Hierin is  een verkorte schrijfwijze voor

Verifieer dat $f_{T'}(\overline{n}) = \overline{2n}$ voor elke $n \in N$, dus dat T' ook werkt als een vermenigvuldiger met 2.

Geef de berekening van 2×3 door T'.

Breng het verschil tussen de werking van T en van T' onder woorden.

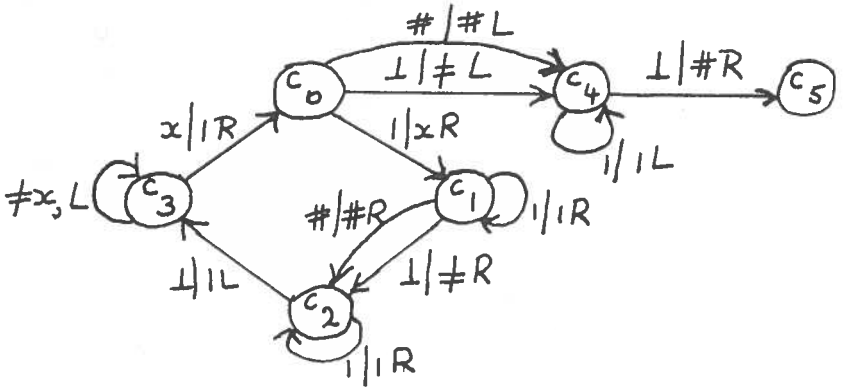
Vergelijk het aantal toestanden en het aantal vijftallen van T en T'.

Construeer zelf een Tm die werkt als een vermenigvuldiger met 3.

3.4. Opmerking. In het vervolg zullen we, ter vereenvoudiging, in het transitiediagram van een Tm met alfabet $\{a_1,a_2,\ldots a_n\}$ vaak een pijl met bijschrift $\neq a_i,d$ geven, waarbij d = L of d = R, in plaats van n - 1 pijlen met de bijschriften:

$a_1|a_1d$, $a_2|a_2d$, ..., $a_{i-1}|a_{i-1}d$, $a_{i+1}|a_{i+1}d$, ..., $a_n|a_nd$.

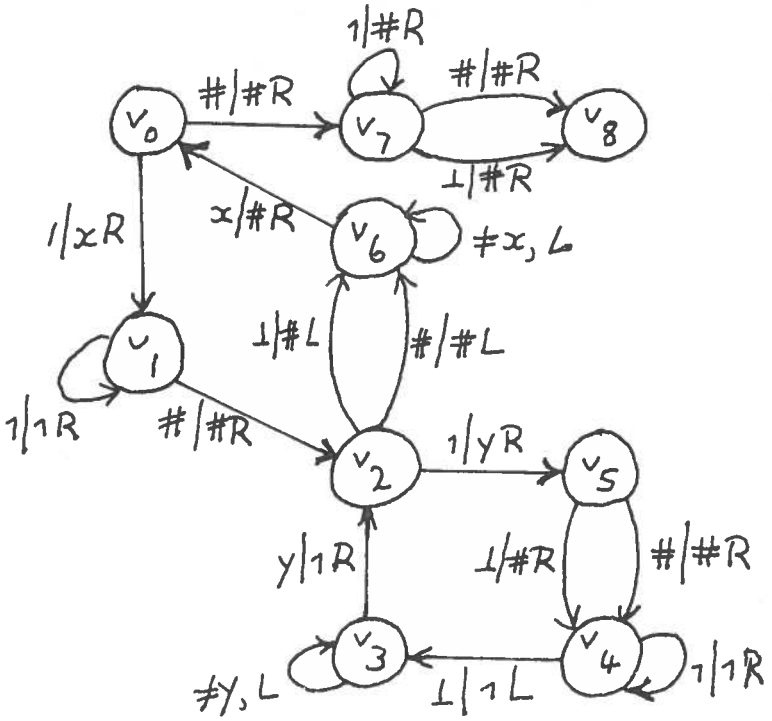
3.5. Voorbeeld. Tm C, gegeven door het volgende diagram, copieert voor elke n ∈ N op zijn band de expressie $\overline{n}$ zodat er een verdubbeling van de beginbandexpressie ontstaat.



Verificer dat beginconfiguratie $c_0\perp$ overgaat in $\#c_5\#$ en dat beginconfiguratie c_0111 overgaat in $\#c_5111\ \#\ 111$ en dat $c_0\overline{n}$ overgaat in $\#c_5\overline{n}\ \#\ \overline{n}$ voor elke $n \in N$, zodat $f_C(\overline{n}) = \#\overline{n}\,\#\overline{n}$.

Geef C in de vorm van vijftallen, vergelijk het aantal ervan met het aantal pijlen in het transitiediagram en verklaar het verschil.

3.6. Voorbeeld. Tm V, gegeven door onderstaand diagram, werkt als vermenigvuldiger, d.w.z. $f_V(\overline{n}\#\overline{m}) = \overline{n \times m}$.

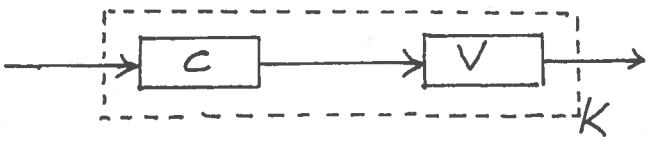


Verifieer dat de berekening van V met beginconfiguratie $v_o 11 \# 111$ als eindconfiguratie heeft $\# \ldots \# v_8 111111$ d.w.z. dat $f_V(\overline{2} \# \overline{3}) = \overline{6}$.
Verifieer dat voor elke n, m $\in$ N geldt $f_V(\overline{n} \# \overline{m}) = \overline{n \times m}$. Geef V in de vorm van vijftallen.

3.7. Opmerking De rekenkundige funktie f_V van voorbeeld (3.6) is een funktie van twee veranderlijken. Als codering van twee of meer veranderlijken wordt hier, zoals gewoonlijk, genomen de normale codering van elke veranderlijke met één speciale scheidingsletter ertussen, waarvoor hier # is genomen. Een rij van k natuurlijke getallen $(n_1, n_2, \ldots, n_k)$ wordt zodoende gecodeerd als: $\overline{n_1} \# \overline{n_2} \# \ldots \# \overline{n_k}$.

3.8, Het is nu eenvoudig om een Tm K te construeren die een gegeven getal kwadrateert, dus zo dat $f_K(\overline{n}) = \overline{n^2}$. Hiertoe laten we eerst Tm C werken op $\overline{n}$, dus met beginconfiguratie $c_o \overline{n}$ zodat configuratie $\# c_5 \overline{n} \# \overline{n}$ ontstaat. Daarna laten we V op dit resultaat werken door v_o te identificeren met c_5, zodat $\# \ldots \# v_8 \overline{n \times n}$ ontstaat.
Geef zelf Tm K in de vorm van een transitiediagram en in de vorm van vijftallen.
Geef de berekening van 3^2 door Tm K.

3.9. Schematisch kunnen we de constructie van Tm K uit de Tms C en V weergeven als volgt:



In het vervolg zullen we vaker zo'n constructie geven van een Tm T uit twee of meer Tms $T_1, T_2, \ldots, T_n$ die na elkaar met dezelfde band werken maar steeds met als beginbandexpressie de eindbandexpressie die de voorgaande Tm heeft achtergelaten. Hiervoor is het nodig dat elke Tm T_i een vaste eindtoestand, zeg q_h, heeft die geidentificeerd

kan worden met de begintoestand van Tm T_{i+1}. Als we deze afspraak gebruiken zullen we de Tm geven als een vijftal (A,Q,τ,q_o,q_h) waarbij $q_h \in Q$ de eindtoestand is. In bovenstaand voorbeeld (3.8) is de kwadrateermachine dus te geven als:
$K = \{1,n,y,\#\}, \{c_o,c_1,\ldots,c_5(=v_o),v_1,\ldots,v_8\},\tau_K,c_o,v_8)$ waarin
$\tau_K = \tau_C \cup \tau_V$ (en we elke τ zien als een verzameling van vijftallen).

3.10. Oefeningen 1. Construeer een Tm O die twee natuurlijke getallen optelt, dus $f_O(\overline{n}\#\overline{m}) = \overline{n+m}$.

2. Construeer een Tm T die elk natuurlijk getal verdubbelt door het eerst te copieren met Tm C en dan op te tellen met Tm O.

3. Construeer een Tm A die twee natuurlijke getallen aftrekt in de volgende zin:
$$f_A(\overline{n}\#\overline{m}) = \begin{cases} \overline{n-m} & \text{als } n \geq m \\ \overline{o} & \text{als } n < m. \end{cases}$$

4. Construeer een Tm D die twee natuurlijke getallen deelt en zowel het quotient als de rest geeft, dus:
$$f_D(\overline{n}\#\overline{t}) = [\overline{\tfrac{t}{n}}]\#\overline{t-n[\tfrac{t}{n}]}.$$

5. Construeer een Tm K die elk natuurlijk getal omzet in een vast gegeven getal, zeg 5, dus $f_K(\overline{n}) = \overline{5}$.

6. Construeer een Tm P die elke rij van meer dan twee natuurlijke getallen projecteert op zijn derde coördinaat, dus
$f_P(\overline{n_1}\#\overline{n_2}\#\overline{n_3}\#\ldots\#\overline{n_k}) = \overline{n_3}$.

4. Een Universele Turingmachine

4.1. Een Tm U noemt men universeel als de funktie f_T van elke, willekeurige, gegeven Tm T door U kan worden gesimuleerd. Hiermee wordt bedoeld: bij gegeven T kan een bandexpressie ρ_T voor U worden geconstrueerd zó dat als we nu een willekeurige bandexpressie b voor T kiezen en deze in gecodeerde vorm ρ_b naast ρ_T op de band van U schrijven, dan wordt ρ_b door U bewerkt zoals b door T bewerkt zou worden en hetzelfde resultaat wordt bereikt. We gaan hier aangeven hoe een universele Tm U te construeren is, om zodoende het bestaan ervan te demonstreren en te laten zien hoe de opbouw ervan kan zijn. De details van de constructie van U zijn gegeven in aanhangsel B.

4.2. Bij de constructie van U gaan we ervan uit dat elke Tm T die we willen simuleren aan de volgende voorwaarden voldoet:

1. T is een éénzijdige Tm.
2. T heeft alfabet $A = \{a_1,\ldots,a_{n-1}\}$ en T heeft a_o als blanco symbool. Het aantal verschillende symbolen dat op de band van T kan voorkomen is dus $|A|+1 = n$. Voor het blanco symbool van T nemen we a_o en niet $\perp$ omdat we dit graag als blanco symbool voor U gebruiken.
3. T heeft toestandsverzameling $Q = \{q_h,q_o,q_1,\ldots,q_{m-2}\}$ met q_h als eindtoestand en q_o als begintoestand, $|Q| = m$.

Volgens (2.16) vormt voorwaarde 1. geen beperking van de door U te simuleren Tms. De voorwaarden 2 en 3 geven ook geen beperking omdat voor elk alfabet A' van n' letters een bijectie op A met $|A| = n'$ is te geven en een zelfde opmerking voor de toestandsverzameling geldt.

4.3. Volgens opmerking (2.4) kan Tm T worden beschreven als een verzameling van t vijftallen, $\{v_1,v_2,\ldots,v_t\}$ waarbij elk vijftal van de vorm $v = q_i a_j q_k a_\ell d$ is met $q_i \in Q-\{q_h\}$, $q_k \in Q$, $a_j \in A$ en $a_\ell \in A \cup \{a_o\}$ en $d \in \{L,R\}$ terwijl geen twee vijftallen met dezelfde q en a beginnen.

De bandexpressie f_T voor U wordt als volgt uit T verkregen, anders gezegd: T wordt gecodeerd als:

$\rho_T = M_4 M_1 \rho_{v_1} M_1 \rho_{v_2} M_1 \ldots M_1 \rho_{v_t} M_2$.

Hierin zijn M_4, M_1 en M_2 merktekens die het begin van ρ_T, het begin van de gecodeerde vijftallen ρ_v en het eind van ρ_T markeren. Verder is de codering ρ_v van elk vijftal $v = q_i a_j q_k a_\ell d$, als volgt: $\rho_v = \rho_{q_i} \rho_{a_j} \rho_{q_k} \rho_{a_\ell} d$.

Nader uitgeschreven is dit:

$$\rho_v = \underbrace{B..B}_{m-i}\underbrace{Q..Q}_{i+1}\underbrace{B..B}_{n-j}\underbrace{A..A}_{j+1}\underbrace{B..B}_{m-k}\underbrace{Q..Q}_{k+1}\underbrace{B..B}_{n-\ell}\underbrace{A..A}_{\ell+1}d$$

Hierbij zijn voor de codering van elke $q \in Q$ steeds $m + 1 = |Q| + 1$ letters gebruikt en wel voor q_i $(\neq q_h)$ $m - i$ maal de letter B en $i + 1$ maal de letter Q met als uitzondering $\rho_{q_h} = \underbrace{B..B}_{m}H$

Elk $a_j \in A$ en ook a_o is gecodeerd met $n + 1 = |A| + 2$ letters, en wel $n - j$ maal de letter B en $j + 1$ maal de letter A en $\rho_{a_o} = \underbrace{B...B}_{n}A$

Voor de codering van elk vijftal van T zijn zodoende $2n + 2m + 5$ hokjes op de band van U nodig: $|\rho_v| = 2(n+m) + 5$. De codering van alle t vijftallen van T beslaat
$t(2\ (n+m) + 5) + t + 2 = 2t(n+m+3) + 2$ hokjes, dus $|\rho_T| = 2t(n+m+3) + 2$.

Omdat T voldoet aan de voorwaarden 2. en 3. van (4.2) is het duidelijk dat het coderen van T uniform is, d.w.z. ρ_T wordt voor elke T op dezelfde manier geconstrueerd uit T. (Hierbij nemen we voorlopig aan dat de volgorde waarin de vijftallen van T gecodeerd zijn geen effect heeft. We zullen nog zien dat dit inderdaad waar is).

4.4. Een bandexpressie $v = a_{b_1} a_{b_2} \ldots a_{b_k}$ van T coderen we als:

$$\rho_b = M_3 \rho_{a_{b1}} M_3 \rho_{a_{b2}} M_3 \ldots M_3 \rho_{a_{bk}} M_4.$$

Hierbij zijn M_3 en M_4 merktekens die het begin van de gecodeerde letters ρ_a en het eind van ρ_b markeren. Verder is de codering ρ_{a_i} van het lege symbool a_o en van elke letter $a_i \in A$ net als in (4.3), dus

$$\rho_{ai} = \underbrace{B..B}_{n-i}\underbrace{A..A}_{i+1}.$$

De ruimte die ρ_b inneemt op de band van U is:

als $|b| = k$ dan is $|\rho_b| = k(n+2) + 1$.

Bij gegeven T is het coderen van de bandexpressie ρ_b uit b uniform. Ga zelf na waardoor de bandexpressie ρ_b afhankelijk is van T.

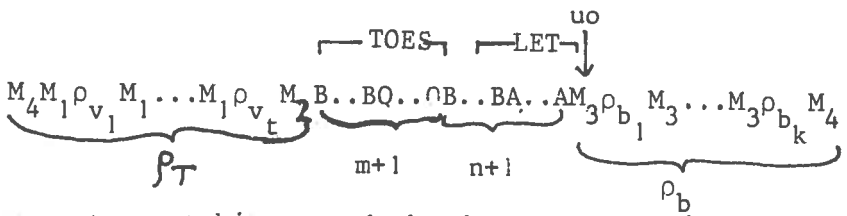
Opgave Geef ρ_T en ρ_b voor de Tm T van voorbeeld (2.8) en de bandexpressie 101 van T volgens bovenstaande coderingen.

4.5. De coderingen ρ_T en ρ_b voor T en b zoals hierboven gegeven zijn niet essentieel voor de constructie van een universele Tm. Ook andere coderingen zouden gekozen kunnen worden en de constructie van de universele Tm daarbij worden aangepast.
Voor de door ons gekozen coderingen geven we nu de samenstelling van een universele Tm U.

$U = (A_U, Q_U, \tau_U, u_o, u_h)$ met

$A_U = \{A,Q,H,B,L,R,M_1,M_2,M_3,M_4,M,X,Y,Z\}$ dus $|A_U| = 14$.

De toestandsverzameling Q_U en de transitiefunctie τ_U zullen we beetje bij beetje geven naarmate we vorderen met de constructie van U.
Voor het simuleren van Tm T die gaat werken op de bandexpressie b schrijven we, zoals hieronder is aangegeven, op de band van U eerst ρ_T en ρ_b met ertussen m + n + 2 hokjes waarvan de m + 1 eerste steeds de actuele toestand van T en de n + 1 volgende steeds de actuele letter van b representeren:



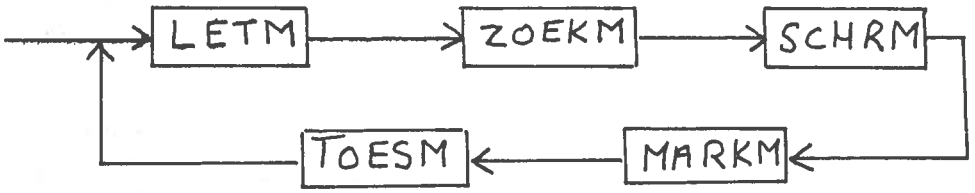
Deze twee stukjes van de band van U doen dienst als registers en we zullen ze TOES (van: toestandregister) respectievelijk LET (van: letterregister) noemen om er gemakkelijk naar te kunnen verwijzen. Als beginsituatie staat in TOES de rij ρ_{q_o} = B...BQ en in LET staat B...B, d.w.z. n + 1 maal B.
De begintoestand van U is u_o en de beginpositie van de ls-kop van U is de linkermarkering M_3 van ρ_b zoals hierboven is aangegeven.
De verschillende akties van U zijn achtereenvolgens:

1. Copieer de te bewerken ρ_a uit ρ_b in register LET. Dit wordt gedaan door Tm LETM (van: lettermachine) waarover in aanhangsel B, §1 alle details zijn gegeven.
2. Zoek de ρ_{v_1} in ρ_T op die begint met de inhoud van de register TOES en LET, stel v_i = qaq'a'd. Dit wordt gedaan door Tm ZOEKM

(van: zoekmachine), zie aanhangsel B, §2 voor de details.

3. Schrijf $\rho_{a'}$ in plaats van de in bewerking zijnde ρ_a in ρ_b. Tm SCHRM (van: schrijfmachine), zie aangangsel B, §3 doet dit.
4. Markeer als d = R de rechts, en als d = L de links, naast $\rho_{a'}$ gelegen ρ_{b_j} van ρ_b met het merkteken M als de volgende die bewerkt moet worden. Dit doet Tm MARKM (van: markeringsmachine), zie aanhangsel B, §4.
5. Copieer $\rho_{q'}$ in register TOES en ga naar de met M gemerkte ρ_{b_j} in ρ_b; als $\rho_{q'}$ = B BH, dus als $q' = q_h$, verander dan de markering M in H en stop maar ga anders verder met punt 1. Dit wordt gedaan door Tm TOESM (van: toestandmachine), zie aanhangsel B, §5.

Elk van deze vijf acties is in aanhangsel B gepreciseerd door de bijbehorende Tms LETM, ZOEKM, SCHRM, MARKM, en TOESM in detail te geven. De universele Tm U is de samengestelde machine die schematisch is weer te geven als:



4.6. Uit de gegeven constructie van de universele Tm U is het (wellicht) duidelijk dat de werking van Tm T op bandexpressie b precies wordt gesimuleerd door U, d.w.z. elke stap in de berekening van $\rho_T(b)$ uit b door Tm T wordt nagebootst door een aantal stappen van U werkend op ρ_b onder "besturing" van ρ_T. Zo wordt de funktie die T berekent gesimuleerd door U voorzien van ρ_T.
Preciezer uitgedrukt: de expressie ρ_d, zeg
$\rho_d = M_3\rho_{d_1}M_3\ldots M_3\rho_{d_i}H\rho_{d_{i+1}}M_3\ldots M_3\rho_{d_r}M_4$,
die rechts van LET op de band van U staat nadat U is gestopt met de ls-kop op de letter H komt overeen met $\rho_T(b)$ in gecodeerde vorm, dus met $\rho_{f_T(b)}$ (Het enige (onbelangrijke) verschil tussen beide is dat er een letter M_3 staat in $\rho_{f_T(b)}$ waar de letter H staat in ρ_d). Het is belangrijk goed te zien dat U onafhankelijk is van T en van b. Voor het simuleren van de werking van een andere Tm T_1 met beginbandexpressie b_1 is het alleen nodig ρ_{T_1}, de begininhoud van $TOES_1$ en van LET_1 en ρ_{b_1} op de band van Tm U te zetten en Tm U hierop te laten werken. In termen van een digitale computer

kunnen we zeggen: Tm U met programma ρ_T werkt op de gegeven ρ_b en Tm U verwerkt ρ_b volgens de algoritme vervat in ρ_T.

4.7. De constructie van de universele Tm U zoals hierboven is gegeven is een variant van de constructie die in Yasuhara (1971) wordt beschreven. Voor andere constructies van universele Tms zie o.a. Shannon (1956), Watanabe (1961) en Minsky (1967).

5. Beslisbare en opsombare verzamelingen.

5.1. Met behulp van T-berekenbare funkties voeren we nu twee belangrijke begrippen in, nl. beslisbare en opsombare verzamelingen. De behandeling is hier erg summier omdat de nog aan de orde komende recursieve funkties beter gereedschap vormen dan Tms om de begrippen beslisbaarheid en opsombaarheid uit te werken.

Definitie Zij A een alfabet en D een deelverzameling van A^*. D is een beslisbare verzameling als de karakteristieke funktie van D een T-berekenbare totale funktie is, d.w.z. als en alleen als er een Turingmachine T is zò dat voor alle $w \in A^*$ geldt:
$f_T(w) = 0$ als $w \in D$ en
$f_T(w) = 1$ als $w \notin D$. (Dus f_T is totaal).

Een voorbeeld van een beslisbare verzameling $D \subseteq A^*$ met $A = \{a,b\}$ is de verzameling D van alle woorden waarin de letters a links staan van de letters b die erin voorkomen. Een Tm T die de karakteristieke funktie van D berekent is gegeven in oefening 5 van (2.10). Verifieer dat f_T totaal is.

Oefeningen

1. Gegeven zijn $A = \{0,1\}$ en $D = \{w \mid w \in A^*$ en het aantal 1-en dat in w voorkomt is even$\}$.
Bewijs dat D beslisbaar is; hierbij kan men een variant van Tm T van oefening 4 in (2.10) gebruiken.
Bewijs dat ook $\overline{D} = A^* - D$ beslisbaar is.
2. Bewijs dat de verzameling van alle gebalanceerde haakjesvormen, deelverzameling van $\{(,)\}^*$, een beslisbare verzameling is m.b.v. Tm T van oefening 6 in (2.10).
3. Bewijs dat, bij gegeven A, zowel $\emptyset$ als A^* beslisbaar zijn.

5.2. Bij het definiëren van opsombare verzamelingen maken we gebruik van Tms met gecodeerde natuurlijke getallen, dus met rijtjes 1-en, als beginbandexpressie en woorden over een (eventueel) ander alfabet als eindbandexpressie.

Definitie Zij A een alfabet en D een deelverzameling van A^*.
D is een opsombare verzameling als $D = \emptyset$ of D is de waardenverzameling van een T-berekenbare totale funktie met de natuurlijke getallen als domein, d.w.z. als en alleen als $D = \emptyset$ of er is een Tm T met alfabet $B \supseteq \{1\}$ en een decodering $\delta: B^* \to A^*$ zó dat f_T totaal is en $D = \{\delta(f_T(\overline{n})) \mid n \in N\}$.

Voorbeeld Zij $A = \{a,b\}$ en $D = \{a^n b^n \mid n \geq 1\} \subseteq A^*$. D is een opsombare verzameling. Bekijk de Tm $T = (\{a,b,1\},\{q_o,q_1,q_2\},\tau,q_o\}$ met τ gegeven door:

$q_o \perp q_1 bL$	$q_1 a q_1 aL$	$q_2 a q_2 aR$
$q_o 1 q_1 bL$	$q_1 b q_1 bL$	$q_2 b q_2 bR$
	$q_1 \perp q_2 aR$	$q_2 1 q_1 bL$

Tm T somt D op, immers:

$f_T(\overline{o}) = ab$, $f_T(\overline{1}) = a^2b^2$, $f_T(\overline{2}) = a^3b^3$ en $f_T(\overline{n}) = a^{n+1}b^{n+1}$

Verifieer dit. N.B. Hierbij is δ dus de identieke afbeelding.
Bovenstaande voorbeeld is enigszins misleidend omdat de elementen van D keurig de een na de ander door T worden opgesomd. In het algemeen zal zo'n opsomming met herhalingen en in grillige volgorde geschieden.

Oefeningen

1. Bewijs dat $D = \{a^n b^n \mid n \geq 1\} \subseteq \{a,b\}^*$ ook een beslisbare verzameling is door een Tm te ontwerpen die de karakteristieke funktie van D berekent.
2. Zij $A = \{a,b,c\}$. Gegeven is de Tm $T = (B,\{q_o,q_1,q_2\},\tau,q_o)$ met $B = \{a,b,c,1\}$ en met τ gegeven als:

$q_o 1 q_1 aR$	$q_1 1 q_2 bR$	$q_2 1 q_2 cL$
$q_o \perp q_1 aR$	$q_1 \perp q_2 bR$	$q_2 \perp q_2 cL$

Zij verder de decodering $\delta : B^* \to A^*$ gegeven als

$\delta(a) = a$ $\delta(b) = b$ $\delta(c) = c$ $\delta(1) = \varepsilon$

en voor elke $x, y \in A^*$ is $\delta(xy) = \delta(x)\delta(y)$.
Verifieer dat $\delta(f_T(\overline{n})) = abc$ voor elke $n \in N$ dus dat $\{abc\} \subseteq A^*$ opsombaar is

3. Waarom is de lege verzameling $\emptyset$ niet de waardeverzameling van een totale T-berekenbare funktie ? (zodat $D = \emptyset$ inderdaad apart in de definitie van opsombare verzamelingen moet worden opgenomen).

4. Zij A = {a,b,c}. Toon aan dat $\{\varepsilon\} \subseteq A^*$ een opsombare verzameling is door een geschikte Turingmachine en decodering te ontwerpen.

5.3. Opmerking 1 De begrippen opsombaar en aftelbaar moeten goed onderscheiden worden. Een verzameling V is aftelbaar als er een bijectie bestaat van V naar de verzameling N van alle natuurlijke getallen. Uiteraard is elke opsombare verzameling $D \subseteq A^*$ hoogstens aftelbaar, d.w.z. eindig of aftelbaar, immers A^* is aftelbaar, maar niet elke aftelbare verzameling is opsombaar zoals we nog zullen zien. Het begrip aftelbaar slaat op de kardinaliteit van een verzameling en de opsombaarheid op de mogelijkheid van het effectief genereren van de elementen van de verzameling.

Opmerking 2 Bij de begrippen beslisbaar en opsombaar hebben we eenvoudshalve alleen gekeken naar deelverzamelingen van de vrije monoide A^* gegenereerd door een alfabet A. Uiteraard is dit een beetje beperkt, we kunnen ook bij deelverzamelingen van een willekeurige gegeven verzameling V over beslisbaar en opsombaar spreken. De definities (5.1) en (5.2) hoeven dan niet essentieel te veranderen: bij beslisbaarheid moeten we een codering van V naar A^* en bij opsombaarheid een decodering van A^* naar V toevoegen. We zullen dit hier niet verder hanteren.

5.4. Het verband tussen beslisbare en opsombare verzamelingen is aangegeven in de volgende:

Eigenschappen: Zij A een alfabet en $D \subseteq A^*$

a) D is beslisbaar impliceert $\overline{D}(= A^* - D)$ is beslisbaar.

b) D is beslisbaar impliceert D is opsombaar.

c) D en $\overline{D}$ zijn opsombaar impliceert D is beslisbaar.

d) Er zijn verzamelingen die niet beslisbaar maar wel opsombaar zijn.

e) Er zijn verzamelingen die niet opsombaar zijn.

Verifieer zelf dat uit a) en b) volgt:
D is beslisbaar impliceert D en $\overline{D}$ zijn opsombaar.
Samen met c) volgt hieruit:
D is beslisbaar als en alleen als D en $\overline{D}$ opsombaar zijn.

We bewijzen bovenstaande eigenschappen hier niet in detail omdat dat te ver voert, we komen er nog op terug nadat de recursieve funkties zijn behandeld. Wel geven we hier aan hoe de bewijzen voor de eigenschappen b) en c) verlopen.

ad b) Als $D = \emptyset$ dan is D per definitie opsombaar. Veronderstel $D \neq \emptyset$ en dat bijvoorbeeld $v \in D$. De elementen van A^* kunnen worden opgesomd, bijvoorbeeld in lexicografische volgorde. Neem aan dat Tm 0 dit doet. Omdat 1) beslisbaar is is er een Tm B_D die de karakteristieke funktie van D berekent. Combineer 0 en B_D in een procedure die als volgt verloopt: 0 somt de woorden van A^* een voor een op. Voor elk opgesomd woord w beslist B_D of w wel of niet tot D behoort. Als $w \in D$, dan wordt w als resultaat geleverd en als $w \notin D$ dan wordt als resultaat het woord v geleverd. De beschreven combinatie somt de woorden van D op, dus D is opsombaar.

ad c) Neem aan dat de Tms 0_D en $0_{\overline{D}}$ de verzamelingen D en $\overline{D} = A^* - D$ opsommen. Combineer 0_D en $0_{\overline{D}}$ tot een procedure die als volgt verloopt. Stel dat een woord $w \in A^*$ is gegeven zodat de beslissingsvraag is: is $w \in D$ of is $w \notin D$? Laat nu 0_D en $0_{\overline{D}}$ om de beurt gaan werken en iedere keer als ze een woord w_1 hebben opgesomd wordt dit vergeleken met w. Uiteraard wordt w ofwel door 0_D ofwel door $0_{\overline{D}}$ opgesomd. Zodra $w_1 = w$ wordt de vraag als volgt beantwoord: als w_1 door 0_D is opgesomd dan is $w \in D$ en als w_1 door $0_{\overline{D}}$ is opgesomd dan is $w \notin D$. De beschreven combinatie bepaalt voor elke $w \in A^*$ of w wel of niet element van D is, dus D is beslisbaar.

In beide redeneringen is aangegeven hoe een bepaalde gewenste procedure kan verlopen. De details van de constructie van een Turingmachine die de gewenste procedure realiseert laten we over aan de ijverige lezer.

6. Oplosbare en onoplosbare problemen.

6.1. Met "probleem" bedoelen we hier een hele klasse van gelijksoortige opgaven. Om dit duidelijk te maken bekijken we de volgende problemen.

a) Het alfabetisch sorteren van de letters van een willekeurig woord w over het alfabet {a,b}.

b) Het vermenigvuldigen met 2 van een willekeurig natuurlijk getal n.

c) Het beslissen voor een willekeurig polynoom P met gehele coefficienten, of er gehele getallen zijn die een nulpunt van P vormen (Dit is het 10^e probleem op de lijst van Hilbert (1901).

Dergelijke problemen bevatten een of meer veranderlijken, zoals w, n en P in bovenstaande voorbeelden. Geven we deze een bepaalde waarde dan krijgen we een concreet geval van het probleem, een opgave, waarvoor een antwoord bestaat. Een oplossing van het probleem is een effectieve procedure die voor ieder concreet geval het antwoord oplevert. Ons beroepend op de these van Turing kunnen we zeggen: een oplossing van het probleem is een Turingmachine die voor ieder concreet geval, d.w.z. als de veranderlijken in geschikte codering als beginbandexpressie zijn gegeven, het antwoord (in gecodeerde vorm) berekent en dus stopt. Als er een oplossing bestaat noemt men het probleem oplosbaar anders onoplosbaar.

Bij een onoplosbaar probleem kan er wel voor één of voor enkele concrete gevallen een algoritme zijn dat het antwoord oplevert, maar er is geen algoritme dat dit voor willekeurig te specificeren concreet geval doet.

De problemen a) en b) zijn oplosbaar: de oplossing is gegeven in voorbeeld (2.3) en voorbeeld (3.2). Van probleem c) is in 1970 bewezen dat het onoplosbaar is, zie Matijasevič (1970). Uiteraard zijn deelproblemen van probleem c) wel oplosbaar. Denk, bijvoorbeeld, aan probleem c) beperkt tot kwadratische polynomen in één veranderlijke, dus polynomen van de vorm: ax^2+bx+c.

Veel problemen waarvan de al of niet oplosbaarheid een interessante vraag is, zijn beslissingsproblemen, d.w.z. problemen waarbij het antwoord voor concrete gevallen slechts "ja" of "nee" kan zijn. We zullen ons hier verder alleen met beslissingsproblemen bezig houden.

6.2. Een standaardvoorbeeld van een onoplosbaar beslissingsprobleem is het stopprobleem: Gegeven een willekeurige Tm T in een willekeurige beginconfiguratie C van T, zal T ooit stoppen ?

Om in te zien dat dit probleem onoplosbaar is bekijken we de volgende beperkte versie van het stopprobleem:

> Zal een willekeurige Tm T stoppen als zij haar eigen beschrijving,in gecodeerde vorm, als beginbandexpressie krijgt ?

Als dit beperkte probleem niet oplosbaar is dan is zeker het algemene stopprobleem niet oplosbaar.
Veronderstel dat het beperkte stopprobleem wel oplosbaar is en dat Tm H de oplossing berekent.
Verander nu H in H' zodat:
1) Als H stopt met als antwoord dat T stopt, dan komt H' in een lus zo dat H' niet meer stopt. (Dus H' stopt niet als T met haar eigen beschrijving stopt).
2) Als H stopt met als antwoord dat T niet stopt, dan stopt H'. (Dus H' stopt als T met haar eigen beschrijving niet stopt).
Als nu H' haar eigen beschrijving als beginbandexpressie krijgt ontstaat een contradictie, immers H' stopt als en alleen als H' niet stopt. (Ga dit goed na!).
De conslusie uit deze tegenspraak is dat de veronderstelling van het bestaan van H niet juist is, dus het beperkte stopprobleem niet oplosbaar is. Maar dan is het algemene stopprobleem ook niet oplosbaar.

6.3. Behalve het stopprobleem is een veel gebruikt standaard onoplosbaar beslissingsprobleem het correspondentieprobleem van Post (zie Post(1946)).

> Beslis voor twee willekeurige n-tallen $(w_1,w_2,\ldots,w_n)$ en $(v_1,v_2,\ldots,v_n)$ van woorden over een alfabet A (met minstens twee letters) of er een rij getallen $(i_1,i_2,\ldots,i_k)$ is met $1 \leq i_j \leq n$ zodat $w_{i_1} w_{i_2} \ldots w_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$.

Voorbeelden: A = {a,b}

Bij de tweetallen (bbb, abb) en (bb, babbb) is er zo'n rij, nl. (1,2,1) want bbb.abb.bbb = bb.babbb.bb. Bij de drietallen (a, bba, aab) en (ba, aaa, ba) is er geen rij, want de overeenkomstige woorden beginnen alle met verschillende letters.
Zonder bewijs delen we mee: er is géén algoritme dat voor ieder geval een beslissing, zoals bij deze voorbeelden, geeft.

6.4. De meeste bewijzen van de onoplosbaarheid van problemen bestaan uit de constructie van een oplossing voor een standaard onoplosbaar probleem uit een hypothetische oplossing van het beschouwde probleem. Daarmee is dan een tegenspraak verdregen zodat er kennelijk geen oplossing bestaat voor het beschouwde probleem.

6.5. We komen nog even terug op het stopprobleem van (6.2) en bekijken de volgende verzameling paren:
$V = \{(T,C) \mid Tm\ T$ met beginconfiguratie C stopt$\}$.
Het stopprobleem:
"Zal willekeurige T met willekeurige beginconfiguratie C ooit stoppen"
kan nu ook geformuleerd worden als de beslissing:
"voor willekeurige (T,C) is $(T,C) \in V$?"
De (on)oplosbaarheid van het stopprobleem komt precies overeen met de (on)beslisbaarheid van de verzameling V. Daarom worden de begrippen (on)oplosbaar beslissingsprobleem en (on)beslisbare verzameling vaak als synoniem gebruikt.

7. Primitief recursieve funkties.

7.1. In de komende hoofdstukken geven we een andere formalisering van het intuitieve begrip "berekenbaar". We beperken ons hierbij tot rekenkundige funkties, d.w.z. de argumenten en de waarden zijn natuurlijke getallen (inclusief 0), maar al het volgende kan veralgemeend worden tot funkties van A^* (of, indien de funkties meerplaatsig zijn, $A^*x...xA^*$) naar A^*, voor willekeurig alfabet A. We definieren de begrippen primitief recursieve funktie, recursieve funktie en partieel recursieve funktie. De these is dat de intuitief berekenbare funkties juist de partieel recursieve zijn. Als argument ter bevestiging hiervan zal aangetoond worden dat de Turing-berekenbare rekenkundige funkties precies de partieel recursieve funkties zijn.

7.2. Afspraak Waar geen misverstand mogelijk is, laten we uit de notatie volgen hoeveel-plaatsig de funkties zijn. Met $\vec{x}$ en $\vec{y}$ etc duiden we een rijtje $x_1,...,x_n$ en $y_1,...,y_m$ aan voor geschikte waarden van n en m. Dus uit de schrijfwijze $h(f_1(\vec{x}),...,f_k(\vec{x}))$ volgt dat de funktie h k-plaatsig is en dat ieder van de funkties f_i n-plaatsig is voor zekere $n \geq 1$; elke funktie is minstens éénplaatsig. Zo nu en dan zullen we een slordig taalgebruik tolereren, door te zeggen

de funktie $f(\vec{x}) = ...$

waar we bedoelen

de funktie f, gedefinieerd door $f(\vec{x}) = ...,....$

7.3. Net als bij Turing machines willen we een klasse van funkties definieren op een zodanige manier dat eenieder met hun intuitieve berekenbaarheid zal instemmen. Wij kiezen hier voor de zogenaamde inductieve definitie. Dat wil zeggen: van enkele met name genoemde funkties wordt gesteld dat ze tot de klasse behoren (we noemen deze funkties de basisfunkties), en we stellen dat verder iedere funktie van de klasse verkregen kan worden uit andere funkties van de klasse (de basisfunkties, bijv.) middels zekere, met name genoemde samen-stellings schema's.

Als we overtuigd zijn van de intuitieve berekenbaarheid van de basisfunkties en van het behoud van berekenbaarheid onder de samenstellingschemas, dan is duidelijk dat alle funkties van de klasse intuitief berekenbaar zijn. Of ook alle intuitief berekenbare funkties tot de klasse behoren is dan nog maar de vraag. Daartoe zullen we de basisfunkties en - vooral - de samenstellingswijzen zo "ruim" mogelijk moeten kiezen.
In de basisfunkties kunnen we bijvoorbeeld de optelling, vermenigvuldiging, machtsverheffing etc. opnemen. Maar bij geschikt gekozen samenstellingswijzen is de vermenigvuldiging wel als herhaalde optelling de definieren. En machtsverheffing als herhaalde vermenigvuldiging. Dus vinden we het "mooier" om van de optelling, machtsverheffing en vermenigvuldiging alléén de optelling in de basisfunkties op te nemen. En zelfs de optelling is in feite een herhaalde "ophoging met één"...
Als samenstellingsschema's zijn er ook heel wat keuzen mogelijk. De twee schema's die in de volgende definitie staan, zijn in ieder geval voldoende gebleken om alle andere schema's uit dit hoofdstuk af te leiden. . .

7.4. Definitie De klasse *PR* van primitief recursieve funkties (prim rec funkties) wordt inductief gedefinieerd door

I: de volgende funkties zijn prim rec,

a. de nulfunktie $Z(x) = 0$ (Z van "Zero")

b. de opvolgerfunktie $S(x) = xh$ (S van "Successor")

c. de projectiefunkties $U_i^n(x_1 \ldots, x_n) = x_i$ (U van "Uitkiezer")

II: samenstelling van prim rec funkties d.m.v. de volgende schema's levert weer een prim rec funktie,

a. het schema van compositie, d.w.z.

als $g_1, \ldots, g_m$ en h prim rec zijn en f is gedefinieerd door $f(x)=h(g_1(\vec{x}), \ldots, g_m(\vec{x}))$, dan is ook f prim rec.

b. het schema van recursie, d.w.z.

als g en h prim rec zijn en f is gedefinieerd door

$f(0,\vec{x}) = g(\vec{x})$

$f(n+1,\vec{x}) = h(f(n,\vec{x}),n,\vec{x})$, dan is ook f prim rec.

(III:en geen andere funkties dan die welke te verdrijgen zijn door herhaalde toepassing van clausules I en II, zijn prim rec.).

7.5. Als gevolg van de eenvoud van de definitie kost het in het begin veel werk om aan te tonen dat een funktie prim rec is, maar de eenvoud van de definitie vergemakkelijkt wel het bewijzen van eigenschappen die gelden voor de gehele klasse der prim rec funkties.

Het is gebruikelijk om definities zoals (7.4) in andere bewoordingen te geven. Wanneer K een klasse van objecten is en $*$ een bewerking is op K die van elementen van K weer een element van K maakt, dan noemt men K gesloten onder $*$. In deze terminologie kunnen we dus stellen: De klasse PR wordt inductief gedefinieerd door

I: Z, S en de U_i^n $(i = 1,2,\ldots,n, n \geq 1)$ behoren tot PR

II: PR is gesloten onder compositie en recursie,

III: en geen andere funkties dan die welke te verkrijgen zijn door herhaalde toepassing van I en II behoren tot PR.

Wanneer een klasse inductief gedefinieerd wordt en dit er uitdrukkelijk bij vermeld staat, dan laat men een clausule zoals III ook wel weg. Een nóg andere formulering is de volgende. De klasse PR is de kleinste klasse van funkties zódat

I: Z,S en de U_i^n behoren tot PR,

II: PR is gesloten onder compositie en recursie.

(Ga na dat deze definitie gelijkwaardig is met de voorgaanden. Dus stel dat door deze laatste definitie een klasse PR' wordt gedefinieerd en bewijs $PR \subseteq PR'$ en $PR' \subseteq PR$, d.w.z. $PR = PR'$).

In de nu volgende stelling vindt U enige voorbeelden van prim rec funkties.

7.6. Stelling De volgende funkties zijn prim rec.

a. Optelling	$f_+(x,y) = x+y$
b. vermenigvuldiging	$f_\cdot(x,y) = x.y$
c. faculteit	$f_!(x) = x!$
d. signum	$sg(x) = \begin{cases} 0 & \text{als } x \neq 0^* \\ 1 & \text{anders} \end{cases}$
e. machtsverheffing	$exp(x,y) = x^y$
f. voorganger (predecessor)	$pred(x) = \begin{cases} x-1 & \text{als } x \geq 1 \\ 0 & \text{anders.} \end{cases}$
g. pariteit	$par(x) = \begin{cases} 0 & \text{als } x \text{ is even} \\ 1 & \text{anders.} \end{cases}$

*) omdat gebleken is dat dit handig werkt, gebruiken we de nul als bevestiging en de één als ontkenning.

h. monus ('n gewijzigde minus)	$\text{monus}(x,y)=x \dot{-} y = \begin{cases} x-y & \text{als } x>y \\ 0 & \text{anders} \end{cases}$
i. gelijkheid (equality)	$\text{eq.}(x,y)= \begin{cases} 0 & \text{als } x=y \\ 1 & \text{anders} \end{cases}$
j. absolute verschil	$\text{abs}(x,y) = \lvert x-y \rvert$
k. minimum	$\min(x,y) = \begin{cases} x & \text{als } x \leq y \\ y & \text{anders} \end{cases}$
l. maximum	$\max(x,y) = \begin{cases} x & \text{als } x \geq y \\ y & \text{anders.} \end{cases}$
m. de constante funkties	$c_k(x) = k$ (voor $k = 0,1,2,....$)

Bewijs

a. Ga na dat $f_+(0,x) = x$ en $f_+(n+1,x) = f_+(n,x)+1$.....(1)

We definieren nu f_+ volgens het schema van recursie:
$f_+(0,x) = g(x)$
$f_+(n+1,x) = h(f_+(n,x), n, x)$.
Voor g kiezen we U_1^1, want dan is $g(x) = U_1^1(x) = x$, conform (1), en bovendien is g dan prim rec. Voor h kiezen we de compositie van S met U_1^3, dus

$$h(x,y,z) = S(U_1^3(x,y,z)),$$

want dan is $h(f_+(n,x),u,x) = S(U_1^3(fin,x),n,x)) = S(f_+(n,x)) = f_+(n,x)+1$, conform (1) en bovendien is h dan prim rec (ga na!).
Dus nu is f_+ gedefinieerd met recursie op de prim rec funkties g en h en derhalve is f_+ prim rec.

b. Ga na dat $f_o(0,x) = 0$ en $f_o(n+1,x) = f_o(n,x) + x$ .....(1)
We definieren f_o nu volgens het schema van recursie:
$f_o(0,x) = g(x)$
$f_o(n+1,x) = h(f_o(n,x),n,x)$
Voor g kiezen we Z, want dan is $g(x) = Z(x) = 0$, conform (1), en bovendien is g prim rec.
Voor h kiezen we de compositie van f_+ met U_1^3 en U_3^3, d.w.z.
$h(x,y) = f_+(U_1^3(x,y,z), U_2^3(x,y,z))$,
want dan is $h(f_o(n,x)) = f_+(U_1^3(f_o(n,x)n,x,U_3^3(f_o(n,x),n,x)) =$
$f_+(f_o(n,x),x) = f_o(n,x) + x$, conform (1), en bovendien is h prim rec. (Ga na!!!).

Dus nu is f_o gedefinieerd met recursie op prim rec funkties g en h, en derhalve is f_o prim rec.

c. Doe zelf.

d. Ga na dat sg(0) = 1 en sg(n+1) = 0. . . . . (1)

We kunnen op grond van (1) niet zonder meer een definitie van sg opstellen volgens het schema van recursie (waarom niet ?).

De truc is het invoeren van een loze variabele.

Laat sg'(0,x) = 1 en sg'(n+1,x) = 0 . . . . . .(2)

dan kunnen we deze sg' met recursie op prim rec funkties definieren als volgt:

$$\begin{cases} sg'(0,x) = g(x) \\ sg'(n+1,x) = h(sg'(n,x),n,x) \end{cases}$$

met $g(x) = S(Z(x))$ en $h(x,y,z) = Z(U_1^3(x,y,z))$,

dus is sg' prim rec.

Nu kunnen we sg definieren door compositie van sg' met U_1^1:

$sg(x) = sg'(U_1^1(x), U_1^1(x))$. (ga na dat sg(0) = 0, sg(n+1) = 1),

dus is sg prim rec.

e. Merk op dat

$$x^o = \begin{cases} 0 \text{ als } x = 0 \\ 1 \text{ als } x > 0 \end{cases} \qquad \text{en } x^{n+1} = x^n \times x.$$

Welnu, U_1^3, U_3^3 en f_o zijn prim rec.

Dus ook h gedefinieerd door $h(x,y,z) = f_o(U_1^3(x,y,z),U_3^3(x,y,z))$,

is prim rec. Ook de funktie g gedefinieerd door g(x) = sg(sg(x))

is prim rec. Dus f gedefinieerd door

$$\begin{cases} f(0,x) = g(x) \\ f(n+1,x) = h(f(n,x),n,x) \end{cases}$$

is prim rec en deze funktie is juist exp.

f. Doe zelf.

m. c_o is prim rec want $c_o(x) = Z(x)$

c_1 is prim rec want $c_1(x)=S(c_o(x))$ en S,c_o zijn prim rec,

c_2 is prim rec want $c_2(x)=S(c_1(x))$ en S,c_1 zijn prim rec,

⋮

c_{k+1} is prim rec want $c_{k+1}(x)=S(c_k(x))$ en S,c_k zijn prim rec,

⋮

Dus alle c_k (k=0,1,2,...) zijn prim rec. X

Naarmate we van meer funkties bewezen hebben dat ze prim rec zijn, kunnen we voor een gegeven funktie sneller het bewijs leveren dat die prim rec is (als die het tenminste is).Dit zelfde kunnen we ook bereiken door meer schema's te geven waarvan we weten dat ze prim rec funkties in prim rec funkties overvoeren.

7.7. Stelling De klasse der prim rec funkties is gesloten onder de volgende schema's,

a. onderscheid naar gevallen, d.w.z.

als $g_1,\ldots,g_{m+1}$ en $h_1,\ldots,h_m$ prim rec zijn en voor alle $\vec{x}$ is er hoogstens één i zó dat $h_i(\vec{x}) = 0$ *) en f is gedefinieerd door

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{als } h_1(\vec{x}) = 0 \\ \vdots & \vdots \\ g_m(\vec{x}) & \text{als } h_m(\vec{x}) = 0 \\ g_{m+1}(\vec{x}) & \text{anders,} \end{cases}$$

dan is ook f een prim rec funktie.

b. selectie en permutatie van de argumenten, d.w.z.

als g prim rec is en f is gedefinieerd door

$$f(x_1,\ldots,x_n) = g(x_{i_1},\ldots,x_{i_m})$$

met $x_{i_1},\ldots,x_{i_m} \in \{x_1,\ldots,x_n\}$ (N.B. eventueel $m \neq n$),

dan is ook f prim rec.

c. eindig aantal voorgeschreven uitzonderingen, d.w.z.

als $n_1,\ldots,n_q$ getallen zijn en $g_1,\ldots,g_q,g$

prim rec funkties, en f is gedefinieerd door

---

*) Let wel, de stelling spreekt zich niet uit over de manier waarop aangetoond zou moeten worden dat er voor alle $\vec{x}$ hoogstens één i is met $h_i(\vec{x}) = 0$. Ook al zou u in een bijzonder geval deze voorwaarde met niet-prim rec middelen aantonen, dan geldt toch dat f prim rec is! Ga maar na in het bewijs van de stelling.

$$\begin{cases} f(n_1,\vec{x}) = g_1(\vec{x}) \\ f(n_2,\vec{x}) = g_2(\vec{x}) \\ \quad\vdots \\ f(n_q,\vec{x}) = g_q(\vec{x}) \\ f(n,x) = g(n,\vec{x}) \quad \text{voor } n \neq n_1,\ldots,n_q, \end{cases}$$

dan is f prim rec.

(N.B. we kunnen voor de g_i ook constante funkties kiezen).

Bewijs

a. Ga na dat f gedefinieerd kan worden door

$$f(\vec{x}) = g_1(\vec{x}).\ sg(h_1(\vec{x})) + \ldots + g_m(\vec{x}).\ sg(h_m(\vec{x})) + g_{m+1}(\vec{x}).\ sg(h_1(\vec{x}).\ \ldots\ h_m(\vec{x})).$$

(Omdat er voor willekeurige $\vec{x}$ hoogstens één i is met $h_i(\vec{x}) = 0$, zijn op één na alle termen nul).

Omdat in de som en in het produkt het aantal termen resp factoren een vast aantal m is - niet afhankelijk van een argument -, kan het rechterlid van de definitie gegeven worden als een herhaalde compositie van $g_1,\ldots,g_{m+1}$, $h_1,\ldots,h_m$ en f_+ en f_o.

Dus f is prim rec.

b. f kan gedefinieerd worden door

$$f(x_1,\ldots,x_n) = g(U^n_{i_1}(x_1\ldots x_n),\ U^n_{i_2}(x_1,\ldots,x_n),\ldots,U^n_{i_m}(x_1,\ldots,x_n)),$$

dus f is prim rec.

c. f kan gedefinieerd worden door

$$f(n,\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{als } eq(n,n_1) = 0 \\ \quad\vdots & \\ g_q(\vec{x}) & \text{als } eq(n,n_q) = 0 \\ g(n,x) & \text{anders} \end{cases}$$

en volgens het al behandelde schema van onderscheid naar gevallen, en selectie en permutatie van argumenten, is f prim rec. ∎

De vorm van de toegelaten voorwaarden in een definitie met onderscheid naar gevallen laat niet veel vrijheid toe. We zouden een voorwaarde graag als eigenschap of als relatie tussen de gegeven argumenten willen formuleren. Dat zullen we nu doen. (In aanhangsel C is nog het een en ander vermeld over kwantificaties over relaties).

7.8. Definitie a. Zij R een relatie van n argumenten. De karakteristieke funktie f_R van R is de funktie gedefinieerd door

$$f_R(x_1,\ldots,x_n) = \begin{cases} 0 & \text{als } R(x_1,\ldots,x_n) \\ 1 & \text{als } \neg R(x_1,\ldots,x_n) \end{cases}$$

b. Een relatie heet prim rec als zijn karakteristieke funktie prim rec is.

7.9. Stelling (onderscheid naar gevallen op prim rec relaties)
Als $g_1,\ldots,g_{m+1}$ prim rec funkties zijn en $R_1,\ldots,R_m$ prim rec relaties zó dat voor alle x er hoogstens één i is zodat $R_i(\vec{x})$ geldt, en f is gedefinieerd door

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{als } R_1(\vec{x}) \\ \vdots & \vdots \\ g_m(\vec{x}) & \text{als } R_m(\vec{x}) \\ g_{m+1}(\vec{x}) & \text{anders.} \end{cases}$$

dan is ook f prim rec.

Bewijs. De voorwaarden $R_i(\vec{x})$ kunnen vervangen worden door $f_{R_i}(\vec{x}) = 0$. Dan is f prim rec volgens (7.7)a. □

Om het onderscheid naar gevallen - geformuleerd als relaties - te kunnen gebruiken geven we nu enkele prim rec relaties en daarna enkele schema's om prim rec relaties te verkrijgen.

7.10. Stelling. De volgende relaties zijn prim rec.

a. $=$, gedefinieerd door $x = y$ $\leftrightarrow$ x is gelijk aan y
b. $\leq$, " " $x \leq y$ $\leftrightarrow$ x is niet groter dan y
c. $<$, " " $x < y$ $\leftrightarrow$ x is kleiner dan y
d. EXP, " " $EXP(x,y,z)$ $\leftrightarrow$ $x^y = z$.

bewijs

a. $f_=(x,y) = eg(x,y)$, zie (7.6), dus $f_=$ en daarmee $=$ is prim rec.

Doe zelf b. - d. □

7.11. Stelling Zij $g_1,\ldots,g_n$ prim rec funkties en $Q_1,\ldots,Q_{m+1}$ en $R_1,\ldots,R_m$ prim rec relaties zó dat er voor alle x hoogstens één $R_i(\vec{x})$ geldig is, dan zijn ook de volgende relaties prim rec,

a. negatie : de relatie $Q(\vec{x}) \leftrightarrow \neg Q_1(\vec{x})$
b. disjunctie : " " $Q(\vec{x}) \leftrightarrow Q_1(\vec{x}) \vee Q_2(\vec{x})$
c. conjunctie : " " $Q(\vec{x}) \leftrightarrow Q_1(\vec{x}) \wedge Q_2(\vec{x})$
d. implicatie : " " $Q(\vec{x}) \leftrightarrow Q_1(\vec{x}) \rightarrow Q_2(\vec{x})$
e. de relatie $Q(\vec{x}) \leftrightarrow Q_1(g_1(\vec{x}),\ldots,g_n(\vec{x}))$
f. de relatie $Q(\vec{x}) \leftrightarrow$
$$\begin{cases} Q_1(\vec{x}) & \text{als } R_1(\vec{x}) \\ \vdots & \\ Q_m(\vec{x}) & \text{als } R_m(\vec{x}) \\ Q_{m+1}(\vec{x}) & \text{anders.} \end{cases}$$

bewijs

a. $f_Q(\vec{x}) = 1 \dot{-} f_{Q_1}(\vec{x})$, dus f_Q is prim rec en daarmee ook Q
b. $f_Q(\vec{x}) = f_{Q_1}(\vec{x}) . f_{Q_2}(\vec{x})$
c. $f_Q(\vec{x}) = sg(sg(f_{Q_1}(\vec{x}) + f_{Q_2}(\vec{x})))$
d. wegens $Q_1 \rightarrow Q_2 \leftrightarrow_{def} \neg Q_1 \vee Q_2$
e.f. doe zelf. ☒

7.12. Oefening. Verzin zelf nog andere mogelijkheden om prim rec relaties te verkrijgen (vgl. stelling (7.7)).

Een belangrijk beginsel dat zeker intuitief berekenbaar is te noemen, is nog niet aan bod gekomen: het zoeken binnen een zekere begrenzing. Als voorbeeld daarvan noemen we de begrensde minimalisatie, begrensde existentiele kwantificatie en begrensde universele kwantificatie.
De begrensde minimalisatie levert het kleinste getal binnen de begrenzing, dat aan een gegeven relatie voldoet. De begrensde existentiele kwantificatie bevestigt het bestaan van een getal binnen de begrenzing, waarmee een gegeven relatie geldig is. En de begrensde universele kwantificatie bevestigt de geldigheid van een gegeven relatie voor alle getallen binnen de begrenzing.
We geven nu eerst de precieze definitie van de minimalisatie en dan de stellingen over de minimalisatie en kwantificatie.

7.13. Definitie (begrensde minimalisatie). Zij R een relatie en n een natuurlijk getal, dan betekent
$\mu\ x < n[R(\vec{x},y)]$:

$$\begin{cases} \text{de kleinste } x \text{ zó dat } R(x,\vec{y}). \ldots\ldots \text{ als die } x \text{ bestaat en } < n \text{ is} \\ 0 \qquad\qquad\qquad\qquad\qquad \ldots\ldots \text{ anders} \end{cases}$$

(N.B. μ komt van μlkpos = klein).

Voorbeeld

$\mu x<15\ [x>13] = 14$
$\mu x<100$ [in de decimale ontwikkeling van π komen er bij de x-de decimaal twee opeenvolgende tweeen] = 0
$\mu x<200$ [$\sim$ idem] = 135
$\mu x<200$ [$\sim$ idem én $x \neq 135$] = 185
$\mu x<200$ [$\sim$ idem én $x = 135$ én $x \neq 185$] = 0

7.14. Stelling De klasse der prim rec funkties is gesloten onder begrensde minimalisatie op prim rec relaties, d.w.z.
als R prim rec is en f is gedefinieerd door
$f(n,\vec{y}) = \mu x<n[R(x,\vec{y})]$,
dan is f een prim rec funktie.

bewijs. f kan gedefinieerd worden door

$$\begin{cases} f(0,\vec{y}) = 0 & \text{(zo'n } x < 0 \text{ bestaat niet)} \\ f(n+1,\vec{y}) = \begin{cases} 0 & \text{als } R(0,\vec{y}) \text{ (de kleinste } x \text{ met } R(x,\vec{y}) \text{ is dus 0,} \\ & \quad \text{deze is } < n+1) \\ f(n,\vec{y}) & \text{als } f(n,\vec{y}) \neq 0 \text{ (dan nl is } x = f(n,\vec{y}) < n \text{ dus} \\ & \quad \text{zeker } < n+1 \\ n & \text{als } \neg R(0,\vec{y}) \wedge \neg f(n,\vec{y}) \neq 0 \text{ én } R(n,\vec{y}) \\ 0 & \text{anders} \quad \text{(een } x \text{ met } R(x,\vec{y}) \text{ en } x<n+1 \text{ bestaat niet)} \end{cases} \end{cases}$$

Deze definitie kunnen we brengen in de vorm

$$\begin{cases} f(0,\vec{y}) = 0 \\ f(n+1,\vec{y} = h(f(n,\vec{y}),n,\vec{y}) \end{cases}$$

waarin h met onderscheid naar gevallen (op prim rec relaties) is gedefinieerd. Hieruit volgt dat ook f prim rec is. ☒

7.15. Oefening

a. Werk bovenstaande bewijsschets uit. Let i.h.b. op de wederzijdse uitsluiting van de gevallen in de definitie van h.

b. Ga na dat de begrenzing ook een rol mag spelen in de relatie waarover geminimaliseerd wordt, d.w.z. ga na dat

$f(n,\vec{y}) = \mu x<n[R(x,n,\vec{y})]$ prim rec is.

Wenk: definieer $h(n,z,\vec{y}) = \mu x<n[R(x,z,\vec{y})]$,
dan is h prim rec, en definieer dan
$f(n,\vec{y}) = h(n,n,\vec{y})$, dan is ook f prim rec.

c. Ga na dat als g en R prim rec zijn, en f is gedefinieerd door $f(\vec{y}) = \mu x<g(\vec{y})[R(x,\vec{y})]$, dan f ook prim rec is.

7.16. Stelling De klasse der prim rec relaties is gesloten onder begrensde existentiele en universele kwantificatie, d.w.z.
als R prim rec is en P en Q zijn gedefinieerd door
$P(n,\vec{y}) \leftrightarrow \exists x<n[R(x,\vec{y})]$, $Q(n,\vec{y}) \leftrightarrow \forall x<n[R(x,\vec{y})]$,
dan zijn ook P en Q prim rec.

bewijs We kunnen P en Q ook definieren door
$P(n,\vec{y}) \leftrightarrow \mu x<n+1[R(x,\vec{y}) \vee x=n]<n$, of ook door
$P(n,\vec{y}) \leftrightarrow R(\mu x<n[R(x,\vec{y})], \vec{y})$.
$Q(n,\vec{y}) \leftrightarrow \neg\exists x<n[\neg R(x,\vec{y})]$

Dus P en Q zijn prim rec. ☒

Nota bene, de prim rec relaties zijn niet gesloten onder onbegrensde kwantificatie!

We zullen het tot nu toe ontwikkelde gereedschap gebruiken om handige, prim rec, coderingen te definieren. Dat gebeurt in (7.17), (7.18) en 7(19).

7.17. Codering van n-tallen in één getal.

Zij $p_1,p_2,p_3,\ldots$ de opeen volgende priemgetallen. Dus $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$ enzovoort. We definieren een codering $<>$ als volgt.

$$<x_1,\ldots,x_n> = p_1^{x_1+1} . p_2^{x_2+1} . \ldots p_n^{x_n+1}$$

(en $<> = 1$)

Ga na dat het lege rijtje en de rijtjes (0) en (0,0) en (0,0,0) enz. alle verschillend gecodeerd worden. Waarvan is 120 een code ? Is 40 een code van een of ander rijtje ? We "definieren" een decodering $()_i$ door te eisen.

$$(<x_1,\ldots,x_n>)_i = x_i \text{ mits } 1 \leq i \leq n$$

Dus $(120)_1 = 2$, $(120)_2 = 0$, $(120)_3 = 0$ want $120 = 2^3.3^1.5^1 = <2,0,0>$.

Voor 40 is niet bepaald wat $()_1,()_2,()_3$ is, want 40 treedt nooit op als waarde van $<>$. We "definieren" een lengtefunktie ℓ door te eisen

$$\ell(<x_1,\ldots,x_n>) = n.$$

Dus als z de code is van een of ander rijtje, dan geeft $\ell(z)$ aan uit hoeveel getallen dat rijtje bestaat. Let wel, we hebben hiermee niet bepaald wat $\ell(z)$ is, indien z géén code is.
Ga na dat $\ell(120) = 3$.

Stelling De codering $<>$, opgevat als n-plaatsige funktie, en de decodering $(\ldots)_i$, opgevat als 2 plaatsige funktie van i en ...., en de lengtefunktie ℓ zijn prim rec.

bewijs De deelbaarheidsrelatie (notatie $x|y$ voor "x deelt y") is

prim rec, want $x|y \leftrightarrow \exists z<y+1[z.x=y]$. De priemrelatie (notatie priem(x) voor "x is priem") is prim rec, want priem(x) $\leftrightarrow x \neq 1 \wedge \forall y<x+1[y|x \rightarrow y=1 \vee y=x]$. De priemfunktie, p(i) = het i-de priemgetal (notatie p_{i+1} i.p.v. p(i)), is prim rec want, omdat er tussen p_n en $p_n! + 1$ tenminste één priemgetal ligt, kunnen we definieren

$$\begin{cases} p(0) = 2 \\ p(n+1)=\mu x<p(n)! + 1\ [priem(x) \wedge x>p(n)] \end{cases}$$

(Door de notatie afspraak $p(i)=p_{i+1}$ is verkregen dat $p_1=2$, $p_2=3$ enz!)
We kunnen de n-plaatsige funktie <> nu definieren door

$$<x_1,\ldots,x_n> = p_1^{x_1+1} \cdot p_2^{x_2+2} \cdot \ldots \cdot p_n^{x_n+1}$$

(let wel, n is een vast getal dus hier staat een vast aantal keer herhaalde toepassing van compositie met de vermenigvuldiging). De funktie P(i,x) = de exponent van de priemfactor p_i in de ontbinding van x, is prim rec, want we kunnen definieren $P(i,x) = \mu y<x[p_i^y|x \wedge \neg p_i^{y+1}|x]$. Nu kunnen we de decoderingsfunktie definieren door $(x)_i = P(i,x) \dot{-} 1$. Dus de decoderingsfunktie is prim rec. Tenslotte is ook de lengtefunktie prim rec, want $\ell(x)$ = het nummer van het grootste priemgetal met exponent $\neq 0$ in de ontbinding van x, dus we kunnen definieren $\ell(x) = \mu y<x[p_y|x \wedge \forall z<x[z>y \rightarrow \neg p_z|x]]$ ☒

Let wel door bovenstaande definities zijn de decodering $()_i$ en lengtefunktie ℓ totaal gedefinieerd. Dus terwijl 40 niet optreedt als waarde van <>, is $\ell(40)$ toch bepaald (ga na wat $\ell(40)$ is) en ook $(40)_1$ en ook $(<x_1,\ldots, x_n>)_i$ voor i=0, i > n (ga na wat $(120)_7$ is).

Oefening Definieer een prim rec funktie * zó dat

$$<x_1,\ldots,x_n> * <y_1,\ldots,y_m> = <x_1,\ldots,x_n,y_1,\ldots,y_m>$$

(dus *(x,y) wordt geschreven als x * y, d.w.z. met infixnotatie i.p.v. prefixnotatie). Ga na wat de door u gedefinieerde * doet met x * y wanneer er geen $x_1,\ldots,x_n$ zijn met $x = <x_1,\ldots,x_n>$, zoals bijvoorbeeld x = 40. Is <0> * <0> = <0,0> ?

Definieer ook een prim rec funktie subst zó dat

$$subst(<y_1,\ldots,y_k,\ldots,y_n>, x, k) = <y_1,\ldots,y_{k-1}, x, y_{k+1},\ldots y_n>$$

voor k = 0, 1,...,n, n+1.

7.18. Coderen van symboolrijen met een letter-woord-zin structuur.

Soms moeten we rijen van woorden op effectieve manier in getallen coderen en decoderen. Bijvoorbeeld om Turingmachines met één getal geheel te kunnen beschrijven, of om met één getal de symbolische definitie van een prim rec funktie te kunnen geven.

Een effectieve codering van dergelijke symboolrijen in getallen noemt men vaak Gödelnummering en de getallen gödelnummers. (Zie lit. K. Gödel, waarin deze methode voor het eerst wordt toegepast). Dit kan bijvoorbeeld als volgt.

Zij $A = \{a_0, a_1, \ldots\}$ een (on)eindig alfabet. We coderen de symbolen $a_0, a_1, \ldots$ als de getallen 1, 2, 5,..., d.w.z. $\langle a_i \rangle = 2.i+1$. Een woord $w = s_1 s_2 \ldots s_n$ over A krijgt dan het gödelnummer $\langle w \rangle = \langle\langle s_1 \rangle, \ldots, \langle s_n \rangle\rangle$, dus $\langle w \rangle =$

$$p_1^{\langle s_1 \rangle + 1} . p_2^{\langle s_2 \rangle + 1} . \ldots . p_n^{\langle s_n \rangle + 1}.$$

Een woordenrij $r = w_1, \ldots, w_m$ wordt evenzo gecodeerd als $\langle r \rangle = \langle\langle w_1 \rangle, \ldots, \langle w_m \rangle\rangle$, dus $\langle r \rangle =$

$$p_1^{\langle w_1 \rangle + 1} . p_2^{\langle w_2 \rangle + 1} . \ldots . p_m^{\langle w_m \rangle + 1}.$$

<u>Oefening</u> Neem $A = \{a, b, c, \ldots, z\}$, dus $\langle a \rangle = 2.0+1=1$, $\langle b \rangle = 2.1+1=3, \ldots, \langle z \rangle = 2.25+1=51$, en ga na of de volgende getallen gödelnummers zijn:

$$2, 5, 9, \quad 2^3.3^5.5^1, \quad 2^4.5^2.7^4, \quad 2^4.3^2.5^4.7^4, \quad 2^{2^4 3^2}, \quad 2^{2^4 3^2 + 1}.3^{2^3 + 1}.$$

Decodeer het getal $2^{2^{14}.3^{30}.5^{10}.7^8 + 1}.3^{2^{14}.3^{10}.5^8.7^2.11^2.13^{28}} + 1.5^{2^{16}.3^{30}.5^{30}.7^{36}} + 1$

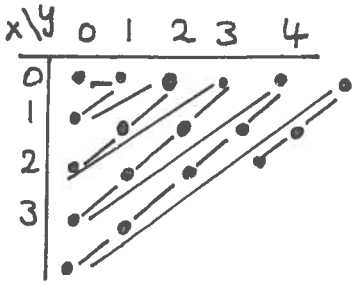
Het is niet toevallig dat U van deze getallen kon beslissen of ze al dan niet gödelnummers waren, en van welke soort. Die eigenschappen, of wel relaties, zijn prim rec! Bijvoorbeeld de relatie is-gödelnummers-van-woord, is-g-v-woord, kan prim rec gedefinieerd worden door

$\text{is-g-v-woord}(z) \leftrightarrow \forall i \cdot \ell(z) \mid \text{is-g-v-e-symb}((z)_i) \vee i = 0 \mid$, waarbij

is-g-v-symb(z) $\longleftrightarrow$ is-oneven(z) , waarbij
is-oneven(z) $\longleftrightarrow \exists\, x<z[2.x+1=z]$

Oefening Definieer net zo de prim rec relatie is g-v-rij-woorden(z). Bovendien kunnen we bewijzen dat de relaties is-g-v- ymb, is g-v--woord en is-g-v-rij-woorden elkaar uitsluiten: de gödelnummers van symbolen zijn oneven, die van woorden beginnen met een even macht van twee, die van rijen met een oneven macht van twee. Dat de symbolen onderlingverschillende gödelnummers hebben is duidelijk, net zo de woorden onderling en de rijen onderling.

7.19. Een surjectieve codering op N.
Soms willen we dat ieder getal van N als code optreedt.
Dit kunnen we bereiken met de volgende 2-plaatsige paringfunktie par2:
par2(x,y) = $\frac{1}{2}((x+y)^2+3x+y) = \frac{1}{2}(x+y)(x+y+1)+x$. Deze is prim rec en telt de paren natuurlijke getallen af op de volgende manier:



(Bewijs evt. zelf met volledige inductie naar n dat er voor willekeurige n een x en y zijn zó dat n = par2(x,y). Ga na dat par2 éénduidig is).

De 1-plaatsige projectiefunkties zijn ook prim rec:
$proj2_1(z) = \mu x\leq z[\exists y\leq z[par2(x,y)=z]]$ en
$proj2_2(z) = \mu y\leq z[\exists x\leq z[par2(x,y)=z]]$.

Voor n > 2 definieren we de paringfunktie parn door
$parn(x_1,\ldots,x_n) = par2(\ldots.par2(par2(x_1,x_2),x_3)\ldots.,x_n)$ en de projectiefunkties $projn_i$ op soortgelijke manier.
Omdat alle paringfunkties surjectief zijn, d.w.z. dat ieder getal eens als waarde optreedt, missen we natuurlijk wel de handige lengte-funktie.
Oefening. Ga na dat par2(0,0) = 0, par2(4,2) = 27. Wat is par2(2,4) ? Wat is par3(0,0,0) ? Wat zijn $proj3_1(27)$, $proj3_2(27)$, $proj3_3(27)$? Wat zijn $proj3_1(0)$, $proj3_2(0)$ en $proj27_1(27)$, $proj27_{27}(27)$?

7.20. Als laatste stap in de afleiding van schema's waaronder de klasse der prim rec funkties gesloten is, geven we nu twee veralgemeningen van het schema van recursie, nl algemene recursie en simulatane recursie.

ad "simultane recursie".

Zoals de naam zegt kunnen volgens dit schema verscheidene funkties simultaan recursief gedefinieerd worden, bijv.

$$\begin{cases} f(0,x) = x \\ g(0,x) = 0 \end{cases} \qquad \begin{cases} f(n+1,x) = g(n,x) * f(n,x) + 3 \\ g(n+1,x) = f(n,x) + g(n,x) * 2 \end{cases}$$

Ga eens voor U zelf na of U deze wijze van definieren nog intuitief berekenbaar zou willen noemen.

ad "algemene recursie".

Volgens dit schema kan een funktie gedefinieerd worden door de waarde bij argumentwaarde n+1 uit te drukken in een of meer van de funktie-waarden bij argumentwaarden n, n-1,...,2,1,0 (en niet alleen bij argumentwaarde n, zoals het eenvoudige recursieschema toelaat).

Bijvoorbeeld, als f gedefinieerd is door

$$\begin{cases} f(0) = 1 \\ f(1) = 1 \\ f(n) = f(n-1) + f(n-2) \text{ voor } n \geq 2 \end{cases}$$

(ga na wat f is! Vindt U f intuitief berekenbaar ?), dan zal op grond van het algemene recursie schema f een prim rec funktie zijn.

Het is echter moeilijk de algemene recursie in een schema te formuleren. Immers, als $f(n,\vec{x})$ recursief uitgedrukt zou mogen worden in een zeker aantal van de waarden $f(n-1,\vec{x}),\ldots,f(0,\vec{x})$, dan zou dit geformuleerd kunnen worden als

$$f(n,\vec{x}) = h(f(n-1,\vec{x}),\ldots,f(0,\vec{x}),\ n,\ \vec{x}).$$

Maar bij variabele n zou h varieren in zijn <u>aantal</u> argumenten! Dat kan niet, omdat h een gegeven, dus vast liggende, prim rec funktie moet zijn.

Om toch iets dergelijks te kunnen formuleren voeren we het begrip geschiedenisfunktie in.

7.21. Definitie Als f een funktie is, dan is de geschiedenisfunktie van f (notatie $\vec{f}$) gedefinieerd door

$$\vec{f}(n,\vec{x}) = \langle f(0,\vec{x}),\ldots,f(n-1,\vec{x})\rangle$$

($\vec{f}(0,\vec{x}) = 1$ in het bijzonder).

Als we $f(n,\vec{x})$ nu "recursief" mogen uitdrukken in de enkele waarde $\vec{f}(n,\vec{x})$, dan zijn daarmee impliciet alle waarden $\vec{f}(n,\vec{x}),\ldots,f(0,\vec{x})$ bereikbaar.

7.22. Stelling (algemene recursie). Als g prim rec is en f (dus tegelijk ook $\vec{f}$) is gedefinieerd door

$$f(n,\vec{x}) = g(\vec{f}(n,\vec{x}),\ n,\ \vec{x}),$$

dan is ook f prim rec.

Bewijs Omdat $g(\vec{f}(n,\vec{x}),n,\vec{x})$ juist de waarde van $f(n,\vec{x})$ is, is het duidelijk dat hetvolgende een juiste definitie van $\vec{f}$ is:

$$\begin{cases} \vec{f}(0,\vec{x}) = 1 \\ \vec{f}(n+1,\vec{x}) = \vec{f}(n,\vec{x}) * \langle g(\vec{f}(n,\vec{x}),\ n,\ \vec{x})\rangle \end{cases}$$

waarbij * gegeven is in oefening (7.17). Hieruit blijkt tevens dat $\vec{f}$ prim rec is. We kunnen nu f prim rec definieren door

$$f(n,\vec{x}) = (\vec{f}(n+1,\vec{x}))_n.$$ ∎

7.23. Stelling (simultane recursie). Als $g_1,\ldots,g_m$ en $h_1,\ldots,h_m$ prim rec zijn en $f_1,\ldots,f_m$ zijn gedefinieerd door

$$\begin{cases} f_1(0,\vec{x}) = g_1(\vec{x}) \\ \vdots \\ f_m(0,\vec{x}) = g_m(\vec{x}) \end{cases} \qquad \begin{cases} f_1(n+1,\vec{x}) = h_1(f_1(n,\vec{x}),\ldots,f_m(n,\vec{x}),\ n,\ \vec{x}) \\ \vdots \\ f_m(n+1,\vec{x}) = h_m(f_1(n,\vec{x}),\ldots,f_m(n,\vec{x}),\ n,\ \vec{x}), \end{cases}$$

dan zijn ook $f_1,\ldots,f_m$ prim rec.

Bewijs We definieren één funktie, notatie $\langle f\rangle$, die als waarde op $(n,\vec{x})$ juist $\langle f_1(n,\vec{x}),\ldots,f_m(n,\vec{x})\rangle$ heeft.

$$\begin{cases} \langle f\rangle\ (0,\vec{x}) = \langle g_1(0,\vec{x}),\ldots,g_m(0,\vec{x})\rangle \\ \langle f\rangle\ (n+1,\vec{x}) = \langle h_1(\ (\langle f\rangle(n,\vec{x}))_1,\ldots,(\langle f\rangle(n,\vec{x}))_m\ ,\ n\ ,\ \vec{x})\ , \\ \qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad h_m(\ (\langle f\rangle(n,\vec{x}))_1,\ldots,(\langle f\rangle(n,\vec{x}))_m\ ,\ n\ ,\ \vec{x})\rangle \end{cases}$$

Dus $\langle f\rangle$ is prim rec.

Elk der f_i's kan nu gedefinieerd worden door

$$f_i(n,\vec{x}) = (\langle f \rangle\ (n,\vec{x}))_i$$

en is dus prim rec. □

Het is nu wel duidelijk dat veel intuitief berekenbaar te noemen funkties prim rec zijn. Het zou kunnen dat we een redelijke karakterisering hebben gevonden van de intuitief berekenbaar te noemen funkties. Maar helaas. . . .

7.24. Stelling Er zijn intuitief berekenbare funkties die niet prim rec zijn.

Bewijs We gebruiken de techniek van diagonalisatie. (Dit is de techniek die Cantor gebruikte om te bewijzen dat de reele getallen overaftelbaar zijn. De techniek van diagonalisatie speelt een grote rol in de recursietheorie). Er zijn slechts aftelbaar veel prim rec funkties. Zij $f_0, f_1, \ldots$ een aftelling van de éénplaatsige prim rec funkties. We definieren een nieuwe funktie f die verschillend is van alle f_i $(i=0,1,2,\ldots)$ als volgt.

f:	+1	+1	+1	+1	...			
n =	0	1	2	3	4	5	6	7 ...
f_0:	*	*	*	*	*	*	*	...
f_1:	*	*	*	*	*	*	...	
f_2:	*	*	*	*	*	...		
⋮				*				
⋮					*			
⋮						*		

(ziet U de diagonalisatie ?)

Het is duidelijk dat f verschillend is van alle f_i en dus niet prim rec is. Dit kunnen we formeler zeggen als volgt: Definieer f door $f(n) = f_n(n) + 1$ voor alle n. Als nu f wél prim rec zou zijn, dan zou f ergens in de aftelling voorkomen, zeg als f_m. Uit $f_m(m) = f(m) = f_m(m)+1$ volgt dan de tegenspraak. Dus f kan niet in de aftelling voorkomen en is dus niet prim rec.

Omdat het in beginsel niet moeilijk is de aftelling $f_0, f_1 \ldots$ effectief

te geven (ga na hoe U dit zou doen, zie (7.25), is het duidelijk dat f intuitief berekenbaar is. □

7.25. Een opsomming van alle prim rec funkties.

Omdat alle prim rec funkties volgens de definitie van *PR* gedefinieerd zijn door herhaalde compositie en recursie uitgaande van de basisfunkties, kunnen wij ze bijv. als volgt allemaal opsommen.
Beschouw de rij $Z^{(1)}$, $S^{(1)}$, $U_1^{(1)}$, $U_1^{(2)}$, $U_2^{(2)}$, $U_1^{(3)}$, $U_2^{(3)}$, $U_3^{(3)}$, $U_1^{(4)}$, ...
(De bovenschriften geven het aantal veranderlijken aan).
We doorlopen deze rij en breiden hem daarbij uit op de volgende manier. Als we zijn aangekomen bij zekere funktie, zeg f, dan voegen we een eindig stel funkties toe en plaatsen die vóór (d.w.z. links van) f. Daarmee is gegarandeerd dat we op den duur iedere funktie uit de oorspronkelijke rij bereiken.
Het eindig stel funkties dat we toevoegen bepalen we als volgt: beschouw alle voorgaande funkties in de huidige rij (dat zijn er eindig veel) en neem voor ieder stel funkties waarvoor het zinvol is, de compositie en recursie-samenstelling. (In geval van compositie moet het stel funkties uit een m-tal bestaan met een gelijk aantal veranderlijken én nog één m-plaatsige funktie. In geval van recursie moet het stel funkties uit één n - 1-plaatsige en één n + 1-plaatsige bestaan, voor een of andere $n \geq 2$). (We kunnen de toe te voegen funkties bijvoorbeeld noteren door $COMP^{(n)}(g_1^{(n)}, g_2^{(n)}, \ldots, g_m^{(n)}, h^{(n)})$ en $REC^{(n)}(g^{(n-1)}, h^{(n+1)})$.
Op deze manier verschijnt op den duur iedere prim rec funktie in de rij. (Immers, zij f prim rec, dan is er een rij funkties $f_o, f_1, \ldots, f_k$ zódat iedere f_i ofwel de compositie of recursie is op voorgaande f_j's ofwel zélf een der basisfunkties is, en $f_k = f$. Met induktie bewijzen we dat alle $f_o, \ldots, f_k$ op den duur voorkomen in de rij. Voor f_o is het duidelijk. Stel nu dat $f_o, \ldots f_i$ in de rij voorkomen, zeg vóór U_1^n. Als f_{i+1} een basisfunktie is dan komt f_{i+1} ook voor, (bijvoorbeeld als U_1^m met $m >> n$!!!), en anders komt f_{i+1} zeker voor wanneer we in de rij bij u_2^n zijn aangekomen.)
Aldus hebben we een opsomming van alle prim rec funkties.

Oefening: Geef zelf een opsomming van alle 1-plaatsige prim rec funkties.

8. Recursieve funkties.

8.1. We doen een volgende poging de intuitief berekenbare funkties te karakteriseren. We breiden de klasse van prim rec funkties uit door onbegrensde minimalisatie toe te laten.

$$\mu x[R(x,\vec{y})] = \begin{cases} \text{de kleinste } x \text{ zó dat } R(x,\vec{y}). \ . \ . \ . & \text{indien zo'n } x \text{ bestaat} \\ \text{ongedefinieerd} \qquad\qquad . \ . \ . \ . & \text{anders} \end{cases}$$

Daarbij moeten we wel de zekerheid hebben dat het "zoek-proces" eindigt, i.e. dat we niet buiten de klasse van totale funkties treden. Dit wordt bereikt door de "regulariteits"-voorwaarde $\forall \vec{y} \exists x[R(x,\vec{y})]$ te eisen, wanneer we de minimalisatie toepassen. Omwille van de "zuinigheid" in de definitie geven we de minimalisatie in eerste instantie alleen voor het geval dat $R(x,\vec{y})$ van de vorm $h(x,\vec{y}) = 0$ is.

8.2. Definitie De klasse R van recursieve funkties, rec funkties, wordt inductief gedefinieerd door

I' : de volgende funkties zijn rec,
Z, S en de U_i^n $(i=1,\ldots,n)$ voor $n=1,2,\ldots$

II': R is gesloten onder de volgende schema's,
a. compositie (van rec funkties),
b. recursie (op rec funkties),
c. reguliere minimalisatie (op rec funkties), d.w.z. als h rec is en $\forall \vec{y} \ \exists x[h(x,\vec{y}) = 0]$ en f is gedefinieerd door $f(\vec{y}) = \mu x[h(x,\vec{y}) = 0]$, dan is ook f rec.

8.3. Oefening Bewijs dat de prim rec funkties ook rec zijn.

8.4. Definitie Een relatie heet recursief als zijn karakteristieke funktie dat is.

8.5. Oefening Geef de analogieen van (7.7), (7.9), (7.11), (7.12), (7.14), (7.15), (7.16), (7.20), (7.22) en (7.23) voor "rec" i.p.v. "prim rec". en bewijs ze. Wat dacht U van een analogon van (7.24) en (7.25) ?

Intuitief verschillen de begrensde en onbegrensde minimalisatie wezenlijk. De vraag is of onze intuitie ons niet misleidt.

8.6. Stelling Er zijn rec funkties die niet prim rec zijn.

Schets van het bewijs.

1. Op grond van de straks te geven These van Church kunnen we geloven dat de in het bewijs van stelling (7.24) gedefinieerde funktie f een rec funktie is. Uit datzelfde bewijs bleek ook dat f niet prim rec is.

2. Wat direkter kunnen we de stelling bewijzen m.b.v. de Ackermann funktie A (zie lit. W. Ackermann), gedefinieerd door

$$A(m,n) = \begin{cases} n + 1 & \text{als } m = 0, \\ A(m\dot{-}1,1) & \text{als } m \neq 0 \wedge n = 0 \\ A(m\dot{-}1,\ A(m,n\dot{-}1)) & \text{anders} \end{cases}$$

Ga na dat hetvolgende schema voor de waarden van A klopt

m \ n	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12	
1	2	3	4	5	6	7	8	9	10	11	12		
2	3	5	7	9	11								
3	5												
4													

Vul dit schema eens aan voor m=4,n=0 en voor m=4, n=4 ?!

Deze funktie is geinspireerd op de manier waarop steeds volgens eenzelfde schema de optelling prim rec kan worden gedefinieerd en m.b.v. de optelling de vermenigvuldiging en m.b.v. de vermenigvuldiging de machtsverheffing enzovoort, steeds m.b.v. de voorgaande funktie een funktie die nog sneller stijgt. De definitie van de Ackermann-funktie diagonaliseert over die funkties en is daarmee kandidaat voor een funktie die sneller stijgt dan enige prim rec funktie en derhalve niet prim rec kan zijn. Dit kan ook met enig gepeuter formeel bewezen worden, zie Yasuhara 6.2 of Hermes 3.§ 13.

Het bewijs dat A recursief is verloopt als volgt.

We coderen de totale berekening die nodig is voor de bepaling van de waarde van A(m,n) op een voor de hand liggende manier (gödelnummering). Met geschikte keuze van de gödelnummering is de relatie $T_A(m,n,y) \leftrightarrow_{def}$ "y is het gödelnummer van de berekening voor A(m,n)" prim rec. Met geschikte keuze is ook de waarde van A(m,n) op prim rec manier te verkrijgen uit de codering van de totale berekening voor A(m,n), zeg middels een prim rec funktie U. Dus A kan dan gedefinieerd worden door

$$A(m,n) = U(\mu y[T_A(m,n,y)])$$

en is dus rec. Als voorbeeld van zo'n geschikte gödelnummering van de totale berekening, noemen we de volgende. "De berekening van A(m,n)" wordt gecodeerd als $\langle a_o,a_1,\ldots,a_k\rangle$ waarin $a_o,a_1,\ldots,a_k$ de verschillende waarden van de in de berekening optredende A(i,j) zijn (in chronologische volgorde, dus $a_k = A(m,n)$. De relatie $T_A(m,n,y)$, i.e. "y is de codering van de berekening van A(m,n)" kan gedefinieerd worden door

$$T_A(m,n,y) \leftrightarrow \begin{cases} y = \langle n+1\rangle & \ldots \text{ als } m = 0 \\ T_A(m\dot{-}1,1,y) & \ldots \text{ als } m \neq 0,\ n = 0 \\ \exists x_1<y\ \exists x_2<y\ [T_A(m,n\dot{-}1,x_1) \wedge T_A(m\dot{-}1,(x_1)_{\ell(x_1)},x_2) \wedge y = x_1 * x_2] & \ldots \text{ anders} \end{cases}$$

$T_A(m,n,y)$ is dus een prim rec relatie.

(nb. de recursie in de twee variabelen m en n kan teruggebracht worden tot een recursie in één variabele door de analoge definitie te geven voor $\tilde{T}_A(k,y)$, waarbij k geinterpreteerd moet worden als $k = \langle m,n\rangle$ zo dat $\tilde{T}_A(k,y) \leftrightarrow T_A((k)_1, (k)_2,y))$. De projectiefunktie U moet zodanig zijn dat $U(\langle a_o,\ldots,a_n\rangle) = a_n$, i.e. $U(y) = (y)_{\ell(y)}$ en is dus prim rec. ∎

8.7. Van het gedeelte over prim rec funkties heeft U in opgave (8.5) van vrijwel alles de analoge formuleringen en bewijzen moeten geven voor rec funkties en relaties i.p.v. prim rec funkties en relaties. Maar om het bestaan aan te tonen van een intuitief berekenbare maar niet rec funktie (vgl. (7.24) en (7.25)) zult U moeilijkheden krijgen. Weliswaar kunt U met behulp van het diagonaalargument inzien dat er een niet rec funktie bestaat, maar het is de vraag of die funktie intuitief berekenbaar

genoemd kan worden. De rec funkties zijn aftelbaar, maar hun aftelling is niet zonder meer effectief. Want bij toepassing van het schema van onbegrensde minimalisatie op een funktie h, moet er de regulariteitsvoorwaarde $\forall \vec{y}\ \exists x[h(x,\vec{y}) = 0]$ gelden. Dit kunnen we niet effectief beslissen!! Ga na. (We zullen het in (10.12) bewijzen).
Ook op andere gronden dan dit "negatieve" resultaat blijkt dat recursiviteit een goede karakterisering is van intuitieve berekenbaarheid. Dit zullen we in stelling (9.10) nader aantonen.

We moeten ons wel realiseren dat we tot nu toe alleen met totale funkties gewerkt hebben, d.w.z. dat voor ieder element uit het domein van de funktie de funktiewaarde bepaald is.
Maar in de vorige hoofdstukken over Turingmachines speelde de partieliteit een wezenlijke rol: niet voor iedere beginbandexpressie is de eindwaarde van de berekening door de Turingmachine bepaald, omdat er gevallen zijn waarin de Turingmachine niet stopt! De door een Tm gerealiseerde partiele funktie wordt echter wel intuitief berekenbaar genoemd.
In het volgende hoofdstuk zullen we de aandacht op de partiele funkties richten.

9. Partieel recursieve funkties.

We doen een poging de intuitief berekenbare partiele funkties te karakteriseren m.b.v. het begrip partieel recursieve funktie.

9.1. Notatie Wanneer bij een partiele funktie de waarde voor $\vec{x}$ onbepaald is (de berekening van f op $\vec{x}$ divergeert), schrijven we $f(\vec{x})\uparrow$. Als de waarde van f op $\vec{x}$ bepaald is (de berekening van f op $\vec{x}$ convergeert), schrijven we $f(\vec{x})\downarrow$.

9.2. Omdat we nu met partiele funkties werken, moeten we van alle samenstellingsschema's precies afspreken (= definieren), wanneer de betreffende samenstellingen bepaald dan wel onbepaald zijn. We beperken ons hier tot de drie bekende schema's. Daarmee zijn immers alle andere samenstellingen gedefinieerd.
We stellen ons op het standpunt dat alle argumentwaarden bepaald moeten zijn om het resultaat van een funktie-toepassing eventueel nog bepaald te laten zijn. Dus $h(g_1(\vec{x}),\ldots,g_m(\vec{x}))\uparrow$ zodra een of meer der $g_i(\vec{x})\uparrow$. Dit is in overeenstemming met een berekeningswijze die we stilzwijgend steeds na zullen leven. Een gevolg is wel dat Z(...) onbepaald is, wanneer de argumentwaarde onbepaald is!! (Er zijn verscheidene afspraken mogelijk. Veel resultaten zijn onafhankelijk van deze afspraken. Sommige echter niet. Bijvoorbeeld de uitspraak dat Z(...) altijd bepaald is! We zullen ons met dit aspect niet bezighouden.).
De nagevolgde berekeningswijze in geval van recursie is dat we, om $f(m,\vec{x})$ te bepalen, achtereenvolgens $f(0,\vec{x}),\ldots,f(m-1,\vec{x})$ berekenen en dan $f(m,\vec{x})$. Dus $f(m,\vec{x})\uparrow$ zodra een voorgaande $f(n,\vec{x})\uparrow$ met $n \leq m$, ofwel $f(n,\vec{x})\uparrow$ voor alle $n \geq m$ zodra $f(m,\vec{x})\uparrow$. (Ook hier zijn verscheidene afspraken mogelijk, ieder al of niet gebaseerd op een efficiente berekeningswijze. We zullen ons er niet mee bezig houden).
In geval van minimalisatie stellen we ons op het volgende standpunt. Graag willen we dat geldt

$$\mu x[h(x,\vec{y}) = 0]\downarrow \quad \text{alsla} \quad \exists x[h(x,\vec{y}) = 0].$$

dit heeft tot gevolg dat we $\mu x[h(x,\vec{y}) = 0]$ soms bepaald moeten laten zijn zonder dat $h(0,\vec{y}),\ldots,h(x-1,\vec{y})$ allemaal bepaald zijn! We

definieren dus

$$\mu x[h(x,\vec{y}) = 0] = \begin{cases} \text{de kleinste } x \text{ zó dat } h(x,\vec{y}) = 0 \ldots \text{als zo'n } x \text{ bestaat} \\ \text{onbepaald} \qquad \text{anders} \end{cases}$$

(Vraag: welke berekeningswijze wordt ons door deze definitie opgelegd ? Kunt U zich andere berekeningswijzen in geval van minimalisatie voorstellen ? We zullen ons er niet mee bezighouden). Deze definitie heeft tot gevolg dat minimalisatie alleen op totale funkties mag worden toegepast. Want de minimalisatie op partiele funkties is zeker niet berekenbaar te noemen: immers, hoe zou U kunnen bepalen of $h(0,\vec{y}),\ldots,h(x-1,\vec{y})$ allen ongelijk nul zijn indien een of meer van hen divergeren ? (Denk aan de onbeslisbaarheid van het stopprobleem).

Oefening. Geef een totale funktie h terwijl toch $\mu x[h(x,y) = 0]$ voor alle x en y onbepaald is. (Wenk: zorg dat h nooit nul is). Geef een funktie g zó dat $f(y) = \mu x[g(x,y) = 0]$ onbepaald is voor $y = 0,\ldots,8$ en verder bepaald! Wat kunt U zeggen van de volgende definitie

$$\mu x[h(x,y) = 0] = \begin{cases} \text{de kleinste } x \text{ zó dat } h(x,\vec{y}) = 0 \text{ als zo'n } x \text{ bestaat} \\ 0 \qquad \text{anders} \end{cases}$$

(vergelijk de begrensde minimalisatie).

Beschouw nog eens de definitie van de klasse *R* der rec funkties. Door de regulariteitsvoorwaarde bij de minimalisatie te laten vervallen, breiden we niet alleen de klasse uit maar worden er ook partiele funkties "gegenereerd".

9.3. Definitie. De klasse *P* der partieel recursieve funkties, part rec funkties, wordt inductief gedefinieerd door

I" : de volgende funkties zijn part rec,
Z,S, en de $U_i^n (i=1,2,\ldots,n)$ voor $n = 1,2,\ldots$

II" : *P* is gesloten onder
a. compositie (van part rec funkties)
b. recursie (op part rec funkties)
c. minimalisatie op totale part rec funkties

9.4. Oefening

a. Zijn de part rec funkties aftelbaar ? En de rekenkundige ?

Bewijs $PR \subseteq R \subseteq P \subseteq$ {rekenkundige funkties}.

b. Bewijs dat de volgende funkties part rec zijn.

$f(n,x) = \mu y[n^y \geq x]$. N.B. is f totaal ? Is voor $x > 0$ $f(x,x)\downarrow$ en $f(1,x)\downarrow$?

$g(n,x) = \mu y[(n+2)^y \geq x]$. N.B. is g totaal ? rec ? prim rec ? (Wenk: geef definities voor f en g volgens II"c en gebruik deel a van deze oefening).

c. Kan een funktie f totaal zijn, wanneer h niet totaal, part rec is en g rec is en $f = h \circ g$? Idem met $f = g \circ h$? Geef voorbeelden ter bevestiging en bewijs de ontkenningen.

Het is duidelijk dat iedere rec funktie ook een totale part rec funktie is. (Het omgekeerde is ook waar en zal straks bewezen worden. Ga na wat de moeilijkheden zijn op dit moment om het nu aan te tonen). Daarom zijn de part rec funkties gesloten onder minimalisatie op rec relaties.

9.5. Oefening Toon nogmaals aan dat in oef. (9.4)b. hierboven f en g part rec zijn.

9.6. Oefening Formuleer en bewijs de analogieen van de in (8.5) bedoelde afsluitingseigenschappen.

Zonder verder nog vingeroefeningen te geven om handigheid te krijgen in het formeel aantonen dat funkties part rec zijn, gaan we nu de gelijkwaardigheid van Turing-berekenbaarheid en recursiviteit bewijzen. Voor goed begrip herhalen we definitie (2.13) in iets gewijzigde terminologie.

9.7. Een funktie f heet Turing-berekenbaar als er een Tm T is zó dat T, indien gestart met als beginbandexpressie (een codering van)$x_o,\ldots,x_n$,

$$\begin{cases} \text{stopt met een codering van } f(x_o,\ldots,x_n) & \ldots \text{ indien } f(x_o,\ldots,x_n)\downarrow \\ \text{niet stopt} & \ldots \text{ indien } f(x_o,\ldots,x_n)\uparrow \end{cases}$$

We zeggen: T berekent (of realiseert) f.

9.8. Stelling (Gelijkwaardigheidsstelling).

a. Iedere part rec funktie is Turingberekenbaar.

b. Iedere Turingberekenbare funktie is part rec.

Bewijs

ad. a. We hoeven slechts aan te tonen dat Z,S en de U_i^n T-berekenbaar zijn (zie (3.10. 5 en 6)) en dat de T-berekenbare funkties gesloten zijn onder compositie (zie (3.9)), recursie en minimalisatie. Dit is in feite de makkelijke kant van het bewijs en kunt U zelf doen.

ad. b. Herinner U de gödelnummering van (7.18); we schrijven die hier als g. Dus $g(a_i) = \langle a_i \rangle = 2.i+1$ en $g(a_{i_1},\ldots a_{i_n}) = \langle\langle a_{i_1}\rangle \ldots,\langle a_{i_n}\rangle\rangle = p_1^{\langle a_{i_1}\rangle+1} . \ldots . p_n^{\langle a_{i_n}\rangle+1}$.

We zullen allereerst voor willekeurige Tm T laten zien dat -de rekenkundige funktie- $g\ f_T\ g^{-1}$ een part rec funktie is.
Welnu, zij $\mathcal{T} = (\{g_o,\ldots,g_n\},\{a_o,\ldots,a_m\}, \tau, g_o, g_n)$. Zonder verlies van algemeenheid mogen we veronderstellen dat τ precies alléén op de stoptoestand g_n niet bepaald is; voor alle andere toestanden g_i en alle symbolen s_j is $\tau(g_i,s_j)$ dus wel bepaald. (Ga dit zelf na!).
Denk U nu funkties S,Q en D in die zodanig zijn dat de transitietabel τ bestaat uit vijftallen $(g_i, s_j, g_{Q(i,j)}, s_{S(i,j)}, d_{D(i,j)})$.
Vanwege de eindig-heid van de tabel τ kunnen Q, S en D prim rec gekozen worden, zie (7.7)c.
Denk U ook funkties band, pos, sym en toes in, die zodanig zijn dat na t overgangen geldt - als de Tm gestart is met een beginbandexpressie waarvan w het gödelnummer is - :
band (t,w) = het gödelnummer van de huidige bandexpressie,
pos (t,w) = de huidige positie van de lees-schrijfkop (van links naar rechts op de bandexpressie geteld),
sym (t,w) = het gödelnummer van het symbool dat gelezen wordt,
toes (t,w) = het nummer van de huidige toestand.

Met behulp van Q, S en D en het schema van simultane recursie kunnen we deze funkties prim rec definieren (zie oefening 2 in (7.17) voor de

funktie subst):

$$\begin{cases} band\ (0,w) = w \\ pos\ (0,w) = 1 \\ sym(0,w)=((w)_1 \dot{-} 1)/2 \\ toes\ (0,w) = 0 \end{cases} \quad \begin{cases} band(t+1,w)=subst(band(t,w),sym(t+1,w).2+1,pos(t,w)) \\ pos(t+1,w)=pos(t,w)+D(toes(t,w),sym(t,w)) \\ sym(t+1,w)=S(toes(t,w),sym(t,w)) \\ toes(t+1,w)=Q(toes(t,w),sym(t,w)). \end{cases}$$

Nu geldt dat het gödelnummer van de uiteindelijke bandexpressie - wanneer die bestaat - gelijk is aan band $(\mu t[toes(t,w) = n],w)$ waarin w gelijk is aan het gödelnummer van de beginbandexpressie. Ofwel:

$$gf_Tg^{-1}(w) = band(\mu t[toes(t,w) = n],w).$$

Hieruit blijkt dat gf_Tg^{-1} part rec is!

Om aan te tonen dat alle rekenkundige Turing-berekenbare funkties part rec zijn, redeneren we als volgt.
Zij f een T-berekenbare funktie, dan is $f = \delta f_T\gamma$, voor zekere Tm T met bandalfabet A en codering γ: Nx...xN $\to A^*$ en decodering δ: $A^* \to$ N. Dus, omdat $gg^{-1}(x) = x$ voor alle x, is ook $f = \delta(g^{-1}g)\ f_T(g^{-1}g)\gamma = (\delta g^{-1})gf_Tg^{-1}(g\gamma)$. hierin zijn δg^{-1} en $g\gamma$ rekenkundige funkties en wanneer δ en γ maar eenvoudig genoeg bekozen zijn, zijn δg^{-1} en $g\gamma$ prim rec. Omdat gf_Tg^{-1} part rec is, volgt direkt dat f part rec is.

<u>Oefening</u> Zij γ: NxN $\to \{A,B,...\}^*$ zódat $\gamma(x,y) = AB\underbrace{...B}_{x}\ AB\underbrace{...B}_{y}A$. Wat is $g(\gamma(x,y))$? Geef een prim rec definitie voor $g\gamma$. ☒

9.9. De gelijkwaardigheid van Turing-Berekenbaarheid en partieel recursiviteit is een belangrijk resultaat.

1. Het stelt ons in staat eigenschappen van T-berekenbaarheid rechtstreeks over te nemen als eigenschappen van part recursiviteit, en omgekeerd. En het stelt ons in staat in bewijzen heen en weer te springen tussen de formulering m.b.v. Turingmachines en de formulering met de inductieve definitie. Hiervan zullen we verderop dankbaar gebruik maken.
2. Het toont aan dat de definities die allebei beogen het intuitieve begrip berekenbaarheid te formaliseren, precies eenzelfde klasse

van funkties definieren.

Natuurlijk kan men nooit bewijzen dat de part rec funkties precies de intuitief berekenbare funkties zijn, omdat het juist het wezenlijke van het adjectief "intuitief" is dat het niet geformaliseerd kan worden. Wél zou de gelijkwaardigheid weerlegd kunnen worden door van een intuitief berekenbare funktie aan te tonen dat die niet part rec is. Dat is tot heden toe niet gelukt. En mede op grond van de gelijkwaardigheid met nog andere formele definities die beogen datzelfde intuitieve begrip te karakteriseren, is men er in gaan geloven dat we hiermee de goede karakterisering hebben. Dit geloof wordt aangeduid met de

These van Church:

De intuitief berekenbare partiele funkties zijn precies de part rec funkties. Vergelijk deze met de These van Turing, (2.14).

Opmerking In bewijzen vindt men nogal eens het argument "op grond van de These van Church is ... part rec". Wat hiermee bedoeld wordt is duidelijker als we bij zo'n bewijs de titel bewijs vervangen door schets van het bewijs en genoemd argument vervangen door "aan de lezer wordt overgelaten (zo hij het niet gelooft) aan te tonen dat ... part rec is".

Rechtstreeks uit de gelijkwaardigheid konkluderen we de

9.10. Stelling

a. de T-berekenbare totale funkties zijn precies de rec funkties,
b. de part rec totale funkties zijn precies de rec funkties,
c. iedere part rec funktie kan gedefinieerd worden met behulp van de inductieve definitie met slechts éénmalig gebruik van de minimalisatie op prim rec funkties.

Bewijs In verkorte notatie,

a. f rec $\Rightarrow$ f part rec en totaal $\Rightarrow$ f T-berekenbaar en totaal;
fT-berekenbaar en totaal $\Rightarrow$ (in de notatie van (9.8))
$\forall w\ \exists t[\text{toes}(t,w) = n] \Rightarrow \mu t[\text{toes}(t,w) = n]$ is een reguliere minimalisatie op een prim rec relatie $\Rightarrow$ f rec.

b. f rec $\Rightarrow$ f part rec en totaal, duidelijk;
f part rec en totaal $\Rightarrow$ f T-berekenbaar en totaal $\Rightarrow$ f rec.

c. f part rec $\Rightarrow$ f T-berekenbaar $\Rightarrow$ (volgens het bewijs van de gelijkwaardigheidsstelling) f te definieren met slechts éénmalig gebruik van de μ operator op prim rec funkties. □

9.11. Net als bij de prim rec funkties kunnen we alle part rec funkties opsommen. We doen dit niet rechtstreeks aan de hand van de definitie van *PR*, want dan moeten we op de totaliteit van een funktie testen voordat we erop minimaliseren. En het is maar de vraag of we effectief kunnen testen of een funktie totaal is! Beter kunnen we van stelling (9.10)c. gebruik maken: geef tijdens de opsomming van de prim rec funkties (zie (7.25)) ook de minimalisatie op de opgesomde funkties.
Er zijn natuurlijk ook andere opsommingen mogelijk.

9.12. Voor het vervolg denken we ons zo'n effectieve opsomming vast gekozen. Een rangnummer van de opsomming zullen we index noemen. Iedere part rec funktie heeft dan een of meer indices en ieder getal is index van een part rec funktie. De opsomming duiden we aan met $\phi_0, \phi_1, \phi_2, \ldots$

Het is niet moeilijk een effectieve opsomming te geven van de n-plaatsige part rec funkties - voor vaste n - . We noteren deze opsomming met $\phi_0^{(n)}, \phi_1^{(n)}, \phi_0^{(n)}, \ldots$. Om typografische redenen zullen we voor n=1 het bovenschrift weglaten. Als we de opsomming van álle part rec funkties bedoelen, zeggen we dat er duidelijk bij.
Ook voor Turingmachines kunnen we een opsomming geven. $T_0, T_1, T_2, \ldots$ (oefening. Geef een schets van zo'n opsomming). Vanwege de effectiviteit kunnen we bij iedere index i van een part rec funktie - uit de totale opsomming - de index j van de realiserende Tm construeren: $\phi_i \approx T_j$, en omgekeerd.
De indices, d.w.z. het bestaan van een effectieve opsomming, spelen een fundamentale rol in de recursietheorie. Men kan ook bewijzen dat alle navolgende resultaten onafhankelijk zijn van de gekozen opsomming.

Twee reeds bewezen resultaten voor Tms kunnen we nu ook voor part rec funkties formuleren.

9.13. Stelling

a. Er is een n+1-plaatsige part rec funktie u die universeel is voor de n-plaatsige part rec funkties, d.w.z. $u(i,\vec{x}) = \phi_i^{(n)}(\vec{x})$ voor alle i, $\vec{x}$.

b. Er is geen part rec totale (dus rec) funktie f met de eigenschap

$$f(i,x) = \begin{cases} 0 & \text{als } \phi_i(x)\downarrow \\ 1 & \text{als } \phi_i(x)\uparrow. \end{cases}$$

Bewijs

a. (Vergelijk de uitspraak met het bestaan van een universele Tm). Geven i en $\vec{x}$, dan kunt U de i-de part rec funktie effectief opsommen: $\phi_i^{(n)}$. Dit verschaft een definitie van $\phi_i^{(n)}$ volgens I" a b c II" a b c in (9.3). Dus U kunt $\phi_i^{(n)}(\vec{x})$ effectief berekenen, zo die waarde bestaat. Op grond van de These van Church is dit proces om van i en $\vec{x}$ de eventuele waarde van $\phi_i^{(n)}(\vec{x})$ te bepalen een part rec funktie.

b. (Vergelijk dit met de onbeslisbaarheid van het stopprobleem voor Tms). Stel dat f wel rec is. Beschouw de rec funktie halt(i) = f(i,i). Definieer halt' door

$$\text{halt}'(x) = \begin{cases} 0 \text{ als halt}(x) = 1. \\ \uparrow \text{ als halt}(x) = 0. \end{cases}$$

Deze funktie halt' is part rec: $\text{halt}'(x) = Z(\mu y[\text{halt}(x)=y \wedge y>0])$, dus halt' heeft een index, zeg n: halt' = ϕ_n.
De beschouwing of $\text{halt}'(n) = \phi_n(n)$ al dan niet bepaald is, leidt tot een tegenspraak. Dus f kan niet rec zijn. □

9.14. De klasse *PR* der prim rec funkties bevat niet alle intuitief berekenbaar te noemen funkties. Dat hebben we met behulp van het beginsel van diagonalisatie laten zien: er werd een funktie gedefinieerd die ongelijk is aan alle prim rec funkties. Deze funktie zelf is wel berekenbaar; de opsomming van *PR* is immers effectief te geven. Bij de recursieve funkties luidt diagonalisatie uiteraard ook weer tot een funktie buiten de klasse *R*. Maar deze funktie is niet berekenbaar omdat de opsomming der rec funkties niet effectief te geven is.

De vraag rijst waartoe diagonalisatie op part rec funkties leidt. In ieder geval zal de funktie intuitief berekenbaar zijn, want de opsomming van de klasse P der part rec funkties is effectief. Dus zij $f_P(i) = \phi_i(i) + 1$. Is f_P dan ongelijk aan alle part rec funkties ? Neen, de veronderstelling dat f voorkomt, zeg als ϕ_n leidt niet tot een tegenspraak in

$$\phi_n(n) = f_P(n) = \phi_n(n) + 1$$

maar slechts tot de conclusie dat f_P op argumentwaarde n onbepaald is: $f_P(n)\uparrow$. In feite kunnen we ook bewijzen dat f_P een part rec funktie is: $f_P(i) = u(i,i) + 1$, waarin u de universele funktie is.
We kunnen nu proberen met behulp van diagonalisatie een totale funktie te definieren. Dan zal de veronderstelling dat f een (totale) part rec funktie is in ieder geval wel tot een tegenspraak leiden.

Oefening: Toon aan dat $f(i) = \begin{cases} \phi_i(i) + 1 & \text{als } \phi_i(i)\downarrow \\ 0 & \text{als } \phi_i(i)\uparrow \end{cases}$ | niet part rec is en ook niet rec.

Echter, deze f lijkt (is) helaas niet intuitief berekenbaar te noemen: hoe zou u kunnen beslissen of $\phi_i(i)\uparrow$?? We kunnen zélfs bewijzen dat $f_P(i) = \phi_i(i) + 1$ op geen enkele wijze tot een totale part rec, dus rec, funktie is uit te breiden.

Oefening: doe dit. Wenk: zij $f_P = \phi_n$. Zij ϕ_m een uitbreiding tot een totale rec funktie. Beschouw $\phi_n(m)$: tegenspraak.

Voor de liefhebbers is in het aanhangsel de afleiding opgenomen van twee fundamentele resultaten uit de recursietheorie. Het zijn de s-m-n stelling en de recursiestelling.
In de s-m-n stelling wordt niet alleen aangetoond dat een m+n-plaatsige part rec funktie met een m-tal vast genomen parameters een n-plaatsige part rec funktie is, maar ook dat de index van de laatste m.b.v. een recursieve funktie S uit de index van de eerste en het m-tal vaste parameters verkregen kan worden.
De recursiestelling staat ook wel bekend onder de wat betere naam van delepuntstelling. Er zijn allerlei verzwakkingen en versterkingen van de stelling mogelijk. Eén verzwakking is de volgende:

bij een definierende gelijkheid van de vorm

$$f(\vec{x}) = \ldots f \ldots f \ldots ,$$

waarin rechts de symbolen f genest nogen voorkomen zonder enige beperking op hun argument (en er verder alleen schema's zijn gebruikt waaronder de klasse der rec funkties gesloten is), wordt het bestaan van een part rec funktie f die aan de gelijkheid voldoet gegarandeerd en is zelfs een effectieve constructie voor f te geven.

(Oefening. De Ackermann funktie *A* is dus part rec. Ga na dat *A* totaal is. Dus *A* is recursief.
Oefening. Wat is de funktie die voldoet aan $f(x) = f(x) + 1$? Welke part rec funkties voldoen aan $f(x) = f(x+1) \dot{-} 1$, zijn ze rec ?).

9.16. We hebben nu de gelijkwaardigheid van Turingmachines [die altijd stoppen] met part rec funkties [die totaal zijn] behandeld en ook enkele fundamentele resultaten betreffende part rec funkties. En we geloven in de gelijkwaardigheid met berekenbare funkties [die totaal zijn]. Als uiteenzetting over berekenbare funkties lijkt ons het voorgaande voorlopig genoeg.
De in hoofdstuk 5 en 6 ingevoerde begrippen "beslisbare" en "opsombare" verzameling vragen om een nadere beschouwing.

10. Recursieve en recursief opsombare verzamelingen.

10.1. Beslisbaarheid en opsombaarheid zijn belangrijke begrippen in de (theoretische) informatika. Er kan bijvoorbeeld geen compiler geschreven worden voor een programmeertaal, als het niet beslisbaar is of willekeurige (programma) tekst syntactisch juist is. Een grammaticaal systeem is erg machtig als alle opsombare talen daarmee te genereren zijn, zoals het systeem dat gebruikt is voor de definitie voor Algol 68. Er kan geen programma geschreven worden dat beslist of willekeurig ALGOL programma zal termineren. Het is niet mogelijk in het algemeen te beslissen of twee ALGOL programma's gelijkwaardig zijn. Het is in ALGOL 60 op grond van de programmatekst niet beslisbaar of gedurend de uitvoering van het programma de aktuele parameters met hun type zullen overeenstemmen met de formele; by ALGOL 68 is dat wel mogelijk. Enzovoort.
We zullen in dit hoofdstuk de begrippen Beslisbaarheid en Opsombaarheid wat uitdiepen. We doen dat in het formalisme van recursieve funkties. Daarom beperken we de objecten tot getallen en deelverzamelingen der natuurlijke getallen (inclusief nul). Veralgemening tot anderssoortige objecten is mogelijk, maar wordt hier niet behandeld.

10.2. In het kader van recursieve funkties noemen we beslisbare en opsombare verzamelingen respectivelijk recursief en recursief opsombaar. Het Definitiegebied van een partiele funktie f is $D_f = \{\vec{x} | f(\vec{x}) \downarrow\} = \{\vec{x} | \exists y: f(\vec{x}) = y\}$. Het waarden-Bereik van een partiele funktie is $B_f = \{y | \exists \vec{x}: f(\vec{x}) = y\}$. Tot nadere aankondiging veronderstellen we steeds dat verzamelingen V deel zijn van de natuurlijke getallen: $V \subseteq N$.
Het woord verzameling wordt afgekort tot verz.

10.3. Definitie Een verz $V \subseteq N$ heet (primitief) recursief als zijn karakteristieke funktie f_V (primitief) recursief is. De funktie f_V wordt ook wel de beslissende funktie genoemd.
Een verz $V \subseteq N$ heet recursief opsombaar (afk. rec opsb) als $V = \emptyset$ of $V = B_f$ voor een of andere 1-plaatsige recursieve funktie. De funktie

f wordt ook wel opsommende of genererende funktie genoemd.

Ga na dat deze definitie in overeenstemming is met de definitie van (Turing) beslisbare en (Turing) opsombare verzameling. De genererende funktie somt op invoer 0,1,2,3,... achtereenvolgens - in een of andere volgorde - uiteindelijk alle elementen van V op.

10.4. Oefening

a. Ga na dat de volgende verz recursief zijn: de lege verz, de verz der nat. getallen, iedere eindige verz, de verz van alle veelvouden van een bepaald getal, de verz der priemgetallen. Ga na dat deze zelfs prim rec zijn.

b. Ga na dat de volgende verz rec opsb zijn: de verz der even getallen, iedere eindige verz, het complement van willekeurige eindige verz, de lege verz. Zijn ze (prim) rec ?

c. Zijn alle deelverzn der nat getallen rec ? rec opsb ? (denk aan aftelbaarheid en overaftelbaarheid).

10.5. De begrippen aftelbaarheid (Engels: denumerability) en opsombaarheid (Engels: enumerability) worden nogal eens verward. In feite zou men er dezelfde betekenis aan kunnen hechten, maar opsombaarheid wordt door ons (en in de literatuur) alleen gebruikt wanneer het (stilzwijgend vaak) voorzien is van het adjectief recursief of Turing-. En dán is het verschil tussen de twee begrippen dat de laatste ook een effectiviteit van de aftelling garandeert. Ten gevolge van de effectiviteit komen er in opsommingen gewoonlijk herhalingen voor. Bij opsommingen van natuurlijke getallen hoeft dat niet. Want gegeven een opsomming mét herhalingen, dan is het eenvoudig een opsomming te construeren zonder herhalingen: getallen die je al gehad hebt som je niet meer op. Zie (10.11) h. Bij opsommingen van funkties zoals de prim rec funkties en de part rec funkties zijn herhalingen onvermijdelijk. We zullen in (10.12) d. bewijzen dat het niet beslisbaar is of twee willekeurige funkties gelijk zijn. Dus kun je de ene niet zomaar weglaten als je al een ander hebt gehad die eraan gelijk is.
De volgende stelling geeft een aantal fundamentele eigenschappen van

rec opsb verzn. Ze geven veel vrijheid om de rec opsbheid van verzn aan te tonen. Tevens zijn er gelijkwaardige definities uit te halen voor het begrip recursieve opsombaarheid.

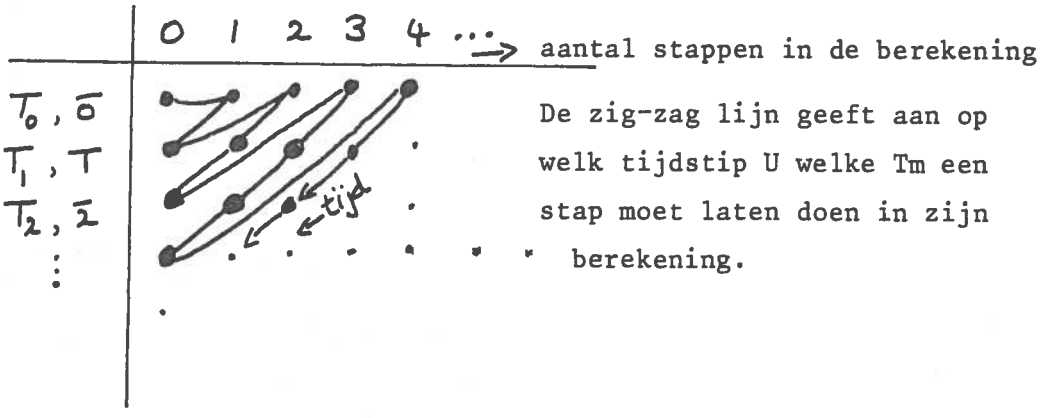
10.6. Stelling Zij $V \subseteq N$. De volgende uitspraken zijn gelijkwaardig,

a. $V = B_f$ voor zekere 1-plaatsige part rec funktie f,
b. $V = \emptyset$ of $V = B_f$ voor zekere 1-plaatsige rec funktie f,
c. $V = \emptyset$ of $V = B_f$ voor zekere 1-plaatsige prim rec funktie f,
d. V kan gedefinieerd worden door $x \in V \leftrightarrow_{def} \exists y R(x,y)$ voor zekere rec R,
e. $V = D_f$ voor zekere part rec funktie f (die dus 1-plaatsig is).

Bewijs We bewijzen hier a. $\Rightarrow$ b., b. $\Rightarrow$ d., d. $\Rightarrow$ e., e. $\Rightarrow$ a. Het bewijs van c. $\Leftrightarrow$ a. volgt later in (10.16).

a. $\Rightarrow$ b.. Als $V = \emptyset$, i.e. $f(x)\uparrow$ voor alle x, dan valt er weinig te bewijzen. Als V eindig is, zeg $V = \{x_o, x_1, \ldots, x_m\}$, dan is de funktie g gedefinieerd door $g(0) = x_o, \ldots, g(m) = x_m$, $g(n) = x_m$ voor $n > m$, een rec funktie die V opsomt, als V oneindig is, geldt de volgende redenering.

Onderstaand schema schetst een effectieve procedure voor een funktie h. Alle T_i zijn copieen van een Tm T die f realiseert ($V = B_f$). De beginbandexpressie voor T_i is $\overline{i}$.



U moet aan $h(0), h(1), \ldots$ de waarden $n_o, n_1, \ldots$ toekennen, indien $t_o, t_1, \ldots$ de opeenvolgende tijdstippen zijn waarop er Tm's stoppen met eindbandexpressie $\overline{n}_o, \overline{n}_1, \ldots$ resp.

Met enige inventiviteit kunt U deze effectieve procedure ter bepaling van een waarde voor h, door een Tm laten realiseren. Dus h is part rec.

Omdat $V = B_f = \{n \mid$ er is een Tm die z'n berekening stopt met eindbandexpressie $\overline{n}\}$ oneindig is, is h totaal, dus recursief.
En h somt V op.
b. $\Rightarrow$ d.. Als $V = \emptyset$ neem dan $R(x,y) \leftrightarrow 0 = 1$. Er geldt dan trivialerwijs dat $x \in V \leftrightarrow \exists\, yR(x,y)$.
Als $V = B_f$ met f rec, neem dan $R(x,y) \leftrightarrow f(y) = x$.
R is dan rec en er geldt $x \in V \leftrightarrow \exists\, y[R(x,y)]$.
d. $\Rightarrow$ e.. Zij V gedefinieerd door $x \in V \leftrightarrow \exists\, yR(x,y)$ met R rec.
Definieer f door $f(x) = \mu y[R(x,y)]$. Dan is f part rec en geldt
$f(x)\downarrow \Rightarrow$ er is een y zó dat $R(x,y) \Rightarrow \exists\, yR(x,y) \Rightarrow x \in V$, en
$x \in V \Rightarrow$ er is een y zó dat $R(x,y) \Rightarrow \mu yR(x,y)\downarrow \Rightarrow f(x)\downarrow$.
Dus $V = D_f$ met f part rec.
e. $\Rightarrow$ a.. Copieer het bewijs van a. $\Rightarrow$ b. met vervanging (tweemaal) van "$V = B_f$" door "$V = D_f$", en met vervanging (tweemaal) van "met eindbandexpressie" door "indien gestart met beginbandexpressie".

∎

We bewijzen nu de in (5.4) aangegeven eigenschappen.
Allereerst volgt de stelling met een recht-toe recht-aan bewijs, zoals in (5.4) geschetst in termen van Tms. In de oefeningen die er op volgen vindt U een aanwijzing waarmee U een korter bewijs van de stelling kunt geven.

10.7. Stelling

a. Een verz V is rec alsla zowel V als $\overline{V}$ rec opsb zijn,
b. Er is een rec opsb verz die niet rec is,
c. Er is een verz die niet rec opsb is.

Bewijs

a. Stel V rec.
Als V leeg of eindig is, is V rec opsb.
Als V oneindig is, en f_v is de rec karakteristieke funktie van V, dan is

$$\begin{cases} g(0) = x_o \\ g(n+1) = \mu x[x > g(n) \wedge f_v(x) = 0]. \end{cases}$$

Een rec funktie die V opsomt. (laat $x_o = \mu x[f_v(x) = 0]$ en zij

$R(x,y) \leftrightarrow x > y \wedge f_V(x) = 0$, dan is R rec, dus
$h(y) = \mu x[R(x,y)]$ is part rec en g kan gedefinieerd worden door

$\begin{cases} g(0) = x_o \\ g(n+1) = h(g(n)). \end{cases}$ Dus g is part rec, en omdat V oneindig is, is g totaal dus zelfs rec.) (V wordt zelfs monotoon opgesomd).

Analoog om te bewijzen dat $\overline{V}$ rec opsb is.

Nu de implicatie van rechts naar links.

Stel zowel V als $\overline{V}$ rec opsb.

Als een van beide leeg is, is het triviaal dat V rec is.

Stel dus beide niet leeg en zij g en $\overline{g}$ hun opsommende rec funkties.

Definieer een funktie h door

$h(n) = \begin{cases} g(n/2) & \text{als n is even} \\ \overline{g}(n-1/2) & \text{als n is oneven,} \end{cases}$

dan is h rec en somt zowel V als $\overline{V}$ op, dus heel N.

Dus er geldt $\forall x \; \exists \; n[h(n) = x]$ en daarom is de funktie f gedefinieerd door

$f(x) = \begin{cases} 0 & \text{als } \mu n[h(n) = x] \text{ is even} \\ 1 & \text{als } \mu n[h(n) = x] \text{ is oneven,} \end{cases}$

rec en is precies de karakteristieke funktie van V.

b. We construeren m.b.v. het diagonaal argument een rec opsb verz V zó dat $\overline{V}$ ongelijk is aan ieder waardenbereik van part rec funkties. Dan is dus volgens voorgaande stelling, (10.6), $\overline{V}$ niet rec opsb:

$n \in \overline{V}$	1	1	0	1			
n	0	1	2	3	4	5	....
$n \in B\phi_0$	0	*	*	*	*	*	
" $B\phi_1$	*	0	*	*	*		
" $B\phi_2$	*	*	1	*			
	*	*		0			

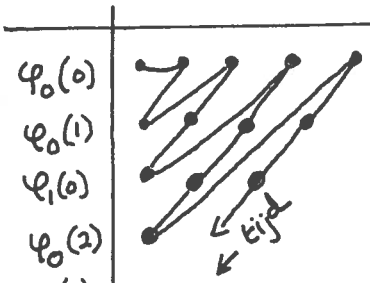
(De 0 duidt bevestiging aan, de 1 een ontkenning).

ofwel, per definitie is
$n \in \overline{V} \leftrightarrow n \not\in B\phi_n$

Per constructie is $\overline{V}$ ongelijk aan alle $B\phi_n$. (Formeel: stel $V = B\phi_e$ dan is wegens $n \in \overline{V} \leftrightarrow n \not\in B\phi_n$:

$e \in B\phi_e$ alsla $e \not\in B\phi_e$. Tegenspraak).

Om aan te tonen dat V wel rec opsb is geven we de volgende effectieve procedure:



hierdoor wordt de volgende funktie h gerealiseerd:
h(0), h(1)... heeft waarde $i_o, i_1, \ldots$
als $t_o, t_i \ldots$ de opeenvolgende tijdstippen zijn waarop de berekeningen van ϕ_{i_o}, ϕ_{i_1} ... stoppen met eindwaarde i_o, $i_1 \ldots$

Dus h is part rec en somt V op.
V is rec opsb, maar kan volgens(i) niet rec zijn.

c. Zie de constructie van $\overline{V}$ hierboven. ☒

10.8. Oefening Bewijs "V rec $\Longrightarrow$ V,$\overline{V}$ rec opsb" als volgt.
Definieer $f(x) = \begin{cases} 0 \text{ als } f_V(x) = 0 \\ \uparrow \text{ als } f_V(x) = 1 \end{cases}$, waarin f_V de kar. funktie van V is.
Concludeer $V = D_f$. Analoog voor $\overline{V}$.

10.9. Oefening (Stopprobleem, I)

Definieer een verz K door $x \in K \leftrightarrow x \in D_{\phi_x}$. Dat wil zeggen

$K = \{x | x \in D_{\phi_x}\} = \{x | \phi_x(x) \downarrow\}$.

a. Laat zien dat K rec opsb is. Wenk: beschouw de diagonalisatie f_P op P (9.14), nl $f_P(x) = \phi_x(x) = U(x,x)$.
Konkludeer: f_p is part rec en $K = D_{f_P}$.

b. Laat zien dat K niet rec is. Wenk: stel van wel, beschouw

$f(x) = \begin{cases} \phi_x(x) + 1 & \text{als } \phi_x(x)\downarrow \\ 0 & \text{anders} \end{cases}$, zie (9.14) en geef m.b.v. de karakteristieke funktie van K een definitie van f waaruit zou blijken dat f rec is. Tegenspraak.

c. Laat zien dat K niet rec is op de volgende manier.
Stel van wel, dan is K rec opsb, zeg $K = D_{\phi_e}$.

Beschouw de gevallen $e \in K$ en $e \in \overline{K}$. Tegenspraak.

d. Geef in een plaatje het diagonalisatie-idee van de definitie van K en van het bewijs in b. en c.

e. Ga na hoe de relatie van K met het stopprobleem is.

f. Vergelijk dit resultaat met dat van (6.2):
het stopprobleem is niet oplosbaar (i.e. niet beslisbaar).

10.10. Oefening (Stopprobleem, II)

Definieer een verz K^* door $z \in K^* \leftrightarrow_{def} \exists x,y<z[z=<x,y> \wedge y \in D_{\phi_x}]$.

D.w.z. $K^* = \{<x,y> | y \in D_{\phi_x}\} = \{<x,y> | \phi_x(y)\downarrow\}$.

a. - f.: als bij de vorige oefening.

De verzn K en K^* spelen een grote rol in de recursietheorie omdat ze een standaard voorbeeld zijn van een rec opsb maar niet rec verz, i.e. van een wel opsombaar maar niet beslisbaar probleem. (Vgl. (6.4)).

10.11. Oefening Bewijs of weerleg de volgende implicaties.

a. V rec en $V' \subseteq V \Rightarrow V'$ rec,
b. V rec opsb en oneindig $\Rightarrow$ er is een oneindige rec $V' \subseteq V$,
c. V rec en $V' \subseteq V \Rightarrow V'$ rec opsb,
d. V rec opsb $\Rightarrow \overline{V}$ rec opsb,
e. V niet rec opsb $\Rightarrow \overline{V}$ eindig,
f. I rec opsb en $\forall i \in I[V_i$ rec opsb$] \Rightarrow \bigcup_{i \in I} V_i$ rec opsb,
g. I willekeurig, $\forall i \in I[V_i$ rec opsb$] \Rightarrow \bigcup_{i \in I} V_i$ rec opsb,
h. V rec opsb en oneindig $\Rightarrow V = B_f$ voor rec f die injectief en stijgend is.

10.12. We geven nu nog enige toepassingen van de begrippen rec en rec opsb verzameling en van het ontwikkelde gereedschap. Beschouw een programmeertaal waarin elke Turingmachine gesimuleerd zou kunnen worden - daar voldoet bijna iedere programmeertaal aan - . De volgende vragen zijn wellicht van belang.

a. Is er een (altijd stoppende) algoritme die op invoer "programma x" een antwoord geeft op de vraag of programma x altijd termineert ?

b. Is er een (altijd stoppende) algoritme die op invoer "programma x" en "data y" een antwoord geeft op de vraag of programma x op data y termineert ?

c. Is er een (altijd stoppende) algoritme die op invoer "programma x" en "getal y" een antwoord geeft op de vraag of programma x resultaat y oplevert ?

d. Is er een (altijd stoppende) algoritme die op invoer "programma x" en "programma y" een antwoord geeft op de vraag of programma x gelijkwaardig is met programma y ?

Omdat elke Tm gesimuleerd kan worden in de programmeertaal, kunnen we zeggen dat iedere part rec funktie een programma is. Uiteraard (These van Church of Turing) is ieder programma - mits invoer en uitvoer getallen zijn - een part rec funktie. Bovenstaande problemen krijgen dan de volgende formulering:

Is er een (altijd stoppende ?) algoritme die

a. op invoer x beantwoordt of ϕ_x totaal is,

b. " " x,y " " $\phi_x(y)\downarrow$,

c. " " x,y " " $y \in B_{\phi_x}$,

d. " " x,y " " $\phi_x = \phi_y$.

En deze formuleringen kunnen we ook schrijven als

a. is $\{x|\phi_x \text{ is totaal}\}$ een rec (rec opsb) verz ?

b. " $\{z|\ \exists\, x,y<z : z = \langle x,y\rangle \text{ en } \phi_x(y)\downarrow\}$ een rec (rec opsb) verz ?

c. " $\{z|\ \exists\, x,y<z : z = \langle x,y\rangle \text{ en } y \in B_{\phi_x}\}$ " " " " ?

d. " $\{z|\ \exists\, x,y<z : z = \langle x,y\rangle \text{ en } \phi_x=\phi_y\}$ " " " " ?

Merk op dat a. juist de vraag is of de totale part rec, dus recursieve

funkties opsombaar zijn, zie (8.7) en (9.11) en (9.14). Vraag 6. is juist het algemeen stopprobleem, zie (9.13) en oefeningen in (9.14) en verz K^* in (10.10). Een ontkennend antwoord op vraag d. is aangekondigd in (10.15): de onvermijdelijkheid van herhalingen van gelijke (= gelijkwaardige) funkties in de opsomming van de part rec funkties.

De antwoorden op de vraagstellingen a - d zijn als volgt.

a. $\{x \mid \phi_x \text{ totaal}\}$ is niet rec en niet rec opsb,
b. $\{\langle x,y\rangle \mid \phi_x(y)\downarrow\}$ " niet " " wel " " ,
c. $\{\langle x,y\rangle \mid y \in B_{\phi_x}\}$ " niet " " wel " " ,
d. $\{\langle x,y\rangle \mid \phi_x = \phi_y\}$ " niet " " niet " " .

10.13. De bewijzen van bovenstaande beweringen gaan als volgt.

a. Stel $V = \{x \mid \phi_x \text{ totaal}\}$ wél rec opsb, zeg $V = B_f$ met f rec, ofwel $V = \{\phi_{f(o)}, \phi_{f(1)}, \phi_{f(2)}, \ldots\}$. We gaan diagonaliseren:

definieer $h(x) = \phi_{f(x)}(x) + 1$, dus $h(x) = u(f(x),x) + 1$, dan is h part rec en totaal dus rec dus in V, zeg $h = \phi_{f(n)}$.

Nu geeft $\phi_{f(n)}(f(n)) = h(f(n)) = \phi_{f(n)}(f(n)) + 1$ een tegenspraak.

b. Zie oefening (10.10).

c. Zeg $V = \{\langle x,y\rangle \mid y \in B_{\phi x}\}$ wel rec, dan is ook $V' = \{x \mid x \in B_{\phi x}\}$ rec, in strijd met het bewijs van (10.9)b. Dat V wel rec opsb is, wordt aangetoond met een soortgelijke redenering als in (10.9)b.

d. Stel $\{\langle x,y\rangle \mid \phi_x = \phi_y\}$ wel rec opsb, dan is ook voor vaste x_o $\{y \mid \phi_{xo} = \phi_y\}$ rec opsb en ook $\{y \mid \phi_{xo} = Z.\phi_y\}$. Kies x_o zó dat $\phi_{xo} = Z$. Dan is $\phi_{xo} = Z.\phi_y$ alsla ϕ_y totaal is. De rec opsbaarheid is in strijd met a. □

We sluiten hiermee de beschouwingen over deelverzamelingen $V \subseteq N$ af. Voor de rest van dit hoofdstuk willen we de begrippen recursiviteit en recursieve opsombaarheid nog definieren voor $V \subseteq Nx....xN = N^n$, en enkele feiten noemen.

10.14. Definitie a. Een verzameling $V \subseteq N^n$ heet recursief als zijn karakteristieke funktie recursief is.
b. Een verzameling $V \subseteq N^n$ heet recursief opsombaar als $V = D_f$ voor zekere part rec funktie f (die dus n-plaatsig is).
Merk op dat deze definitie voor n = 1 in geval van recursiviteit precies overeenkomt met (10.3) en in geval van rec opsombaarheid volgens stelling (10.6) gelijkwaardig is met de definitie (10.3).

10.15. Deelverzamelingen $R \subseteq N^n$ kunnen ook als n-plaatsige relaties R worden opgevat: we schrijven dan $R(x_1,\ldots,x_n)$ i.p.v. $(x_1,\ldots,x_n) \in R$. Bijvoorbeeld, voor n = 1, zij PRIEM = $\{x \mid 1<x$ en x is alleen deelbaar door 1 en zichzelf$\}$, dan kunnen we schrijven $5 \in$ PRIEM of PRIEM(5). Een ander voorbeeld is DB = $\{(x,y) \mid x$ deelt $y\}$, we kunnen schrijven $(25,625) \in$ DB maar ook DB(25,625).
Omgekeerd kunnen relaties ook als verzamelingen worden opgevat. Zij $EQ(x,y) \leftrightarrow x=y$, dan kunnen we in plaats van $\neg EQ(17,18)$ ook schrijven niet $(17,18) \in EQ$, ofwel $(17,18) \notin EQ$.
Merk op dat de definitie voor recursiviteit van een verzameling $V \subseteq N^n$ (19,14) precies overeenstemt met de recursiviteit van V beschouwd als een relatie, (8.4), en omgekeerd. Nu kunnen we ook definieren wanneer een relatie recursief opsombaar is: R is rec opsb wanneer $R = D_f$ ofwel wanneer $R(\vec{x}) \leftrightarrow \vec{x} \in D_f$, voor een of andere part rec funktie f.

10.16. Het T-predikaat (predikaat is een ander woord voor relatie) van Kleene is in de literatuur een bekend begrip. De letter T staat voor Turing. Het T-predikaat wordt gedefinieerd door $T(z,\vec{x},y) \leftrightarrow$ y is het gödelnummer van een berekening van de Tm (ofwel de part rec funktie) met index z op argument $\vec{x}$.
Er kan bewezen worden dat het T-predikaat een prim rec relatie is. Dit is als volgt in te zien. Gegeven $z,\vec{x}$, en y, decodeer eerst y en simulier dan de Tm (part rec funktie) met index z op argument $\vec{x}$. Kijk voortdurend of deze berekening de decodering van y volgt en zo ja

geef dan aan het eind gekomen, een 0 als resultaat; zo nee, geef dan een 1 als resultaat. Dit is duidelijk een effectieve procedure en dus met een beroep op de These van Church een recursieve funktie. Die blijkt zelfs prim rec te zijn. Dus de karakteristieke funktie van T is een prim rec funktie: T is prim rec.

Het T-predikaat speelt een grote rol. $\exists$ y $T(z,\vec{x},y)$ is een relatie tussen z en $\vec{x}$. $\exists$ y $T(z,\vec{x},y)$ betekent: er is een terminerende (!) berekening van ϕ_z op $\vec{x}$, ofwel $\phi_z(\vec{x})\downarrow$. Precies wanneer $\exists$ y $T(z,\vec{x},y)$ geldig is, is $\mu y[T(z,x,y)]$ bepaald en in dat geval gelijk aan het kleinste (en zelfs enige) gödelnummer van een (de) terminerende berekening van ϕ_z op $\vec{x}$. Kiezen we een geschikte gödelnummering, dan kunnen we met een prim rec funktie, zeg U, de eindwaarde van de berekening uit het gödelnummer verkrijgen. Alsdus hebben we de Normaalvormstelling van Kleene: er is een prim rec funktie U en een prim rec predikaat T zó dat er voor willekeurige part rec functie f een z is (de index van f) waarvoor geldt $f(\vec{x}) = U(\mu y[T(z,\vec{x},y)])$ voor alle $\vec{x}$.

We zijn deze stelling als eens tegengekomen in een bijzonder geval: de Ackermann funktie. Vergelijk maar eens met hetgeen in (8.6) is afgeleid: $A(m,n) = U(\mu y[T_A(m,n,y)])$!! Bovendien laat de Normaalvormstelling van Kleene nog eens zien wat we al eerder bewezen hebben: iedere part rec funktie kan gedefinieerd worden met een slechts éénmalig gebruik van de minimalisatie, en wel op prim rec funkties.

Met het T-predikaat is het bewijs van a. $\Rightarrow$ c. van stelling (10.16) gemakkelijk te geven. Zij immers $V \neq \emptyset$ en, zeg $v_o \in V = B_h$ met h part rec, zeg met index z. De volgende funktie is prim rec en somt V op:

$$f(x) = \begin{cases} U((X)_2) & \text{als } x = \langle (x)_1,(x)_2\rangle \wedge T(z,(x)_1,(x)_2) \\ V_o & \text{anders} \end{cases}$$

(Het omgekeerde, c. $\Rightarrow$ a. is triviaal).

11. Markovalgoritmen.

11.1. Het begrip berekenbaarheid van funkties kan behalve via Turingmachines en via partieel recursieve funkties ook worden benaderd via Markovalgoritmen of "normale algoritmen" zoals ze genoemd worden in Markov (1951).
Intuitief gezien is een Markovalgoritme een algoritme voor het omzetten van woorden over een gegeven alfabet A in woorden over A. Een Markovalgoritme wordt gegeven als een soort programma dat door een primitieve computer kan worden geinterpreteerd. Zo'n programma bestaat uit een rij van zgn. (Markov) producties, die elk een omzetting van symbolen aangeven, en de interpretatie bestaat uit het op een bepaalde manier doorlopen en uitvoeren van deze producties.

11.2. Definitie Een Markovalgoritme M over een alfabet A is een drietal, M = (A, H, P) waarvoor geldt:

1) A is een alfabet, A heet het alfabet van M en de elementen van A heten de letters van M.

2) H = $\emptyset$ of H is een alfabet, H heet het hulpalfabet van M, de elementen van H heten de hulpletters van M.

3) $A \cap H = \emptyset$ en $A \cup H = V$

4) P is een lineair geordende verzameling van producties van M. Een productie is een symboolrij van de vorm $\alpha \to \beta$ of van de vorm $\alpha \to. \beta$ met $\alpha, \beta \in V^*$ en $\to, \to. \notin V$. Een productie van de vorm $\alpha \to. \beta$ heet eindproductie.

Voorbeeld Bekijk de Markovalgoritme M = (A,H,P) met A = {a,b} en H = {h} en

P = (ha → ah
hb → bh
h →. bb
ε → haa) (N.B. ε is het lege woord)

M heeft vier producties waarvan er één eindproductie is. De volgorde

van de produkties is hier benadrukt door ze onder elkaar op een rij te schrijven in plaats van naast elkaar met komma's ertussen. We zullen dit steeds doen.

11.3. De werking van een Markovalgoritme, dus de interpretatie ervan, verloopt door het toepassen van de produkties. De manier waarop dit gebeurt gaan we nu informeel beschrijven. Een produktie $\alpha \to \beta$ of $\alpha \to .\ \beta$ is <u>van toepassing</u> op een woord $\gamma \in V^*$ indien α als deelrij voorkomt in γ, i.e. indien $\gamma = \gamma_1\alpha\gamma_2$ met $\gamma_1\gamma_2 \in V^*$.
Het <u>toepassen</u> van een produktie $\alpha \to \beta$ of $\alpha \to .\ \beta$ op een woord $\gamma = \gamma_1\alpha\gamma_2$ is het vervangen van de <u>linker</u> deelrij α in γ door β. Het resultaat van dit toepassen is het woord $\gamma_1\beta\gamma_2$ (mits α niet in γ_1 voorkomt!). Bij het toepassen van een produktie $\varepsilon \to \beta$ is het van belang te bedenken dat ε in elk woord γ links van de linker letter van γ voorkomt, immers $\varepsilon\gamma = \gamma$. Zo'n produktie is dus altijd van toepassing en het resultaat is $\beta\gamma$.
De werking van Markovalgoritme $M = (A,H,P)$ op een woord $w \in V^*$ bestaat uit een rij stappen. In elke stap wordt de rij produkties P in de volgorde waarin ze gegeven is doorzocht om de eerste te vinden die van toepassing is op w. Veronderstel dat dit produktie $p_i \in P$ is. Nu wordt p_i toegepast op w met als resultaat het woord $w' \in V^*$. Als p_i een eindproduktie is stopt de algoritme en anders wordt P opnieuw doorzocht vanaf het begin om de eerste produktie te vinden die van toepassing is op w'. Als geen produktie die van toepassing is wordt gevonden stopt de algoritme ook.
Het woord $w \in V^*$ waarop een Markovalgoritme M begint te werken heet het <u>invoerwoord</u> en als M met invoerwoord w stopt nadat w is omgezet in $w' \in V^*$ dan heet w' het <u>uitvoerwoord</u>.

<u>Voorbeeld</u> We bekijken de Markovalgoritme van voorbeeld (11.2) Bij invoerwoord ab is de eerste produktie die van toepassing is $\varepsilon \to$ haa en door toepassen hiervan ontstaat haaab. We geven dit aan door ab $\vdash$ haaab te schrijven. Vervolgens is ha $\to$ ah als eerste van toepassing zodat haaab $\vdash$ ahaab. Zo verder gaand krijgen wij na nog drie stappen aaabh en nu is als eerste de eindproduktie h $\to .$ bb van

toepassing zodat aaabh $\vdash$ aaabbb in de algoritme stopt. Het uitvoerwoord van M met invoerwoord ab is dus a^3b^3. Ga zelf na dat M elk invoerwoord $w \in A^*$ omzet in uitvoerwoord aawbb.

11.4. Afspraak over notatie Zij M = (A,H,P) en Markovalgoritme en zij $w \in V^*$. Als M invoerwoord w door het toepassen van één produktie omzet in $w' \in V^*$ dan schrijven we $w \vdash w'$ en als daarbij een eindproduktie wordt toegepast schrijven we $w \vdash .w'$. Als M invoerwoord w door het toepassen van nul, één of meer produkties omzet in $w' \in V^*$ dan schrijven we $w \vdash^* w'$ en als daarbij de laatst toegepaste produktie een eindproduktie is dan schrijven we $w \vdash^* .w'$.
In voorbeeld (11.2) is dus ahaab $\vdash$ aahab en aaabh $\vdash$ .aaabbb en ab $\vdash^*$ aaahb en ab $\vdash^*$ ab en ab $\vdash^*$ .aaabbb.

11.5. We hebben tot nu toe Markovalgoritmen gezien als een manier om invoerwoorden om te zetten in uitvoerwoorden. Daarbij is het onderscheid tussen letters en hulpletters niet gebruikt maar het speelt wel een rol in de volgende definitie.

Definitie a) Zij M = (A,H,P) een Markovalgoritme. M berekent de partiele funktie $f_M: A^* \to A^*$ die voor elke $w \in A^*$ bepaald is door:

$$f_M(w) = \begin{cases} w' \text{ als M met invoerwoord w stopt met w' als uitvoerwoord en } w' \in A^* \\ \text{ongedefinieerd als M met invoerwoord w niet stopt of stopt met w' als uitvoerwoord en } w' \notin A^* \end{cases}$$

b) Zij $f: V \to W$ een partiele funktie. f is Markov-berekenbaar als er een Markovalgoritme M met alfabet A is en een codering $\gamma: V \to A^*$ en een decodering $\delta: A^* \to W$ zodanig dat voor elke $v \in V$ geldt:

$\delta(f_M(\gamma(v))) = f(v)$ (dus $\delta f_M \gamma$ en f hebben dezelfde definitie-verzameling); m.a.w. zodanig dat het volgende diagram commuteert:

$$\begin{array}{ccc} V & \xrightarrow{f} & W \\ \gamma\downarrow & & \uparrow\delta \\ A^* & \xrightarrow{f_M} & A^* \end{array}$$

11.6. In voorbeeld (11.3) zagen we al dat de Markovalgoritme van voorbeeld (11.2) de (totale) funktie f_M berekent die bepaald is door $f_M(w) = a^2wb^2$ voor elke $w \in A^*$.
We geven nu nog enkele voorbeelden.

1. $M = (\{a,b,c\}, \emptyset, (\varepsilon \to .c))$.
   Ga na dat $f_M(w) = cw$ voor elke $w \in A^*$.

2. $M = (\{a,b,c\}, \emptyset, (\varepsilon \to c))$.
   Ga na dat $f_M(w)$ ongedefinieerd is voor elke $w \in A^*$.

3. $M = (\{1,\perp\}, \emptyset, P)$ met
   $P = (\perp \to .\varepsilon$
   $1 \to 1)$
   Ga na dat M elk invoerwoord van de vorm $\underbrace{11..1}_{n} \perp \underbrace{11..1}_{m}$ omzet in uitvoerwoord $\underbrace{11..1}_{n+m}$, dus M telt twee
   getallen op als ze gecodeerd en gedecodeerd worden als voor een Turingmachine.

4. $M = (A,H,P)$ met $A = \{a_1,a_2,\ldots,a_n\}$ en $H = \{h\}$ en
   $P = (ha_1 \to a_1h$
   $ha_2 \to a_2h$
   $\vdots$
   $ha_n \to a_nh$
   $h \to .a_1$
   $\varepsilon \to h)$
   Ga na dat $f_M(w) = wa_1$ voor elke $w \in A^*$.
   Ga na wat f_M is als de produktie $\varepsilon \to h$ niet op de laatste plaats staat.

   N.B. In voorbeeld 4 kan de beschrijving van de Markovalgoritme vereenvoudigd worden door de eerste n produkties te vervangen door het produktieschema: $ha \to ah$ voor elke $a \in A$. Zo'n vereenvoudiging kan alleen worden gebruikt als het schema een eindig aantal produkties veergeeft waarvan de volgorde onbelangrijk is.

5. $M = (\{a,b\},\{p,q,r\},P)$ met

$P = (xyq \to yqx$ voor alle $x,y \in A = \{a,b\}$

$px \to xqxp$ voor elke $x \in A$

$q \to r$

$r \to \varepsilon$

$p \to . \varepsilon$

$\varepsilon \to p)$. Ga na dat $f_M(w) = ww$ voor elke $w \in A^*$.

6. $M = (A,\{h_1,h_2\},P)$ met $A = \{a_1,a_2,\ldots,a_n\}$ en

$P = (h_1h_1 \to h_2$

$h_2a \to ah_2$ voor elke $a \in A$

$h_2h_1 \to h_2$

$h_2 \to . \varepsilon$

$h_1ab \to bh_1a$ voor alle $a,b \in A$

$\varepsilon \to h_1)$

Ga na dat $f_M(w) = w^S$ voor elke $w \in A^*$ waarbij w^S het spiegelbeeld van w is, d.w.z. als $w = x_1x_2\ldots x_k$ met $x_i \in A$ dan is $w^S = x_kx_{k-1}\ldots x_2x_1$.

11.7. We bekijken nu de gelijkwaardigheid van Turingmachines en Markovalgoritmen en daarmee die van Turing- en Markov-berekenbaarheid van funkties.

Stelling Bij elke Turingmachine T is een Markovalgoritme M te construeren zó dat $f_M = f_T$.

Bewijs Zij gegeven een Tm $T = (A,Q,\tau,q_o)$.

We construeren de Markovalgoritme M over A zó dat M een beginband-expressie $w \in A^*$ eerst omzet in $\#q_ow\#$, vervolgens de werking van T nabootst door configuraties om te zetten in configuraties net zoals T dat doet en tenslotte alle overbodige letters met de eindconfiguratie weghaalt.

Hiertoe nemen we $M = (A,H,P)$ met $H = Q \cup \{h,\#,s,t\}$ en

$P = (ha \to ah$ voor elke $a \in A$ p1

$h \to \#$ p2

als qaq'a'R vijftal van T is	$qab \to a'q'b$ voor elke $b \in A$	p3
	$qa\# \to a'q'\sqcup\#$	p4
als qaq'a'L vijftal van T is	$bqa \to q'ba'$ voor elke $b \in A$	p5
	$\#qa \to \#q'\sqcup a'$	p6
Als geen vijftal van T met qa begint	$qa \to sa$	p7
	$as \to sa$ voor elke $a \in A$	p8
	$\#s \to t$	p9
	$ta \to at$ voor elke $a \in A$	p10
	$t\# \to .\ \varepsilon$	p11
	$\varepsilon \to \#q_0 h)$	p12

We hebben de namen p1,...,p12 aan de produkties en produktieschema's gegeven om ze gemakkelijk te kunnen aanduiden.
Het is direct in te zien dat voor elk invoerwoord $w \in A^*$ eerst produktie p12 van toepassing is en daarna p1 en dan p2 zodat $w \vdash^* \#q_0 w \#$
Vervolgens zijn p3, p4, p5 en p6 van toepassing en wordt $\#\, q_0 w \#$ omgezet zoals $q_0 w$ door Tm T wordt omgezet. Als T met beginconfiguratie $q_0 w$ niet stopt doet M met invoerwoord w het ook niet en als T stopt met eindconfiguratie $w_1 q w_2$ dan geldt $w \vdash^* \# w_1 q w_2 \#$.
Hierop is p7 van toepassing en wordt de letter q vervangen door s. Vervolgens zijn p8, p9, p10 en p11 van toepassing en stopt M met $w_1 w_2$ als uitvoerwoord. Uit deze argumentatie volgt $f_M = f_T$.

11.8. Oefeningen 1. Construeer op bovenstaande manier de Markovalgoritme bij de Tm van (2.3).
2. Construeer de Markovalgoritme bij de Tm van oefening 1 in (3.10) en vergelijk het resultaat met de Markovalgoritme van voorbeeld 3 in (11.6).

11.9. Uit stelling (11.7) volgt direct dat een universele Markovalgoritme bestaat en ook geconstrueerd kan worden met de universele Turingmachine van hoofdstuk 4. Voor een directe constructie van een universele Markovalgoritme verwijzen we naar Markov (1954).

Het omgekeerde van stelling (11.7) is:

Stelling Bij elke Markovalgoritme M is een Turingmachine T te construeren zó dat $f_T = f_M$.

We bewijzen deze stelling hier niet maar verwijzen daarvoor naar Asser (1959) of Cernjavskie (1959) of Kurki-Suonio (1971), van deze drie is de constructie in Kurki-Suonio verreweg het eenvoudigste. Combineren we de stellingen (11.7) en (11.9) met de definities (2.13) en (11.5)b dan krijgen we
Stelling Zij f: V → W een partiele funktie. f is Turingberekenbaar als en alleen als f Markov-berekenbaar is.

11.10. We bekijken nu het verband tussen Markovalgoritmen en partieel recursievefunkties aan de hand van de volgende stellingen:

Stelling Bij elke partieel recursieve funktie f is een Markov-algoritme M te construeren, zodanig dat $f_M = f$.

Stelling Bij elk Markovalgoritme M kunnen we effectief een partieel recursieve funktie f vinden, zodanig dat $f = f_M$.

Deze twee stellingen zijn bewezen door Detlovs (1958). Samen met definitie (11.5b) leveren ze de volgende

Stelling Zij f: V → W een partiele funktie. f is partieel recursief als en alleen als f Markov-berekenbaar is.

11.11. We kunnen tenslotte concluderen dat alle drie tot dusver besproken formaliseringen van het begrip berekenbaar gelijkwaardig zijn. We geven dit aan met het volgende diagram: