kaleidoscoop van termherschrijfsystemen

Algemeen.

H1. Woorden.

- Zantema's puzzle.
- open problem tag system
- cellulaire automaten
- 2-dimensionale cell. automaten; conway's life.
- vincent's knikkers en kommen als exercise
- knopen, reidemeister moves
- retoucheren
- voorbeeld met overlap uit string rewriting book, otto-book

H2. Termen.

- menu's alsvoorbeeld van term
- 2 x 2
- poolse notatie
- shoenfield'sl emma, haakjes etc. niet nodig.
- applicatieve notatie
    - CL
    - S-termen
    - J-termen
    - Stenlund bases
- oneindige termen: eratosthenes
- ackerman functie, applicatief en functioneel
- Morse sequence

H3. Confluentie.

- unieke normaalvormen; un^= en un^->

    puzzel vincent
- CR is modulair
- infinitair: non cr-inf

H4. Abstract rewriting, met indices.

- balanced rewriting. behandeling van kommen en knikkers
- Newman's lemma.

• Diagram van relaties tussen cr, sn, un, wn, etc.
• braids
• decreasing diagrams

H5. Orthogonaal
• CR, PML
• EL
• modulaire SN for OTRs
• WN => AC
• UN-inf

H6. Terminatie.
• SN onbeslisbaar
• IPO
• Buchholz
• multisets, Konig's Lemma
• ordinalen om multiset terminatie te bewijzen
• (cantor normaalvorm); ook voor infinitair herschrijven

H7. Kritieke paren
• huet's lemma
• weakly orthogonal
• completion
• voorbeelden levy trs e.a.

H8. Strategieen

H9. Conditioneel herschrijven.

H10. CRSen

1

WOORDEN

1.1. *Woorden herschrijven*. Als eerste voorbeeld ter introductie bekijken we het herschrijven van eindige woorden, opgebouwd uit alleen de letters a en b. Voorbeelden van zulke a, b-woorden zijn a, b, ab, bab, abaab, bbbbbb, ... Een oneindig woord zoals abababab... (ad infinitum) is op dit moment uitgesloten, maar later zullen we ook oneindige woorden bekijken. In dit voorbeeld bekijken we de volgende vier *woord-herschrijfregels* (Tabel 1.1).

ab → bbba
ba → a
aa → bbbb
bb → b

Tabel 1.1

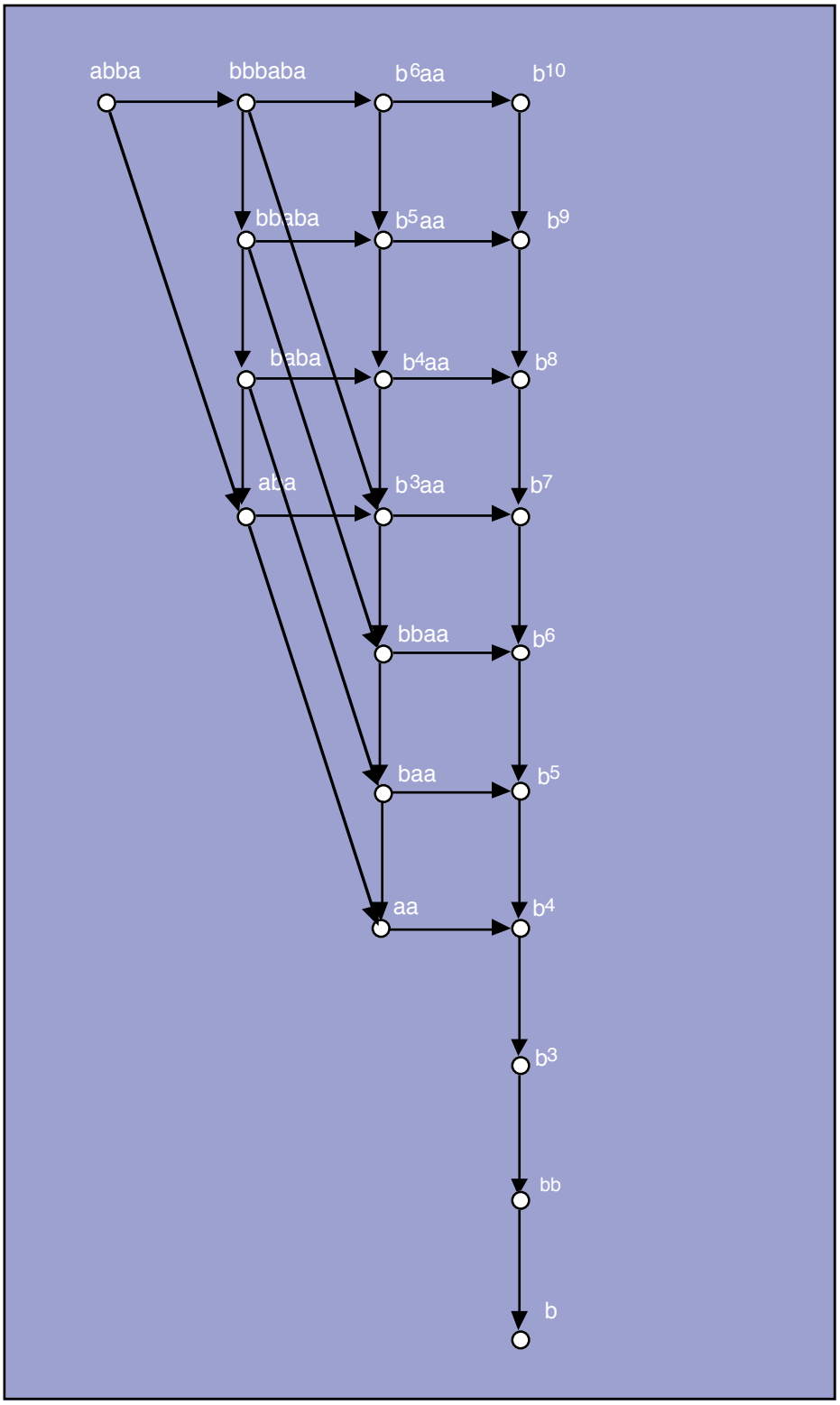
Hierbij hebben we de notatie → gebruikt, met de bedoeling dat in een gegeven woord de linkerkant vervangen mag woorden door de rechterkant van die regel. We zijn vrij om te kiezen welk deel (ook subwoord genoemd) van het woord we herschrijven, of welke regel we daarbij gebruiken. Zo kan bijvoorbeeld het woord abba op de volgende drie manieren één stap herschreven worden.

abba → bbbaba
abba → aba
abba → aba

1.1.1. OPGAVE. Is het mogelijk een woord met deze herschrijfregels zowel tot het woord a als tot b te herschrijven?
Hint: Merk op dat het aantal a's in alle vier regels hetzelfde blijft, of 2 minder wordt.

Zo'n stap noemen we een *herschrijfstap*. In een andere terminologie wordt herschrijven ook wel *reduceren* genoemd, en een herschrijfstap een *reductiestap*. Het subwoord dat herschreven wordt, heet ook een '*redex*' (afkorting van reducible expression), en het einde van een herschrijfstap heet een *reduct* van het begin van die stap. Als we herschrijfstappen achter elkaar zetten (concateneren), ontstaat een *herschrijfrij*. Bijvoorbeeld: ab → bbba → ba → a is een herschrijfrij van drie stappen.

Herschrijfrijen kunnen in principe ook oneindig lang zijn. *Vraag:* kan dat bij het nu beschouwde



Figuur 1.1

woord-herschrijfsysteem? (Een woord-herschrijfsysteem heet in de literatuur ook 'string rewrite system', of SRS.)

Vaak zijn we geinteresseerd in de verzameling van *alle* reducten van een woord, bij gegeven SRS. Als we daarbij ook de onderlinge structuur wat betreft $\rightarrow$ aangeven, hebben we de *'reductiegraaf'* van dat woord. Zo ziet de reductiegraaf van het woord abba er als volgt uit. (Figuur 1.1.) We zien dat alle herschrijfrijen vanuit abba eindig zijn, en ook dat ze allemaal in hetzelfde woord uitkomen. In de figuur hebben we exponenten als afkorting gebruikt, dus bijvoorbeeld b^6a^2 staat voor bbbbbbaa.

Nog een belangrijk begrip is dat van een *normaalvorm*. Een normaalvorm is een woord dat niet verder herschreven kan worden. In dit voorbeeld zijn er maar twee normaalvormen; welke?

1.2. *Zantema's puzzle*. Hoewel eindige woorden relatief eenvoudige objecten zijn, leidt het herschrijven van zulke objecten al gauw tot moeilijke vragen. Een voorbeeld van zo'n niet-triviale kwestie is 'Zantema's puzzle'. Hierbij heet een SRS *terminerend*, als er geen oneindige herschrijfrijen zijn. Dus als elke herschrijfrij te zijner tijd noodzakelijk stopt.

Is de SRS met als enige regel aabb $\rightarrow$ bbbaaa *terminerend?*

1.2.1. OPGAVE. Schrijf (zo mogelijk) in deze SRS een herschrijfrij op met minstens 10 stappen.

1.3. *Tag systems*. Een formeel systeem dat enigszins lijkt op een SRS is het *tag system*. Dit werd vooral in de jaren 1930 bestudeerd, in de oertijd van de informatica toen het begrip 'berekenbaar' in de logica scherp werd gedefinieerd. Vooral de naam van Emil Post is verbonden aan het tag systeem. Een voorbeeld is in de volgende figuur (1.2) gegeven, met een voor zichzelf sprekende afbeelding van de twee regels van dit tag systeem, en een weergave van de reductierij $1 \rightarrow 10 \rightarrow 010 \rightarrow 1001 \rightarrow \ldots$ Er zijn moeilijke open problemen voor tag systems. De regels in dit voorbeeld kunnen we met variabelen zo schrijven:

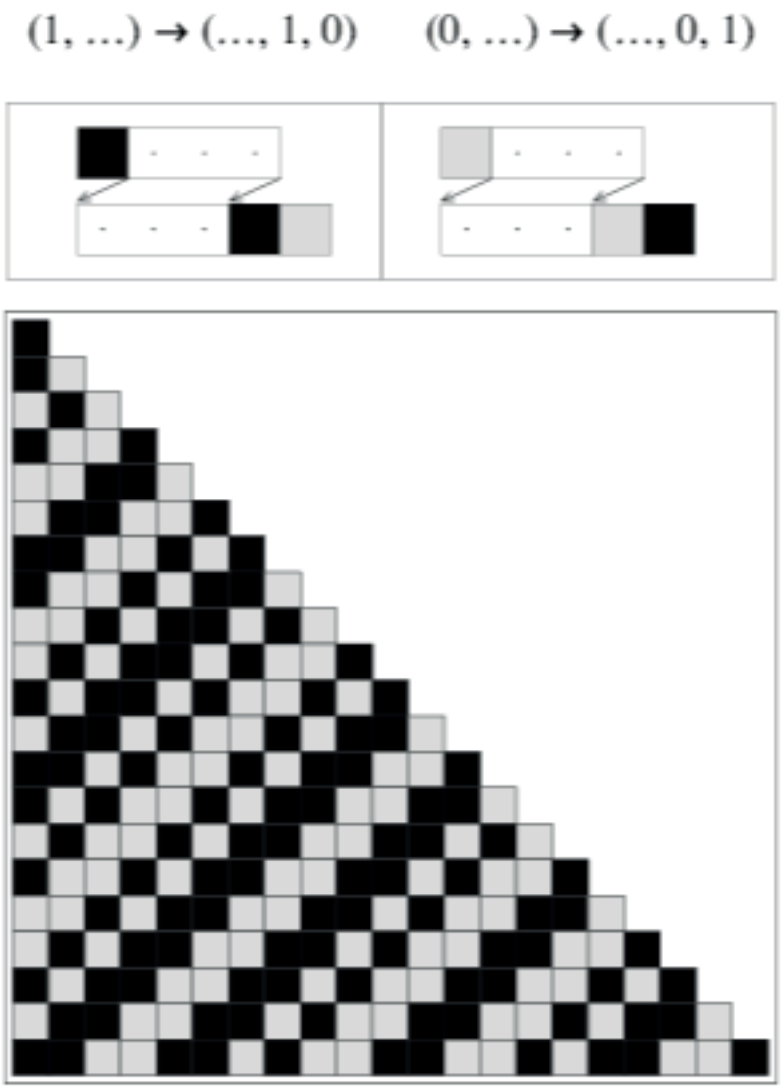
$1x \rightarrow x10$
$0x \rightarrow x01.$

Hierbij staat 'x' voor een willekeurig woord.
Van belang is te begrijpen dat deze regels niet passen in het stramien van de SRS.
(Zie ook `http://mathworld.wolfram.com/TagSystem.html`)

1.4. Een nog ander verwant systeem dat betrekking heeft op woorden, is *de 1-*

dimensionale cellulaire automaat. Zie hiervoor het boek en de web-site van S. Wolfram, A New Kind of Science.

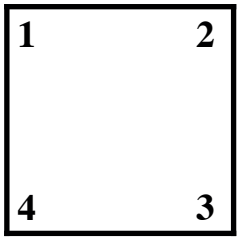
1.4.1. OPGAVE. Schrijf een scriptie van maximaal 10 pagina's over de systemen SRS, tag systems, Post Production Systems, 1-dimensionale cellulaire automaten, 2-dimensionale cellulaire automaten (Bijv. Conway's life), en verzamel informatie over de berekeningskracht van deze formalismen. (Zijn ze 'Turing complete'?) Bijv., is er een universele (i.e. Turing complete) SRS?



adapted from Wolfram, S. *A New Kind of Science.* Wolfram Media, p. 93, 2002.

Figuur 1.2

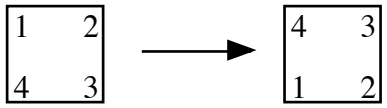
1.5. *Symmetrieën van het vierkant.*



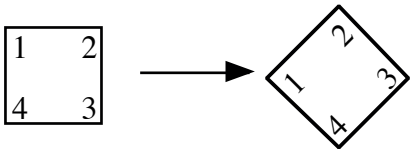
We bekijken nu een voorbeeld van een SRS waarin de letters en de woorden een betekenis (semantiek) hebben. Gegeven is een vierkant in het platte vlak, met hoekpunten genummerd. Denk aan dit vierkant als een star object, van karton, dat op het platte vlak ligt. De notie van symmetrie van en object is heel belangrijk in de wiskunde, met name in de groepentheorie. Een symmetrie is een "afstand-behoudende" afbeelding (transformatie) die het object weer 'op zichzelf afbeeldt'. In de woorden van de beroemde wiskundige Hermann Weyl:

> An object is said to be **symmetrical** if one can subject it to a certain **operation** and it appears exactly the same after the operation as before. Any such operation is called a **symmetry** of the object.

Bijv. de horizontale spiegeling is zo'n symmetrie.



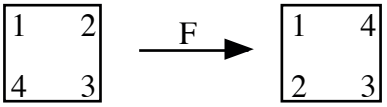
Daarentegen is een draaiing over -45 graden geen symmetrie.



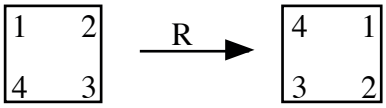
We werken nu met twee symmetrieën die als het ware een 'basis' vormen voor alle symmetrieën. Dat zijn er 8.

1.5.1. OPGAVE. Ga na dat er 8 symmetrieën zijn en schrijf deze op in de notatie als boven.

De twee symmetrieën die we als basis nemen, zijn F:
(*flip diagonaal, twee-dimensionaal*)



en R:
(*roteer 90 graden met de klok mee*).



Met F en R kunnen we woorden vormen zoals RFRR, met betekenis: roteer, flip. roteer, roteer achtereenvolgens. De F,R-woorden kunnen we herschrijven en daarmee vereenvoudigen met de volgende regels:

$$FF \rightarrow \lambda$$

$RRRR \to \lambda$
$FR \to RRRF$

waarbij λ het lege woord is, met de betekenis 'doe niets', oftewel de identiteitstransformatie.

1.5.2. OPGAVE. (i) Bepaal de normaalvormen van deze SRS.
(ii) Laat zien dat deze normaalvormen precies overeenkomen met de 8 symmetrieën die we eerder vonden in Opgave 1.5.1.

Met deze SRS kunnen we het 'woordprobleem' voor F, R-woorden bij de gegeven semantiek oplossen. Bijvoorbeeld stel dat we de woorden

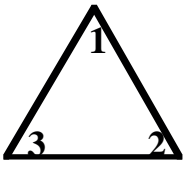
α: FRFRFRFR
en β: RRFRRFRRFRRF

gegeven hebben, met de vraag of α en β dezelfde symmetrie voorstellen. De naieve methode is om dit expliciet uit te rekenen, door vanuit het oorspronkelijke vierkant de transformaties achtereenvolgens toe te passen en te zien of de twee eindresultaten gelijk zijn.
De slimme methode is beide woorden met de SRS tot normaalvorm te herleiden, en die resultaten te vergelijken. Waarom dit werkt zullen we nog zien.

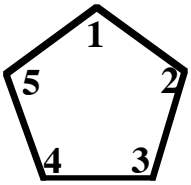
1.5.3. OPGAVE. Voer dit uit.

1.5.4. OPGAVE. Bepaal de reductiegraaf van het woord R FR FR. (Bewerkelijk!)

1.5.5. OPGAVE. Geef een SRS voor de symmetrieen van een gelijkzijdige driehoek. geef ook de normaalvormen van de SRS.



1.5.6. OPGAVE. Idem voor het regelmatige pentagon.



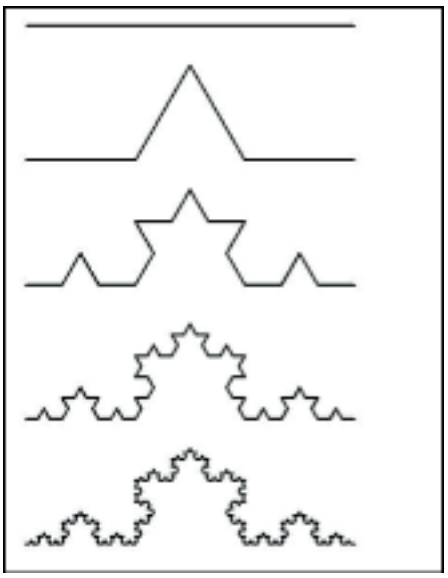
1.5. In deze 'microscoop' sectie zullen we preciezer ingaan op de vraag waarom bovenstaande methode, het reduceren naar normalvorm, deze vergelijken, en daarmee het woordprobleem oplossen, inderdaad werkt.
We voeren eerst een notatie in voor de semantiek van een woord α. Die schrijven we als $[\![\alpha]\!]$. De definitie is:

$[\![\lambda]\!] = \mathrm{id}$,
$[\![R]\!] = \mathrm{roteer}$,
$[\![F]\!] = \mathrm{flip}$
$[\![\alpha\beta]\!] = [\![\alpha]\!]\ [\![\beta]\!]$.

Verder schrijven we $nf(\alpha)$ voor de normaalvorm van α. We kunnen deze functie-notatie *nf* gebruiken, om dat een woord α precies één normaalvorm heeft. Dat is weer het gevolg van het later te bewijzen feit dat de reductie in deze SRS *terminerend* en *confluent* is. Deze begrippen komen later uitvoerig aan de orde. We kunnen nu de werking van boven gebruikte methode weergeven en bewijzen. Nog een laatste notatie is die voor '*syntactische gelijkheid*', $\equiv$. We schrijven $\alpha \equiv \beta$, als de woorden α en β syntactisch gelijk zijn, dat wil zeggen als het dezelfde woorden zijn.

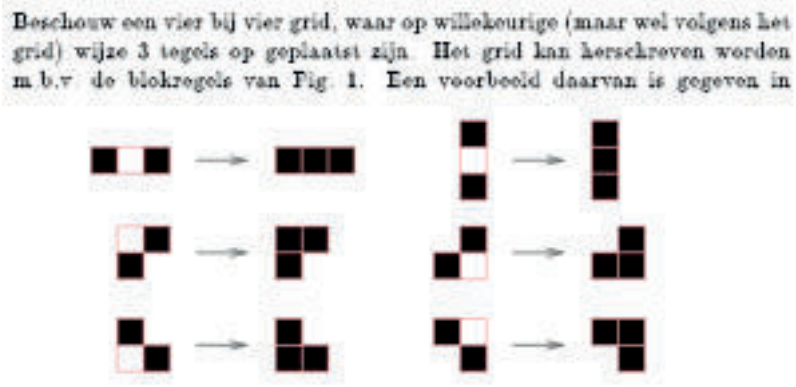
STELLING. $[\![\alpha]\!] = [\![\beta]\!] \Leftrightarrow nf(\alpha) \equiv nf(\beta)$.

BEWIJS. ($\Leftarrow$) We bewijzen eerst voor alle regels $\alpha \to \beta$ van de SRS dat $[\![\alpha]\!] = [\![\beta]\!]$.Vervolgens bewijzen we eenvoudig dat voor meerstapsreducties $\alpha \twoheadrightarrow \beta$ eveneens geldt $[\![\alpha]\!] = [\![\beta]\!]$. Wegens $\alpha \twoheadrightarrow nf(\alpha)$, hebben we dus $[\![\alpha]\!] = [\![nf(\alpha)]\!]$. $\Leftarrow$
($\Rightarrow$) Equivalent is de contrapositie te bewijzen: $\neg(nf(\alpha) \equiv nf(\beta)) \Rightarrow [\![\alpha]\!] \neq [\![\beta]\!]$.
Dit is Opgave 1.5.2. $\Rightarrow$



Figuur 1.3. Sneeuwvlokcurve

1.5. *Mini-life* (V. van Oostrom)



Figuur 1.4. Blokregels

1.6. *Life* (J.H. Conway.)

1.6.1. OPGAVE. Zoek op het internet de regels voor Conway's life. Zijn er normalvormen? Sommige configuraties heten 'Garden of Eden', omdat ze geen voorganger hebben. Is deze notie interessant voor de SRS-en die we gezien hebben?

1.7. *Sneeuwvlok curve* (Koch curve.) Deze kromme wordt verkregen als limiet-resultaat van een iteratie zoals in de figuur.

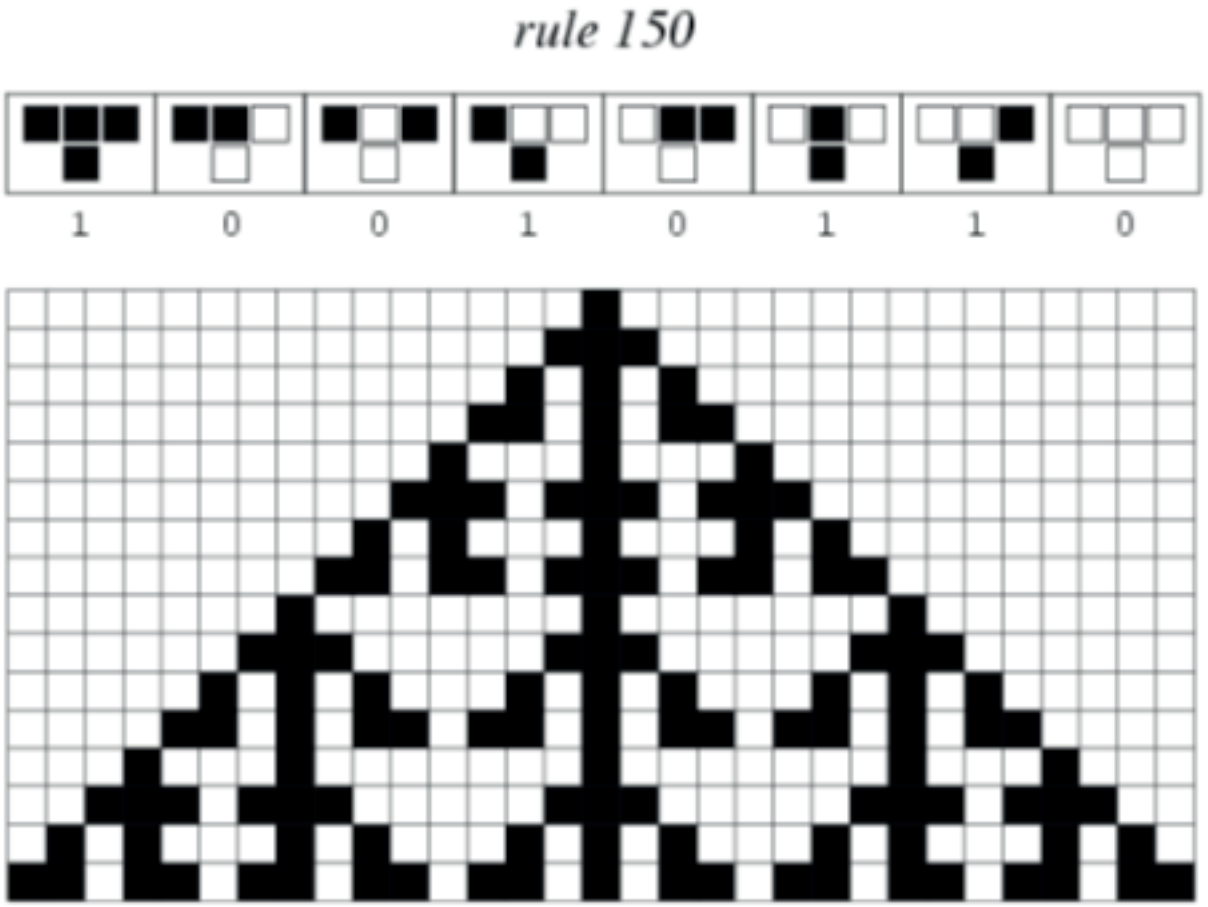
1.7.1. OPGAVE. Probeer met een SRS de opvolgende iteraties van de sneeuwvlok kromme te 'beschrijven'.

http://www.irisa.fr/lande/genet/ra.html#[1]
http://www.bath.ac.uk/~cs1spw/notes/CompIII/notes03.html

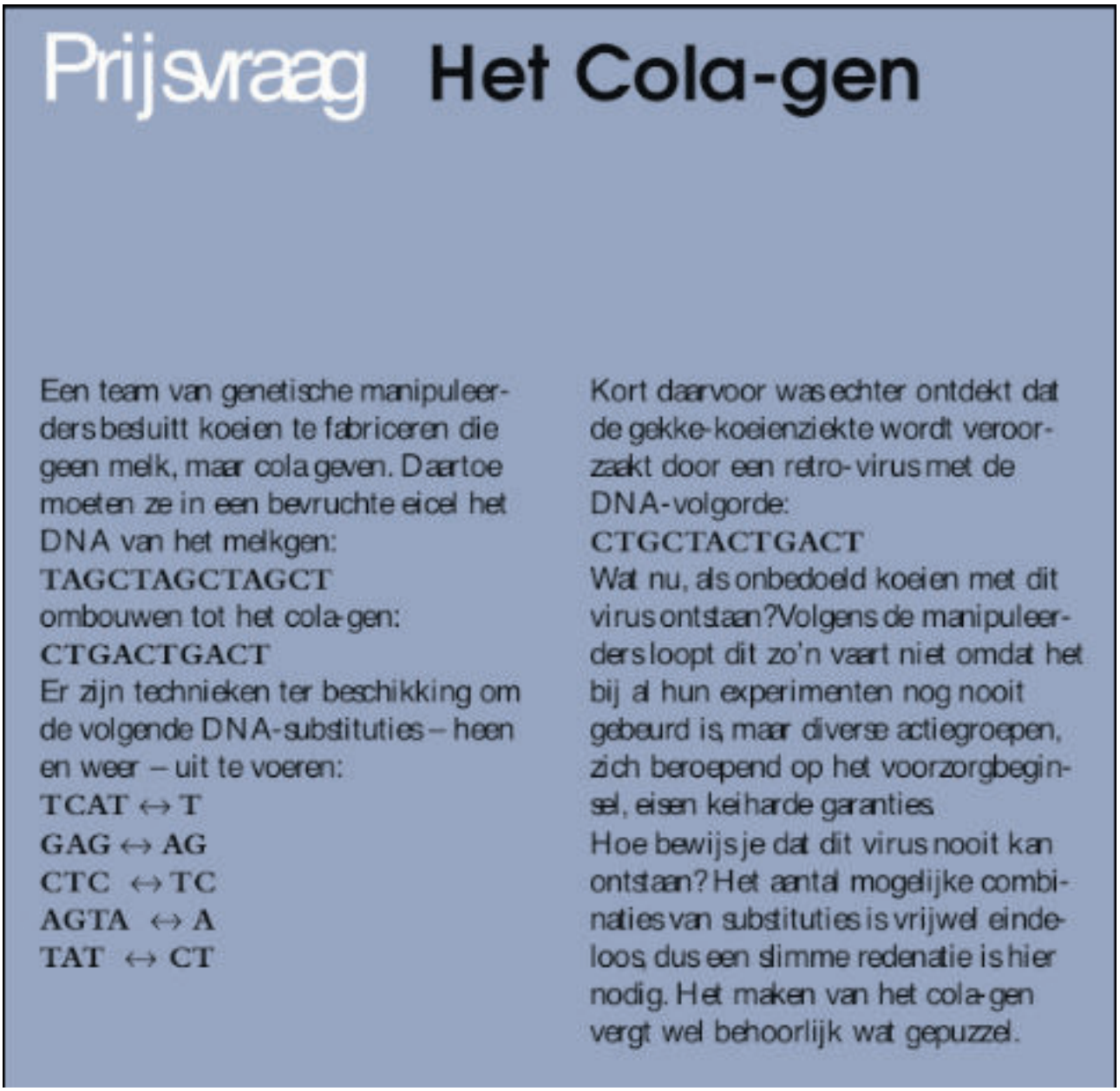
1.7. *knopen, reidemeister moves.*
Op dit onderwerp komen we bij de behandeling van Abstract herschrijven.

1.8. *Retoucheren.*

1.7.2. OPGAVE. Experimenteer met een simpel algoritme voor een 2-dimensionale cellulaire automaat, die een vorm van retoucheren oplevert. Dat wil zeggen dat bijv. in een omgeving die rood geleurd is, een verstoring de kleur van de omgeving aanneemt.



Figuur 1.5. Rule 150.



Figuur 1.6. Woord herschrijven: januari 2005 puzzel uit Natuurwetenschap en Techniek.

The theorem by Hans Zantema and Alfons Geser is:

Let p , q , r and s denote positive integers. Then the one-rule string rewriting system

$$0^p1^q \rightarrow 1^r0^s$$

terminates if and only if:
(by 0^p1^q we mean p zeros followed by q ones.)

1. $p \geq s$ or $q \geq r$ or
2. $p < s < 2p$ and $q < r$ and q is not a divisor of r or
   $q < r < 2q$ and $p < s$ and p is not a divisor of s.

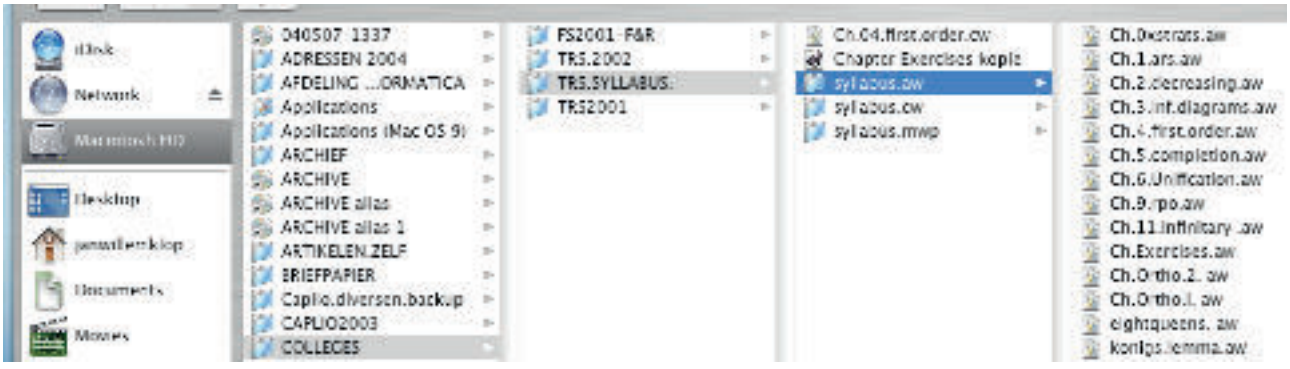
2

TERMEN

INHOUD.

2.0. *Introductie.* In dit hoofdstuk worden termen geintroduceerd, samen met herschrijfregels. We bezien ook enkele notatie formats. In een 'microscoop sectie' geven we precieze definities; eerst voeren we de begrippen in op kaleidoscoop manier.

2.1. *Termen.* Woorden zijn ééndimensionaal. Termen zijn tweedimensionaal, tenminste in de boomnotatie. Daarmee zijn termen meer expressief dan woorden. Het is een betere informatiedrager. We zien al termen bij het zoeken in een filestructuur, zoals in de menu's in Figuur 2.1.



Figuur 2.1. Zoeken in file-termboom.

2.2. *Twee maal twee is vier.* Het volgende voorbeeld beschrijft optelling (A) en vermenigvuldiging (M) van natuurlijke getallen 0,1,2,3, ... , of in de notatie van dit voorbeeld 0, S(0), S(S(0)), S(S(S(0))), ...

r_1	$A(x,0) \rightarrow x$
r_2	$A(x,S(y)) \rightarrow S(A(x,y))$
r_3	$M(x,0) \rightarrow 0$
r_4	$M(x,S(y)) \rightarrow A(M(x,y),x)$

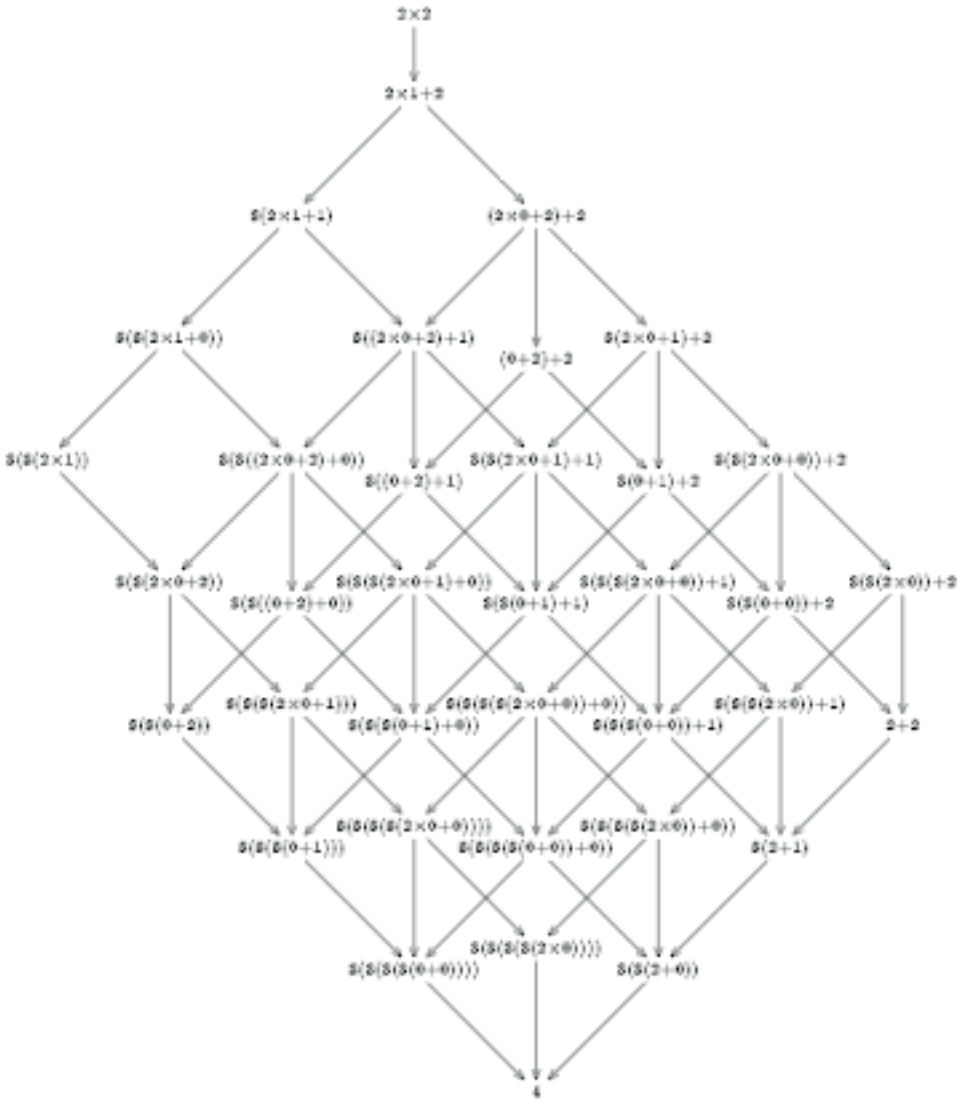
Tabel 2.1. De AMS0 TRS

Nu hebben we de meerstapsreductie M(S(S(0)), S(S(0))) Now M(S(S(0)), S(S(0))) $\twoheadrightarrow$ S(S(S(S(0)))), met de volgende reductiestappen (of herschrijfstappen):

M(S(S(0)), S(S(0))) →
A(M(S(S(0)), S(0)), S(S(0))) →
S(**A(M(S(S(0)), S(0)),S(0))**) →
S(S(**A(M(S(S(0)), S(0)),0)**)) →
S(S(**M(S(S(0)), S(0))**)) →
S(S(A(**M(S(S(0)),0)**, S(S(0))))) →
S(S(**A(0,S(S(0)))**)) →
S(S(S(**A(0,S(0))**))) →
S(S(S(S(**A(0,0)**))) →
S(S(S(S(0)))).

In iedere stap is de sub-term die herschreven wordt, het redex, rood gekleurd. Het woord 'redex' is een samentrekking van 'reducible expression'. Merk op dat dit niet de enige reductie is van M(S(S(0)), S(S(0))) naar S(S(S(S(0)))). In Figuur 2.2 zijn alle mogelijke reducties van M(S(S(0)), S(S(0))) naar S(S(S(S(0)))) zichtbaar, in een iets afwijkende notatie (infix notatie). Er zijn dus heel wat mogelijke reducties van M(S(S(0)), S(S(0))) naar S(S(S(S(0)))), oftewel heel wat manieren om de berekening $2 \times 2 = 4$ uit te voeren. Het is helemaal niet vanzelfsprekend dat al die manieren hetzelfde resultaat opleveren. Later zullen we inzien waarom dit zo is. Daarop vooruit lopend: de AMS0 TRS is *confluent en terminerend*. En dat garandeert unieke normaalvormen.

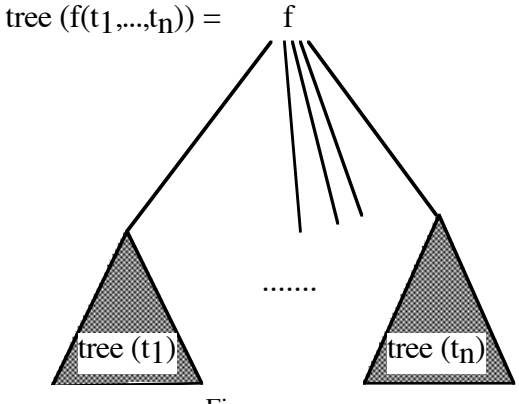
2.1. OPGAVE. Geef een andere reductie van M(S(S(0)), S(S(0))) naar S(S(S(S(0)))) dan bovenstaande in de notatie als boven, niet de infix notatie.



Figuur 2.2. *Twee maal twee is vier*.

2.2. OPGAVE. Schrijf bovenstaande reductie ook in Boom-notatie.

Deze is inductief gedefinieerd door: tree(x) = x; en verder als in de Figuur.



Figuur xx

2.3. OPGAVE. Hoeveel reducties van M(S(S(0)), S(S(0))) naar S(S(S(S(0)))) zijn er?
En modulo Lévy equivalentie? (Voor later.)

2.4. OPGAVE. Bewijs dat de optelling en vermeigvuldiging gedefinieerd door de herschrijfregels in Tabel 2.1 commutatief en associatief zijn. (Bewerkelijk.)

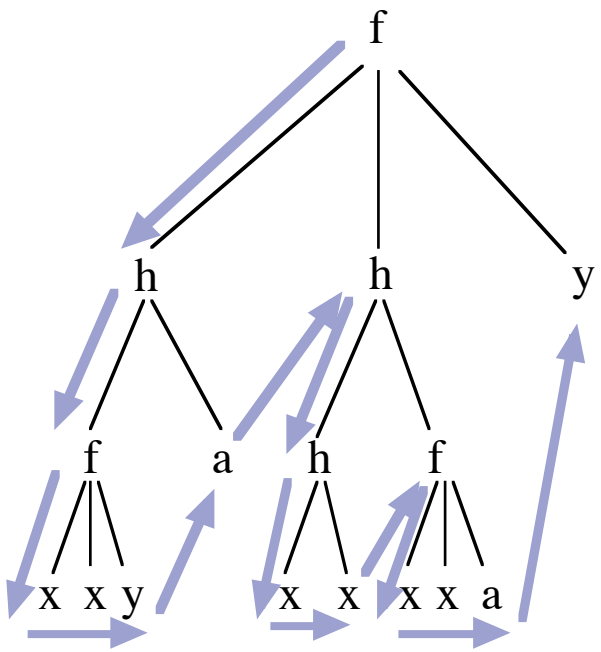
2.3. *Poolse notatie.*

Er zijn verschillende manieren om termen te noteren. Behalve de 'gewone' notatie, die we de 'functionele notatie' zullen noemen, en de 'boom-notatie', is er de poolse notatie en de reversed polish notation. Leesover de laatste zelf in de encyclopedie (bijv.) Wikipedia, zie de slotsectie met verwijzingen (met het 'boek'). Laten we nu alleen naar de poolse notatie kijken. Wikipedia vermeldt:

Polish notation, also known as **prefix notation** was created by Jan Lukasiewicz. Operators are placed before operands.
`+ 1 2`
Yields (as expected) 3.
It is not limited to only two values, nor to just addition.
`(* (+ 0 1) (+ 2 3))`
Returns 5.
While the examples above use parentheses, one of the benefits of Polish notation is that, assuming the arity of each operator is known, parentheses are unnecessary: the order of operations is unique and easy to determine, if the expression is known to be correct. For example, assuming * and + are binary,
`* + 0 1 + 2 3`
can refer *only* to

`(* (+ 0 1) (+ 2 3))`

Jan Lukasiewicz



Figuur 2.3. Boomnotatie en poolse notatie.

Er is een interessant verband tussen de boom notatie en de poolse notatie. Namelijk: uit de boom notatie is de poolse notatie af te lezen door een 'left-to-right tree traversal', als aangegeven in Figuur 2.3.

OPGAVE. Bewijs dit door de volgende definities na te gaan. Zij *trav* (tree traversal) de afbeelding van bomen naar woorden. Deze kunnen we compact inductief definieren, als volgt:

$$trav(make\text{-}tree(f,T_1,...,T_n)) = f\ trav(T_1)\ ...\ trav(T_n)$$

voor bomen $T_1,...,T_n$ en n-aire f.
Verder hebben we de afbeelding *tree* van termen Ter(Σ) naar bomen, gedefinieerd door

$$tree\ (f(t_1,...,t_n)) = make\text{-}tree(f,\ tree(t_1),...,tree(t_n)).$$

Met de definitie van de afbeelding *polish* (poolse notatie) van termen naar woorden, die eenvoudig alle haakjes en komma's verwijdert, hebben we dus

$$polish = trav \circ tree.$$

We hebben nu dus drie belangrijke data typen gerelateerd: *woorden*, *termen*, en *bomen*.

In de nu volgende M-sectie zullen we bewijzen wat Wikipedia boven beweert, namelijk dat haakjes en komma's niet nodig zijn, en dat een term uniek teruggelezen kan worden uit zijn poolse notatie. Dit vereist wat precisiewerk.

Gegeven is een alphabet (signatuur) Σ van functiesymbolen met ariteit. Termen in poolse notatie worden als volgt inductief verkregen:

(i) variabelen $x, y, z, \ldots \in Ter$;
(ii) $t_1,\ldots,t_n \in Ter$, f een n-air functiesymbool $\Rightarrow ft_1,\ldots,t_n \in Ter$. Hierbij zijn f, $t_1,\ldots,t_n$ geconcateneerd, dat wil zeggen, achter elkaar gezet. In het bijzonder hebben we voor een constante c (die 0-air is) dat $c \in Ter$.

Voorbeeld. Zij $\Sigma = \{A, M, S, 0\}$ met A, M binair, S unair, 0 0-air. Dan zijn $MSS0SS0$, $AxAyz \in Ter$.

GENERATIE LEMMA. *Als* t *een term is, dan is er een* n*-aire* f *en zijn er termen* $s_1,\ldots,s_n$ *zodat* $t \equiv fs_1\ldots s_n$.

Dit onschuldig ogende lemma zegt dus dat elke term ontbonden kan worden in kopsymbool gevolgd door een rij argumenten, zoveel als de ariteit is van het kopsymbool. Het bewijs is een direct gevolg van de inductieve definitie van termen. Wat we nog niet hebben vastgesteld, is of elke term ook *uniek* ontbonden kan worden.

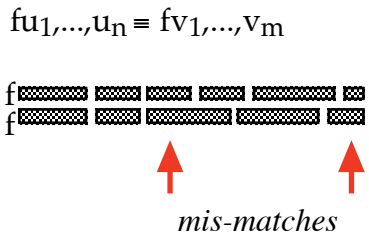
Om dat te bewijzen, moeten we uitsluiten dat gegeven een term fw met $w \in \Sigma^*$, deze op twee verschillende manieren is te "ontleden":

$$fu_1,\ldots,u_n \equiv fv_1,\ldots,v_m \text{ voor zekere } n, m \in \mathbb{N} \text{ en } u_i, v_j \in Ter.$$

Als we deze twee ontledingen onder elkaar zetten en vergelijken, dan zien we meteen dat ofwel

Geval 1: 'het past precies', en de ontledingen zijn identiek, dus n = m en $u_i \equiv v_j$ voor i = 1,...,n; of

Geval 2: er is een *mis-match*. Dat is een syntactische vergelijking t ≡ sw met t, s ∈ Ter, w ∈ Σ*, w niet λ, het lege woord. We definieren als de *grootte* van een mis-match de lengte van t.



Figuur xx. Mis-match

Dus stel er is een mis-match t ≡ sw. Met het Generatie Lemma kunnen we zeggen dat t ≡ $ft_1...t_n$ voor zekere n-aire f en termen $t_1,...,t_n$. En dat s ≡ $gs_1...s_m$ voor zekere m-aire g en termen $s_1,...,s_m$. Dus de mis-match ziet er uit als

$$ft_1...t_n \equiv gs_1...s_mw.$$

Evident is dat f ≡ g. Dus m = n, en de mis-match is in feite:

$$ft_1...t_n \equiv fs_1...s_nw.$$

Maar nu geeft een letter-voor-letter vergelijking weer een mis-match - en wel een kortere! En dus zijn we klaar. (Waarom?)

2.6. *Ackerman functie*
Ackermann's function

Definition: A function of two parameters whose value grows very fast.

Formal Definition:
- A(0, j)=j+1 for j ≥ 0
- A(i, 0)=A(i-1, 1) for i > 0
- A(i, j)=A(i-1, A(i, j-1)) for i, j > 0

A(4, 2) is greater than the number of particles in the universe raised to the power 200. A(5,

2) is the item at column A(5, 1) in the m = 4 row, and cannot be written as a decimal expansion in the physical universe. Beyond row 4 and column 1, the values can no longer be feasibly written with any standard notation other than the Ackermann function itself — writing them as decimal expansions, or even as references to rows with lower m, is not possible.

If you were able to expand every particle in the universe to a universe the size of ours by snapping your fingers, and likewise with all the particles in the created universes, and did this repeatedly, you would die of old age before the number of particles reached A(4, 3). Note that A(5, 1) is larger than even this number.

http://en.wikipedia.org/wiki/Ackermann_function

OPGAVE. Bereken A(3, 3).

A(0, x)	→	S(x)
A(S(x), 0)	→	A(x, S(0))
A(S(x), S(y))	→	A(x, A(S(x),y))

.4. *Combinatory Logic.*

Combinatory logic is concerned with certain basic notions of the foundations of mathematics which are usually used in an intuitive and unanalysed way. Such notions include substitution, usually introduced by the use of variables, and classification of the entities of a system into types, which is usually provided for by rules which are auxiliary to, but not part of, the system. The part of combinatory logic which is concerned with questions of a fundamental nature which, like substitution, involve variables, is called the theory of combinators.

2.8. *Formele definities.*

A (first-order) Term Rewriting System (TRS) is a pair (Σ,R) of an *alphabet* or *signature* Σ and a set of reduction rules (rewrite rules) R. The alphabet Σ consists of:

(i) a countably infinite set of *variables* x_1, x_2, x_3, ... also denoted as x, y, z, x', y', ...

(ii) a non-empty set of *function symbols* or *operator symbols* F, G, ..., each equipped with an 'arity' (a natural number), i.e. the number of 'arguments' it is supposed to have. We not only (may) have unary, binary, ternary, etc., function symbols, but also 0-ary: these are also called *constant symbols.*

The set of terms (or expressions) 'over' Σ is $Ter(\Sigma)$ and is defined inductively:

(i) $x, y, z, ... \in Ter(\Sigma)$,

(ii) if F is an n-ary function symbol and $t_1,...,t_n \in Ter(\Sigma)$ $(n \geq 0)$, then $F(t_1,...,t_n) \in Ter(\Sigma)$. The t_i $(i = 1,...,n)$ are the *arguments* of the last term.

Terms not containing a variable are called *ground* terms (also: *closed* terms), and $Ter_0(\Sigma)$ is the set of ground terms. Terms in which no variable occurs twice or more, are called *linear*.

Contexts are 'terms' containing one occurrence of a special symbol , denoting an empty place. A context is generally denoted by C[ ]. If $t \in Ter(\Sigma)$ and t is substituted in , the result is $C[t] \in Ter(\Sigma)$; t is said to be a subterm of C[t], notation $t \subseteq C[t]$. Since is itself a context, the trivial context, we also have $t \subseteq t$.

Often this notion of subterm is not precise enough, and we have to distinguish *occurrences* of subterms (or symbols) in a term; it is easy to define the notion of occurrence formally, using sequence numbers denoting a 'position' in the term, but here we will be satisfied with a more informal treatment.

EXAMPLE. Let $\Sigma = \{A, M, S, 0\}$ where the arities are 2,2,1,0 respectively. Then A(M(x, y), y) is a (non-linear) term, A(M(x, y), z) is a linear term, A(M(S(0), 0), S(0)) is a ground term, A(M(, 0),S(0)) is a context, S(0) is a subterm of A(M(S(0),0),S(0)) having two occurrences: A(M(**S(0)**,0),**S(0)**).

A *substitution* σ is a map from $Ter(\Sigma)$ to $Ter(\Sigma)$ satisfying $\sigma(F(t_1,...,t_n)) = F(\sigma(t_1),...,\sigma(t_n))$ for every n-ary function symbol F (here $n \geq 0$). So, σ is determined by its restriction to the set of variables. We also write t^σ instead of $\sigma(t)$.

A *reduction rule* (or rewrite rule) is a pair (t, s) of terms $\in Ter(\Sigma)$. It will be written as $t \to s$. Often a reduction rule will get a name, e.g. r, and we write $r: t \to s$. Two conditions will be imposed:

(i) the LHS (left-hand side) t is not a variable,

(ii) the variables in the right-hand side s are already contained in t.
A reduction rule r: $t \to s$ determines a set of *rewrites* $t^\sigma \to_r s^\sigma$ for all substitutions σ. The LHS t^σ is called a *redex* (from 'reducible expression'), more precisely an r-redex. A redex t^σ may be replaced by its '*contractum*' s^σ inside a context C[ ]; this gives rise to *reduction steps* (or one-step rewritings)

$$C[t^\sigma] \to_r C[s^\sigma].$$

A term without a redex is a *normal form*. We call $\to_r$ the *one-step reduction relation* generated by r. Concatenating reduction steps we have (possibly infinite) *reduction sequences* $t_0 \to t_1 \to t_2 \to ...$ or *reductions* for short. If $t_0 \to ... \to t_n$ we also write $t_0 \twoheadrightarrow t_n$, and t_n is a *reduct* of t_0.

It is understood that R does not contain two reduction rules that originate from each other by a 1-1 renaming of variables.

Semi-Thue systems. Semi-Thue Systems (STSs), as in Chapter 1, can be 'viewed' as TRSs, as follows. We demonstrate this by the following example of a STS:

Let T = {(aba, bab)} be a one-rule STS. Then T corresponds to the TRS R with unary function symbols a, b and a constant o, and the reduction rule $a(b(a(x))) \to b(a(b(x)))$. Now a reduction step in T, e.g.: bbabaaa → bbbabaa, translates in R to the reduction step $b(b(a(b(a(a(o))))))) \to b(b(b(a(b(a(a(o))))))))$. It is easy to see that this translation gives an 'isomorphism' between T and R (or more precisely $(R)_0$, the restriction to ground terms).

Applicative Term Rewriting Systems. In some important TRSs there is a special binary operator, called *application* (Ap). E.g. Combinatory Logic (CL), based on S, K, I, has the rewrite rules

Ap(Ap(Ap(S, x), y), z)	→ Ap(Ap(x, z), Ap(y,z))
Ap(Ap(K, x), y)	→ x
Ap(I, x)	→ x

Table xx

Here S, K, I are constants. Often one uses the infix notation (t.s) instead of Ap(t, s), in which case the rewrite rules of CL read as follows:

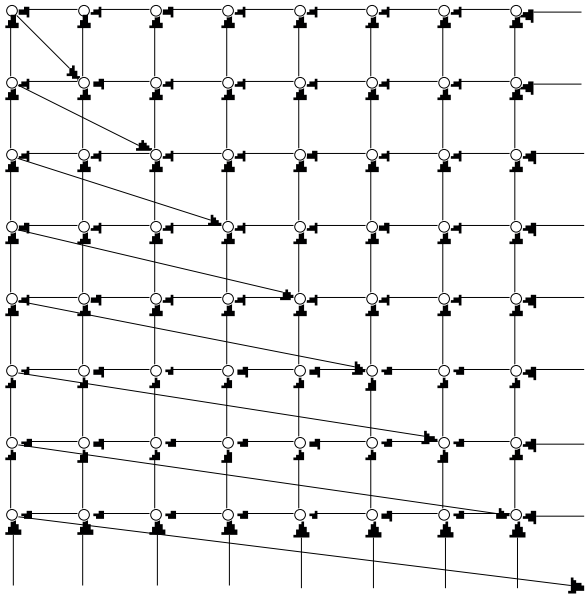
As in ordinary algebra, the dot is mostly suppressed; and a further notational simplification is that many pairs of brackets are dropped in the convention of *association to the left*. That is, one restores

the missing brackets choosing in each step of the restoration the leftmost possibility. Thus the three rules become:

((S.x).y).z	→ (x.z).(y.z)
(K.x).y	→ x
I.x	→ x

Table xx

SII(SII)



Figuur xx. Reductiegraaf van de combinator SII(SII)

The TRS CL has ‘universal computational power’: every (partial) recursive function on the natural numbers can be expressed in CL.

http://en.wikipedia.org/wiki/Reverse_Polish_notation
http://en.wikipedia.org/wiki/Polish_notation

3

COMBINATOREN

INHOUD.

3.0. *Introductie.*
In dit hoofdstuk leren we een bijzondere TRS kennen, die uit de dertiger jaren van de vorige eeuw stamt en Combinatorische Logica of kortweg CL heet. Het bijzondere van dit herschrijfsysteem is dat het universeel is, en alle mogelijke berekenbare functies kan beschrijven. Dit heet 'Turing compleet'. Dit is dus in contrast met TRSen zoals de AMS0 TRS, of de TRS voor het inverteren van een woord, die een 'special purpose' karakter hebben. De TRS CL is 'general purpose'. De oorspronkelijke ontdekker van CL was Moses Schönfinkel (boven rechts), in zijn artikel 'Über die Bausteine der mathematischen Logik' (1924). In 1930 werd CL herontdekt door Haskell Curry (boven links), die een systematische studie wijdde aan CL en de verwante lambda calculus.

3.1. *De termen van CL.* De signatuur van CL bestaat uit drie constanten, S, K en I, en een binair functiesymbool genaamd 'applicatie', en genoteerd als Ap of @. We zullen de laatste notatie hier gebruiken, tenminste als inleiding, want we zullen namelijk een notatie te zien krijgen, de zogenaamde 'applicatieve notatie', waarin het applicatiesymbool onzichtbaar is. Behalve S, K, I en @ hebben we ook oneindig veel variabelen x, y, z, ... waaruit de termen van CL opgebouwd kunnen worden. De verzameling Ter(CL) van CL-termen wordt dus zo gedefinieerd:

(i) S, K, I, x, y, z, ... ∈ Ter(CL)
(ii) M, N ∈ Ter(CL) ⇒ @(M, N) ∈ Ter(CL).

Dit is nog de functionele notatie. De applicatieve notatie ontstaat als volgt.

- Eerst schrijven we in plaats van @(M, N) korter in infix notatie (M·N), warbij '@' een infix 'punt' geworden is.
- Ook deze punt laten we meestal weg, net als in algebra notatie waarin de vermenigvuldigingspunt ook vaak wordt weggelaten.
- Ten derde sparen we veel haakjes uit door zoveel mogelijk haakjes weg te laten, onder de conventie van *'associatie naar links'*, Dit betekent dat de ontbrekende haakjes op zo 'links mogelijke' manier moeten worden gelezen. Dit vergt wel wat oefening. Vul om te beginnen daartoe in onderstaande tabel de ontbrekende vakjes in. Geef ook de boomnotatie van deze termen.

functionele notatie	*Alle haakjes, met infix punt*	*Alle haakjes zonder infix punt*	*zo weinig mogelijk haakjes*
@(@(x, y), z)	((x·y)·z)	((xy)z)	xyz
@(@(@(S, x), y), z)	(((S·x)·y)·z)	(((Sx)y)z)	Sxyz
@(@(x, z), @(y, z))	((x·z)·(y·z))	((xz)(yz))	xz(yz)
			x(zyz)
			x(z(yz))

Tabel 3.2. *Notatie van CL-termen*

Nu we de notatie van CL-termen hebben ingevoerd, is de beurt aan de herschrijfregels van CL. Voor elk van de constanten S, K en I is er één.

Sxyz → xz(yz)
Kxy → x
Ix → x

Tabel 3.1. *Reductieregels van CL*

Eigenlijk zou het nog iets zuiniger kunnen, en kunnen we de derde regel missen. De constante kan namelijk gedefinieerd worden uit S en K. Want:

SKKx → Kx(Kx) → x,

dus SKK doet hetzelfde op input x als I, zij het in meer stappen. Ook voor SKM geldt dit, met M willekeurig. We vervolgen onze ontdekkingstocht:

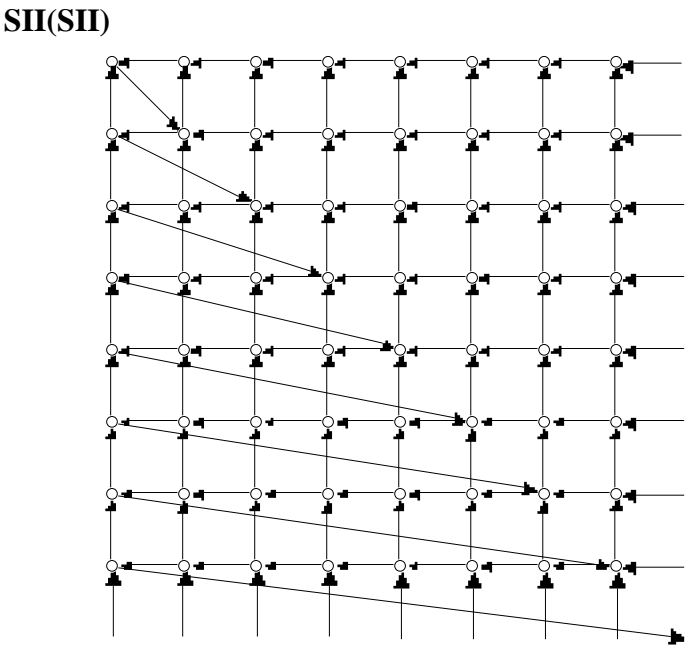
$$SIIx \to Ix(Ix) \to x(Ix) \to xx.$$

Dus SII is een verdubbelaar: een input M wordt verdubbeld, want $SIIM \twoheadrightarrow MM$. Dit heeft een merkwaardig gevolg, door SII op zichzelf toe te passen:

$$SII(SII) \to I(SII)(I(SII)) \to SII(I(SII)) \to SII(SII),$$

een *reductiecykel* of cyclische reductie.

In Figuur 3.1 zijn alle reducties van deze cyclische term weergegeven.



Figuur 3.1. *Reductiegraaf van SII(SII)*

De term $\Omega \equiv SII(SII)$ is interessant, maar niet nuttig. In feite is Ω het schoolvoorbeeld van een 'ongedefinieerde term', een term zonder betekenis. Een term die wel erg nuttig is, is $B \equiv S(KS)K$:

$$Bxyz \twoheadrightarrow x(yz).$$

$S(KS)Kxyz$
$KSx(Kx)yz$

S(Kx)yz
Kxz(yz)
x(yz).

En voor C ≡ S(BBS)(KK) geldt: Cxyz ↠ xzy.

S(BBS)(KK)xyz
BBSx(KKx)yz
B(Sx)(KKx)yz
Sx(KKxy)z
xz(KKxyz)
xz(Kyz)
xzy

SII(SII)
I(SII)(I(SII))
SII(I(SII))
I(I(SII))(I(I(SII)))
I(SII)(I(I(SII)))
SII(I(I(SII)))
I(I(I(SII)))(I(I(I(SII))))
I(I(SII))(I(I(I(SII))))
I(SII)(I(I(I(SII))))
SII(I(I(I(SII))))
I(I(I(I(SII))))(I(I(I(I(SII)))))
I(I(I(SII)))(I(I(I(I(SII)))))
I(I(SII))(I(I(I(I(SII)))))
I(SII)(I(I(I(I(SII)))))
SII(I(I(I(I(SII)))))
I(I(I(I(I(SII))))(I(I(I(I(I(SII))))))
I(I(I(SII)))(I(I(I(I(I(SII))))))
I(I(SII))(I(I(I(I(I(SII))))))
I(SII)(I(I(I(I(I(SII))))))
SII(I(I(I(I(SII)))))
I(I(I(I(I(SII))))(I(I(I(I(I(SII))))))
I(I(I(I(SII)))(I(I(I(I(I(SII))))))

I(I(I(I(SII))))(I(I(I(I(I(SII)))))))
I(I(I(SII)))(I(I(I(I(I(SII)))))))
I(I(SII))(I(I(I(I(I(SII)))))))
I(SII)(I(I(I(I(I(SII)))))))
SII(I(I(I(I(I(SII)))))))
I(I(I(I(I(I(SII)))))))(I(I(I(I(I(I(SII)))))))

Solution 3.3.14. (i) As shorthand define A ≡ SSS, D ≡ SAA, E ≡ SAD and, inductively, $A^0E \equiv E$, $A^{n+1}E \equiv A(A^nE)$, $n \geq 0$. We claim:

(1) $(\exists C[\,])\ AAA \twoheadrightarrow C[D(AE)]$;

(2) $(\forall n \geq 1)(\exists C[\,])A^nEx \twoheadrightarrow C[AEx]$;

(3) $(\forall n \geq 1)(\exists C[\,])D(A^nE) \twoheadrightarrow C[D(A^{n+1}E)]$.

From (1) and (3) it follows that SSS(SSS)(SSS) has an infinite reduction.
(2) will be used in the proof of (3). We now prove (1), (2) and (3).

(1) $AAA \equiv SSSAA \rightarrow$

$SA(SA)A \rightarrow AAD \equiv SSSAD \rightarrow$

$SA(SA)D \rightarrow AD(SAD) \equiv SSSDE \rightarrow$

$SD(SD)E \rightarrow DE(SDE) \equiv C_1[DE] \equiv C_1[SAAE] \rightarrow$

$C_1[AE(AE)] \equiv C_1[SSSE(AE)] \rightarrow$

$C_1[SE(SE)(AE)] \rightarrow$

$C_1[E(AE)(SE(AE))] \rightarrow$

$C_2[E(AE)] \equiv C_2[SAD(AE)] \rightarrow$

$C_2[A(AE)(D(AE))] \equiv C_3[D(AE)]$.

(2) $A^nEx \equiv SSS(A^{n-1}E)x \rightarrow$

$S(A^{n-1}E)(S(A^{n-1}E))x \rightarrow$

$A^{n-1}Ex(S(A^{n-1}E)x) \equiv C[A^{n-1}Ex]$ and the result follows by a simple induction argument.

(3) $D(A^nE) \equiv SAA(A^nE) \rightarrow$

$A(A^nE)(A(A^nE)) \equiv A^{n+1}E(A^{n+1}E) \twoheadrightarrow$

$$C_1[AE(A^{n+1}E)] \equiv C_1[SSSE(A^{n+1}E)] \rightarrow$$

$$C_1[SE(SE)(A^{n+1}E)] \rightarrow$$

$$C_1[E(A^{n+1}E)(SE(A^{n+1}E))] \equiv C_2[E(A^{n+1}E)] \equiv C_2[SAD(A^{n+1}E)] \rightarrow$$

$$C_2[A(A^{n+1}E)(D(A^{n+1}E))] \equiv C_3[D(A^{n+1}E)].$$

(ii) Define the length of a CL-term M, notation $|M|$, as the number of occurrences of S, K, I in it, so: $|S| = |K| = |I| = 1$; $|MN| = |M| + |N|$.

Define the weight of a CL-term M, notation $\|N\|$, as follows:

$\|S\| = \|K\| = \|I\| = 1$, $\|MN\| = 2\|M\| + \|N\|$. It is easy to prove that

(a) If $M \rightarrow N$ then $|M| \leq |N|$.

(b) If $M \rightarrow N$ and $|M| = |N|$, then the contracted redex must be an S-redex of the form SABS for some A, B.

(c) For every 'S-context' C[ ] (i.e., an S-term with one open place [ ]) we have that $\|M\| > \|N\|$ implies $\|C[M]\| > \|C[N]\|$.

Now suppose a reduction cycle consisting of S-terms exists; let it be

$M_0 \rightarrow M_1 \rightarrow \ldots \rightarrow M_n \equiv M_0$, $(n \geq 1)$. According to (a) we have $|M_0| = |M_1| = \ldots = |M_n|$. According to (b) every redex contracted in this cyclic reduction has the form SABS. However:

$\|SABS\| = 4\|A\| + 2\|B\| + 9$, while $\|AS(BS)\| = 4\|A\| + 2\|B\| + 3$. But then we have by (c): $\|M_0\| > \|M_1\| > \ldots > \|M_n\| = \|M_0\|$, contradiction.

(iii) We prove SN for a class SP of CL-terms that includes the flat S, K-terms, defined as follows:

(1) $S, K \in SP$;

(2) $M \in SP \Rightarrow MS, MK \in SP$;

(3) $M \in SP \Rightarrow KKM, KSM, SKM, SSM \in SP$.

It is easy to check that SP is closed under reduction. Also one easily checks that if $M, N \in SP$ and $M \rightarrow N$, we have $|M| = |N|$ in case an S-redex is contracted and $|M| > |N|$ in case a K-redex is contracted. Now suppose that not all terms in SP are SN. From the last observation it is clear that an infinite reduction of SP terms $M_0 \rightarrow M_1 \rightarrow \ldots$ must contain a cycle $M_i \rightarrow M_{i+1} \rightarrow \ldots \rightarrow Mi_{i+j+1}$ for some $i, j \geq 0$. In this cycle only S-rexes can be contracted. But then we can make a cycle of S-terms by replacing in M_i, M_{i+1}, . . . , M_{i+j+1} all occurrences of K by occurrences of S. This contradicts (ii) above; so we have proved that all SP terms

are SN, in particular flat S, K-terms.

(iv) SISSSII ↠ SII(SII) ↠ SII(SII).

Solution (3.3.15). There are 8 convertibility classes; representants are:

S
SK
SKS
SKSK = KK(SK) = K
SKSKS = KS
SKSKSKS = SS
SKSKSKSK = SSK
SKSKSKSKS = SSKS = SS(KS)
All subsequent terms in the sequence are convertible with one of these 8. Note that we have SKSKSK = KSK = S

Solution 3.3.18(iv). Let Y be a fixed-point combinator, so YM = M(YM) for all CL-terms M. Let Y′ ≡ Y(SI). Then Y′M ≡ Y(SI)M = SI(Y(SI))M ≡ SIY′M = IM(Y′M). Hence Y′ is a fixed point combinator.l

SOLUTION 3.3.16. We employ the sets S (S-terms with variables), and T (variable-tail S-terms, inductively defined as follows.

(i) Var ⊆ S;

(ii) S ∈ S;

(iii) s, s ∈ S ⇒ ss∈ S;

(iv) Var ⊆ T ;

(v) s ∈ S, t ∈ T ⇒ st ∈ T .

CLAIM. If s ∈ S and sxyzu ↠ s′ ≡ s″u with u not ∈ s″.

Note that T ⊆ S. If X, Y are sets of terms, then XY = {MN | M ∈ X, N ∈ Y }. Consider the sets T^i (i = 1, 2, 3, 4). All four are closed under rewriting at the root. That is, for i = 1, 2, 3, 4 we have:

$$SABC \in T^i \Rightarrow AC(BC) \in T^i.$$

Next one proves that these four sets are even closed under general rewriting. (For T^4 this is vacuously true since T^4 contains no terms of the form SABC.) Finally we observe that $sxyzu \in T^4$. The statement now follows immediately from the fact that T^4 is closed under reduction and admits no root steps.

Figuur xx

3.1. OPGAVE. (i) Voer deze reducties voor B en C uit.
(ii) Een alternatieve definitie voor C is S(BS(BKS))(KK). (Hankin p.53)

3.2. OPGAVE. Hoe zien de normaalvormen van CL eruit? Geef een inductieve definitie.

3.3. OPGAVE. Geef de 'leftmost' reductie van de term Ω aan in de bovenstaande reductiegraaf.
3.4. OPGAVE. Noem een combinator (dat is een CL-term zonder varabelen) *plat* als hij geen zichtbare haakjes heeft. Bv. SKSKSKSK is plat.
(i) Bewijs dat platte combinatoren die alleen uit S-en en K's bestaan, een normaalvorm hebben. [Hint: bewijs eerst Opgave 4.4.]
(ii) Is dit ook zo voor platte combinatoren opgebouwd uit S, K en I?

*3.5. OPGAVE. Een S-term is een term die alleen uit S-en bestaat, bijv. SS(SSS)SSSS.
Bewijs dat er geen cyclische S-termen zijn.

*3.6. OPGAVE. Bewijs dat de S-term AAA met A ≡ SSS een oneindige reductie heeft.

3.7. OPGAVE. Hoeveel verschillende termen bevatten de volgende reeksen:
(i) K, KK, KKK, KKKK, KKKKK, ...
*(ii) S, SS, SSS, SSSS, SSSSS, ...
(iii) S, SK, SKS, SKSK, SKSKS, ...
(iv) K, KS, KSK, KSKS, KSKSK, ...

3.8. OPGAVE. (Uit Barendregt [84], Opgave van C.E. Schaap.)
Zij X ≡ SI. Bewijs dat XXXX = X(X(XX)), waarbij '=' convertibiliteit' in CL is.

3.9. OPGAVE. Zij W ≡ SS(KI).
(i) Bereken Wxy.
(ii) Reduceer WWW. Wat merk je op?

3.10. OPGAVE. We korten af: ω ≡ SII. Laat zien dat (S(KI)ω)(S(KI)ω) een cyclische reductie heeft.

3.11 OPGAVE. Een andere 'basis' dan {S, K, I} wordt gevormd door het viertal {B, C, W, K} ,en reductieregels

Babc → a(bc)
Cabc → acb
Wab → abb

Kab → a

Deze basis is 'equivalent' met de S, K, I - basis. Er geldt namelijk dat

S gedefinieerd kan worden als S ≡ B(B(BW)C)(BB). Ga dit na. Check het ook met John Tromp's combinator calculator.

3.12. OPGAVE. Vind een combinator P zodat Pxy ↠ y

3.13. OPGAVE. Vind een combinator *swap* zodat *swap* x y ↠ y x.

3.14. OPGAVE. Hoe hongerig zijn combinatoren? Vaak gebruiken we de beeldspraak van combinatore die hun argumenten (de termen waarop ze worden toegepast) 'opeten'. Bijvoorbeeld in de reductie Sxyz ↠ xz(yz) eet de S de drie variabelen op, om er iets mee te doen. De vraag is nu: als we een combinator een onbeperkt lange rij variabelen te eten geven, hoeveel kan hij er dan van opeten? De combinator I heeft maar een eetlust van 1; K van 2, S van 3.

(i) Laat zien dat er voor elke *n* een combinator is die een eetlust van minstens *n* heeft.
(iii) Bewijs dat een *S-term* (dat wil zeggen een term alleen uit S-en opgebouwd) een eetlust van maximaal 3 heeft.
(iv) Geef een term met *oneindige* eetlust.
(v) (Antwoord mij niet bekend.) Hoeveel kan een J-term eten?

OPMERKING. Variabele ariteit. De constanten S, K en I accepteren elk aantal 'argumenten'. Je zou kunnen zeggen dat ze een variabele ariteit hebben.

3.15. OPGAVE. (Singleton basis {X})

3.16. OPGAVE. CL bezit een bepaalde 'redundantie': er zijn voor een zelfde 'taak' vaak verschillende combinatoren (ook modulo convertibiliteit) die taak kunnen uitvoeren. Een voorbeeld is het volgende:
Laat zien dat S(KK)I hetzelfde doet als K.

3.17. OPGAVE. Er is een variant van CL, genaamd CL_I, die ook Turing compleet is, maar waarin niets 'weggegooid' kan worden (zoals de K zijn tweede argument weggooit). Voor deze sub-TRS van CL zijn er de volgende bases.

(i) De basis {I, S, C, B} met reductieregels als eerder gezien .
(ii) De basis {I, J} met reductieregels Ix → x, Jabcd → ab(adc)

Deze bases kunnen worden verkregen uit S, K, I; maar omgekeerd niet. Waarom niet?

3.18. OPGAVE. (Hankin p.54)

(i) Als $Y \equiv S(CB(SII))(CB(SII))$, dan $Yf = f(Yf)$. (Hankin p.54)

(ii) Als $Y_T \equiv B(SI)(SII)(B(SI)(SII))$, dan $Y_T f \twoheadrightarrow f(Y_T f)$.

(iii) jwk p.16: een andere $Y_T \equiv [S(K(SI))(SII)][S(K(SI))(SII)]$, dan $Y_T f \twoheadrightarrow f(Y_T f)$.

[under construction]

3.2. *Lambda calculus en CL.*

Om te begrijpen waarom CL zo krachtig is, dat wil zeggen 'Combinatorisch compleet' is en Turing compleet (alle berekenbare functies kunnen in CL beschreven worden), moeten we een verwant systeem invoeren, de lambda calculus. Dit doen we in de volgende versie van dit hoofdstuk.

3.4. *Fixed point combinators.*

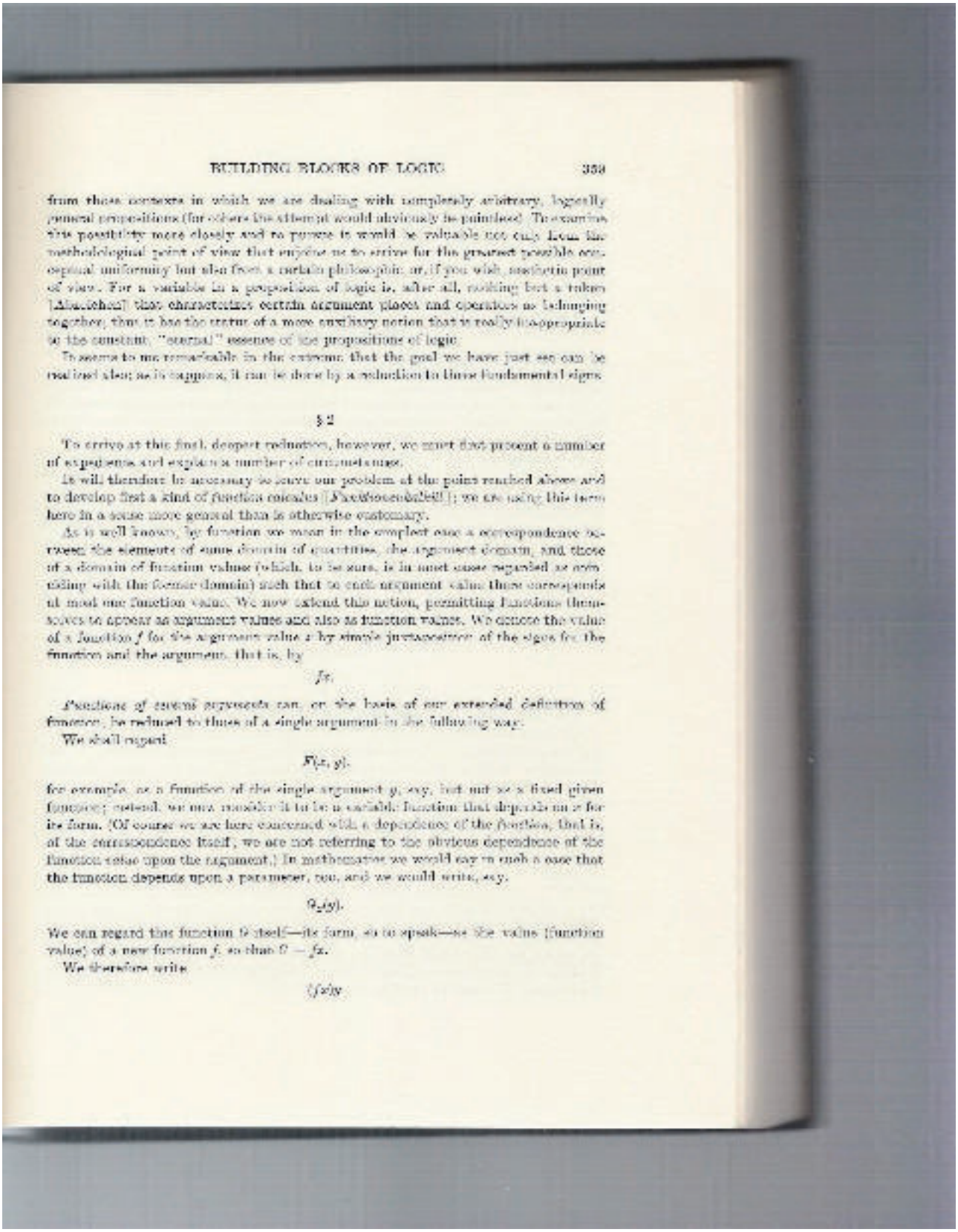
A fixed point combinator is a function which computes fixed points of other functions. A 'fixed point' of a function is a value left 'fixed' by that function; for example, 0 and 1 are fixed points of the squaring function. Formally, a value x is a fixed point of a function f if $f(x) = x$; a fixed point combinator is a function Y which, given another function f, computes a fixed point of f, so that $f(Y(f)) = Y(f)$ for all functions f.

In certain formalizations of mathematics, such as the lambda calculus and combinatorial calculus, *every* function has a fixed point. In these formalizations, it is possible to produce a function, often denoted **Y**, which computes a fixed point of any function it is given. Since a fixed point x of a function f is a value that has the property $f(x) = x$, a fixed point combinator **Y** is a function with the property that $f(\mathbf{Y}(f)) = \mathbf{Y}(f)$ for all functions f.

One well-known fixed point combinator, discovered by Haskell B. Curry, is
$\mathbf{Y} = \lambda f.(\lambda x.(f\ (x\ x))\ \lambda x.(f\ (x\ x)))$

and can be expressed in the SKI-calculus as
$\mathbf{Y} = S\ (K\ (S\ I\ I))\ (S\ (S\ (K\ S)\ K)\ (K\ (S\ I\ I)))$

The simplest fixed point combinator in the SK-calculus, found by John Tromp, is
$\mathbf{Y} = S\ S\ K\ (S\ (K\ (S\ S\ (S\ (S\ S\ K))))\ K$



BUILDING BLOCKS OF LOGIC 353

from those contexts in which we are dealing with completely arbitrary, logically general propositions (for where the attempt would obviously be pointless). To examine this possibility more closely and to pursue it would be valuable not only from the methodological point of view that enjoins us to strive for the greatest possible conceptual uniformity but also from a certain philosophic or, if you wish, aesthetic point of view. For a variable in a proposition of logic is, after all, nothing but a token [Abzeichen] that characterizes certain argument places and operations as belonging together; thus it has the status of a more auxiliary notion that is really inappropriate to the constant, "eternal" essence of the propositions of logic.

It seems to me remarkable in the extreme that the goal we have just set can be realized also; as is suggested, it can be done by a reduction to three fundamental signs.

§ 2

To arrive at this final, deepest reduction, however, we must first present a number of explanations and explain a number of circumstances.

It will therefore be necessary to leave our problem at the point reached above and to develop first a kind of function calculus [*Funktionenkalkül*]; we are using this term here in a sense more general than is otherwise customary.

As is well known, by function we mean in the simplest case a correspondence between the elements of some domain of quantities, the argument domain, and those of a domain of function values (which, to be sure, is in most cases regarded as coinciding with the former domain) such that to each argument value there corresponds at most one function value. We now extend this notion, permitting functions themselves to appear as argument values and also as function values. We denote the value of a function f for the argument value x by simple juxtaposition of the signs for the function and the argument, that is, by

$$fx.$$

Functions of several arguments can, on the basis of our *extended definition* of function, be reduced to those of a single argument in the following way.

We shall regard

$$F(x, y),$$

for example, as a function of the single argument y, say, but not as a fixed given function; instead, we may consider it to be a variable function that depends on x for its form. (Of course we are here concerned with a dependence of the *function*, that is, of the correspondence itself; we are not referring to the obvious dependence of the function value upon the argument.) In mathematics we would say in such a case that the function depends upon a parameter, too, and we would write, say,

$$\Phi_x(y).$$

We can regard this function Φ itself—its form, so to speak—as the value [function value] of a new function f, so that $\Phi = fx$.

We therefore write

$$(fx)y$$

existence that were defined above. I, C, T, Z, and S, are not mutually independent; that, rather, two of them, namely C and S, suffice to define the others. In fact, the following relations obtain here.

1. According to the definitions of the functions I and C,

$$Ix = x = Cxy$$

Since y is arbitrary, we can substitute any object or any function for it; hence, for example, Cx. This yields

$$Ix = (Cx)(Cx).$$

According to the definition of S, however, this means

$$SCCx,$$

so that we obtain

$$I = SCC.^{2}$$

The use of C, incidentally, does not matter in the expression SCC in an essential way. For, if above we put for y not Cx but an arbitrary function φx, we obtain in a similar way

$$I = S(C\varphi),$$

where any function can then be substituted for φ.[3]

2. According to the definition of Z,

$$Zfgx = f(gx).$$

Furthermore, by virtue of the transformations already employed,

$$f(gx) \quad (Cfx)(gx) \quad S(Cf)gx \quad (CSf)(Cf)gx.$$

Fusion over g yields

$$S(CSf)Cgx,$$

therefore

$$Z = S(CSf)C.$$

3. In an entirely analogous way,

$$Tfgx = fxg$$

can be further transformed thus:

$$fx(Cgx) = (fx)(Cgx) = Sf(Cy)x = (Sf)(Cy)x = Z(Sf)Cyx$$
$$= ZZ(Sf)Cyx = (ZZSf)Cyx = (ZZSf)(CC)yx = S(ZZS)(CC)fgx.$$

Therefore we have

$$T = S(ZZS)(CC).$$

If we here substitute for Z the expression found above, T too will have been reduced to C and S.

[2] This reduction was communicated to me by Richard[illegible]; even before then, Rosser had called the somewhat less simple one $(SC)(CC)$ to my attention.

[3] Only to such a function, of course, as has meaning for every x.

4

COMBINATOREN EN LAMBDA TERMEN

INHOUD.

3.0. *Introductie.* Om de werking van CL-termen beter te begrijpen, gaan we nu de lambda calculus bezien. We voeren eerst de termen in.

3.1. De termen van de λ-calculus. Net als in CL, is er de *binaire applicatie*. Als gewoonlijk bij een signatuur zijn er ook *variabelen* x, y, z, ... Behalve deze twee ingrediënten is er iets nieuws, namelijk *lambda abstractie* λx. Toegepast op een term M hebben we λx.M, met als betekenis: de functie die aan x de waarde M toekent. Bij het vormen van de abstractie-term λx.M wordt de variabele x gebonden, een nieuw fenomeen. Binding van variabelen treedt ook op bij de ∃ en ∀ quantor van de predicatenlogica, in formules ∃x.F(x) en ∀x.F(x). De verzameling van λ-termen Ter(λ) wordt nu zo verkregen:

(i) $x, y, z, \ldots \in Ter(\lambda)$

(ii) $M, N \in Ter(\lambda) \Rightarrow (MN) \in Ter(\lambda)$

(iii) $M \in Ter(\lambda) \Rightarrow (\lambda x.M) \in Ter(\lambda)$

Notatie.

- Vergeleken met het vorige is ook de 'punt' in λx.M nieuw. Dit is niet essentiëel; het is een restant van een oude notatievorm van Church, die naast een enkele punt ook een

dubbele punt en zelfs een driedubbele punt gebruikte om haakjes te besparen.

- Net als in CL gebruiken we de applicatieve notatie met associatie naar links. Maar nu komt daar nog iets bij: herhaalde abstracties ($\lambda x_1.(\lambda x_2.(\lambda x_3.$ ... $(\lambda x_n.M)$...) worden afgekort tot ($\lambda x_1 x_2 x_3 ... x_n.$ M)...). Dus hierbij geldt juist associatie naar *rechts*!

- Zeer belangrijk is het notatie-principe dat bekend staat als *α-conversie*, of *herbenoeming van gebonden variabelen*. Zo is $\lambda x.\ xx$ α-equivalent met $\lambda y.\ yy$, notatie: $\lambda x.\ xx \equiv_\alpha \lambda y.\ yy$.
En $\lambda x.\ (x\lambda x.\ xy) \equiv_\alpha \lambda z.\ (z\lambda x.\ xy)$. Om dit precies te definieren, moeten we weten wat *gebonden* en *vrije* variabelen zijn in een λ-term.

3.2. *De reductie van λ-termen.* Er is maar één reductieregel in de simpelste vorm van λ-calculus, de β-reductieregel. Deze zegt dat een *β-redex* $(\lambda x.Z)A$, dat is een instantie van de linkerkant van de β-regel, herschreven kan worden tot $Z[x :=A]$, dat wil zeggen Z met overal waar x vrij voorkomt in Z, de x vervangen door de argument term A. Hierbij heet $[x :=A]$ een substitutie-operator.

3.2.1. OPMERKING. De substitutie $[x :=A]$ is zelf niet een λ-term. Het is een 'meta-begrip', dat buiten de λ-calculus staat. Soms is het handig om de β-regel suggestief als volgt te schrijven: $(\lambda x.Z(x))A \rightarrow Z(A)$. In feite zullen we deze 'slordige', intuitieve notatie rechtvaardigen en precies maken bij het onderwerp 'hogere orde termherschrijfsystemen'.

$(\lambda x.Z)A \rightarrow Z[x :=A]$

Tabel 3.1. *De β-reductie regel*

Lambda abstractie en applicatie zijn in zekere zin elkaars inverse. Want:

$$(\lambda x.\ M)x = M.$$

Dit geldt voor elke λ-term M. We moeten dit officieel met inductie naar de opbouw van de term M bewijzen, maar voorlopig stellen we ons tevreden met de overtuiging die enkele voorbeelden geven:

$$(\lambda x.\ abxxb)x = abxxb$$
$$(\lambda x.\ ab(\lambda x.yx)xb)x = ab(\lambda x.yx)xb$$

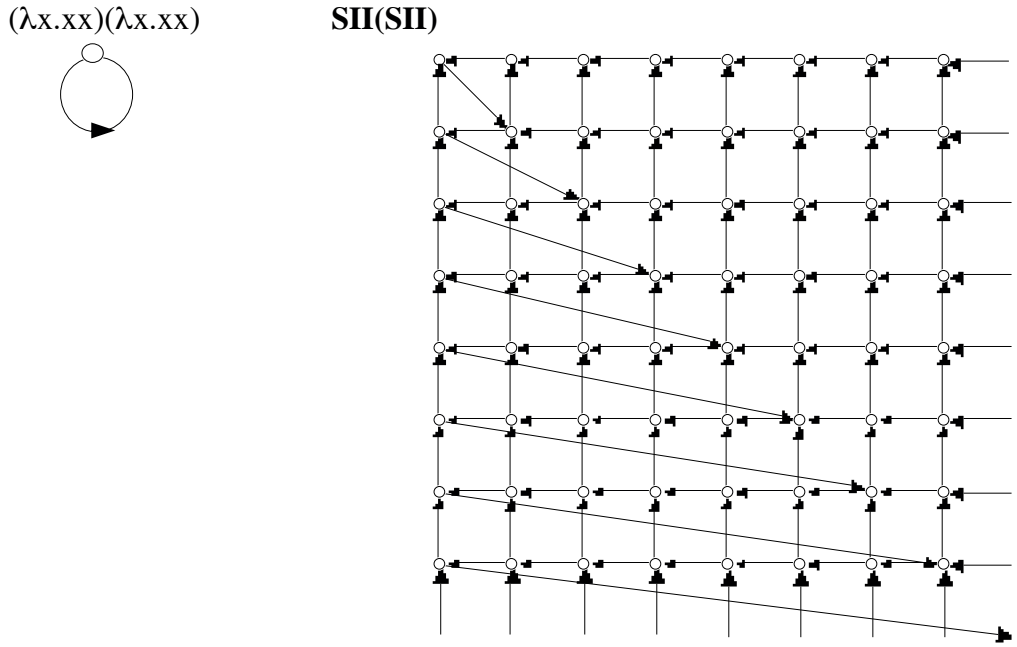
En voor herhaalde abstractie en applicatie dus

$(\lambda xyz.\ M)xyz = M,$

bijvoorbeeld $(\lambda xyz.\ xz(yz))xyz = xz(yz)$.

Hier zien we dat de links-associatieve applicatie precies past bij de rechts-associatieve abstractie.

Het is nu heel eenvoudig om vergelijkingen zoals bijvoorbeeld $Mxy = xyx$ op te lossen; neem $M \equiv \lambda xy.xyx$, dan hebben we niet alleen $Mxy = xyx$, maar sterker nog, $Mxy \twoheadrightarrow xyx$.



Figuur 4.1. *Reductiegraaf van (λx.xx)(λx.xx) en SII(SII).*

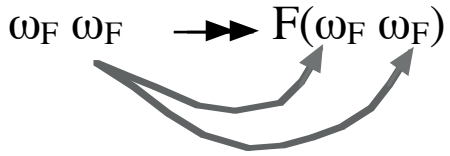
4.1. OPGAVE. DE term $(\lambda x.xx)(\lambda x.xx)$ is een *pure 1-cycle*, dat wil zeggen een term die in 1 stap naar zichzelf reduceert, en waarvan de reductiegraaf precies uit die ene stap bestaat. Geef ook een term die dat in precies n stappen doet, voor elke $n \geq 1$. Het is de bedoeling dat de hele reductiegraaf gevormd wordt door de cycle - er mogen geen andere reducten zijn. Zo'n term noemen we een pure n-cycle.
Opmerking. In CL bestaan er geen pure n-cycles, voor geen enkele n.

4.3. *Het bouwen van een fixed point combinator.*
Zij F een term. We zoeken een 'fixed point' van F, dat is een term X zodat $FX = X$. We doen dat als volgt. We proberen een term Y te construeren zodat $YF = F(YF)$. Dan is YF een fixed point als gezocht. Onze constructie zal zelfs 'uniform' werken voor elke F.

We proberen een term Ω_F te vinden, die van F af zal hangen (gesuggereerd door het

subscript), zodanig dat $\Omega_F \twoheadrightarrow M\Omega_F$. Stel nu dat $\Omega_F \equiv \omega_F\omega_F$, waarbij de eerste ω_F bedoeld is voor de nodige 'sturende actie' en de tweede ω_F bestemd is voor de 'passieve reconstructie' van de oorspronkelijke ω_F. Dus, intuitief, als in de volgende figuur; en dat leidt ertoe te eisen dat $\omega_F\, x \twoheadrightarrow F(xx)$. Daartoe nemen we $\omega_F \equiv \lambda x.F(xx)$. Dus kunnen we nemen

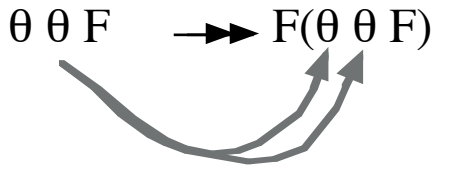
$$Y = \lambda f.\ \omega_f\omega_f \equiv \lambda f.\ (\lambda x.f(xx))(\lambda x.f(xx)).$$



Figuur 4.2. *The making of Curry's fixed point combinator*

Dit is *Curry's fixed point combinator*. Met een iets andere constructie vinden we *Turing's fixed point combinator* Θ die het voordeel heeft boven Curry's fixed point combinator Y dat $\Theta F \twoheadrightarrow F(\Theta F)$. (Voor Y hebben we alleen *convertibiliteit*, niet *reductie* in $YF = F(YF)$.)

De constructie gaat nu als volgt. Bij de constructie van Curry's fpc hakten we het probleem in tweeën. Dat doen we weer: $\Theta \equiv \theta\theta$. Dus we willen dat $\Theta F \equiv \theta\theta F \twoheadrightarrow F(\theta\theta F)$. Daarbij is de eerste θ de actief sturende en de tweede θ voor de replicatie. Dus willen we dat $\theta xf \twoheadrightarrow f(xxf)$. Dat kan simpel: neem $\theta \equiv \lambda xf.\ f(xxf)$ en tenslotte $\Theta \equiv$
$(\lambda xf.\ f(xxf))(\lambda xf.\ f(xxf))$, Turing's fpc.



Figuur 4.3. *The making of Turing's fixed point combinator*

Op deze manier kan iedereen zijn eigen fpc Γ maken. Begin met te stellen $\Gamma \equiv \gamma\gamma...\gamma$ ($n \geq 2$ keer) en voer een constructie uit als boven. Dan kunnen we voor elke keuze $\gamma \equiv \lambda a_1a_2...a_{(n-1)}f.\ f(wf)$ waarbij w een willekeurig woord is over het alfabet $\{a_1,...,a_{(n-1)}\}$ van lengte n, een fpc krijgen. Zo kunnen we bijvoorbeeld ook een fpc maken die zijn identiteit zelf verkondigt:

$$\mathbf{Y_{fpc}} \equiv (L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L\ L)$$

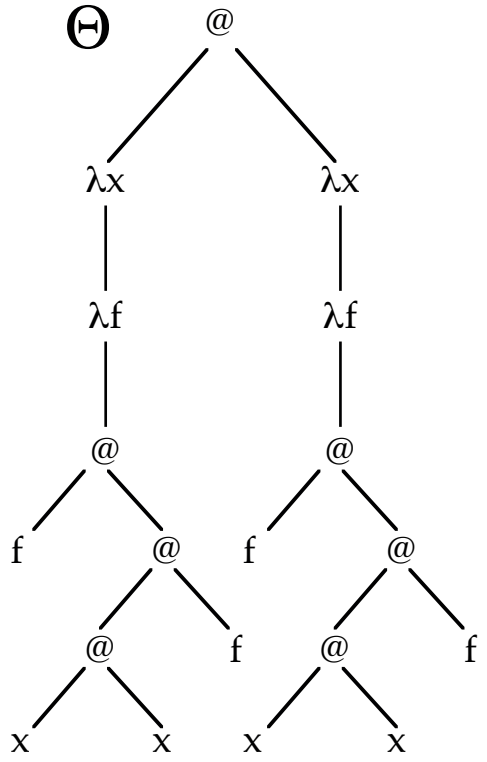
met L ≡ λabcdefghijklmnopqstuvwxyzr.(r (t h i s i s a f i x e d p o i n t c o m b i n a t o r))

OPGAVE. Er is een mooie manier om uit een fpc nieuwe fpc's te maken. Een fpc is namelijk per definitie een term Y met Yf = f(Yf). Equivalent: Y = [λyf.f(yf)]Y. Dat wil dus zeggen dat Y *zelf* een fixed point is, nl. van λyf.f(yf). Dus als we al een fpc Y′ hebben, dan is Y″ ≡ Y′λyf.f(yf) weer een fpc. Op deze manier krijgen we startend met Curry's fpc, een oneindige reeks fpc's, en deze blijken allemaal verschillend te zijn. Interessant is dat Turing's fixed point combinator Θ de tweede uit deze reeks is.

We kunnen met behulp van een fixed point combinator dit nog versterken. Stel dat we een vergelijking op willen lossen zoals Mxy = xMyxM, waar M zelf dus weer rechts voorkomt. Dit doen we als volgt:

$$
\begin{aligned}
&Mxy = xMyxM\\
&M = \lambda xy.\ xMyxM\\
&M = (\lambda m.\ (\lambda xy.\ xmyxm))M \equiv (\lambda mxy.\ xmyxm)M\\
&M \equiv \Theta(\lambda mxy.\ xmyxm)
\end{aligned}
$$

en nu hebben we niet alleen een conversie, maar een reductie Mxy ↠ xMyxM.



Figuur 4.4. Turing's fixed point combinator als boom.

OPGAVE. Beschrijf de reductie $\Theta F \twoheadrightarrow F(\Theta F)$ in boom-notatie.

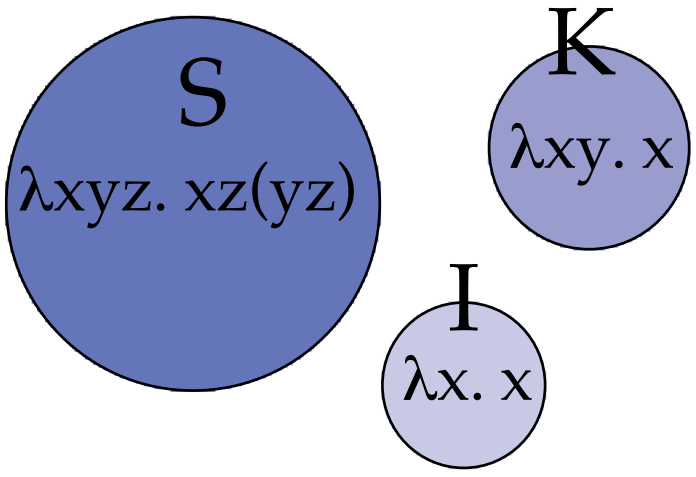
We hebben boven gezien dat de lambda term λabc. ac(bc) hetzelfde doet als de combinator S:

$$Sabc \rightarrow ac(bc)$$
$$(\lambda abc.\ ac(bc))abc \twoheadrightarrow ac(bc)$$

En ook voor K en I kunnen we gemakkelijk λ-termen vinden die hetzelfde doen. Een verschil is wel dat de λ-termen soms meer stappen nodig hebben, de combinatoren S en K doen hun werk in één stap. Met deze observatie hebben we een 'fijnstructuur' gevonden van de combinatoren: we kunnen nu een vertaling geven van CL naar λ-calculus die de reductie behoudt. Deze vertaling bestaat eenvoudig uit het vervangen in een CL-term van S door λabc. ac(bc), van K door λab. a en van I door λa.a. Als M een CL-term is, noemen we $(M)_\lambda$ het resultaat van de vertaling.
Iets preciezer:

DEFINITIE.

STELLING. $M \twoheadrightarrow N$ *in CL* $\Rightarrow (M)_\lambda \twoheadrightarrow (N)_\lambda$ *in λ-calculus*.



Figuur 4.5. *Fijnstructuur van de basis combinatoren*

De andere kant op is er ook een vertaling, die veel meer voeten in aarde heeft. Maar juist die kant op is de vertaling erg nuttig. zoals we straks zullen zien.

Vrije en gebonden variabelen. Zij x een variabele en M een λ-term. Met inductie naar de opbouw van M definieren we de plaatsen ('occurrences') waar x vrij voorkomt in M:

(i) $\phi_x(x) = \underline{x}$

$\phi_x(y) = y$ als niet $x \equiv y$

(ii) $\phi_x(MN) = \phi_x(M)\ \phi_x(N)$

(iii) $\phi_x(\lambda x.M) = \lambda x.M$

$\phi_x(\lambda y.M) = \lambda y.\phi_x(M)$ als niet $x \equiv y$.

VOORBEELD. $\phi_x\,(\lambda y.xx((\lambda x.xx)(yx))) =$

$\lambda y.\underline{x}\ \underline{x}((\lambda x.xx)(y\underline{x}))$. Dus ϕ_x onderlijnt de vrije plaatsen van x.

Een λ-term heet *gesloten*, als er geen variabele vrij in voorkomt. Een voorkomen van een variabele x in M, dat niet vrij is, heet een *gebonden* voorkomen. Een uitzondering vormen de voorkomens van x in λx, die zijn vrij noch gebonden. In het volgende voorbeeld zijn de vrije x-en door onderlijning aangegeven, en de gebonden x-en door dubbele onderlijning.

$$\lambda y.\underline{x}\ \underline{x}((\lambda x.\underline{\underline{x}}\ \underline{\underline{x}})(\underline{y}\ \underline{x}))$$

Nog duidelijker zou zijn om aan te geven door *welke* λ een voorkomen van x gebonden wordt. Dit doen we nu niet.

Substitutie. Voor elke variabele x en λ-term A hebben we een substitutie-operator σ_x, notatie [x := A], een afbeelding van Ter(λ) naar Ter(λ) die weer inductief gedefinieerd is:

(i) $\sigma_x(x) = A$

$\sigma_x(y) = y$ als niet $x \equiv y$

(ii) $\sigma_x(MN) = \sigma_x(M)\ \sigma_x(N)$

(iii) $\sigma_x(\lambda x.M) = \lambda x.M$

$\sigma_x(\lambda y.M) = \lambda y.\ \sigma_x(M)$ als niet $x \equiv y$.

Merk op dat deze definitie 'parallel loopt' met die van de vrije plaatsen van x in M. Dus de

afbeelding [x := A] substitueert A voor op alle plaatsen waar x vrij voorkomt in M. *Notatie:* we schrijven [x := A] vaak achter de term M waarop hij wordt toegepast: M [x := A]. Maar omgekeerd mag ook: [x := A] M.

Variable capture. Een vrije variabele x (beter: een vrij voorkomen van een variabele x) kan na een substitutie *'gevangen'* worden door een λx die die x kan 'zien'. Bijvoorbeeld:

(λy.yx) [x:=yy] = λy.y(**yy**). De twee **boldface** voorkomens van y zijn hier gevangen door de λy. Bij de definitie van β-reductie zijn we eraan voorbij gegaan, maar dat preciseren we nu, door het volgende af te spreken:

HERBENOEMINGSAFSPRAAK. *Bij het uitvoeren van een β-reductiestap mag geen variable capture optreden - om dat te vermijden passen we eerst zo nodig een of meerdere α-reductiestappen (herbenoeming van gebonden variabelen) toe.*

Opmerking. Bij een α-stap λx. M → λy. M [x:=y] is het het beste een geheel 'verse' variabele y te nemen - anders moeten we mogelijk in M nog andere α-stappen doen.

VOORBEELD. Wat kan er mis gaan als we tegen deze afspraak zondigen? Bekijk de term ω**1**, waarbij ω ≡ λx.xx en **1** ≡ λxy.xy (Church's numeral één).

Dan:

(λx.xx)(λxy.xy) →β
(λxy.xy)(λxy.xy) →β
λy. (λxy.xy)y →β
λy. (λy.y**y**).

De boldface **y** is hierbij gevangen. De laatste stap (in rood) is fout. Wat we hadden moeten doen is:

(λx.xx)(λxy.xy) →β
(λxy.xy)(λxy.xy) →β
λy. (λxy.xy)y →α
λy. (λxz.xz)y →β
λy. (λz.yz) ≡ λyz.yz.

Dit is de juiste reductie, met de groene α-stap.

$\lambda x.Mx \rightarrow M$ (*mits* x *niet vrij in* M)

Tabel 4.2. *De η-reductie regel*

$\lambda x.M \rightarrow \lambda y.M[x := y]$

Tabel 4.3. *De α-reductie regel*

4.2. Wat zijn de vrije variabelen van ((λx.(λy.yx)z(λx.yv)x)?

OPGAVE. (Corrado Böhm) Bepaal de reductiegraaf van de λ-term LLI met L ≡ λxy.x(yy)x.

OPGAVE. Zij W ≡ λxy.xyy. Teken de β-reductiegraaf van WWW.

OPGAVE. Teken de β-reductiegraaf van NNNN met N ≡ λabc.cbaa.

OPGAVE. Zij X ≡ λz.zKSK. Bewijs dat XXX = K en X(XX) = S. Omdat we elke λ-term kunnen uitdrukken in S en K, kunnen we dus elke λ-term, nog sterker, schrijven als een applicatieve combinatie van enkel X-en. Daarom heet {X} een *singleton-basis* voor de λ-calculus.

OPGAVE. Als M een CL-term is die een normaalvorm heeft, wil dat niet zeggen dat de λ-vertaling M_λ ook een normaalvorm heeft. Voorbeeld: M ≡ S(Kω)(Kω).

OPGAVE. Bewijs dat α-conversie niet nodig is, mits men niet 'onder een λ' reduceert.

OPGAVE. (John Tromp) Zij L ≡ S(K(SS(S(SSK))))K.
(i) Dan Lxy $\twoheadrightarrow$ x(SSKyx).
(ii) Zij Y_{john} ≡ SSKL. Dan $Y_{john}x \twoheadrightarrow x(Y_{john}\, x)$.
Y_{john} is de *kortste* fixed point combinator, aldus John Tromp.

OPGAVE. Definieer X = λx.xJAJCD, met

C = λabc.cab
D = λxy.xIIII(yE)
E = λxy.xIII(yIIII)
A = λxy.yIIIIx

Dan is XX = I en XI = J. Dus X is een singleton basis voor de I,J-versie van CL.

OPGAVE. Definieer X = λx.x(xS(KK))K, dan XXX = K en XK = S.

OPGAVE. (C. Grabmayer, M. van Handel) Zij C een combinator met $Cx_1...x_n \twoheadrightarrow x_1M_2...M_m$ met $m \leq n-2$ voor zekere $M_2,...,M_m$. Dan heeftYC onbeperkte eetlust. Merk op dat K en J deze eigenschap hebben, maar S niet.

4.4. We komen nu toe aan de vertaling van λ-termen in CL-termen. Dit gebeurt met een algoritme dat als input een λ-term krijgt, en als output een CL-term geeft. Onderweg is er een mengvorm, λ-termen met daarin ook S, K en I. Zulke 'hybride' termen zullen we λ-CL-termen noemen. We geven het algoritme in de vorm van een reductierelatie $\to_\tau$ op de verzameling van λ-CL-termen. Deze reductie is terminerend en confluent, zoals we later zullen zien, en daarmee zijn de eindresultaten uniek–met andere woorden, het algoritme is welgedefinieerd. We geven met τ(M) de vertaling volgens het τ-algoritme aan, dus de τ-normaalvorm van M.

(i)	$\lambda x.x \to_\tau$	I	
(ii)	$\lambda x.M \to_\tau$	KM	*als* x *niet vrij voorkomt in* M.
(iii)	$\lambda x.MN \to_\tau$	S(λx.M)(λx.N)	*als de vorige regel niet van toepassing is.*

VOORBEELD. $(\lambda x.xx)(\lambda x.xx) \to_\tau S(\lambda x.x)(\lambda x.x)(\lambda x.xx) \to_\tau \to_\tau SII(\lambda x.xx) \to_\tau \to_\tau \to_\tau SII(SII)$.

VOORBEELD. $\lambda xy.x \twoheadrightarrow_\tau S(KK)I$. Natuurlijk is het de bedoeling dat de vertaling 'hetzelfde doet' als de originele λ-term, en inderdaad:

$$S(KK)Ixy \to KKx(Ix)y \to K(Ix)y \to Ix \to x.$$

Toch is deze vertaling nog niet bevredigend. Want:

VOORBEELD. $\lambda xyz.xz(yz) \twoheadrightarrow_\tau$
S(S(KS)(S(KK)(S(KS)(S(S(KS)(S(KK)I))(KI)))))(K(S(S(KS)(S(KK)I))(KI))).
Is dit juist? Ja, want toegepast op xyz levert deze term inderdaad xz(yz) op; zie de reductie in de tabel. Maar er is een beter algoritme dat veel kortere vertalingen geeft. Dat algoritme verloopt via een reductierelatie $\to_{\tau'}$, die als volgt is gedefinieerd:

(i)	$\lambda x.x \to_{\tau'}$	I	
(ii)	$\lambda x.M \to_{\tau'}$	KM	*als* x *niet vrij voorkomt in* M.
(iii)	$\lambda x.Mx \to_{\tau'}$	M	*als* x *niet vrij voorkomt in* M.
(iv)	$\lambda x.MN \to_{\tau'}$	S(λx.M)(λx.N)	*als de vorige regels niet van toepassing zijn.*

De extra regel is dus de η-reductieregel. Het herziene algoritme is veel zuiniger. De vertaling van λxy.x is nu simpelweg K, en die van λxyz.xz(yz) is S!

OPGAVE. Ga dit na.

S(S(KS)(S(KK)(S(KS)(S(S(KS)(S(KK)I))(KI)))))(K(S(S(KS)(S(KK)I))(KI)))xyz
S(KS)(S(KK)(S(KS)(S(S(KS)(S(KK)I))(KI))))x(K(S(S(KS)(S(KK)I))(KI))x)yz
KSx(S(KK)(S(KS)(S(S(KS)(S(KK)I))(KI)))x)(K(S(S(KS)(S(KK)I))(KI))x)yz
S(S(KK)(S(KS)(S(S(KS)(S(KK)I))(KI)))x)(K(S(S(KS)(S(KK)I))(KI))x)yz
S(KK)(S(KS)(S(S(KS)(S(KK)I))(KI)))xy(K(S(S(KS)(S(KK)I))(KI))xy)z
KKx(S(KS)(S(S(KS)(S(KK)I))(KI))x)y(K(S(S(KS)(S(KK)I))(KI))xy)z
K(S(KS)(S(S(KS)(S(KK)I))(KI))x)y(K(S(S(KS)(S(KK)I))(KI))xy)z
S(KS)(S(S(KS)(S(KK)I))(KI))x(K(S(S(KS)(S(KK)I))(KI))xy)z
KSx(S(S(KS)(S(KK)I))(KI)x)(K(S(S(KS)(S(KK)I))(KI))xy)z
S(S(S(KS)(S(KK)I))(KI)x)(K(S(S(KS)(S(KK)I))(KI))xy)z
S(S(KS)(S(KK)I))(KI)xz(K(S(S(KS)(S(KK)I))(KI))xyz)
S(KS)(S(KK)I)x(KIx)z(K(S(S(KS)(S(KK)I))(KI))xyz)
KSx(S(KK)Ix)(KIx)z(K(S(S(KS)(S(KK)I))(KI))xyz)
S(S(KK)Ix)(KIx)z(K(S(S(KS)(S(KK)I))(KI))xyz)
S(KK)Ixz(KIxz)(K(S(S(KS)(S(KK)I))(KI))xyz)
KKx(Ix)z(KIxz)(K(S(S(KS)(S(KK)I))(KI))xyz)
K(Ix)z(KIxz)(K(S(S(KS)(S(KK)I))(KI))xyz)
Ix(KIxz)(K(S(S(KS)(S(KK)I))(KI))xyz)
x(KIxz)(K(S(S(KS)(S(KK)I))(KI))xyz)
x(Iz)(K(S(S(KS)(S(KK)I))(KI))xyz)
xz(K(S(S(KS)(S(KK)I))(KI))xyz)
xz(S(S(KS)(S(KK)I))(KI)yz)
xz(S(KS)(S(KK)I)y(KIy)z)
xz(KSy(S(KK)Iy)(KIy)z)
xz(S(S(KK)Iy)(KIy)z)
xz(S(KK)Iyz(KIyz))
xz(KKy(Iy)z(KIyz))
xz(K(Iy)z(KIyz))
xz(Iy(KIyz))
xz(y(KIyz))
xz(y(Iz))
xz(yz)

Nu is het helaas niet zo dat de τ-vertaling β-reductie omzet in CL-reductie. Bijvoorbeeld

$\lambda x.(\lambda y.y)(xx) \rightarrow \lambda x.xx$, maar voor de vertalingen hebben we niet $S(KI)(SII \rightarrow SII$.

Het probleem blijkt te zitten bij reductiestappen waarbij een redex herschreven wordt dat *'onder een λ'* zit, dat wil zeggen, subterm is van een λx.M. Reducties zonder 'sub-λ' stappen, vertalen wel goed. Een voorbeeld van een reductie zonder sub-λ stappen is *head-*

reductie. Daarbij wordt telkens het head-redex herschreven. Een redex heet *head-redex* of kop-redex als het helemaal aan de linkerkant van de term zit. Bijv. in $\underline{(\lambda x.xx)(\lambda x.xx)}y$ is het onderstreepte redex het head-redex maar in $y(\underline{(\lambda x.xx)(\lambda x.xx)})$ is het onderstreepte redex *niet* een head-redex.

PROPOSITIE. *Als* $M \twoheadrightarrow N$ *een head-reductie is in* λ*-calculus, dan* $\tau(M) \twoheadrightarrow \tau(N)$ *in* CL. *Idem voor* τ' *in plaats van* τ.

STELLING. (*Combinatorische compleetheid van* CL.)
Zij $M(x_1,...,x_n)$ *een* CL*-term waarin de variabelen* $x_1,...,x_n$ *mogen voorkomen. Dan is er een* CL-term N *zodat*

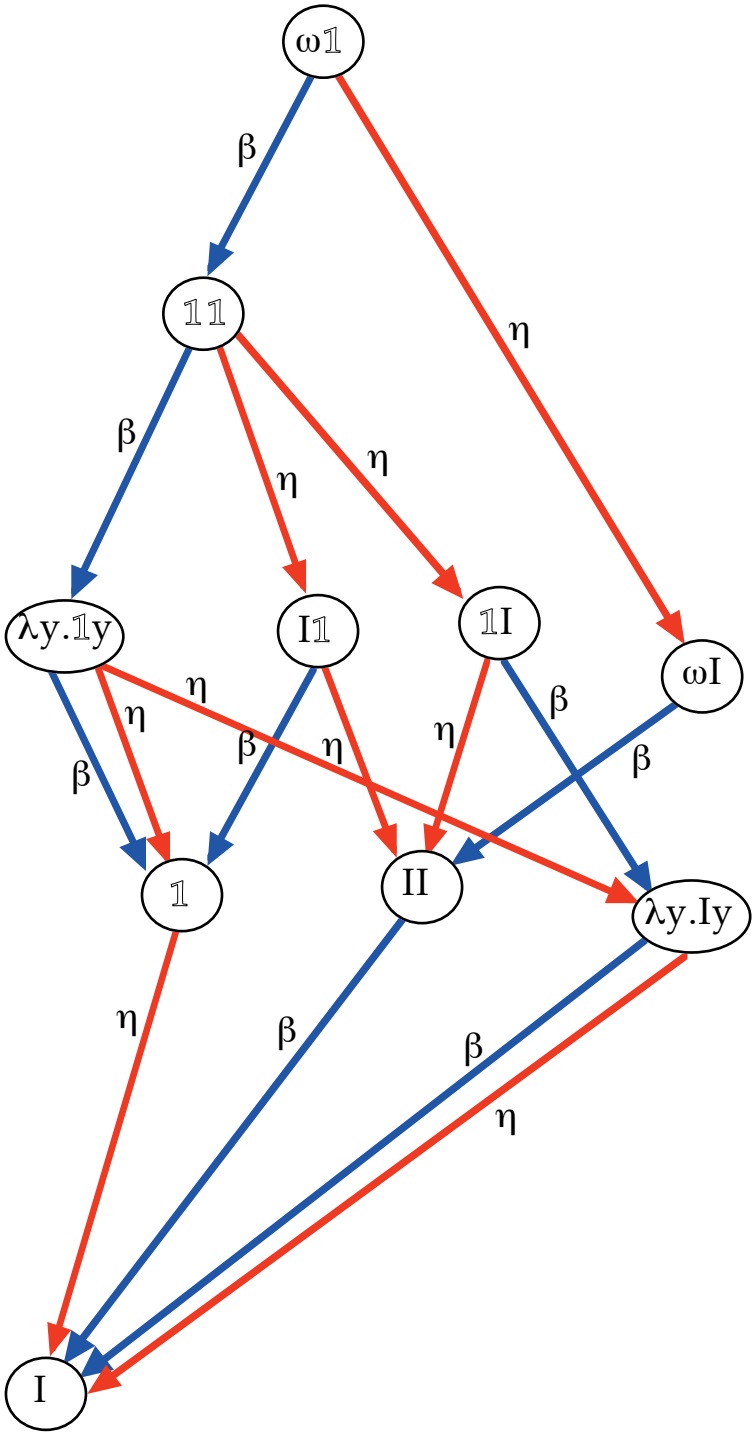
$$Nx_1...x_n \twoheadrightarrow_{CL} M(x_1,...,x_n) .$$

BEWIJS. Neem de λ-CL-term $N' \equiv \lambda x_1...x_n.\ M(x_1,...,x_n)$. Dan hebben we een head-reductie

$$N'\ x_1...x_n \twoheadrightarrow_{\beta} M(x_1,...,x_n).$$

Dus $\tau(N'\ x_1...x_n) \twoheadrightarrow_{CL} \tau(M(x_1,...,x_n)) \equiv M(x_1,...,x_n)$.

De laatste '$\equiv$' is omdat M geen λ's bevat. Neem nu $N \equiv \tau(N')$. □



Figuur 4.6. De βη-reductiegraaf van (λx.xx)(λxy.xy)

OPGAVE. *Vind een CL-term* N *zodat* Nxy ↠ xNy.

OPLOSSING. Als we een λ-term N kunnen vinden met N ↠ λxy.xNy, zijn we een heel eind op weg. Die is gemakkelijk te vinden:

Neem N ≡ Θ(λnxy.xny).

We vertalen dit nu in CL. In feite zullen we Θ niet vertalen, maar daarvoor meteen Tromp's fpc gebruiken. Deze is SSKL met L = S(K(SS(S(SSK))))K. Het blijkt iets handiger te zijn om niet SSKL te nemen, maar zijn eenstapsreduct SL(KL). Dan hebben we met .po reductie in Wiedijk's reducer namelijk de typische fpc reductie:

SL(KL)x
SL(KL)x
Lx(KLx)
S(K(SS(S(SSK))))KxL
K(SS(S(SSK)))x(Kx)L
SS(S(SSK))(Kx)L
S(Kx)(S(SSK)(Kx))L
KxL(S(SSK)(Kx)L)
x(SSKL(KxL))
x(SL(KL)x) .

Vervolgens bepalen we de tau' vertaling van λnxy.xny; die is M=S(K(SI))K.

Met de reducer hebben we nu, gebruik makend van de afkortingen voor L en M als boven:

```
.c
.po
L=S(K(SS(S(SSK))))K
   L=S(K(SS(S(SSK))))K
M=S(K(SI))K
   M=S(K(SI))K
```

(De reducer herhaalt de afkortingen een keer.)

De bewering is nu dat SL(KL)M de gezochte N is. We checken dit met de reducer door er xy achter te zetten:

Inderdaad reduceert SL(KL)Mxy met .po (*parallel outermost reductie*) keurig naar:

SL(KL)Mxy
SL(KL)Mxy
LM(KLM)xy
S(K(SS(S(SSK))))KMLxy

K(SS(S(SSK)))M(KM)Lxy
SS(S(SSK))(KM)Lxy
S(KM)(S(SSK)(KM))Lxy
KML(S(SSK)(KM)L)xy
M(SSKL(KML))xy
S(K(SI))K(SSKL(KML))xy
K(SI)(SSKL(KML))(K(SSKL(KML)))xy
SI(K(SL(KL)M))xy
Ix(K(SL(KL)M)x)y
x(SL(KL)M)y .

Here's another way of interpreting the pure lambda calculus for arithmetic, called Church numerals. Church numerals have a particularly simple form when expressed in the textual notation.
zero = λ*fx*.*x* = KI
succ = λ*afx*.*f*(*afx*) = SB
So
one = λ*fx*.*fx*
two = λ*fx*.*f*(*fx*)
three = λ*fx*.*f*(*f*(*fx*))
etc.
add = λ*abfx*.*af*(*bfx*)
multiply = λ*abf*.*a*(*bf*)
power = λ*ab*.*ab*

In telescoopsectie:
over Bohm's stelling, p.37 hindley-seldin

he construction can be made more efficient by adding extra combinators **B** and **C** (first introduced by DavidTurner):

(**B** x y z) -> (x z y)

(**C** x y z) -> (x (y z))

, and extra mapping:

- \x.(p q) = (**B** \x.p q) if x does not occur free in q
- \x.(p q) = (**C** p \x.q) if x does not occur free in p

Using Turner's combinators, the derivation above goes like this:

2
= \f.\x.(f (f x))
= \f.(C f \x.(f x))

```
= \f.(C f f)
= (S \f.(C f)) \f.f)
= (S C I)
```

And indeed, (S C I f x) reduces to (f (f x)):

```
  (S C I f x)
= (C f (I f) x)   (reduce the S)
= (f (I f x))     (reduce the C)
= (f (f x))       (reduce the I)
```

```
I=\x.x
K=\xy.x
S=\xyz.xz(yz)
B=\xyz.x(yz)
C=\xyz.xzy
W=\xy.xyy
1=\xy.xy
Y=\f.(\x.f(xx))\x.f(xx)
T=\xy.x
F=\xy.y
D=\x.xx
J=\abcd.ab(adc)
C'=JII
  .li leftmost innermost
  .lo leftmost outermost [default]
  .po parallel outermost
  .gk gross knuth
  .l lambda reduction [default]
  .c combinator reduction
  .ex eta reduction
  .in no eta reduction [default]
  ./ fold combinators
  ./IKS translate to IKS
  .//IKS translate to IKS using eta
  ./IKBCS translate to IKBCS
  .//IKBCS translate to IKBCS using eta
  ./IJK translate to IJK
  .//IJK translate to IJK using eta
  .. normalize
  .<< previous input term
  .< previous term
  .> next term
  .>> next input term
  .? help
  . exit
\ab.aab
 \ab.aab .//IJK
```

J(JII)(JII)(JI(J(JII)(JII)(J(JII)I(J(JII)IJ))))

Opgave. Turing's en Curry's fpc in \-calculus hebben geen normaalvorm.
Is dat zo voor elke fpc in \?
Na vertaling in CL hebben deze termen wel een normaalvorm.
is er in CL ook een fpc zonder normaalvorm?
(Ja, stop een omega in een I.)
Welke \-termen geven een normaalvorm na vertaling?

5

CONFLUENTIE

INHOUD.

5.0. *Introductie.*

In dit Hoofdstuk bekijken we de *confluentie* eigenschap van TRSen, die samen met terminatie de belangrijkste eigenschap van TRSen is. We hebben confluentie nodig om te garanderen dat normaalvormen uniek zijn. Ook verder is confluentie een hoeksteen van de hele theorie van termherschrijven. Er zijn diverse wegen naar confluentie. In dit hoofdstuk geven we een aanzet tot de drie belangrijkste wegen.

De eerste is via *abstract herschrijven*, met name via het Lemma van Newman.

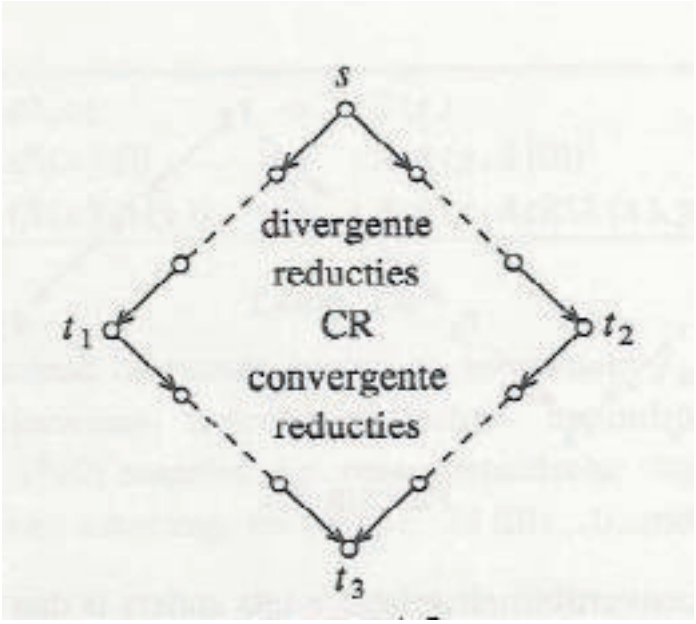
De tweede route naar confluentie is via een *modulaire* aanpak: we verdelen daarbij de TRS in kleinere stukken en bewijzen dan confluentie voor de afzonderlijke delen.

De derde hoofdweg is die van het *orthogonaal herschrijven*, waarvan we in het vorige hoofdstuk twee belangrijke vertegenwoordigers zagen: lambda calculus en CL.

5.1. *Newman's Lemma.*

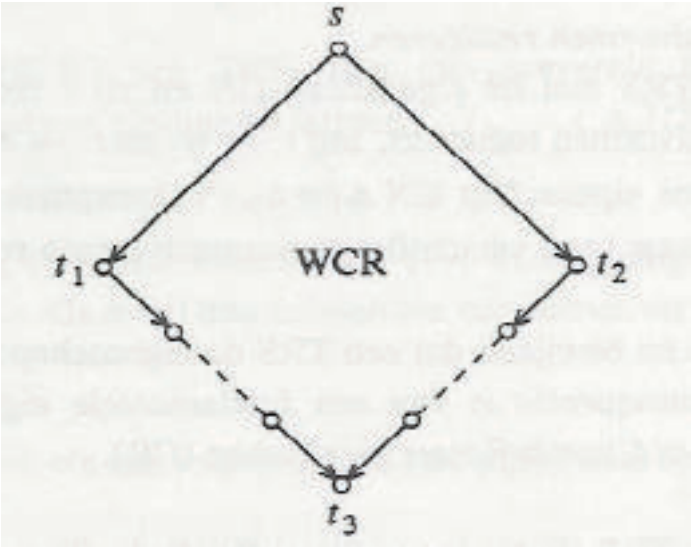
We beginnen met de formele definitie van de confluentie of Church-Rosser eigenschap.

5.1.1. DEFINITIE. Een TRS (Σ, R) is *confluent* of *heeft de Church-Rosser eigenschap* (*is CR*) als er voor elke twee 'divergente' reducties $s \twoheadrightarrow t_1$, $s \twoheadrightarrow t_2$ 'convergente' reducties $t1 \twoheadrightarrow t_3$ en $t_2 \twoheadrightarrow t_3$ zijn. (Figuur 5.1). De convergente reducties mogen uit 0 stappen bestaan.



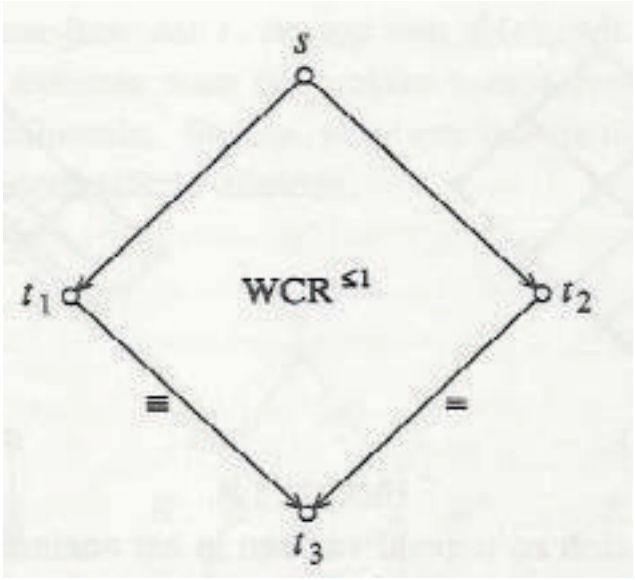
Figuur 5.1. *De eigenschap CR.*

Een belangrijke zwakkere eigenschap is WCR ('weak Church-Rosser', zwakke confluentie).



Figuur 5.2. *De eigenschap WCR.*

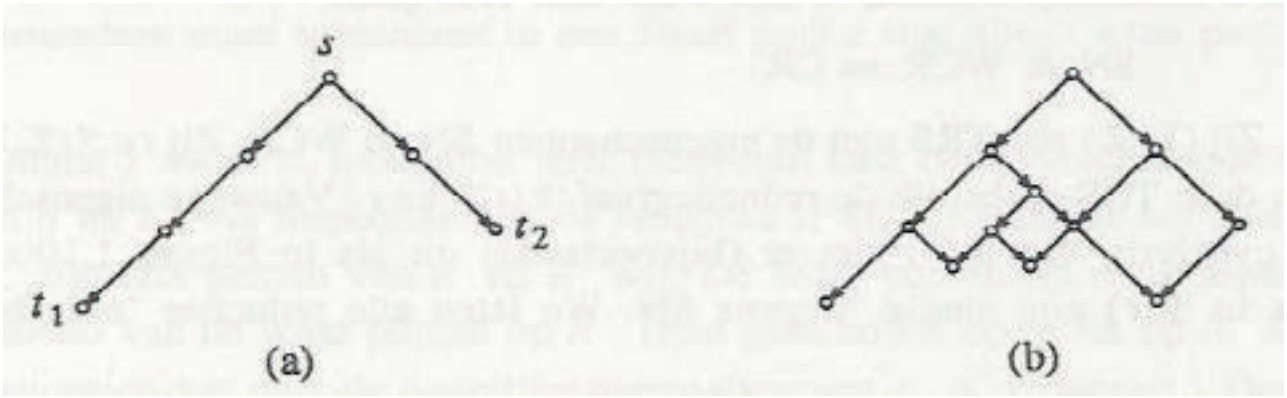
5.1.2. DEFINITIE. (i) Een TRS (Σ, R) is *zwak confluent* (WCR) als er voor elke twee divergente *éénstapsreducties* $s \to t_1$, $s \to t_2$ convergente reducties $t_1 \twoheadrightarrow t_3$ en $t_2 \twoheadrightarrow t_3$ zijn. (Figuur 5.2). De convergente reducties mogen willekeurige lengte hebben.



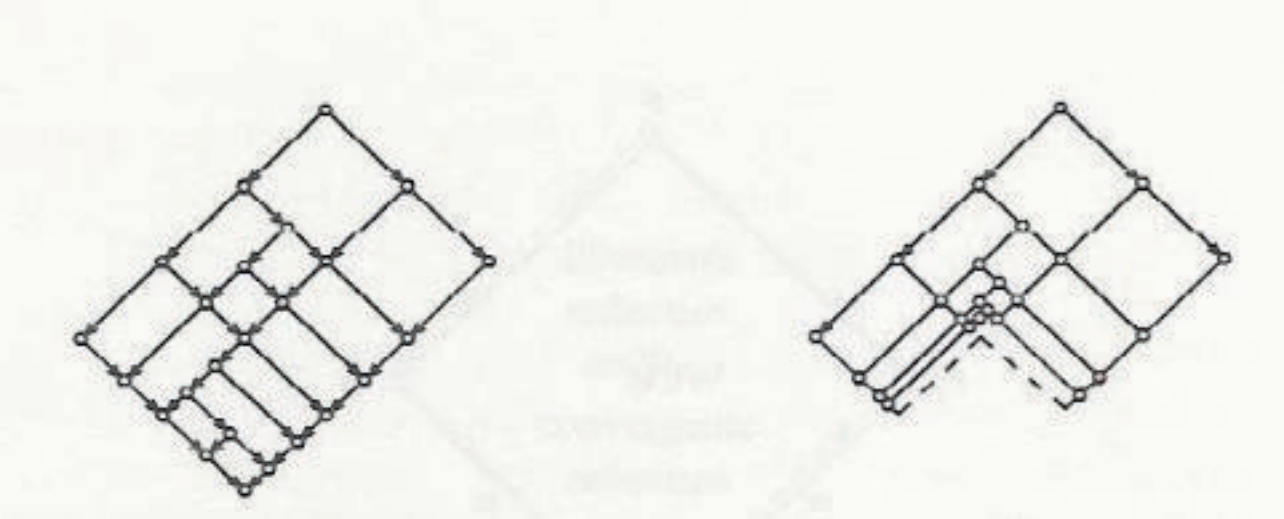
Figuur 5.3. *De eigenschap WCR.*

(ii) Als bovendien altijd convergente reducties van *nul of één stappen* gevonden kunnen worden, dan zeggen we dat de TRS (Σ, R) de eigenschap $WCR^{\leq 1}$ heeft (Figuur 5.3).
Hierbij is $\rightarrow^{=}$ de reflexieve afsluiting van $\rightarrow$, dat wil zeggen: $s \rightarrow^{=} t \Leftrightarrow s \rightarrow t$ of $s \equiv t$.

Het is duidelijk dat CR $\Rightarrow$ WCR voor elke TRS. Maar het omgekeerde geldt niet. We kunnen dit als volgt inzien, of althans vermoeden. Zij (Σ, R) een TRS die WCR is, en laten we proberen te bewijzen dat (Σ, R) ook CR is. We bekijken daartoe divergente reducties $s \twoheadrightarrow t_1$ en $s \twoheadrightarrow t_2$. De eigenschap WCR nodigt uit het diagram beginnend met (a) van Figuur 5.4 op te vullen met 'elementaire diagrammen', zoals gegeven door WCR, als in Figuur 5.4(b.

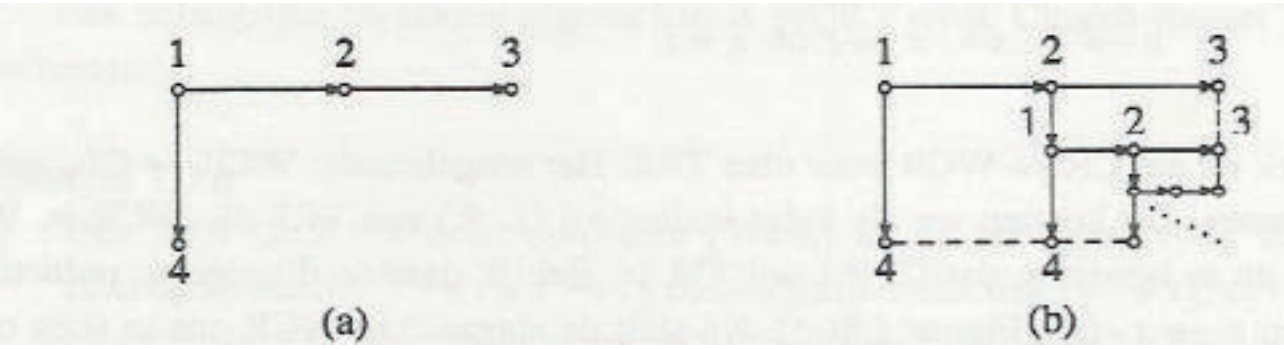


Figuur 5.4. *Betegelen met elementaire diagrammen.*

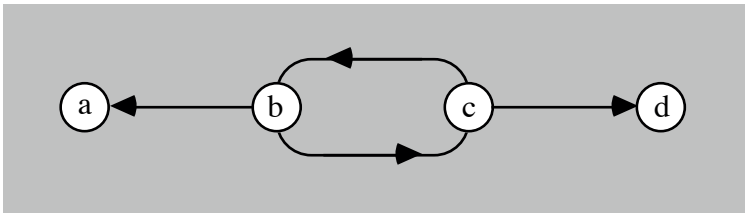


Figuur 5.5. *Geslaagde en niet geslaagde betegeling.*

Dit betegelen met elementaire diagrammen zou wel succesvol kunnen eindigen zoals in Figuur 5.5(links), maar dat hoeft niet. We zouden ook een oneindig diagram kunnen krijgen zoals in Figuur 5.5(rechts). Er is inderdaad een TRS waarbij het betegelen in het oneindige wegloopt; zie Figuur 5.6. Hierbij is TRS gebruikt die in Figuur 5.7 is afgebeeld; er zijn vier constanten en reductieregels als in de Figuur af te lezen. (*Correctie:* Lees in Figuur 5.6 i.p.v. 1, 2, 3, 4 respectievelijk a, b, c, d.)



Figuur 5.6. *Weglopende diagramconstructie.*



Figuur 5.7. *Tegenvoorbeeld tegen WCR ⇒ CR.*

Figuur 5.7 stelt eigenlijk een *Abstract Reductie Systeem (ARS)* voor, een notie waar we later op terug komen; maar we kunnen er ook een TRS in 'zien', namelijk de TRS met $\Sigma = \{a, b,$

c, d} (vier constanten) en verzameling reductieregels R = {b → a, c → d, b → c en c → b}.

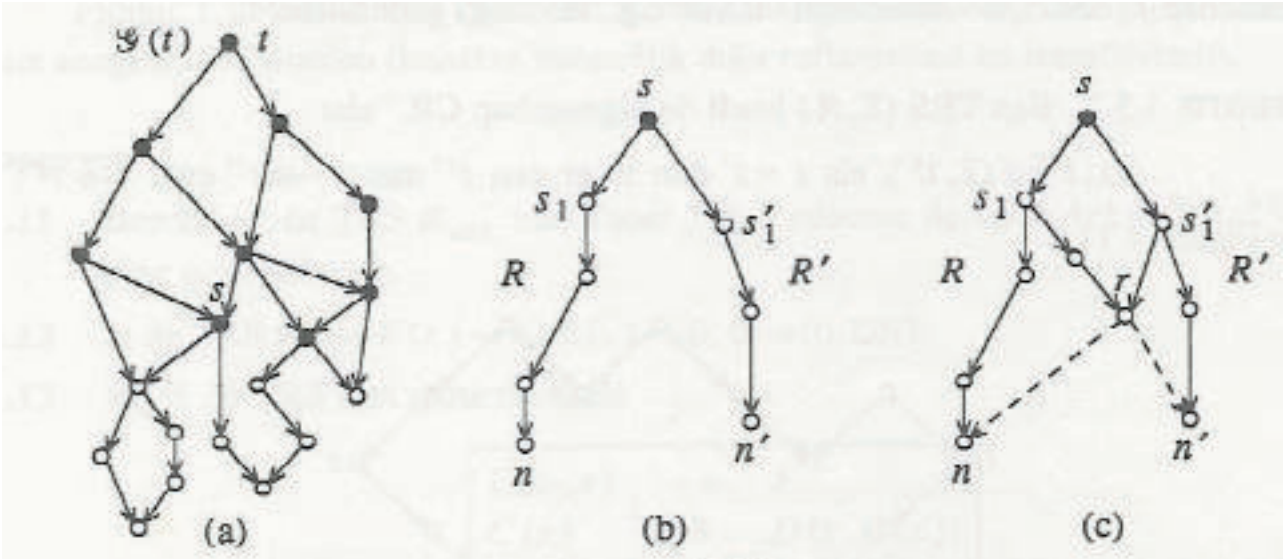
We hebben dus gezien dat niet elke zwak confluente TRS confluent is. Maar we hebben wel een vermoeden gekregen wat eraan schort. Wat experimenteren leert, dat altijd wanneer het betegelen vanuit het gegeven WCR om tot CR te komen mislukt, er *oneindige reducties* ontstaan. Als er geen oneindige reducties zijn (dit is de eigenschap SN) dan zou het goed moeten gaan. Inderdaad:

5.1.3. STELLING. (Newman's Lemma, 1942).
Voor elke TRS *geldt:*

$$SN \,\&\, WCR \Rightarrow CR.$$

BEWIJS. Zij (Σ, R) een TRS met de eigenschappen SN en WCR. We zullen CR indirect bewijzen: eerst bewijzen we $UN^{\rightarrow}$. Zoals we gezien hebben, levert dit samen met SN (zelfs al met WN) ook CR op.

Zij $t \in Ter(\Sigma, R)$ en bekijk de reductiegraaf $\mathcal{G}(t)$.



Figuur 5.8. *Bewijs van Newman's Lemma.*

Vanwege eigenschap SN is $\mathcal{G}(t)$ cykelvrij, dus $\mathcal{G}(t)$ ziet er (bijvoorbeeld) uit als in Figuur

5.8(a). Alle reducties in $\mathcal{G}(t)$ zijn eindig, wegens SN. We laten alle reducties 'naar beneden lopen'.

Knopen in $\mathcal{G}(t)$ zullen wit of zwart gekleurd worden: $s \in \mathcal{G}(t)$ is zwart als s naar twee of meer verschillende normaalvormen reduceert, en wit anders. In Figuur 5.8(a) is de wit-zwart kleuring aangebracht. Merk op dat de eindpunten van de reducties in $\mathcal{G}(t)$, dat wil zeggen de normaalvormen van t, wit moeten zijn.

Als we kunnen bewijzen dat t, de top van G(t), wit is zijn we klaar. Want dan reduceren alle reducten naar één unieke normaalvorm, dus wat betreft de reducties vanuit t is er dan confluentie. Stel nu, voor een bewijs uit het ongerijmde, dat t zwart is. We willen een contradictie afleiden.

Merk eerst op dat 'zwart' naar boven toe 'vererft': een punt boven een zwart punt is zelf zwart. We proberen nu een onderste zwart punt, zeg s, te vinden. Dat wil zeggen, een punt s waarbeneden alleen witte punten liggen (zie Figuur 5.8(b). We kunnen een onderste zwart punt vinden op de volgende manier:

(1) t *is zwart;*
(2) *als* t *alleen witte punten onder zich heeft, dan zijn we klaar;*
(3) *zo niet, kies dan een zwart punt* t' *onder* t *en herhaal de procedure met* t' *in plaats van* t.

Deze procedure moet termineren in een zwart punt s met alleen witte punten onder zich.

Omdat s zwart is, moet deze term reduceren naar twee verschillende normaalvormen n en n', via respectievelijk de reducties R en R'. Behalve het zwart beginstuk s, zijn alle punten van R en R' wit. De witte punten op reductiepad R zijn verschillend van de witte punten op R'. (Een punt zowel op R als op R' zou zwart zijn, omdat het naar de ongelijke normaalvormen n, n' reduceert.) Dus hebben we de situatie van Figuur 5.8(b).

Volgens eigenschap WCR hebben s_1 en s_1' een gemeenschappelijk reduct r (zie Figuur 5.8(c)). Omdat s_1 wit is, moet r naar n reduceren. Omdat s_1' wit is, moet r naar n' reduceren. Dus heeft r twee verschillende normaalvormen, n en n'. Dus r is zwart. Maar dan zijn ook de hogere punten s_1, s_1' zwart. Tegenspraak.

Alternatief bewijs van Newman's Lemma. Het nu volgende alternatieve bewijs is veel compacter dan het bovenstaande.

Noem een element *goed* als het precies één normaalvorm heeft, *slecht*, als het er

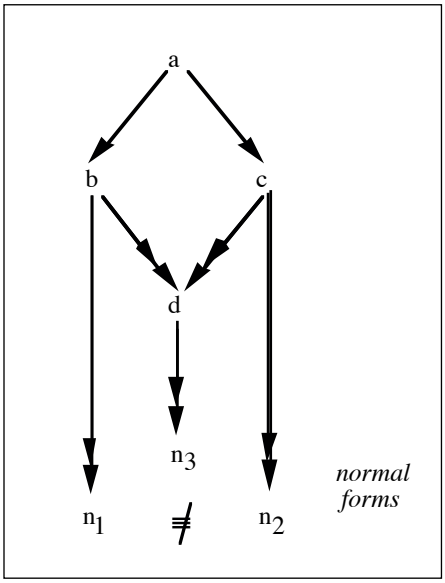
meer dan één heeft. We zullen bewijzen:

(*Claim*) *Een slecht punt heeft een éénstapsreduct dat weer slecht is.*

Dus stel 'a' is slecht, en reduceert naar verschillende normaalvormen n_1 en n_2:

$a \to b \to ... \to n_1$ en
$a \to c \to ... \to n_2$.

Als $b \equiv c$ is de Claim bewezen: b is slecht. Anders passen we WCR toe op de divergente stappen $a \to b$, $a \to c$ met als resultaat een reduct d zodat $b \twoheadrightarrow d$ en $c \twoheadrightarrow d$. (Figuur 5.9.)



Figuur 5.9. *Alternatief bewijs van Newman's Lemma.*

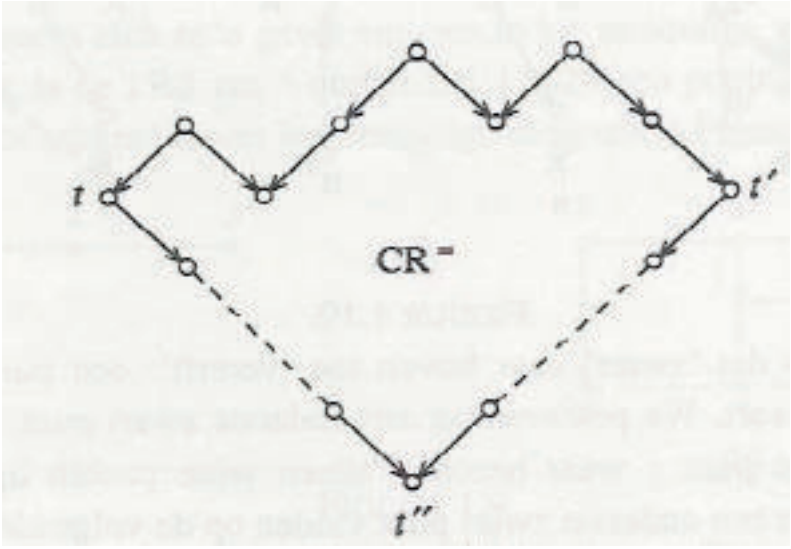
Wegens SN kunnen we d naar een normaalvorm n_3 reduceren. Omdat n_1 en n_2 verschillend zijn, is n_3 ofwel verschillend van n_1, ofwel van n_2. In het eerste geval is b slecht, in het tweede geval is c slecht. In elk geval heeft a dus een slechte eenstapsopvolger, waarmee de Claim bewezen is. Met de Claim kunnen we een oneindige reductierij van slechte punten vinden; maar dat is tegenspraak met SN.

In de volgende stelling vatten we een aantal betrekkingen tussen de eigenschappen WN, SN, UN, NF, $WCR^{\leq 1}$, WCR, CR en $CR^{=}$ samen. De laatste eigenschap is een equivalente variant van CR, als volgt gedefinieerd:

5.1.4. DEFINITIE. Een TRS (Σ, R) heeft de eigenschap $CR^{=}$ als:

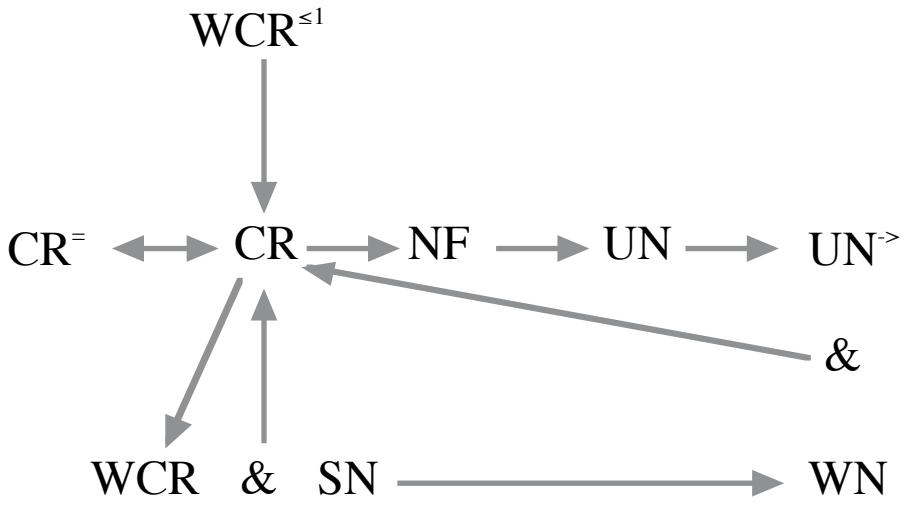
$\forall t, t' \in Ter(\Sigma, R)$: als $t = t$ dan is er een t'' met $t \twoheadrightarrow t''$ en $t' \twoheadrightarrow t''$.

(Zie Figuur 5.10.)



Figuur 5.10. *Zigzag versie van CR.*

5.1.5. STELLING. *Voor elke* TRS *gelden de volgende implicaties:*



Figuur 5.11. *Relaties tussen diverse TRS eigenschappen.*

BEWIJS. SN & WCR ⇒ CR is Newman's Lemma. De implicaties CR ⇒ WCR, SN ⇒ WN en $CR^{=}$ ⇒ CR zijn triviaal. De overige implicaties worden als opgave gelaten.

Figuur 5.11 bevat alle algemeen geldende implicaties–meer pijlen kunnen niet aangebracht worden (behalve natuurlijk door reflexiviteit en transitiviteit).

5.1.6.1. OPGAVE. Is de TRS $\{F(x) \to x, F(x) \to 0, 0 \to 0\}$ UN?

5.1.6.2. OPGAVE. Zij $\mathcal{R}$ de TRS met reductieregels

$D(x, x) \to E$
$C(x) \to D(x, C(x))$
$A \to C(A)$

Tabel 5.1.

a) Bewijs dat $\mathcal{R}$ WCR is.

b) Bewijs dat $C(A) \twoheadrightarrow E$ en $C(A) \twoheadrightarrow C(E)$.

c) Laat zien dat $\mathcal{R}$ niet CR is, door de termen E en C(E) te bekijken.

d) Bewijs dat $\mathcal{R}$ UN is.

5.1.6.3. OPGAVE. Bewijs dat voor elke TRS de volgende implicaties gelden:

a) $CR \Rightarrow UN$

b) $UN \;\&\; WN \Rightarrow CR$

c) $WCR^{\leq 1} \Rightarrow CR$

d) $CR \Rightarrow CR^{=}$.

5.1.6.4. OPGAVE. Construeer een TRS die wel WCR en WN is, maar niet CR.

5.1.6.5. OPGAVE. Een TRS (Σ, R) heeft de eigenschap $UN^{\to}$ als:

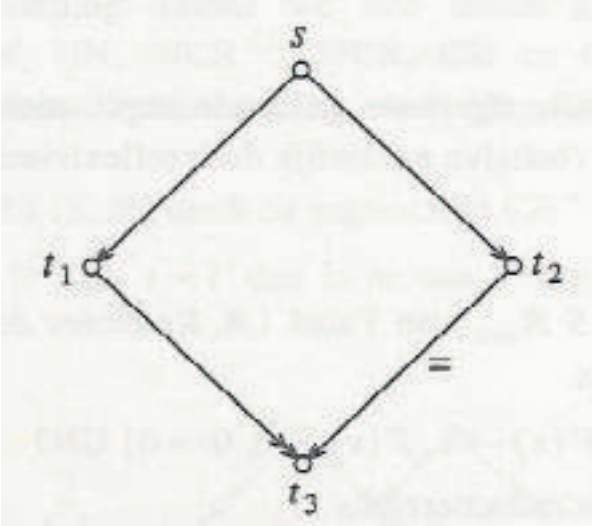
$$\forall s,t,t' \in Ter(\Sigma, R)\ (t, t' \text{ normaalvorm } s \twoheadrightarrow t, s \twoheadrightarrow t' \Rightarrow t \equiv t').$$

a) Bewijs dat voor elke TRS de implicatie $UN \Rightarrow UN^{\to}$ geldt.

b) Geef een tegenvoorbeeld voor de omgekeerde implicatie.

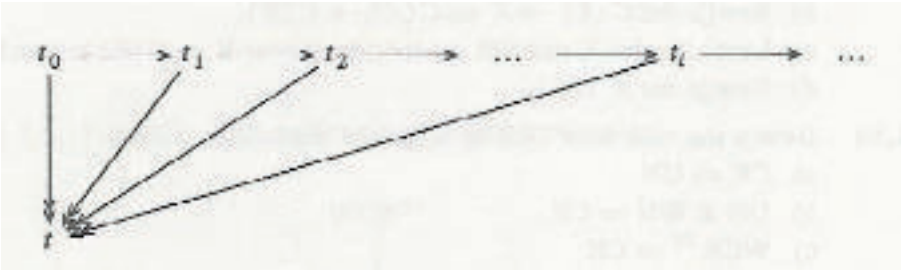
5.1.6.6. OPGAVE. Een TRS $\mathcal{R}$ heet *sterk confluent* als er voor elke twee éénstapsreducties $s \to t_1$, $s \to t_2$ een t_3 bestaat zodat $t_1 \twoheadrightarrow t_3$ en $t_2 \to^{=} t_3$ (zie Figuur 5.12)

Bewijs dat elke sterk confluente TRS confluent is.



Figuur 5.12. *Sterke confluentie.*

5.1.6.7. OPGAVE. Een TRS heet *inductief* (IND) als er voor elke reductierij $t_0 \to t_1 \to t_2 \to \ldots$ een term t bestaat zodat $t_i \twoheadrightarrow t$, voor alle $i \geq 0$ (zie Figuur 5.13).

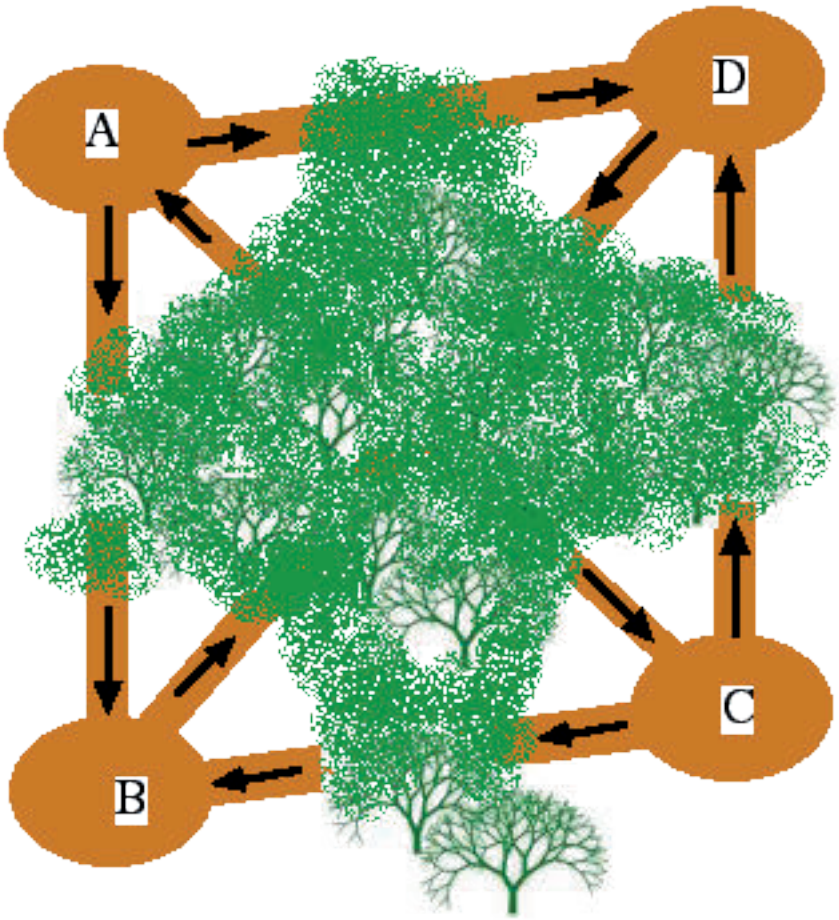


Figuur 5.13. *Eigenschap IND.*

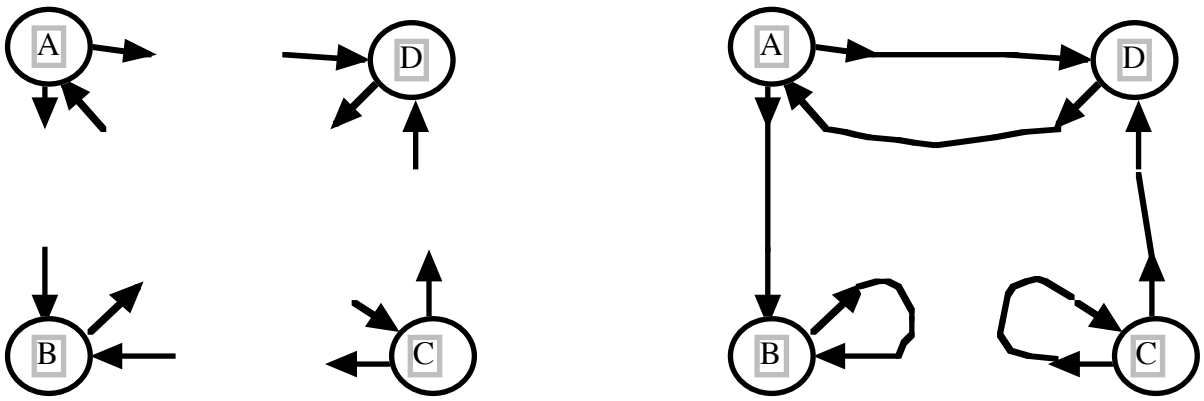
a) Bewijs dat voor elke TRS de implicatie UN & WN ⇒ IND geldt.
b) Construeer een TRS die wel CR maar niet IND is.
c)* Construeer een TRS die wel WCR en WN maar niet IND is.

5.1.6.8. OPGAVE (Toyama [89]. In een bos (Figuur 5.14) zijn wandelpaden tussen vier plaatsen A, B, C, D. De wandelpaden hebben éénrichtingsverkeer, als aangegeven door de pijlen. Van bovenaf gezien is een groot deel van de paden verborgen onder de bomen. Gegeven is dat het wandelbos de eigenschap WCR heeft, maar niet CR. Hoe lopen de

verborgen gedeeltes van de paden, zodat ze op elkaar aansluiten? In Figuur 5.15 staat een mogelijkheid om de paden op elkaar aan te laten sluiten; maar die 'oplossing' is fout, want wel CR.



Figuur 5.14. *Toyama's bos.*



Figuur 5.15. *Toyama's bos.*

5.2. *Confluentie is modulair.*

De tweede ingang van dit hoofdstuk naar confluentie is via modulariteit. Dat houdt in dat we bij het bewijzen van eigenschappen de 'verdeel-en-heers' strategie toe proberen te passen, door de gevraagde eigenschap eerst voor de in delen opgesplitste TRS te bewijzen. De meest eenvoudige definitie van 'delen' wordt gegeven door het begrip 'disjuncte vereniging'.

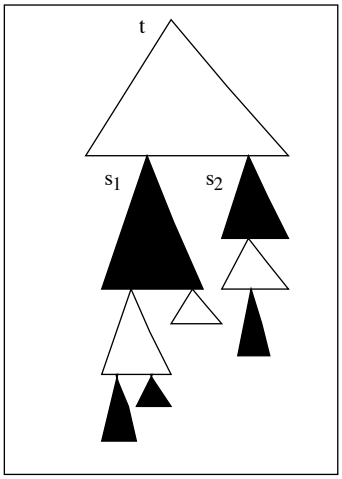
5.2.1. DEFINITIE. (i) De *vereniging* van twee TRSen $\mathcal{R}_i = (\Sigma_i, R_i)$ $(i = 1, 2)$ is:

$$\mathcal{R}_0 \cup \mathcal{R}_1 = (\Sigma_0 \cup \Sigma_1, R_0 \cup R_1)$$

(ii) Als de signaturen Σ_0 en Σ_1 disjunct zijn, $\Sigma_0 \cup \Sigma_1 = \varnothing$, heet $\mathcal{R}_0 \cup \mathcal{R}_1$ een *disjuncte* vereniging. Om deze conditie niet te hoeven noemen gebruiken we de speciale notatie

$$\mathcal{R}_0 \oplus \mathcal{R}_1 .$$

Het is vaak handig om de termen in de TRS $\mathcal{R}_0$ *wit* te kleuren, en in de TRS $\mathcal{R}_1$ *zwart*. Een term uit $\mathcal{R}_0 \oplus \mathcal{R}_1$ heeft dan een wit-zwart kleuring als in Figuur 5.16.



Figuur 5.16. *Term uit* $\mathcal{R}_0 \oplus \mathcal{R}_1$

5.2.2. STELLING (Toyama, 1987). *Voor alle* TRSen $\mathcal{R}_1$ *en* $\mathcal{R}_2$ *geldt:*

$\mathcal{R}_1 \oplus \mathcal{R}_2$ *is* CR $\Leftrightarrow$ $\mathcal{R}_1$ *is* CR *en* $\mathcal{R}_2$ *is* CR.

Figuur 5.17. *Yoshihito Toyama*.

5.2.3. STELLING (Middeldorp, 1988). *Voor alle* TRSen $\mathcal{R}_1$ *en* $\mathcal{R}_2$ *geldt:*

$\mathcal{R}_1 \oplus \mathcal{R}_2$ *is* UN $\Leftrightarrow$ $\mathcal{R}_1$ *is* UN *en* $\mathcal{R}_2$ *is* UN.

Figuur 5.18. *Aart Middeldorp.*

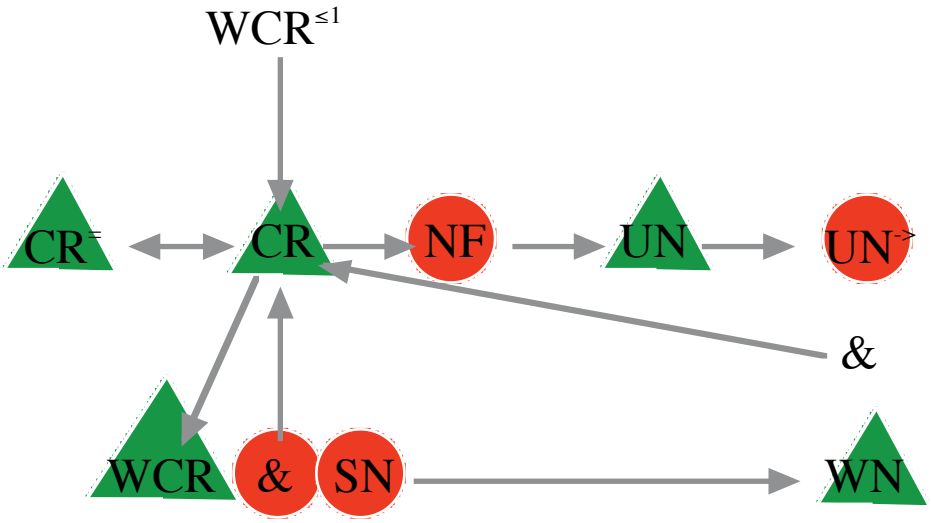
5.2.3. DEFINITIE. We noemen een eigenschap $\mathcal{P}$ van TRSen *modulair*, als voor alle TRSen $\mathcal{R}_1$ en $\mathcal{R}_2$ geldt:

$$\mathcal{R}_1 \oplus \mathcal{R}_2 \text{ is } \mathcal{P} \Leftrightarrow \mathcal{R}_1 \text{ is } \mathcal{P} \text{ en } \mathcal{R}_2 \text{ is } \mathcal{P}.$$

(*Notatie:* In plaats van '$\mathcal{R}$ *is* $\mathcal{P}$' schrijven we ook $\mathcal{R} \models \mathcal{P}$.)

In bovenstaande stellingen is de disjunctheid van de signaturen essentiëel.
(*Opgave: Waarom?*)

De stellingen boven zeggen dus dat CR en UN modulaire eigenschappen zijn. Lang niet alle eigenschappen van TRSen zijn modulair. In Figuur 5.19 zijn met groen (driehoekig) enkele modulaire eigenschappen aangegeven en met rood (rond) enkele niet-modulaire. Voor de eigenschap $\text{WCR}^{\leq 1}$ is de status een Opgave.



Figuur 5.19. *Enkele modulaire en niet modulaire eigenschappen.*

Het volgende tegenvoorbeeldlaat zien dat SN een niet modulaire eigenschap is.

5.2.4. TEGENVOORBEELD. (Toyama, 1987). Bekijk de volgende TRSen:

$\mathcal{R}_1 = \{F(0, 1, x) \to F(x, x, x)\}$,

$\mathcal{R}_2 = \{or(x, y) \to x, or(x, y) \to y\}$.

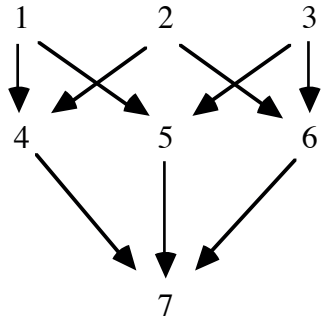
Beide TRSen zijn SN (Opgave!), maar $\mathcal{R}_1 \oplus \mathcal{R}_2$ is niet SN. We hebben namelijk de volgende cyclische reductie:

$F(or(0, 1), or(0, 1), or(0, 1)) \to$
$F(0, or(0, 1), or(0, 1)) \to$
$F(0, 1, or(0, 1)) \to$
$F(or(0, 1), or(0, 1), or(0, 1))$

In dit tegenvoorbeeld is $\mathcal{R}_2$ niet confluent. We zouden kunnen dus kunnen vermoeden dat 'het daaraan ligt', en dat als we SN versterken met CR we wel een modulaire eigenschap hebben. De combinatie van CR en SN, dus CR & SN, wordt 'compleet' genoemd. De vraag is nu dus of CR & SN een modulaire eigenschap is. Maar ook dat is niet het geval. Er is het volgende tegenvoorbeeld:

$\mathcal{R}_1$ heeft deze elf regels:

$F(4, 5, 6, x) \to F(x, x, x, x)$
$F(x, y, z, w) \to 7$



en $\mathcal{R}_2$ heeft drie regels:

$G(x, x, y) \to x$

$G(x, y, x) \to x$
$G(y, x, x) \to x.$

Nu zijn $\mathcal{R}_1$ en $\mathcal{R}_1$ beide compleet, maar $\mathcal{R}_1 \oplus \mathcal{R}_2$ is dat niet, want er is de volgende oneindige reductie:

$F(G(1, 2, 3), G(1, 2, 3), G(1, 2, 3), G(1, 2, 3)) \twoheadrightarrow$
$F(G(4, 4, 3), G(5, 2, 5), G(1, 6, 6), G(1, 2, 3)) \twoheadrightarrow$
$F(\ 4,\ 5,\ 6,\ G(1, 2, 3)) \to$
$F(G(1, 2, 3), G(1, 2, 3), G(1, 2, 3), G(1, 2, 3)).$

(Voor een iets eenvoudiger tegenvoorbeeld zie Opgave 5.2.11.8.)

Toch zijn we nu 'warm'. Het tegenvoorbeeld dat we net zagen (en ook dat in Opgave 5.2.11.8) behelst een niet-links-lineaire TRS. Dat wil zeggen:

5.2.6. DEFINITIE. (i) Een term is *lineair* als er geen variabele dubbel in voor komt, en *niet-lineair* als dat wel zo is. (Bijv. G(x, x, y) is niet-lineair.)
(ii) Een reductieregel $t \to s$ is *links-lineair* als t lineair is.
(iii) Een TRS is *links-lineair* als zijn reductieregels links-lineair zijn.

5.2.7. STELLING. (Toyama, Klop & Barendregt [89]).
Laten $\mathcal{R}_1$, $\mathcal{R}_2$ links-lineaire TRSen zijn. Dan:

$\mathcal{R}_1 \oplus \mathcal{R}_2$ is compleet dan en slechts dan als $\mathcal{R}_1$ and $\mathcal{R}_2$ compleet zijn.

Tot slot van deze sectie vermelden we, eveneens zonder bewijs, nog twee nuttige feiten. Eerst geven we een definitie:

5.2.8. DEFINITIE.
(1) Een herschrijfregel $t_1 \to t_2$ heet een *collaps-regel* (c-regel) als de rechterkant t_2 een variabele is.
(2) Een herschrijfregel $t_1 \to t_2$ heet een *duplicerende regel* (d-regel) als een zekere variabele x meer keren in de rechterkant t_2 dan in de linkerkant t_1 voorkomt.

Voorbeeld: F(x,x) → G(x, x) is niet een d-regel; maar F(x, x) → H(x, x, x) wel. Ook P(x) → G(x, x) is een d-regel. De AMS0 TRS bevat één c-regel en één d-regel.

5.2.9. STELLING (Middeldorp [88]) *Stel dat $\mathcal{R}_1$ en $\mathcal{R}_2$ twee disjuncte* TRSen *zijn, beide* SN.

(i) *Als $\mathcal{R}_1$ noch $\mathcal{R}_2$ c-regels bevat, dan heeft $\mathcal{R}_1 \oplus \mathcal{R}_2$ de eigenschap* SN.

(ii) *Als $\mathcal{R}_1$ noch $\mathcal{R}_2$ d-regels bevat, dan heeft $\mathcal{R}_1 \oplus \mathcal{R}_2$ de eigenschap* SN.

(iii) *Als één van beide TRSen $\mathcal{R}_1$, $\mathcal{R}_2$ noch c-regels, noch d-regels bevat, dan is $\mathcal{R}_1 \oplus \mathcal{R}_2$* SN.

Deze Stelling kan korter geformuleerd worden, als volgt. Het bewijs van de equivalentie is een opgave.

5.2.10. STELLING. *Stel dat $\mathcal{R}_1$ en $\mathcal{R}_2$ twee disjuncte* TRSen *zijn, beide* SN, *maar zodat hun disjuncte vereniging $\mathcal{R}_1 \oplus \mathcal{R}_2$ niet* SN *is. Dan is $\mathcal{R}_1$ dupliceren d en $\mathcal{R}_2$ collapsend, of omgekeerd.*

5.2.11.1. OPGAVE. Bewijs één van de implicaties in Stelling 5.2.3.

5.2.11.2. OPGAVE. Bewijs dat de TRSen $\mathcal{R}_1$ en $\mathcal{R}_2$ van Tegenvoorbeeld 5.2.4 SN zijn.

5.2.11.3. OPGAVE. Bewijs dat WN een modulaire eigenschap is.

5.2.11.4. OPGAVE. Laat zien dat $UN^{\rightarrow}$ (zie Opgave 5.1.5.5) geen modulaire eigenschap is.

5.2.11.5. OPGAVE. Een TRS (Σ, R) heeft de eigenschap NF (*normaalvorm eigenschap, normal form property*) als:

$$\forall s, t \in Ter(\Sigma, R): \text{als } s = t \text{ en } t \text{ is normaalvorm dan } s \twoheadrightarrow t.$$

a) Bewijs dat voor elke TRS de implicaties CR ⇒ NF en NF ⇒ UN gelden.
b) Geef tegenvoorbeelden voor de omgekeerde implicaties.
c) Laat zien dat NF (in tegenstelling tot CR en UN) geen modulaire eigenschap is.

5.2.11.6. OPGAVE. Bewijs de equivalentie van Stelling 5.2.9 en 5.2.10.

5.2.11.7. OPGAVE.

(i) Bekijk de volgende TRS:

$$\infty \rightarrow S(\infty)$$

$eq(x, x) \to true$
$eq(x, S(x)) \to false$.

Laat zien dat deze niet links-lineaire TRS niet CR is.
(ii) Laat zien dat zonder de eerste regel de TRS wel CR is.

(iii) Stel dat we aan CL bovenstaande twee niet links-lineaire regels toevoegen: CL $\cup$ {$eq(x, x) \to true$, $eq(x, S(x)) \to false$}. In feite is dit een disjuncte vereniging, dus we kunnen ook schrijven: CL $\oplus$ {$eq(x, x) \to true$, $eq(x, S(x)) \to false$}. Is de resulterende TRS confluent?

(iv) En als we λ-calculus uitbreiden met de twee niet-links-lineaire regels, tot

$\lambda \oplus$ {$eq(x, x) \to true$, $eq(x, S(x)) \to false$}?

5.2.11.8. OPGAVE. $\mathcal{R}_1$ heeft de regels

$F(0, 1, x) \to F(x, x, x)$,
$F(x, y, z) \to 2$,
$0 \to 2$,
$1 \to 2$

$\mathcal{R}_2$ heeft de regels

$D(x, y, y) \to x$,
$D(x, x, y) \to y$.

(i) Nu zijn $\mathcal{R}_1$ en $\mathcal{R}_2$ compleet.

Maar de disjuncte vereniging $\mathcal{R}_1 \oplus \mathcal{R}_2$ is het niet:

(ii) Laat zien dat de term F(M, M, M) met M $\equiv$ D(0, 1, 1) een cyclische reductie heeft.

5.2.11.9. OPGAVE.
(i) Bewijs dat de een-regelige TRS {$D(x, x) \to E$} CR is.
(ii) Bewijs dat CL $\oplus$ {$D(x, x) \to E$} ook CR is. (Gegeven is dat CL confluent is, zoals we later zullen zien, in Sectie 5.3.)
(iii) Bekijk de TRS in Tabel 5.1 en de Opgave aldaar, en maak aannemelijk dat CL $\cup$ {$Dxx \to E$} *niet* confluent is.
(iv) Wat is de oplossing van de paradox gevormd door (ii) en (iii)?

5.2.12. OPMERKING. Er zijn drie manieren om begrippen zoals CR, WCR, WCR$^{\leq 1}$, UN, ... weer te geven:
(i) Grafisch, met *diagrammen* zoals boven;
(ii) Met *predicaatlogische formules* zoals bijvoorbeeld ...
(iii) Als uitdrukking in '*relatie-calculus*', zoals ...

Deze 'notaties' kunnen door elkaar gebruikt worden. Methode (iii) is compact, maar niet vlot leesbaar. Methode (i) is het duidelijkst; (ii) het meest precies. Het is ook een kwestie van smaak. Voor een aantal voorbeelden: zie de korte 'reader' uit Terese op het eind van dit Hoofdstuk.

5.2.13. OPGAVE. In deze opgave bewijzen we een confluentie stelling die als voorkennis vereist:
(1) het *Hindley-Rosen Lemma*,
(2) het begrip *'commuterende reducties'*,
(3) de notie van een *parallelle reductie*, en
(4) het begrip *'overlapping'* van reductieregels.

Voor (1,2) zie de reader uit Terese op het eind van dit Hoofdstuk. De notie (3) wordt in de volgende sectie behandeld.

5.2.14. DEFINITIE. Laten $\mathcal{R}_1$, $\mathcal{R}_2$ twee TRSen zijn. Definieer: $\mathcal{R}_1 \perp \mathcal{R}_2$ ($\mathcal{R}_1$ en $\mathcal{R}_2$ zijn *onderling orthogonaal*) als er geen overlapping is van een regel van $\mathcal{R}_1$ met een van $\mathcal{R}_2$. (Er mag wel overlap zijn tussen $\mathcal{R}_1$-regels, of tussen $\mathcal{R}_2$ regels).

5.2.15. STELLING. (Raoult & Vuillemin [88].) *Laten $\mathcal{R}_1$ en $\mathcal{R}_2$ links-lineaire en confluente TRSen*

zijn zodat $\mathcal{R}_1 \perp \mathcal{R}_2$. *Dan is* $\mathcal{R}_1 \cup \mathcal{R}_2$ *confluent.*

BEWIJS. Zie Figuur5.20. We bewijzen dat

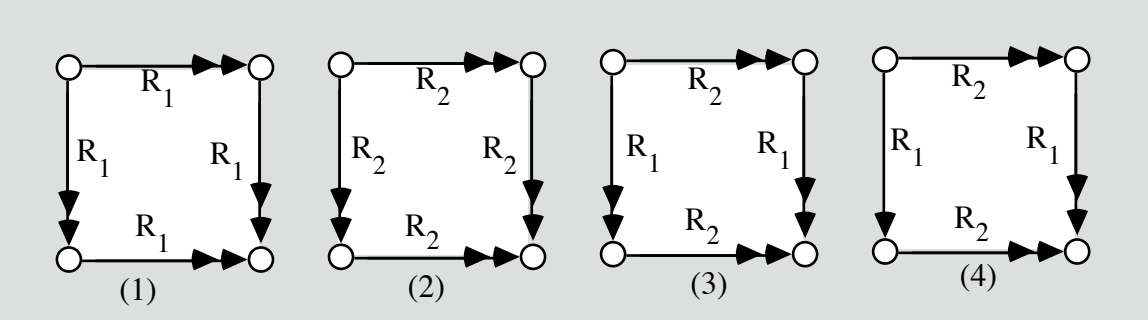
(1) $\mathcal{R}_1$ reducties commuteren;

(2) $\mathcal{R}_1$-reducties commuteren met $\mathcal{R}_2$-reducties;

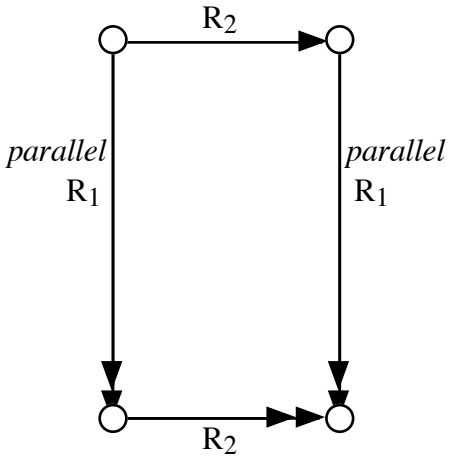
(3) Om (3) te bewijzen, is het voldoende om (4) als in Figuur xx te bewijzen.

(4) Om 4 te bewijzen, hebben we de links-lineariteit en de onderlinge orthogonaliteit nodig. Om precies te zijn: we kunnen gemakkelijk bewijzen dat een parallelle $\mathcal{R}_1$-reductie door een $\mathcal{R}_2$-stap wordt overgevoerd in weer een parallelle $\mathcal{R}_1$-reductie, als in Figuur 5.21. Tenslotte volgt het resultaat door het Hindley-Rosen Lemma toe te passen.

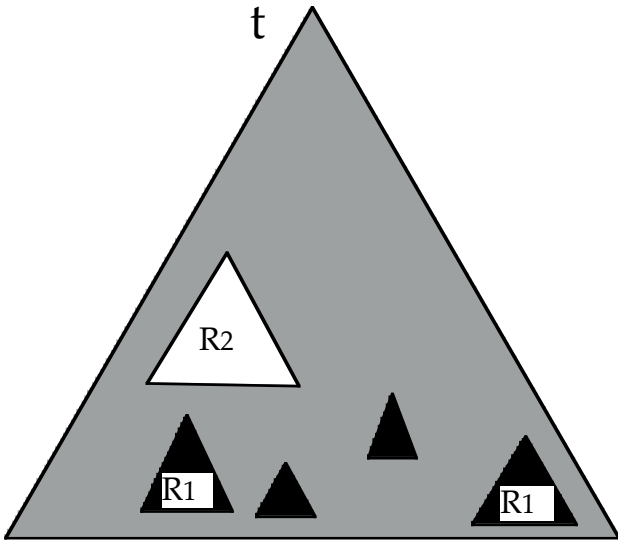
Merk op dat de links-lineariteit nodig is–anders hebben we het tegenvoorbeeld in Opgave 5.2.11.7.



Figuur 5.20. *Bewijs van de stelling van Raoult en Vuillemin.*



Figuur 5.21. *Parallel Moves Proposition PMP.*



Figuur 5.22. *Positie van redexen in PMP.*

5.2.17. OPGAVE. Zij $\mathcal{R}$ een links-lineaire, confluente TRS die alleen de applicatie operator gemeen heeft met CL. Is CL $\cup$ $\mathcal{R}$ dan noodzakelijk confluent?

5.2.18. OPGAVE. Zij $\mathcal{R}$ een links-lineaire, confluente TRS. Stel dat de signatuur van $\mathcal{R}$ disjunct is van die van λ-calculus, dat wil zeggen dat $\mathcal{R}$ niet de applicatie-operator bevat. Dan is $\lambda \oplus \mathcal{R}$, de disjuncte vereniging van λ-calculus en $\mathcal{R}$, weer confluent.
Als voorbeeld hebben we dat $\lambda \oplus \{or(true, x) \to true, or(x, true) \to true\}$ confluent is.

(Bewijsschets: gebruik dezelfde strategie als voor Stelling 5.2.15.)

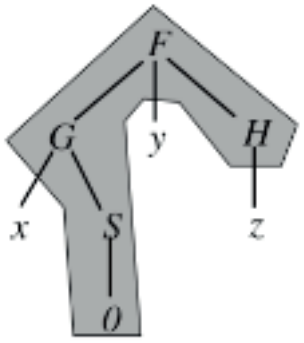
5.2.19. OPGAVE. Als we in de vorige Opgave toestaan dat $\mathcal{R}$ de applicatie-operator bevat, hebben we een probleem, namelijk door *'variable-applying'* rules, zoals bijvoorbeeld xy → x. Als we zulke regels verbieden (de linkerkant van een regel mag niet een context van een subterm xt zijn voor variabele x en term t) dan gaat confluentie wel door. Als voorbeeld hebben we dan dat $\lambda \cup$ (*or true* x → *true*, *or* x *true* → *true*} confluent is.

5.2.20 STELLING. (Bachmair & Dershowitz [86].
Neem aan dat de twee TRSen $\mathcal{R}_0$ en $\mathcal{R}_1$ SN *zijn, maar niet noodzakelijk disjunct. Dan: als $\mathcal{R}_0$ links-lineair is, $\mathcal{R}_1$ rechts-lineair, en er is geen overlap tussen linkerkanten van $\mathcal{R}_0$ en rechterkanten van $\mathcal{R}_1$, dan is $\mathcal{R}_0 \cup \mathcal{R}_1$ weer SN.*

5.3. Orthogonaal herschrijven.

De derde belangrijke weg naar confluentie is het *orthogonaal* herschrijven. We definieren eerst wat het *patroon* van een redex is (*redex pattern*). Hiertoe nemen we de regel die bij het redex hoort, en laten uit de linkerkant de variabelen weg. De '*context*' (term met gaten) die dan ontstaat, is het patroon van de reductieregel, en ook van de daardoor gegenereerde redexen.

5.3.1. VOORBEELD. Bekijk de regel F(G(x), S(0), y, H(z)) → x. Dan is het patroon de context F(G( ), S(0), , H( )), in de volgende figuur door arcering aangegeven.

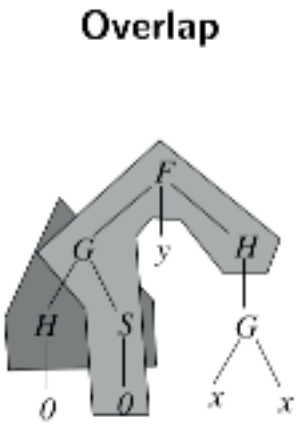


Figuur 5.23. *Redex patroon.*

5.3.2. DEFINITIE. Een TRS R is *orthogonaal*, als

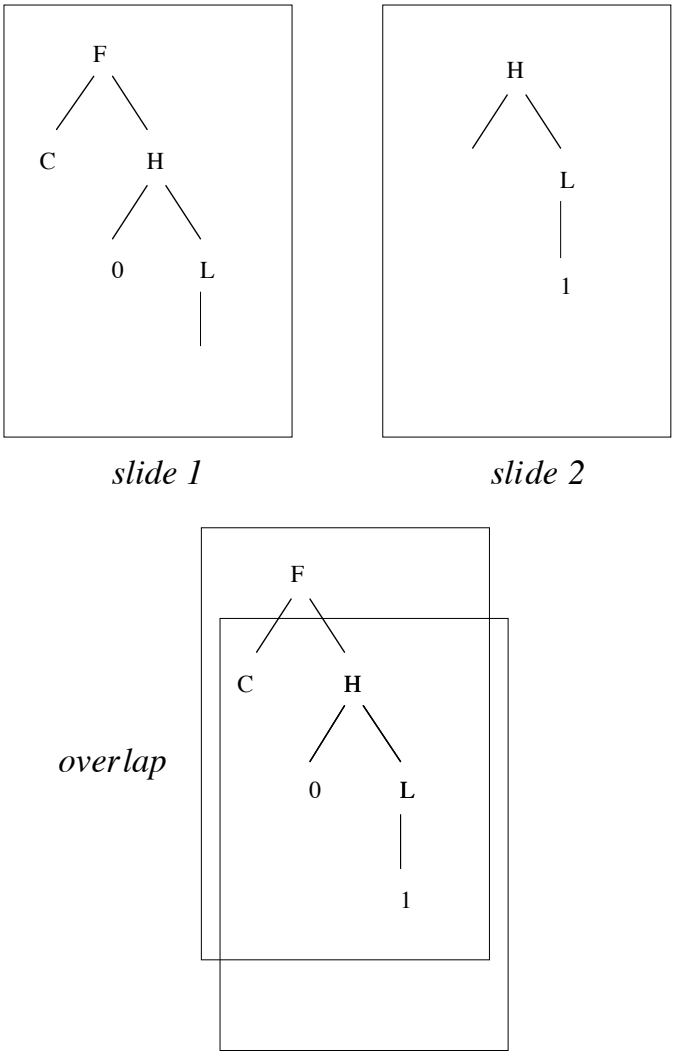
(1) de reductieregels links-lineair zijn;
(2) er geen overlappende redex-patronen zijn.

5.3.1. VOORBEELD. In de volgende figuren zijn een aantal voorbeelden gegeven van overlap tussen reductieregels.

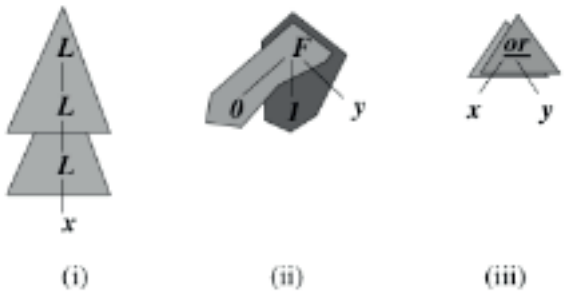


$$\begin{array}{lll} \rho_1: & F(G(x,S(0)),y,H(z)) & \to \quad x \\ \rho_2: & G(H(x),S(y)) & \to \quad y \end{array}$$

Figuur 5.24. *Overlappende redex-patronen.*



Figuur 5.25. *Overlappende redex-patronen.*



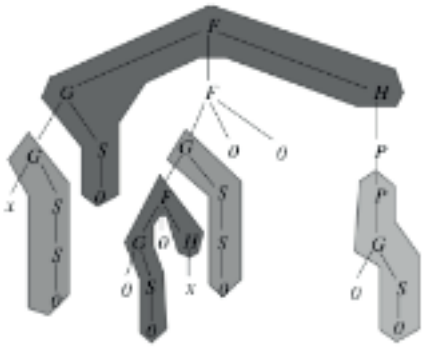
(i) $L(L(x)) \to 0$

(ii) $F(0,x,y) \to 0,\ F(x,1,y) \to 1$

(iii) $or(x,y) \to x,\ \ or(x,y) \to y$

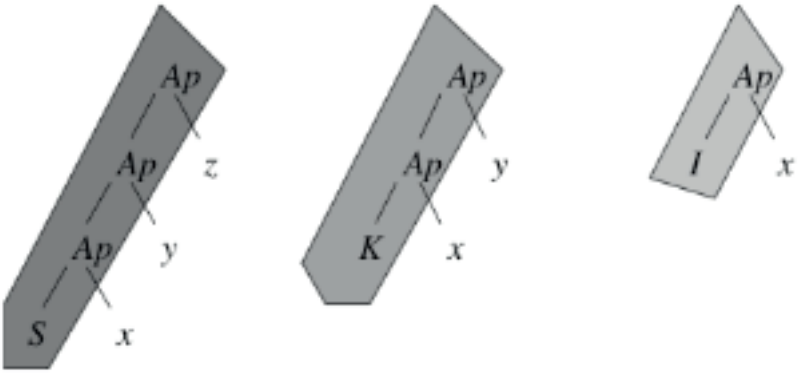
Figuur 5.26. *Overlappende redex-patronen.*

$$
\begin{array}{lll}
r_1: & F(G(x,S(0)),y,H(z)) & \to \ x \\
r_2: & G(x,S(S(0))) & \to \ 0 \\
r_3: & P(G(x,S(0))) & \to \ S(0)
\end{array}
$$



Figuur 5.27. *Niet-overlappende redex-patronen.*

5.3.1. STELLING. *Orthogonale TRsen zijn confluent.*



Figuur 5.28. *Redex patronen van CL.*

5.3.1. VOORBEELD.
(i) De TRS uit Figuur 5.27.
(ii) CL is orthogonaal, en dus confluent.

5.3.2. OPGAVE. Laat zien dat de AMS0 TRS orthogonaal is.

5.3.3. OPGAVE.
(i) Teken de patronen van de reductieregels van de TRS voor de Ackermann functie.
(ii) Laat zien dat de TRS voor de Ackermann functie orthogonaal is, en dus confluent.

5.3.4. OPGAVE. Laat zien dat SKIM (Figuur 5.29) confluent is.

SKIM

$Sxyz$	$\rightarrow$	$xz(yz)$	
Kxy	$\rightarrow$	x	
Ix	$\rightarrow$	x	
$Cxyz$	$\rightarrow$	xzy	
$Bxyz$	$\rightarrow$	$x(yz)$	
Yx	$\rightarrow$	$x(Yx)$	
$P_0(Pxy)$	$\rightarrow$	x	
$P_1(Pxy)$	$\rightarrow$	y	
$Uz(Pxy)$	$\rightarrow$	zxy	
$Cond\ True\ x\ y$	$\rightarrow$	x	
$Cond\ False\ x\ y$	$\rightarrow$	y	
$Plus\ \mathbf{n}\ \mathbf{m}$	$\rightarrow$	$\mathbf{n}+\mathbf{m}$	
$Times\ \mathbf{n}\ \mathbf{m}$	$\rightarrow$	$\mathbf{n}\cdot\mathbf{m}$	
$Eq\ \mathbf{n}\ \mathbf{n}$	$\rightarrow$	$True$	
$Eq\ \mathbf{n}\ \mathbf{m}$	$\rightarrow$	$False$	if $n \neq m$

Figuur 5.25. *SKI-reductiemachine.*

De drie volgende bladzijden bevatten een korte 'reader' uit het boek Term Rewriting systems van Terese. Hierin worden Abstracte Reductie Systemen (ARS) gedefinieerd, met verschillende reductierelaties. Verder komen een aantal bekende eigenschappen in deze bladzijden voor, in de drie vormen: diagrammatisch, predicaatlogisch, in relatie-calculus.
De reader eindigt met het Lemma van Hindley en Rosen, nodig voor het bewijs van de Stelling van Raoult-Vuillemin boven.

1.2.1. DEFINITION (basics). An *abstract reduction system* (ARS) is a structure $\mathcal{A} = (A, \{\to_\alpha |\ \alpha \in I\})$ consisting of a set A and a set of binary relations $\to_\alpha$ on A, indexed by a set I. We write $(A, \to_1, \to_2)$ instead of $(A, \{\to_\alpha |\ \alpha \in \{1,2\}\})$. For $\alpha \in I$, the relations $\to_\alpha$ are called *reduction* or *rewrite* relations. Sometimes we will refer to $\to_\alpha$ as α. In the case of just one reduction relation, we simply write $\to$. We write $\to_I$ for the union $\bigcup\{\to_\alpha |\ \alpha \in I\}$. Reversed arrows denote the inverse relations.

1.2.8. DEFINITION (confluence). Let $\mathcal{A} = (A, \{\to_\alpha |\ \alpha \in I\})$ be an ARS, $\alpha, \beta \in I$ and let $\to = \to_\alpha$.

(i) We say $\to_\alpha$ *commutes weakly* with $\to_\beta$ if the diagram of Figure 1.3(a) holds, i.e. if $\forall a,b,c \in A\ (c \leftarrow_\beta a \to_\alpha b \Rightarrow \exists d \in A\ c \twoheadrightarrow_\alpha d \twoheadleftarrow_\beta b)$.

(ii) We say $\to_\alpha$ *commutes*[2] with $\to_\beta$ if $\twoheadrightarrow_\alpha$ and $\twoheadrightarrow_\beta$ commute weakly.

(iii) $a \in A$ is *weakly confluent* if $\forall b,c \in A\ (c \leftarrow a \to b \Rightarrow \exists d \in A\ c \twoheadrightarrow d \twoheadleftarrow b)$. The reduction relation $\to$ is *weakly confluent* or *weakly Church–Rosser* (WCR) if every $a \in A$ is weakly confluent, see Figure 1.3(b).

(iv) $a \in A$ is *subcommutative* if $\forall b,c \in A\ (c \leftarrow a \to b \Rightarrow \exists d \in A\ c \to^{\equiv} d \leftarrow^{\equiv} b)$. The reduction relation $\to$ is *subcommutative* (notation $\mathrm{CR}^{\leq 1}$) if every $a \in A$ is subcommutative, see the diagram in Figure 1.3(c).

(v) $a \in A$ has the *diamond property* if $\forall b,c \in A\ (c \leftarrow a \to b \Rightarrow \exists d \in A\ c \to d \leftarrow b)$. The reduction relation $\to$ has the *diamond property* if (notation DP) if every $a \in A$ has the *diamond property.*

(vi) $a \in A$ has the *triangle property* if $\exists a' \in A\ \forall b \in A\ (a \to b \Rightarrow b \to a')$. (The name 'triangle property' refers to cases in which we have $a \to a$. In all cases the triangle property implies the diamond property.) The reduction relation $\to$ has the *triangle property* if (notation TP) if every $a \in A$ has the *triangle property.*

(vii) $a \in A$ is *confluent* if $\forall b,c \in A\ (c \twoheadleftarrow a \twoheadrightarrow b \Rightarrow \exists d \in A\ c \twoheadrightarrow d \twoheadleftarrow b)$. The reduction relation $\to$ is *confluent* or *Church–Rosser*, or has the Church–Rosser property (CR), if every $a \in A$ is confluent, see Figure 1.3(d).

In the following proposition we state some alternative characterizations of various notions of confluence defined above. These characterizations are algebraic in the sense that they are expressed in terms of compositions of relations. Terse as they are, they are preferably visualized by Figure 1.3. We omit the easy equivalence proofs.

1.2.9. PROPOSITION. *Let conditions be as in Definition 1.2.8. Then*

(i) $\to_\alpha$ *commutes weakly with* $\to_\beta$ *if and only if* $\leftarrow_\beta \cdot \to_\alpha \subseteq \twoheadrightarrow_\alpha \cdot \twoheadleftarrow_\beta$,

(ii) $\to_\alpha$ *commutes with* $\to_\beta$ *if and only if* $\twoheadleftarrow_\beta \cdot \twoheadrightarrow_\alpha \subseteq \twoheadrightarrow_\alpha \cdot \twoheadleftarrow_\beta$,

(iii) $\to$ *is weakly confluent if and only if* $\leftarrow \cdot \to \subseteq \twoheadrightarrow \cdot \twoheadleftarrow$, *that is,* $\to$ *is weakly self-commuting,*

(iv) $\to$ *is subcommutative if and only if* $\leftarrow \cdot \to \subseteq \to^{\equiv} \cdot \leftarrow^{\equiv}$,

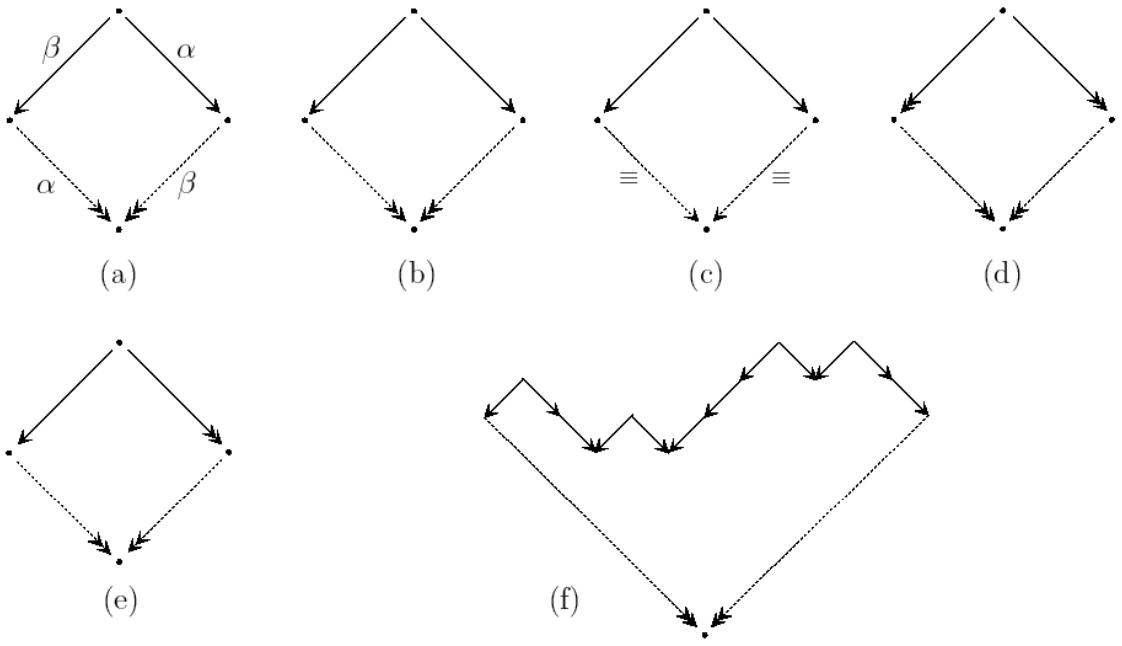


Figure 1.3: Various confluence patterns

(v) $\to$ *has the diamond property if and only if* $\leftarrow \cdot \to \subseteq \to \cdot \leftarrow$ *(for reflexive reduction relations, subcommutativity is equivalent to the diamond property),*

(vi) $\to$ *is confluent if and only if* $\twoheadleftarrow \cdot \twoheadrightarrow \subseteq \twoheadrightarrow \cdot \twoheadleftarrow$.

1.2.10. PROPOSITION. *For every ARS $(A, \to)$, the following are equivalent:*

(i) $\to$ *is confluent;*
(ii) $\twoheadrightarrow$ *is weakly confluent;*
(iii) $\twoheadrightarrow$ *is self-commuting;*
(iv) $\twoheadrightarrow$ *is subcommutative (or has the diamond property, as* $\twoheadrightarrow$ *is reflexive);*
(v) $\twoheadleftarrow \cdot \twoheadrightarrow \subseteq \twoheadrightarrow \cdot \twoheadleftarrow$, *see the diagram in Figure 1.3(e);*
(vi) $= \subseteq \twoheadrightarrow \cdot \twoheadleftarrow$, *see the diagram in Figure 1.3(f).*

1.4.1. Confluence

1.4.1. EXERCISE. Let $(A, \to_1, \to_2)$ be an ARS such that $\to_1$ and $\to_2$ commute weakly and $\to_{12}$ is SN; then $\to_1$ and $\to_2$ commute. Show also that the condition $\to_{12}$ is SN cannot be weakened to $\to_1$ is SN and $\to_2$ is SN.

1.4.2. EXERCISE (Rosen [1973]). If $(A, \to_1, \to_2)$ is an ARS such that $\twoheadrightarrow_1 = \twoheadrightarrow_2$ and $\to_1$ is subcommutative, then $\to_2$ is confluent.

1.4.3. EXERCISE (Hindley [1964]). Let $(A, \{\to_\alpha | \ \alpha \in I\})$ be an ARS such that for all $\alpha, \beta \in I$, $\to_\alpha$ commutes with $\to_\beta$. (In particular, $\to_\alpha$ commutes with itself.) Then the union $\to \ = \bigcup\{\to_\alpha | \ \alpha \in I\}$ is confluent. (This proposition is usually referred to as the Lemma of Hindley and Rosen; see e.g. Barendregt [1984], Proposition 3.3.5.)

6
ABSTRACT HERSCHRIJVEN

INHOUD.

5.0. *Introductie.*

In dit hoofdstuk bekijken we enkele onderwerpen op het gebied van abstract herschrijven, waarbij we objecten zonder termstructuur hebben. In de eerste Sectie is dat een voorbeeld van V. van Oostrom, dat elegant met een abstract herschrijflemma (van Toyama) aangepakt kan worden. In dit voorbeeld bewijzen we SN, uitgaande van WN.

De tweede sectie is een voorbeeld van confluentie voor een topologische notie, vlechten (*braids*). Het eigenlijke confluentiebewijs wordt hier niet gegeven.

De derde sectie behandelt een belangrijke methode om confluentie te bewijzen voor abstracte herschrijfsystemen (ARSen). Deze methode van '*decreasing diagrams*' is afkomstig van N. de Bruijn en V. van Oostrom.

De slotsectie is een voorbeeld van een abstract herschrijflemma van V. van Oostrom waarvan het bewijs lastig is.

6.1.*Kommen en Knikkers.*

Gegeven is een naar weerskanten oneindige rij kommen, die elk eindig veel knikkers kunnen bevatten. Het totale aantal knikkers dat verdeeld is over de kommen moet eindig zijn. Zo'n rij van kommen met knikkers erin noemen we een *situatie*. Wiskundig geformuleerd is een situatie s dus een afbeelding s: $\mathbb{Z} \to \mathbb{N}$, van de verzameling $\mathbb{Z}$ van gehele getallen naar de natuurlijke getallen $\mathbb{N}$, met de eis dat $\Sigma_{i \in \mathbb{Z}}$ s(i) eindig is. Op de verzameling van situaties hebben we de volgende herschrijfregel: wanneer kom i twee of meer knikkers bevat, wordt één knikker in de linkerkom (kom i-1) gedaan en één in de rechterkom (kom i+1). In Figuur 6.1 is dit weergegeven. De andere kommen dan deze drie, i-1, i, i+1 blijven zoals ze waren.

Mathematical modeling

Definition 1 • *A* situation *is a map* $s{:}\mathbb{Z}\to\mathbb{N}$.

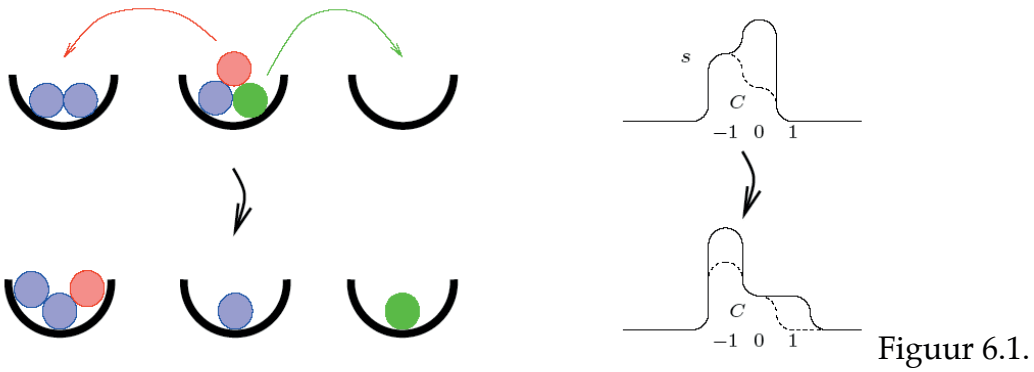
- *The* bean rule *is the pair* $\langle l, r\rangle$ *of triples, with* $l = \langle 0,2,0\rangle$ *and* $r = \langle 1,0,1\rangle$.
- *A* bean step at i *has shape* $C + l^i \curvearrowright_i C + r^i$, *for some situation* C *and integer* i. *Here,* $+$ *denotes pointwise addition, the* i-shift s^i *of a situation* s *is defined by* $s^i(j) = s(j-i)$, *and a triple* $\langle m,n,k\rangle$ *of natural numbers denotes the situation defined by* $-1 \mapsto m$, $0 \mapsto n$, $1 \mapsto k$, *and* 0 *elsewhere.*

Example 2 • *The initial situation top left in Figure 1 is modeled as the situation to its right, i.e. as* s *defined by* $s(0) = 3$, $s(-1) = 2$, *and* 0 *everywhere else,*

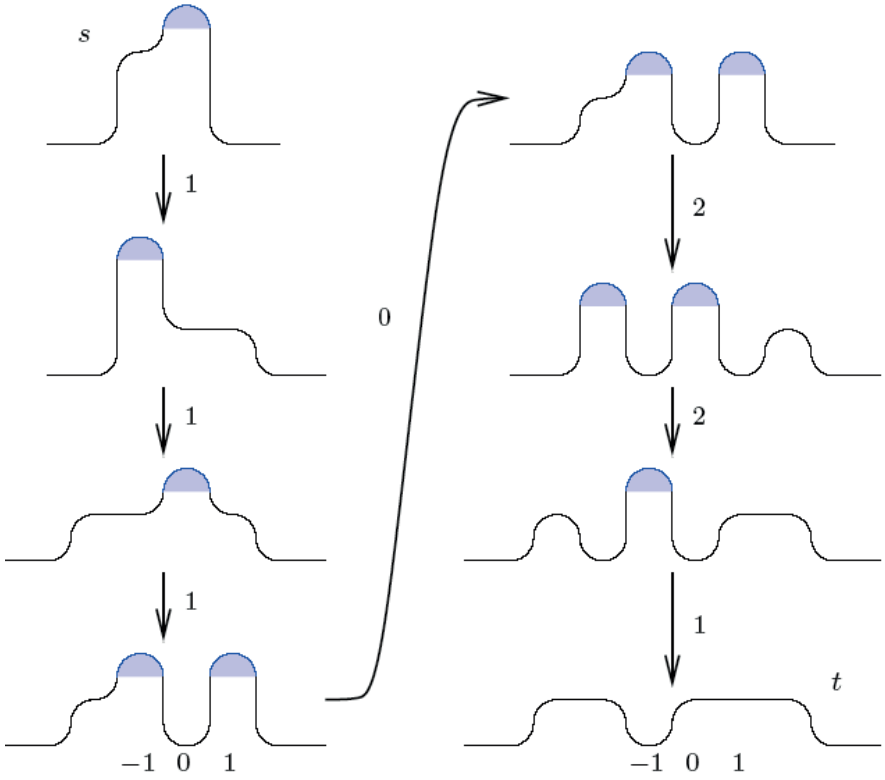
De puzzel is nu:

> *Bewijs dat deze herschrijfrelatie SN en CR is, en dat bovendien alle reducties naar normaalvorm even lang zijn.*

Notatie: We zullen reductiestappen 'indiceren' met subscript i om aan te geven dat de kom i degene was waar twee knikkers uit gehaald zijn.

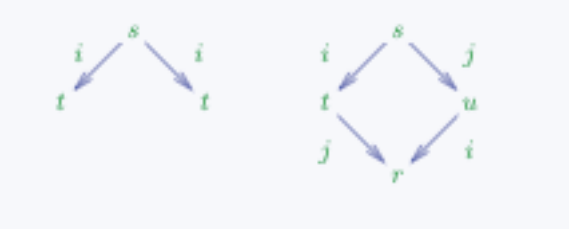


Figuur 6.1.



Figuur 6.2. *Een herschrijfrij van kommen en knikkers.*

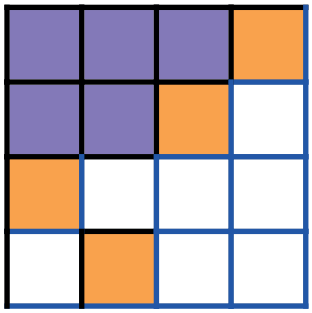
6.1. DEFINITIE. Een ARS (A, →) is *lineair orthogonaal* (LO) als voor elke vork s →t en s →u geldt: ofwel t = u, of er is een v met t → v en u → v.



Figuur 6.3. *Lineair orthogonaal.*

6.2. STELLING (Toyama [92]). *Als* (A, →) LO *en* WN *is, dan is* (A, →) *bovendien* SN, UN$^{\to}$, *en alle reducties naar de normaalvorm zijn even lang.*

BEWIJS. Opgave. □



Figuur 6.4. *Bewijs van LO stelling: tegels leggen.*

6.3. STELLING. *De 'kommen en knikkers' herschrijfrelatie → is* LO.

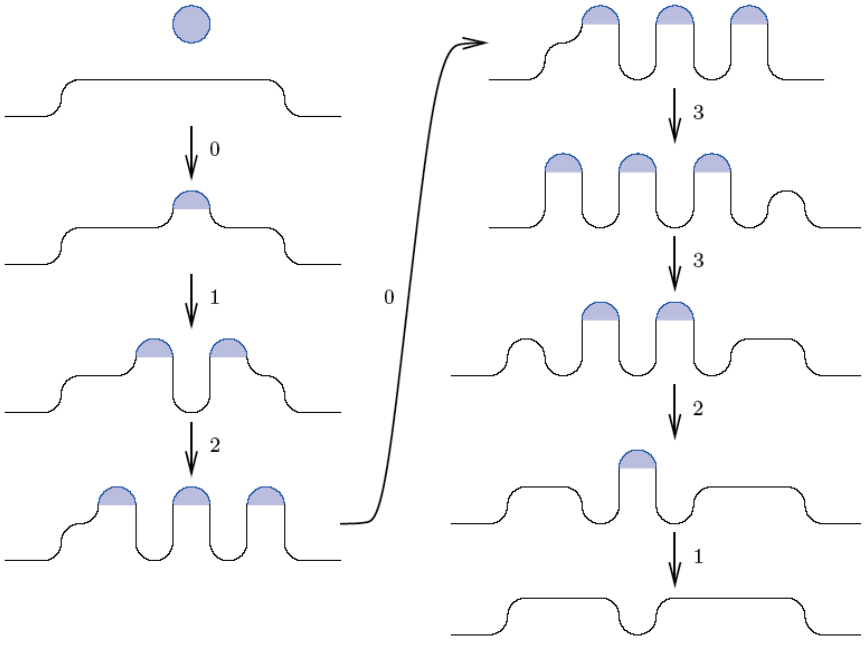
BEWIJS. Opgave. □

6.4. OPGAVE. Bewijs dat de kommen en knikkers herschrijfrelatie WN is.

(i) Ga eerst na wat de normaalvormen zijn.

(ii) Voeg aan een normaalvormsituatie één knikker toe, waar dan ook, en laat zien dat deze situatie weer tot normaalvorm gereduceerd kan worden.

(iii) Concludeer nu WN met inductie naar het totaal aantal knikkers in de situatie.



Figuur 6.5. *Golfbeweging van normaalvorm plus één knikker.*

Het kommen en knikkers voorbeeld heeft veel te maken met *golven en golfvoortplanting*. Om dat te zien, moeten we bovenstaande reductie *parallel* toepassen. Dat wil

0	0	0	0	0	10	0	0	0	0	0
0	0	0	0	1	8	1	0	0	0	0
0	0	0	0	2	6	2	0	0	0	0
0	0	0	1	1	6	1	1	0	0	0
0	0	0	1	2	4	2	1	0	0	0
0	0	0	2	1	4	1	2	0	0	0
0	0	1	0	3	2	3	0	1	0	0
0	0	1	1	2	2	2	1	1	0	0
0	0	1	2	1	2	1	2	1	0	0
0	0	2	0	3	0	3	0	2	0	0
0	1	0	2	1	2	1	2	0	1	0
0	1	1	0	3	0	3	0	1	1	0
0	1	1	1	1	2	1	1	1	1	0
0	1	1	1	2	0	2	1	1	1	0
0	1	1	2	0	2	0	2	1	1	0
0	1	2	0	2	0	2	0	2	1	0
0	2	0	2	0	2	0	2	0	2	0
1	0	2	0	2	0	2	0	2	0	1
1	1	0	2	0	2	0	2	0	1	1
1	1	1	0	2	0	2	0	1	1	1
1	1	1	1	0	2	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1

Figuur 6.6. *Golfvoortplanting vanuit situatie met één piek.*

Figuur 6.7. *Golfbeweging op positie naast piek.*

zeggen dat we in elke parallelle reductiestap *alle* redexen tegelijkertijd herschrijven. Dus we doen alle mogelijke i-stappen simultaan. Van elke kom die minstens twee knikkers bevat, leggen we er dus één links en één rechts in de kom. Dat dit geen problemen oplevert (i.e. 'welgedefinieerd' is), ook als de twee redex-kommen naast elkaar staan, is niet moeilijk in te zien; we hebben dat in feite al ingezien bij bovenstaand bewijs van de LO eigenschap. In de volgende figuur is weergegeven hoe een 'piek' van 10 knikkers zich als golf uitbreidt, en uitsterft. Verschillende golfverschijnselen zijn gemakkelijk in te zien, of te realiseren:

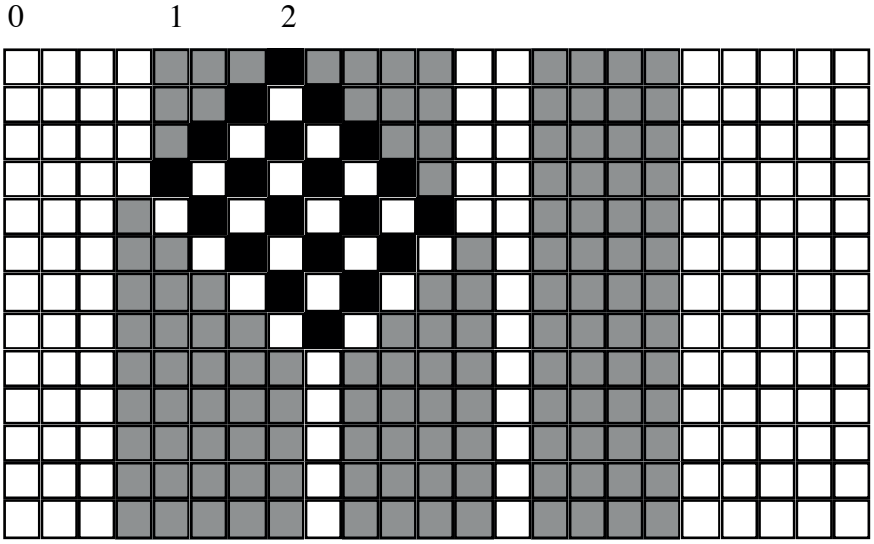
(i) *Superpositie* van golven;
(ii) *Weerkaatsing* tegen vaste rand;
(iii) *Tweedimensionale* golfvoortplanting;
(iv) Toevoegen van een *golfbron* zodat de golven niet uitsterven.

Nog een interessante opmerking (van V. van Oostrom) is dat het kommen en knikkers voorbeeld, als we de beperking laten vallen dat er in een situatie in totaal maar eindig veel knikkers zijn, zelfs equivalent is met een Turing machine:

On rules One may vary on the rules. E.g. if we allow for an infinite number of beans, but still require that for every rule the numbers of beans in its left- and right-hand sides are the same, one obtains universal computing power. To see this, view the array of bowls as the tape of a Turing machine. The ith symbol of its signature, say of total size m, can then be represented on this tape by a sequence of three bowls the first of which contains $m+1$ beans (a marker), the second i beans (the symbol), and the third $m-i$ beans (the complement). It is easy to see that by encoding blanks, the position of the head, and the rules of the Turing machine in a similar way, the Turing machine can be faithfully simulated. (The markers are used to enforce that rules can only be applied with an empty, i.e. everywhere 0, context situation C.)

6
ABSTRACT HERSCHRIJVEN

INHOUD.



(*Correctie*) Figuur 6.8. *Golfbeweging van normaalvorm plus één knikker.*

6.2. *Braids.*

Figuur 6.2.1. *Modigliani: girl with braids.*

Vlechten (*braids*) zijn een onderwerp in de wiskunde, en wel in de topologie. Ze werden voor het eerst bestudeerd door de wiskundige Emil Artin, in zijn artikel *Theorie der Zöpfe* in 1926. Ze komen ook voor in de quantum-mechanica. Voor ons doel leveren ze een interessant voorbeeld van een confluente abstracte herschrijfrelatie. Eigenlijk is het een voorbeeld dat de beperktheid aantoont van onze methoden om confluentie te bewijzen, want de confluentie van vlechten is niet te bewijzen met de gangbare methoden die we tot nu toe gezien hebben. De beschrijving van het probleem is (nog) in het engels, als volgt.

A girl has two braids consisting of, say, 6 strings (see Figure 6.2.2). The father starts braiding the left braid, the mother of the girl starts braiding the right braid. After some initial 'twists' as indicated in the figure, they notice that they do it in a different way. But they want to arrive, eventually, at two identical braids. *Question: can they go on and still arrive at identical braids?*

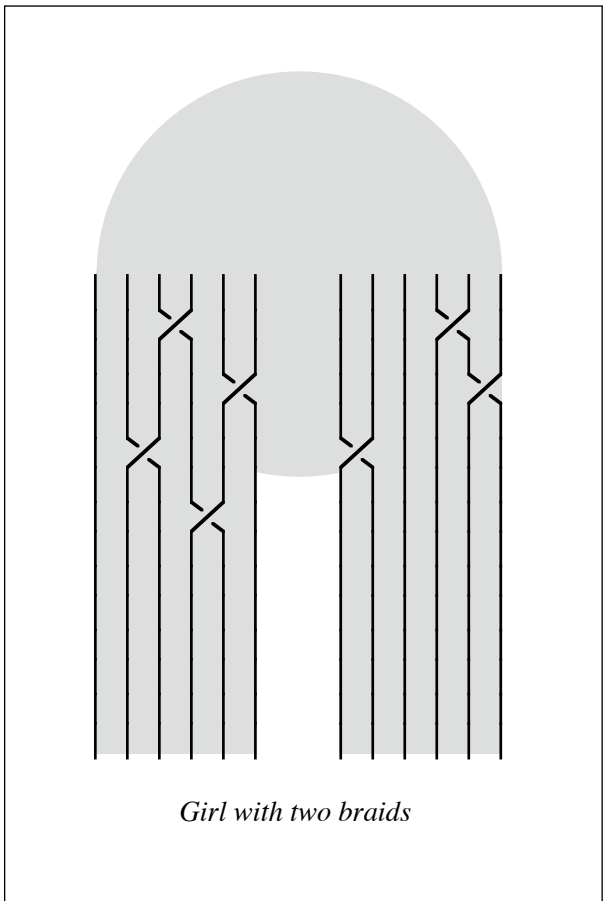
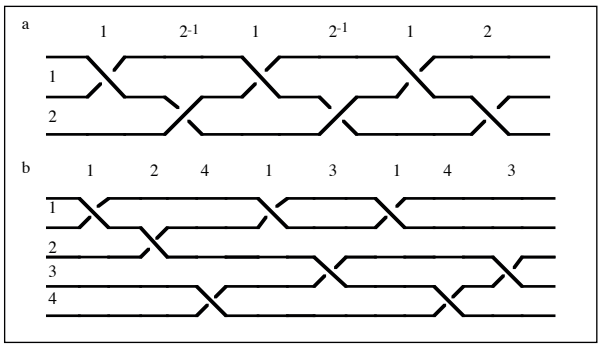


Fig. 6.2.2.

Note that braids are subject to a topological equivalence, which will be explained now. First we need a notation for braids. Consider Figure 6.2.3. The openings between the strings are numbered 1, 2, 3,... . A twist or crossing in which the upper string moves over the lower is denoted 'positively', just as the corresponding opening: if this is i, the positive twist twist at this position is also denoted by i. Otherwise we have a 'negative' crossing, denoted by i^{-1} if it is in the i-th opening. Thus the braid in Figure 6.2.3a is $1.2^{-1}.1.2^{-1}.1.2$.

Now we restrict attention to *positive crossings only*. E.g. in Figure 6.2.3b we have the braid 1.2.4.1.3.1.4.3. The restriction means that we work in the *semi-group* generated by 1, 2, 3, 4 (if there are five strings).



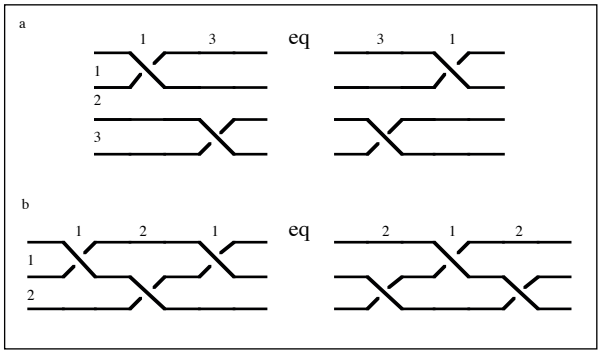
Figuur 6.2.3. *Notatie van vlechten.*

Not all these braids are really different. See Figure 6.2.4a. The braid 1.3 is 'the same', topologically viewed, as 3.1, just by shifting the crossings in the other order. Also 1.4 is equivalent with 4.1. We will write 1.3 = 3.1, and 1.4 = 4.1. In general we have:

$$i.j = j.i \quad \text{if } |i-j| \geq 2$$

For consecutive openings like 1 and 2, respective crossings do not commute. But it is not hard to see that starting with 1.2 and 2.1, we can make them (topologically) equal by continuing 1.2 with 1 and 2.1 with 2. So 1.2.1 = 2.1.2. See Figure 6.2.4b. Note that 1.2.1 and 2.1.2 are indeed topologically the same; an experiment with actual strings of wire will demonstrate this. In general we have for all i:

$$i.(i+1).i = (i+1).i.(i+1)$$



Figuur 6.2.4. *Equivalente vlechten.*

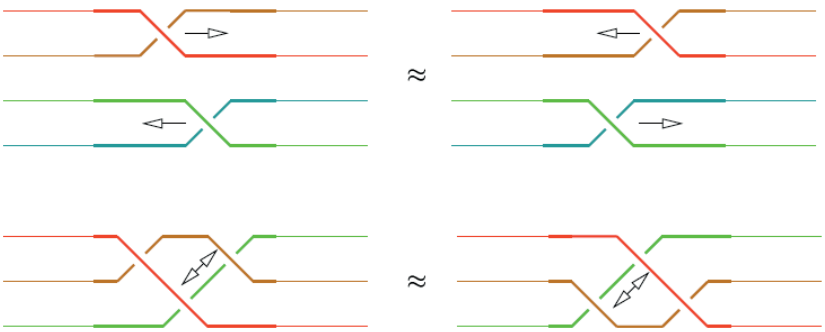
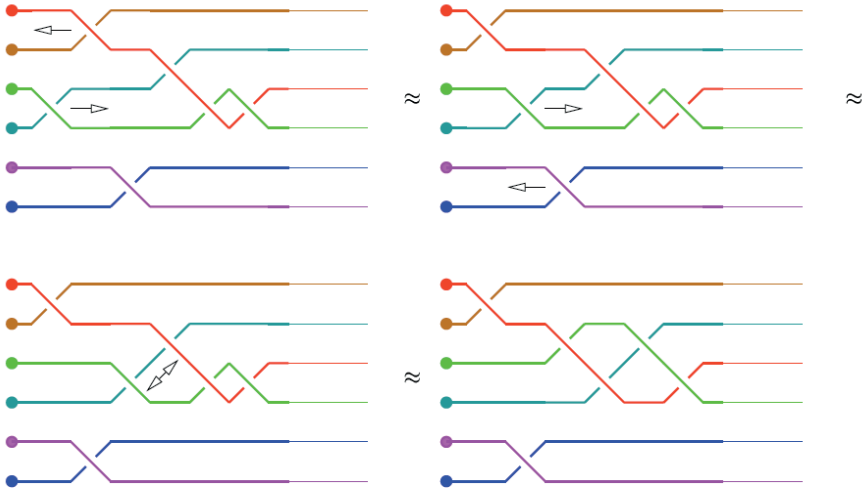


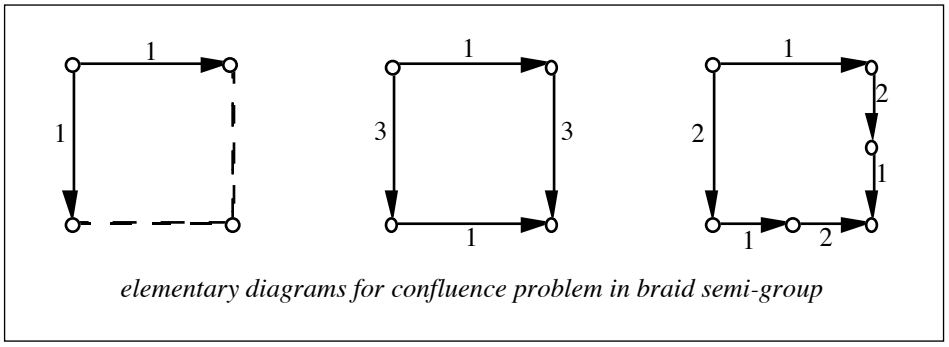
Figure 1: Transformaties op vlechten

Example 2 *Using the transformations in Figure 1, which of course can be applied anywhere in a braid, we can transform our example braid as follows (into an equivalent one):*



The arrows indicate how a braid is transformed into its successor.

The equations above completely define the topological equivalence considered (see Artin [47]). The confluence problem is now: given two elements u, v of this braid semi-group, can we always find elements x, y such that ux = vy? The problem can be approached by means of an abstract rewriting analysis using the "elementary diagrams" as in Figure 6.2.5. (Only some of the generators 1,2,3,... are mentioned in the figure.) These diagrams are just another way of phrasing the equations above; e.g. the second e.d. states that 1.3 = 3.1 and the last one states that 1.2.1 = 2.1.2. The first e.d is trivial, it states that 1 = 1; but such trivial e.d's still are useful as will be apparent in the next example. The question is now whether 'tiling' with these diagrams always succeeds in a confluent reduction diagram.



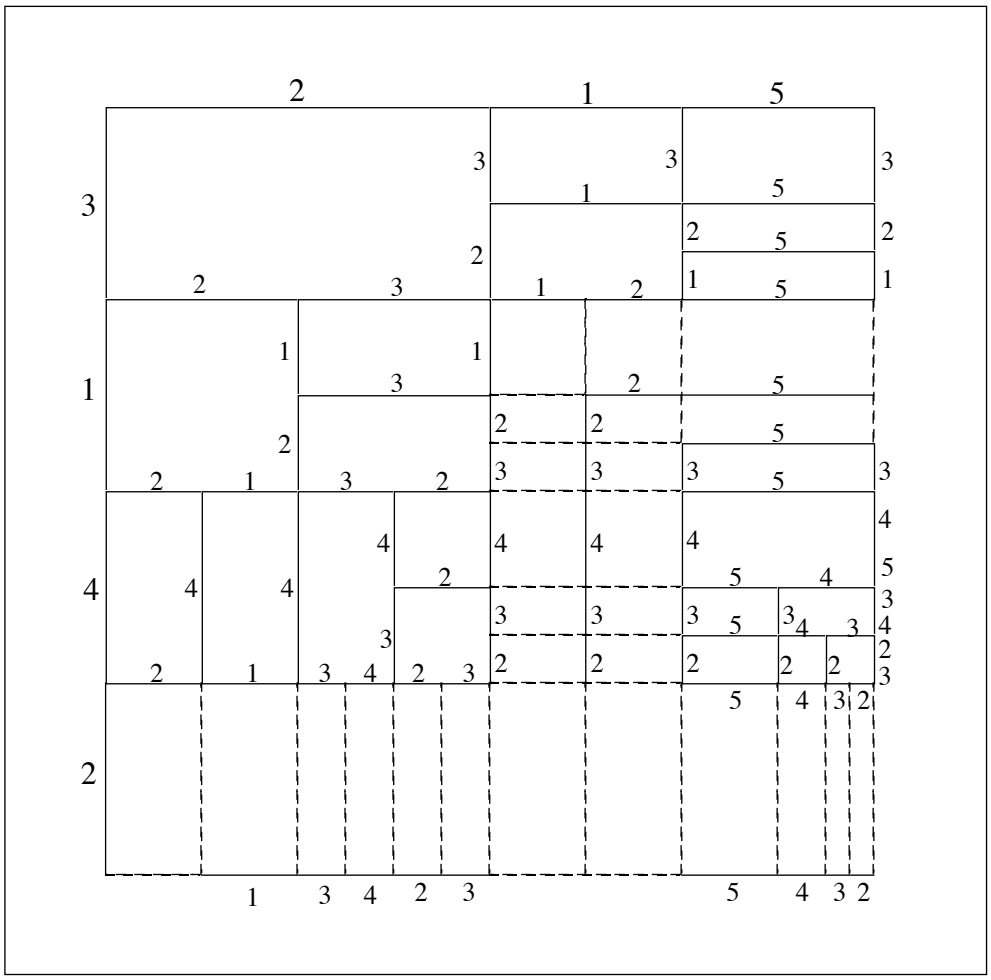
Figuur 6.2.5. *Elementaire diagrammen voor vlechten.*

6.2.1. EXAMPLE. We complete in a diagram the braidings started by the father and the mother as in Figure 6.2.2: the braids there are 3142 and 215 (counting the openings from right to left). See Figure 6.2.6.

As it turns out, we are lucky in this example; the tiling procedure terminates succesfully in a completed "reduction diagram", whose lower and right sides yield the (or rather, an) answer to our question.

6.2.2. THEOREM (Garside [69]). *Braids are confluent. That is: For all* u, v *of the braid semi-group, there exist elements* x, y *such that* ux = vy.

6.2.3. REMARK.(i) Actually, braids are confluent in a canonical way, namely by the tiling procedure as demonstrated in the example. This was proved recently by Mellies and van Oostrom.
(ii) Note that 'empty steps', as introduced in the trivial e.d.'s, propagate 'through' steps in the obvious way; see the reduction diagram in Figure 6.2.6.



Figuur 6.2.6. *Oplossing van vlechtenprobleem.*

6.2.4. LITERATUUR.

ARTIN, E.(1926). *Theorie der Zöpfe.* Abh. math. Semin. Hamburg Univ. 4 (1926), 47-72.

ARTIN, E.(1947). *Theory of braids.* Ann. of Math. (2) 48 (1947) 101-126

ARTIN, E.(1947a). *Braids and permutations.* Ann. of Math. vol. 48 (1947), 643-649

GARSIDE, F.A. (1969). *The braid group and other groups.* Quart. J. Math. 20 (1969) 235-254

MAKANIN, G.S. (1968). *The conjugacy problem in the braid group.* Soviet Math. Dokl. 9 (1968) 1156-1157.

6.3. *Decreasing diagrams.*

We can infer CR from WCR when given the additional assumption of SN (Newman's Lemma). Also in this Section we will infer CR from WCR, this time given extra information on the nature of WCR, that is, on the form of the 'elementary diagrams' or e.d.'s that WCR provides. In the braid confluence problem we already encountered the procedure of tiling with e.d.'s to obtain a confluent diagram. We will now look in detail at this 'tiling game', starting with Huet's strong confluence lemma, in a sequence of introductory examples.

6.3.1 EXAMPLE.

(i) DEFINITION. For an ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ we define: $\rightarrow$ is *strongly confluent* if

$$\forall a, b, c \in A \ \exists d \in A \ (b \leftarrow a \rightarrow c \ \Rightarrow \ c \twoheadrightarrow d \ {}^{=}\!\!\leftarrow b)$$ (See Fig. 6.3.1(a))

(Here $\rightarrow^{=}$ is the reflexive closure of $\rightarrow$, so $b \rightarrow^{=} d$ is zero or one step.)

(ii) LEMMA (Huet [80]). *Let $\mathcal{A}$ be strongly confluent. Then $\mathcal{A}$ is CR.*

The proof is simple. The assumption of strong confluence provides us with elementary diagrams (e.d.'s) as in Figure 6.3.1(a), which can be used to obtain CR as suggested in the diagram in Figure 6.3.1(b), where we profit from the fact that "splitting" occurs in the direction of our choice.
(This is so, because the quantification over a, b, c implicit in Figure 6.3.1(a) is universal, so we can mirror the e.d. in that figure around the main diagonal.)

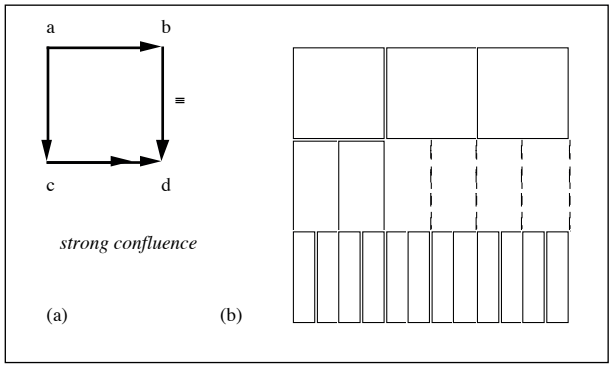


Figure 6.3.1

6.3.2. EXAMPLE. When splitting of e.d.'s would occur in both directions, our "diagram chase" to obtain confluence may very well fail: given e.d.'s of the form as in Figure 2.2(a), so corresponding to the WCR assumption

$$\forall a,b,c \in A \ \exists d,e,f \in A \ (c \leftarrow a \rightarrow b \ \Rightarrow \ c \rightarrow d \rightarrow e \leftarrow f \leftarrow b),$$

we may fail in our attempt to construct a confluent diagram by tiling; see Figure 6.3.2(b), where the diagram construction of diagram D falls in the trap of an infinite regress.

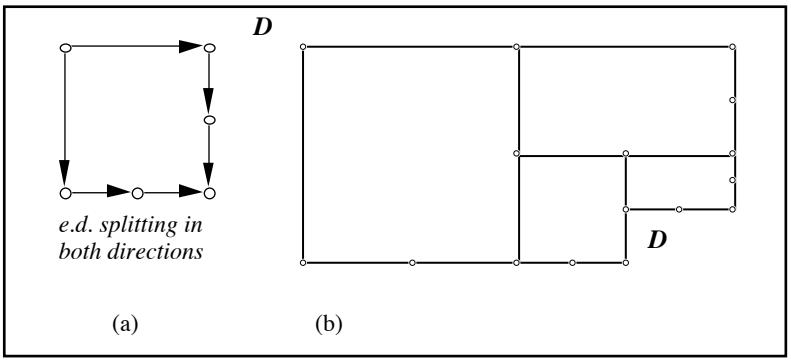


Figure 6.3.2

6.3.3. EXAMPLE. This tiling game is even more interesting when dealing with more than one reduction relation. Suppose we have two reduction relations, labeled (or indexed) with 1, 2, and that we have for their union $\rightarrow_{12}$ WCR in the form of the e.d.'s as in Figure 6.3.3.

Question: does CR hold for $\rightarrow_{12}$?
Answer: No; for we may have a situation as in Figure 6.3.3, lower diagram.

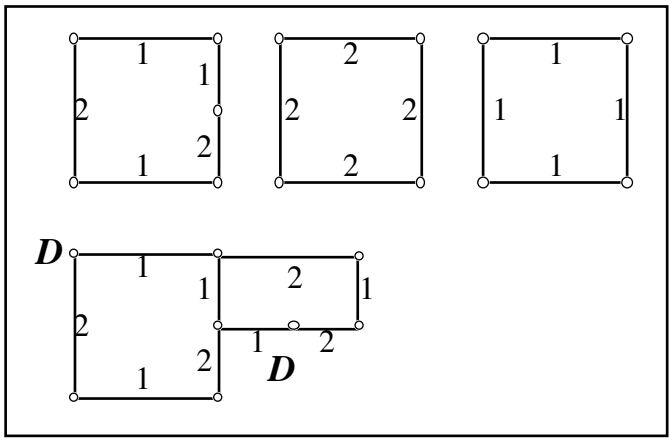


Figure 6.3.3

6.3.4. EXAMPLE. However, if we change the e.d.'s of Example 6.3.3 slightly as in Figure 6.3.4 we do have CR!

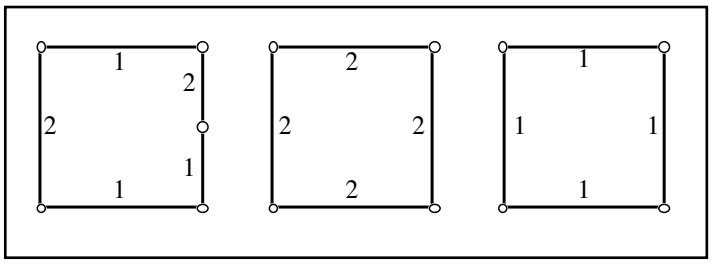


Figure 6.3.4

6.3.5. EXAMPLE. Question: do we have CR given the e.d.'s in Figure 6.3.5? This is not at all easy to see. The answer is yes, as will be clear at the end of this topic.

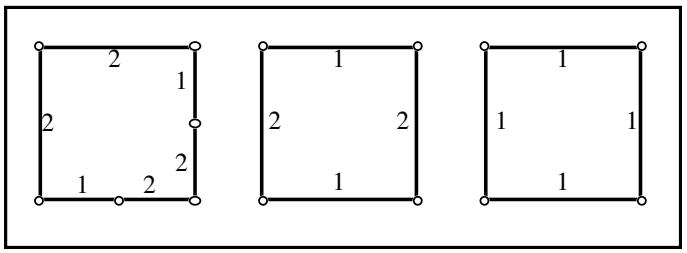


Figure 6.3.5

6.3.6. Reduction diagrams. We will now be more precise about elementary diagrams. They are of the following shapes; see Figure 6.3.6.

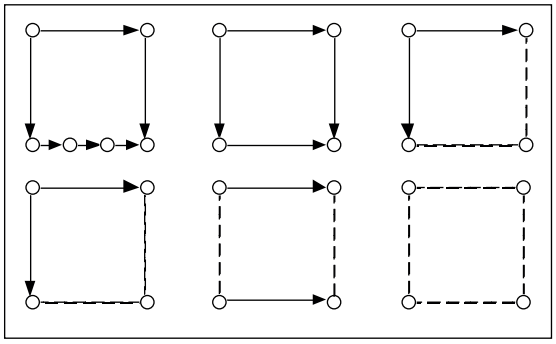


Figure 6.3.6

They are the 'atomic' or basic building blocks for constructing reduction diagrams. A non-trivial elementary diagram consists of two diverging steps (arrows), joined by two sequences of steps of arbitrary length. Note that in the e.d.'s we may use empty sides (the dashed sides, in some figures shaded), to keep matters orthogonal. This gives rise to some trivial e.d.'s as in the lower part of Figure 6.3.6. The e.d.'s are used as scalable 'tiles' with the intention to obtain a completed reduction diagram as in Figure 6.3.7. Usually we will forget the direction of the arrows (second picture in Figure 6.3.7): they always are from left to right, or downwards (except the empty 'steps' that have no direction).

6.3.7. Indexed Abstract Reduction Systems. In this topic we will consider an Abstract Reduction System (ARS) $\mathcal{A}$, equipped with a collection of rewrite or reduction relations $\rightarrow_\alpha$, indexed by some

set I: $\mathcal{A} = \langle A, (\rightarrow_\alpha)_{\alpha \in I} \rangle$. The index set I is in this topic always a *well-founded partial order*. In examples, we will use the set of natural numbers with the usual ordering as index set. The union of the rewrite relations $\rightarrow_\alpha$ will be $\rightarrow$.

2.8. Multisets. We will use multisets over the index set I, together with the multiset ordering induced by that of I. It is well-known that this is again a well-founded partial order. (Furthermore, if I is a total order, then the p.o. of multisets over I is again total.)

We will use the notations: multiset ordering $\geq_\mu$, strict multiset ordering $>_\mu$, multiset union $\cup$.

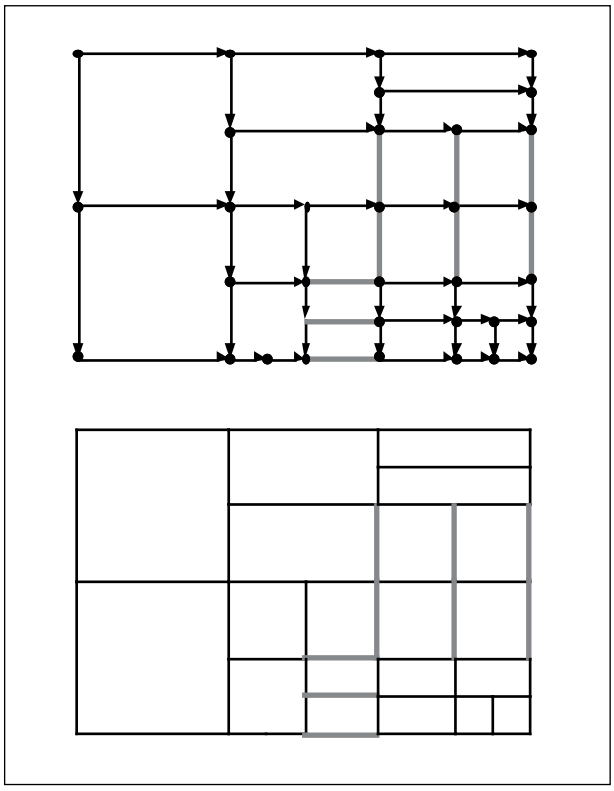


Figure 6.3.7

We will show somewhat more structure on the multiset p.o. If we have $X \geq_\mu Y$, there is a 'descendant' relation between the elements of X and Y. Some elements of X are 'preserved' in Y: this is indicated by heavy arrows (see Figure 6.3.8). Some elements of X will be replaced by some elements in Y that are strictly smaller (in the p.o. I); this is indicated by light arrows. Heavy arrows cannot split, light arrows can. From an element of X also zero light arrows can exit: that element just disappears. (E.g. the '1' in Figure 6.3.8.) A descendant relation for $X \geq_\mu Y$ by means of 'multiset arrows' need not be unique, e.g. the pair of multisets in Figure 2.8 admits several other descendant relations.

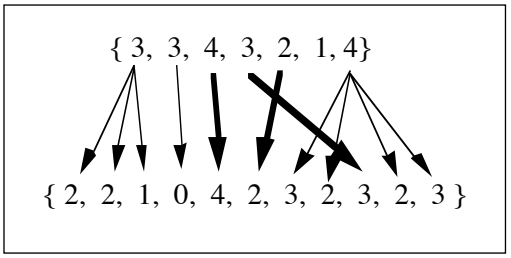


Figure 6.3.8

(Actually in the present treatment of this topic we will suppress all detailed proofs and therefore not really use this extra descendant structure for $X \geq_\mu Y$, but mention of these heavy and light tracing arrows anticipates their use in the next topic.)

6.3.**9. Monotonic filtering.** We start with an important definition. Given a tuple σ of natural numbers, *filter*(σ) is the tuple obtained by 'reading' σ from left-to-right, removing the elements that are less than what was already encountered, and taking the tuple of the remaining elements. See example in Figure 6.3.9. Another operation on tuples is *multiset*; it yields the corresponding multiset. In the sequel we will be especially interested in *multiset* (*filter*(σ)).

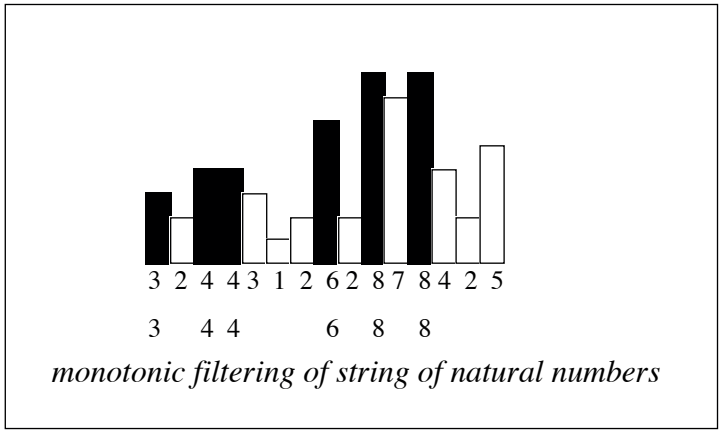


Figure 6.3.9

6.3.10. Decreasing diagrams. Before defining what a decreasing diagram is, we need the notion of 'norm of a reduction sequence' in the ARS with indexed rewrite relations. This will be a tuple of natural numbers (in general, elements of I). Par abus de langage, we will also denote reduction sequences with σ, τ. If σ is a reduction sequence, *label*(σ) is the string of indexes of consecutive reduction steps in σ. Single steps will be denoted by α, β. So *label*(α) is the index of the step α.

2.10.1. DEFINITION.

(i) Let σ be a reduction sequence. Then |σ|, the *norm* of σ, is *multiset*(*filter*(*label*(σ))).
(ii) The norm of two diverging reductions σ, τ is |σ| ∪ |τ| . (Figure 6.3.10.)

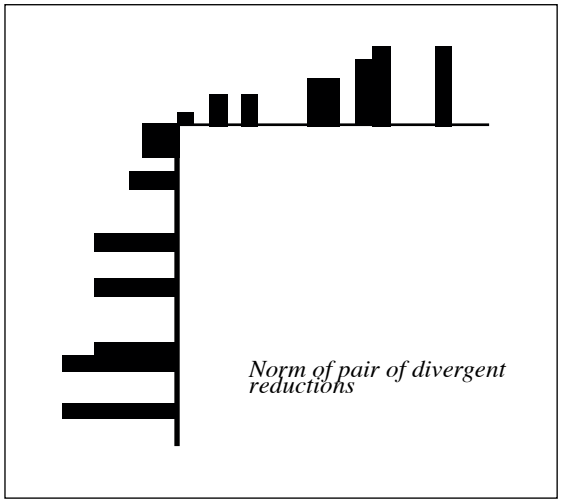


Figure 6.3.10

6.3.10.2. DEFINITION. Let σ: a ↠ b, τ: a ↠ c, σ': c ↠ d, τ': b ↠ d be reductions forming the reduction diagram $\mathcal{D}$ with corners a,b,c,d. (Figure 6.3.11.)

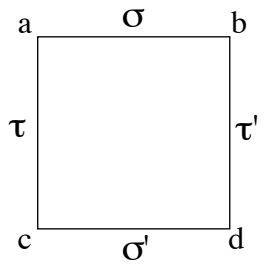


Figure 6.3.11

Then $\mathcal{D}$ is a *decreasing diagram*, if

$$|\sigma| \cup |\tau| \geq_\mu |\sigma\cdot\tau'| \text{ and}$$
$$|\sigma| \cup |\tau| \geq_\mu |\tau\cdot\sigma'|.$$

Note: we merely require $\geq_\mu$, not $>_\mu$!

6.3.11. Decreasing elementary diagrams. We will now see what the decreasingness condition means for elementary diagrams. Some consideration shows readily that decreasing e.d.'s have the following shape, as in Figure 6.3.12.
Explanation: Given two diverging steps a $\rightarrow_n$ b and a $\rightarrow_m$ c with indices n, m there is a common

reduct d such that

$$b \twoheadrightarrow_{<n} \cdot \to^{\equiv}_{m} \cdot \twoheadrightarrow_{<n \text{ or } <m} d \text{ and dually}$$

$$c \twoheadrightarrow_{<m} \cdot \to^{\equiv}_{n} \cdot \twoheadrightarrow_{<n \text{ or } <m} d$$

So from b we take some steps with indices < n, followed by 0 or 1 step with index m, followed by some steps with index < n or < m, with result d. Dually, from c we have a reduction to d as indicated.

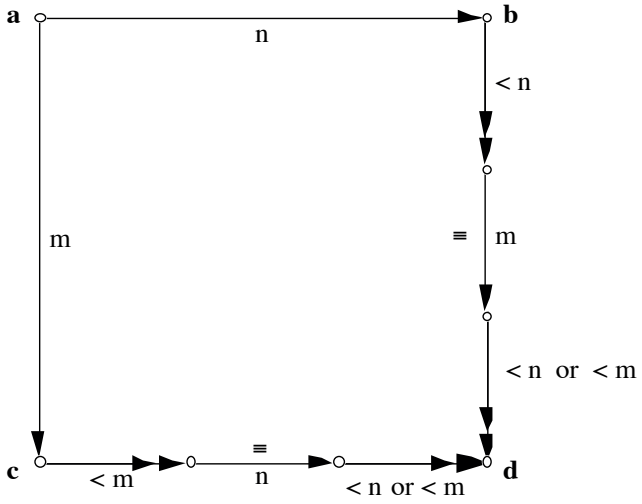


Figure 6.3.11

6.3.11.1. EXAMPLE. (i) So, we have examples as in Figure 6.3.13 of some decreasing and non-decreasing e.d.'s. (Note that the first e.d., upper-left, is an e.d. encountered in the braids confluence problem; so confluence of braids cannot be proved by an appeal on the theorem in this topic about confluence by decreasing diagrams!)

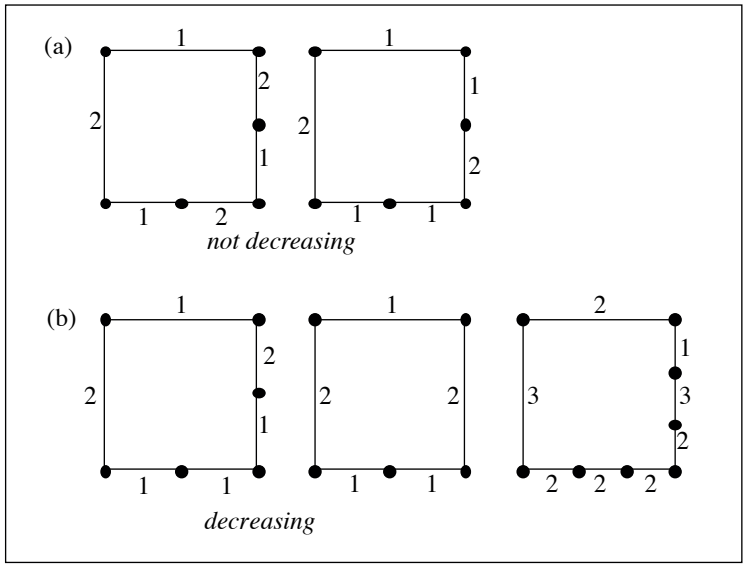


Figure 6.3.13

(ii) The e.d. in Example 6.3.2, Fig. 6.3.2 is not decreasing; of the e.d.'s in Fig.6.3.3 the first one is not decreasing, the other two are; the e.d.'s in Fig. 6.3.4 are decreasing; the e.d.'s in Fig. 6.3.5 are decreasing.

Now we will mention the two important properties of decreasing diagrams that give confluence. The first is indicated in Figure 6.3.14: pasting preserves decreasingness.

6.3.12. PROPOSITION. *Let two decreasing diagrams be joined as in Figure 6.3.14. Then the resulting diagram is again decreasing.*

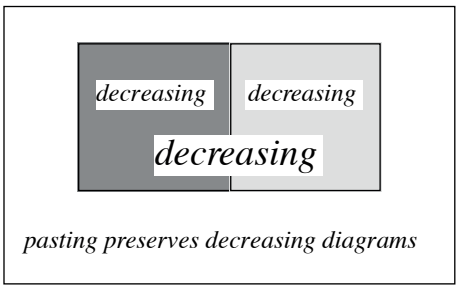


Figure 6.3.14

The second important property is indicated in Figure 2.15: inserting a decreasing diagram in a pair of co-initial reductions reduces the norm of the resulting pair of co-initial reductions.

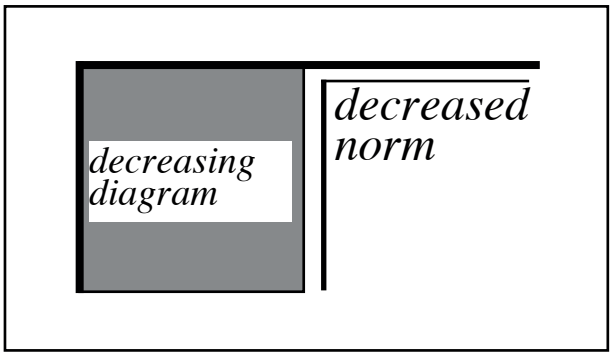


Figure 6.3.15

6.3.13. PROPOSITION. *Let a decreasing diagram be inserted as in Figure 6.3.15 into a pair of diverging reductions. Then the resulting pair of diverging reductions has a smaller norm.*

Finally, we can combine the two important properties to yield a proof of confluence, based on well-founded induction. See Figure 6.3.16.

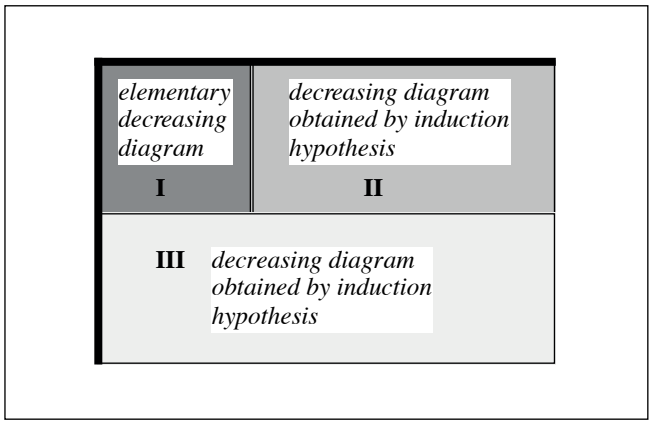


Figure 6.3.16

Let us give this final argument a bit more explicitly. See Fig. 6.3.17. The original norm is norm ($\beta.\tau$, $\alpha.\sigma$) = $|\beta.\tau| \cup |\alpha.\sigma|$. The induction hypothesis (IH) states that for all pairs of divergent reductions with smaller norm, tiling with decreasing e.d.'s succeeds. We have $|\beta| <_\mu |\beta.\tau|$, so norm ($\beta$, $\alpha.\sigma$) $<_\mu$ norm ($\beta.\ \tau$, $\alpha.\sigma$). Now norm (β', σ) $<_\mu$ norm (β, $\alpha.\sigma$) $<_\mu$ norm ($\beta.\ \tau$, $\alpha.\sigma$). Part I + II is again decreasing by Proposition 6.3.12. Hence (Proposition 6.3.13) norm (τ, $\alpha'.\sigma'$) $<_\mu$ norm ($\beta.\ \tau$, $\alpha.\sigma$). Now IH yields part III.

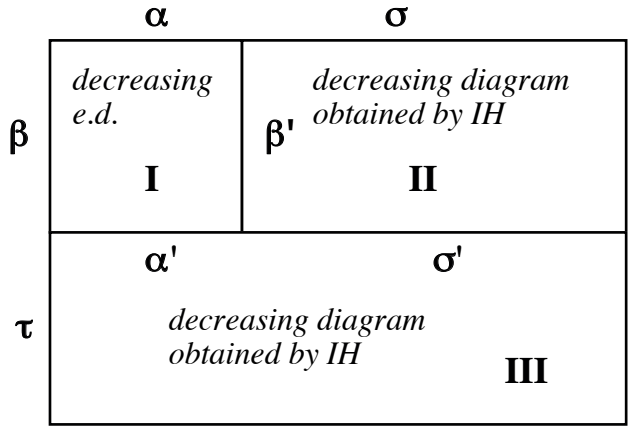


Figure 6.3.17

6.3.14. THEOREM (De Bruijn - Van Oostrom)
Every ARS with reduction relations indexed by a well-founded partial order I, *and satisfying the decreasing criterion for its e.d.'s, is confluent.*

6.3.15. REMARK. The unpublished note De Bruijn [78] is the first appearance of this theorem. There an asymmetrical version of the notion of decreasing elementary diagram is given. The notion of 'decreasing' as presented in this section was not present there and appears in Van Oostrom [94, 94a]. In Bezem et al. [96] the theorem is proved using the notion of 'trace-decreasing' which is slightly stronger than decreasing.

6.3.16. REMARK. The question arises how strong this method of decreasing diagrams is. In a way, it is best possible, at least for countable ARSs, since there is the following 'completeness' result:
Define an ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ to have the property DCR (decreasing Church-Rosser), if there is an indexed ARS $\mathcal{A}^I = \langle A, (\rightarrow_\alpha)_{\alpha \in I} \rangle$ with rewrite relations $(\rightarrow_\alpha)_{\alpha \in I}$, such that $\mathcal{A}^I$ has decreasing e.d.'s with respect to some well-founded order on I, and such that the union of the rewrite relations $\rightarrow_\alpha$ is $\rightarrow$. So we have seen above that DCR $\Rightarrow$ CR. Now we have:

THEOREM (Van Oostrom [94]. *For countable ARSs*: DCR $\Leftrightarrow$ CR.

The proof, also present in Bezem et al. [96], employs the fact that: CR $\Leftrightarrow$ CP for countable ARSs.

It seems to be a difficult exercise to establish the (conjectured) result that the condition 'countable' is necessary.

6.3.17. Some applications. We have already seen some applications in the examples of decreasing diagrams.More interesting is that also Huet's Strong Confluence Lemma in Example 6.3.1 and Newman's Lemma are corollaries of confluence by decreasing diagrams. For both, a little trick is

required, as follows.

(i) *Huet's strong confluence lemma*. Note that e.d.'s corresponding to Figure 6.3.1(a), the assumption of strong confluence, are in general not decreasing. E.g. one with two steps in the lower side is not. Yet, this situation can be seen in the scope of the present method as follows. Take two copies of $\rightarrow$, one labeled with h (horizontal): $\rightarrow_h$, one with v (vertical): $\rightarrow_v$. Now the e.d.'s as in Figure 6.3.18 are decreasing, under the ordering h < v.

Therefore, $\mathcal{A}^{\{h,v\}} = \langle A, (\rightarrow_h, \rightarrow_v) \rangle$ with the three types of e.d.'s as shown, is CR by the theorem. This means that the original $\mathcal{A} = \langle A, \rightarrow \rangle$ is also CR.

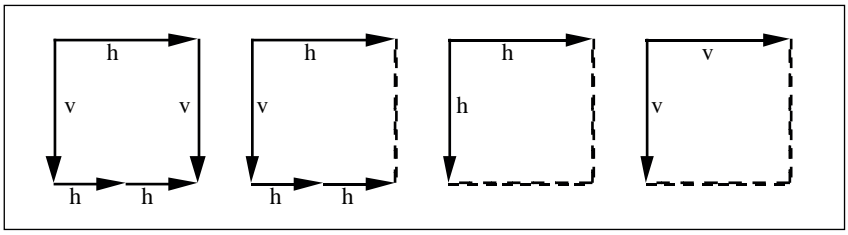


Figure 6.3.18

(ii) *Newman's Lemma*. Let ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ be SN and WCR. For the sake of exposition, let us assume that $\mathcal{A}$ is FB (finitely branching); then by SN each reduction graph G(a), a $\in$A, is finite, using Konig's Lemma. Let a be the cardinality of G(a), so a is a natural number. Call it the size of a. Note that if $a \rightarrow b$, then $a > b$. Now take for each step $a \rightarrow b$, an indexed relation $\rightarrow_a$ such that $a \rightarrow b \Leftrightarrow a \rightarrow_a b$; in other words, label each step with the size of its left-hand side element. Now it is not hard to see that WCR gives us decreasing e.d.'s. Hence the labeled ARS is CR, and therefore the original one also.

The general argument, not assuming FB, is equally simple, using well-founded trees as size of an element instead of natural numbers.

References

BEZEM, M., KLOP, J.W. & VAN OOSTROM, V.,(1996) Diagram Techniques for Confluence. Information and Computation, Vol.141, No.2, p.172-204, 1998.

DE BRUIJN, N.G. (1978). *A note on weak diamond properties*. Memorandum 78-08, Eindhoven University of Technology, August 1978.

HUET, G. (1980). *Confluent reductions: Abstract properties and applications to term rewriting systems*. JACM, Vol.27, No.4 (1980), 797-821.

VAN OOSTROM, V. (1994). *Confluence for Abstract and Higher-Order Rewriting*. Ph.-D. thesis, Vrije Universiteit, Amsterdam, March 1994.

VAN OOSTROM, V. (1994a). *Confluence by decreasing diagrams*. Theoretical Computer Science 126. p.259-280.

Iterative Path Orders

Extended abstract

Jan Willem Klop
Department of Theoretical Computer Science, Vrije Universiteit,
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands;
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands;
Department of Computer Science, University of Nijmegen, Toernooiveld 1,
6525 ED Nijmegen, The Netherlands
jwk@cs.vu.nl

Vincent van Oostrom[1]
Department of Philosophy, Utrecht University,
P.O.Box 80089, 3508 TB Utrecht, The Netherlands;
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

oostrom@phil.uu.nl

Roel de Vrijer
Department of Theoretical Computer Science, Vrije Universiteit,
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
rdv@cs.vu.nl

In the first half of this paper we give an alternative version of the recursive path order (RPO) for first-order term rewriting, which is 'iterative' rather than recursive. Hence the name iterative path order (IPO).

In the second part of the paper we prove that IPO is a well-founded order, by a simple argument familiar from Proof Theory. To this end we employ a labeled extension of IPO.

The result is an easy to grasp and powerful termination proof technique, whose correctness proof avoids the usual appeal to Kruskal's Tree Theorem.

We finally argue that the method is extendible to the lexicographic case and to the higher-order case.

The IPO method is easily seen to be as powerful as the usual RPO. In fact, the orderings are equivalent. This result requires a fine-grained analysis of IPO, that is added in an Appendix.

Contents

[1] Contact author: Vincent van Oostrom, tel. +31 30 2532761; fax +31 30 2532816.

0. Introduction

We are concerned with proving termination (or Strong Normalization, SN) for first order term rewriting systems. There are various methods to do so–see Dershowitz & Jouannaud [90], Zantema [02]. Of all these methods, probably the most well-known and most powerful is the method of recursive path orders (RPO), developed by Dershowitz [82,87] on the basis of Plaisted's precursor of that notion, associative path orders Plaisted [78]. The method has been widely paraphrased and extended.

There are two 'problems' with the RPO method. The first is that its definition is a somewhat complicated recursive definition, which is not easily digested. In fact, it takes a non-trivial analysis to demonstrate that the usual definition indeed is well-defined, as pointed out in Ferreira [96], Zantema [02].

The second is that it is difficult to show the essential property of RPO, namely that it is a well-founded order. The usual route is via the powerful and beautiful Kruskal Tree Theorem (KTT). (See Kruskal [60].)This difficulty becomes manifest in class-room: a full treatment of KTT is often beyond the scope of an introductory course.

In the present paper both 'problems' are eliminated. We give an 'operational' definition of RPO using a 'meta'-rewrite system that is neither terminating nor confluent. The definition could be called 'iterative' rather than 'recursive', hence our name 'iterative path orders' (IPO).

The distinctive feature of this meta-rewrite system is the use of a marker '*' with intuitive content: 'make this term smaller'. Thus, if t is a term of the TRS to be proved SN, then t* ambiguously denotes some term smaller than t, in the order that we aim to define.

At this point we have only solved the first problem; the definition of IPO using the starred terms t* is conceptually easy, certainly well-defined, and simple in its use. But we are not home-free; it is not easy to prove that IPO is a strict p.o.; and for the well-foundedness we still need the appeal to KTT. In the second part of the paper, this problem is overcome, as follows.

Recently we noted an extension of the IPO using terms with natural number labels instead of '*', that has all the benefits of the starred terms, but in addition employs a meta-rewrite system that is itself terminating, in contrast with the one described above. Thus the use of KTT is no longer necessary.

The upshot is that the method is now perfectly well suited for class-room. This didactical gain is not the only motivation however. The IPO method also may be interesting in itself, as it is readily extendible to the setting of lexicographic or multiset 'status', and with more effort, to the higher-order case. In addition, the IPO notion facilitates a closer analysis, as will be shown in the closing section of this paper.

1. Transformation rules on starred terms: IPO with stars

Let Σ be a first-order signature, and $>$ a strict partial ordering (called '*precedence ordering*') of the function and constant symbols of Σ. Let Σ^* be a copy of Σ where every function and constant symbol has a marker *:

$$\Sigma^* = \{F^* \mid F \in \Sigma\}.$$

Now we consider the signature $\Sigma \cup \Sigma^*$ of marked and unmarked symbols. Variables will not be marked. Furthermore, the arity of F* equals the arity of F. Consider the following TRS $\mathfrak{R}$ on $(\Sigma \cup \Sigma^*)$-terms. We will write the reduction relation of $\mathfrak{R}$ as $\rightarrow$, not the usual $\rightarrow$ that will be used for the TRS that we intend to prove SN. We write $\rightarrow^*$ for the transitive-reflexive closure of $\rightarrow$, and $\rightarrow^+$ for the transitive closure. With T we denote Ter(Σ), the set of unmarked terms, and with T^*, Ter($\Sigma \cup \Sigma^*$), the set of marked and unmarked terms. The four rules of $\mathfrak{R}$ are as in Table 1.1.

put	$F(\mathbf{x}) \rightarrow F^*(\mathbf{x})$	
copy	$F^*(\mathbf{x}) \rightarrow G(F^*(\mathbf{x}),...,F^*(\mathbf{x}))$	$(F > G)$
select	$F^*(x_1,...,x_n) \rightarrow x_i$	$(1 \le i \le n)$
down	$F^*(\mathbf{x}, G(\mathbf{y}), \mathbf{z}) \rightarrow F(\mathbf{x}, G^*(\mathbf{y}), \mathbf{z})$	

Table 1.1. Transformation rules for IPO with stars.

Here $\mathbf{x} \equiv x_1,...,x_n$, $\mathbf{y} \equiv y_1,...,y_m$, $\mathbf{z} \equiv z_1,...,z_k$ are disjoint lists of pairwise different variables; in rule *copy* the right-hand side has just as many copies of F*(**x**) as the arity of G admits; in rules *put*, *select* F, F* are n-ary; in rule *down*, the arity of F, F* is n+k+1, the arity of G,G* is m. The rules are actually rule schemes, e.g. the rule scheme *down* stands for all for all possibilities that respect the arities.

1.1. NOTATION. If $t \equiv F(\mathbf{t})$ with $\mathbf{t} = t_1,...,t_n \in \mathsf{T}^*$, then $t^* \equiv F^*(\mathbf{x})$.

1.2. EXAMPLE. Let Σ contain binary symbols F, H, a unary G and constants A, B with the ordering F > H and B > A. Then we have in T^* the reduction (which is rendered in Figure 1.1 in tree format):

$$\begin{aligned}
&F(A, G(B)) && \rightarrow_{put}\\
&F^*(A, G(B)) && \rightarrow_{copy}\\
&H(F^*(A, G(B)), F^*(A, G(B))) && \rightarrow_{select}\\
&H(G(B), F^*(A, G(B))) && \rightarrow_{put}\\
&H(G(B), F^*(A, G^*(B))) && \rightarrow_{select}\\
&H(G(B), F^*(A, B)) && \rightarrow_{copy}\\
&H(G(B), H(F^*(A, B), F^*(A, B))) && \rightarrow_{select}\\
&H(G(B), H(A, F^*(A, B))) && \rightarrow_{select}\\
&H(G(B), H(A, B))
\end{aligned}$$

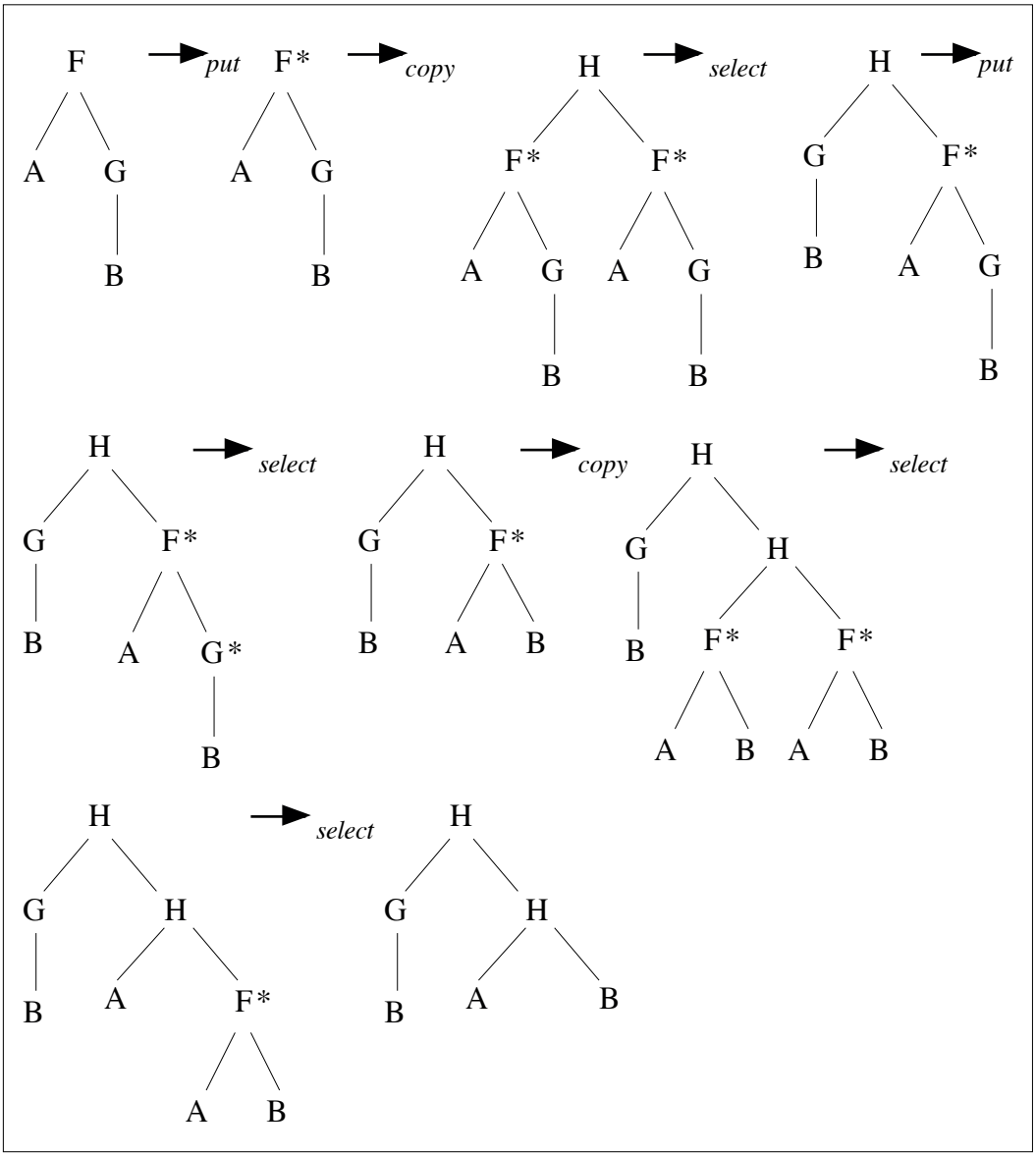


Figure 1.1. Reduction in $\mathfrak{R}$ (IPO with stars).

Clearly, $\mathfrak{R}$ is not CR, nor SN. For the latter, note that rule *copy* rewrites the left-hand side to a context of the left-hand side.

2. Application to termination proofs; examples

Let (Σ, R) be a TRS with finite Σ. We will be especially interested in the relation $\rightarrow^+$ restricted to the 'real' or 'proper' terms $Ter(\Sigma)$. In Example 1.2 we had $F(A, G(B)) \rightarrow^+ H(G(B), H(A, B))$. In fact, $\rightarrow^+$ on $Ter(\Sigma)$ is the partial order that we aim to establish. It is a strict, well-founded partial order. Both properties are not easily proved. Transitivity of $\rightarrow^+$ is trivial, but acyclicity is hard: there is no $t \in Ter(\Sigma)$ such that $t \rightarrow^+ t$. Note that on the auxiliary, marked terms we do have cycles as Example 2.1

shows.

2.1. EXAMPLE. (In the setting of Example 1.2.)

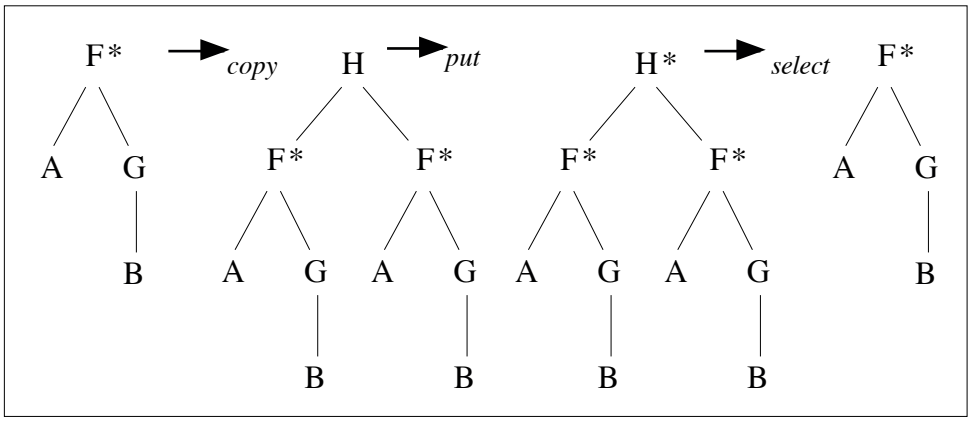


Figure 2.1. Cyclic reduction in $\mathfrak{R}$.

(Note that , hence, $\rightarrow$ is not SN on T^* .)

The well-foundedness of $\rightarrow^+$ on Ter(Σ) is even harder to show. For the moment we will put these issues aside, and demonstrate the use of $\rightarrow^+$, to be called *IPO with stars*, for termination proofs.

2.2. THEOREM. *Let* (Σ, R) *be a TRS with finite* Σ. *Suppose the function and constant symbols of* Σ *can be patially ordered in such a way that for the corresponding* IPO $\rightarrow^+$ *we have, for every reduction rule* $s \rightarrow t$ of R, *that* $s \rightarrow^+ t$.

Then (Σ, R) *is SN.*

PROOF. (1) $\rightarrow^+$ on Ter(Σ) is a strict, well-founded partial order. (To be proved later!)
(2) $\rightarrow^+$ is closed under contexts and substitutions:

$$s \rightarrow^+ t \Rightarrow C[t^\sigma] \rightarrow^+ C[s^\sigma].$$

So, given an infinite reduction $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots$ in R, we would have by (2) and the assumption on reduction rules, that

$$t_0 \rightarrow^+ t_1 \rightarrow^+ t_2 \rightarrow^+ \ldots,$$

contradicting the well-foundedness of $\rightarrow^+$ on Ter(Σ). ❑

We now give some examples of termination proofs using Theorem 2.2.

2.3. EXAMPLE. Let R be the TRS specifying addition A and multiplication M on the natural numbers generated by zero 0 and successor S, with rules as in Table 2.1:

$A(x, 0) \to x$
$A(x, S(y)) \to S(A(x, y))$
$M(x, 0) \to 0$
$M(x, S(y) \to A(x, M(x, y))$

Table 2.1. Addition and multiplication on natural numbers.

Adopting the precedence order $M > A > S$, we can check that indeed for these four rules $s \to t$ we have $s \to^+ t$. E.g. for the fourth rule (see Figure 2.2):

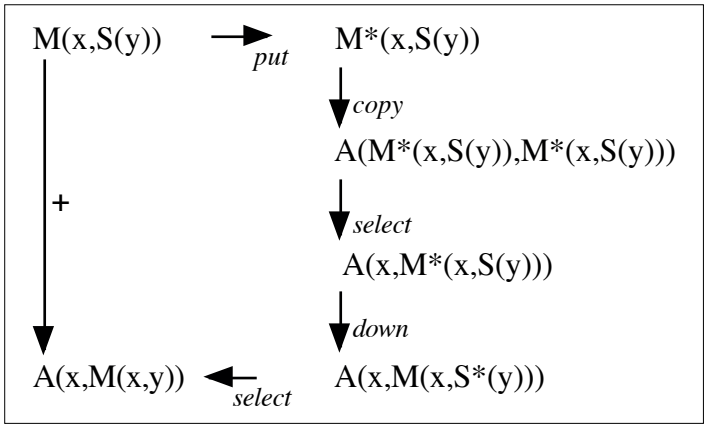


Figure 2.2. $M(x,S(y)) \to^+ A(x,M(x,y))$

2.4. EXAMPLE (Dershowitz & Jouannaud [90]. Consider the string rewrite system R given by the four rules as in Table 2.2.

$10 \to 0001$
$01 \to 1$
$11 \to 0000$
$00 \to 0$

Table 2.2. String rewrite system on 0,1-words.

So we have e.g. the reduction

$1\mathbf{10}1 \to 1\mathbf{00}011 \to \mathbf{10}011 \to \mathbf{00}01011 \to$
$0010\mathbf{11} \to 00\mathbf{10}0000 \to 0000010000 \to \ldots$

To capture this string rewrite system in the framework of term rewriting, we perceive the symbols 0,1 as unary function symbols and read the rules accordingly; e.g. $10 \to 0001$ is the term rewrite rule

$$1(0(x)) \to 0(0(0(1(x)))).$$

To show SN for R, we adopt the precedence order $1 > 0$ and check, e.g. for the displayed rule (dropping all brackets in the convention of association to the right- see Figure 2.3:)

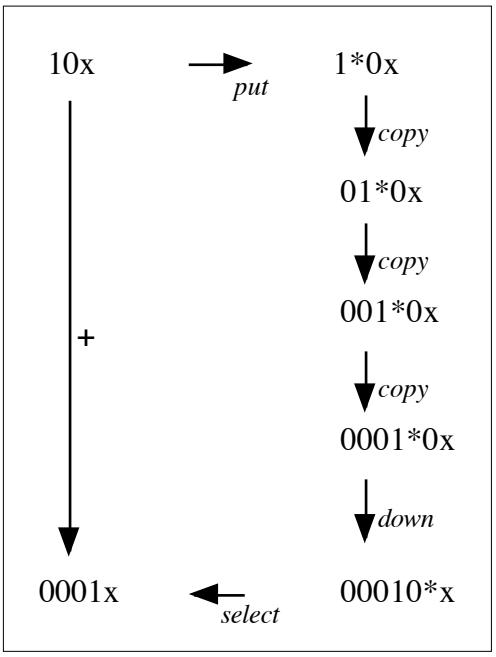


Figure 2.3. $10x \to^+ 0001x$

2.5. EXAMPLE. (*Primitive recursion.*) Given a TRS which is IPO-terminating, and having a unary function symbol g and a ternary function symbol h, we can define a binary function f on natural numbers by the rules:

$f(0,x) \to g(0)$
$f(S(n),x) \to h(f(n,x),n,x)$

The resulting system is IPO-terminating again by taking the precedence order such that $f > g$ and $f > h$.

3. Transformation rules on labeled terms: IPO with labels

In Figure 3.1 we review in an abstract way the situation at hand. Figure 3.1(a) sets the goal: to prove SN for the reduction relation $\rightarrow_1$. This can be done as in Figure 3.1(b), which depicts the method in the first half of this paper: IPO with stars.

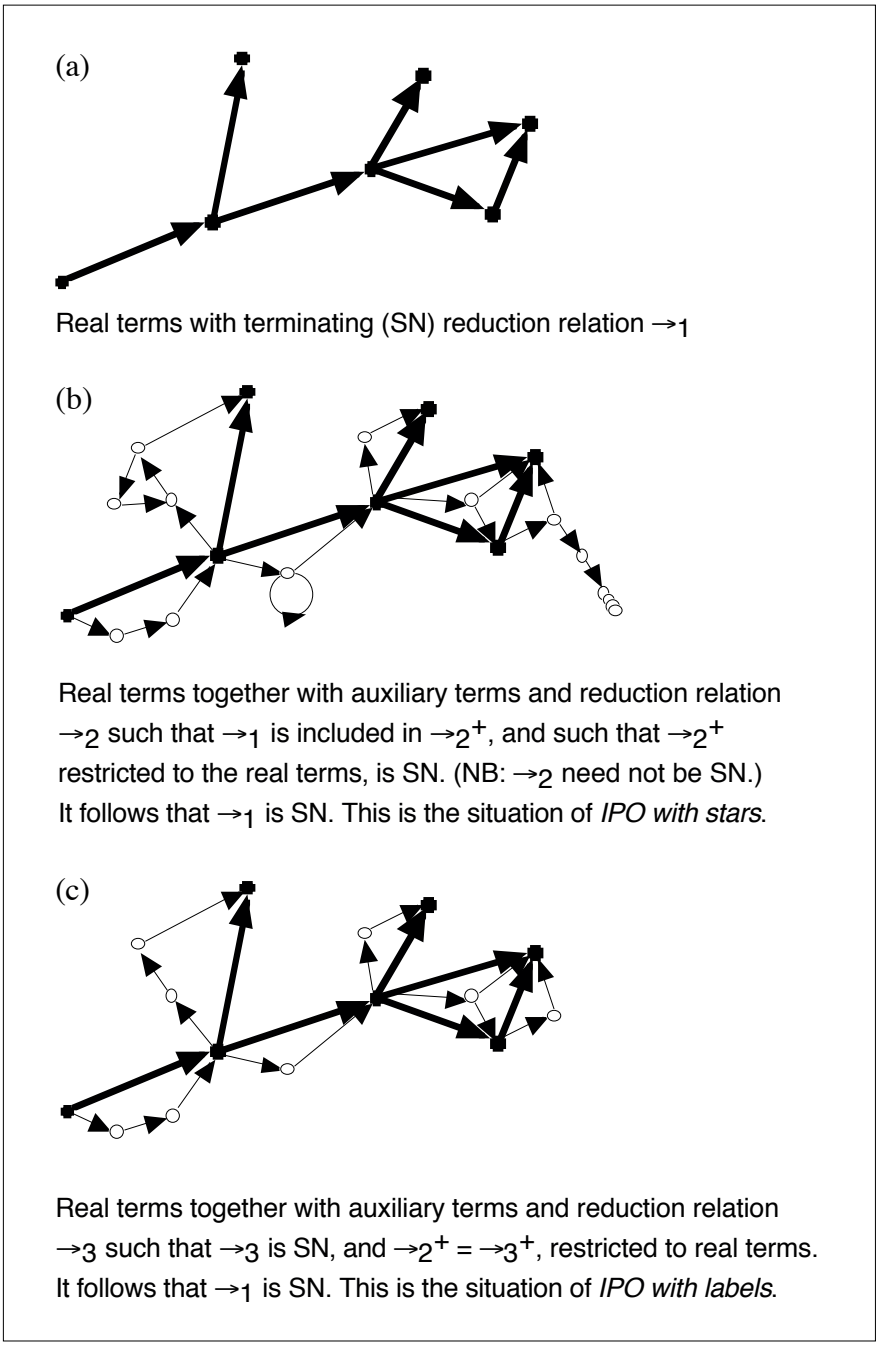


Figure 3.1. Proper and auxiliary terms

We will now improve that situation, as in Figure 3.1(c), by 'refining' IPO with stars. There,

the marker * could be seen as a quantum of energy, enabling the term t* to 'stay alive for a while'. But this energy quantum was unspecified. We will now assign instead of *, precise energy quanta in the form of natural numbers. Thus we have next to the real terms Ter(Σ), *labeled* terms Ter($\Sigma^{\mathbb{N}}$) where

$$\Sigma^{\mathbb{N}} = \{F^k \mid F \in \Sigma, k \in \mathbb{N}\}.$$

In analogy with Section 1, we consider now the set Ter($\Sigma \cup \Sigma^{\mathbb{N}}$) of labeled and unlabeled terms. As before, the arity of F^k equals the arity of F.

Next, we consider the following TRS $\mathfrak{R}^{\mathbb{N}}$ on ($\Sigma \cup \Sigma^{\mathbb{N}}$)-terms. We will write the reduction relation of $\mathfrak{R}^{\mathbb{N}}$ as before as $\rightarrow$, an overloading that will be solved by the context. The transformation rules of $\mathfrak{R}^{\mathbb{N}}$ are in Table 3.1:

put	$F(\mathbf{x}) \rightarrow F^k(\mathbf{x})$	$(k \in \mathbb{N})$
copy	$F^{k+1}(\mathbf{x}) \rightarrow G(F^k(\mathbf{x}),...,F^k(\mathbf{x}))$	$(F > G, k \in \mathbb{N})$
select	$F^k(x_1,...,x_n) \rightarrow x_i$	$(1 \le i \le n, k \in \mathbb{N})$
down	$F^k(\mathbf{x}, G(\mathbf{y}), \mathbf{z}) \rightarrow F(\mathbf{x}, G^{k'}(\mathbf{y}), \mathbf{z})$	$(k, k' \in \mathbb{N})$

Table 3.1. Transformation rules for IPO with labels.

The same arity conventions apply as for the analogous rules in Table 1.1 for IPO with stars. Our first observation is that when we replace in Table 3.1 all k, k' by *, we arrive at the rules of Table 1.1. Note furthermore the remarkable fact in the labeled rule *down*: the label k, when pushed down to the subterm G(**y**), may jump to a much higher label k'!

We now arrive at the first important fact about IPO with labels, that we can paraphrase as:

IPO with stars = IPO with labels

More precisely:

3.1. THEOREM. *Let* t,s ∈ Ter(Σ). *Then:*

$$t \rightarrow^+_{stars} s \Leftrightarrow t \rightarrow^+_{labels} s.$$

Here $\rightarrow_{stars}$ refers to the rules in Table 1.1, and $\rightarrow_{labels}$ refers to the rules in Table 3.1. Let us simplify this notation by writing $\rightarrow_s$ and $\rightarrow_{\mathbb{N}}$ respectively.

PROOF. ($\Leftarrow$) Given a labeled reduction $t \rightarrow^+_{\mathbb{N}} s$, we replace all labels by stars *, thus arriving by our

observation above, at a starred reduction $t \to^+_s s$.

($\Rightarrow$) Consider a starred reduction $t \equiv t_n \to_s t_{n-1} \to_s \ldots \to_s t_0 \equiv s$. We will translate it stepwise to a labeled reduction $t'_n \to\!| \; t_{n-1} \to\!| \ldots \to\!| \; t'_0$. This is done such that:

1. Each t'_i is a labeled variant of t_i, i.e., obtained from t_i by replacing each * by some natural number. (So, since t,s are unstarred, we again have $t'_n \equiv t$ and $t'_0 \equiv s$.)

2. A step $t'_{i+1} \to\!| \; t'_i$ corresponds to the original step $t_{i+1} \to_s t_i$ by using the same rule (*put*, *select*, etc.) to the subterm at the same position in t'_{i+1}.

So, for example, a starred step

$$t_{i+1} \equiv H(F^*(r),a^*) \to_{copy} H(G(F^*(r),F^*(r),a^*)) \equiv t_i$$

will be translated to a labeled step

$$t'_{i+1} \equiv H(F^{k+1}(r'),a^m) \to_{copy} H(G(F^k(r'),F^k(r'),a^m)) \equiv t'_i.$$

This example illustrates that for the label of F in t'_{i+1} we have to make sure that it is not 0, since otherwise the *copy*-step would be blocked.

Note that, granted this no-blocking requirement, the translation is almost determined by what is said in 1. and 2. The only remaining choices are the labels k introduced in *put*-steps and the labels k' introduced in *down*-steps. These choices are fixed by 3.

3. Let $t_{i+1} \to_s t_i$ be either a *put*- or a *down*-step. In both cases we choose for the new label in t'_i the number i.

We now have the following invariant throughout the reduction $t'_n \to\!| \; t_{n-1} \to\!| \ldots \to\!| \; t'_0$:

For all labels m *in the term* t'_i *we have* $m \geq i$.

This is easily verified. In t'_n there are no labels; hence the statement is trivially true. For a label m in t'_i we have the following possibilities.

- It is copied from t'_{i+1}, in which case we have even $m \geq i+1$.
- It results by a *copy*-step from a label m+1 in t'_{i+1}.

Then $m+1 \geq i+1$, hence $m \geq i$.

- It is introduced by a *put*- or a *down*-step. Then we have $m = i$.

The invariant implies that in the terms $t'_n, t'_{n-1}, \ldots t'_1$ all labels are non-zero. Hence the requirement that a *copy*-step is never blocked, is fulfilled.

Alternative proof of (⇒). Given a starred reduction from t to s.

We use the notion of "raising a labeling by 1". Let t be a labeled term. We can raise all its labels by 1. That is, an occurrence of a labeled function symbol F^m is replaced by F^{m+1}. Function symbols without label are left untouched. So, e.g., raising the labels of the term $F(G^6(x,a^0))$ by 1 yields the term $F(G^7(x,a^1))$. If the original labeling term was t^L (with labeling L) we denote the result by t^{L+1}.

Also a reduction step can be raised. One easily verifies that if $t^L \to\!|\ s^{L'}$, then also $t^{L+1} \to\!|\ s^{L'+1}$. A reduction sequence can be raised by raising all its steps.

We want to translate a starred reduction $t \equiv t_0 \to_s t_1 \to_s \ldots \to_s t_n \equiv s$. We do this by, starting out with t_0, performing exactly the same reduction steps, only now taking each time a labeled variant. For the rules *copy* and *select* this can be done uniquely, in an application of the rules *put* and *down* a new label k or k' has to be chosen. Just do this arbitrarily.

There is only one potential problem: one may get stuck if one needs to use the rule *copy* on a function symbol of which the label happens to be 0. Say this happens after i steps. So we have already reached a partial result $t \equiv t'_0 \to\!|\ \ldots \to\!|\ t'_i$, but the intended *copy*-step from t'_i is blocked. Then just raise the whole reduction $t \equiv t'_0 \to\!| \ldots \to\!|\ t'_i$ by 1 to a reduction $t \equiv t''_0 \to\!|\ \ldots \to\!|\ t''_i$. The new end term t''_i has no label 0 anymore, hence the required *copy*-step $t''_i \to\!|\ t''_{i+1}$ can be adjoined. And so on till n. ❑

Next we state the remarkable difference between $\to_s$ and $\to\!|$:

3.2. THEOREM. *The rewrite relation* $\to\!|$ *is SN.*

PROOF. We will prove the statement

$$\forall l\ \forall F \in \Sigma\ \forall t_1,\ldots,t_n \in SN\ F^l(t_1,\ldots,t_n) \in SN$$

where $t_1,\ldots,t_n \in Ter(\Sigma \cup \Sigma^l)$, SN is the set of strongly normalizing terms (with respect to $\to\!|$), and l is a label $\in \mathbb{N}$ or absent.We will prove the statement using well-founded induction to the triple

$$(F, (t_1,\ldots,t_n), l)$$

with a lexicographical ordering as follows:

(1) the first item in the triple refers to the well-founded precedence order of the function symbols F;

(2) the second item is the 'product order'of the well-founded orders of the t_i (i = 1,...,n) that were assumed to be SN;

(3) the third item is the label l of F in $F^l(t_1,...,t_n)$; it is a natural number n or ω in case l is absent. We have $\omega > n$, for all n.

Now in order to prove the statement, it clearly suffices to prove that the *one step reducts* of $F^l(t_1,...,t_n)$are SN.

So, consider a term $F^l(t_1,...,t_n)$, where by assumption the t_i (i = 1,...,n) are SN. We have the following induction hypothesis (IH):

> *For all terms* $F'^{l'}(t'_1,...,t'_{n'})$ *such that* $t'_1,...,t'_{n'}$) *are SN, and of which the triple is less than that of* $F^l(t_1,...,t_n)$*, we have SN.*

We now check successively all possible one step reducts of $F^l(t_1,...,t_n)$, and prove these SN.

Case 1. The reduction step is internal:

$$F^l(t_1,...,t_n) \rightarrow_l F^l(t_1,...,t_{i'},...,t_n).$$

Then the corresponding triple decreases in the second coordinate, so by IH $F^l(t_1,...,t_{i'},...,t_n) \in$ SN.

Case 2. The reduction step is

$$F(t_1,...,t_n) \rightarrow_{put} F^k(t_1,...,t_n).$$

Then we have a decrease in the third coordinate of the corresponding triple, since the 'no-label' (*i.e.* ω) is greater than k.

Case 3. $F^l(t_1,...,t_n) \rightarrow_{select} t_i$. By assumption, $t_i \in$ SN.

Case 4. $F^{k+1}(\mathbf{t}) \rightarrow_{copy} G(F^k(\mathbf{t}),...,F^k(\mathbf{t}))$. Here F > G in the precedence order. By IH, the copied arguments $F^k(\mathbf{t})$ of G are SN since k+1 > k. By a second application of IH, the G-term itself is SN since F > G.

Case 5. $F^k(\mathbf{t}, G(\mathbf{s}), \mathbf{r}) \rightarrow_{down} F(\mathbf{t}, G^{k'}(\mathbf{s}), \mathbf{r})$ $(k, k' \in \mathbb{N})$

By assumption, the arguments **t**, G(**s**), **r** are SN. So $G^{k'}(\mathbf{s})$ is also SN, being a reduct of G(**s**) by the *put* rule. Now $F(\mathbf{t}, G^{k'}(\mathbf{s}), \mathbf{r})$ is less with respect to the triple, due to a decrease in the second component. So by IH, $F(\mathbf{t}, G^{k'}(\mathbf{s}), \mathbf{r}) \in$ SN.

Having proved the statement in the beginning of the proof, we are through: a simple induction to term formation finally shows that *every* term $F^l(t_1,...,t_n)$is SN. ❑

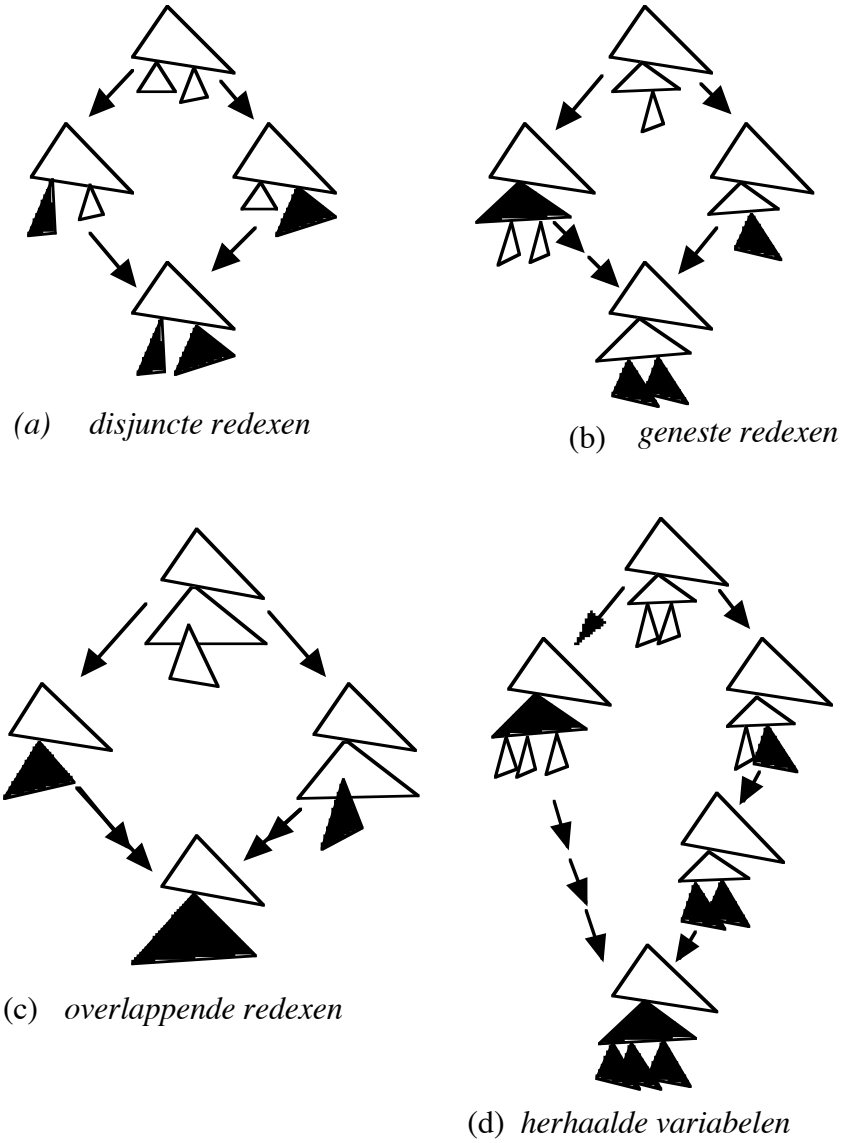
8

COMPLETERING

INHOUD.

6.0. *Introductie.*

6.1. *Het kritieke paren lemma*



(a) *disjuncte redexen*

(b) *geneste redexen*

(c) *overlappende redexen*

(d) *herhaalde variabelen*

Figuur xx.

6.3. *Successor en predecessor*.

Consider the following equational specification (or theory) E of the integers ($\mathbb{Z}$) with 0, +, successor S and predecessor P (left column, (a)):

(a)		(b)		
	0 + x = x		(1)	0 + x → x
	x + 0 = x		(2)	x + 0 → x
	S(x) + y = S(x + y)		(3)	S(x) + y → S(x + y)
	x + S(y) = S(x + y)		(4)	x + S(y) → S(x + y)
	x + P(y) = P(x + y)		(5)	x + P(y) → P(x + y)
	P(S(x)) = x		(6)	P(S(x)) → x

Tabel xx

(Since this specification is intended to be symmetrical with respect to permuting S and P one might expect also the equations S(P(x) = x and P(x) + y = P(x + y) to be included in E. Actually these equations are derivable.)
*

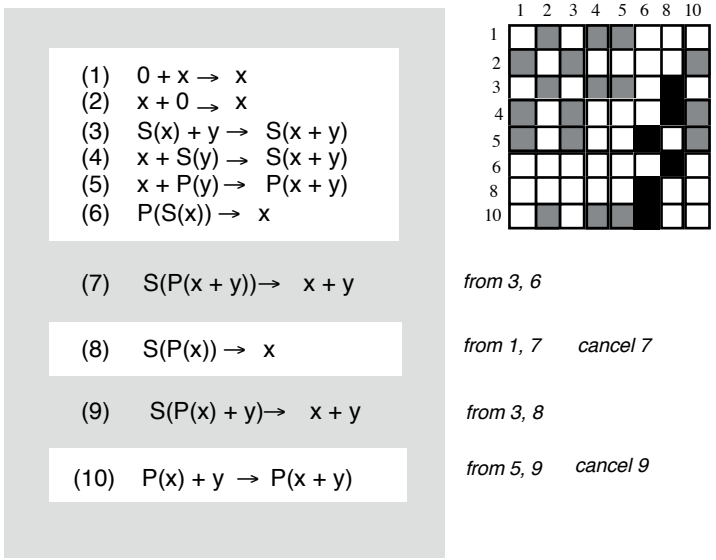


Figure 7

We will now perform an 'intuition guided' completion of E. First let us adopt as rewrite rules all equations from E, oriented from left to right. This yields rules (1 - 6) as above in column (b). This is not yet a complete TRS; though it is terminating (as will be seen later), it is not confluent. The reason is that there is an overlap between the left-hand sides of (3), (5) that is harmful: S(x) + P(y) reduces with (3) to S(x + P(y)) and with (5) to P(S(x) + y). These two terms form what is called a *critical pair*. The terms can be reduced further: S(x + P(y)) → S(P(x + y)) and P(S(x) + y) → P(S(x + y)) → x + y, but then we are stuck since S(P(x + y)) and x + y are normal forms with respect to (1 - 6). So confluence fails.

Actually, this is not the only critical pair generated by (1 - 6); there are also overlaps

between (1), (2), between (1), (4), between (1), (5), between (2), (3), between (3), (4), between (5), (6). But these critical pairs are harmless. For instance, (5), (6) yield the critical pair P(x + S(y)), x + y. These terms have a common reduct: P(x + S(y)) → P(S(x + y)) → x + y.

We try to solve the problem of non-confluence posed by the terms S(P(x + y)) and x + y in a drastic way: we simply add as a new rule (7) S(P(x + y)) → x + y. Now the critical pair given by (3), (5) is harmless too. However, new critical pairs arise: overlap between (1), (7) yields S(P(0 + y)) with as reducts 0 + y, S(P(y)). This causes us to adopt a new rule: (8) S(P(y)) → y. We can cancel (7) now, as it is a consequence of (8). We consider the possible overlaps: the overlap between (8), (6) is harmless; likewise between (8), (4). But not the one between (8), (3), which causes the introduction of rule (9): S(P(x) + y) → x + y. In this way we continue, and luckily after a few more steps as in Figure 7 we reach a successful conclusion: a TRS R where all critical pairs are harmless. Moreover, R is terminating; this can be seen by noting that all our rules were chosen such that they respected the recursive path ordering (to be explained in the next section) obtained by putting + > S and + > P.

Let us give a precise definition of critical pairs:

2.2.1. DEFINITION. Let the TRS R contain the rewrite rules r: t → s and r': t' → s'. Suppose r, r' are 'standardized apart', i.e. renamed such that they have no variables in common. Suppose furthermore that t ≡ C[u], u not a variable, and that u and t' can be unified with most general unifier σ. Then $t^\sigma \equiv C^\sigma[u^\sigma] \equiv C^\sigma[t'^\sigma]$ is subject to an r'-reduction as well as an r-reduction, with result: $C^\sigma[s'^\sigma]$ respectively s^σ. Now $\langle C^\sigma[s'^\sigma], s^\sigma \rangle$ is called a *critical pair* of R.

If r, r' are (renamed versions of) the same rewrite rule, we moreover require that the context C[ ] is not the trivial context.

Note that if C[ ] in the above situation is trivial (so that t, t' unify 'at the root') two critical pairs are obtained which are mirror-images: $\langle s'^\sigma, s^\sigma \rangle$ and $\langle s^\sigma, s'^\sigma \rangle$. Such critical pairs are sometimes called 'overlays'. (See the chess-board-like table in Figure 7, where it is mentioned which pairs of rules of our example above give rise to critical pairs. The grey squares denote overlays.)

A critical pair ⟨ s, t ⟩ is *convergent* if s, t have a common reduct (∃r s ↠ r & t ↠ r), notation: s ↓ t. The significance of the fact that all critical pairs ⟨ s, t ⟩ are 'harmless', i.e. convergent, is expressed by the following lemma.

2.2.1. CRITICAL PAIR LEMMA (Huet [80]). *A TRS is weakly confluent iff all its critical pairs are convergent*.

Convergence of all critical pairs is not sufficient for confluence; a counterexample is the ABCD-TRS in Figure 3 (Section 1.3), with critical pairs (overlays) $\langle A, C\rangle$, $\langle C, A\rangle$, $\langle B, D\rangle$, $\langle D, B\rangle$. However, in addition to termination, it is sufficient for confluence, according to Newman's Lemma, and we have:

2.2.2. THEOREM (Knuth & Bendix [70]). *A terminating TRS is confluent iff all its critical pairs are convergent*.

As we noted above, convergence of all critical pairs is sufficient for weak confluence, but not for confluence. Huet [80] gave a criterion for critical pairs, stronger than convergence, which does imply confluence while not requiring termination as in the Knuth-Bendix theorem. First define *parallel reduction* as follows: $t \to_{\|} s$ if t reduces to s via a reduction sequence consisting of contracting a set of disjoint redexes in t. Thus, if $t_i\sigma_i \to s_i\sigma_i$ (i = 1, ..., n) are contractions, i.e. instances of reduction rules $t_i \to s_i$ (i = 1, ..., n), then $C[t_1\sigma_1, ..., t_n\sigma_n] \to_{\|} C[s_1\sigma_1, ..., s_n\sigma_n]$.

2.2.3. THEOREM (Huet [80]). *Let $\mathcal{R}$ be a left-linear TRS such that we have* $s \to_{\|} t$ *for every critical pair* $\langle s, t\rangle$. *Then $\mathcal{R}$ is confluent.*

Note the direction involved here. As far as we know, it is an open problem whether the reverse condition also implies confluence: *for all critical pairs* $\langle s, t\rangle$ *we have* $t \to_{\|} s$. Also open seems to be the problem whether confluence is implied by the property: *for all critical pairs* $\langle s, t\rangle$ *we have* $s \to^{=} t$ *or* $t \to^{=} s$.

An important aspect in critical pair completion algorithms is that we need to have an ordering of terms at our disposal, guiding us (or the algorithm) how to choose orientations. Thus, at the start of a completion procedure, one must provide the algorithm with a so-called *reduction ordering* on terms; this is a well-founded partial order among terms which is closed under substitutions and contexts, i.e. if $s > t$ then $C[s\sigma] > C[t\sigma]$ for all substitutions σ and contexts C[ ].

The original specification E does not prove $x + y = y + x$ (this follows immediately from the fact that $x + y$ and $y + x$ are different normal forms of $\mathcal{R}$); yet for all ground terms t, s we have $E \vdash t + s = s + t$. That is: $x + y = y + x$ is valid in the initial algebra of E. Such an equation is called an *inductive theorem* (since its validity is usually proved with induction to the structure of ground terms). Completion techniques provide the means to prove inductive theorems without using induction ("inductionless induction"); another phrase in this respect is "proof by consistency". For an account of proof by consistency applications, see Dershowitz & Jouannaud [90] and Bachmair [88].

6.2. *Groepen*

5.3. Critical pair completion.

We resume the question how to find a complete TRS (for the case of open terms, henceforth) for an equational specification (Σ, E). This is in fact what the Knuth-Bendix completion algorithm is trying to do. We will now explain the essential features of the completion algorithm first by an informal, "intuition-guided" completion of the equational specification of groups:

$e \cdot x$	$=$	x
$I(x) \cdot x$	$=$	e
$(x \cdot y) \cdot z$	$=$	$x \cdot (y \cdot z)$

Tabel xx.

First we give these equations a 'sensible' orientation:

1.	$e \cdot x$	$\rightarrow$	x
2.	$I(x) \cdot x$	$\rightarrow$	e
3.	$(x \cdot y) \cdot z$	$\rightarrow$	$x \cdot (y \cdot z)$

(Note that the orientation in rules 1, 2 is forced, by the restrictions on rewrite rules in Section 2.1. As to the orientation of rule 3, the other direction is just as 'sensible'.) These rules are not confluent, as can be seen by superposition of e.g. 2 and 3. Redex $I(x) \cdot x$ can be unified (after variable renaming) with a *non-variable* subterm of redex $(x \cdot y) \cdot z$ (the underlined subterm), with result $(I(x) \cdot x) \cdot z$. This term is subject to two possible reductions: $(I(x) \cdot x) \cdot z \rightarrow e \cdot z$ and $(I(x) \cdot x) \cdot z \rightarrow I(x) \cdot (x \cdot z)$. The pair of reducts $\langle e \cdot z, I(x) \cdot (x \cdot z) \rangle$ is called a

critical pair, since the confluence property depends on the reduction possibilities of the terms in this pair. Formally, we have the following definition which at a first reading is not easily digested. For the concept of a 'most general unifier' we refer to Chapter 7 below.

5.3.1. DEFINITION. Let $\alpha \to \beta$ and $\gamma \to \delta$ be two rewrite rules such that α is *unifiable* (after renaming of variables) with a subterm of γ which is not a variable (a non-variable subterm). This means that there is a context C[ ], a non-variable term t and a 'most general unifier' σ such that $\gamma \equiv C[t]$ and $t^\sigma \equiv \alpha^\sigma$. The term $\gamma^\sigma \equiv C[t]^\sigma$ can be reduced in two possible ways: $C[t]^\sigma \to C[\beta]^\sigma$ and $\gamma^\sigma \to \delta^\sigma$.

Now the pair of reducts $\langle C[\beta]^\sigma, \delta^\sigma \rangle$ is called a *critical pair* obtained by the superposition of $a \to \beta$ on $\gamma \to \delta$. If $\alpha \to \beta$ and $\gamma \to \delta$ are the same rewrite rule, we furthermore require that α is unifiable with a proper (i.e. not $\equiv \alpha$) non-variable subterm of $\gamma \equiv \alpha$.

5.3.2. DEFINITION. A critical pair $\langle s, t \rangle$ is called *convergent* if s and t have a common reduct.

Our last critical pair $\langle e{\cdot}z, I(x){\cdot}(x{\cdot}z) \rangle$ is not convergent: $I(x){\cdot}(x{\cdot}z)$ is a normal form and $e{\cdot}z$ only reduces to the normal form z. So we have the problematic pair of terms z, $I(x){\cdot}(x{\cdot}z)$; problematic because their equality is derivable from E, but they have no common reduct with respect to the reduction available so far. Therefore we adopt a new rule

4. $I(x){\cdot}(x{\cdot}z) \to z$

Now we have a superposition of rule 2 and 4: $I(I(y)){\cdot}(I(y){\cdot}y) \to_4 y$ and $I(I(y)){\cdot}(I(y){\cdot}y) \to_2 I(I(y)){\cdot}e$. This yields the critical pair Ÿy, $I(I(y)){\cdot}e\pm$ which cannot further be reduced. Adopt new rule:

5. $I(I(y)){\cdot}e \to y$ canceled later

As it will turn out, in a later stage this last rule will become superfluous. We go on searching for critical pairs:

Superposition of 4, 1: $I(e){\cdot}(e{\cdot}z) \to_4 z$ and $I(e){\cdot}(e{\cdot}z) \to_1 I(e){\cdot}z$.

Adopt new rule:

6. $I(e){\cdot}z \to z$ canceled later

Superposition of 3, 5: $(I(Iy)){\cdot}e){\cdot}x \to_3 I(I(y)){\cdot}(e{\cdot}x)$ and $(I(Iy)){\cdot}e){\cdot}x \to_5 y{\cdot}x$.

Adopt new rule:

7. $I(Iy)){\cdot}x \to y{\cdot}x$ canceled later

Superposition of 5, 7: $I(I(y)){\cdot}e \to_7 y.e$ and $I(I(y)){\cdot}e \to_5 y$.

Adopt new rule:

8. $y \cdot e \to y$

Superposition of 5, 8: $I(I(y)) \cdot e \to_5 y$ and $I(I(y)) \cdot e \to_8 I(I(y))$.
Adopt new rule

9. $I(I(y)) \to y$ cancel 5 and 7

(Rule 5 is now no longer necessary to ensure that the critical pair $\langle y, I(I(y)) \cdot e \rangle$ has a common reduct, because: $I(I(y)) \cdot e \to_9 y \cdot e \to_8 y$. Likewise for rule 7.)
Superposition of 6, 8: $I(e) \cdot e \to_6 e$ and $I(e) \cdot e \to_8 I(e)$.
Adopt new rule

10. $I(e) \to e$ cancel 6

Superposition of 2, 9: $I(I(y)) \cdot I(y) \to_2 e$ and $I(I(y)) \cdot I(y) \to_9 y \cdot I(y)$.
Adopt new rule

11. $y \cdot I(y) \to e$

Superposition of 3, 11: $(y \cdot I(y)) \cdot x \to_3 y \cdot (I(y) \cdot x)$ and $(y \cdot I(y)) \cdot x \to_{11} e \cdot x$.
Adopt new rule

12. $y \cdot (I(y) \cdot x) \to x$

Superposition (again) of 3, 11: $(x \cdot y) \cdot I(x \cdot y) \to_{11} e$ and $(x \cdot y) \cdot I(x \cdot y) \to_3 x \cdot (y \cdot I(x \cdot y))$.
Adopt new rule

13. $x \cdot (y \cdot (y \cdot I(x \cdot y)) \to e$ canceled later

Superposition of 13, 4: $I(x) \cdot (x \cdot (y \cdot I(x \cdot y))) \to_4 y \cdot I(x \cdot y)$ and $I(x) \cdot (x \cdot (y \cdot I(x \cdot y))) \to_{13} I(x) \cdot e$.
Adopt new rule

14. $y \cdot I(x \cdot y) \to I(x)$ canceled later
cancel 13

Superposition of 4, 14: $I(y) \cdot (y \cdot I(x \cdot y)) \to_4 I(x \cdot y)$ and $I(y) \cdot (y \cdot I(x \cdot y)) \to_{14} I(y) \cdot I(x)$.
Adopt new rule

15. $I(x \cdot y) \to I(y) \cdot I(x)$ cancel 14

At this moment the TRS has only convergent critical pairs, e.g.:

$$\begin{array}{lll} I(y \cdot I(y)) & \to_{15} & I(I(y)) \cdot I(y) \\ & & \downarrow_9 \\ \downarrow_{11} & & y \cdot I(y) \\ & & \downarrow_{11} \\ I(e) & \to_{10} & e \end{array}$$

The significance of this fact is stated in the following lemma.

5.3.3. CRITICAL PAIR LEMMA (Knuth & Bendix [70], Huet [80]).
A TRS R *is WCR iff all critical pairs are convergent.*

So the TRS R_c with rewrite rules as in Table 5.4 is WCR.

1.	e·x	→	x
2.	I(x)·x	→	e
3.	(x·y)·z	→	x·(y·z)
4.	I(x)·(x·z)	→	z
8.	y·e	→	y
9.	I(I(y))	→	y
10.	I(e)	→	e
11.	y·I(y)	→	e
12.	y·(I(y)·x)	→	x
15.	I(x·y)	→	I(y)·I(x)

Tabel xx.

Furthermore, one can prove SN for $\mathcal{R}_c$ by the recursive path ordering explained in Section 2.2. (In fact we need the extended lexicographic version, due to the presence of the associativity rule.) According to Newman's Lemma $\mathcal{R}_c$ is therefore CR and hence complete. We conclude that the validity problem for the equational specification of groups is solvable.

The following theorem of Knuth and Bendix is an immediate corollary of the Critical Pair Lemma 5.3.3 and Newman's Lemma:

5.3.4. COROLLARY (Knuth & Bendix [70]). *Let* R *be a TRS which is SN. Then* R *is CR iff all critical pairs of* R *are convergent*. □

The completion procedure above by hand was naive, since we were not very systematic in searching for critical pairs, and especially since we were guided by an intuitive sense only of what direction to adopt when generating a new rule. In most cases there was no other possibility (e.g. at 4: z → I(x)·(x·z) is not a reduction rule due to the restriction that the LHS is not a single variable), but in case 15 the other direction was at least as plausible, as it is even length-decreasing. However, the other direction I(y)·I(x) → I(x·y) would have led to disastrous complications (described in Knuth & Bendix [70]).

The problem of what direction to choose is solved in the actual Knuth-Bendix algorithm and its variants by preordaining a 'reduction ordering' on the terms.

5.3.5. DEFINITION. A *reduction ordering* $>$ is a well-founded partial ordering among terms, which is closed under substitutions and contexts, i.e. if $s > t$ then $s^\sigma > t^\sigma$ for all substitutions σ, and if $s > t$ then $C[s] > C[t]$ for all contexts $C[\ ]$.

We now have immediately the following fact (noting that if $\mathcal{R}$ is SN, then $\rightarrow_{\mathcal{R}}^+$ satisfies the requirements of Definition 5.3.5):

5.3.6. PROPOSITION. *A TRS $\mathcal{R}$ is SN iff there is a reduction ordering $>$ such that $\alpha > \beta$ for every rewrite rule $\alpha \rightarrow \beta$ of $\mathcal{R}$.* □

Simple version of the Knuth-Bendix completion algorithm

Input: - an equational specification (Σ, E)
- a reduction ordering $>$ on $Ter(\Sigma)$ (i.e. a program which computes $>$)

Output: - a complete TRS R such that for all $s,t \in Ter(\Sigma)$: $s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

```
R := Ø;
while E ≠ Ø do
        choose an equation s = t ∈ E;
        reduce s and t to respective normal forms s' and t' with respect to R;
        if s' ≡ t' then
                E := E - {s = t}
        else
                if s' > t' then
                        α := s'; β := t'
                else if t' > s' then
                        α := t'; β := s'
                else
                        failure
                fi;
                CP := {P = Q | ⟨P, Q⟩ is a critical pair between the rules in R and α → β};
                R := R ∪ {α → β};
                E := E ∪ CP - {s = t}
        fi
od;
success
```

Figure 5.1

Simple version of the Knuth-Bendix completion algorithm

Input: - an equational specification (Σ, E)
- a reduction ordering $>$ on $Ter(\Sigma)$ (i.e. a program which computes $>$)

Output: - a complete TRS R such that for all $s,t \in Ter(\Sigma)$: $s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

```
R := Ø;
while E ≠ Ø do
        choose an equation s = t ∈ E;
        reduce s and t to respective normal forms s' and t' with respect to R;
        if s' ≡ t' then
                E := E - {s = t}
        else
                if s' > t' then
                        α := s'; β := t'
                else if t' > s' then
                        α := t'; β := s'
                else
                        failure
                fi;
                CP := {P = Q | ⟨P, Q⟩ is a critical pair between the rules in R and α → β};
                R := R ∪ {α → β};
                E := E ∪ CP - {s = t}
        fi
od;
success
```

Figuur xx.

In Figure 5.1 a simple version of the Knuth-Bendix completion algorithm is presented. As to the reduction ordering $>$ on Ter(S) which is an input to the algorithm: finding this is a matter of ingenuity, or experimentation. (Also without reduction ordering, computer systems for Knuth-Bendix completion equipped with an interactive question for orientation of equations into rewrite rules are of great help.)

The program of Figure 5.1 has three possibilities: it may (1) terminate successfully, (2) loop infinitely, or (3) fail because a pair of terms s, t cannot be oriented (i.e. neither $s > t$ nor $t > s$). The third case gives the most important restriction of the Knuth-Bendix algorithm: equational specifications with commutative operators cannot be completed.

If one still wants to deal with equational specifications having commutative/associative operators as in Exercise 5.4.3, one has to work modulo the equations of associativity and commutativity. For completion modulo such equations we refer to Peterson & Stickel [81] and Jouannaud & Kirchner [86].

In case (1) the resulting TRS is complete. To show this requires a non-trivial proof, see e.g. Huet [81]. In the next chapter we will give an abstract formulation of Knuth-Bendix completion, following Bachmair, Dershowitz & Hsiang [86], which streamlines

considerably this kind of correctness proofs.

The completion program of Figure 5.1 does not 'simplify' the rewrite rules themselves. Such an optimization can be performed after termination of the program, as follows.

5.3.7. DEFINITION. A TRS $\mathcal{R}$ is called *irreducible* if for every rewrite rule $\alpha \to \beta$ of $\mathcal{R}$ the following hold:

(i) β is a normal form with respect to $\mathcal{R}$,
(ii) α is a normal form with respect to $R - \{\alpha \to \beta\}$.

5.3.8. THEOREM (Métivier [83]). *Let $\mathcal{R}$ be a complete TRS. Then we can find an irreducible complete TRS $\mathcal{R}'$ such that the convertibilities $=_{\mathcal{R}}$ and $=_{\mathcal{R}'}$ coincide.*

Instead of optimizing the TRS which is the output of the above simple completion algorithm *after* the completion, it is more efficient to do this *during* the completion. Figure 5.2 contains a more efficient Knuth-Bendix completion algorithm, which upon successful termination yields irreducible TRSs as output.

We conclude this section with a theorem (5.3.10) stating that the Knuth-Bendix completion algorithm, given an equational specification and a reduction ordering, cannot generate two different complete irreducible TRSs. According to Dershowitz, Marcus & Tarlecki [88] the theorem is originally due to M. Ballantyne, but first proved in Métivier [83].

5.3.9. DEFINITION. Let $>$ be a reduction ordering. We call a TRS $\mathcal{R}$ *compatible* with $>$ if for every rewrite rule $\alpha \to \beta$ of $\mathcal{R}$ we have $\alpha > \beta$.

More efficient version of the Knuth-Bendix completion algorithm

Input: - an equational specification (Σ, E)
- a reduction ordering $>$ on $Ter(\Sigma)$ (i.e. a program which computes $>$)

Output: - a complete irreducible TRS R such that for all $s,t \in Ter(\Sigma)$: $s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

```
R := Ø;
while E ≠ Ø do
        choose an equation s = t ∈ E;
        reduce s and t to respective normal forms s' and t' with respect to R;
        if s' ≡ t' then
                E := E - {s = t}
        else
                if s' > t' then
                        α := s'; β := t'
                else if t' > s' then
                        α := t'; β := s'
                else
                        failure
                fi;
                R := {γ → δ' | γ→ δ ∈ R and δ' is a normal form of δ with respect to R ∪ {α → β} };
                CP := {P = Q | ⟨ P, Q⟩ is a critical pair between the rules in R and α → β};
                E := E ∪ CP ∪ {γ = δ | γ → δ ∈ R and γ is reducible by α → β} - {s = t};
                R := R ∪ {α → β} - {γ → δ | γ is reducible by α → β}
        fi
od;
success
```

Tabel xx.

5.3.10. THEOREM. (Métivier [83]) *Let* R_1 *and* R_2 *be two complete irreducible TRSs compatible with a given reduction ordering* $>$. *Suppose* R_1 *and* R_2 *define the same convertibility. Then* R_1 *and* R_2 *are equal (modulo a renaming of variables).* □

6.4. *Modulo rekenen.*

[1]	x+0 → x
[2]	x*0 → 0
[4]	x+s(y) → s(x+y)
[5]	s(s(x)) → x
[6]	x*s(y) → x+(x*y)
[7]	x+(x+(x*y)) → x*y
[8]	x+x → 0
[10]	s(x)+x → s(0)

Tabel xx.

http://cime.lri.fr/cime-1.15.html

modulo n rekenen, een compleet systeem met n+6 regels geeft, waarvan de eerste vier de + en * regels, dan een regel (die je schematisch kunt schrijven als) n+x → x, regels nx → en nx + x*y → x*y, en voor iedere 0 < i < n, een regel (n-1)(i+x) + x → n-i.

[1]	x+0 → x
[2]	x*0 → 0
[4]	x+s(y) → s(x+y)
[5]	s(s(s(x))) → x
[6]	x*s(y) → x+(x*y)
[7]	x+(x+(x+(x*y))) → x*y
[8]	x+(x+x) → 0
[11]	s(x)+(s(x)+x) → s(s(0))
[13]	s(s(x))+(s(s(x))+x) → s(0)

Tabel xx.

[1]	x+0 → x
[2]	x*0 → 0
[3]	x+s(y) → s(x+y)
[5]	s(s(s(s(x)))) → x
[6]	x*s(y) → x+(x*y)
[7]	x+(x+(x+(x+(x*y)))) → x*y
[8]	x+(x+(x+x)) → 0
[11]	s(x)+(s(x)+(s(x)+x)) → s(s(s(0)))
[13]	s(s(x))+(s(s(x))+(s(s(x))+x)) → s(s(0))
[16]	s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+x)) → s(0)

Tabel xx.

[1]	x+0 → x
[2]	x*0 → 0
[3]	x+s(y) → s(x+y)
[4]	x*s(y) → x+(x*y)
[6]	s(s(s(s(s(x))))) → x
[7]	x+(x+(x+(x+(x+(x*y))))) → x*y
[8]	x+(x+(x+(x+x))) → 0
[12]	s(x)+(s(x)+(s(x)+(s(x)+x))) → s(s(s(s(0))))
[15]	s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+x))) → s(s(s(0)))
[17]	s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+x))) → s(s(0))
[19]	s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+x))) → s(0)

Tabel xx.

x+0 → x
x*0 → 0
x+s(y) → s(x+y)
x*s(y) → x+(x*y)
s(s(s(s(s(s(x)))))) → x
x+(x+(x+(x+(x+(x+(x*y)))))) → x*y
x+(x+(x+(x+(x+x)))) → 0
s(x)+(s(x)+(s(x)+(s(x)+(s(x)+(s(x)+x))))) → s(s(s(s(s(0)))))
s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+x))))) → s(s(s(s(s(0)))))
s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+x))))) → s(s(s(s(0))))
s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+x))))) → s(s(s(0)))
s(s(s(s(s(x)))))+(s(s(s(s(s(x)))))+(s(s(s(s(s(x)))))+(s(s(s(s(s(x))))) +(s(s(s(s(s(x)))))+(s(s(s(s(s(x)))))+x))))) → s(s(0))
s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+x))))) → s(0)

Figuur xx.

5.4. Exercises.

5.4.1. EXERCISE. **Equational deduction systems.**
Often the inference system in Table 5.1 is presented slightly different, as follows. Prove the equivalence of the two versions below with the system above.

Axioms (in addition to the equations in E):

$t = t$ *reflexivity*

Rules:

$$\frac{t_1 = t_2}{t_2 = t_1} \quad \textit{symmetry}$$

$$\frac{t_1 = t_2 \quad t_2 = t_3}{t_1 = t_3} \quad \textit{transitivity}$$

$$\frac{t_1 = t_2}{t_1[x:=t] = t_2[x:=t]} \quad \textit{substitution (1)}$$

$$\frac{t_1 = t_2}{t[x:=t_1] = t[x:=t_2]} \quad \textit{substitution (2)}$$

Here [x:=t] denotes substitution of t for all occurrences of x. (The assignment notation is chosen to avoid the usual confusion between [x/t], [t/x], [x\t], [t\x].) An equivalent formulation is to combine the two substitution rules in one:

$$\frac{t_1 = t_2 \quad t = t'}{t_1[x:=t] = t_2[x:=t']} \quad \textit{substitution}$$

5.4.2. EXERCISE. (Puzzle.) (*Ternary Boolean Algebras*, from: Wos, Overbeek, Lusk & Boyle [84], p.263.)
Let $E = \{A_1, A_2, A_3\}$, with

A_1: $F(F(v, w, x), y, F(v, w, z)) = F(v, w, F(x, y, z))$
A_2: $F(y, x, x) = x$
A_3: $F(x, y, G(y)) = x$.

Prove: $E \vdash F(x, x, y) = x$. (A solution can be found before the References.)

5.4.3. EXERCISE. Let (S, E) be the specification given by the equations

$x + 0 = x$
$x + S(y) = S(x + y)$
$x + y = y + x$

Prove that there is no complete TRS R 'for' E, i.e. such that for all terms $s, t \in Ter(\Sigma)$: $s =_R t \Leftrightarrow s =_E t$. (Consider in a supposed complete TRS R, the normal forms of the open terms $x + y$ and $y + x$.)

5.4.4. EXERCISE. Consider the specification as in the previous exercise and find a TRS (S, R) such that $(S, R)_0$ (i.e. the restriction of (S, R) to ground terms) is complete.

5.4.5. EXERCISE (Bergstra & Klop). Prove the following fact:

THEOREM. *Let* (S, E) *be the specification with* $S = \{0, +\}$ *and* $E = \{x + y = y + x\}$. *Then there is no finite TRS* R *such that the restriction to ground terms,* $(R)_0$, *is complete and such that* $=_R$ *and* $=_E$ *coincide on ground terms*.

Prove that there is no complete TRS R 'for' E, i.e. such that for all terms s, t ∈ Ter(Σ): $s =_R t \Leftrightarrow s =_E t$. (Consider in a supposed complete TRS R, the normal forms of the open terms x + y and y + x.)

5.4.4. EXERCISE. Consider the specification as in the previous exercise and find a TRS (Σ, R) such that $(\Sigma, R)_0$ (i.e. the restriction of (Σ, R) to ground terms) is complete.

5.4.5. EXERCISE (Bergstra & Klop). Prove the following fact:

THEOREM. *Let* (Σ, E) *be the specification with* Σ = {0, +} *and* E = {x + y = y + x}. *Then there is no finite TRS* R *such that the restriction to ground terms,* $(R)_0$, *is complete and such that* $=_R$ *and* $=_E$ *coincide on ground terms.*

PROOF SKETCH. Define terms $t_0 \equiv 0$, $t_{n+1} \equiv t_n + t_n$ $(n \geq 0)$. Suppose R is a TRS with finitely many rewrite rules such that $=_R$ and $=_E$ coincide on ground terms. Let N be the maximum of the depths of the LHSs of the rewrite rules in R. (Here 'depth' refers to the height of the corresponding term formation tree.)

Figure 5.3

Consider the terms $t^* \equiv t_N + t_{2N}$ and $t^{**} \equiv t_{2N} + t_N$. Clearly, $t^* =_E t^{**}$. In fact, {t*, t**} is an E-equivalence class, hence also an R-convertibility class. Therefore there must be a rewrite rule r such that t* is an r-redex or t** is an r-redex (since there are only two elements in the convertibility class) and such that $t^* \text{Æ}_r t^{**}$. Say t* is an r-redex. Now one can easily show that $t^* \text{Æ}_r t^{**} \text{Æ}_r t^*$. Hence R is not even SN on ground terms. ø

5.4.6. EXERCISE. Prove the Critical Pair Lemma.
(The proof is not hard, after distinguishing cases as in Figure 5.3. Some care has to be taken to deal with repeated variables in left-hand sides of reduction rules.)

5.4.7. EXERCISE. Prove, using the Critical Pair Lemma: If the TRS R has finitely many rules and is SN, then WCR and CR are decidable.

5.4.8. EXERCISE. Prove that every irreducible ground TRS is complete.
(Hint: use Exercise 2.7.2 to show SN and Corollary 5.3.4 to show CR.)

5.4.9. EXERCISE. A proof of Theorem 5.3.8 can be given along the following line. Let R_1 be the TRS
{α → β' | α → β ∈ R and β' is the normal form of β with respect to R}. We may assume that R_1 does not contain rewrite rules that are a renaming of another rewrite rule.

Further, define R' = {α → β ∈ R_1 | α is a normal form with respect to R_1 - {α → β} }. Now the proof that $s =_R t \Leftrightarrow s =_{R'} t$ follows from the (easy) proofs of the sequence of statements:

(1) if $s \to_{R_1} t$ then $s \to_R^+ t$;
(2) R and R_1 define the same set of normal forms;
(3) R_1 is SN;
(4) if $s \ddot{y}_R t$ and t is a normal form then $s \ddot{y}_{R_1} t$;
(5) $s =_R t \Leftrightarrow s =_{R_1} t$;
(6) R_1 is CR;
(7) if $s \to_{R'} t$ then $s \to_{R_1} t$;
(8) R_1 and R' define the same set of normal forms;
(9) R' is SN;

(10) if $s \ddot{y}_{R1} t$ and t is a normal form then $s \ddot{y}_{R'} t$;
(11) $s =_{R1} t \Leftrightarrow s =_{R'} t$;
(12) R' is CR;
(13) R' is irreducible.

5.4.9. EXERCISE. A proof of Theorem 5.3.8 can be given along the following line. Let R_1 be the TRS
$\{\alpha \to \beta' \mid \alpha \to \beta \in R$ and β' is the normal form of β with respect to $R\}$. We may assume that R_1 does not contain rewrite rules that are a renaming of another rewrite rule.

Further, define $R' = \{\alpha \to \beta \in R_1 \mid \alpha$ is a normal form with respect to $R_1 - \{\alpha \to \beta\}\}$. Now the proof that $s =_R t \Leftrightarrow s =_{R'} t$ follows from the (easy) proofs of the sequence of statements:

(1) if $s \to_{R1} t$ then $s \to_R^+ t$;
(2) R and R_1 define the same set of normal forms;
(3) R_1 is SN;
(4) if $s \ddot{y}_R t$ and t is a normal form then $s \ddot{y}_{R1} t$;
(5) $s =_R t \Leftrightarrow s =_{R1} t$;
(6) R_1 is CR;
(7) if $s \to_{R'} t$ then $s \to_{R1} t$;
(8) R_1 and R' define the same set of normal forms;
(9) R' is SN;
(10) if $s \ddot{y}_{R1} t$ and t is a normal form then $s \ddot{y}_{R'} t$;
(11) $s =_{R1} t \Leftrightarrow s =_{R'} t$;
(12) R' is CR;
(13) R' is irreducible.

5.4.10. EXERCISE (Huet [80]). **Properties of critical pairs as criteria for confluence.**
In this exercise we collect some criteria for confluence in terms of properties of critical pairs, as well as some counterexamples, from Huet [80]. Also some questions are listed which are, as far as we know, open. See Table 5.5.

1	$\ddot{Y}$ t, s ±	$t \downarrow s$	WCR	¬CR
2	$\ddot{Y}$ t, s ±	$t \downarrow s$ & SN	WCR	CR
3	$\ddot{Y}$ t, s ±	LL & RL & strongly closed	strongly confluent	CR
4	$\ddot{Y}$ t, s ±	LL & strongly closed	WCR	¬CR
5	$\ddot{Y}$ t, s ±	LL & $t \to_{\|\|} s$	$\to_{\|\|}$ is $WCR^{\leq 1}$	CR
6	$\ddot{Y}$ t, s ±	LL & $s \to_{\|\|} t$	WCR	**CR?**
7	$\ddot{Y}$ t, s ±	LL & $s \to^{\equiv} t$	WCR	**CR?**
8	$\ddot{Y}$ t, s ±	LL & $t \to^{\equiv} s$ or $s \to^{\equiv} t$	WCR	**CR?**

Table 5.5

(1) In row 1 of the table the Critical Pair Lemma 5.3.3 is stated: if every critical pair Ÿt, s± is convergent (notation: t Ø s), then WCR holds. However, CR need not to hold; a counterexample is given by the TRS with four constants a, b, c, d and rules as in Figure 1.3.
(2) Row 2 of the table is Theorem 5.3.4 of Knuth and Bendix.
(3) In row 3, LL means that the TRS is left-linear, RL right-linear (i.e. no right-hand side of a reduction rule contains repetitions of a variable). Strongly confluent is defined in Exercise 1.7.10. We furthermore define:

DEFINITION. A TRS is *strongly closed* if for every every critical pair Ÿt, s± there are t', t" such that t ÿ t' ←≡ s and s ÿ t" ←≡ t.

(1) In row 1 of the table the Critical Pair Lemma 5.3.3 is stated: if every critical pair Ÿt, s± is convergent (notation: t ↓ s), then WCR holds. However, CR need not to hold; a counterexample is given by the TRS with four constants a, b, c, d and rules as in Figure 1.3.
(2) Row 2 of the table is Theorem 5.3.4 of Knuth and Bendix.
(3) In row 3, LL means that the TRS is left-linear, RL right-linear (i.e. no right-hand side of a reduction rule contains repetitions of a variable). Strongly confluent is defined in Exercise 1.7.10. We furthermore define:

DEFINITION. A TRS is *strongly closed* if for every every critical pair Ÿt, s± there are t', t" such that t ÿ t' ←≡ s and s ÿ t" ←≡ t.

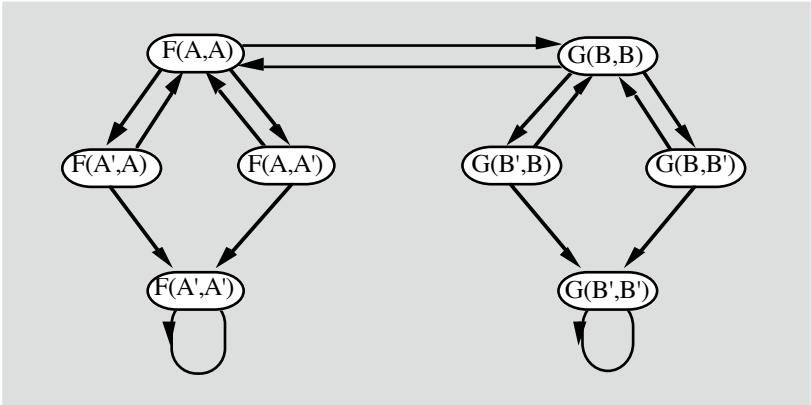


Figure 5.4

Prove that 'strongly closed' is not sufficient to guarantee CR, by considering the non-left-linear TRS
{F(x, x) → A, F(x, G(x)) → B, C → G(C)}. However, if the TRS is left-linear, right-linear and strongly closed, then CR holds (for a proof see Huet [80]); in fact, we then have strong confluence.

(4) In 3, RL cannot be dropped. A nice counterexample is in Huet [80], given by J.-J. Lévy: it contains the following eight left-linear rules. See also Figure 5.4.

F(A, A) → G(B, B)	G(B, B) → F(A, A)
A → A'	B →B'
F(A', x) → F(x, x)	G(B', x) → G(x, x)
F(x, A') → F(x, x)	G(x, B') → G(x, x)

Check that CR does not hold, and that the TRS is strongly closed.

(5) This is a remarkable fact: if the TRS is left-linear, and for every critical pair Ÿt, s± we have t →| | s, then WCR$^{\leq 1}$ holds, and hence CR. Here →| | (parallel reduction) denotes a sequence of redex contractions at disjoint occurrences.

(6, 7, 8) If in 5 we replace t →| | s by s →| | t, then the CR question is open. Likewise (7) if t →| | s is replaced by

s →∫ t, or (8) replaced by: " t →∫ s or s →∫ t ".

5.4.11. EXERCISE. Knuth & Bendix [70] contains completions of two specifications which closely resemble the specification of groups (see Table 5.6), called 'L-R theory' and 'R-L theory'.

Prove, using the completions, that x·e = x is not derivable in L-R theory and that in R-L theory the equations e·x = x and x·I(x) = e are not derivable. Furthermore, in L-R theory the equation x·e = x is not derivable. Hence the three theories are different, i.e. determine different varieties of algebras.

In fact, note that the variety of groups is the intersection of both the variety of L-R algebras and that of R-L algebras, and that the latter two varieties are incomparable with respect to set inclusion.

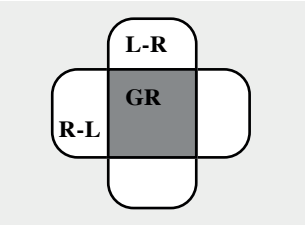


Figure 5.5

group theory	*L-R theory:*	*R-L theory:*
e·x = x	e·x = x	x·e = x
I(x)·x = e	x·I(x) = e	I(x)·x = e
(x·y)·z = x·(y·z)	(x·y)·z = x·(y·z)	(x·y)·z = x·(y·z)
completion:	*completion:*	*completion:*
e·x → x	e·x → x	
x·e → x		x·e → x
I(x)·x → e		I(x)·x → e
x·I(x) → e	x·I(x) → e	
(x·y)·z → x·(y·z)	(x·y)·z → x·(y·z)	(x·y)·z → x·(y·z)
I(e) → e	I(e) → e	I(e) → e
I(x·y) → I(y)·I(x)	I(x·y) → I(y)·I(x)	I(x·y) → I(y)·I(x)
x·(I(x)·y) → y	x·(I(x)·y) → y	
		e·x → I(I(x))
I(x)·(x·y) → y	I(x)·(x·y) → y	
		x·I(I(y) → x·y
I(I(x)) → x		
	x·e → I(I(x))	
	I(I(I(x))) → I(x)	I(I(I(x))) → I(x)
		x·(y·I(y)) → x
	I(I(x))·y → x·y	
		x·(I(I(y))·z) → x·(y·z)
		x·(y·(I(y)·z)) → x·z
		I(x)·(x·y) → I(I(y))

Table 5.6

6.10. Exercises.

6.10.1. EXERCISE. To illustrate the concept of proof orderings we will give an alternative proof of Newman's Lemma 1.6(ii) using this notion. ('Alternative' with respect to the proofs that we have seen in the literature.) See also Exercise 4.12.5 for our multiset notations.

Let R be a TRS which is SN and WCR. Let $P \equiv (s_0,...,s_n)$ be a proof of the conversion $s_0 = s_n$. We define the *complexity* $|P|$ of the proof P as the multiset $\{s_0,...,s_n\}$. The ordering $\sqsupset$ which we

will use is induced by the multiset extension of $\rightarrow_R^+$, notation: $(\rightarrow_R^+)^\mu$. So

$$P \sqsupset P' \text{ iff } |P| \ (\rightarrow_R^+)^\mu \ |P'|.$$

(This means that $P \sqsupset P'$ if the multiset $|P'|$ arises from the multiset $|P|$ by repeatedly replacing an element of the multiset by arbitrarily many elements which are less in the sense of the well-founded ordering $\rightarrow_R^+$. I.e. by repeatedly replacing a term t in the multiset of terms by a number (≥ 0) of proper reducts of t.)

(i) Prove that $\sqsupset$ is a proof ordering.

(ii) If $P \equiv (s_0,...,s_n)$ is not a rewrite proof, then there is a proof P' of the equation $s_0 = s_n$ such that $P \sqsupset P'$. (Hint: consider a 'peak' in the conversion P, and replace it by a 'valley', using WCR. See Figure 6.4.)

(iii) Conclude that R is CR.

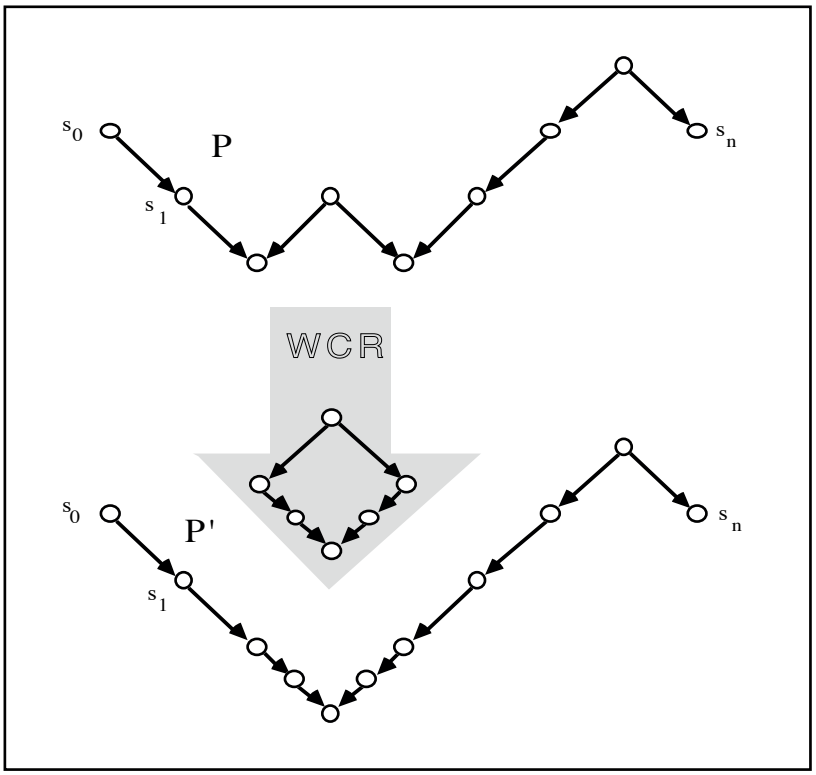


Figure 6.4

Opgave 3. Gegeven is de TRS R met regels

$a(a(x)) \rightarrow b(x)$
$b(b(x)) \rightarrow x$
$b(a(x)) \rightarrow a(b(x))$

(i) Is R een OTRS?

(ii) Wat zijn de normaalvormen van R?

(iii) Bewijs dat R de eigenschap WCR heeft.

(iv) Bewijs dat R de eigenschap SN heeft.

Opgave 5. Bewijs dat de TRS met de regels

$g(x) \rightarrow f(f(x))$
$h(f(x)) \rightarrow g(g(x))$

Opgave 8. Completeer de TRS met de drie regels

$a(b(c(x))) \rightarrow x$
$a(b(x)) \rightarrow x$
$b(c(x)) \rightarrow x$

21. Bewijs dat de TRS R met herschrijfregels

$F(x) \rightarrow A$
$F(x) \rightarrow G(F(x))$
$G(F(x)) \rightarrow F(H(x))$

WCR is.

3.Completeer de volgende equationele specificaties:

1.$F(F(F(x))) = x$
$F(F(F(F(F(x))))) = x$

2.(x.y).(y.z) =y (met oplossing)

28. Ga na of de volgende TRS-en SN zijn, en geef tegenvoorbeeld of bewijs.

1.g(x) → f(x), f(x)→ x

2.g(f(x)) → f(f(g(x)))

3.g(f(x)) → f(f(g(g(x))))

4.f(f(x)) → f(g(f(x)))

29. Gegeven is de "AMS0"-TRS.

i.Geldt A(t, s) = A(s, t) voor alle termen s, t?

ii.Bewijs dat voor gesloten termen s, t geldt A(t, s) = A(s, t).

30. Gegeven de reductieregel L(L(L(x))) → H(x).
Hoe kan deze regel met zichzelf overlappen?
Welke kritieke paren worden daardoor gegenereerd?

31. Zelfde vragen voor het paar regels

S(P(x)) → x

P(S(x)) → x

32. Zelfde vragen voor het trio regels

A(x, 0) → x

A(x, y) → A(y, x)

A(A(x,y),z) → A(x, A(y,z))

56. Gegeven is de TRS met regels

or(true, true) → true

or(x, y) → or (y, x)

1.Is dit een orthogonaleTRS?

2.Is de TRS CR? Geef bewijs of tegenvoorbeeld.

57. Beschouw de AMS0 TRS in combinatie met

$\min(\min(x)) \rightarrow x$
$A(x, \min(x)) \rightarrow 0$
$A(\min(x), x) \rightarrow 0$

i. Welke kritieke paren zijn er?
ii. Bewijs of weerleg WCR.

85. Is de volgende TRS SN?

$F(x, G(y)) \rightarrow H(x, x)$
$F(G(x), F(y,y)) \rightarrow y$

86. Is de volgende TRS SN?

$-(x+y) \rightarrow (-(-x)+y)+y$

87. Bewijs SN:

$G(x, y) \rightarrow H(x, y)$
$H(F(x), y) \rightarrow F(G(x, y))$

6.5. *Het cola-gen.*

Virus problem:

The invariant to look at is the parity of the sum of the amounts of As and Ts. For example, the cola gene counts 2 As and 3 Ts for a total of 5, which is odd.

It is easy to see that the second and third rewriting rules do not modify the number of As and Ts, while the other three modify each by one, thus modifying the sum by two.

The milk gene counts 3 As and 4 Ts, totalling 7, which is odd. The virus gene counts 2 As and 4 Ts, totalling 6, which is even. Thus neither can be converted into the other.

(In fact, there is an even better invariant: the difference between the

number of As and Ts, as an integer rather than just the parity. The milk and cola genes have T-A=1, while the virus gene has T-A=2.)

Cola problem:

Lowercase letters are the ones being substituted.

TAGctAGCTAGCT
TagtaTAGCTAGCT
tatAGCTAGCT
CTagCTAGCT
CTGAGctAGCT
CTGagtaTAGCT
CTGATAGct
CTGATagtaT
CTGAtAT
CTGAtcATAT
CTGACtcatAT
CTGACTaT
CTGACTagTAT
CTGACTGagtaT
CTGACTGAt
CTGACTGAtcAT
CTGACTGACtcat
CTGACTGACT

Beschouw een signatuur met constante 0 en binaire functiesymbolen . and |. Beschouw de TRS met reductieregels

1.	x.0 -> x
2.	0.x -> x
3.	x | 0 -> x
4.	0 | x -> 0
5.	x | x -> 0
6.	(x.y) | z -> (x | z).(y | (z | x))
7.	z | (x.y) -> (z | x) | y

Tabel xx.

Als feit is gegeven dat deze TRS de eigenschap SN heeft. Bewijs met de stelling van Knuth-Bendix dat hij ook CR is.

5.1. Equational specifications: syntax and semantics.

We can be short about introducing the syntax of equational specifications: an equational specification is just a TRS "without orientation". More precisely, an equational specification is a pair (Σ, E) where the signature (or alphabet) Σ is as in Section 2.1 for TRSs (Σ, R), and where E is a set of equations $s = t$ between terms $s, t \in Ter(\Sigma)$. *)

If an equation $s = t$ is *derivable* from the equations in E, we write $(\Sigma, E) \vdash s = t$ or $s =_E t$. Formally, derivability is defined by means of the inference system of Table 5.1.

$(\Sigma, E) \vdash s = t$	if $s = t \in E$
$\dfrac{(\Sigma, E) \vdash s = t}{(\Sigma, E) \vdash s^\sigma = t^\sigma}$	for every substitution σ
$\dfrac{(\Sigma, E) \P s_1 = t_1 \ ,..., \ (\Sigma, E) \P s_n = t_n}{(\Sigma, E) \P F(s_1,\ldots,s_n) = F(t_1,\ldots,t_n)}$	for every n-ary $F \in \Sigma$
$(\Sigma, E) \P t = t$	
$\dfrac{(\Sigma, E) \P t_1 = t_2 \ , \ (\Sigma, E) \P t_2 = t_3}{(\Sigma, E) \P t_1 = t_3}$	
$\dfrac{(\Sigma, E) \P s = t}{(\Sigma, E) \P t = s}$	

Table 5.1

If Σ is a signature, a Σ-*algebra* A is a set A together with functions $F^{\mathsf{A}}: A^n \to A$ for every n-ary function symbol $F \in \Sigma$. (If F is 0-ary, i.e. F is a constant, then $F^{\mathsf{A}} \in A$.) An equation $s = t$ ($s, t \in Ter(\Sigma)$) is assigned a meaning in A by interpreting the function symbols in s,t via the corresponding functions in A. Variables in $s = t$ are (implicitly) universally quantified. If the universally quantified statement corresponding to $s = t$ ($s, t \in Ter(\Sigma)$) is true in A, we write
$\mathsf{A} \bullet s = t$ and say that $s = t$ is *valid* in A. A is called a *model* of a set of equations E if every equation in E is valid in A. Abbreviation: $\mathsf{A} \bullet E$. The *variety* of Σ-algebras defined by an equational specification (Σ, E), notation $Alg(\Sigma, E)$, is the class of all Σ-algebras A such that
$\mathsf{A} \bullet E$. Instead of $\hat{a}\mathsf{A} \in Alg(\Sigma, E)$ $\mathsf{A} \bullet F$, where F is a set of equations between Σ-terms, we will write $(\Sigma, E) \bullet F$. There is the well-known completeness theorem for equational logic of Birkhoff [35]:

5.1.1. THEOREM. *Let* (Σ, E) *be an equational specification. Then for all* $s, t \in Ter(\Sigma)$:

$$(\Sigma, E) \P s = t \quad \Leftrightarrow \quad (\Sigma, E) \bullet s = t. \qquad \text{ø}$$

Now the *validity problem* or *uniform word problem* for (Σ, E) is:

Given an equation $s = t$ *between* Σ*-terms, decide whether or not* $(\Sigma, E) \bullet s = t$.

According to Birkhoff's completeness theorem for equational logic this amounts to deciding $(\Sigma, E) \P s = t$. Now we can state why *complete* TRSs (i.e. TRSs which are SN and CR) are important. Suppose for the equational specification (Σ, E) we can find a complete TRS (Σ, R) such that for all terms $s, t \in Ter(\Sigma)$:

$$t =_R s \quad \Leftrightarrow \quad E \P t = s \qquad\qquad (e)$$

Then (provided R has finitely many rewrite rules only *)) we have a positive solution of the validity problem. The decision algorithm is simple:

(1) *Reduce* s *and* t *to their respective normal forms* s', t'
(2) *Compare* s' *and* t': $s =_R t$ *iff* $s' \equiv t'$.

We are now faced with the question how to find a complete TRS R for a given set of equations E such that (e) holds. In general this is not possible, since not every E (even if finite) has a solvable validity problem. The most famous example of such an E with unsolvable validity problem is the set of equations obtained from CL, Combinatory Logic, in Tables 2.3, 2.4 above after replacing '$\rightarrow$' by '=': see Table 5.2. (For a proof of the unsolvability see Barendregt [84].) So the validity

problem of (Σ, E) can be solved by providing a complete TRS (Σ, R) for (Σ, E). Note however, that there are equational specifications (Σ, E) with decidable validity problem but without a complete TRS (Σ, R) satisfying (e): see Exercise 5.4.3.

$Sxyz = xz(yz)$
$Kxy = x$
$Ix = x$

Table 5.2

It is important to realize that we have considered up to now equations s = t between possibly *open* Σ-terms (i.e. possibly containing variables). If we restrict attention to equations s = t between *ground* terms s, t, we are considering the *word problem* for (Σ, E), which is the following decidability problem:

> *Given an equation* s = t *between ground terms* s, t ∈ Ter(Σ)*, decide whether or not* (Σ, E) • s = t *(or equivalently,* (Σ, E) ¶ s = t *).*

Also for the word problem, complete TRSs provide a positive solution. In fact, we require less than completeness (SN and CR) for all terms, but only for ground terms. (See Example 2.1.3 for an example where this makes a difference.) It may be (as in Exercise 5.4.4) that a complete TRS for E cannot be found with respect to all terms, while there does exist a TRS which is complete for the restriction to ground terms.

5.1.2. REMARK. Note that there are finite equational specifications (Σ, E) which have a decidable word problem (so for ground terms) for which no complete TRS R (complete with respect to ground terms) exists. This strengthens the observation in Exercise 5.4.3. The simplest such (Σ, E) is the specification consisting of a single binary commutative operator + and a constant 0, and equations E = {x + y = y + x}. According to Exercise 5.4.3 (which also works for the present simpler specification) no complete TRS R can be found such that for all (open) s, t we have $s =_R t \Leftrightarrow s =_E t$. According to Exercise 5.4.5, we also have the stronger result that no TRS R exists which is complete for ground terms and such that for ground terms s, t we have $s =_R t \Leftrightarrow s =_E t$.

5.2. Term rewriting and initial algebra semantics.

We will now make more explicit the connection between term rewriting and initial algebra semantics. We suppose familiarity with the concept of an initial algebra in the class of models of an equational specification (Σ, E), i.e. the variety Alg(Σ, E), as defined by universal properties in terms of homomorphisms. (See e.g. Meinke & Tucker [90], Goguen & Meseguer [85].) Although the initial algebra is only determined up to isomorphism, we will speak of 'the' initial algebra and use the notation I(Σ, E) for it. It is well-known that I(Σ, E) can be obtained from the set of ground terms $Ter_0(\Sigma)$ by dividing out the congruence relation $=_E$. Thus we can equate the initial algebra I(Σ, E) with the quotient algebra $Ter_0(\Sigma)/=_E$.

Now suppose that (Σ, R) is a TRS 'for' (Σ, E), that is, $=_R$ coincides with $=_E$. (So the initial

algebra of (Σ, E) can also be written as $Ter_0(\Sigma)/=_R$.) If R is a *complete* TRS, then $I(\Sigma, E)$ is in fact a *computable* algebra. This is merely a rephrasing of: the word problem (for ground terms) for (Σ, E) is solvable. As noted in Exercise 5.4.5, the reverse is not necessarily the case; for some (Σ, E) with computable initial algebra there does not exist a complete TRS—at least not *in the same signature*. However, a remarkable theorem of Bergstra and Tucker states that if we allow an extension of the signature, with some functions and constants (no new sorts), then a complete TRS can always be found. More precisely:

5.2.1. DEFINITION. (i) The algebra $\mathsf{A} \in Alg(\Sigma, E)$ is *minimal*, if it is (isomorphic to) a quotient algebra $Ter(\Sigma)/\equiv$ for some congruence $\equiv$. In particular, $I(\Sigma, E)$ is a minimal algebra. In other words, an algebra is minimal if its elements are generated by functions and constants in the signature.
(ii) A minimal algebra A is *computable*, if its equality is decidable, i.e. if the relation $\mathsf{A} \bullet t = s$ for ground terms $t, s \in Ter(\Sigma)$ is decidable.

5.2.2. THEOREM (Bergstra & Tucker [80]).
Let A *be a minimal* Σ*-algebra,* Σ *a finite signature. Then the following are equivalent:*
(i) A *is a computable algebra;*
(ii) *there is an extension of* Σ *to a finite* Σ'*, obtained by adding some function and constant symbols, and there is a complete TRS* (Σ', R) *such that*

$$\mathsf{A} \cong I(\Sigma', R^=)|_\Sigma.$$

Here $R^=$ is the equational specification obtained by viewing the reduction rules in R as equations, and $|_\Sigma$ is the restriction to the signature Σ. So A is a '*reduct*' (see Meinke & Tucker [90]) of an initial algebra given by a complete TRS. (The TRS R as in the theorem is not only ground complete, but complete with respect to all terms. Actually, it is an orthogonal TRS as defined in the next chapter; and for orthogonal TRSs possessing at least one ground term, ground completeness implies completeness.) The functions (including the constants as 0-ary functions) to be added to Σ are sometimes referred to as '*hidden functions*'. Note that according to the statement in the theorem no new sorts are needed, thus the present theorem has also a bearing on the homogeneous (i.e. one-sorted) case that we are considering in this paper.

For more information concerning the connection between term rewriting and computability aspects of initial algebra semantics (and 'final' algebra semantics), also for the heterogeneous (many-sorted) case, we refer to the very complete survey Goguen & Meseguer [85].

8

COMPLETERING

INHOUD.

6.0. *Introductie.*

6.3. *Successor en predecessor.*

Consider the following equational specification (or theory) E of the integers (**Z**) with 0, +, successor S and predecessor P (left column, (a)):

(a)	$0 + x = x$	(b)	(1)	$0 + x \rightarrow x$
	$x + 0 = x$		(2)	$x + 0 \rightarrow x$
	$S(x) + y = S(x + y)$		(3)	$S(x) + y \rightarrow S(x + y)$
	$x + S(y) = S(x + y)$		(4)	$x + S(y) \rightarrow S(x + y)$
	$x + P(y) = P(x + y)$		(5)	$x + P(y) \rightarrow P(x + y)$
	$P(S(x)) = x$		(6)	$P(S(x)) \rightarrow x$

Tabel xx

(Since this specification is intended to be symmetrical with respect to permuting S and P one might expect also the equations $S(P(x) = x$ and $P(x) + y = P(x + y)$ to be included in E. Actually these equations are derivable.)

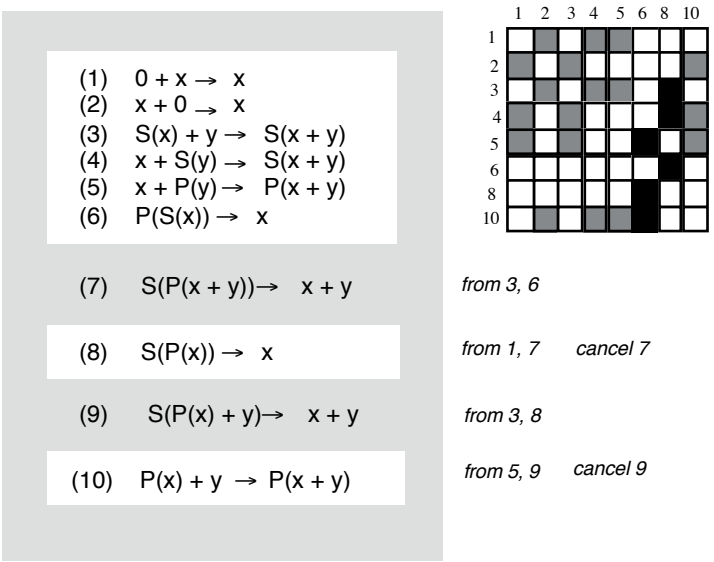


Figure 6.1.

We will now perform an 'intuition guided' completion of E. First let us adopt as rewrite rules all equations from E, oriented from left to right. This yields rules (1 - 6) as above in column (b). This is not yet a complete TRS; though it is terminating (as will be seen later), it is not confluent. The reason is that there is an overlap between the left-hand sides of (3), (5) that is harmful: S(x) + P(y) reduces with (3) to S(x + P(y)) and with (5) to P(S(x) + y). These two terms form what is called a *critical pair*. The terms can be reduced further: S(x + P(y)) → S(P(x + y)) and P(S(x) + y) → P(S(x + y)) → x + y, but then we are stuck since S(P(x + y)) and x + y are normal forms with respect to (1 - 6). So confluence fails.

Actually, this is not the only critical pair generated by (1 - 6); there are also overlaps between (1), (2), between (1), (4), between (1), (5), between (2), (3), between (3), (4), between (5), (6). But these critical pairs are harmless. For instance, (5), (6) yield the critical pair P(x + S(y)), x + y. These terms have a common reduct:

$$P(x + S(y)) \rightarrow P(S(x + y)) \rightarrow x + y.$$

We try to solve the problem of non-confluence posed by the terms S(P(x + y)) and x + y in a drastic way: we simply add as a new rule

(7) S(P(x + y)) → x + y.

Now the critical pair given by (3), (5) is harmless too. However, new critical pairs arise:

overlap between (1), (7) yields S(P(0 + y)) with as reducts 0 + y, S(P(y)). This causes us to adopt a new rule:

(8) $S(P(y)) \to y$.

We can cancel (7) now, as it is a consequence of (8). We consider the possible overlaps: the overlap between (8), (6) is harmless; likewise between (8), (4). But not the one between (8), (3), which causes the introduction of rule

(9): $S(P(x) + y) \to x + y$. In this way we continue, and luckily after a few more steps as in Figure 7 we reach a successful conclusion: a TRS $\mathcal{R}$ where all critical pairs are harmless. Moreover, $\mathcal{R}$ is terminating; this can be seen by noting that all our rules were chosen such that they respected the recursive path ordering (to be explained in the next section) obtained by putting $+ > S$ and $+ > P$.

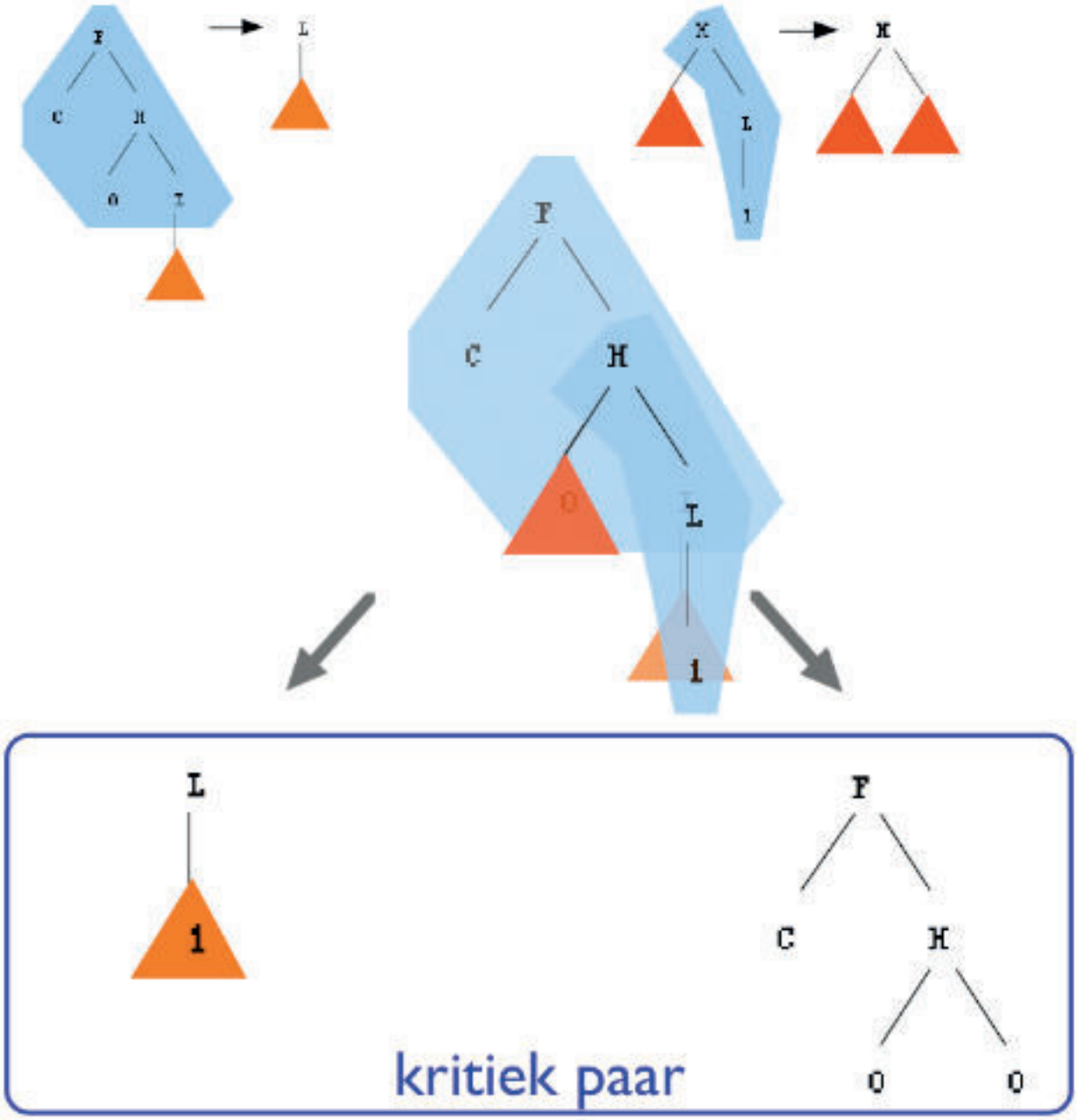
Let us give a precise definition of critical pairs:

6.3.1. DEFINITION. Let the TRS $\mathcal{R}$ contain the rewrite rules r: $t \to s$ and r': $t' \to s'$. Suppose r, r' are 'standardized apart', i.e. renamed such that they have no variables in common. Suppose furthermore that $t \equiv C[u]$, u not a variable, and that u and t' can be unified with most general unifier σ. Then $t^\sigma \equiv C^\sigma[u^\sigma] \equiv C^\sigma[t'^\sigma]$ is subject to an r'-reduction as well as an r-reduction, with result: $C^\sigma[s'^\sigma]$ respectively s^σ. Now $\langle C^\sigma[s'^\sigma], s^\sigma \rangle$ is called a *critical pair* of $\mathcal{R}$.

If r, r' are (renamed versions of) the same rewrite rule, we moreover require that the context C[ ] is not the trivial context.

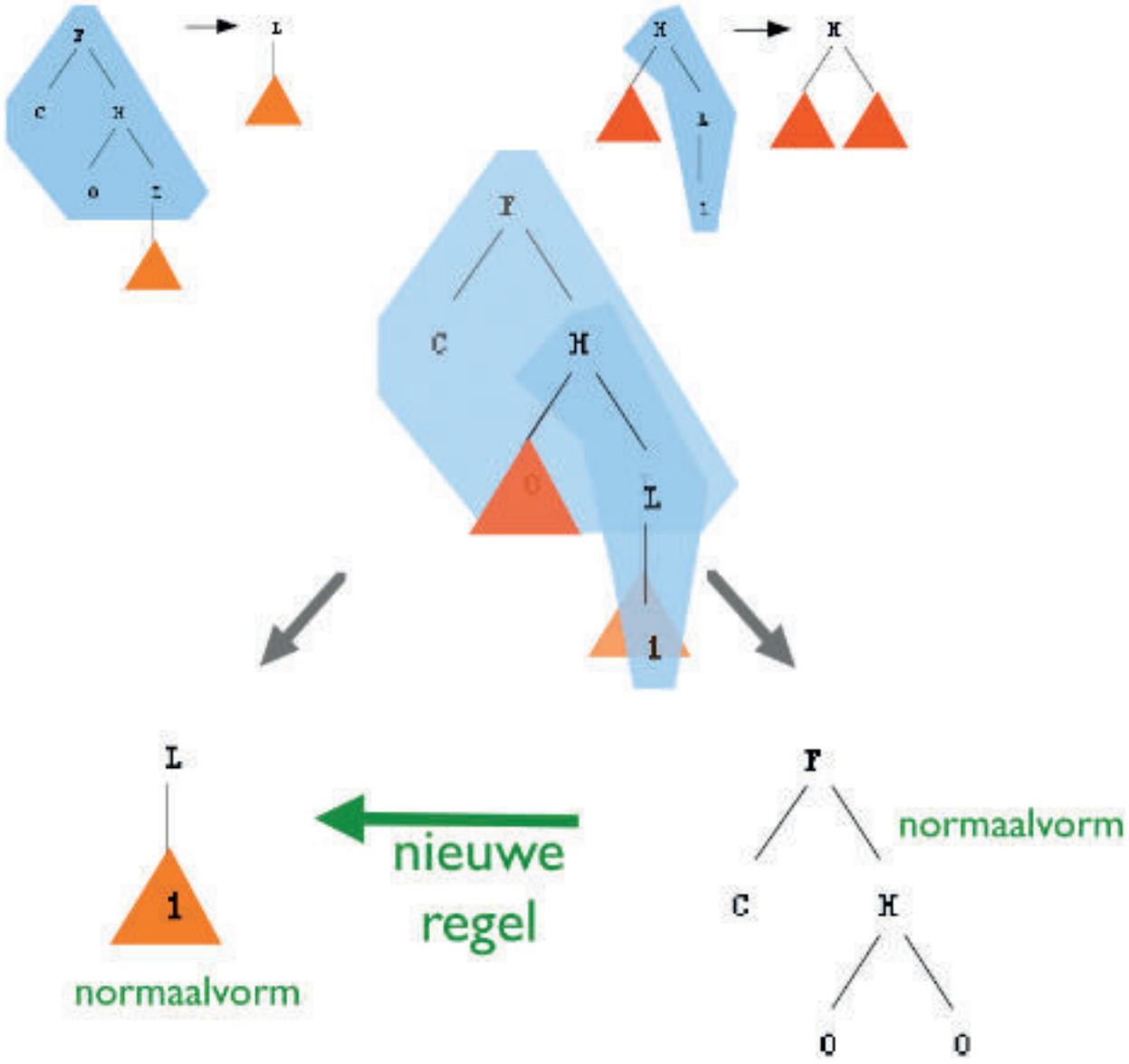
Note that if C[ ] in the above situation is trivial (so that t, t' unify 'at the root') two critical pairs are obtained which are mirror-images: $\langle s'^\sigma, s^\sigma \rangle$ and $\langle s^\sigma, s'^\sigma \rangle$. Such critical pairs are sometimes called 'overlays'. (See the chess-board-like table in Figure 6.1, where it is mentioned which pairs of rules of our example above give rise to critical pairs. The grey squares denote overlays.)

A critical pair $\langle s, t \rangle$ is *convergent* if s, t have a common reduct ($\exists r\ s \twoheadrightarrow r\ \&\ t \twoheadrightarrow r$), notation: $s \downarrow t$. The significance of the fact that all critical pairs $\langle s, t \rangle$ are 'harmless', i.e. convergent, is expressed by the following lemma.

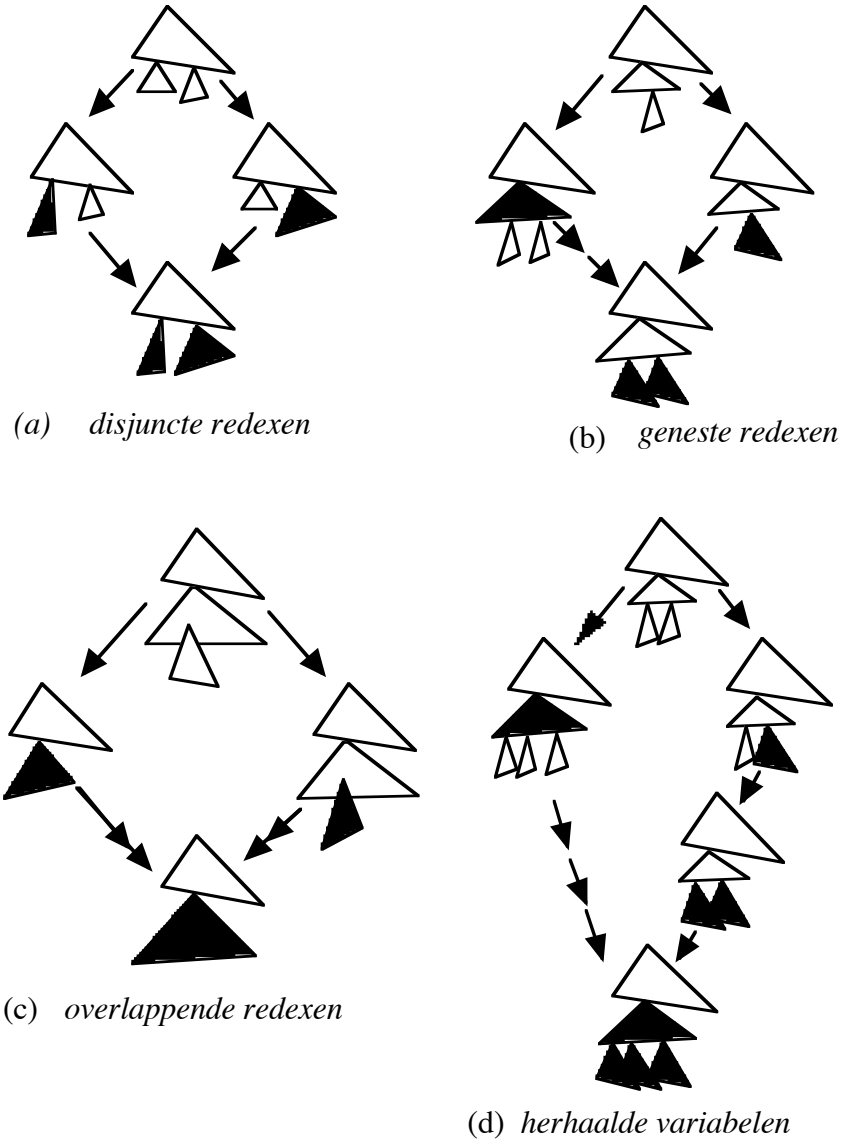


Figuur 6.3.

6.3.2. CRITICAL PAIR LEMMA (Huet [80]). *A TRS is weakly confluent iff all its critical pairs are convergent*.



Figuur 6.4.



(a) *disjuncte redexen*

(b) *geneste redexen*

(c) *overlappende redexen*

(d) *herhaalde variabelen*

Figuur 6.5.

Convergence of all critical pairs is not sufficient for confluence; a counterexample is the ABCD-TRS in the Chapter Confluence, with critical pairs (overlays) $\langle A, C\rangle$, $\langle C, A\rangle$, $\langle B, D\rangle$, $\langle D, B\rangle$. However, in addition to termination, it is sufficient for confluence, according to Newman's Lemma, and we have:

6.3.2. THEOREM (Knuth & Bendix [70]). *A terminating TRS is confluent iff all its critical pairs are convergent*.

As we noted above, convergence of all critical pairs is sufficient for weak confluence, but not for confluence. Huet [80] gave a criterion for critical pairs, stronger than

convergence, which does imply confluence while not requiring termination as in the Knuth-Bendix theorem. First define *parallel reduction* as follows: $t \to_{||} s$ if t reduces to s via a reduction sequence consisting of contracting a set of disjoint redexes in t. Thus, if $t_i\sigma_i \to s_i\sigma_i$ (i = 1, ..., n) are reduction steps, i.e. instances of reduction rules $t_i \to s_i$ (i = 1, ..., n), then $C[t_1\sigma_1, ..., t_n\sigma_n] \to_{||} C[s_1\sigma_1, ..., s_n\sigma_n]$.

6.3.3. THEOREM (Huet [80]). *Let $\mathcal{R}$ be a left-linear TRS such that we have* $s \to_{||} t$ *for every critical pair* $\langle s, t\rangle$. *Then $\mathcal{R}$ is confluent.*

Note the direction involved here. As far as we know, it is an open problem whether the reverse condition also implies confluence: *for all critical pairs* $\langle s, t\rangle$ *we have* $t \to_{||} s$. Also open seems to be the problem whether confluence is implied by the property: *for all critical pairs* $\langle s, t\rangle$ *we have* $s \to^{=} t$ *or* $t \to^{=} s$.

An important aspect in critical pair completion algorithms is that we need to have an ordering of terms at our disposal, guiding us (or the algorithm) how to choose orientations. Thus, at the start of a completion procedure, one must provide the algorithm with a so-called *reduction ordering* on terms; this is a well-founded partial order among terms which is closed under substitutions and contexts, i.e. if $s > t$ then $C[s\sigma] > C[t\sigma]$ for all substitutions σ and contexts C[ ].

The original specification E does not prove $x + y = y + x$ (this follows immediately from the fact that $x + y$ and $y + x$ are different normal forms of $\mathcal{R}$); yet for all ground terms t, s we have $E \vdash t + s = s + t$. That is: $x + y = y + x$ is valid in the initial algebra of E. Such an equation is called an *inductive theorem* (since its validity is usually proved with induction to the structure of ground terms).

6.2. *Groepen*

We will now explain the essential features of the completion algorithm by an informal, "intuition-guided" completion of the equational specification of groups:

e·x	=	x
I(x)·x	=	e
(x·y)·z	=	x·(y·z)

Tabel 6.1.

First we give these equations a 'sensible' orientation:

1.	e·x	→	x
2.	I(x)·x	→	e
3.	(x·y)·z	→	x·(y·z)

(Note that the orientation in rules 1, 2 is forced, by the restrictions on rewrite rules. As to the orientation of rule 3, the other direction is just as 'sensible'.) These rules are not confluent, as can be seen by superposition of e.g. 2 and 3. Redex I(x)·x can be unified (after variable renaming) with a *non-variable* subterm of redex (x·y)·z (the underlined subterm), with result (I(x)·x)·z. This term is subject to two possible reductions: (I(x)·x)·z → e·z and (I(x)·x)·z → I(x)·(x·z). The pair of reducts ⟨e·z, I(x)·(x·z)⟩ is called a *critical pair*, since the confluence property depends on the reduction possibilities of the terms in this pair. Formally, we have the following definition which at a first reading is not easily digested.

6.3.3. DEFINITION. Let $\alpha \to \beta$ and $\gamma \to \delta$ be two rewrite rules such that α is *unifiable* (after renaming of variables) with a subterm of γ which is not a variable (a non-variable subterm). This means that there is a context C[ ], a non-variable term t and a 'most general unifier' σ such that $\gamma \equiv C[t]$ and $t^\sigma \equiv \alpha^\sigma$. The term $\gamma^\sigma \equiv C[t]^\sigma$ can be reduced in two possible ways: $C[t]^\sigma \to C[\beta]^\sigma$ and $\gamma^\sigma \to \delta^\sigma$.

Now the pair of reducts $\langle C[\beta]^\sigma, \delta^\sigma \rangle$ is called a *critical pair* obtained by the superposition of $\alpha \to \beta$ on $\gamma \to \delta$. If $\alpha \to \beta$ and $\gamma \to \delta$ are the same rewrite rule, we furthermore require that α is unifiable with a proper (i.e. not $\equiv \alpha$) non-variable subterm of $\gamma \equiv \alpha$.

6.3.4. DEFINITION. A critical pair ⟨s, t⟩ is called *convergent* if s and t have a common reduct.

Our last critical pair ⟨e·z, I(x)·(x·z)⟩ is not convergent: I(x)·(x·z) is a normal form and e·z only reduces to the normal form z. So we have the problematic pair of terms z, I(x)·(x·z); problematic because their equality is derivable from E, but they have no

common reduct with respect to the reduction available so far. Therefore we adopt a new rule

4. $I(x)\cdot(x\cdot z) \to z$

Now we have a superposition of rule 2 and 4: $I(I(y))\cdot(I(y)\cdot y) \to_4 y$ and $I(I(y))\cdot(I(y)\cdot y) \to_2 I(I(y))\cdot e$. This yields the critical pair $\langle y, I(I(y))\cdot e\rangle$ which cannot further be reduced. Adopt new rule:

5. $I(I(y))\cdot e \to y$ canceled later

As it will turn out, in a later stage this last rule will become superfluous. We go on searching for critical pairs:
Superposition of 4, 1: $I(e)\cdot(e\cdot z) \to_4 z$ and $I(e)\cdot(e\cdot z) \to_1 I(e)\cdot z$.
Adopt new rule:

6. $I(e)\cdot z \to z$ canceled later

Superposition of 3, 5: $(I(Iy))\cdot e)\cdot x \to_3 I(I(y))\cdot(e\cdot x)$ and $(I(Iy))\cdot e)\cdot x \to_5 y\cdot x$.
Adopt new rule:

7. $I(Iy))\cdot x \to y\cdot x$ canceled later

Superposition of 5, 7: $I(I(y))\cdot e \to_7 y.e$ and $I(I(y))\cdot e \to_5 y$.
Adopt new rule:

8. $y\cdot e \to y$

Superposition of 5, 8: $I(I(y))\cdot e \to_5 y$ and $I(I(y))\cdot e \to_8 I(I(y))$.
Adopt new rule

9. $I(I(y)) \to y$ cancel 5 and 7

(Rule 5 is now no longer necessary to ensure that the critical pair $\langle y, I(I(y))\cdot e\rangle$ has a common reduct, because: $I(I(y))\cdot e \to_9 y\cdot e \to_8 y$. Likewise for rule 7.)
Superposition of 6, 8: $I(e)\cdot e \to_6 e$ and $I(e)\cdot e \to_8 I(e)$.
Adopt new rule

10. $I(e) \to e$ cancel 6

Superposition of 2, 9: $I(I(y))\cdot I(y) \to_2 e$ and $I(I(y))\cdot I(y) \to_9 y\cdot I(y)$.
Adopt new rule

11. $y\cdot I(y) \to e$

Superposition of 3, 11: $(y\cdot I(y))\cdot x \to_3 y\cdot(I(y)\cdot x)$ and $(y\cdot I(y))\cdot x \to_{11} e\cdot x$.
Adopt new rule

12. $y\cdot(I(y)\cdot x) \to x$

Superposition (again) of 3, 11: $(x\cdot y)\cdot I(x\cdot y) \to_{11} e$ and $(x\cdot y)\cdot I(x\cdot y) \to_{3} x\cdot(y\cdot I(x\cdot y))$.
Adopt new rule

13. $x\cdot(y\cdot(y\cdot I(x\cdot y)) \to e$ canceled later

Superposition of 13, 4: $I(x)\cdot(x\cdot(y\cdot I(x\cdot y))) \to_{4} y\cdot I(x\cdot y)$ and $I(x)\cdot(x\cdot(y\cdot I(x\cdot y))) \to_{13} I(x)\cdot e$.
Adopt new rule

14. $y\cdot I(x\cdot y) \to I(x)$ canceled later
cancel 13

Superposition of 4, 14: $I(y)\cdot(y\cdot I(x\cdot y)) \to_{4} I(x\cdot y)$ and $I(y)\cdot(y\cdot I(x\cdot y)) \to_{14} I(y)\cdot I(x)$.
Adopt new rule

15. $I(x\cdot y) \to I(y)\cdot I(x)$ cancel 14

At this moment the TRS has only convergent critical pairs, e.g.:

$I(y\cdot I(y))$	$\to_{15}$	$I(I(y))\cdot I(y)$
		$\downarrow_{9}$
$\downarrow_{11}$		$y\cdot I(y)$
		$\downarrow_{11}$
$I(e)$	$\to_{10}$	e

Tabel 6.2.

The significance of this fact is stated in the following lemma.

6.3.5. CRITICAL PAIR LEMMA (Knuth & Bendix [70], Huet [80]).
A TRS R *is WCR iff all critical pairs are convergent.*

So the TRS $\mathcal{R}_c$ with rewrite rules as in Table 6.3 is WCR.

1.	e·x	→	x
2.	I(x)·x	→	e
3.	(x·y)·z	→	x·(y·z)
4.	I(x)·(x·z)	→	z
8.	y·e	→	y
9.	I(I(y))	→	y
10.	I(e)	→	e
11.	y·I(y)	→	e
12.	y·(I(y)·x)	→	x
15.	I(x·y)	→	I(y)·I(x)

Tabel 6.3.

Furthermore, one can prove SN for $\mathcal{R}_c$ by the recursive path ordering(vorige Hoofdstuk). (In fact we need the extended lexicographic version, due to the presence of the associativity rule.) According to Newman's Lemma $\mathcal{R}_c$ is therefore CR and hence complete. We conclude that the validity problem for the equational specification of groups is solvable.

The following theorem of Knuth and Bendix is an immediate corollary of the Critical Pair Lemma and Newman's Lemma:

6.3.6. COROLLARY (Knuth & Bendix [70]). *Let* R *be a TRS which is SN. Then* R *is CR iff all critical pairs of* R *are convergent*. □

The completion procedure above by hand was naive, since we were not very systematic in searching for critical pairs, and especially since we were guided by an intuitive sense only of what direction to adopt when generating a new rule. In most cases there was no other possibility (e.g. at 4: z → I(x)·(x·z) is not a reduction rule due to the restriction that the LHS is not a single variable), but in case 15 the other direction was at least as plausible, as it is even length-decreasing. However, the other direction I(y)·I(x) → I(x·y) would have led to disastrous complications (described in Knuth & Bendix [70]).

The problem of what direction to choose is solved in the actual Knuth-Bendix algorithm and its variants by preordaining a 'reduction ordering' on the terms.

6.3.7. DEFINITION. A *reduction ordering* > is a well-founded partial ordering among terms,

which is closed under substitutions and contexts, i.e. if $s > t$ then $s^\sigma > t^\sigma$ for all substitutions σ, and if $s > t$ then $C[s] > C[t]$ for all contexts $C[\ ]$.

We now have immediately the following fact (noting that if $\mathcal{R}$ is SN, then $\rightarrow_{\mathcal{R}}^+$ satisfies the requirements of Definition 6.3.7):

6.3.8. PROPOSITION. *A TRS $\mathcal{R}$ is SN iff there is a reduction ordering $>$ such that $\alpha > \beta$ for every rewrite rule $\alpha \rightarrow \beta$ of $\mathcal{R}$.* □

Simple version of the Knuth-Bendix completion algorithm

Input: - an equational specification (Σ, E)
- a reduction ordering $>$ on $Ter(\Sigma)$ (i.e. a program which computes $>$)

Output: - a complete TRS R such that for all $s,t \in Ter(\Sigma)$: $s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

```
R := Ø;
while E ≠ Ø do
        choose an equation s = t ∈ E;
        reduce s and t to respective normal forms s' and t' with respect to R;
        if s' ≡ t' then
                E := E - {s = t}
        else
                if s' > t' then
                        α := s'; β := t'
                else if t' > s' then
                        α := t'; β := s'
                else
                        failure
                fi;
                CP := {P = Q | ⟨P, Q⟩ is a critical pair between the rules in R and α → β};
                R := R ∪ {α → β};
                E := E ∪ CP - {s = t}
        fi
od;
success
```

Figuur 6.4.

In Figure 6.4 a simple version of the Knuth-Bendix completion algorithm is presented. The program of Figure 6.4 has three possibilities: it may

(1) *terminate successfully,*
(2) *not terminate and run forever,*
(3) *fail*

because a pair of terms s, t cannot be oriented (i.e. neither $s > t$ nor $t > s$). The third case gives the most important restriction of the Knuth-Bendix algorithm: equational specifications with commutative operators cannot be completed.

If one still wants to deal with equational specifications having commutative/associative operators as in Exercise 5.4.3, one has to work modulo the equations of associativity and commutativity.

In case (1) the resulting TRS is complete.

The completion program of Figure 6.4 does not 'simplify' the rewrite rules themselves. Such an optimization can be performed after termination of the program, as

follows.

6.3.9. DEFINITION. A TRS $\mathcal{R}$ is called *irreducible* if for every rewrite rule $\alpha \to \beta$ of $\mathcal{R}$ the following hold:

(i) β is a normal form with respect to $\mathcal{R}$,
(ii) α is a normal form with respect to R - $\{\alpha \to \beta\}$.

6.3.10. THEOREM (Métivier [83]). *Let $\mathcal{R}$ be a complete TRS. Then we can find an irreducible complete TRS $\mathcal{R}'$ such that the convertibilities $=_{\mathcal{R}}$ and $=_{\mathcal{R}'}$ coincide.*

Instead of optimizing the TRS which is the output of the above simple completion algorithm *after* the completion, it is more efficient to do this *during* the completion. Tabel 6.5 contains a more efficient Knuth-Bendix completion algorithm, which upon successful termination yields irreducible TRSs as output.

We conclude this section with a theorem (6.3.12) stating that the Knuth-Bendix completion algorithm, given an equational specification and a reduction ordering, cannot generate two different complete irreducible TRSs.

6.3.11. DEFINITION. Let $>$ be a reduction ordering. We call a TRS $\mathcal{R}$ *compatible* with $>$ if for every rewrite rule $\alpha \to \beta$ of $\mathcal{R}$ we have $\alpha > \beta$.

More efficient version of the Knuth-Bendix completion algorithm

Input: - an equational specification (Σ, E)
- a reduction ordering $>$ on $Ter(\Sigma)$ (i.e. a program which computes $>$)

Output: - a complete irreducible TRS R such that for all $s,t \in Ter(\Sigma)$: $s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

```
R := Ø;
while E ≠ Ø do
        choose an equation s = t ∈ E;
        reduce s and t to respective normal forms s' and t' with respect to R;
        if s' ≡ t' then
                E := E - {s = t}
        else
                if s' > t' then
                        α := s'; β := t'
                else if t' > s' then
                        α := t'; β := s'
                else
                        failure
                fi;
                R := {γ → δ' | γ→ δ ∈ R and δ' is a normal form of δ with respect to R ∪ {α → β} };
                CP := {P = Q | ⟨ P, Q⟩ is a critical pair between the rules in R and α → β};
                E := E ∪ CP ∪ {γ = δ | γ → δ ∈ R and γ is reducible by α → β} - {s = t};
                R := R ∪ {α → β} - {γ → δ | γ is reducible by α → β}
        fi
od;
success
```

Tabel 6.5.

6.3.12. THEOREM. (Métivier [83]) *Let* R_1 *and* R_2 *be two complete irreducible TRSs compatible with a given reduction ordering* $>$. *Suppose* R_1 *and* R_2 *define the same convertibility. Then* R_1 *and* R_2 *are equal (modulo a renaming of variables).* □

6.4. *Modulo rekenen.*

Het derde voorbeeld van een completering doen we niet 'met de hand', maar automatisch, namelijk met het herschrijftool CIME 1.15 te vinden op:

`http://cime.lri.fr/cime-1.15.html`

We voegen aan de AMS0 TRS toe: $S(S(0)) \rightarrow 0$. Dat wil zeggen dat we *modulo 2* willen rekenen. De resulterende TRS is niet meer orthogonaal, en niet meer confluent. We gaan de TRS dus completeren, en een handmatige poging leert al gauw dat dit verrassend ingewikkeld is, zozeer dat we machinale hulp nodig hebben. Vincent van Oostrom voerde

met behulp van CIME 1.15 de volgende completeringen uit, waarvan de resultaten staan in Tabellen 6.8 -6.12. Het blijkt dat de completering 'modulo n' een compleet systeem met n+6 regels geeft, waarvan de eerste vier de + en * regels, dan een regel (die je schematisch kunt schrijven als) n+x → x, regels nx → 0 en nx + x*y → x*y, en voor iedere 0 < i < n, een regel (n-1)(i+x) + x → n-i. (Opmerking van Vincent van Oostrom.)

r_1	A(x,0)	→	x
r_2	A(x,S(y))	→	S(A(x,y))
r_3	M(x,0)	→	0
r_4	M(x,S(y))	→	A(M(x,y),x)
r_5	(S(S(0))	→	0

Tabel 6.6. De AMS0 TRS modulo 2.

r_1	$\oplus(x, \bot)$	→	x
r_2	$\oplus(x, \neg(y))$	→	$\neg(\oplus(x, y))$
r_3	$\wedge(x, \bot)$	→	$\bot$
r_4	$\wedge(x, \neg(y))$	→	$\oplus(\wedge(x, y), x)$
r_5	$\neg(\neg(\bot))$	→	$\bot$

Tabel 6.7. De $\oplus \wedge \neg \bot$ TRS.

[1]	x+0	→	x
[2]	x*0	→	0
[4]	x+s(y)	→	s(x+y)
[5]	s(s(x))	→	x
[6]	x*s(y)	→	x+(x*y)
[7]	x+(x+(x*y))	→	x*y
[8]	x+x	→	0
[10]	s(x)+x	→	s(0)

Tabel 6.8. *Modulo 2.*

6.1. OPGAVE. Laat zien dat de regel S(S(x)) → 0 'vanzelf' gegenereerd wordt door de completering. Dit heet een 'inductive theorem', i.e. een vergelijking die geldt in het initiële model van de specificatie, maar niet afleidbaar is.
(ii) Laat zien dat de vergelijking S(S(x)) = x inderdaad niet afleidbaar is uit de equationele

versie van AMS0, door middel van een tegenmodel, met in plaats van AMS0 de vier operatoren $\oplus \wedge \neg \bot$ (*xor*, *and*, *not*, *false*)

[1]	x+0 → x
[2]	x*0 → 0
[4]	x+s(y) → s(x+y)
[5]	s(s(s(x))) → x
[6]	x*s(y) → x+(x*y)
[7]	x+(x+(x+(x*y))) → x*y
[8]	x+(x+x) → 0
[11]	s(x)+(s(x)+x) → s(s(0))
[13]	s(s(x))+(s(s(x))+x) → s(0)

Tabel 6.9. *Modulo 3*.

[1]	x+0 → x
[2]	x*0 → 0
[3]	x+s(y) → s(x+y)
[5]	s(s(s(s(x)))) → x
[6]	x*s(y) → x+(x*y)
[7]	x+(x+(x+(x+(x*y)))) → x*y
[8]	x+(x+(x+x)) → 0
[11]	s(x)+(s(x)+(s(x)+x)) → s(s(s(0)))
[13]	s(s(x))+(s(s(x))+(s(s(x))+x)) → s(s(0))
[16]	s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+x)) → s(0)

Tabel 6.10. *Modulo 4*.

[1]	x+0 → x
[2]	x*0 → 0
[3]	x+s(y) → s(x+y)
[4]	x*s(y) → x+(x*y)
[6]	s(s(s(s(s(x))))) → x
[7]	x+(x+(x+(x+(x+(x*y))))) → x*y
[8]	x+(x+(x+(x+x))) → 0
[12]	s(x)+(s(x)+(s(x)+(s(x)+x))) → s(s(s(s(0))))
[15]	s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+x))) → s(s(s(0)))
[17]	s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+x))) → s(s(0))
[19]	s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+x))) → s(0)

Tabel 6.11. *Modulo 5.*

x+0 → x
x*0 → 0
x+s(y) → s(x+y)
x*s(y) → x+(x*y)
s(s(s(s(s(s(x)))))) → x
x+(x+(x+(x+(x+(x+(x*y)))))) → x*y
x+(x+(x+(x+(x+x)))) → 0
s(x)+(s(x)+(s(x)+(s(x)+(s(x)+(s(x)+x))))) → s(s(s(s(s(0)))))
s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+(s(s(x))+x))))) → s(s(s(s(0))))
s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+(s(s(s(x)))+x))))) → s(s(s(0))))
s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+(s(s(s(s(x))))+x))))) → s(s(0)))
s(s(s(s(s(x)))))+(s(s(s(s(s(x)))))+(s(s(s(s(s(x)))))+(s(s(s(s(s(x))))) +(s(s(s(s(s(x)))))+(s(s(s(s(s(x)))))+x))))) → s(s(0))
s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+(s(s(s(s(s(s(x))))))+x)))))
→ s(0)

Figuur 6.12. *Modulo 7.*

6.5. Exercises.

6.5.1. EXERCISE. Prove, using the Critical Pair Lemma: If the TRS $\mathcal{R}$ has finitely many rules and is SN, then WCR and CR are decidable.

6.5.2. EXERCISE. Prove that every irreducible ground TRS is complete.

(Hint: use the following Exercise 6.5.3 to show SN and the theorem of Knuth and Bendix to show CR.)

6.5.3. EXERCISE. If R is not SN, there is a redex of R whose contractum is not SN.

6.5.4. EXERCISE. Prove, using the completions, that x·e = x is not derivable in L-R theory and that in R-L theory the equations e·x = x and x·I(x) = e are not derivable. (see Table 6.13.) Furthermore, in L-R theory the equation x·e = x is not derivable. Hence the three theories are different, i.e. determine different varieties of algebras.

In fact, note that the variety of groups is the intersection of both the variety of L-R algebras and that of R-L algebras, and that the latter two varieties are incomparable with respect to set inclusion.

group theory	*L-R theory:*	*R-L theory:*
e·x = x	e·x = x	x·e = x
I(x)·x = e	x·I(x) = e	I(x)·x = e
(x·y)·z = x·(y·z)	(x·y)·z = x·(y·z)	(x·y)·z = x·(y·z)
completion:	*completion:*	*completion:*
e·x → x	e·x → x	
x·e → x		x·e → x
I(x)·x → e		I(x)·x → e
x·I(x) → e	x·I(x) → e	
(x·y)·z → x·(y·z)	(x·y)·z → x·(y·z)	(x·y)·z → x·(y·z)
I(e) → e	I(e) → e	I(e) → e
I(x·y) → I(y)·I(x)	I(x·y) → I(y)·I(x)	I(x·y) → I(y)·I(x)
x·(I(x)·y) → y	x·(I(x)·y) → y	
		e·x → I(I(x))
I(x)·(x·y) → y	I(x)·(x·y) → y	
		x·I(I(y) → x·y
I(I(x)) → x		
	x·e → I(I(x))	
	I(I(I(x))) → I(x)	I(I(I(x))) → I(x)
		x·(y·I(y)) → x
	I(I(x))·y → x·y	
		x·(I(I(y))·z) → x·(y·z)
		x·(y·(I(y)·z)) → x·z
		I(x)·(x·y) → I(I(y))

Tabel 6.13.

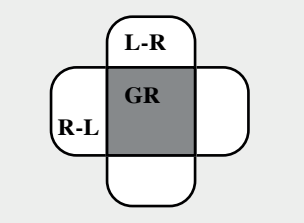


Figure 6.5

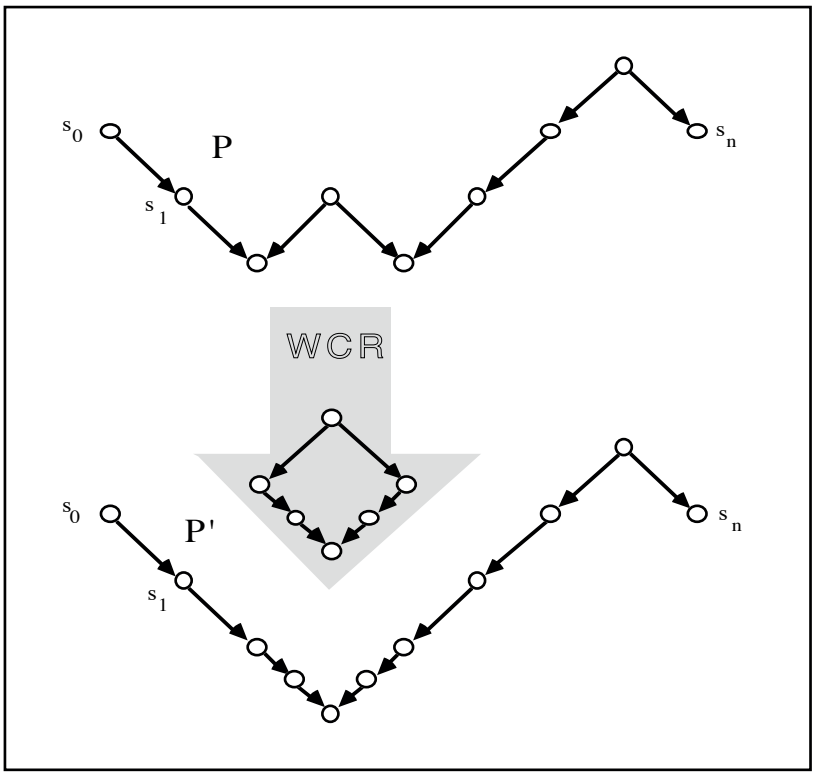


Figure 6.6

6.5.5. OPGAVE. Gegeven is de TRS R met regels

$$a(a(x)) \to b(x)$$
$$b(b(x)) \to x$$
$$b(a(x)) \to a(b(x))$$

(i) Is R een OTRS?
(ii) Wat zijn de normaalvormen van R?
(iii) Bewijs dat R de eigenschap WCR heeft.
(iv) Bewijs dat R de eigenschap SN heeft.

6.5.6. OPGAVE. Bewijs dat de TRS met de regels

$g(x) \to f(f(x))$
$h(f(x)) \to g(g(x))$

6.5.7. OPGAVE. Completeer de TRS met de drie regels

$a(b(c(x))) \to x$
$a(b(x)) \to x$
$b(c(x)) \to x$

6.5.8. OPGAVE. Bewijs dat de TRS R met herschrijfregels

$F(x) \to A$
$F(x) \to G(F(x))$
$G(F(x)) \to F(H(x))$

WCR is.

6.5.9. OPGAVE. Completeer de volgende equationele specificaties:

1. $F(F(F(x))) = x$
   $F(F(F(F(F(x))))) = x$

2. $(x.y).(y.z) = y$ (met oplossing)

6.5.10. OPGAVE. Ga na of de volgende TRS-en SN zijn, en geef tegenvoorbeeld of bewijs.

1. $g(x) \to f(x)$, $f(x) \to x$
2. $g(f(x)) \to f(f(g(x)))$
3. $g(f(x)) \to f(f(g(g(x))))$
4. $f(f(x)) \to f(g(f(x)))$

6.5.11. OPGAVE. Gegeven de reductieregel $L(L(L(x))) \to H(x)$.
Hoe kan deze regel met zichzelf overlappen?
Welke kritieke paren worden daardoor gegenereerd?

6.5.12. OPGAVE. Zelfde vragen voor het paar regels

$S(P(x)) \rightarrow x$
$P(S(x)) \rightarrow x$

6.5.13. OPGAVE. Zelfde vragen voor het trio regels

$A(x, 0) \rightarrow x$
$A(x, y) \rightarrow A(y, x)$
$A(A(x,y),z) \rightarrow A(x, A(y,z))$

6.5.14. OPGAVE. Gegeven is de TRS met regels

or(true, true) → true
or(x, y) → or (y, x)

1. Is dit een orthogonaleTRS?
2. Is de TRS CR? Geef bewijs of tegenvoorbeeld.

6.5.15. OPGAVE. Beschouw de AMS0 TRS in combinatie met

$\min(\min(x)) \rightarrow x$
$A(x, \min(x)) \rightarrow 0$
$A(\min(x), x) \rightarrow 0$

i. Welke kritieke paren zijn er?
ii. Bewijs of weerleg WCR.

6.5.16. OPGAVE. Is de volgende TRS SN?

$F(x, G(y)) \rightarrow H(x, x)$
$F(G(x), F(y,y)) \rightarrow y$

6.5.17. OPGAVE. Is de volgende TRS SN?

$-(x+y) \rightarrow (-(-x)+y)+y$

6.5.18. OPGAVE. Bewijs SN:

$G(x, y) \rightarrow H(x, y)$
$H(F(x), y) \rightarrow F(G(x, y))$

6.5.19. OPLOSSING. *Het cola-gen.*

(i) Oplossing van Milo van Handel.
Virus problem:

The invariant to look at is the parity of the sum of the amounts of As and Ts. Forexample, the cola gene counts 2 As and 3 Ts for a total of 5, which is odd.

It is easy to see that the second and third rewriting rules do not modify the number of As and Ts, while the other three modify each by one, thus modifying the sum by two.

The milk gene counts 3 As and 4 Ts, totalling 7, which is odd. The virus gene counts 2 As and 4 Ts, totalling 6, which is even. Thus neither can be converted into the other.

(In fact, there is an even better invariant: the difference between the number of As and Ts, as an integer rather than just the parity. The milk and cola genes have T-A=1, while the virus gene has T-A=2.)

Cola problem:

Lowercase letters are the ones being substituted.

TAGctAGCTAGCT
TagtaTAGCTAGCT
tatAGCTAGCT
CTagCTAGCT
CTGAGctAGCT

CTGagtaTAGCT
CTGATAGct
CTGATagtaT
CTGAtAT
CTGAtcATAT
CTGACtcatAT
CTGACTaT
CTGACTagTAT
CTGACTGagtaT
CTGACTGAt
CTGACTGAtcAT
CTGACTGACtcat
CTGACTGACT

6.5.20. OPGAVE. Beschouw een signatuur met constante 0 en binaire functiesymbolen . and |. Beschouw de TRS met reductieregels

1.	$x.0 \rightarrow x$
2.	$0.x \rightarrow x$
3.	$x \mid 0 \rightarrow x$
4.	$0 \mid x \rightarrow 0$
5.	$x \mid x \rightarrow 0$
6.	$(x.y) \mid z \rightarrow (x \mid z).(y \mid (z \mid x))$
7.	$z \mid (x.y) \rightarrow (z \mid x) \mid y$

Tabel 6.14.

Als feit is gegeven dat deze TRS de eigenschap SN heeft. Bewijs met de stelling van Knuth-Bendix dat hij ook CR is.

Week 9: ma 19 april 1999

OPGAVE 3.2: Completeer de equationele specificatie

(xy)(yz) = y.

Het gewicht van een term is gedefinieerd als het aantal symbolen in die term. De ordening op de termen is dan de natuurlijke ordening op hun gewichten. Hieronder zijn de stappen van het Knuth-Bendix algoritme (de efficiente versie) weergegeven in een tabel. Merk op de R twee keer per stap geevalueerd wordt.

```
   a -> b            E                   R                CP

initiele situatie:
                 (xy)(yz) = y           empty


                                                           (1 vs 1):
1: (xy)(yz) -> y   y((yz)w) = yz     - empty               y((yz)w) = yz
                   (w(xy))y = xy     - (xy)(yz) -> y       (w(xy))y = xy


                                                           (1 vs 2):
2: y((yz)w) -> yz  (w(xy))y = xy     - (xy)(yz) -> y       (yz)(((yz)w)u) = (yz)w
                   plus 6 kritieke   - (xy)(yz) -> y       (xy)(yz) = y
                   paren               y((yz)w) -> yz      yz = yz
                   (zie rechts)                            (xy)(yw) = (xy)(yz)
                                                           (2 vs 2):
                                                           y((yz)u) = yz
                   /\                                      y((yz)w) = y((yz)u)
                   ||
                 deze kritieke paren zijn allemaal triviaal na de
                 reductie mbv twee regels van de huidige R
                 - er blijft nog alleen (w(xy))y = xy over in E

                                                           (3 vs 3):
3: (w(xy))y -> xy  de kritieke    - (xy)(yz) -> y         (w(xy))y = xy
                   paren (zie       y((yz)w) -> yz        (w(xy))y = (u(xy))y
                   rechts)        - (xy)(yz) -> y         (2 vs 3):
                                    y((yz)w) -> yz        w(xy) = w(xy)
                                    (w(xy))y -> xy        (yz)w = (yz)w
                                                          (w(xy))y = (w(xy))((xy)u)
                                                          (u(yz))((yz)w) = y((yz)w)
                                                          (1 vs 3):
                                                          yz = yz
                                                          (xy)(yz) = (wy)(yz)
                                                          (xy)(yz) = y
                                                          (u(w(xy)))(xy) = w(xy)
                   /\                                     y((yz)w) = y((yz)u)
                   ||
                 deze kritieke paren zijn allemaal triviaal na de
                 reductie mbv drie regels van de huidige R

                   E is leeg

                                 /\
                                 ||
                                 resulterende R
```

De complete TRS is dus R = { (xy)(yz) -> y, y((yz)w) -> yz, (w(xy))y -> xy}.

6.6. *Equationele logica, initiele algebra semantiek en de stelling van Birkhoff.*

An equational specification is just a TRS "without orientation". More precisely, an equational specification is a pair (Σ, E) where Σ is the signature (or alphabet), and where E is a set of equations $s = t$ between terms $s, t \in Ter(\Sigma)$.

If an equation $s = t$ is *derivable* from the equations in E, we write $(\Sigma, E) \vdash s = t$ or $s =_E t$. Formally, derivability is defined by means of the inference system of Table 5.1.

$(\Sigma, E) \vdash s = t$	if $s = t \in E$
$\dfrac{(\Sigma, E) \vdash s = t}{(\Sigma, E) \vdash s^\sigma = t^\sigma}$	for every substitution σ
$\dfrac{(\Sigma, E) \vdash s_1 = t_1 \,,\ldots,\, (\Sigma, E) \vdash s_n = t_n}{(\Sigma, E) \vdash F(s_1,\ldots,s_n) = F(t_1,\ldots,t_n)}$	for every n-ary $F \in \Sigma$
$(\Sigma, E) \vdash t = t$	
$\dfrac{(\Sigma, E) \vdash t_1 = t_2\,,\ (\Sigma, E) \vdash t_2 = t_3}{(\Sigma, E) \vdash t_1 = t_3}$	
$\dfrac{(\Sigma, E) \vdash s = t}{(\Sigma, E) \vdash t = s}$	

Tabel 6.14.

If Σ is a signature, a Σ-*algebra* $\mathcal{A}$ is a set A together with functions $F^{\mathcal{A}}: A^n \to \mathcal{A}$ for every n-ary function symbol $F \in \Sigma$. (If F is 0-ary, i.e. F is a constant, then $F^{\mathcal{A}} \in A$.) An equation $s = t$ $(s, t \in Ter(\Sigma))$ is assigned a meaning in $\mathcal{A}$ by interpreting the function symbols in s, t via the corresponding functions in $\mathcal{A}$. Variables in $s = t$ are (implicitly) universally quantified. If the universally quantified statement corresponding to $s = t$ $(s, t \in Ter(\Sigma))$ is true in $\mathcal{A}$, we write $\mathcal{A} \models s = t$ and say that $s = t$ is *valid* in $\mathcal{A}$. $\mathcal{A}$ is called a *model* of a set of equations E if every equation in E is valid in $\mathcal{A}$. Abbreviation: $\mathcal{A} \models E$. The *variety* of Σ-algebras defined by an equational speci-

fication (Σ, E), notation $Alg(\Sigma, E)$, is the class of all Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \models E$. Instead of $\forall \mathcal{A} \in Alg(\Sigma, E)\ \mathcal{A} \models F$, where F is a set of equations between Σ-terms, we will write $(\Sigma, E) \models F$. There is the well-known completeness theorem for equational logic of Birkhoff:

5.1.1. THEOREM. *Let* (Σ, E) *be an equational specification. Then for all* $s, t \in Ter(\Sigma)$:

$$(\Sigma, E) \vdash s = t \ \Leftrightarrow\ (\Sigma, E) \models s = t. \ \square$$

Now the *validity problem* or *uniform word problem* for (Σ, E) is:

> *Given an equation* $s = t$ *between* Σ*-terms, decide whether or not* $(\Sigma, E) \models s = t$.

According to Birkhoff's completeness theorem for equational logic this amounts to deciding $(\Sigma, E) \vdash s = t$. Now we can state why *complete* TRSs (i.e. TRSs which are SN and CR) are important. Suppose for the equational specification (Σ, E) we can find a complete TRS (Σ, R) such that for all terms $s, t \in Ter(\Sigma)$:

$$t =_R s \ \Leftrightarrow\ E \vdash t = s$$

Then (provided R has finitely many rewrite rules only) we have a positive solution of the validity problem. The decision algorithm is simple:

> (1) *Reduce* s *and* t *to their respective normal forms* s', t'
> (2) *Compare* s' *and* t': $s =_R t$ *iff* $s' \equiv t'$.

It is important to realize that we have considered up to now equations $s = t$ between possibly *open* Σ-terms (i.e. possibly containing variables). If we restrict attention to equations $s = t$ between *ground* terms s, t, we are considering the *word problem* for (Σ, E), which is the following decidability problem:

Given an equation s = t *between ground terms* s, t ∈ Ter(Σ),
decide whether or not (Σ, E) ⊨ s = t
or equivalently, (Σ, E) ⊢ s = t .

Also for the word problem, complete TRSs provide a positive solution.

5.1.2. REMARK. Note that there are finite equational specifications (Σ, E) which have a decidable word problem (so for ground terms) for which no complete TRS R (complete with respect to ground terms) exists. This strengthens the observation in Exercise 5.4.3. The simplest such (Σ, E) is the specification consisting of a single binary commutative operator + and a constant 0, and equations E = {x + y = y + x}. According to Exercise 5.4.3 (which also works for the present simpler specification) no complete TRS R can be found such that for all (open) s, t we have $s =_R t \Leftrightarrow s =_E t$. According to Exercise 5.4.5, we also have the stronger result that no TRS R exists which is complete for ground terms and such that for ground terms s, t we have $s =_R t \Leftrightarrow s =_E t$.

5.2. Term rewriting and initial algebra semantics.

We will now make more explicit the connection between term rewriting and initial algebra semantics. We suppose familiarity with the concept of an initial algebra in the class of models of an equational specification (Σ, E), i.e. the variety Alg(Σ, E), as defined by universal properties in terms of homomorphisms. Although the initial algebra is only determined up to isomorphism, we will speak of 'the' initial algebra and use the notation $\mathcal{I}(\Sigma, E)$ for it. It is well-known that $\mathcal{I}(\Sigma, E)$ can be obtained from the set of ground terms $Ter_0(\Sigma)$ by dividing out the congruence relation $=_E$. Thus we can equate the initial algebra $\mathcal{I}(\Sigma, E)$ with the quotient algebra $Ter_0(\Sigma)/=_E$.

Now suppose that (Σ, $\mathcal{R}$) is a TRS 'for' (Σ, E), that is, $=_{\mathcal{R}}$ coincides with $=_E$. (So the initial algebra of (Σ, E) can also be written as $Ter_0(\Sigma)/=_R$.) If $\mathcal{R}$ is a *complete* TRS, then $\mathcal{I}(\Sigma, E)$ is in fact a *computable* algebra. This is merely a rephrasing of: the word problem (for ground terms) for (Σ, E) is solvable. As noted in Exercise 5.4.5, the reverse is not necessarily the case; for some (Σ, E) with computable initial algebra there does not exist a complete TRS—at least not *in the same signature*. However, a theorem of Bergstra and Tucker states that if we allow an extension of the signature, with some functions and constants (no new sorts), then a complete TRS can always be found. More precisely:

5.2.1. DEFINITION. (i) The algebra $\mathcal{A} \in$ Alg(Σ, E) is *minimal*, if it is (isomorphic to) a quotient algebra Ter(Σ)/≡ for some congruence ≡. In particular, $\mathcal{I}(\Sigma, E)$ is a minimal algebra. In other words, an algebra

is minimal if its elements are generated by functions and constants in the signature.
(ii) A minimal algebra $\mathcal{A}$ is *computable*, if its equality is decidable, i.e. if the relation $\mathcal{A} \models t = s$ for ground terms $t, s \in Ter(\Sigma)$ is decidable.

5.2.2. THEOREM (Bergstra & Tucker [80]).
Let $\mathcal{A}$ be a minimal Σ-algebra, Σ a finite signature. Then the following are equivalent:
(i) *$\mathcal{A}$ is a computable algebra;*
(ii) *there is an extension of Σ to a finite Σ', obtained by adding some function and constant symbols, and there is a complete TRS $(\Sigma', \mathcal{R})$ such that*

$$\mathcal{A} \cong \mathcal{I}(\Sigma', \mathcal{R}^{=})|_{\Sigma}.$$

Here $\mathcal{R}^{=}$ is the equational specification obtained by viewing the reduction rules in $\mathcal{R}$ as equations, and $|_{\Sigma}$ is the restriction to the signature Σ. So $\mathcal{A}$ is a '*reduct*' (see Meinke & Tucker [90]) of an initial algebra given by a complete TRS. (The TRS $\mathcal{R}$ as in the theorem is not only ground complete, but complete with respect to all terms. Actually, it is an orthogonal TRS; and for orthogonal TRSs possessing at least one ground term, ground completeness implies completeness.) The functions (including the constants as 0-ary functions) to be added to Σ are sometimes referred to as '*hidden functions*'.

5.4.1. EXERCISE. **Equational deduction systems.**
Often the inference system in Table 5.1 is presented slightly different, as follows. Prove the equivalence of the two versions below with the system above.

Axioms (in addition to the equations in E*):*		
Rules:	$t = t$	*reflexivity*
	$\dfrac{t_1 = t_2}{t_2 = t_1}$	*symmetry*
	$\dfrac{t_1 = t_2 \quad t_2 = t_3}{t_1 = t_3}$	*transitivity*
	$\dfrac{t_1 = t_2}{t_1[x:=t] = t_2[x:=t]}$	*substitution (1)*
	$\dfrac{t_1 = t_2}{t[x:=t_1] = t[x:=t_2]}$	*substitution (2)*

Tabel 6.15.

Here [x:=t] denotes substitution of t for all occurrences of x. An equivalent formulation is to combine the two substitution rules in one:

$$\frac{t_1 = t_2 \quad t = t'}{t_1[x:=t] = t_2[x:=t']} \qquad \textit{substitution}$$

5.4.2. EXERCISE. (Puzzle.) (*Ternary Boolean Algebras*, from: Wos, Overbeek, Lusk & Boyle [84], p.263.)
Let E = {A_1, A_2, A_3}, with

A_1: F(F(v, w, x), y, F(v, w, z)) = F(v, w, F(x, y, z))

A_2: F(y, x, x) = x

A_3: F(x, y, G(y)) = x.

Prove: E ⊢ F(x, x, y) = x.

5.4.3. EXERCISE. Let (Σ, E) be the specification given by the equations

x + 0 = x
x + S(y) = S(x + y)
x + y = y + x

Prove that there is no complete TRS $\mathcal{R}$ 'for' E, i.e. such that for all terms s, t ∈ Ter(Σ): $s =_{\mathcal{R}} t$ ⇔ $s =_E t$. (Consider in a supposed complete TRS $\mathcal{R}$, the normal forms of the open terms x + y and y + x.)

5.4.4. EXERCISE. Consider the specification as in the previous exercise and find a TRS (Σ, $\mathcal{R}$) such that $(\Sigma, \mathcal{R})_0$ (i.e. the restriction of (Σ, $\mathcal{R}$) to ground terms) is complete.

5.4.5. EXERCISE (Bergstra & Klop). Prove the following fact:

THEOREM. *Let* (Σ, E) *be the specification with* Σ = {0, +} *and* E = {x + y = y + x}. *Then there is no finite TRS* $\mathcal{R}$ *such that the restriction to ground terms,* $(\mathcal{R})_0$, *is complete and such that* $=_{\mathcal{R}}$ *and* $=_E$ *coincide on ground terms*.

PROOF SKETCH. Define terms $t_0 \equiv 0$, $t_{n+1} \equiv t_n + t_n$ ($n \geq 0$). Suppose $\mathcal{R}$ is a TRS with finitely many rewrite rules such that $=_{\mathcal{R}}$ and $=_E$ coincide on ground terms. Let N be the maximum of the depths of the LHSs of the rewrite rules in $\mathcal{R}$. (Here 'depth' refers to the height of the corresponding term formation tree.)

Consider the terms $t^* \equiv t_N + t_{2N}$ and $t^{**} \equiv t_{2N} + t_N$. Clearly, $t^* =_E t^{**}$. In fact, {t*, t**} is an E-equivalence class, hence also an $\mathcal{R}$-convertibility class. Therefore there must be a rewrite rule r such that t* is an r-redex or t** is an r-redex (since there are only two elements in the convertibility class) and such that $t^* \to_r t^{**}$. Say t* is an r-redex. Now one can easily show that $t^* \to_r t^{**} \to_r t^*$. Hence R is not even SN on ground terms.

□

9
ONEINDIG HERSCHRIJVEN

In this chapter we will consider infinite terms over a first order signature. So, our starting point is an ordinary TRS (Σ, R). In fact, we will suppose throughout that our TRSs are *orthogonal*. Now it is obvious that the rules of the TRS (Σ, R) just as well apply to infinite terms as to the usual finite ones. First, let us explain the notion of infinite term that we have in mind. As before, let $Ter(\Sigma)$ be the set of finite Σ-terms. Then $Ter(\Sigma)$ can be equipped with a distance function d such that for $t, s \in Ter(\Sigma)$, we have $d(t, s) = 2^{-n}$ if the n-th level of the terms s, t (viewed as labeled trees) is the first level where a difference appears, in case s and t are not identical; furthermore, $d(t, t) = 0$. It is well-known that this construction yields $(Ter(\Sigma), d)$ as a metric space. Now infinite terms are obtained by taking the completion of this metric space, and they are represented by infinite trees. We will refer to the complete metric space arising in this way as $(Ter^{\infty}(\Sigma), d)$, where $Ter^{\infty}(\Sigma)$ is the set of finite and infinite terms over Σ.

A natural consequence of this construction is the emergence of the notion of *Cauchy convergence*: we say that $t_0 \to t_1 \to t_2 \to \ldots$ is an infinite reduction sequence with limit t, if t is the limit of the sequence $t_0, t_1, \ldots$ in the usual sense of Cauchy convergence. See Figure 9.1 for an example, based on a rewrite rule $F(x) \to P(x, F(S(x)))$ in the presence of a constant 0.

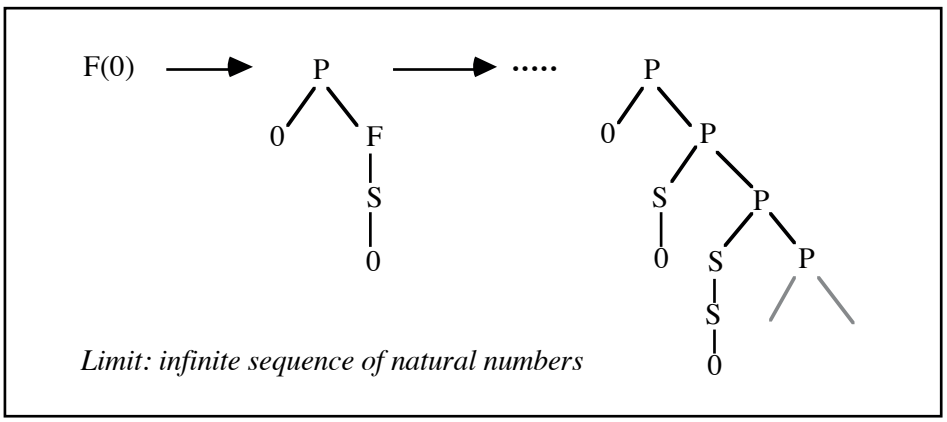


Figure 9.1

In the sequel we will however adopt a stronger notion of converging reduction sequence which turns out to have better properties. First, let us argue that it makes sense to consider not only reduction sequences of length ω, but even reduction sequences of length α for arbitrary ordinals α. Given a notion of convergence, and limits, we may iterate reduction sequences beyond length ω and consider

e.g.

$$t_0 \to t_1 \to t_2 \to \ldots \to t_n \to \ldots$$
$$s_0 \to s_1 \to s_2 \to s_3 \to \ldots \; r$$

where $\lim_{n\to\infty} t_n = s_0$ and $\lim_{n\to\infty} s_n = r$. See Figure 8.2 for such a reduction sequence of length $\omega + \omega$, which may arise by evaluating first the left part of the term at hand, and next the right part. Of course, in this example a 'fair' evaluation is possible in only ω many reduction steps, but we do not want to impose fairness requirements at the start—even though we may (and will) consider it to be a desirable feature that reductions of length α could be 'compressed' to reductions of length not exceeding ω steps, yielding the same 'result'.

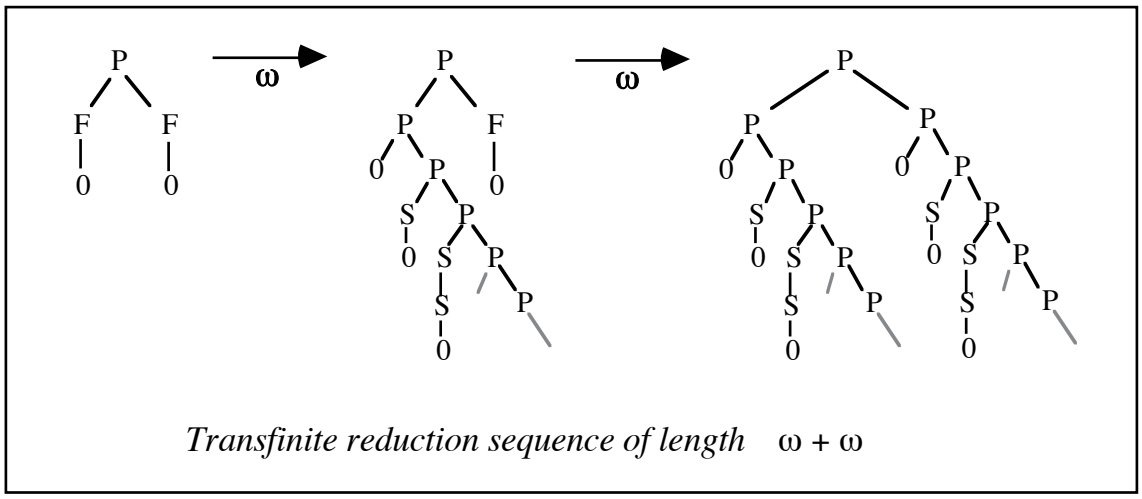


Figure 8.2

We will give a formal definition now.

8.1. DEFINITION. Let (Σ, R) be a TRS. A *(Cauchy-) convergent* R*-reduction sequence of length* α (an ordinal) is a sequence $\langle t_\beta \mid \beta \le \alpha \rangle$ of terms in $\mathrm{Ter}^\infty(\Sigma)$, such that

(i) $t_\beta \to_R t_{\beta+1}$ for all $\beta < \alpha$,

(ii) $t_\lambda = \lim_{\beta<\lambda}$ for every limit ordinal $\lambda \le \alpha$).

Here (ii) means: $\forall n\, \exists \mu < \lambda\, \forall \nu\, (\mu \le \nu \le \lambda \;\Rightarrow\; d(t_\nu, t_\lambda) \le 2^{-n})$.

Notation: If $\langle t_\beta \mid \beta \le \alpha \rangle$ is a Cauchy-convergent reduction sequence we write $t_0 \to_\alpha^c t_\alpha$ ('c' for 'Cauchy').

The notion of normal form as a final result has to be considered next. We simply generalize the old finitary notion of normal form to the present infinitary setting thus: a (possibly infinite) term is a normal form *when it contains no redexes*. The only difference with the finitary case is that here a

redex may be itself an infinite term. But note that a redex is still so by virtue of a finite prefix, that was called the redex pattern—this is so because our rewrite rules are orthogonal and hence contain no repeated variables.

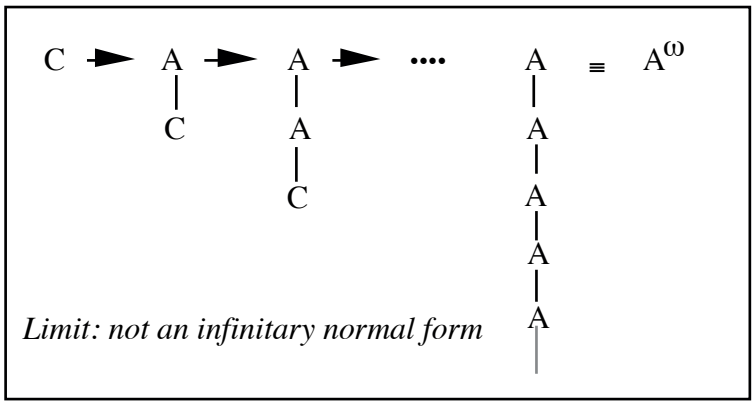


Figure 8.3

So, in Figure 8.3 we have, with as TRS {C → A(C), A(x) → x}, a (Cauchy-) converging reduction sequence with as limit the infinite term A(A(A(A..., abbreviated as A^ω; this limit is not a normal form: A^ω reduces to itself: $A^\omega \to A^\omega$, a nd only to itself. (Note that this step can be performed in infinitely many different ways, since every A in A^ω is the root of a redex.) Normal forms are shown in Figures 8.1, 8.2 as the rightmost terms (if no other reduction rules are present than the one mentioned above). Henceforth we will often drop the reference 'infinite' or 'infinitary'. Thus a term, or a normal form, may be finite or infinite. The notion of Cauchy converging reduction sequence that was considered so far, is not quite satisfactory. We would like to have the *compression property*:

$$t_0 \to_\alpha^c t_\alpha \;\Rightarrow\; t_0 \to_{\leq\omega}^c t_\alpha.$$

That is, given a reduction $t_0 \to_\alpha^c t_\alpha$, of length α, the result t_α can already be found in at most ω many steps. ('At most', since it may happen that a transfinite reduction sequence can be compressed to finite length, but not to length ω.) Unfortunately, $\to_\alpha^c$ lacks this property:

8.2. COUNTEREXAMPLE. Consider the orthogonal TRS with rules {A(x) → A(B(x)), B(x) → E(x)}. Then $A(x) \to_\omega A(B^\omega) \to A(E(B^\omega))$, so $A(x) \to_{\omega+1} A(E(B^\omega))$. However, we do not have $A(x) \to_{\leq\omega} A(E(B^\omega))$, as can easily be verified.

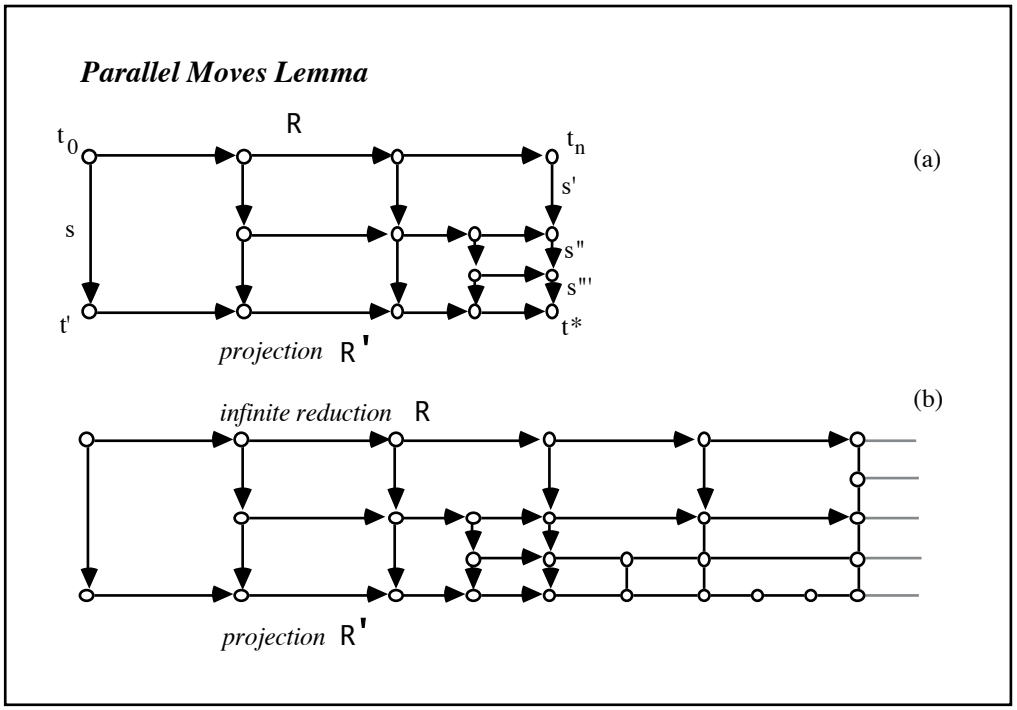


Figure 8.4

Another obstacle for $\rightarrow_\alpha^c$ is that the well-known Parallel Moves Lemma resists a generalization to the present transfinite case. We recall the PML in Figure 8.4(a): setting out a finite reduction R: $t_0 \twoheadrightarrow t_n$ against a one step reduction $t_0 \rightarrow_s t'$ (where s is the contracted redex), one can complete the reduction diagram in a canonical way, thereby obtaining as the righthand side of the diagram a reduction $t_n \twoheadrightarrow t^*$ which consists entirely out of contractions of all the *descendants of* s *along* R. Furthermore, the reduction R': $t' \twoheadrightarrow t^*$ arising as the lower side of this reduction diagram, is called the *projection of* R *over the reduction step* $t_0 \rightarrow_s t'$. Notation: $R' = R/ (t_0 \rightarrow_s t')$.

We would like to have a generalization of PML where R is allowed to be infinite, and converging to a limit. In this way we would have a good stepping stone towards establishing infinitary confluence properties. However, it is not clear at all how such a generalization can be established. The problem is shown in Figure 8.5. First note that we can without problem generalize the notion of 'projection' to infinite reductions, as in Figure 8.4(b): there R' is the projection of the infinite R over the displayed reduction step. This merely requires an iteration of the finitary PML, no infinitary version is needed. Now consider the two rule TRS $\{A(x, y) \rightarrow A(y, x), C \rightarrow D\}$. Let R be the infinite reduction $A(C, C) \rightarrow A(C, C) \rightarrow A(C, C) \rightarrow \ldots$, in fact a reduction cycle of length 1. Note that R is converging, with limit A(C, C). The projection R' of R over the step $A(C, C) \rightarrow A(D, C)$, however, is no longer converging. For, this is $A(D, C) \rightarrow A(C, D) \rightarrow A(D, C) \rightarrow \ldots$, a 'two cycle'. So, the class of infinite converging reduction sequences is not closed under projection. This means that in order to get

some decent properties of infinitary reduction in this sense, one has to impose further restrictions.

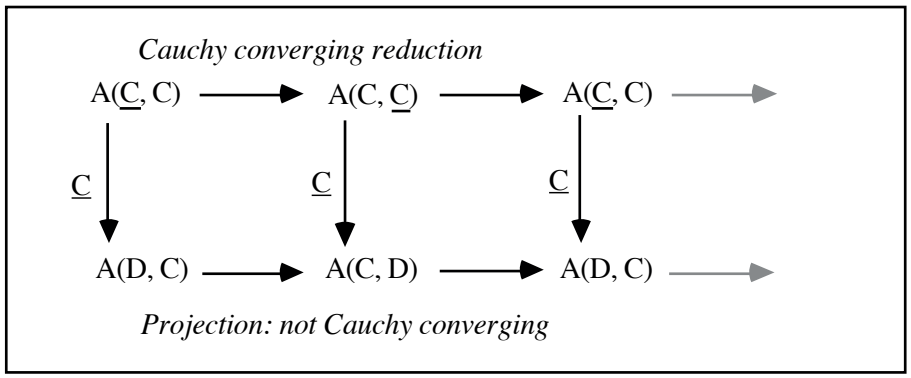


Figure 8.5

As the last example shows, there is a difficulty in that we loose the notion of descendants which is so clear and useful in finite reductions. Indeed, after the infinite reduction A(C, C) → A(C, C) → A(C, C) → ... , with Cauchy limit A(C, C), what is the descendant of the original underlined redex C in the limit A(C, C)? There is no likely candidate.

We will now describe the stronger notion of converging reduction sequence that does preserve the notion of descendants in limits. If we have a converging reduction sequence $t_0 \to_{s0} t_1 \to_{s1} \ldots t$, where s_i is the redex contracted in the step $t_i \to t_{i+1}$ and t is the limit, we now moreover require that

$$\lim_{i\to\infty} \text{depth}(s_i) = \infty. \qquad (*)$$

Here depth(s_i), the depth of redex s_i, is the distance of the root of t_i to the root of the subterm s_i. If the converging reduction sequence satisfies this additional requirement (*), it is called *strongly convergent*. The difference between the previous notion of (Cauchy) converging reduction sequence and the present one, is suggested by Figure 8.6. The circles in that figure indicate the root nodes of the contracted redexes; the shaded part is that prefix part of the term that does not change anymore in the sequel of the reduction. The point of the additional requirement (*) is that this growing non-changing prefix is required really to be non-changing, in the sense that no activity (redex contractions) in it may occur at all, even when this activity would by accident yield the same prefix.

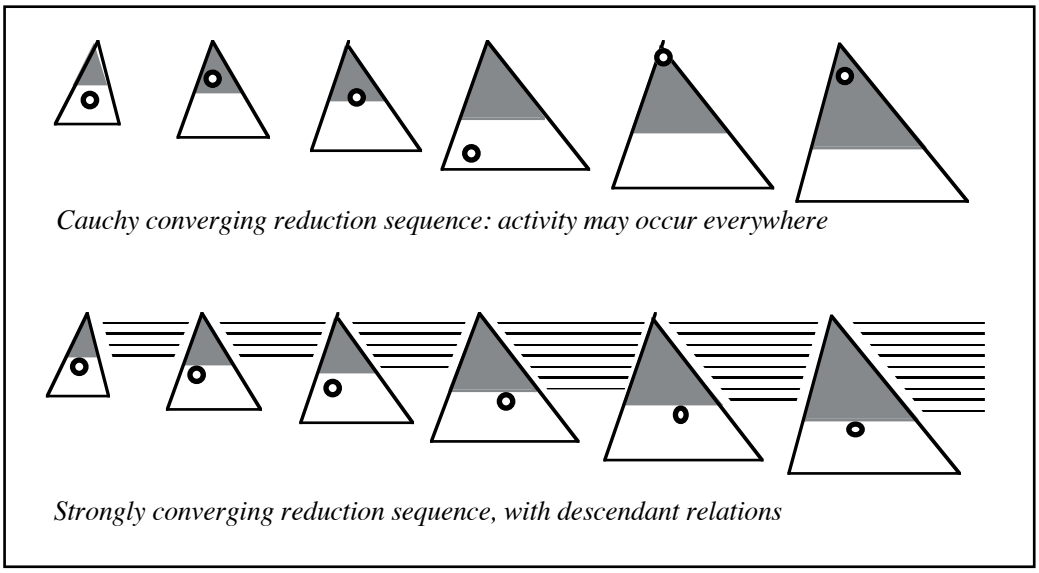


Figure 8.6

Note that there is now an obvious definition of descendants in the limit terms; the precise formulation is not hard to make explicit.

In fact, we define strongly converging reductions of length α for every ordinal α, by imposing the additional condition (*) whenever a limit ordinal $\lambda \leq \alpha$ is encountered. (It will turn out however that only countable ordinals may occur.) More formally:

8.3. DEFINITION. Let (Σ, R) be a TRS. A *strongly convergent* R*-reduction sequence of length* α is a sequence $\langle t_\beta \mid \beta \leq \alpha \rangle$ of terms in $Ter^\infty(\Sigma)$, such that

(i) $t_\beta \to_R t_{\beta+1}$ for all $\beta < \alpha$,

(ii) for every limit ordinal $\lambda \leq \alpha$: $\forall n\, \exists \mu < \lambda\, \forall \nu\ (\mu \leq \nu \leq \lambda \Rightarrow d(t_\nu, t_\lambda) \leq 2^{-n}$ & $\text{depth}(s_\nu) \geq n)$.

Here s_ν is the redex contracted in the step $t_\nu \to t_{\nu+1}$. (See Fig. 8.7.)

Notation: If $\langle t_\beta \mid \beta \leq \alpha \rangle$ is a strongly convergent reduction sequence we write $t_0 \to_\alpha t_\alpha$.

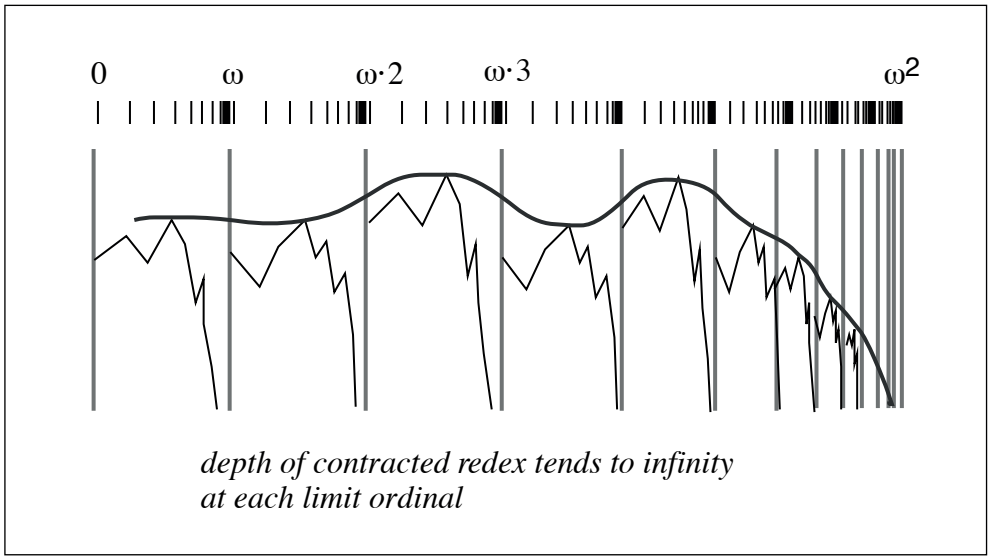


Figure 8.7

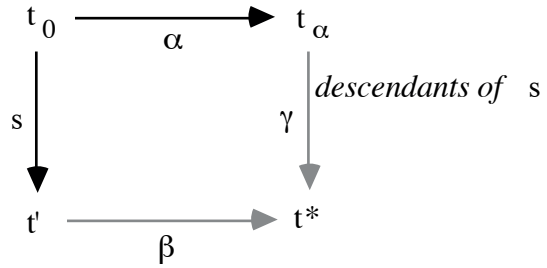
Henceforth all our infinitary reductions will be strongly convergent. Now we can state the benefits of this notion.

8.4. COMPRESSION LEMMA. *In every orthogonal TRS:*

$$t \to_\alpha t' \;\Rightarrow\; t \to_{\leq\omega} t'.$$

(Note that the counterexample 8.2 to compression for Cauchy converging reductions was not strongly converging.)

8.5. INFINITARY PARALLEL MOVES LEMMA. *In every orthogonal TRS:*



That is, whenever $t_0 \to_\alpha t_\alpha$ *and* $t_0 \to_s t'$, *where* s *is the contracted redex (occurrence), there are infinitary reductions* $t' \to_\beta t^*$ *and* $t_\alpha \to_\gamma t^*$. *The latter reduction consists of contractions of all descendants of* s *along the reduction* $t_0 \to_\alpha t_\alpha$.

Actually, by the Compression Lemma we can find $\beta, \gamma \leq \omega$.

As a side-remark, let us mention that in every TRS (even with uncountably many symbols and

rules), all transfinite reductions have countable length. All countable ordinals can indeed occur as length of a strongly convergent reduction. (For ordinary Cauchy convergent reductions this is not so: the rewrite rule $C \to C$ yields arbitrarily long convergent reductions $C \to_\alpha^c C$. However, these are not strongly convergent.)

The infinitary PML is "half of the infinitary confluence property". The question arises whether full infinitary confluence (CR^∞)holds. That is, given $t_0 \to_\alpha t_1$, $t_0 \to_\beta t_2$, is there a t_3 such that $t_1 \to_\gamma t_3$, $t_2 \to_\delta t_3$ for some γ, δ? Using the Compression Lemma and the PML all that remains to prove is: given $t_0 \to_\omega t_1$, $t_0 \to_\omega t_2$, is there a t_3 such that $t_1 \to_{\leq\omega} t_3$, $t_2 \to_{\leq\omega} t_3$? Surprisingly, the answer is negative: *full infinitary confluence for orthogonal rewriting does not hold.* The counterexample is in Figure 8.8, consisting of an orthogonal TRS with three rules, two of which are 'collapsing rules'. (A rule $t \to s$ is collapsing if s is a variable.) Indeed, in Figure 8.8(a) we have $C \to_\omega A^\omega$, $C \to_\omega B^\omega$ but A^ω, B^ω have no common reduct as they only reduce to themselves. Note that these reductions are indeed strongly convergent. (Figure 8.8(b) contains a rearrangement of these reductions that we need later on.)

However, the good news is that in spite of the failure of CR^∞ we do have unicity of (possibly infinite) normal forms (UN^∞).

8.6. THEOREM. *For all orthogonal TRSs: Let* $t \to_\alpha t'$, $t \to_\beta t''$ *where* t', t'' *are (possibly infinite) normal forms. Then* $t' \equiv t''$.

Here $\equiv$ denotes syntactical equality. Note that in the ABC counterexample in Figure 8.8 the terms A^ω and B^ω are not normal forms.

This Unique Normal Form property, by the way, also holds for Cauchy converging reductions, that is, with $\to_\alpha$ replaced by $\to_\alpha^c$ and likewise for β. The reason is that we have:

$$t \to_\alpha^c t' \;\&\; t' \textit{ is a normal form} \;\Rightarrow\; t \to_{\leq\omega} t'.$$

(For $\alpha = \omega$ this is easy to prove; in fact a converging reduction of length ω to a normal form is already strongly convergent. For general α, the proof of the statement requires some work.)

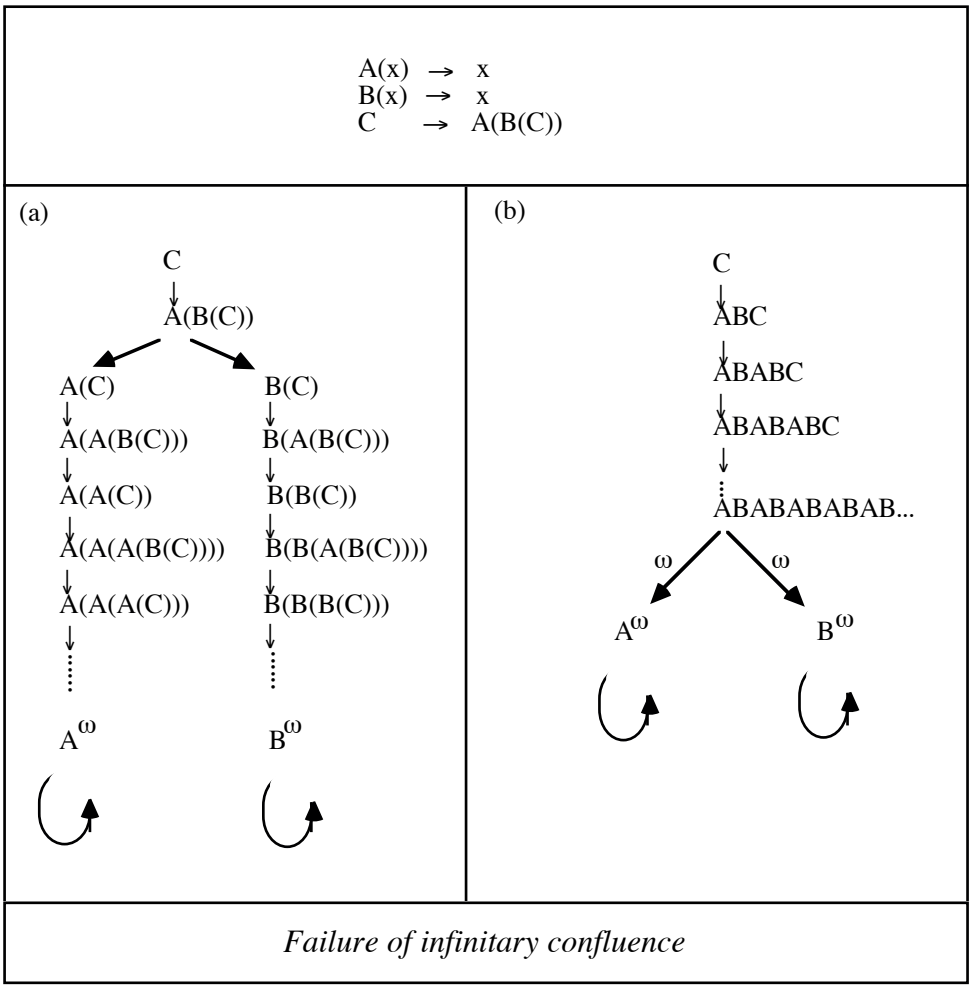


Figure 8.8

We will now investigate the extent to which infinitary orthogonal rewriting lacks full confluence. It will turn out that non-confluence is only marginal, and that terms which display the bad behaviour are included in a very restricted class. The following definition is inspired by a classical notion in λ-calculus; see Barendregt [84].

8.7. DEFINITION. (i) The term t is in *head normal form* (hnf) if $t \equiv C[t_1,...,t_n]$ where $C[t_1,...,t_n]$ is a non-empty context (prefix) such that no reduction of t can affect the prefix C[,...,]. More precisely, if $t \ddot{y} s$ then $s \equiv C[s_1,...,s_n]$ for some s_i (i =1,...,n), and every redex of s is included in one of the s_i (i =1,...,n).

(ii) t *has a hnf* if $t \twoheadrightarrow s$ and s is in hnf.

Actually, this definition is equivalent to one of DKP[89]; there a term t is called 'top-terminating' if there is no infinite reduction $t \to t' \to t'' \to ...$ in which infinitely many times a redex contraction *at the root* takes place. So: t is top-terminating $\Leftrightarrow$ t has a hnf. We need one more

definition before formulating the next theorem.

8.8. DEFINITION. If t is a term of the TRS R, then the *family* of t is the set of subterms of reducts of t, i.e. $\{s \mid t \twoheadrightarrow_R C[s] \text{ for some context } C[\,]\}$.

8.9. THEOREM. *For all orthogonal TRSs: Let* t *have no term without hnf in its family. Then* t *is infinitary confluent.*

Here we want to reconsider the last theorem. Actually, it can be much improved. Consider again the ABC example in Figure 8.8. Rearranging the reductions $C \to_\omega A^\omega$, $C \to_\omega B^\omega$ as in Figure 8.8(b) into reductions $C \to_\omega (AB)^\omega \to_\omega A^\omega$ and $C \to_\omega (AB)^\omega \to_\omega B^\omega$ makes it more perspicuous what is going on: $(AB)^\omega$ is an infinite 'tower' built from two different collapsing contexts A( ), B( ), and this infinite tower can be collapsed in different ways.

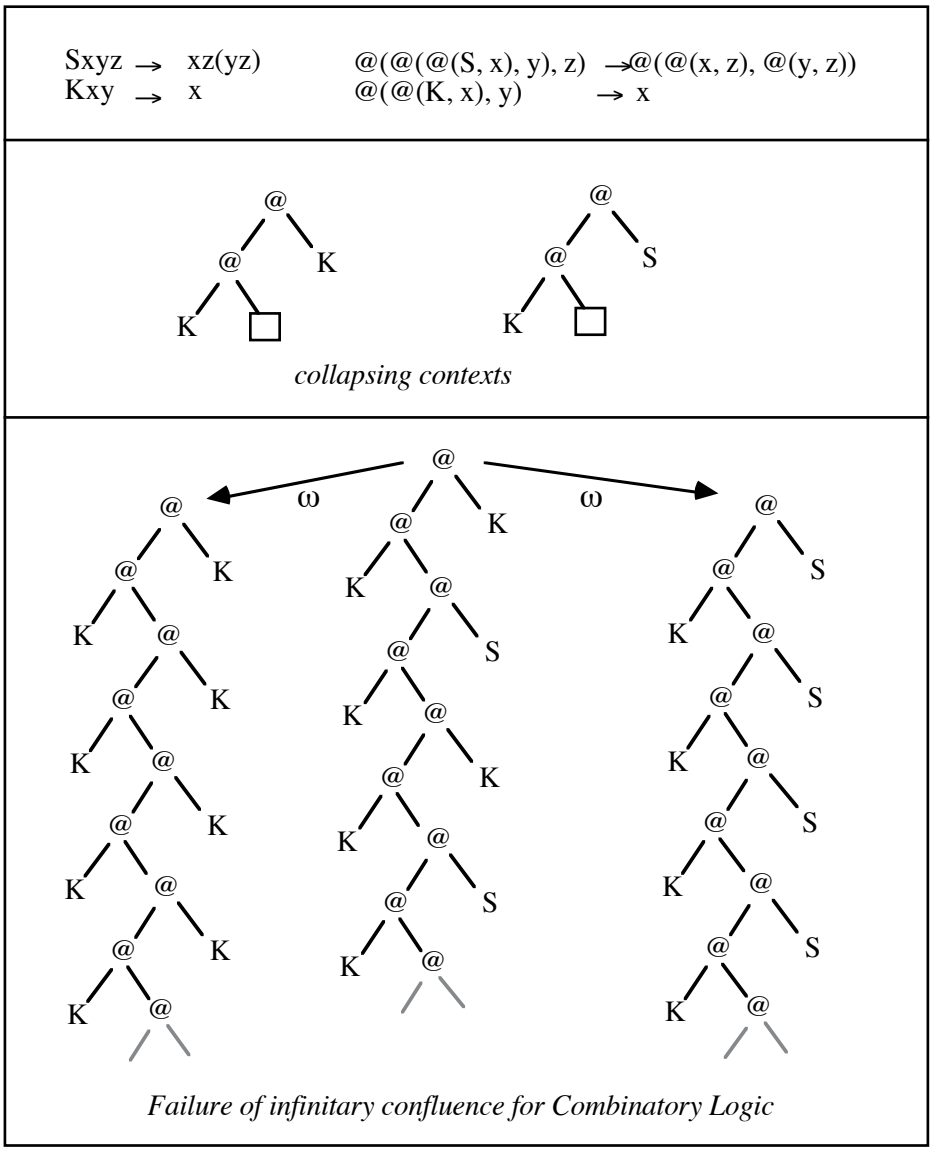


Figure 8.9

The ABC example (Figure 8.8) is not merely a pathological example; the same phenomenon (and therefore failure of infinitary confluence) occurs in Combinatory Logic, as in Figure 8.9, where an infinite tower built from the two different collapsing contexts K□K and K□S is able to collapse in two different ways. (Note that analogous to the situation in Figure 8.8, the middle term, built alternatingly from K□K and K□S, can be obtained after ω steps from a finite term which can easily be found by a fixed point construction.) Also for λ-calculus one can now easily construct a counterexample to infinitary confluence.

Remarkably, it turns out that the collapsing phenomenon is the *only* cause of failure of infinitary confluence. Thus we have:

8.10. THEOREM. (i) *Let the orthogonal TRS* R *have no collapsing rewrite rules* $t(x_1,...,x_n) \to x_i$. *Then* R *is infinitary confluent.*
(ii) *If* R *is an orthogonal TRS with as only collapsing rule:* $I(x) \to x$, *then* R *is infinitary confluent.*

Call an infinite term $C_1[C_2[...C_n[...]...]]$, built from infinitely many non-empty collapsing contexts $C_i[\ ]$, a *hereditarily collapsing* (hc) term. (A context C[ ] is collapsing if C[ ] contains one hole [] and $C[\ ] \twoheadrightarrow []$.) Also a term reducing to a hc term is called a hc term. E.g. C from the ABC example in Figure 8.8 is a hc term. Clearly, hc terms do not have a hnf.

8.11. THEOREM. *Let* t *be a term in an orthogonal TRS, which has not a hc term in its family. Then* t *is infinitary confluent.*

This theorem can be sharpened somewhat, as follows. Let us introduce a new symbol ● (*black hole*) to denote hc terms, with the rewrite rule:

$t \to \bullet$ ● *if* t *is a hc term.*

Of course this rule is not 'constructive', i.e. the reduction relation $\to_\bullet$ may be undecidable (as it is in CL, Combinatory Logic). However, we now have that orthogonal reduction extended with $\to_\bullet$ is infinitary confluent.

Appendix: The property UN$^{\infty}$ for first order orthogonal infinitary rewriting

Consider two infinite reductions

$$\mathsf{R}: M_0 \to M_1 \to \ldots \to^{\omega} M_{\omega},$$

$$\mathsf{R}': M_0 \to M'_1 \to \ldots \to^{\omega} M'_{\omega},$$

of a term M_0 to two infinite normal forms M_ω and M'_ω. We wish to prove that $M_\omega \equiv M'_\omega$, by showing that their finite approximations (prefixes) coincide:

$$\forall n\ M_\omega/n \equiv M'_\omega/n.$$

So consider M_ω/n and M'_ω/n. By definition of the limit notion in infinitary rewriting, we know that in reduction R the 'action' is after some stage N deeper than n, i.e. $\forall k > N\ d_k > n$, where d_k is the depth of the redex r_k contracted in the step $M_k \to M_{k+1}$. Likewise in reduction R': after some N' all action is deeper than n. So we know that in the reduction $M_N \to \ldots \to^{\omega} M_\omega$ the prefix M_ω/n is 'untouched', and likewise M'_ω/n is untouched in the reduction $M'_{N'} \to \ldots \to^{\omega} M'_\omega$. Now consider the initial parts of R and R', up to N and N' respectively:

$$M_0 \to M_1 \to \ldots \to M_N,$$
$$M_0 \to M_1 \to \ldots \to M'_{N'},$$

and construct the reduction diagram determined by these two finite reductions, see Figure xx. Let M* be the common reduct of M_N and $M'_{N'}$ thus found. If we could now assume that the prefix M_ω/n also remained untouched in the reduction

$$M_N \to \ldots \to M^*,$$

and likewise for M'_ω/n in

$$M_N \to \ldots \to M^*,$$

we would have $M_\omega/n \equiv M'_\omega/n$ as desired. But unfortunately we cannot assume that. A priori, the action may go 'upwards', during the reduction $M_N \to \ldots \to M^*$ in M_N. In fact it will not, by the assumption of orthogonality, as we will prove.

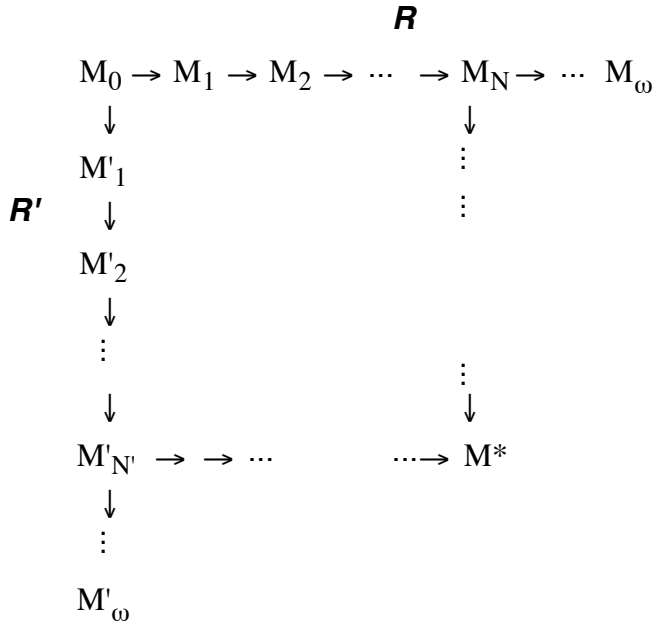


FIG. xx. Normal form evaluations.

DEFINITION xx. Prefix Π is *stable with respect to reduction* R if not only no reduction takes place in Π, but there is not even a redex in Π activated ('triggered') during R.

PROPOSITION xx. *Let* $M_0 \in Ter^{\infty}(R)$ *have prefix* Π, *and let* Π *be stable with respect to the infinite normalizing reduction* R: $M_0 \to M_1 \to ... \to^{\omega} M_{\omega}$, *a normal form. Then* Π *is stable with respect to any reduction.*

Proof. If the statement does not hold, there must be a symbol F in the prefix Π, such that the subterm headed by F is stable with respect to the normalizing reduction R, but such that F is triggered as the head of a redex in another reduction R'. So, without loss of generality, we can assume that $M_0 \equiv F(t_1,...,t_n)$:

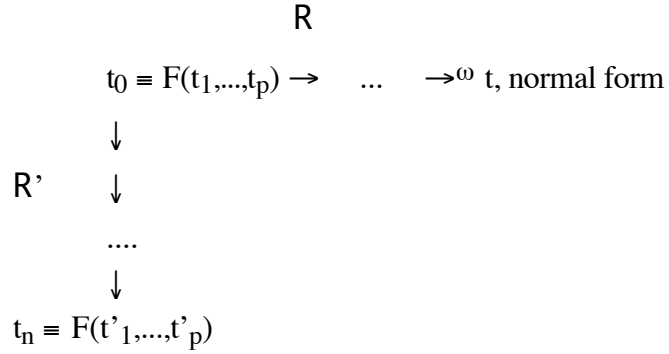


FIG. xx. Proof of Proposition xx

.

where in R, the F (or rather the context F(Ω,..., Ω)) is stable, but in R' the F has become the head of a redex $F(t'_1,...,t'_p)$. Now we invoke the Parallel Moves Lemma for infinitary orthogonal rewriting, PML$^\infty$, and construct the projection of R_0 along the first step of R': result R_1.

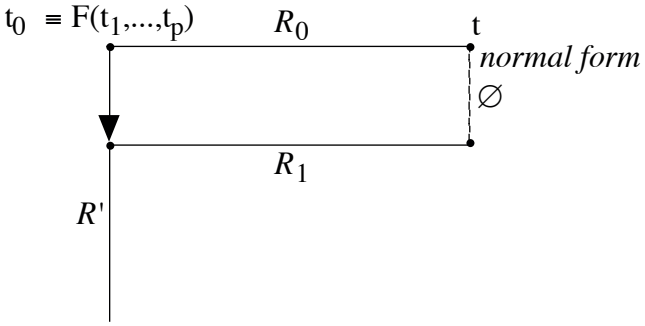


FIG. xx. Using PML$^\infty$.

The right-hand side of the strip determined by $t_0 \to t_1$ and R_0, is ∅, the empty reduction, by PML$^\infty$ (since there are no residuals of the redex contracted in $t_0 \to t_1$ present in t, a normal form). Now in R_0 there was no step at the root, by assumption. It follows that the same is true in R_1, by elementary reasoning with residuals in orthogonal reduction diagrams. Let us look at this fact somewhat closer:

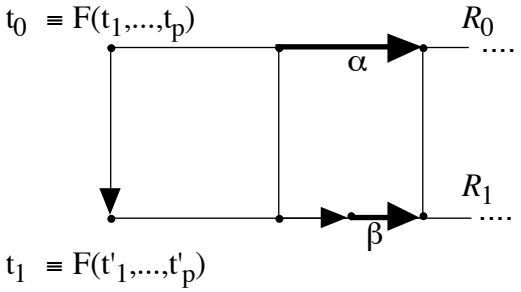


FIG. xx. A closer look.

Paint the head F in the initial term $F(t_1,...,t_p)$ blue, all other symbol occurrences (including other F-occurrences) a different color. Call a redex blue, red, ... if that is the color of its head. So in R_0 no blue redex is contracted. Now in orthogonal rewriting, a step α contracts a redex with the same color as the redex in step β. So, R_1 cannot contract a blue redex; i.e. the F heading $F(t'_1,...,t'_p)$ is not

triggered in R_1.

We now iterate this argument, projecting R_1 over the second step of R', etc. Thus we arrive at R_n:

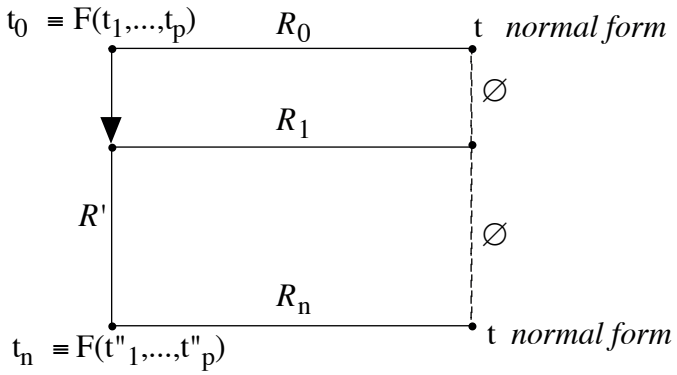


FIG. xx. Iterating PML$^\infty$.

Now R_n starts with the term $t_n \equiv F(t'_1,...,t'_p)$, by assumption a redex, and proceeds to the normal form t, without ever contracting a root redex—i.e. the redex headed by F. But then, that root redex is still present in the normal form t, a contradiction.
{use terminology of inner reduction}
This proves the Proposition and thereby also the theorem UN$^\infty$ for orthogonal TRSs. z

Remark. Note that the proof of UN$^\infty$ relied heavily on the properties of orthogonal rewriting. The question arises whether orthogonality is *necessary* forUN$^\infty$; would mere confluence (finitary CR) not be enough? In fact we can answer this question CR $\Rightarrow$ UN$^\infty$ negatively, with the following counterexample.
Consider the TRS R with the three rules:

$C \to A(C)$
$C \to B$
$A(B) \to B$.

So R is not orthogonal. We have the following reductions:

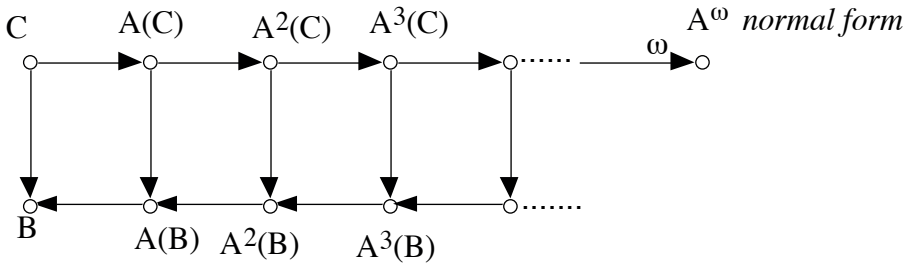


FIG. xx. Counterexample to CR ⇒ UN$^{\infty}$.

These are all the reducts of C. There are two normal forms, A^{ω} and B. Hence UN$^{\infty}$ does not hold. All terms of Ter(R) are displayed; clearly, R is CR. R is not CR$^{\infty}$, though.

9.1. OPGAVEN.

OPGAVE 9.1.1. We geven nu een voorbeeld van een mooie TRS die de rij van priemgetallen 2,3,5,7,11,13,... berekent als oneindige normaalvorm.

filter(x:y, 0, m) ↠ 0:*filter*(y, m, m)
filter(x:y, S(n), m) ↠ x:*filter*(y, n,m)
sieve(0:y) ↠ *sieve*(y)
sieve(S(n):y) ↠ S(n):*sieve*(*filter*(y, n, n))
nats(n) ↠ n:*nats*(S(n))
primes ↠ *sieve*(*nats*(S(S(0)))

OPGAVE 9.1.2. Reduceer primes tot de eerste priemgetallen zichtbaar worden.

OPGAVE 9.1.3. Definiëer met een TRS een operator 'search' die als input streams natuurlijke getallen krijgt, en als output geeft: het nummer van de plaats in de rij waar voor het eerst een 0 optreedt, als die er is. Dus bijvoorbeeld search (7,6,3,8,0,7,7,7,...) = 5.

OPGAVE 9.1.4. Geef een TRS met een operator square die de elementen van een stream natuurlijke getallen kwadrateert. Dus bijvoorbeeld square (2,3,5,8,...) = 4,9,25,64,...

OPGAVE 9.1.5. Geef een TRS met een binaire operator plus die van twee streams

natuurlijke getallen de elementen coördinaatsgewijs optelt. Dus bijvoorbeeld plus ((1,2,3,6....), (3,3,8,9....) = 4,5,11,15,...

OPGAVE 9.1.6. Ontwerp een orthogonaleTRS met zelf gekozen signatuur die de rij van Fibonacci-getallen 1,1,2,3,5,8,13,... genereert.

OPGAVE 9.1.7.Completeer de TRS met de drie regels

$a(b(c(x))) \twoheadrightarrow x$
$a(b(x)) \twoheadrightarrow x$
$b(c(x)) \twoheadrightarrow x$

OPGAVE 9.1.8.
Ga bij elk van de volgende ARS-en na of geldt:
SN, WN, CR, WCR, UN, en geef de normaalvormen.

(1). De ARS (N, →) met N de verzameling natuurlijke getallen en
waarbij → gedefinieerd is door:

n → m als m een echte deler is van n
(d.w.z. er is een k zodanig dat (k*m = n)) en m < n en m ≠ 1.)

(2). Idem met → als volgt:

n → m als m een veelvoud is van n (dus er is een k zdd n*k = m).

(3). De ARS (N, <) (met < de gewone ordening).

(4). De ARS (N, >).

OPGAVE 9.1.9.
Gegeven zijn braids 3312 en 2133. Vind voortzettingen .... en ---- zodat 3312.... = 2133---- .

OPGAVE 9.1.10.
Voorbeeld van 'functioneel programmeren' met TRSen: maximum van twee natuurlijke getallen.

$max(x,y) \rightarrow aux(x,y,0)$

aux(S(x), S(y), z) → aux(x, y, S(z))
aux(S(x), 0, y) → S(x) + y
aux(0, S(x), y) → S(x) + y
aux(0,0,y) → y

(Opm. Gemakshalve is hier S(x) + y geschreven; eigenlijk moeten
we schrijven (A(S(x), y) met de regels van de AMS0 TRS voor A er nog bij.)
Geef ook een TRS voor het minimum van twee natuurlijke getallen.

OPGAVE 9.1.11. Berry's TRS heeft de volgende regels:

F(A,B,x) → 0
F(x,A,B) → 1
F(B,x,A) → 2

Bewijs dat deze TRS orthogonaal is.

OPGAVE 9.1.12.
Gegeven is de ARS (A, $\to_1$, $\to_2$).

DEFINITIE: (i) $\to_2$ heet een *verfijning* van $\to_1$ als $\forall a,b$
($a \to_1 b \Rightarrow a \twoheadrightarrow_2 b$.
(ii) $\to_2$ heet een *mooie verfijning* van $\to_1$ als

$\forall a,b\ \exists c\ (a \twoheadrightarrow_2 b \Rightarrow a \twoheadrightarrow_1 c <<-_1 b)$

Zij $\to_2$ een mooie verfijning van $\to_1$.
Bewijs dat $\to_1$ is CR $\Leftrightarrow$ $\to_2$ is CR.

[Hint: Bewijs eerst dat $\forall a,b,c\ \exists d$
($a \twoheadrightarrow_2 b \twoheadrightarrow_1 c \Rightarrow a \twoheadrightarrow_1 d <<-_1 c$)]

[*Waarschuwing*: het bewijs $\to_2$ is CR $\Rightarrow$ $\to_1$ is CR is lastig, je loopt gauw in een dead end, dan even 'back tracken' in je bewijspoging. Het bewijs is niet lang. Teken vooral de figuren telkens!]

OPGAVE 9.1.13.

Zij R een TRS met de eigenschap WN. Een *innermost* redex is er een die geen redex als subterm bevat. Definieer $\rightarrow_{in}$ s als s uit t ontstaat dooreen willekeurig innermost redex r te kiezen, en dit te vervangen dooreen normaalvorm van r. Bewijs dat de reductierelatie $\rightarrow_{in}$ SN is.

In the next exercise we will construct a non-terminating term *ham*, which rewrites to any finite initial part of the sequence of Hamming numbers. (For an introduction to Hamming numbers see Dijkstra [1976].)

3.1.2. EXERCISE. The Hamming numbers are all natural numbers that can be written in the form $2^i \cdot 3^j \cdot 5^k$, for some $i, j, k \geq 0$. The aim of this exercise is to construct a TRS generating, as a stream, the sequence of Hamming numbers in ascending order. So this sequence starts with

$$1\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 10\ 12\ 15\ \ldots$$

One may proceed in three steps.

(i) Construct a TRS computing the maximum of two natural numbers.

10
STRATEGIEEN

Terms in a TRS may have a normal form as well as admitting infinite reductions. So, if we are interested in finding normal forms, we should have some strategy at our disposal telling us what redex to contract in order to achieve that desired result. We will in this section present some strategies which are guaranteed to find the normal form of a term whenever such a normal form exists. We will adopt the restriction to orthogonal TRSs. The strategies below will be of two kinds: one step or 'sequential' strategies (which point in each reduction step to just one redex as the one to contract) and many step or 'parallel' strategies (in which a set of redexes is contracted simultaneously). Of course all strategies must be computable.

Apart from the objective of finding a normal form, we will consider the objective of finding a 'best possible' reduction even if the term at hand does not have a normal form.

10.1. DEFINITION. Let R be a TRS.

(i) A *one-step reduction strategy* $\mathbb{F}$ *for* R is a map $\mathbb{F}$: Ter(R) $\to$ Ter(R) such that

(1) $t \equiv \mathbb{F}(t)$ if t is a normal form,

(2) $t \to \mathbb{F}(t)$ else.

(ii) A *many-step reduction strategy* $\mathbb{F}$ *for* R is a map $\mathbb{F}$: Ter(R) $\to$ Ter(R) such that

(1) $t \equiv \mathbb{F}(t)$ if t is a normal form,

(2) $t \to^+ \mathbb{F}(t)$ else.

Here $\to^+$ is the transitive (but not reflexive) closure of $\to$.

10.2. DEFINITION. (i) A reduction strategy (one step or many step) $\mathbb{F}$ for R is *normalizing* if for each term t in R having a normal form, the sequence $\{\mathbb{F}^n(t) \mid n \geq 0\}$ contains a normal form.

(ii) $\mathbb{F}$ is *cofinal* if for each t the sequence $\{\mathbb{F}^n(t) \mid n \geq 0\}$ is cofinal in $\mathcal{G}(t)$, the reduction graph of t. This means that from every term in $\mathcal{G}(t)$ there is a reduction to some $\mathbb{F}^n(t)$. (See Figure 10.1.)

Trivially we have:

10.3. PROPOSITION. *Cofinal strategies are normalizing.* □

10.4. REMARK.

A normalizing reduction strategy is good, but a cofinal one is even better: it finds, when applied on term t, the best possible reduction sequence starting from t (or rather, a best possible) in the following sense. Consider a reduction $t \to s$ as a gain in information; thus normal forms have maximum information. In case there is no normal form in $\mathcal{G}(t)$, one can still consider infinite reductions as developing more and more information. Now the cofinal reductions $t \equiv t_0 \to t_1 \to t_2 \to$ ... are optimal since for every t' in $\mathcal{G}(t)$ they contain a t_n with information content no less than that of t' (since $t' \twoheadrightarrow t_n$ for some t_n, by definition of 'cofinal').

.

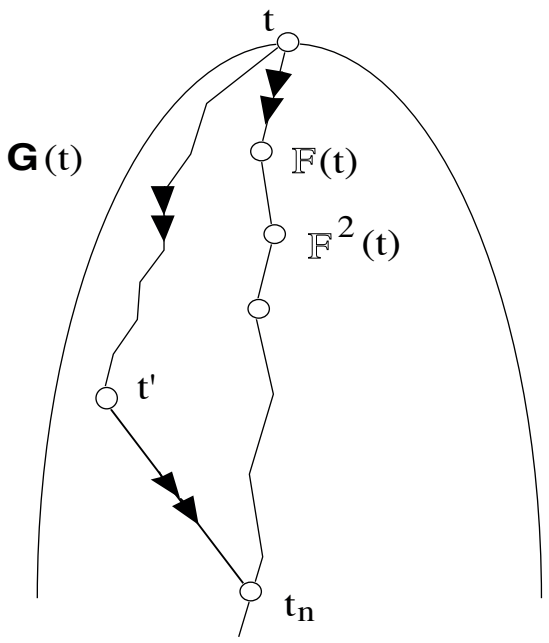


Figure 7.1

We now present the most usual reduction strategies. We illustrate the definitions with an example in the TRS in Table 10.1.

and(true, x) → x
and(false, x) → false
or(true, x) → true
or(false, x) → x

Table 10.1

10.3. DEFINITION. (i) The *leftmost-innermost* (one step) strategy is the strategy in which in each step the leftmost of the minimal or innermost redexes is contracted.

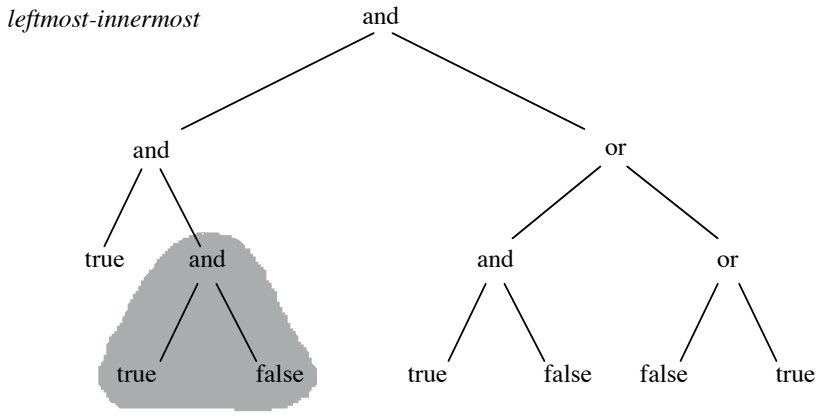


Figure 10.2

(ii) The *parallel-innermost* (many step) strategy reduces simultaneously all innermost redexes. Since these are pairwise disjoint, this is no problem.

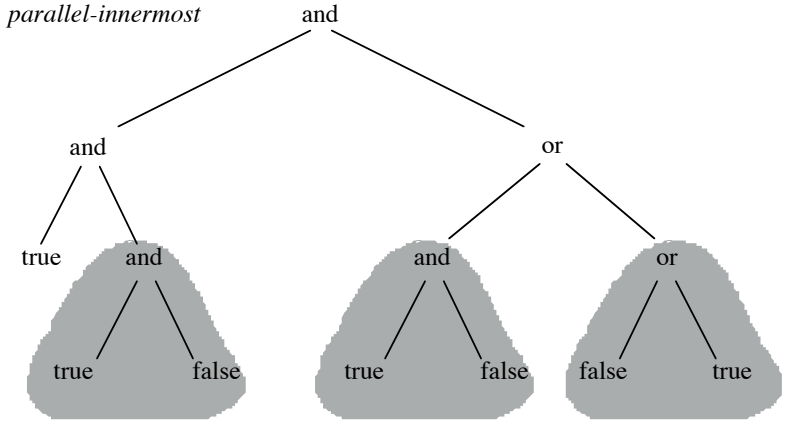


Figure 10.3

(iii) The *leftmost-outermost* (one step) strategy: in each step the leftmost redex of the maximal (or outermost) redexes is reduced. Notation: $\mathbb{F}_{\mathrm{lm}}$.

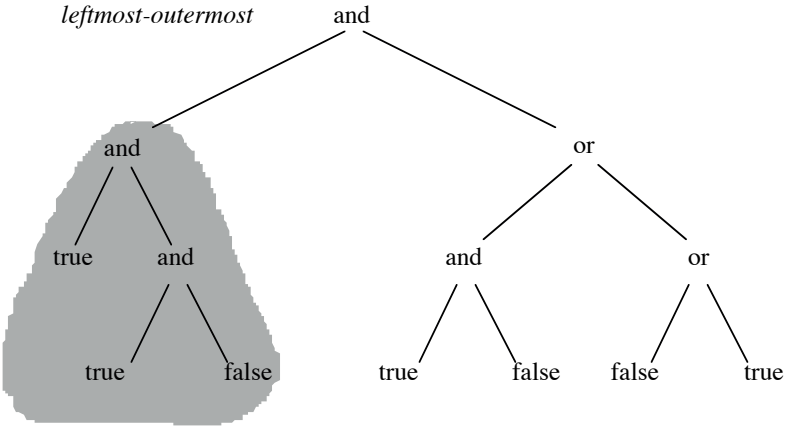


Figure 10.4

(iv) The *parallel-outermost* (many step) strategy reduces simultaneously all maximal redexes; since these are pairwise disjoint, this is no problem. Notation: $\mathbb{F}_{po}$.

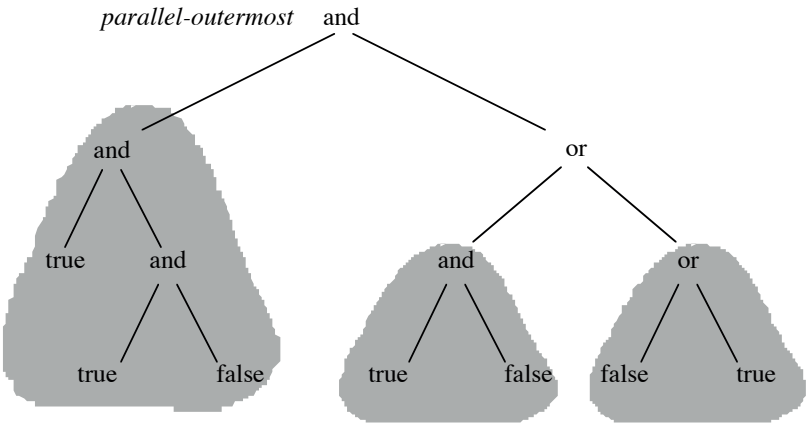


Figure 10.5

(v) The *full substitution rule* (or *Kleene reduction*, or *Gross-Knuth reduction*): this is a many step strategy in which all redexes are simultaneously reduced. Notation: $\mathbb{F}_{GK}$. More precisely: $\mathbb{F}_{GK}(t)$ is the result of the *complete development* (not yet explained) of the set of all redexes in t.

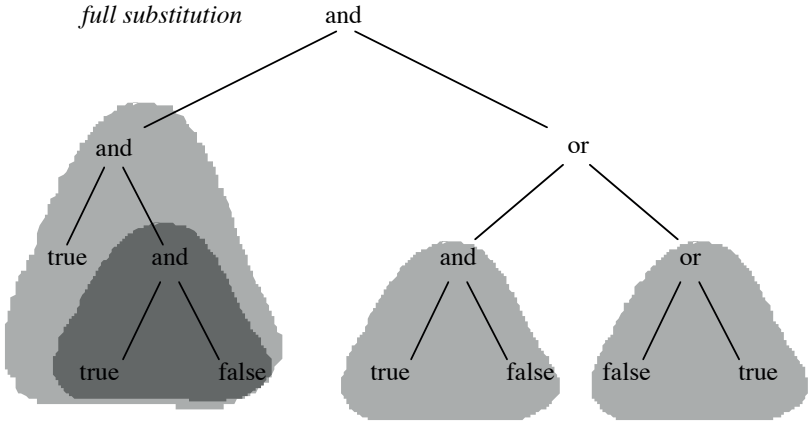


Figure 10.6

Strategies (i)-(iv) are well-defined for general TRSs. Strategy (v) is only defined for orthogonal TRSs, since for a general TRS it is not possible to define an unequivocal result of simultaneous reduction of a set of possibly nested redexes.

We will be mainly interested here in the strategies (iii)-(v), for a reason that will be clear by inspection of Table 10.2 below.

THEOREM. $\mathbb{F}_{po}$ *is a normalizing reduction strategy.*

10.5. REMARK. For λ-calculus this theorem also holds. Moreover, $\mathbb{F}_{lm}$ is there also a normalizing strategy, just as it is for the orthogonal TRS CL (Combinatory Logic). However, in general $\mathbb{F}_{lm}$ is not a normalizing strategy for orthogonal TRSs:

10.6. EXAMPLE.
(i) An example showing that the leftmost-outermost strategy is not normalizing in general: take the orthogonal TRS $\{F(x,B) \to D,\ A \to B,\ C \to C\}$ and consider the term F(C, A). This term has a normal form which is not found by the leftmost-outermost strategy.
(ii) An example showing that parallel-outermost reduction need not be cofinal, can be found in CL. Namely, define the term I_t as SKt, and check that $I_t x \twoheadrightarrow x$. Furthermore, define the term Ω_t as $SI_tI_t(SI_tI_t)$. Now the parallel-outermost strategy, applied on Ω_{II}, yields a cyclic reduction sequence $\Omega_{II} \twoheadrightarrow \Omega_{II}$ which is not cofinal since $\Omega_{II} \twoheadrightarrow \Omega_I$ but not $\Omega_I \twoheadrightarrow \Omega_{II}$.

Even though $\mathbb{F}_{lm}$ is in general for orthogonal TRSs not normalizing, there is a large class of orthogonal TRSs for which it is:

10.6. DEFINITION. An orthogonal TRS is *left-normal* if in every reduction rule $t \to s$ the constant and function symbols in the left-hand side t precede (in the linear term notation) the variables.

10.7. EXAMPLE. (i) CL (Combinatory Logic) is left-normal.
(ii) $F(x, B) \to D$ is not left-normal; $F(B, x) \to D$ is left-normal.

THEOREM. $\mathbb{F}_{lm}$ *is normalizing for left-normal orthogonal TRSs.*
Relaxing the constraints in $\mathbb{F}_{lm}$, $\mathbb{F}_{GK}$ *and* $\mathbb{F}_{po}$.

In the reduction strategy $\mathbb{F}_{GK}$ (full substitution) every redex is 'killed' *as soon as it arises*, and this repeatedly. Suppose we relax this requirement, and allow ourselves some time (i.e. some number of reduction steps) before getting rid of a particular redex—but with the obligation to deal with it *eventually*. The reductions arising in this way are all cofinal.

10.9. DEFINITION. (i) Let $\rho = t_0 \to t_1 \to \ldots$ be a finite or infinite reduction sequence. Consider some redex s in some term t_n of ρ. We say that s is *secured in* ρ if eventually there are no descendants of s left, i.e.

$$\exists m>n\ (t_m \text{ contains no descendants } s', s'', \ldots \text{ of } s \subseteq t_n).$$

(ii) ρ is *fair* if every redex in ρ is secured.

REMARK. Note that we have the following simple fact. Let σ: t ÿ t' be a finite reduction, and ρ: t' $\to$ t'' $\to$ ... be an infinite reduction starting where σ left off. Then the concatenation is $\sigma.\rho$ is fair iff is ρ fair.

10.10. THEOREM . *For reductions* ρ *in orthogonal TRSs:* ρ *is fair* $\Rightarrow$ ρ *is cofinal.*

PROOF. For finite fair reductions the theorem is trivial. So consider the infinite fair reduction

$$\rho: t_0 \equiv s_0 \twoheadrightarrow s_1 \twoheadrightarrow s_2 \twoheadrightarrow \ldots$$

and the reduction $t_0 \to t_1 \to \ldots \to t_n$. (See Figure xx.) Let r_i be the redex contracted in the step $t_i \to t_{i+1}$ for i = 0,...,n-1. Because ρ is fair and by the Parallel Moves Lemma, the step $t_0 \to t_1$ is absorbed by ρ (that is, "pushing $t_0 \to t_1$ through" ρ yields eventually the empty reduction $\varnothing$). Let this be the case at the term $^{s}i_0$ in ρ. So $\{r_0\}/\{s_0 \twoheadrightarrow {}^{s}i_0\} = \varnothing$. Now the projection $\rho' = \rho/\{t_0 \to t_1\}$ is again a

fair reduction, by Remark xx, since ρ and ρ' coincide from ${}^{s}i_0$ onwards. So we apply the same procedure to the second step $t_1 \to t_2$, which is absorbed by ρ at the term ${}^{s}i_1$, that is, $\{r_1\}/\{t_1 \twoheadrightarrow {}^{s}i_1\} = \varnothing$. Continuing in this way we find a term ${}^{s}i_k$ in ρ such that the whole vertical reduction $t_0 \twoheadrightarrow t_n$ is absorbed at that point, i.e. $\{t_0 \twoheadrightarrow t_n\}/\{s_0 \twoheadrightarrow {}^{s}i_k\} = \varnothing$. This means that $t_n \twoheadrightarrow {}^{s}i_k$, which proves that the sequence ρ is indeed cofinal. □

REMARK. Fairness of a reduction ρ implies that no redex in a term of ρ "stays alive forever", that is, has an infinite chain of descendants. Note, however, that this last property is strictly weaker than fairness as the following example shows. Let R = {a → b, f(x) → g(x, f(x))}, and

$\rho =$ f(a) →
g(a, f(a)) →
g(b, f(a)) →
g(b, g(a, f(a))) →
g(b, g(b, f(a))) →
g(b, g(b, g(a, f(a)))) →
g(b, g(b, g(b, f(a)))) → ...

Then no redex in ρ has an infinite chain of descendants, but still ρ is not cofinal; namely, f(a) → f(b) but f(b) does not reduce to any term in ρ.

10.4. COROLLARY. *For orthogonal TRSs:*
(i) $\mathbb{F}_{GK}$ *is a cofinal (and hence also normalizing) reduction strategy.*

PROOF. In each $\mathbb{F}_{GK}$-step all redexes are contracted; so the reduction delivered by $\mathbb{F}_{GK}$ is fair. □

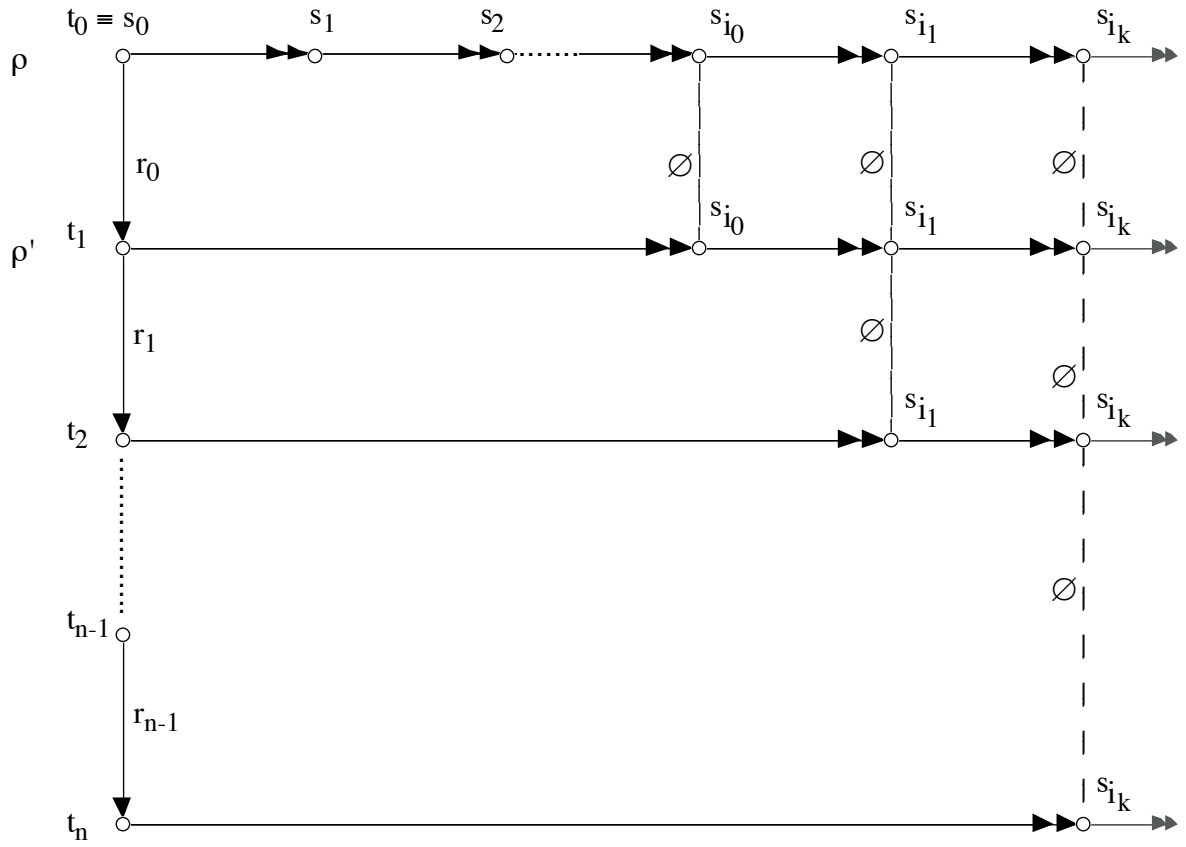


Figure 10.7. Fair implies cofinal

A similar relaxation of constraints applies to the other two strategies $\mathbb{F}_{po}$ and $\mathbb{F}_{lm}$:

10.11. DEFINITION. (i) A reduction ρ is *leftmost-fair* if ρ ends in a normal form or infinitely many times the leftmost outermost redex is contracted in ρ.
(ii) $\rho = t_0 \to t_1 \to \ldots$ is *outermost-fair* if ρ does not contain a term t_n with an outermost redex which infinitely long stays an outermost redex but which is never contracted.

10.12. THEOREM. *Let* R *be an orthogonal TRS. Then:*
(i) *Outermost-fair reductions are normalizing.*
(ii) *If* R *is moreover left-normal, then leftmost-fair reductions are normalizing.*
□

We will now summarize some of the main properties of the various reduction strategies (and their 'relaxed' versions) in Table 10.2. Before doing so, we introduce one more property of strategies:

10.13. DEFINITION. A reduction strategy $\mathbb{F}$ for R is *perpetual*, if for all t: $\infty(t) \Rightarrow \infty(\mathbb{F}(t))$.

Here ∞(t) means that t has an infinite reduction, i.e. ¬SN(t). So a perpetual strategy is the opposite of a normalizing one; it tries to avoid normal forms whenever possible, and could therefore also be called 'anti-normalizing'.

In Table 10.2 p, n, c stand for perpetual, normalizing, cofinal respectively. In case a property is not mentioned, it does not hold generally. Note that for the leftmost-outermost strategy, when applied to orthogonal TRSs in general, none of the three properties holds generally.

	orthogonal TRSs	*orthogonal left-normal TRSs*	*orthogonal non-erasing TRSs*
leftmost-innermost	p	p	p n
parallel-innermost	p	p	p n
leftmost-outermost (leftmost-fair)		n	p n
parallel-outermost (outermost-fair)	n	n	p n
full substitution (fair)	n c	n c	p n c

Table 10.2

**Computable reduction strategies.*

A strategy is *recursive* or *computable* if it is, after a coding of the terms into natural numbers, a recursive function. Obviously we are primarily interested in computable strategies; and indeed all five strategies in Definition xx are computable. We may now ask whether there is always for an orthogonal TRS a *computable one-step normalizing* reduction strategy. A priori this is not at all clear, in view of TRSs such as the one given by G. Berry: CL extended with rules

$$FABx \to C$$
$$FBxA \to C$$
$$FxAB \to C$$

which is an orthogonal TRS. This TRS seems to require a parallel reduction strategy (so, not a one-step or sequential strategy), because in a term of the form FMNL we have no way to see the 'right' argument for computation: a step in the third argument may be unnecessary, namely if the first and second argument evaluate to A and B respectively (which is undecidable due to the presence of CL); likewise a step in the other arguments may be unnecessary. In the next section about sequential TRSs

this problem will be analyzed extensively.

When we want to be more liberal, we can consider the same problem for the weakly orthogonal TRS obtained by extending CL with Parallel-or:

$$or(true, x) \to true$$
$$or(x, true) \to true.$$

Such TRSs seem to require a parallel evaluation. However, there is the following surprising fact.

10.14. THEOREM *For every weakly orthogonal TRS there exists a computable sequential normalizing reduction strategy.*

The algorithm involved is however too complicated to be of more than theoretical interest.

**Standard reductions in orthogonal TRSs*

For λ-calculus and CL there is a very convenient tool: the Standardization Theorem. For orthogonal TRSs there is unfortunately not a straightforward generalization of this theorem. The obstacle is the same as for the normalizing property of the leftmost reduction strategy, observed in Remark 10.6.(i). When we restrict ourselves again to left-normal orthogonal TRSs, there is a straightforward generalization.

10.15. DEFINITION. (*Standard reductions*)
Let $\mathcal{R}$ be a TRS and $\mathsf{R} = t_0 \to t_1 \to \ldots$ be a reduction in R. Mark in every step of R all symbols to the left of the head symbol of the contracted redex, with '*'. Furthermore, markers are persistent in subsequent steps.

Then R is a *standard reduction* if in no step a redex is contracted with a marked head operator. (So the action in R moves literally from left to right, and an increasing left part of the term is 'frozen'.)

10.16. STANDARDIZATION THEOREM for left-normal orthogonal TRSs.
Let R *be a left-normal orthogonal TRS. Then: if* $t \twoheadrightarrow s$ *there is a standard reduction in* R *from* t *to* s.
□

10.8. THEOREM. *Let* R *be a left-normal orthogonal TRS. Then* $\mathbb{F}_{lm}$ *is a normalizing reduction*

strategy for R.

PROOF. Suppose t has a normal form t_0. By the Standardization Theorem, there is a standard reduction from t to t_0. This is in fact the reduction as given by the strategy $\mathbb{F}_{lm}$.) □

10.8. REMARK.This fact is in λ-calculus and CL literature also known as the *Normalization Theorem.*

10.19. EXERCISE. **Primitive recursive functions.**
The primitive recursive functions from $\mathbb{N}$ to $\mathbb{N}$ are defined by the following inductive definition:

(i) The *constant* functions $C_{n,k}$, the *projection* functions $P_{n,i}$ and the *successor* function S are primitive recursive. (Here $C_{n,k}(x_1,\ldots,x_n) = k$; $P_{n,i}(x_1,\ldots,x_n) = x_i$; $S(x) = x+1$.)
(ii) If G, $H_1,\ldots,H_k$ are primitive recursive, then F defined by

$$F(\mathbf{x}) = G(H_1(\mathbf{x}),\ldots,H_k(\mathbf{x}))$$

(where $\mathbf{x} = x_1,\ldots,x_n$) is primitive recursive.
(iii) If G and H are primitive recursive, then F defined by

$$F(0, \mathbf{x}) = G(\mathbf{x})$$
$$F(S(y), \mathbf{x}) = H(F(y, \mathbf{x}), y, \mathbf{x})$$

is primitive recursive. Here $\mathbf{x} = x_1,\ldots,x_n$.

Show that, by replacing every '=' by '→' in the defining equations, every primitive recursive function is defined by a terminating, left-normal, orthogonal constructor TRS.

10.20. EXERCISE (Hindley).
(i) Consider CL extended with Recursor, where Recursor = {Rxy0 → x, Rxy(Sz) → yz(Rxyz)}. Note that this applicative TRS is not left-normal, and show that $\mathbb{F}_{lm}$ is not normalizing.
(ii) However, for the following TRS: CL + Recursor* where Recursor* = {R*0xy → x, R*(Sz)xy → yz(Rzxy)} the strategy $\mathbb{F}_{lm}$ *is* normalizing.

10.22. EXERCISE (Klop, see Kennaway [89]). According to Kennaway's theorem xx, the weakly orthogonal TRS R = CL ⊕ {*or*(*true*, x) → *true*, *or* (x, *true*) → *true*} has a computable sequential normalizing reduction strategy, even though this may seem counter-intuitive. In this exercise we give a stylized version of the problem.

(i) Let functions f, g: $\mathbb{N}^+ \to \mathbb{N}$ be given. Here $\mathbb{N}^+ = \mathbb{N} - \{0\}$, the set of positive natural numbers. Consider the ARS $\mathsf{A} = \langle \mathbb{N} \times \mathbb{N}, \to \rangle$ where → is defined by:

$$\begin{array}{lll} (x, y) & \to (f(x), y) & \text{if } x, y > 0 \\ (x,y) & \to (x, g(y)) & \text{if } x, y > 0 \\ (x, 0) & \to (0, 0) & \text{if } x > 0 \\ (0, y) & \to (0, 0) & \text{if } y > 0. \end{array}$$

(Comment: intuitively (x, y) in A is to be compared with a term *or*(X, Y) in R; 0 is *true*; (0, 0) is *true*; application of f, g stands for performing a reduction step in the first (respectively second) argument.)

Show that A is confluent. Note that (x, y) has a normal form (which is (0, 0)) iff $\exists n\ (f^n(x) = 0 \ \vee\ g^n(y) = 0)$.

(ii) A 1-step (or sequential) reduction strategy S for A is a map $\mathbb{N} \times \mathbb{N} \to \{L, R\}$, specifying the left or right component of (x, y) to be reduced. So, if S(x, y) = L, x, y > 0, then the corresponding step is (x, y) → (f(x), y).

Prove that for every recursive f, g there exists a 'good' strategy S, i.e. one that is computable, sequential, normalizing.

[Prooof sketch. Given (x, y), distinguish the cases x < y and x ≥ y. In case 1, consider the least n such that : (i) $f^n(x) = 0$ or (ii) $f^n(x) \geq y$ or (iii) $\exists i < n \;\; f^n(x) = f^i(x) \neq 0$. For this least n, the cases (i-iii) are mutually exclusive. Compute this n. In case (i) or (ii), define S(x, y) = L, in case (iii), define S(x, y) = R.

In case 2 (x ≥ y), compute likewise the least n such that (i) $g^n(y) = 0$ or (ii) $g^n(y) \geq x$ or (iii) $\exists i < n \;\; g^n(y) = g^i(y) \neq 0$. In case (i), (ii): define S(x, y) = R; in case (iii), S(x, y) = L. Now show that S is a 'good' strategy.]

10.23. EXERCISE. Consider CL + Pairing, where Pairing = $\{D_0(Dxy) \to x, D_1(Dxy) \to y\}$.
(i) Show that the reduction $D_0(D(KII)I) \to D_0(DII) \to I$ is not standard.
(ii) $D_0(D(KII)I) \to KII \to I$ is standard.

10.24. EXERCISE (Hindley).
Consider in the applicative TRS R = {PxQ → xx, R → S, Ix → x} the reduction

$$R = PR(IQ) \to PRQ \to RR \to SR$$

and show that there is no standard reduction for R (i.e. a reduction $PR(IQ) \twoheadrightarrow SR$ which is standard).

10.25. Non-erasing reductions

Apart from confluence, many interesting facts can be proved for orthogonal TRSs. These facts are stated in terms of the following notions:

10.25.1. DEFINITION. (i) A TRS is *non-erasing* if in every rule t → s the same variables occur in t and in s. (E.g. CL is not non-erasing, due to the rule Kxy → x.)
(ii) A TRS is *weakly innermost normalizing* (WIN) if every term has a normal form which can be reached by an *innermost* reduction. (In an innermost reduction a redex may only be 'contracted' if it contains no proper subredexes.)

EXAMPLE. (i) Just as the λ-calculus has a non-erasing version, called λI-calculus, there is a non-erasing version of CL, Combinatory Logic. It is called CL_I, and has instead of I, K, S of CL the four basic combinators I, B, C, S with rules as in Table xx.

Ix → x
Bxyz → x(yz)
Cxyz → xzy
Sxyz → xz(yz)

Table 10.25.2

CL_I is an orthogonal, non-erasing TRS. Another 'basis' for CL_I consists of the two constants I and J with rules

$$Ix \to x$$
$$Jxyzw \to (xy(xwz)$$

10.25.2. DEFINITION. A reduction step $M \to M'$ is called *critical* if $\infty(M)$ but $\neg\infty(M')$. So a critical step is one where the possibility of performing an infinite reduction is lost.

For the proof of the Erasure Lemma we need a simple proposition.

10.25.3. PROPOSITION ('absorption') . *Let* t *be a term in an orthogonal TRS, containing mutually disjoint redexes* $s_1,...,s_n$, *and a redex* s. *Let* $t \twoheadrightarrow t'$ *be the* n*-step reduction obtained by contraction, in some order, of the redexes* $s_1, ..., s_n$. *Suppose* s *has after the reduction* $t \twoheadrightarrow t'$ *no descendants in* t'.

Then for some $i \in \{1,...,n\}$: $s \subseteq s_i$. *This means that either* s *coincides with some* s_i *('coincidence') or is contained in an argument of some* s_i *('erasure').*

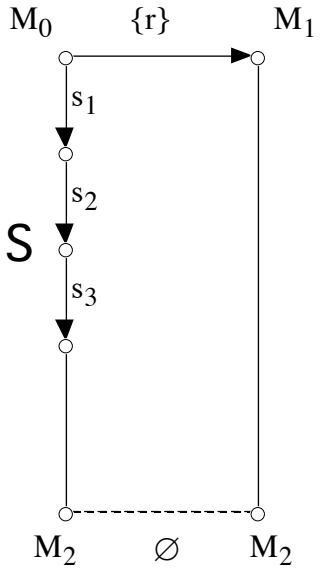


Figure 10.12

10.25.4. PROPOSITION. (Erasure Lemma)
Let $M \to M'$ *be a critical step in an orthogonal TRS, where* s *is the contracted redex. Then the step* {s} *erases a subterm p with* $\infty(p)$.

PROOF. As $M_0 \to M'_0$ is a critical step, we have $\infty(M_0)$ and $\neg\infty(M'_0)$. So M_0 has an infinite reduction R, but M'_0 has not. Hence, the projection $R/\{s\}$ must be the empty reduction $\varnothing$ from a

certain stage onwards. S_n is the 'parallel move' contracting the descendants of s after $M_0 \twoheadrightarrow M_n$. In $M_n \rightarrow M_{n+1}$ redex r_n is contracted. So from a certain n onwards, $\{r_n\}$ is absorbed by S_n.By the preceding Proposition, this can happen by coincidence, or by erasure. Coincidence is possible only finitely many times, since after such an event, the resulting S_{n+1} contracts one redex less than S_n. We conclude that from a certain N onwards, the redexes $r_N, r_{N+1}, \ldots$ are always aborbed by erasure (by one of the redexes in S_n).

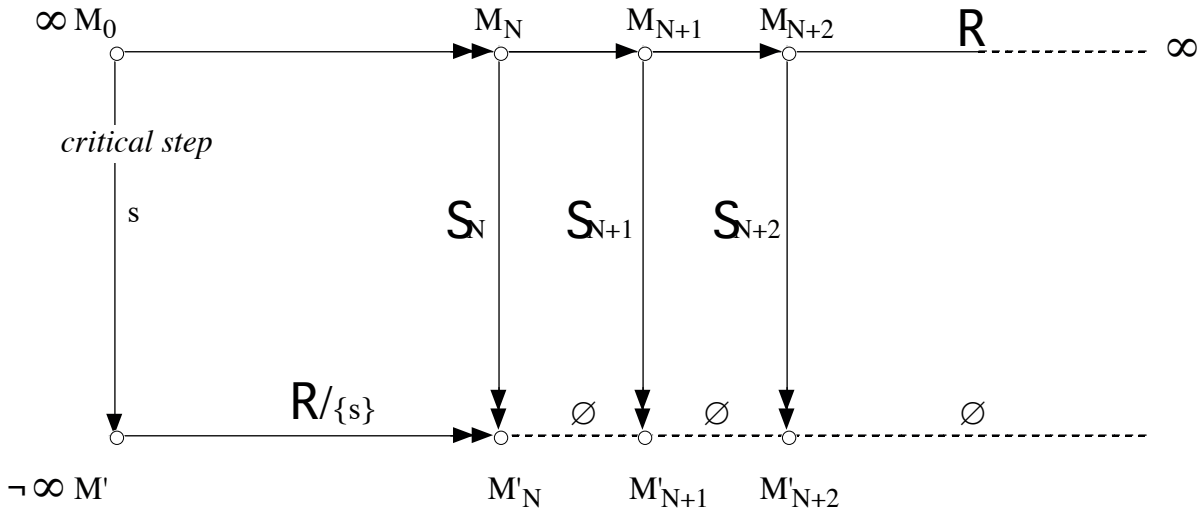


Figure 10.13

We note that $s \equiv C[t_1,\ldots,t_n]$ and that the descendants of s in the parallel move S_n have always the form $C[t'_1,\ldots,t'_n]$, for the same context C[,...], with $t_i \twoheadrightarrow t'_i$ (i = 1,...,n). Let us take as a concrete example: C[,...,] ≡ F(G(□), □, H(□)), the pattern of the rule according to which s is a redex. So $s \equiv F(G(t_1), t_2, H(t_3))$. And S_N contracts the (say) k descendants of s:
$F(G(t_{1,j}), t_{2,j}, H(t_{3,j}))$, for j = 1,...,k. Likewise for all the subsequent S_{N+k} ($k \geq 0$). We can visualize this as a sequence of blocks of triples □□□, as in the figure, each block having 3 times k boxes ø. From each block to the next, there is a step in one of the boxes. Hence, by the pigeon-hole principle (see chapter Background, xx), one of the triples □□□ in the first block has to be infinitely many times 'active'; and of this triple again one box □. It follows that one of the original t_1,t_2,t_3 in s has the property ∞, say t_2. Now this t_2 has to be erased in the step $M_0 \rightarrow M'_0$, for otherwise also M'_0 would have an infinite reduction.

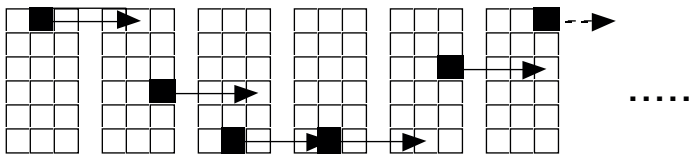


Figure 10.14. Infinite sequence of reduction steps.

The next theorem was proved in Church [41] for the case of the non-erasing version of λ-calculus, the λI-calculus, where the restriction on term formation is adopted saying that in every abstraction term λx.M the variable x must have a free occurrence in M.

10.7.5. THEOREM. *Let* R *be orthogonal and non-erasing. Then:* R *is WN iff* R *is SN.*

PROOF. Using the Lemma it is now easy to prove Theorem 8.10: 'critical'steps $t \to t'$ in which $\infty(t)$ but $\neg \infty(t')$, cannot occur in a non-erasing TRS. □

Another useful theorem, which also reduces the burden of a termination (SN) proof for orthogonal TRSs, is:

10.25.6. THEOREM (O'Donnell [77]). *Let* R *be an orthogonal TRS. Then:* R *is WIN iff* R *is SN.*

PROOF. follows from the Lemma in (ii) by observing that an innermost contraction cannot erase a proper subterm which admits an infinite reduction, since otherwise the contracted redex would not have been innermost. □

The last two theorems can be refined to terms: call a term WN if it has a normal form, SN if it has no infinite reductions, WIN if it has a normal form reachable by an innermost reduction. The 'local' version of Theorem xx then says that for a term in an orthogonal, non-erasing TRS the properties WN and SN coincide. Likewise there is a local version of Theorem xx. Thus, if in CL a term can be normalized via an innermost reduction, all its reductions are finite.

As an illustration of the use of the Erasure Lemma we will prove the following theorem. Actually, the theorem is an instance from Theorem xx, stating modularity of completeness (i.e. SN & CR) for general left-linear TRSs; since the proof of that theorem is very complicated, the proof below is relevant.

10.25.7. THEOREM. *SN is modular for orthogonal TRSs.*

PROOF. Consider a minimal counterexample $t \equiv C[\![t_1,...,t_n]\!]$, minimal with respect to the length of the term (also the rank can be used). So t admits an infinite reduction, but the principal subterms $t_1,...,t_n$ are SN. Now normalize the t_i, result t'_i (i = 1,...,n). We then have $t' \equiv C'[t'_1,...,t'_n] \equiv D[\![s_1,...,s_n]\!]$, with the s_j in normal form. Clearly, t' is SN. But that means that during the reduction $t \ddot{y} t'$ we have "lost infinity". According to the Erasure Lemma, this can only happen if a subterm has been erased that

admits an infinite reduction. However, the reduction from t to t' took place in the SN terms t_i; so no subterm with infinite reduction could be erased. Contradiction. □

11
OVERZICHT

In dit overzicht geven we per Hoofdstuk summier enkele dingen aan die goed gekend moeten worden, maar ook onderdelen die overgeslagen kunnen worden. Verder recapituleren we de belangrijkste stellingen en begrippen. Maar deze lijst is niet uitputtend, en in principe is de tentamenstof datgene wat behandeld is op college en in werkcollege!

11.1. Hoofdstuk 1. Woorden.

Woorden kunnen herschrijven in een SRS (String Rewrite System.) Reductiegraaf van een woord. Nomaalvorm. Niet: 1.3. Tag systems. Sectie 1.5., Symmetrieen: opgaven hierover kunnen maken. Op p.6 en 7 is een font probleem, daar zijn griekse letters steeds vervangen door pijl => in sommige viewers.

11.2. Hoofdstuk 2. Termen.

Kunnen herschrijven in een TRS. In 2.3 Opgave: Levy-equivalentie is niet behandeld. Bewijs van Generatie Lemma hoeft niet. In het algemeen worden geen bewijzen van Stellingen en Lemma's gevraagd.

11.3. Hoofdstuk 3. Combinatoren.

Reductieregels van CL, in alle versies (met expliciete applicatiefunctie, met notatie 'associatie naar links') goed kennen, en reducties van termen uit kunnen voeren. Ook, in Hoofdstuk 10, de daar behandelde strategieen (leftmost, parallel-outermost, full substitution,...) toe kunnen passen op een CL-term. Normaalvorm . Reductiegraaf. Fixed point combinator.

11.4. Hoofdstuk 4. Combinatoren en lambda termen.

Notatie van lambda-termen goed kennen. Alfa, beta, eta-reductie toe kunnen passen. Fpc van Curry en Turing kennen en kunnen gebruiken. Vertaling van Lambda naar CL kennen. Weten wat combinatorische compleetheid is. Ook met I,J-termen kunnen reduceren. Van de reductieregels voor I,K,S,I,J etc. weten of het c-of d-regels zijn (als in Hoofdstuk 5 gedefinieerd.) CL is een orthogonale TRS, zowel met basis I,K,S als met basis I,J. ***Dus CR.*** *De I,J versie is non-erasing (NE). Dan geldt de stelling van Church: NE => (WN <=> SN). OOk voor een enkele term deze stelling kunnen toepassen! Bv. een term in CL met basis I,J is WN <=> SN.*
Niet: op pag.13 van pdf versie op het web, de Propositie over head-reductie (boven Stelling

Combinatorische Compleetheid).

11.5. Hoofdstuk 5. Confluentie.

CR, CR^=, WCR^≤1, WCR, zeer goed kennen, idem eigenschappen SN, WN, UN, UN^->, NF, en hun onderlinge relaties waarvan Newman's Lemma een van de belangrijkste is. Begrip 'sterk confluent', opg. 5.1.6.5 p.9 en op practicum uitvoerig behandeld. Bewijs van NL niet. Definitie 'modulair', belangrijk. Stelling 5.2.2 goed kennen. Lineair, links-lineair, collapsing, duplicating rule, belangrijk. Stellingen 5.2.3, 5.2.9, 5.2.10 goed kennen en kunnen gebruiken. Noties orthogonaal, redex patroon, overlap zeer goed kennen. Zeer belangrijk, op bord behandeld, maar misschien niet expliciet in syllabus aanwezig is ***de stelling dat CR geldt voor alle orthogonale TRSen****. Ook belangrijk: van een TRS kunnen nagaan of hij orthogonaal is. Het begrip 'weakly orthogonaal' kennen;* ***ook voor weakly orthogonale TRSen geldt CR algemeen****. (Staat niet in syllabus.)*

Opg. 5.1.6.7 p.10 niet. Ook niet: 5.2.14, 5.2.15, 5.2.18, 5.2.19, 5.2.20.

11.6. Hoofdstuk 6. Abstract herschrijven.

Kommen en knikkers voorbeeld wel begrijpen, bv. weten wat de normaalvormen zijn. Begrip LO (linear orthogonal) kennen. Pag. 6 niet. Braids diagram (zoals Fig. 6.2.6) kunnen afmaken. Zeer belangrijk: DD, decreasing diagrams: stelling hierover kennen: DD => CR. Kunnen bepalen of e.d.'s (elementaire diagrammen) decreasing zijn of niet.

6.2.2, 6.3.9, 6.3.10 niet. Wel 6.3.1. Wel 6.3.6. pag 11, 2.8 niet. ***Pag.13 6.3.11 wel!*** *Rest van 6.3 niet.*

11.7. Hoofdstuk 7. Terminatie: Iterative Path orders.

Het enige wat van dit hoofdstuk zeer goed gekend moet worden, zijn de regels voor IPO, put, down, copy, select, deze toe kunnen passen om voor TRSen SN te bewijzen, zoals in voorbeeld in Fig.2.3 p.7. Merk op dat dit dezelfde TRS (SRS) is als in H1.

Het bewijs dat IPO werkt, hoeft niet, dus ook niet de regels met labels als in Fig. 3.1, p.9.

Een andere terminatiemethode die wel goed gekend moet worden, is de MST methode, multiset terminatie. Deze staat niet in de syllabus, maar is op het bord behandeld. Toen is ook Koenig's Lemma (KL) behandeld, maar alleen op slides, staat niet in syllabus. Wel kennen. PHP (pigeon-hole principle) wordt ook bekend verondersteld.

11.8. Hoofdstuk 8. Completering.

Zeer belangrijk. Goed kunnen toepassen. Kritieke paren kunnen bepalen, Kritieke paren lemma kennen en toe kunnen passen. ***6.3.2!****, 6.3.3 weten. Maar 6.3.7-12 niet.* ***Opgaven. 6.5.5-18 zijn belangrijk.***

Niet: sectie 6.6, pag. 26-31.

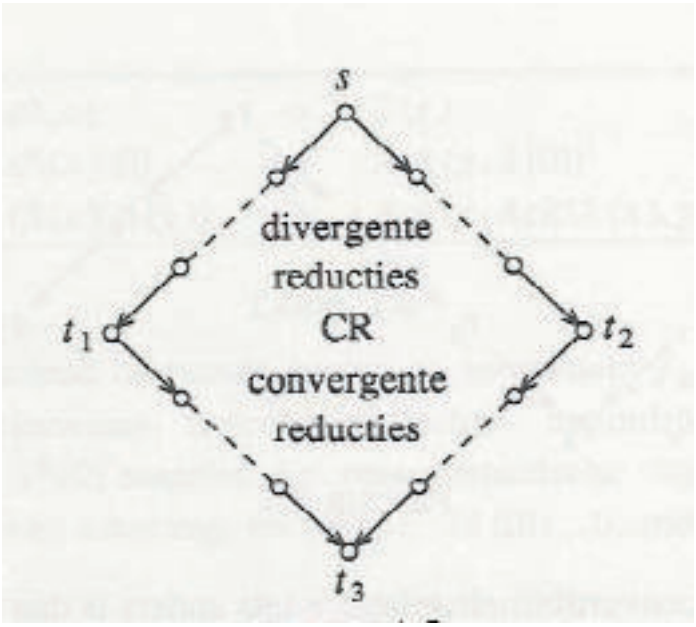
11.9. Hoofdstuk 9. Oneindig herschrijven.
Basis noties kennen: oneindige reductie, limiet, normaalvormen kunnen herkennen. Appendix op pag.13 e.v. niet. Wel tegenvoorbeelden tegen infinitaire CR. Zeer belangrijk: sterk convergente reducties, goed kunnen onderscheiden van reducties die slechts Cauchy-convergent zijn. Belangrijk: stelling dat ook al geldt CR^inf niet, UN^inf geldt wel.

11.10. Hoofdstuk 10. Strategieen.
De behandelde strategieen kennen, en hun eigenschappen kennen, bv. dat F_po normaliserend is.Belangrijk: EL, Erasure Lemma, en de gevolgen: Stelling van Church is heel belangrijk. Bewijs van EL hoeft niet.

Nu volgen enkele definities en stellingen uit Hoofdstuk 5, Confluentie. Dit omdat per abuis niet elk begrip in de syllabus werd gedefinieerd, met name SN niet. Deze definitie stond wel op een aparte hand-out note. IJsbrand vond nog een nuttige samenvatting van Aart Middeldorp, die veel voorbeelden bevat. Het gaat om de eerste 10 pagina's van het artikel "Term Rewriting Systems: Divide and Conquer" van Aart Middeldorp. Dit artikel is van de volgende locatie te downloaden:
`http://web.yl.is.s.u-tokyo.ac.jp/pub/okadasemi/ami.ps.Z`

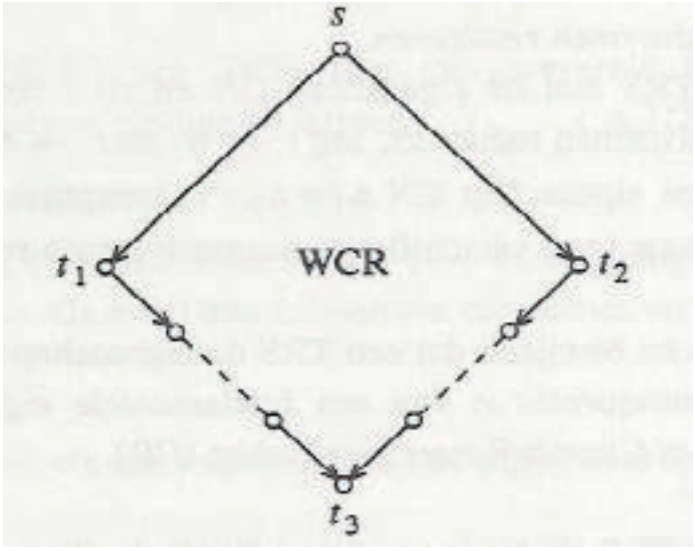
WAARSCHUWING: Het diagram in dit artikel op pag. 2 is niet correct. Bijvoorbeeld staat daar WCR -> CR en andere implicaties, maar die zijn niet allemaal juist. Sla deze figuur dus over!

5.1.1. DEFINITIE. Een TRS (Σ, R) is *confluent* of *heeft de Church-Rosser eigenschap* (*is CR*) als er voor elke twee 'divergente' reducties $s \twoheadrightarrow t_1$, $s \twoheadrightarrow t_2$ 'convergente' reducties $t1 \twoheadrightarrow t_3$ en $t_2 \twoheadrightarrow t_3$ zijn. (Figuur 5.1). De convergente reducties mogen uit 0 stappen bestaan.



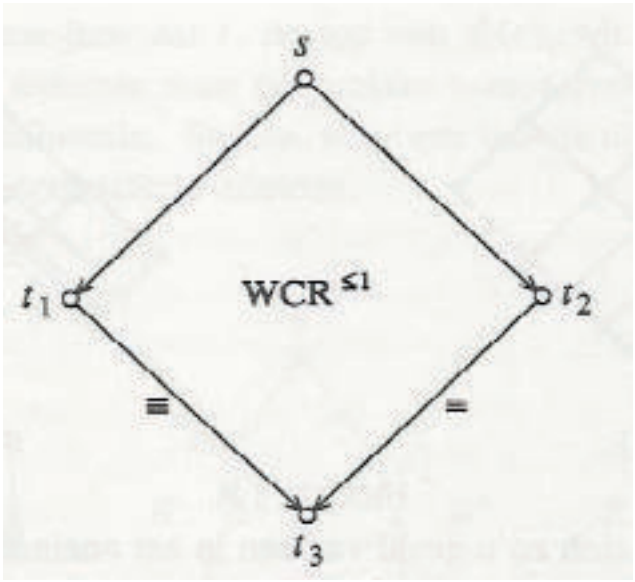
Figuur 5.1. *De eigenschap CR.*

Een belangrijke zwakkere eigenschap is WCR ('weak Church-Rosser', zwakke confluentie).



Figuur 5.2. *De eigenschap WCR.*

5.1.2. DEFINITIE. (i) Een TRS (Σ, R) is *zwak confluent* (WCR) als er voor elke twee divergente *éénstapsreducties* $s \to t_1$, $s \to t_2$ convergente reducties $t_1 \twoheadrightarrow t_3$ en $t_2 \twoheadrightarrow t_3$ zijn. (Figuur 5.2). De convergente reducties mogen willekeurige lengte hebben.



Figuur 5.3. *De eigenschap WCR.*

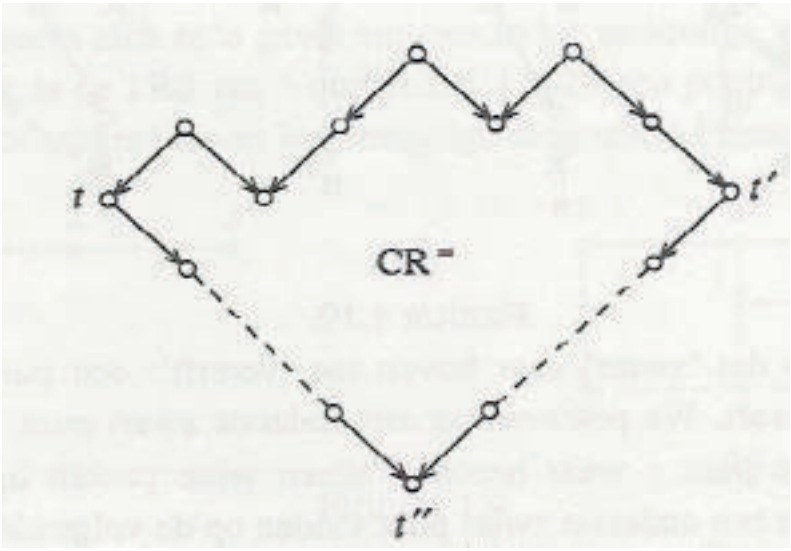
(ii) Als bovendien altijd convergente reducties van *nul of één stappen* gevonden kunnen worden, dan zeggen we dat de TRS (Σ, R) de eigenschap $WCR^{\leq 1}$ heeft (Figuur 5.3).
Hierbij is $\rightarrow^{=}$ de reflexieve afsluiting van $\rightarrow$, dat wil zeggen: $s \rightarrow^{=} t \Leftrightarrow s \rightarrow t$ of $s \equiv t$.

5.1.3. STELLING. (Newman's Lemma, 1942).
Voor elke TRS *geldt:*

$$SN \,\&\, WCR \Rightarrow CR.$$

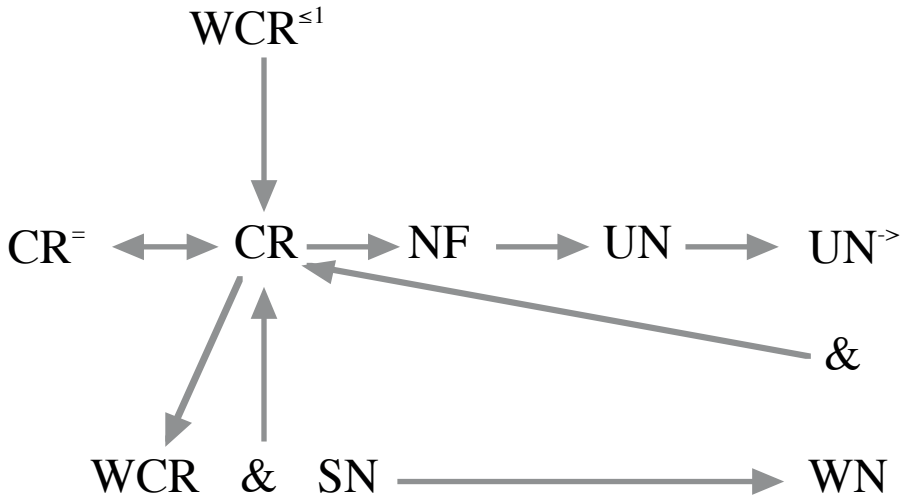
5.1.4. DEFINITIE. Een TRS (Σ, R) heeft de eigenschap $CR^{=}$ als:

$\forall t, t' \in Ter(\Sigma, R)$: als $t = t$ dan is er een t'' met $t \twoheadrightarrow t''$ en $t' \twoheadrightarrow t''$.
(Zie Figuur 5.10.)



Figuur 5.10. *Zigzag versie van CR.*

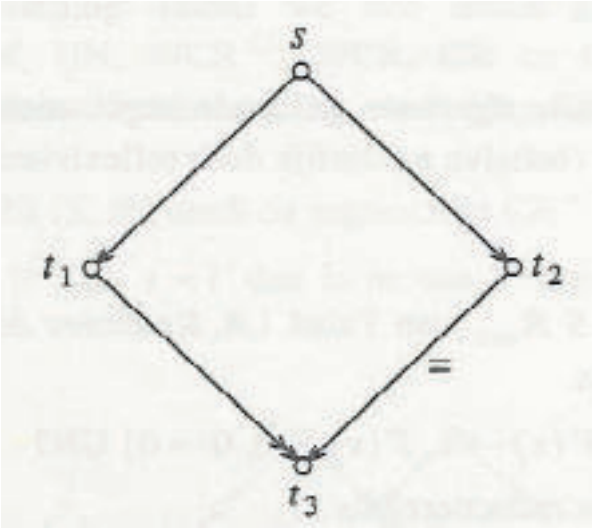
5.1.5. STELLING. *Voor elke* TRS *gelden de volgende implicaties:*



Figuur 5.11. *Relaties tussen diverse TRS eigenschappen.*

5.1.6.6. OPGAVE. Een TRS $\mathcal{R}$ heet *sterk confluent* als er voor elke twee éénstapsreducties $s \to t_1$, $s \to t_2$ een t_3 bestaat zodat $t_1 \twoheadrightarrow t_3$ en $t_2 \to^{=} t_3$ (zie Figuur 5.12)

Bewijs dat elke sterk confluente TRS confluent is.



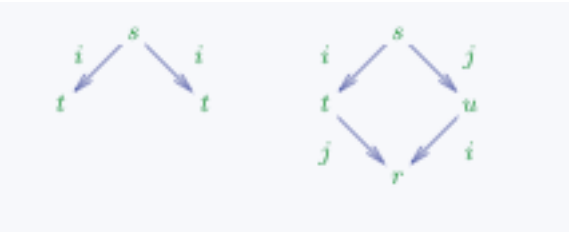
Figuur 5.12. *Sterke confluentie*

5.2.11.5. OPGAVE. Een TRS (Σ, R) heeft de eigenschap NF (*normaalvorm eigenschap, normal form property*) als:

$$\forall s, t \in Ter(\Sigma, R): \text{als } s = t \text{ en } t \text{ is normaalvorm dan } s \twoheadrightarrow t.$$

a) Bewijs dat voor elke TRS de implicaties CR ⇒ NF en NF ⇒ UN gelden.

6.1. DEFINITIE. Een ARS (A, →) is *lineair orthogonaal* (LO) als voor elke vork
s →t en s →u geldt: ofwel t = u, of er is een v met t → v en u → v.



Figuur 6.3. *Lineair orthogonaal.*

6.2. STELLING (Toyama [92]). *Als* (A, →) LO *en* WN *is, dan is* (A, →) *bovendien* SN, $UN^{\rightarrow}$, *en alle reducties naar de normaalvorm zijn even lang.*

(Nederlandse versie beneden.)

DEFINITION. Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS.

(i) We say that $a \in A$ is a *normal form* if there is no $b \in A$ such that $a \rightarrow b$. Further, $b \in A$ *has a normal form* if $b \twoheadrightarrow a$ for some normal form $a \in A$.

(ii) The reduction relation $\rightarrow$ is *weakly normalizing* (WN) if every $a \in A$ has a normal form. In this case we also say that A is WN.

(iii) A (or $\rightarrow$) is *strongly normalizing* (SN) if every reduction sequence $a_0 \rightarrow a_1 \rightarrow ...$ eventually must terminate. (Other terminology: $\rightarrow$ is *terminating*, or *noetherian.*)

(iv) A (or $\rightarrow$) has the *unique normal form property* (UN) if
$\forall a, b \in A$ ($a = b$ & a,b are normal forms $\Rightarrow a \equiv b$).

(v) A (or $\rightarrow$) has the *unique normal form property with respect to reduction* ($UN^{\rightarrow}$) if
$\forall a, b,c \in A$ ($c \twoheadrightarrow a$ & $c \twoheadrightarrow b$ & a,b are normal forms $\Rightarrow a \equiv b$).

(vi) A (or $\rightarrow$) has the *normal form property* (NF) if
$\forall a, b \in A$ (a is a normal form & $a = b \Rightarrow b \twoheadrightarrow a$).

DEFINITIE. Zij $\mathcal{A} = \langle A, \rightarrow \rangle$ een ARS. (Dus deze definities gelden ook voor een SRS of TRS.)

(i) We zeggen dat $a \in A$ een normaalvorm is als er geen $b \in A$ is zodat $a \rightarrow b$. Verder, $b \in A$ *heeft een normaalvorm* als $b \twoheadrightarrow a$ voor zekere normaalvorm $a \in A$.

(ii) De reductierelatie $\rightarrow$ is *zwak normaliserend*, *weakly normalizing* (WN) als elke $a \in A$ een normaalvorm heeft. In dat geval zeggen we ook: $\mathcal{A}$ is WN.

(iii) $\mathcal{A}$ (of $\rightarrow$) is sterk normalserend, *strongly normalizing* (SN) als elke reductierij $a_0 \rightarrow a_1 \rightarrow ...$ op den duur moet eindigen (*eventually terminates*). (Andere terminologie: $\rightarrow$ is terminerend, *terminating*, of noethers, *noetherian.*)

(iv) $\mathcal{A}$ (of $\rightarrow$) heeft de unieke normaalvormeigenschap, *unique normal form property* (UN) als $\forall a, b \in A$ ($a = b$ & a, b are normal forms $\Rightarrow a \equiv b$).

(v) $\mathcal{A}$ (of $\rightarrow$) heeft de uniekenormaalvormeigenschap t.o.v. reductie, *unique normal form property with respect to reduction* ($UN^{\rightarrow}$) als
$\forall a, b, c \in A$ ($c \twoheadrightarrow a$ & $c \twoheadrightarrow b$ & a,b zijn normaalvormen $\Rightarrow a \equiv b$).

(vi) $\mathcal{A}$ (of $\rightarrow$) heeft de normaalvormeigenschap, the *normal form property* (NF)als
$\forall a, b \in A$ (a is een normaalvorm & $a = b \Rightarrow b \twoheadrightarrow a$).

5.2.6. DEFINITIE. (i) Een term is *lineair* als er geen variabele dubbel in voor komt, en *niet-lineair* als dat wel zo is. (Bijv. G(x, x, y) is niet-lineair.)
(ii) Een reductieregel $t \rightarrow s$ is *links-lineair* als t lineair is.
(iii) Een TRS is *links-lineair* als zijn reductieregels links-lineair zijn.

5.2.7. STELLING. (Toyama, Klop & Barendregt [89]).

Laten $\mathcal{R}_1$, $\mathcal{R}_2$ links-lineaire TRSen zijn. Dan:

$\mathcal{R}_1 \oplus \mathcal{R}_2$ is compleet dan en slechts dan als $\mathcal{R}_1$ and $\mathcal{R}_2$ compleet[1] *zijn.*

5.2.9. STELLING (Middeldorp [88]) *Stel dat $\mathcal{R}_1$ en $\mathcal{R}_2$ twee disjuncte* TRSen *zijn, beide* SN.

(i) *Als $\mathcal{R}_1$ noch $\mathcal{R}_2$ c-regels bevat, dan heeft $\mathcal{R}_1 \oplus \mathcal{R}_2$ de eigenschap* SN.

(ii) *Als $\mathcal{R}_1$ noch $\mathcal{R}_2$ d-regels bevat, dan heeft $\mathcal{R}_1 \oplus \mathcal{R}_2$ de eigenschap* SN.

(iii) *Als één van beide TRSen $\mathcal{R}_1$, $\mathcal{R}_2$ noch c-regels, noch d-regels bevat, dan is $\mathcal{R}_1 \oplus \mathcal{R}_2$* SN.

[1] Ter herinnering: compleet = CR & SN. (Op bord gedefinieerd, maar misschien niet in syllabus terecht gekomen.)

LAMBDA DEFINIEERBAARHEID

In dit hoofdstuk zullen we bewijzen dat de klasse van *recursieve functies* precies samenvalt met de functies die in λ-calculus defineerbaar zijn. We zullen ons beperken tot *totale* functies van $\mathbb{N}$ naar $\mathbb{N}$ en houden ons dus niet met de grotere klasse van partiëel recursieve functies (die ook in λ-calculus definieerbaar zijn) bezig. We voeren eerst de *primitief recursieve* functies in.

1. DEFINITIE. PRIMREC is de klasse van *primitief recursieve* functies van $\mathbb{N}$ naar $\mathbb{N}$, inductief gedefinieerd door:

(i) PRIMREC bevat de *initiële* functies U_i^p (projectie-functies), Succ, Z gedefinieerd door:

$U_i^p(n_1,...,n_p) = n_i \quad (1 \leq i \leq p)$
$Succ(n) = n + 1$
$Z(n) = 0.$

(ii) PRIMREC is gesloten onder *compositie*, d.w.z. als $\chi, \psi_1,..., \psi_m \in$ PRIMREC, dan

$\varphi(\mathbf{n}) = \chi(\psi_1(\mathbf{n}),...,\psi_m(\mathbf{n})) \in$ PRIMREC.

(**n** staat voor $n_1,...,n_p$, $p \geq 1$)

(iii) PRIMREC is gesloten onder *primitieve recursie*, d.w.z. als $\chi, \psi \in$ PRIMREC, dan is φ gedefinieerd door

$\varphi(0, \mathbf{n}) = \chi(\mathbf{n})$
$\varphi(k+1, \mathbf{n}) = \psi(\varphi(k, \mathbf{n}), k, \mathbf{n})$

in PRIMREC. Hier is $\mathbf{n} = n_1,...,n_p$, $p \geq 0$, de *parameters* van de primitieve recursie. Het geval zonder parameters ($p = 0$) is

$\varphi(0) = q$
$\varphi(k+1) = \psi(\varphi(k), k)$

met q een constante.

1.1. VOORBEELD. Zij j(b, a) = a + b. Volgens de bekende recursieve definitie hebben we a + 0 = a, a + Succ(b) = Succ(a + b), i.e.

$$\varphi(0, a) = a$$
$$\varphi(\mathrm{Succ}(b), a) = \mathrm{Succ}(\varphi(b, a)),$$

ook te schrijven als

$$\varphi(0, a) = U_1^1(a)$$
$$\varphi(\mathrm{Succ}(b), a) = \mathrm{Succ}(U_1^3(\varphi(b,a),b,a))$$

en dit past in het schema van primitieve recursie, omdat de functie

$$\lambda(n_1, n_2, n_3).\ \mathrm{Succ}(U_1^3(\, n_1, n_2, n_3))$$

primitief recursief is volgens het compositieschema (ii). (Hier is λ de 'meta-lambda' ter onderscheid van λ als λ-calculus symbool.)

1.2. OPGAVE. Bewijs dat de volgende functies van $\mathbb{N}$ naar $\mathbb{N}$ primitief recursief zijn:

(i) φ' met $\varphi'(a,b) = a + b$ (dus $\varphi' = \lambda(a,b).\ (a + b)$)

(ii) $a \cdot b$

(iii) a^b

(iv) $a!$

(v) pred (de predecessor functie)

Een nadere analyse van primitieve recursie laat zien dat er verbazend veel functies al met dit eenvoudige middel te definiëren zijn. (Zie bijv. 'Introduction to metamathematics', Kleene, p.220 e.v., of 'Introduction to mathematical logic', Mendelson.) We noemen er nog enkele: $a \cdot b$, $\min(a,b)$, $\min(a_1,...,a_n)$, $\max(a_1,...,a_n)$, $\lambda i.\ p_i$ (het i +1-de priemgetal), de Fibonacci functie φ gedefinieerd door $\varphi(0) = 0$, $\varphi(1) = 1$, $\varphi(n+2) = \varphi(n+1) + \varphi(n)$. Van de laatste twee is niet zomaar duidelijk hoe ze te definieren zijn met bovenstaande schema's (i)-(iii) voor primitieve recursie.

In de dertiger jaren rees de vraag of er nog andere vormen van recursie zijn die buiten het kader van de primitief recursieve functies treden. Ackermann gaf in 1928 een bevestigend antwoord (Kleene, IM p.271): beschouw

$\xi_0(b, a) = a + b$

$\xi_1(b, a) = a \cdot b$

$\xi_2(b, a) = a^b$

en breid deze rij functies uit door successievelijke primitieve recursies

$$\xi_{n+1}(0, a) = a$$
$$\xi_{n+1}(b+1, a) = \xi_n(\xi_{n+1}(b, a), a) \quad (n \geq 2).$$

Beschouw nu $\xi_n(b, a)$ als ternaire functie $\xi(n, b, a)$, en zij α de primitief recursieve functie gedefinieerd door $\alpha(0, a) = 0$, $\alpha(1, a) = 1$, $\alpha(n+2, a) = a$. Dan wordt ξ gedefinieerd door het volgende recursiestelsel:

$$\xi(0, b, a) = a + b$$
$$\xi(n+1, 0, a) = \alpha(n, a)$$
$$\xi(n+1, b+1, a) = \xi(n, \xi(n+1, b, a), a).$$

Merk op dat hier in tegenstelling tot de situatie bij primitieve recursie 'dubbele recursie' (i.e. op twee variabelen tegelijkertijd) optreedt. Ackermann toonde nu aan dat ξ niet primitief recursief kan zijn, omdat de unaire functie ψ gedefinieerd door $\psi(a) = \xi(a, a, a)$ dan ook primitief recursief zou zijn; en van ψ bewees hij dat deze functie sneller groeit dan welke unaire primitief recursieve functie ook. Dit voorbeeld werd in 1935 vereenvoudigd door Péter tot de functie die tegenwoordig als Ackermann's functie bekend staat.

Het is gebleken dat toevoeging van één enkel recursieschema aan die voor primitieve recursie een drastische uitbreiding geeft, waaronder ook functies als die van Ackermann vallen: het schema van *minimalisatie*. Met dit schema erbij krijgen we de *recursieve* functies, en men mag volgens de *these van Church* aannemen dat dit alle in enigerlei zin berekenbare totale functies zijn van $\mathbb{N}$ naar $\mathbb{N}$.

Notatie. Zij P(m) een unair predicaat op $\mathbb{N}$. Dan is $\mu m\ [P(m)]$ de kleinste m zodat P(m) geldt, als zo'n m bestaat; anders is $\mu m\ [P(m)]$ ongedefinieerd.

2. DEFINITIE. De klasse REC van *recursieve* functies van $\mathbb{N}$ naar $\mathbb{N}$ is als volgt inductief gedefinieerd:
(i), (ii), (iii) als bij PRIMREC;
(iv) als $\chi \in$ REC de eigenschap

$$\forall \mathbf{n}\ \exists m\ \chi(\mathbf{n}, m) = 0 \quad (*)$$

heeft, dan is φ gedefinieerd door

$$\varphi(\mathbf{n}) = \mu m\,[\chi(\mathbf{n},m) = 0]$$

in REC.

Opmerking: Als in (iv) de restrictie dat χ eigenschap (*) heeft wordt weggelaten, krijgen we de klasse van de zg. *partiëel recursieve* functies.

In de jaren 1930 is nu bewezen dat de partieel recursieve functies precies samenvallen met de in λ-calculus of Combinatorische Logica (CL) definieerbare functies, en dat de recursieve functies precies samenvallen met de in λ-calculus definieerbare totale functies. Dit laatste feit zullen we nu gaan bewijzen. We volgen de methode beschreven in Barendregt's boek.

3. DEFINITIE.. (i) *(Booleans)*
T ≡ λxy.x (≡ **K**) en **F** ≡ λxy.y. Merk op dat **KI** ↠ **F**, en dat

$$\mathbf{T}MN \twoheadrightarrow M,\ \mathbf{F}MN \twoheadrightarrow N.$$

(ii) *(Conditional)* Zij B een λ-term met 'waarde' een boolean. Dan schrijven we voor BMN:

if B *then* M *else* N.

Volgens (i) is dit inderdaad conform de betekenis van de conditional.

(iii) *(Paring)* $[M, N] \equiv \lambda z.zMN$
$(M)_0 \equiv M\mathbf{T}$
$(M)_1 \equiv M\mathbf{F}$

Nu geldt $([M, N])_0 \twoheadrightarrow M$ en $([M, N])_1 \twoheadrightarrow N$.

(iv) *(Numerals)*
$0 \equiv \mathbf{I}$
$n+1 \equiv [\mathbf{F}, n]$

Merk op dat de numerals onderling verschillende normaalvormen zijn.

(v) *(Successor, predecessor, test for zero)*

$\mathbf{Succ} \equiv \lambda x.[\mathbf{F}, x]$
$\mathbf{Pred} \equiv \lambda x.x\mathbf{F}$
$\mathbf{Zero} \equiv \lambda x.x\mathbf{T}$

Nu is **Succ** n = n +1, **Pred** n +1 = n, **Zero** 0 = **T**, **Zero** n+1 = **F**.
We gebruiken de volgende afkorting:

$M^+ \equiv \mathbf{Succ}\ M$

$M^- \equiv \mathbf{Pred}\ M.$

4. DEFINITIE. Zij φ een functie van $\mathbb{N}^p$ naar $\mathbb{N}$. Dan is φ λ-*definieerbaar* als er een λ-term F is zodat

$$\forall n_1,...,n_p \in \mathbb{N} \quad F\ n_1\ ...\ n_p = \varphi(n_1,...,n_p)$$

In dat geval zeggen we dat φ λ-gedefinieerd wordt door F. Merk op dat uit de Church-Rosser stelling en het feit dat numerals in normaalvorm zijn, volgt dat we zelfs hebben:

$$F\ n_1\ ...\ n_p \twoheadrightarrow \varphi(n_1,...,n_p)$$

Gebruik makend van de vector notatie $\mathbf{n}$ voor $n_1,...,n_p$ kunnen we schrijven:

$$F\mathbf{n} = \varphi(\mathbf{n})$$

5. LEMMA. *Alle primitief recursieve functies zijn λ-definieerbaar.*

BEWIJS. (i) De initiële functies U_i^p, Succ en Z zijn λ-definieerbaar, door respectievelijk:

$\mathbf{U}_i^p \equiv \lambda x_1...x_p.\ x_i$
$\mathbf{Succ} \equiv \lambda x.[\mathbf{F}, x]$
$\mathbf{Z} \equiv \lambda x.\ 0.$

(ii) De λ-definieerbare functies zijn gesloten onder compositie: laten namelijk χ, ψ_1,..., ψ_m λ-gedefinieerd worden door resp. G, H_1,..., H_m. Dan wordt

$$\varphi(\mathbf{n}) = \chi(\psi_1(\mathbf{n}),...,\psi_m(\mathbf{n}))$$

λ-gedefinieerd door

$$F \equiv \lambda \mathbf{x}.\ G(H_1\mathbf{x})...(H_m\mathbf{x}).$$

(iii) De λ-definieerbare functies zijn gesloten onder het schema van primitieve recursie:
Zij φ gedefinieerd door

$$\varphi(0, \mathbf{n}) = \chi(\mathbf{n})$$
$$\varphi(k+1, \mathbf{n}) = \psi(\varphi(k, \mathbf{n}), k, \mathbf{n})$$

waarbij χ en ψ λ-gedefinieerd zijn door resp. G en H. Laat nu F een λ-term zijn zodat

$$F\mathbf{xy} = \textit{if } \textbf{Zero } x \textit{ then } G\mathbf{y} \textit{ else } H(Fx^{-}\mathbf{y})x^{-}\mathbf{y}.$$

Met de fixed point stelling is zo'n F inderdaad te vinden. Nu volgt met inductie naar k dat

$$F\ k\ \mathbf{n} = \varphi(k, \mathbf{n})$$

Dus wordt F λ-gedefinieerd door φ.

Uit (i)-(iii) en de definitie van PRIMREC volgt dat elke primitief recursieve functie λ-definieerbaar is.

□

Uit het vorige hoofdstuk herinneren we ons de definitie van *Turing's fixed point combinator*

$$\Theta \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy)),$$

met de eigenschap dat voor elke λ-term F:

$$\Theta F \twoheadrightarrow F(\Theta F).$$

Om nu *alle* recursieve functies te λ-definieren, moeten we ook het minimalisatieschema ((iv) in Definitie 2) in λ-calculus representeren. Hiertoe definieren we voor elke λ-term P:

$$H_P \equiv \Theta(\lambda hz.\ \textit{if } Pz \textit{ then } z \textit{ else } hz^{+})$$

$$\mu P \equiv H_P 0.$$

Uit de definitie van H_P volgt direct

$$\mathrm{H}Pz \twoheadrightarrow \textit{if}\ Pz\ \textit{then}\ z\ \textit{else}\ \mathrm{H}Pz^{+}.$$

Stel nu dat voor elke $n \in \mathbb{N}$ geldt: $Pn = \mathbf{T}$ of $Pn = \mathbf{F}$, en bovendien dat $\exists n\,(Pn = \mathbf{T})$.
Zij $m = \mu n[Pn = \mathbf{T}]$. Dan geldt $\mu P = m$.

6. OPGAVE. Bewijs deze laatste bewering.

7. STELLING. *Elke recursieve functie is λ-definieerbaar.*

BEWIJS. We hoeven alleen nog te bewijzen dat het schema van minimalisatie in λ-calculus representeerbaar is. Zij dus

$$\varphi(\mathbf{n}) = \mu m[\chi(\mathbf{n},m) = 0]$$

waarbij χ λ-gedefinieerd wordt door G. Dan wordt φ λ-gedefinieerd door F met

$$F\mathbf{x} = \mu[\lambda y.\mathbf{Zero}(G\mathbf{x}y)].$$

□

7.1. OPMERKING. Ook het omgekeerde van Stelling 7 geldt: elke λ-definieerbare functie is recursief. Dit zullen we hier niet bewijzen; het vereist nog heel wat werk.

8. Numeral systemen.

Het is opmerkelijk dat er maar heel weinig eigenschappen van de numerals nodig zijn om Stelling 7 te bewijzen. Sectie 6.4 uit Barendregt's boek laat zien hoe Stelling 7 overgedragen kan worden naar andere numeral systemen, o.a. dat van de *Church's numerals*. (Zie bijlage.)

8.1. OPGAVE. (Exercise 6.8.6 in boek Barendregt; afkomstig van Rosser.)
Definieer termen $A_+, A_{\mathbf{x}}, A_{exp}$ als volgt:

$$A_+\ xy = \lambda pq.xp(ypq)$$

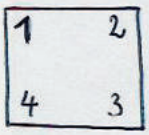
$$A_{\mathbf{x}}\ xy = x \circ y$$

$$A_{exp}\ xy = yx.$$

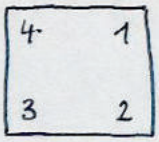
(Hierbij is $M \circ N \equiv \mathbf{B}MN$ met $\mathbf{B} \equiv \lambda xyz.x(yz).$)

Bewijs dat t.o.v. Church's numerals $\mathbf{c}_n \equiv \lambda fx.f^n(x)$ de functies optelling, vermenigvuldiging en machtsverheffing λ-gedefinieerd worden door resp. $A_+, A_{\mathbf{x}}, A_{exp}$ (uitgezonderd $\mathbf{c}_0\mathbf{c}_n$).

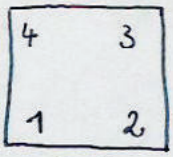
Composition of symmetries of the square



$[\![\lambda]\!]$



$[\![R]\!]$



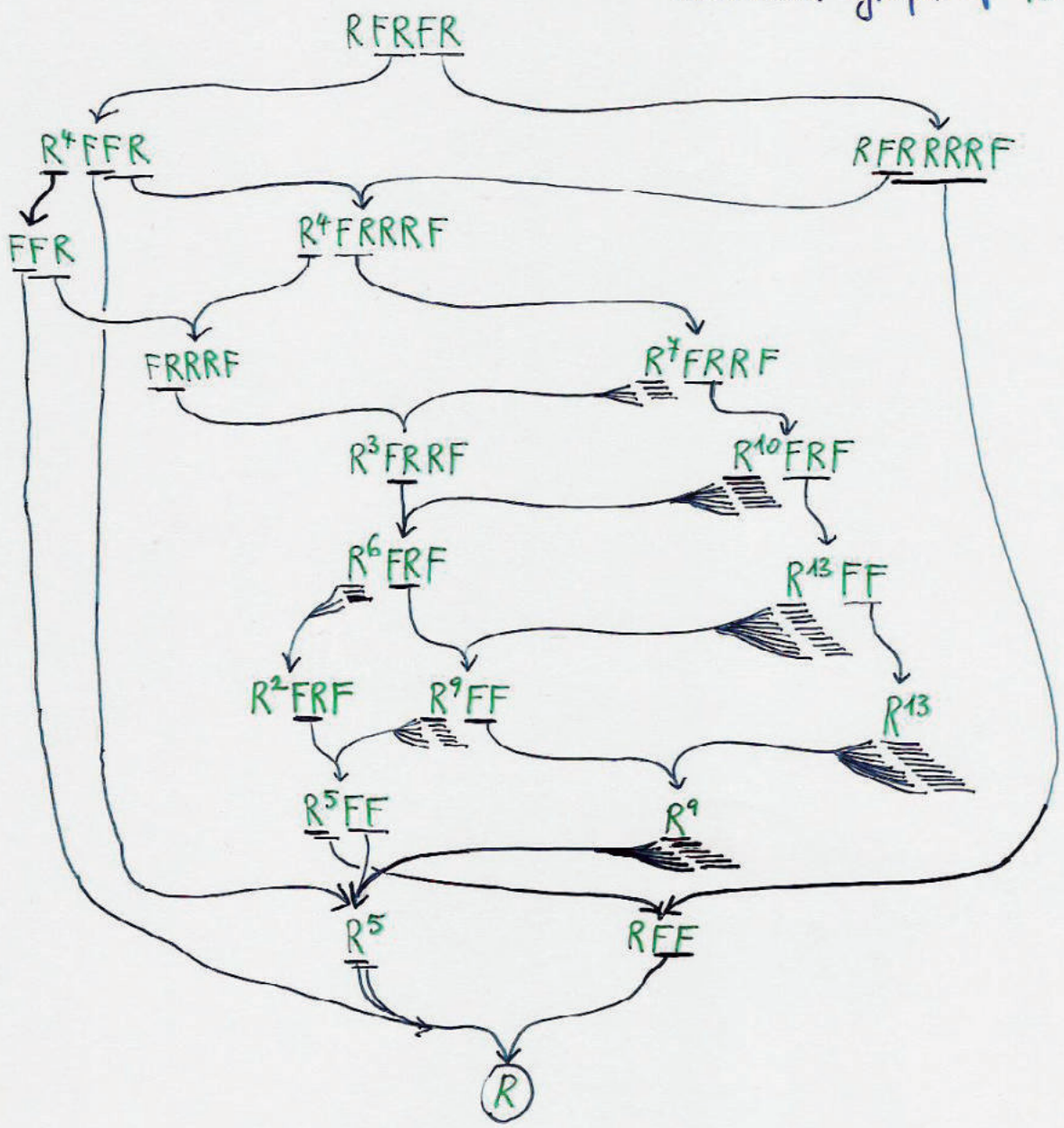
$[\![F]\!]$

$\cdot$	λ	R	RR	RRR	F	RF	RRF	RRRF
λ	λ	R	RR	RRR	F	RF	RRF	RRRF
R	R	RR	RRR	λ	RF	RRF	RRRF	F
RR	RR	RRR	λ	R	RRF	RRRF	F	RF
RRR	RRR	λ	R	RR	RRRF	F	RF	RRF
F	F	RRRF	RRF	RF	λ	RRR	RR	R
RF	RF	F	RRRF	RRF	R	λ	RRR	RR
RRF	RRF	RF	F	RRRF	RR	R	λ	RRR
RRRF	RRRF	RRF	RF	F	RRR	RR	R	λ

A Reduction Graph:

SRS $\begin{cases} FF \to \lambda \\ RRRR \to \lambda \\ FR \to RRRF \end{cases}$

Reduction Graph of RFRFR



THE N QUEEN PROBLEM

Het volgende probleem is een klassiek probleem in functioneel programmeren: plaats 8 koninginnen op een schaakbord zodanig dat geen koningin een andere kan slaan. Het volgende Miranda programma is van Stefan Blom. Afgezien van de in Miranda voorgedefinieerde manoeuvre 'filter' is dit een orthogonale TRS. De orthogonaliteit is eenvoudig te zien.

De TRS is ook SN. De term `queens 8` geeft als unieke normaalvorm een lijst van 92 posities als gevraagd. Hierbij zijn alle 'goede' posities verkregen, ook al zijn sommige posities bv. elkaars gespiegelde en dus niet ‘essentieel’ verschillend.

Merk op dat de TRS (of het Miranda programma) de applicatieve notatie gebruikt, net als in CL.

queens n geeft alle oplossingen voor het n-queens probleem.
Bij de recursieve aanroep moeten we weten welk probleem we oplossen en dus moeten we de n(m) mee geven. safe is een incrementele variant en checkt alleen het laatst toegevoegde element. safe1 zoekt het laatste element op en telt de lengte. safe2 bouwt vervolgens het product.

```
queens n = qw n n

qw m 0 = [[]]
qw m (n+1) = filter safe (extend m (qw m n) m)


extend :: num->[[num]]->num->[[num]]
extend m [] n = []
extend m (a:x) 0 = extend m x m
extend m (a:x) (n+1) = (a++[n+1]) : (extend m (a:x) n)
```

Hierbij is [] het lege rijtje, [a] het singleton rijtje met alleen een a; ++ is concatenatie; (a:x) is het rijtje x voorafgegaan door a.

Deze regels zijn het best te begrijpen door een boom te tekenen van de stadia die de berekening doorloopt. Op bv. het derde nivo van de boom komen de posities ter lengte 3 te staan die safe zijn. In het volgend stadium worden al deze elementen (posities) uitgebreid met de getallen 1,...,8 en wordt deze rij weer gefilterd op het safe predikaat.

```
safe x = (safe1 0 [] x ~= 0)
```

Dit wil zeggen dat het predikaat safe gedefinieerd is als:
safe1 0 [] x is ongelijk 0.
Verder is filter als volgt gedefinieerd:

```
filter safe [] = []
filter safe (a:x) = if safe a then a:(filter safe x) else filter
safe x.
```

Dus filter safe loopt door een rij heen en verwijdert alle elementen van de rij die niet safe zijn.
De if-then-else is ook gemakkelijk in orthogonale TRS regels te schrijven; we kunnen ook direct gebruik maken van een operator testzero:

```
testzero 0 x y  = x
testzero (succ z) x y = y

safe1 :: num -> [num] -> [num]-> num
safe1 n x [b] = safe2 (n+1) 1 1 x b
safe1 n x (b:y) = safe1 (n+1) (x++[b]) y

safe2 :: num -> num -> num ->[num]->num->num
safe2 n i p [] b  = p
safe2 n i p (a:x) b
 = safe2 n (i+1) (p*(a-b)*((a-b)^2-(n-i)^2)) x b
```

Het product dat hier wordt opgebouwd heeft de volgende bedoeling. Beschouw een positie $(x_1,x_2,\ldots x_n)$. Als hierin voor zekere i,j $(x_i - x_j) = 0$, dan is de positie niet safe, want de koninginnen in i-de resp. j-de kolom kunnen elkaar kennelijk horizontaal slaan. Als verder $x_i - x_j$ in absolute waarde gelijk is aan i - j in absolute waarde, dan kunnen deze koninginnen elkaar diagonaal slaan. Equivalent: als $(x_i - x_j)^2 = (i - j)^2$. Als we dus voor alle i en j het product vormen van de factoren $(x_i - x_j)$ en $((x_i - x_j)^2 - (i - j)^2)$, dan geeft het al of niet 0 zijn van dit product juist aan of de positie safe is of niet: namelijk safe als het product niet 0 is.

De werking van deze regels is te begrijpen door bv. safe1 0 [] (1,3,5,7) te berekenen.

Merk op dat de TRS mbv RPO terminerend bewezen kan worden; dit echt doen zou erg omslachtig worden door de aanwezigheid van het product in de definitie van safe2. Maar het is gemakkelijk in te zien dat het zou kunnen.

Het resultaat is de volgende normaalvorm:

Miranda queens 8

[[8,4,1,3,6,2,7,5],[8,3,1,6,2,5,7,4],[8,2,5,3,1,7,4,6],[8,2,4,1,7,5,3,6],[7,5,3,1,6,8,2,4],[7,4,2,8,6,1,3,5],[7,4,2,5,8,1,3,6],[7,3,8,2,5,1,6,4],[7,3,1,6,8,5,2,4],[7,2,6,3,1,4,8,5],[7,2,4,1,8,5,3,6],[7,1,3,8,6,4,2,5],[6,8,2,4,1,7,5,3],[6,4,7,1,8,2,5,3],[6,4,7,1,3,5,2,8],[6,4,2,8,5,7,1,3],[6,4,1,5,8,2,7,3],[6,3,7,4,1,8,2,5],[6,3,7,2,8,5,1,4],[6,3,7,2,4,8,1,5],[6,3,5,8,1,4,2,7],[6,3,5,7,1,4,2,8],[6,3,1,8,5,2,4,7],[6,3,1,8,4,2,7,5],[6,3,1,7,5,8,2,4],[6,2,7,1,4,8,5,3],[6,2,7,1,3,5,8,4],[6,1,5,2,8,3,7,4],[5,8,4,1,7,2,6,3],[5,8,4,1,3,6,2,7],[5,7,4,1,3,8,6,2],[5,7,2,6,3,1,8,4],[5,7,2,6,3,1,4,8],[5,7,2,4,8,1,3,6],[5,7,1,4,2,8,6,3],[5,7,1,3,8,6,4,2],[5,3,8,4,7,1,6,2],[5,3,1,7,2,8,6,4],[5,3,1,6,8,2,4,7],[5,2,8,1,4,7,3,6],[5,2,6,1,7,4,8,3],[5,2,4,7,3,8,6,1],[5,2,4,6,8,3,1,7],[5,1,8,6,3,7,2,4],[5,1,8,4,2,7,3,6],[5,1,4,6,8,2,7,3],[4,8,5,3,1,7,2,6],[4,8,1,5,7,2,6,3],[4,8,1,3,6,2,7,5],[4,7,5,3,1,6,8,2],[4,7,5,2,6,1,3,8],[4,7,3,8,2,5,1,6],[4,7,1,8,5,2,6,3],[4,6,8,3,1,7,5,2],[4,6,8,2,7,1,3,5],[4,6,1,5,2,8,3,7],[4,2,8,6,1,3,5,7],[4,2,8,5,7,1,3,6],[4,2,7,5,1,8,6,3],[4,2,7,3,6,8,5,1],[4,2,7,3,6,8,1,5],[4,2,5,8,6,1,3,7],[4,1,5,8,6,3,7,2],[4,1,5,8,2,7,3,6],[3,8,4,7,1,6,2,5],[3,7,2,8,6,4,1,5],[3,7,2,8,5,1,4,6],[3,6,8,2,4,1,7,5],[3,6,8,1,5,7,2,4],[3,6,8,1,4,7,5,2],[3,6,4,2,8,5,7,1],[3,6,4,1,8,5,7,2],[3,6,2,7,5,1,8,4],[3,6,2,7,1,4,8,5],[3,6,2,5,8,1,7,4],[3,5,8,4,1,7,2,6],[3,5,7,1,4,2,8,6],[3,5,2,8,6,4,7,1],[3,5,2,8,1,7,4,6],[3,1,7,5,8,2,4,6],[2,8,6,1,3,5,7,4],[2,7,5,8,1,4,6,3],[2,7,3,6,8,5,1,4],[2,6,8,3,1,4,7,5],[2,6,1,7,4,8,3,5],[2,5,7,4,1,8,6,3],[2,5,7,1,3,8,6,4],[2,4,6,8,3,1,7,5],[1,7,5,8,2,4,6,3],[1,7,4,6,8,2,5,3],[1,6,8,3,7,4,2,5],[1,5,8,6,3,7,2,4]]

LAMBDA DEFINIEERBAARHEID

In dit hoofdstuk zullen we bewijzen dat de klasse van *recursieve functies* precies samenvalt met de functies die in λ-calculus defineerbaar zijn. We zullen ons beperken tot *totale* functies van $\mathbb{N}$ naar $\mathbb{N}$ en houden ons dus niet met de grotere klasse van partiëel recursieve functies (die ook in λ-calculus definieerbaar zijn) bezig. We voeren eerst de *primitief recursieve* functies in.

1. DEFINITIE. PRIMREC is de klasse van *primitief recursieve* functies van $\mathbb{N}$ naar $\mathbb{N}$, inductief gedefinieerd door:

(i) PRIMREC bevat de *initiële* functies U_i^p (projectie-functies), Succ, Z gedefinieerd door:

$U_i^p(n_1,...,n_p) = n_i \quad (1 \leq i \leq p)$
$Succ(n) = n + 1$
$Z(n) = 0.$

(ii) PRIMREC is gesloten onder *compositie*, d.w.z. als $\chi, \psi_1,..., \psi_m \in$ PRIMREC, dan

$\varphi(\mathbf{n}) = \chi(\psi_1(\mathbf{n}),...,\psi_m(\mathbf{n})) \in$ PRIMREC.

(**n** staat voor $n_1,...,n_p$, $p \geq 1$)

(iii) PRIMREC is gesloten onder *primitieve recursie*, d.w.z. als $\chi, \psi \in$ PRIMREC, dan is φ gedefinieerd door

$\varphi(0, \mathbf{n}) = \chi(\mathbf{n})$
$\varphi(k+1, \mathbf{n}) = \psi(\varphi(k, \mathbf{n}), k, \mathbf{n})$

in PRIMREC. Hier is $\mathbf{n} = n_1,...,n_p$, $p \geq 0$, de *parameters* van de primitieve recursie. Het geval zonder parameters ($p = 0$) is

$\varphi(0) = q$
$\varphi(k+1) = \psi(\varphi(k), k)$

met q een constante.

1.1. VOORBEELD. Zij j(b, a) = a + b. Volgens de bekende recursieve definitie hebben we a + 0 = a, a + Succ(b) = Succ(a + b), i.e.

$$\varphi(0, a) = a$$
$$\varphi(\mathrm{Succ}(b), a) = \mathrm{Succ}(\varphi(b, a)),$$

ook te schrijven als

$$\varphi(0, a) = U_1^1(a)$$
$$\varphi(\mathrm{Succ}(b), a) = \mathrm{Succ}(U_1^3(\varphi(b,a),b,a))$$

en dit past in het schema van primitieve recursie, omdat de functie

$$\lambda(n_1, n_2, n_3).\ \mathrm{Succ}(U_1^3(\, n_1, n_2, n_3))$$

primitief recursief is volgens het compositieschema (ii). (Hier is λ de 'meta-lambda' ter onderscheid van λ als λ-calculus symbool.)

1.2. OPGAVE. Bewijs dat de volgende functies van $\mathbb{N}$ naar $\mathbb{N}$ primitief recursief zijn:

(i) φ' met $\varphi'(a,b) = a + b$ (dus $\varphi' = \lambda(a,b).\ (a + b)$)

(ii) $a \cdot b$

(iii) a^b

(iv) $a!$

(v) pred (de predecessor functie)

Een nadere analyse van primitieve recursie laat zien dat er verbazend veel functies al met dit eenvoudige middel te definiëren zijn. (Zie bijv. 'Introduction to metamathematics', Kleene, p.220 e.v., of 'Introduction to mathematical logic', Mendelson.) We noemen er nog enkele: $a \cdot b$, $\min(a,b)$, $\min(a_1,...,a_n)$, $\max(a_1,...,a_n)$, $\lambda i.\ p_i$ (het i +1-de priemgetal), de Fibonacci functie φ gedefinieerd door $\varphi(0) = 0$, $\varphi(1) = 1$, $\varphi(n+2) = \varphi(n+1) + \varphi(n)$. Van de laatste twee is niet zomaar duidelijk hoe ze te definieren zijn met bovenstaande schema's (i)-(iii) voor primitieve recursie.

In de dertiger jaren rees de vraag of er nog andere vormen van recursie zijn die buiten het kader van de primitief recursieve functies treden. Ackermann gaf in 1928 een bevestigend antwoord (Kleene, IM p.271): beschouw

$\xi_0(b, a) = a + b$

$\xi_1(b, a) = a \cdot b$

$\xi_2(b, a) = a^b$

en breid deze rij functies uit door successievelijke primitieve recursies

$$\xi_{n+1}(0, a) = a$$
$$\xi_{n+1}(b+1, a) = \xi_n(\xi_{n+1}(b, a), a) \quad (n \geq 2).$$

Beschouw nu $\xi_n(b, a)$ als ternaire functie $\xi(n, b, a)$, en zij α de primitief recursieve functie gedefinieerd door $\alpha(0, a) = 0$, $\alpha(1, a) = 1$, $\alpha(n+2, a) = a$. Dan wordt ξ gedefinieerd door het volgende recursiestelsel:

$$\xi(0, b, a) = a + b$$
$$\xi(n+1, 0, a) = \alpha(n, a)$$
$$\xi(n+1, b+1, a) = \xi(n, \xi(n+1, b, a), a).$$

Merk op dat hier in tegenstelling tot de situatie bij primitieve recursie 'dubbele recursie' (i.e. op twee variabelen tegelijkertijd) optreedt. Ackermann toonde nu aan dat ξ niet primitief recursief kan zijn, omdat de unaire functie ψ gedefinieerd door $\psi(a) = \xi(a, a, a)$ dan ook primitief recursief zou zijn; en van ψ bewees hij dat deze functie sneller groeit dan welke unaire primitief recursieve functie ook. Dit voorbeeld werd in 1935 vereenvoudigd door Péter tot de functie die tegenwoordig als Ackermann's functie bekend staat.

Het is gebleken dat toevoeging van één enkel recursieschema aan die voor primitieve recursie een drastische uitbreiding geeft, waaronder ook functies als die van Ackermann vallen: het schema van *minimalisatie*. Met dit schema erbij krijgen we de *recursieve* functies, en men mag volgens de *these van Church* aannemen dat dit alle in enigerlei zin berekenbare totale functies zijn van $\mathbb{N}$ naar $\mathbb{N}$.

Notatie. Zij P(m) een unair predicaat op $\mathbb{N}$. Dan is $\mu m\ [P(m)]$ de kleinste m zodat P(m) geldt, als zo'n m bestaat; anders is $\mu m\ [P(m)]$ ongedefinieerd.

2. DEFINITIE. De klasse REC van *recursieve* functies van $\mathbb{N}$ naar $\mathbb{N}$ is als volgt inductief gedefinieerd:
(i), (ii), (iii) als bij PRIMREC;
(iv) als $\chi \in$ REC de eigenschap

$$\forall \mathbf{n}\ \exists m\ \chi(\mathbf{n}, m) = 0 \quad (*)$$

heeft, dan is φ gedefinieerd door

$$\varphi(\mathbf{n}) = \mu m\,[\chi(\mathbf{n},m) = 0]$$

in REC.

Opmerking: Als in (iv) de restrictie dat χ eigenschap (*) heeft wordt weggelaten, krijgen we de klasse van de zg. *partiëel recursieve* functies.

In de jaren 1930 is nu bewezen dat de partieel recursieve functies precies samenvallen met de in λ-calculus of Combinatorische Logica (CL) definieerbare functies, en dat de recursieve functies precies samenvallen met de in λ-calculus definieerbare totale functies. Dit laatste feit zullen we nu gaan bewijzen. We volgen de methode beschreven in Barendregt's boek.

3. DEFINITIE.. (i) *(Booleans)*
T ≡ λxy.x (≡ **K**) en **F** ≡ λxy.y. Merk op dat **KI** ↠ **F**, en dat

$$\mathbf{T}MN \twoheadrightarrow M,\ \mathbf{F}MN \twoheadrightarrow N.$$

(ii) *(Conditional)* Zij B een λ-term met 'waarde' een boolean. Dan schrijven we voor BMN:

if B *then* M *else* N.

Volgens (i) is dit inderdaad conform de betekenis van de conditional.

(iii) *(Paring)* $[M, N] \equiv \lambda z.zMN$
$(M)_0 \equiv M\mathbf{T}$
$(M)_1 \equiv M\mathbf{F}$

Nu geldt $([M, N])_0 \twoheadrightarrow M$ en $([M, N])_1 \twoheadrightarrow N$.

(iv) *(Numerals)*
$0 \equiv \mathbf{I}$
$n+1 \equiv [\mathbf{F}, n]$

Merk op dat de numerals onderling verschillende normaalvormen zijn.

(v) *(Successor, predecessor, test for zero)*

$\mathbf{Succ} \equiv \lambda x.[\mathbf{F}, x]$
$\mathbf{Pred} \equiv \lambda x.x\mathbf{F}$
$\mathbf{Zero} \equiv \lambda x.x\mathbf{T}$

Nu is **Succ** n = n +1, **Pred** n +1 = n, **Zero** 0 = **T**, **Zero** n+1 = **F**.
We gebruiken de volgende afkorting:

$M^+ \equiv \mathbf{Succ}\ M$

$M^- \equiv \mathbf{Pred}\ M.$

4. DEFINITIE. Zij φ een functie van $\mathbb{N}^p$ naar $\mathbb{N}$. Dan is φ λ-*definieerbaar* als er een λ-term F is zodat

$$\forall n_1,...,n_p \in \mathbb{N} \quad F\ n_1\ ...\ n_p = \varphi(n_1,...,n_p)$$

In dat geval zeggen we dat φ λ-gedefinieerd wordt door F. Merk op dat uit de Church-Rosser stelling en het feit dat numerals in normaalvorm zijn, volgt dat we zelfs hebben:

$$F\ n_1\ ...\ n_p \twoheadrightarrow \varphi(n_1,...,n_p)$$

Gebruik makend van de vector notatie $\mathbf{n}$ voor $n_1,...,n_p$ kunnen we schrijven:

$$F\mathbf{n} = \varphi(\mathbf{n})$$

5. LEMMA. *Alle primitief recursieve functies zijn λ-definieerbaar.*

BEWIJS. (i) De initiële functies U_i^p, Succ en Z zijn λ-definieerbaar, door respectievelijk:

$\mathbf{U}_i^p \equiv \lambda x_1...x_p.\ x_i$
$\mathbf{Succ} \equiv \lambda x.[\mathbf{F}, x]$
$\mathbf{Z} \equiv \lambda x.\ 0.$

(ii) De λ-definieerbare functies zijn gesloten onder compositie: laten namelijk χ, ψ_1,..., ψ_m λ-gedefinieerd worden door resp. G, H_1,..., H_m. Dan wordt

$$\varphi(\mathbf{n}) = \chi(\psi_1(\mathbf{n}),...,\psi_m(\mathbf{n}))$$

λ-gedefinieerd door

$$F \equiv \lambda \mathbf{x}.\ G(H_1\mathbf{x})...(H_m\mathbf{x}).$$

(iii) De λ-definieerbare functies zijn gesloten onder het schema van primitieve recursie:
Zij φ gedefinieerd door

$$\varphi(0, \mathbf{n}) = \chi(\mathbf{n})$$
$$\varphi(k+1, \mathbf{n}) = \psi(\varphi(k, \mathbf{n}), k, \mathbf{n})$$

waarbij χ en ψ λ-gedefinieerd zijn door resp. G en H. Laat nu F een λ-term zijn zodat

$$F\mathbf{xy} = \textit{if } \textbf{Zero } x \textit{ then } G\mathbf{y} \textit{ else } H(Fx^{-}\mathbf{y})x^{-}\mathbf{y}.$$

Met de fixed point stelling is zo'n F inderdaad te vinden. Nu volgt met inductie naar k dat

$$F\ k\ \mathbf{n} = \varphi(k, \mathbf{n})$$

Dus wordt F λ-gedefinieerd door φ.

Uit (i)-(iii) en de definitie van PRIMREC volgt dat elke primitief recursieve functie λ-definieerbaar is.

□

Uit het vorige hoofdstuk herinneren we ons de definitie van *Turing's fixed point combinator*

$$\Theta \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy)),$$

met de eigenschap dat voor elke λ-term F:

$$\Theta F \twoheadrightarrow F(\Theta F).$$

Om nu *alle* recursieve functies te λ-definieren, moeten we ook het minimalisatieschema ((iv) in Definitie 2) in λ-calculus representeren. Hiertoe definieren we voor elke λ-term P:

$$H_P \equiv \Theta(\lambda hz.\ \textit{if } Pz \textit{ then } z \textit{ else } hz^{+})$$

$$\mu P \equiv H_P 0.$$

Uit de definitie van H_P volgt direct

$$\mathrm{H}Pz \twoheadrightarrow \textit{if}\ Pz\ \textit{then}\ z\ \textit{else}\ \mathrm{H}Pz^{+}.$$

Stel nu dat voor elke $n \in \mathbb{N}$ geldt: $Pn = \mathbf{T}$ of $Pn = \mathbf{F}$, en bovendien dat $\exists n\,(Pn = \mathbf{T})$.
Zij $m = \mu n[Pn = \mathbf{T}]$. Dan geldt $\mu P = m$.

6. OPGAVE. Bewijs deze laatste bewering.

7. STELLING. *Elke recursieve functie is λ-definieerbaar.*

BEWIJS. We hoeven alleen nog te bewijzen dat het schema van minimalisatie in λ-calculus representeerbaar is. Zij dus

$$\varphi(\mathbf{n}) = \mu m[\chi(\mathbf{n},m) = 0]$$

waarbij χ λ-gedefinieerd wordt door G. Dan wordt φ λ-gedefinieerd door F met

$$F\mathbf{x} = \mu[\lambda y.\mathbf{Zero}(G\mathbf{x}y)].$$

□

7.1. OPMERKING. Ook het omgekeerde van Stelling 7 geldt: elke λ-definieerbare functie is recursief. Dit zullen we hier niet bewijzen; het vereist nog heel wat werk.

8. Numeral systemen.

Het is opmerkelijk dat er maar heel weinig eigenschappen van de numerals nodig zijn om Stelling 7 te bewijzen. Sectie 6.4 uit Barendregt's boek laat zien hoe Stelling 7 overgedragen kan worden naar andere numeral systemen, o.a. dat van de *Church's numerals*. (Zie bijlage.)

8.1. OPGAVE. (Exercise 6.8.6 in boek Barendregt; afkomstig van Rosser.)
Definieer termen $A_+, A_{\mathbf{x}}, A_{exp}$ als volgt:

$$A_+\ xy = \lambda pq.xp(ypq)$$

$$A_{\mathbf{x}}\ xy = x \circ y$$

$$A_{exp}\ xy = yx.$$

(Hierbij is $M \circ N \equiv \mathbf{B}MN$ met $\mathbf{B} \equiv \lambda xyz.x(yz).$)

Bewijs dat t.o.v. Church's numerals $\mathbf{c}_n \equiv \lambda fx.f^n(x)$ de functies optelling, vermenigvuldiging en machtsverheffing λ-gedefinieerd worden door resp. $A_+, A_{\mathbf{x}}, A_{exp}$ (uitgezonderd $\mathbf{c}_0\mathbf{c}_n$).