Universiteit Utrecht

Bachelorscriptie van 7,5 ECTS voor de opleiding Kunstmatige Intelligentie

---

Lindenmayer-systemen en turingmachines

De equivalentie van turingmachines en 2L-systemen

Door:
Mariëlle Rietdijk
(3636518)

Begeleider:
Rosalie Iemhoff

10 juli 2015

Inhoudsopgave

1 Introductie

In 1986 brengt Aristid Lindenmayer een artikel uit waarin hij een wiskundig model beschrijft om de groei van multicellulaire organismes te beschrijven en te simuleren [2]. Deze zogenoemde Lindenmayer-systemen zijn een specifiek type herschrijfgrammatica's: een systeem heeft een beginwoord waar regels op worden toegepast die beschrijven waar de symbolen in het woord naar herschreven moeten worden. De kern van een Lindenmayer-systeem is de verzameling regels die parallel worden toegepast - in tegenstelling tot andere grammatica's waar regels één voor één worden toegepast.

Volgens Kari et al. is het dat parallelisme dat Lindenmayer-systemen zo geschikt maakt om de ontwikkeling van verschillende levensvormen te beschrijven. Levende systemen kunnen uit miljoenen delen bestaan die zich allemaal parallel ontwikkelen. Om deze reden kunnen Lindenmayer-systemen erg nuttig zijn om (kunstmatige of imaginair) leven te simuleren [5].

De systemen worden tegenwoordig voor verschillende doeleinden gebruikt die niet strikt beperkt zijn tot de biologie. Lima de Campos et al. bijvoorbeeld gebruiken Lindenmayer-systemen als een model voor de ontwikkeling van neuronen en de connecties tussen neuronen in hun ontwerp van een methodologie om automatisch optimale ANN's ('Artificial Neural Networks') te genereren [4] - een voorbeeld van het gebruik en nut van Lindenmayer-systemen in de kunstmatige intelligentie.

Modellen waarin processen gesimuleerd en berekend kunnen worden, zoals Lindenmayer-systemen, nemen een belangrijke plaats in binnen de kunstmatige intelligentie. Het is daarom van belang om te weten wat de computationele kracht is van deze modellen - wat er theoretisch gezien allemaal berekend kan worden met welk model. We weten bijvoorbeeld welke talen verschillende soorten grammatica's kunnen beschrijven en dat we met turingmachines alles kunnen beschrijven wat we als berekenbaar beschouwen.

Van Dalen bewijst in 'A Note on Some Systems of Lindenmayer' [1] dat een specifiek soort Lindenmayer-systemen minstens zo sterk zijn als turingmachines. Een turingmachine kan dienen als een model voor een computer - alles wat een computer kan, kan een turingmachine ook. Door te bewijzen dat Lindenmayer-systemen minstens zo sterk zijn als turingmachines, weten we dat dit ook voor Lindenmayer-systemen geldt. Vanwege de Church-Turing-

hypothese die stelt dat iedere berekening op een turingmachine kan worden uitgevoerd, zou dit betekenen dat Lindenmayer-systemen equivalent zijn aan turingmachines.

We zullen in deze scriptie onderzoeken hoe de Lindenmayer-systemen die Van Dalen gebruikt zich verhouden tot de gebruikelijk definities van Lindenmayer-systemen en zijn bewijs onderzoeken over de computationele kracht van een specfiek soort Lindenmayer-systemen, in de hoop dat dit verheldering en context geeft met betrekking tot Van Dalens zeer compacte artikel.

Allereerst zullen we Lindenmayer-systemen introduceren, met formele definities, voorbeelden en een vergelijking met contextvrije grammatica's om duidelijk te maken wat de essentie is van een Lindenmayer-systeem. Vervolgens onderzoeken we hoe deze Lindenmayer-systemen zich verhouden tot de Lindenmayer-systemen waar Van Dalen over schrijft. Hetzelfde doen we in het daarop volgende hoofdstuk over turingmachines; hoe verhouden de turingmachines zoals Van Dalen die gebruikt zich tot de gebruikelijke turingmachines gedefinieerd door Sipser in 'Introduction to the Theory of Computation' [3]? Tenslotte zullen we het bewijs bespreken van de stelling dat een bepaald soort Lindenmayer-systemen minstens zoveel computationele kracht heeft als een turingmachine.

2 Lindenmayer-systemen

Er bestaan verschillende definities van Lindenmayer-systemen. We zullen de meest gangbare definitie van zowel de gebruikelijke contextvrije en minder gebruikelijke contextsensitieve Lindenmayer-systemen geven. Een vergelijking tussen contextvrije grammatica's en Lindenmayer-systemen zal duidelijker maken wat de essentie is van een Lindenmayer-systeem.

2.1 Definities

Er bestaan verschillende definities van Lindenmayer-systemen. Vaak wordt een Lindenmayer-systeem gedefinieerd als een tupel $\langle V, C, w, P\rangle$ waarbij V een eindig alfabet van variabelen is, C een eindig alfabet van constanten disjunct van V, $w \in (V \cup C)^*$ een beginwoord en P een verzameling productieregels.

Het hangt van het type Lindenmayer-systeem af hoe deze productieregels eruit zien. In principe zijn de regels een relatie $P \subseteq (V \times (V \cup C)^*)$. Een element uit deze relatie wordt vaak geschreven als $a \to u$. Aan de linkerkant van de pijl staat dus een variabele $a \in V$ en aan de rechterkant van de pijl een string bestaande uit variabelen en constanten ($u \in (V \cup C)^*$). Voor iedere variabele is er minstens één productieregel.

Beschouw een Lindenmayer-systeem $L = \langle V, C, w, P\rangle$, een string $u = u_0...u_n$ met $u_i \in (V \cup C)$ en een string $v = v_0...v_n$ met $v_i \in (V \cup C)^*$ waarbij $i = 0, ..., n$ en $n \geq 0$. De string v kan worden afgeleid uit u dan en slechts dan als voor alle i geldt dat $u_i \to v_i \in P$ wanneer $u_i \in V$ en $u_i = v_i$ wanneer $u_i \in C$. Alle constanten in u blijven dus staan en op iedere variabele in u wordt een regel uit P toegepast. We schrijven dan ook wel $u \Rightarrow v$, of $u \underset{L}{\Rightarrow} v$ wanneer we duidelijk willen maken dat het om een specifiek Lindenmayer-systeem L gaat. $u \overset{*}{\Rightarrow} v$ betekent dat er een $n \geq 0$ en een afleiding $u_0 \Rightarrow ... \Rightarrow u_n$ bestaat met $u_0 = u$ en $u_n = v$. De taal van een Lindenmayer-systeem L is $\mathcal{L}(L) = \{u \mid w \overset{*}{\Rightarrow} u\}$ - dat zijn dus alle woorden die afgeleid zijn uit het beginwoord. Merk op dat deze woorden ook variabelen mogen bevatten en dat het beginwoord ook tot de taal behoort.

voorbeeld Beschouw een Lindenmayer-systeem $L = \langle V, C, w, P\rangle$ met $V = \{A, B, C, D, E\}$, $C = \{a, b, c, d, e\}$ en $w = aAaA$. De productieregels zijn als

volgt:

$$P = \{A \to CBDB, B \to \epsilon, C \to EbcE, D \to Bd, E \to e\}$$

Om de taal van L te berekenen, beginnen we met het beginwoord - deze zit sowieso in de taal van L. Vervolgens zoeken we voor iedere variabele in het beginwoord een regel in P waar de variabele aan de linkerkant van de pijl staat en vervangen we de variabele door wat er aan de rechterkant van de pijl staat van die regel. Hetzelfde doen we voor het woord wat hieruit resulteert en zo verder. Dit leidt tot de volgende afleiding:

$$aAaA \Rightarrow aCBDBaCBDB \Rightarrow aEbcEBdaEbcEBd \Rightarrow aebcedaebced$$

In het laatste woord zitten geen variabelen meer, dus hier houdt de taal op. De taal van L is als volgt:

$$\mathcal{L}(L) = \{aAaA, aCBDBaCBDB, aEbcEBdaEbcEBd, aebcedaebced\}$$

□

Het bovenstaande Lindenmayer-systeem heeft een eindige taal, maar vaak hebben systemen een oneindige taal. Onderstaande is hier een voorbeeld van.

voorbeeld Beschouw een Lindenmayer-systeem $L = \langle V, C, w, P\rangle$ met $V = \{A\}$, $C = \emptyset$, $w = A$ en $P = \{A \to AAA\}$. De eerste woorden die het systeem genereert zijn A, AAA en $AAAAAAAAA$. De taal is oneindig groot en is te omschrijven als $\mathcal{L}(L) = \{(A)^{3^n} \mid n \geq 0\}$, waarbij $(A)^n$ een string van n A's is. □

De productieregels van Lindenmayer-systemen zijn vaak deterministisch. Dat houdt in dat er voor iedere variabele precies één productieregel is. Om deze reden wordt de verzameling productieregels vaak gezien als een functie $\delta : V \to (V \cup C)^*$ - een functie is namelijk per definitie deterministisch. δ wordt dan ook wel de transitiefunctie genoemd, in plaats van de verzameling productieregels. De hierboven gegeven voorbeelden van Lindenmayer-systemen zijn beide deterministisch.

De productieregels kunnen ook, in plaats van $P \subseteq V \times (V \cup C)^*$, van de vorm $P \subseteq V \times (V \cup C)^+$ zijn. Dit houdt in dat de regels in P niet van de vorm $a \to \epsilon$ met $a \in V$ kunnen zijn, wat wil zeggen dat variabelen niet kunnen worden afgebeeld op ϵ. Bij het berekenen van de taal van zo'n Lindenmayer-systeem kan het volgende woord dus nooit kleiner zijn dan zijn voorganger.

2.2 Contextsensitieve Lindenmayer-systemen

De hiervoor genoemde systemen zijn contextvrij. Echter gebruikt Van Dalen in zijn artikel een bepaald soort contextsensitieve systemen. In contextsensitieve systemen wordt er ook gekeken naar de context van een symbool wanneer een productieregel wordt gezocht om toe te passen op het symbool. De context van een symbool bestaat uit de symbolen die links en/of rechts van het symbool staan in een woord en bepaalt dus mede door welke string het symbool wordt vervangen in een afleidingsstap. De productieregels van zo'n systeem zien er dan ook anders uit: in plaats van slechts regels van de vorm $a \to u$ waarbij $a \in V$ en $u \in (V \cup C)^*$, hebben we nu ook te maken met regels van de vorm $v, a \to u$; $a, w \to u$ en $v, a, w \to u$ waarbij $v, w \in (V \cup C)^*$. De strings v en w zijn hier dus de context van het symbool a. P is dan als volgt gedefinieerd:

$$\begin{array}{rcllll} P & \subseteq & (& V\times & & (V \cup C)^*) \\ & \cup & ((V \cup C)^*\times & V\times & & (V \cup C)^*) \\ & \cup & (& V\times & (V \cup C)^*\times & (V \cup C)^*) \\ & \cup & ((V \cup C)^*\times & V\times & (V \cup C)^*\times & (V \cup C)^*) \end{array}$$

De berekening van de taal van een contextsensitief systeem is ook anders als die van contextvrije systemen. Beschouw een contextsensitief Lindenmayer-systeem $L = \langle V, C, w, P\rangle$, een string $u = u_0...u_n$ met $u_i \in (V \cup C)$ en een string $v = v_0...v_n$ met $v_i \in (V \cup C)^*$ waarbij $i = 0, ..., n$ en $n \geq 0$. De string v kan worden afgeleid uit u dan en slechts dan als voor alle i geldt dat $u_i = v_i$ wanneer $u_i \in C$ en als één van de volgende vier gevallen geldt wanneer $u_i \in V$:

$$\begin{array}{rrl} & u_i, & \to v_i \in P \\ w, & u_i, & \to v_i \in P \\ & u_i, \quad x & \to v_i \in P \\ w, & u_i, \quad x & \to v_i \in P \end{array}$$

waarbij $w, x \in (V \cup C)^*$. Hierbij geldt dat:

$w = \epsilon$ wanneer $i = 0$
$w = u_j...u_{i-1}$, met $0 \leq j \leq i - 1$ wanneer $i \neq 0$
$x = \epsilon$ wanneer $i = n$
$x = u_{i+1}...u_k$, met $i + 1 \leq k \leq n$ wanneer $i \neq n$.

voorbeeld Beschouw een contextsensitief Lindenmayer-systeem $L = \langle V, C, w, P \rangle$ met $V = \{A\}$, $C = \{a, b\}$, $w = AbA$ en $P = \{\epsilon, A, b \rightarrow a; A \rightarrow b\}$. Dan $\mathcal{L}(L) = \{AbA, abb, bbb\}$ □

Merk op dat bovenstaand systeem non-deterministisch is: op het eerste voorkomen van A in het beginwoord kunnen beide productieregels worden toegepast. Een manier om determinisme te garanderen, dus een manier om te voorkomen dat er een symbool kan zijn waarop meer dan één regel toepasbaar is, is door te bepalen naar welke context gekeken wordt (links, rechts, allebei en/of geen) en hoe groot de context gaat zijn waar naar gekeken wordt. Productieregels zijn dan slechts gedefinieerd met een context van die vorm en grootte. Onderstaand voorbeeld heeft productieregels waarbij naar de linker- en rechtercontext gekeken wordt en de context zowel links en rechts één symbool groot is.

voorbeeld Beschouw een contextsensitief Lindenmayer-systeem $L = \langle V, C, w, P \rangle$ met $V = \{A\}$, $C = \{a, b\}$, $w = AbA$ en $P = \{\epsilon, A, b \rightarrow a; b, A, \epsilon \rightarrow b\}$. Dan $\mathcal{L}(L) = \{AbA, abb\}$ □

2.3 Vergelijking met grammatica's

Zoals misschien al was opgevallen, hebben Lindenmayer-systemen een groot aantal overeenkomsten met grammatica's. Om beter te begrijpen wat de essentie is van een Lindenmayer-systeem, zullen we contextvrije grammatica's, zoals gedefinieerd in Sipser, vergelijken met contextvrije Lindenmayer-systemen.

Een contextvrije grammatica (CFG) is, net als een Lindenmayer-systeem, een tupel met vier elementen. Een contextvrije grammatica is een tupel $\langle V, C, S, P \rangle$, waarbij V, C en P hetzelfde gedefinieerd zijn als V, C en P in een (non-deterministisch) Lindenmayer-systeem. Het enige verschil in de definitie is dat een CFG in plaats van een beginwoord $w \in (V \cup C)^*$ een beginletter $S \in V$ heeft. Het beginwoord w van het Lindenmayer-systeem kan natuurlijk ook een $S \in V$ zijn, maar heeft meer mogelijkheden dan slechts een variabele als beginwoord. Echter maakt dit geen verschil; een CFG zou even zo goed gedefinieerd kunnen zijn met een beginwoord; dat wil zeggen: een CFG met een beginwoord heeft dezelfde computationele kracht als een CFG met een startvariabele.

stelling 1. *CFG's zijn equivalent aan CFG's met een beginwoord in plaats van een startvariabele.*

Bewijs. Het is triviaal dat er voor iedere CFG G een CFG G' is met een beginwoord in plaats van een startvariabele waarvoor geldt dat $\mathcal{L}(G) = \mathcal{L}(G')$, namelijk $G = G'$.

Voor iedere CFG $G = \langle V, C, w, P\rangle$ met een beginwoord is er een CFG $G' = \langle V', C, S, P'\rangle$ met een startvariabele zodanig dat $\mathcal{L}(G) = \mathcal{L}(G')$. S is hierbij een nieuwe variabele die niet in V zit en $V' = V \cup \{S\}$. De verzameling constanten blijft hetzelfde. De productieregels P' zijn hetzelfde als de productieregels P met daarbij een regel $S \to w$.

Neem een willekeurig woord $u \in \mathcal{L}(G)$. Er geldt $u \in C^*$ en $w \overset{*}{\underset{G}{\Rightarrow}} u$ (zie hieronder voor de definitie van de taal van een CFG). Omdat $S \to w \in P'$ en $P \subseteq P'$, geldt dat $S \underset{G'}{\Rightarrow} w$ en $w \overset{*}{\underset{G'}{\Rightarrow}} u$, dus $S \overset{*}{\underset{G'}{\Rightarrow}} u$ en $u \in \mathcal{L}(G')$.

Neem een willekeurig woord $u \in \mathcal{L}(G')$. Er geldt $u \in C^*$ en $S \overset{*}{\underset{G'}{\Rightarrow}} u$. We weten dat $S \to w \in P'$ de enige regel is in P' met aan de linkerkant van de pijl de startvariabele S, dus $S \underset{G'}{\Rightarrow} w$ en $w \overset{*}{\underset{G'}{\Rightarrow}} u$. Omdat P hetzelfde is als P' zonder $S \to w$, geldt ook $w \overset{*}{\underset{G}{\Rightarrow}} u$ en dus $u \in \mathcal{L}(G)$.

Omdat er voor iedere CFG met een startvariabele een CFG is met een beginwoord met dezelfde taal en andersom, geldt dat CFG's equivalent zijn aan CFG'S met een beginwoord in plaats van een startvariabele. □

Er dus geen verschil tussen een CFG en een Lindenmayer-systeem wanneer we kijken naar de tupels waaruit de systemen bestaan. De berekening van een Lindenmayer-systeem is echter wel anders dan die van een contextvrije grammatica. Het verschil zit hem er in dat een CFG één variabele per afleidingsstap vervangt, terwijl een Lindenmayer-systeem in één stap alle variabelen in een woord vervangt. Voor een CFG $G = \langle V, C, S, P\rangle$ geldt dat als $u, v \in (V \cup C)^*$ en $a \to w \in P$, dan $uav \Rightarrow uwv$. De definitie van $u \overset{*}{\Rightarrow} v$ is hetzelfde als bij een Lindenmayer-syteem. Het verschil in berekening is dus een verschil in de betekenis van $u \Rightarrow v$.

voorbeeld Beschouw het Lindenmayer-systeem $L = \langle V, C, w, P\rangle$ met $V =$

$\{A\}$, $C = \emptyset$, $w = A$ en $P = \{A \to AAA\}$ en een CFG G met dezelfde definitie. Voor L geldt $A \underset{L}{\Rightarrow} AAA \underset{L}{\Rightarrow} AAAAAAAAA$, maar voor G geldt $A \underset{G}{\Rightarrow} AAA \underset{G}{\Rightarrow} AAAAA \underset{G}{\Rightarrow} AAAAAAA \underset{G}{\Rightarrow} AAAAAAAAA$. □

De talen van Lindenmayer-systemen en CFG's zijn ook verschillend gedefinieerd. De taal van een CFG $G = \langle V, C, S, P\rangle$ is $\mathcal{L}(G) = \{u \in C^* \mid S \overset{*}{\Rightarrow} u\}$. Een Lindenmayer-systeem is minder streng, de taal van een systeem $L = \langle V, C, w, P\rangle$ is namelijk $\mathcal{L}(L) = \{u \mid w \overset{*}{\Rightarrow} u\}$. Het verschil is dat de woorden in de taal van een Lindenmayer-systeem ook variabelen mogen bevatten, bij een CFG mag dat niet.

voorbeeld Beschouw weer het Lindenmayer-systeem L en de CFG G uit het vorige voorbeeld. De taal van L is oneindig groot en is te omschrijven als $\mathcal{L}(L) = \{(A)^{3^n} \mid n \geq 0\}$. De taal van G is $\mathcal{L}(G) = \emptyset$ □

voorbeeld Beschouw het Lindenmayer-systeem $L = \langle V, C, w, P\rangle$ met $V = \{A, B, C\}$, $C = \{c\}$, $w = ABA$ en $P = \{A \to B, B \to C, C \to a\}$ en een CFG G met dezelfde definitie. $\mathcal{L}(L) = \{ABA, BCB, CcC, ccc\}$, maar $\mathcal{L}(G) = \{ccc\}$ □

3 Van Dalens Lindenmayer-systemen

Van Dalen bespreekt een drietal Lindenmayer-systeem. Het 2L-systeem is contextsensitief; de context van een symbool bestaat hier uit de enkele symbolen die direct links en rechts van het symbool staan. 1L-systemen zijn ook contextsensitief, maar beperkter; de context van een symbool is alleen het symbool direct links ervan. 0L-systemen zijn contextvrije Lindenmayer-systemen. 0L- en 1L-systemen worden niet gebruikt in het uiteindelijke bewijs in hoofdstuk 5, maar het is nuttig om deze toch te bespreken om het 2L-systeem beter te begrijpen.

De Lindenmayer-systemen zoals Van Dalen ze definieert lijken heel anders dan degene die we hierboven besproken hebben. In drie stappen zullen we de Lindenmayer-systemen zoals we die hierboven hebben gedefinieerd omvormen naar de Lindenmayer-systemen die Van Dalen gebruikt.

De productieregels van de systemen van Van Dalen zijn deterministisch. We zullen de productieregels daarom de functie δ noemen. Het bereik van de functie δ is bij alledrie de systemen $(V \cup C)^*$.

3.1 Lindenmayer-systemen zonder constanten

Allereerst gaan we van een Lindenmayer-systeem dat gedefinieerd is als een tupel met vier elementen naar een Lindenmayer-systeem dat gedefinieerd is als een tupel met drie elementen. Een Lindenmayer-systeem is nu een tupel $\langle \Sigma, w, \delta \rangle$ waarbij Σ een eindig alfabet is, $w \in \Sigma^*$ het beginwoord en δ de verzameling productieregels. In plaats van een verzameling variabelen V en een verzameling constanten C hebben we hier dus met slechts één verzameling symbolen te maken. We zullen naar deze systemen verwijzen als L^{Σ}-systemen.

De functie δ is gedefinieerd als $\Sigma \to \Sigma^*$ voor $0\mathrm{L}^{\Sigma}$-systemen. Een $1\mathrm{L}^{\Sigma}$-systeem beschouwt ook altijd de context van een symbool wanneer wordt bepaald waar een symbool op wordt afgebeeld. De context van een symbool $a \in \Sigma$ is altijd het symbool dat direct links van a staat in een woord. Deze context kan ook afwezig en dus het symbool ϵ zijn wanneer het symbool het meest linkse symbool van een woord is. Daarom is δ in $1\mathrm{L}^{\Sigma}$-systemen gedefinieerd

als $\Sigma^\epsilon \times \Sigma \to \Sigma^*$, waar $\Sigma^\epsilon = \Sigma \cup \{\epsilon\}$[1]. $\delta(a, b) = c$ met $a \in \Sigma^\epsilon$, $b \in \Sigma$ en $c \in \Sigma^*$ kan dan gelezen worden als: als a links van b staat, dan wordt b afgebeeld op c. In $2L^\Sigma$-systemen is δ als volgt gedefinieerd: $\Sigma^\epsilon \times \Sigma \times \Sigma^\epsilon \to \Sigma^*$[2]. $\delta(a, b, c) = d$ met $a, c \in \Sigma^\epsilon$, $b \in \Sigma$ en $d \in \Sigma^*$ kan dus gelezen worden als: als a links van b staat en c rechts van b staat, dan wordt b afgebeeld op d.

Beschouw een L^Σ-systeem $L = \langle \Sigma, w, \delta \rangle$, een string $u = u_0...u_n$ met $u_i \in \Sigma$ en een string $v = v_0...v_n$ met $v_i \in \Sigma^*$ waarbij $i = 0, ..., n$ en $n \geq 0$. Waneer L een $0L^\Sigma$-systeem is, geldt dat de string v kan worden afgeleid uit u ($u \Rightarrow v$) dan en slechts dan als voor alle i geldt dat $u_i \to v_i \in \delta$. Wanneer L een $1L^\Sigma$-systeem is geldt $u \Rightarrow v$ dan en slechts dan als voor alle i geldt dat $u_{i-1}, u_i \to v_i \in \delta$ wanneer $i \neq 0$ en $\epsilon, u_i \to v_i \in \delta$ wanneer $i = 0$. Ten slotte geldt dat als L een $2L^\Sigma$-systeem is dat $u \Rightarrow v$ dan en slechts dan als voor alle i geldt dat $u_{i-1}, u_i, u_{i+1} \to v_i \in \delta$ wanneer $i \neq 0$ en $i \neq n$; $\epsilon, u_i, u_{i+1} \to v_i \in \delta$ wanneer $i = 0$ en $i \neq n$; $u_{i-1}, u_i, \epsilon \to v_i \in \delta$ wanneer $i \neq 0$ en $i = n$ en $\epsilon, u_i, \epsilon \to v_i \in \delta$ wanneer $i = n = 0$.

In tegenstelling tot de gebruikelijke Lindenmayer-systemen zijn er dus geen constanten waar geen regels uit de functie δ op worden toegepast; alle symbolen van een string worden tijdens een afleidingsstap vervangen door een element uit Σ^*. Om het effect van een constante te bereiken voor een symbool $b \in \Sigma$, wordt de regel $b \to b$ toegevoegd aan δ in het geval van een $0L^\Sigma$-systeem. In het geval van respectievelijk $1L^\Sigma$- en $2L^\Sigma$-systemen worden de regels $a, b \to b$ en $a, b, c \to b$ voor alle $a, c \in \Sigma^\epsilon$ aan δ toegevoegd.

In onderstaand bewijs zullen we laten zien dat de $L0^\Sigma$-systemen even sterk zijn als gebruikelijke Lindenmayer-systemen met een verzameling constanten. We hebben het hier over deterministische systemen. Met een 0L-systeem bedoelen we een contextvrij Lindenmayer-systeem.

lemma 1. *Voor ieder 0L-systeem L is er een $0L^\Sigma$-systeem L' zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$*

[1]Lindenmayer en Van Dalen definiëren δ van 1L-systemen als $\Sigma \times \Sigma \to \Sigma^*$. Ze leggen hierin dus niet vast wat er gebeurt met $\delta(\epsilon, a)$, voor $a \in \Sigma$. Van Dalen lost dit op door te stellen dat altijd $\delta(\epsilon, a) = a$. Dit impliceert dat het meest linkse symbool in een woord altijd hetzelfde blijft. Lindenmayer lost het probleem op door een willekeurig symbool $b \in \Sigma$ aan te nemen als context van a. Dit levert moeilijkheden op omdat de taal van een systeem dan kan verschillen per keer dat die berekend wordt.

[2]Ook hier hebben Lindenmayer en Van Dalen Σ in plaats van Σ^ϵ gebruikt als de verzameling die dient als context.

Bewijs. Beschouw een willekeurig 0L-systeem $L = \langle V, C, w, \delta \rangle$. Definieer het $0L^{\Sigma}$-systeem L' als $L' = \langle \Sigma, w, \delta' \rangle$. Het alfabet van L' is gedefinieerd als $\Sigma = V \cup C$ en de transitiefunctie als $\delta' = \delta \cup \{\langle c, c \rangle \mid c \in C\}$. We voegen dus regels aan δ' toe, naast de regels van δ, die ervoor zorgen dat de constanten uit L zich ook als constanten gedragen in L'.

Neem een willekeurig woord $u \in \mathcal{L}(L)$. Dan geldt $w \overset{*}{\underset{L}{\Rightarrow}} u$ en dus dat er voor $n \geq 0$ een afleiding $u_0 \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} u_n$ is met $u_0 = w$ en $u_n = u$. Voor alle u_i met $i = 0, ..., n-1$ geldt dat er een $m \geq 0$ is waarvoor geldt dat $u_i = u_i^0 ... u_i^m$ zodanig dat $u_i^j \in (V \cup C)$ met $j = 0, ..., m$. Er is dan een $u_{i+1} = u_{i+1}^0 ... u_{i+1}^m$ met $u_{i+1}^j \in (V \cup C)^*$ zodanig dat $\delta(u_i^j) = u_{i+1}^j$ wanneer $u_i^j \in V$ en $u_i^j = u_{i+1}^j$ wanneer $u_i^j \in C$. Dan geldt dat $\delta'(u_i^j) = \delta(u_{i+1}^j)$ wanneer $u_i^j \in V$ en $\delta'(u_i^j) = u_i^j$ wanneer $u_i^j \in C$. Er geldt dus voor alle u_i^j dat $\delta'(u_i^j) = u_{i+1}^j$. Voor alle u_i geldt dus dat $u_i \underset{L'}{\Rightarrow} u_{i+1}$. Daarom geldt ook $u_0 \underset{L'}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_n$ met $u_0 = w$ en $u_n = u$ en dus $w \overset{*}{\underset{L'}{\Rightarrow}} u$. Hieruit volgt dat $u \in \mathcal{L}(L')$ en dus dat $\mathcal{L}(L) \subseteq \mathcal{L}(L')$.

Neem een willekeurig woord $u \in \mathcal{L}(L')$. Dan geldt $w \overset{*}{\underset{L'}{\Rightarrow}} u$ en dus dat er voor $n \geq 0$ een afleiding $u_0 \underset{L'}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_n$ is met $u_0 = w$ en $u_n = u$. Voor alle u_i met $i = 0, ..., n-1$ geldt dat er een $m \geq 0$ is waarvoor geldt dat $u_i = u_i^0 ... u_i^m$ zodanig dat $u_i^j \in \Sigma$ met $j = 0, ..., m$. Er is dan een $u_{i+1} = u_{i+1}^0 ... u_{i+1}^m$ met $u_{i+1}^j \in \Sigma^*$ zodanig dat $\delta'(u_i^j) = u_{i+1}^j$. Wanneer $u_i^j \neq u_{i+1}^j$, dan weten we dat $u_i^j \in V$ en $\delta(u_i^j) = u_{i+1}^j$. Wanneer $u_i^j = u_{i+1}^j$, dan $u_i^j \in C$ of $u_i^j \in V$ en $\delta(u_i^j) = u_i^j = u_{i+1}^j$. Voor alle u_i^j geldt dus dat $\delta(u_i^j) = u_{i+1}^j$ wanneer $u_i^j \in V$ en $u_i^j = u_{i+1}^j$ wanneer $u_i^j \in C$. Voor alle u_i geldt dus dat $u_i \underset{L}{\Rightarrow} u_{i+1}$. Daarom geldt ook $u_0 \underset{L}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_n$ met $u_0 = w$ en $u_n = u$ en dus $w \overset{*}{\underset{L}{\Rightarrow}} u$. Hieruit volgt dat $u \in \mathcal{L}(L)$ en dus dat $\mathcal{L}(L') \subseteq \mathcal{L}(L')$. □

lemma 2. *Voor ieder $0L^{\Sigma}$-systeem L is er een 0L-systeem L' zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$*

Bewijs. Beschouw een willekeurig $0L^{\Sigma}$-systeem $L = \langle \Sigma, \delta, w \rangle$. Definieer het 0L-systeem L' als $L' = \langle V, C, w, \delta' \rangle$. De verzameling constanten C is een verzameling van alle $a \in \Sigma$ waarvoor geldt dat $a \to a \in \delta$. De verzameling

variabelen V bevat vervolgens alle symbolen uit Σ die niet in C terecht komen. De functie δ' bevat alle regels $a \to u \in \delta$ waarvoor geldt dat $a \in V$.

Neem een willekeurig woord $u \in \mathcal{L}(L)$. Dan geldt $w \overset{*}{\underset{L}{\Rightarrow}} u$ en dus dat er voor $n \geq 0$ een afleiding $u_0 \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} u_n$ is met $u_0 = w$ en $u_n = u$. Voor alle u_i met $i = 0, ..., n-1$ geldt dat er een $m \geq 0$ is waarvoor geldt dat $u_i = u_i^0 ... u_i^m$ zodanig dat $u_i^j \in \Sigma$ met $j = 0, ..., m$. Er is dan een $u_{i+1} = u_{i+1}^0 ... u_{i+1}^m$ met $u_{i+1}^j \in \Sigma^*$ zodanig dat $\delta(u_i^j) = u_{i+1}^j$. Als voor u_i^j geldt dat $u_i^j \to u_i^j \in \delta$, dan $u_i^j \in C$ en $u_i^j \to u_i^j \notin \delta'$. Als het niet zo is dat $u_i^j \to u_i^j \in \delta$, dan geldt dat $u_i^j \in V$ en $\delta'(u_i^j) = \delta(u_i^j) = u_{i+1}^j$, Wanneer $u_i^j \in V$, dan $\delta'(u_i^j) = u_{i+1}^j$ en wanneer $u_i^j \in C$, dan $u_i^j = u_{i+1}^j$. Voor alle u_i geldt dus dat $u_i \underset{L'}{\Rightarrow} u_{i+1}$. Daarom geldt ook $u_0 \underset{L'}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_n$ met $u_0 = w$ en $u_n = u$ en dus $w \overset{*}{\underset{L'}{\Rightarrow}} u$. Hieruit volgt dat $u \in \mathcal{L}(L')$ en dus dat $\mathcal{L}(L) \subseteq \mathcal{L}(L')$.

Neem een willekeurig woord $u \in \mathcal{L}(L')$. Dan geldt $w \overset{*}{\underset{L'}{\Rightarrow}} u$ en dus dat er voor $n \geq 0$ een afleiding $u_0 \underset{L'}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_n$ is met $u_0 = w$ en $u_n = u$. Voor alle u_i met $i = 0, ..., n-1$ geldt dat er een $m \geq 0$ is waarvoor geldt dat $u_i = u_i^0 ... u_i^m$ zodanig dat $u_i^j \in (V \cup C)$ met $j = 0, ..., m$. Er is dan een $u_{i+1} = u_{i+1}^0 ... u_{i+1}^m$ met $u_{i+1}^j \in (V \cup C)^*$ zodanig dat $\delta'(u_i^j) = u_{i+1}^j$ wanneer $u_i^j \in V$ en $u_i^j = u_{i+1}^j$ wanneer $u_i^j \in C$. We weten dat $\delta(u_i^j) = u_i^j = u_{i+1}^j$ wanneer $u_i^j \in C$ en $\delta(u_i^j) = \delta'(u_i^j) = u_{i+1}^j$ wanneer $u_i^j \in V$. Er geldt dus voor alle u_i^j dat $\delta(u_i^j) = u_{i+1}^j$. Voor alle u_i geldt dus dat $u_i \underset{L}{\Rightarrow} u_{i+1}$. Daarom geldt ook $u_0 \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} u_n$ met $u_0 = w$ en $u_n = u$ en dus $w \overset{*}{\underset{L}{\Rightarrow}} u$. Hieruit volgt dat $u \in \mathcal{L}(L)$ en dus dat $\mathcal{L}(L') \subseteq \mathcal{L}(L)$. □

stelling 2. *0L$^\Sigma$-systemen zijn equivalent aan 0L-systemen.*

Bewijs. Omdat er voor ieder 0L-systeem een 0L$^\Sigma$-systeem is met dezelfde taal (zie lemma 1) en andersom (zie lemma 2), geldt dat 0L$^\Sigma$-systemen equivalent zijn aan 0L-systemen. □

In onderstaand bewijs laten we zien dat 2L-systemen even sterk zijn als 2L$^\Sigma$-systemen. 2L-systemen zijn contextsensitieve systemen. We weten dat de context hier altijd bestaat uit een enkel symbool aan de linkerkant en

een enkel symbool aan de rechterkant. We kunnen daarom de zeer uitgebreide definitie van de berekening van een contextsensitieve taal uit het vorige hoofdstuk vereenvoudigen naar het volgende: beschouw een 2L-systeem $L = \langle V, C, w, P\rangle$, een string $u = u_0...u_n$ met $u_i \in (V \cup C)$ en een string $v = v_0...v_n$ met $v_i \in (V \cup C)^*$ waarbij $i = 0, ..., n$ en $n \geq 0$. De string v kan worden afgeleid uit u dan en slechts dan als voor alle i geldt dat $u_i = v_i$ wanneer $u_i \in C$ en als één van de volgende vier gevallen geldt wanneer $u_i \in V$:

$$\begin{array}{lll} u_{i-1}, u_i, u_{i+1} \to v_i & \in P & \text{wanneer } i \neq 0 \text{ en } i \neq n \\ \epsilon, u_i, u_{i+1} \to v_i & \in P & \text{wanneer } i = 0 \text{ en } i \neq n \\ u_{i-1}, u_i, \epsilon \to v_i & \in P & \text{wanneer } i \neq 0 \text{ en } i = n \\ \epsilon, u_i, \epsilon \to v_i & \in P & \text{wanneer } i = n = 0 \end{array}$$

stelling 3. *2L$^\Sigma$-systemen zijn equivalent aan 2L-systemen.*

Bewijs. Voor ieder 2L-systeem $L = \langle V, C, w, \delta\rangle$ is er een 2L$^\Sigma$-systeem $L' = \langle \Sigma, w, \delta'\rangle$ zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$. Het alfabet van L' is gedefinieerd als $\Sigma = V \cup C$ en de transitiefunctie is volgt:

$$\delta' = \delta \cup \{\langle a, c, b, c\rangle \mid c \in C; a, b \in (V \cup C \cup \{\epsilon\})\}$$

We voegen dus regels aan δ' toe, naast de regels van δ, die ervoor zorgen dat de constanten uit L zich ook als constanten gedragen in L'.

Neem een willekeurig woord $u \in \mathcal{L}(L)$. Dan geldt $w \overset{*}{\underset{L}{\Rightarrow}} u$ en dus dat er voor $n \geq 0$ een afleiding $u_0 \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} u_n$ is met $u_0 = w$ en $u_n = u$. Voor alle u_i met $i = 0, ..., n-1$ geldt dat er een $m \geq 0$ is waarvoor geldt dat $u_i = u_i^0...u_i^m$ zodanig dat $u_i^j \in (V \cup C)$ met $j = 0, ..., m$. Er is dan een $u_{i+1} = u_{i+1}^0...u_{i+1}^m$ met $u_{i+1}^j \in (V \cup C)^*$ zodanig dat $u_i^j = u_{i+1}^j$ wanneer $u_i^j \in C$. Als $u_i^j \in V$ en $m = 0$, dan geldt $\delta(\epsilon, u_i^j, \epsilon) = u_{i+1}^j$. Als $u_i^j \in V$ en $m \neq 0$, dan geldt $\delta(\epsilon, u_i^j, u_i^{j+1}) = u_{i+1}^j$ als $j = 0$, $\delta(u_i^{j-1}, u_i^j, \epsilon) = u_{i+1}^j$ als $j = m$ en $\delta(u_i^{j-1}, u_i^j, u_i^{j+1}) = u_{i+1}^j$ als $j = 0$ noch $j = m$. We weten dat als $u_i^j \in C$, dan $a, u_i^j, b \to u_i^j = u_{i+1}^j \in \delta'$ voor alle $a, b \in \Sigma^\epsilon$. Wanneer $u_i^j \in V$, dan geldt voor alle regels $a, u_i^j, b \to u_{i+1}^j \in \delta$ dat ook $a, u_i^j, b \to u_{i+1}^j \in \delta'$. Er geldt dus voor alle u_i^j dat $\delta'(a, u_i^j, b) = u_{i+1}^j$, waarbij $a, b \in \Sigma^\epsilon$ respectievelijk de linker- en rechtercontext zijn van u_i^j. Voor alle u_i geldt dus dat $u_i \underset{L'}{\Rightarrow} u_{i+1}$.

Daarom geldt ook $u_0 \underset{L'}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_n$ met $u_0 = w$ en $u_n = u$ en dus $w \overset{*}{\underset{L'}{\Rightarrow}} u$. Hieruit volgt dat $u \in \mathcal{L}(L')$ en dus dat $\mathcal{L}(L) \subseteq \mathcal{L}(L')$.

Het bewijs voor $\mathcal{L}(L') \subseteq \mathcal{L}(L)$ verloopt op een vergelijkbare manier als het bewijs voor $\mathcal{L}(L) \subseteq \mathcal{L}(L')$ en het bewijs van lemma 1.

Voor ieder $2L^{\Sigma}$-systeem $L = \langle \Sigma, \delta, w \rangle$ is er een 2L-systeem $L' = \langle V, C, w, \delta' \rangle$ zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$. De verzameling constanten C is een verzameling van alle symbolen $b \in \Sigma$ waarvoor $a, b, c \to b \in \delta$ voor alle $a, c \in \Sigma^{\epsilon}$. De verzameling variabelen V bevat alle symbolen uit Σ die niet in C komen. De functie δ' bevat alle regels $a, b, c \to u \in \delta$ waarvoor geldt dat $b \in V$.

Het bewijs voor $\mathcal{L}(L') = \mathcal{L}(L)$ verloopt op een vergelijkbare manier als hierboven en het bewijs van lemma 2.

Omdat er voor ieder 2L-systeem een $2L^{\Sigma}$-systeem is met dezelfde taal en andersom, geldt dat $2L^{\Sigma}$-systemen equivalent zijn aan 2L-systemen. □

Ook 1L-systemen zijn equivalent aan $1L^{\Sigma}$-systemen. Het bewijs hiervoor lijkt veel op de bewijzen van stelling 2 en stelling 3 en laten we dus buiten beschouwing.

3.2 Inductief uitgebreide transitiefunctie

In het vorige hoofdstuk zagen we dat Lindenmayer-systemen equivalent zijn aan L^{Σ}-systemen: systemen zonder een verzameling constanten. Het volgende verschil is dat Van Dalen geen definitie van berekening geeft zoals gebruikelijk, maar in plaats daarvan de transitiefunctie δ inductief uitbreidt. We zullen laten zien dat dit geen verschil maakt en dat de systemen, die we IL-systemen zullen noemen, equivalent zijn aan L^{Σ}-systemen en dus aan de gebruikelijke Lindenmayer-systemen.

Een 0IL-systeem ziet er hetzelfde uit als een $0L^{\Sigma}$-systeem $\langle \Sigma, w, \delta \rangle$, waarbij δ dus in principe een functie is van Σ naar Σ^*. Het domein van de functie δ wordt door middel van een inductieve definitie uitgebreid van Σ naar Σ^*, dus δ wordt dan een functie van Σ^* naar Σ^*. Stel dat $a_i \in \Sigma$, $i = 0, ..., n$ en $n \geq 0$, dan is δ volgt gedefinieerd: $\delta(a_0...a_n) = \delta(a_0) * \delta(a_1...a_n)$ en $\delta(\epsilon) = \epsilon$. Het laatste geval geeft aan dat er niet 'iets' uit 'niets' kan ontstaan, wat ook

al het geval was bij de gebruikelijke Lindenmayer-systemen. $*$ is het symbool voor concatenatie. $u \Rightarrow v$ betekent nu $\delta(u) = v$.

Het uitbreiden van de definitie van δ is in essentie een andere manier van definieren wat $u \Rightarrow v$ voor $u, v \in \Sigma^*$ betekent. We zullen dat illustreren met een voorbeeld en verderop bewijzen.

voorbeeld Beschouw het 0L-systeem $L = \langle \Sigma, w, \delta \rangle$ met $\Sigma = \{a\}$, $w = a$ en $\delta = \{a \rightarrow aaa\}$. Als we op de gebruikelijke manier de regels uit δ toepassen op w blijkt dat de eerste woorden die het systeem genereert a, aaa en $aaaaaaaaa$ zijn. Wanneer we de inductief uitgebreide δ op het beginwoord toepassen geldt dat $\delta(a) = aaa$ en vervolgens dat $\delta(aaa) = \delta(a) * \delta(aa) = \delta(a) * \delta(a) * \delta(a) = aaaaaaaaa$. Het twee keer toepassen van de inductief uitgebreide transitiefunctie levert dus dezelfde woorden op als twee keer de regels van de transitiefunctie toepassen op het beginwoord. □

Ook het domein van δ van 1IL-systemen wordt inductief uitgebreid; we breiden het uit van $\Sigma^\epsilon \times \Sigma$ naar $\Sigma^\epsilon \times \Sigma^*$[3]. Laat $a \in \Sigma^\epsilon$ en $b_i \in \Sigma$ voor $i = 0, ..., n$. δ is dan als volgt: $\delta(a, b_0...b_n) = \delta(a, b_0) * \delta(b_0, b_1...b_n)$ en $\delta(a, \epsilon) = \epsilon$. Uit het laatste blijkt weer dat 'niets' nooit kan uitgroeien tot 'iets'. $u \Rightarrow v$ betekent $\delta(\epsilon, u) = v$ in een 1IL-systeem[4].

Het domein van δ van een 2IL-systeem wordt op een soortgelijke manier uitgebreid. Laat $a, c \in \Sigma^\epsilon$ en $b_i \in \Sigma$, $i = 0, ..., n$ en $n \geq 0$. De inductieve uitbreiding van het domein van $\Sigma^\epsilon \times \Sigma \times \Sigma^\epsilon$ naar $\Sigma^\epsilon \times \Sigma^* \times \Sigma^\epsilon$[5] gaat als volgt: $(a, b_0...b_n, c) = \delta(a, b_0, b_1) * \delta(b_0, b_1...b_n, c)$ en $\delta(a, \epsilon, c) = \epsilon$. In $\delta(a, \epsilon, c) = \epsilon$ ligt weer vastgelegd dat het niet mogelijk is dat ϵ uitgroeit tot iets. $u \Rightarrow v$ betekent $\delta(\epsilon, u, \epsilon) = v$ in een 2IL-systeem[6].

stelling 4. *$0L^\Sigma$-systemen zijn equivalent aan 0IL-sytemen.*

Bewijs. Voor ieder $0L^\Sigma$-systeem $L = \langle \Sigma, w, \delta \rangle$ is er een 0IL-systeem $L' =$

[3]Van Dalen breidt het domein uit naar $\Sigma^* \times \Sigma^*$. De context kan hier dus een element uit Σ^* zijn in plaats van slechts een element uit Σ^ϵ. Echter wordt er nooit gebruik gemaakt van de optie om een string langer als één symbool als context te gebruiken, dus laten we deze optie buiten beschouwing.

[4]Van Dalen definieert $\delta(\gamma, u) = v$ als $u \stackrel{\gamma}{\Rightarrow} v$ waarbij $\gamma \in \Sigma^*$

[5]Net als bij de 1IL-systemen breidt Van Dalen de contexten hier uit naar Σ^* in plaats van Σ^ϵ

[6]Van Dalen definieert $\delta(\gamma, u, \gamma') = v$ als $u \stackrel{\gamma\gamma'}{\Longrightarrow} v$ waarbij $\gamma, \gamma' \in \Sigma^*$

$\langle \Sigma, w, \delta' \rangle$ zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$. L' heeft hetzelfde alfabet Σ, beginwoord w en productieregels δ - maar dan uitgebreid volgens de inductieve definitie zoals hierboven beschreven - als L.

Neem een willekeurig woord $u \in \mathcal{L}(L)$. Omdat $w \overset{*}{\underset{L}{\Rightarrow}} u$ weten we dat er een afleiding $u_0 \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} u_n$ is met $n \geq 0$ zodanig dat $u_0 = w$ en $u_n = u$. Voor alle $i = 0, ..., n-1$ geldt dat $u_i = u_i^0 ... u_i^m$ met $m \geq 0$ en $u_i^j \in \Sigma$ voor $j = 0, ..., m$. Voor alle u_i geldt dat $\delta(u_i^0) * ... * \delta(u_i^m) = u_{i+1}$, dus $\delta'(u_i) = u_{i+1}$, dus $u_i \underset{L'}{\Rightarrow} u_{i+1}$ en dus $u_0 \underset{L'}{\Rightarrow} ... \underset{L'}{\Rightarrow} u_{i+1}$. Omdat $u_{i+1} = u_n = u$ en $u_0 = w$ geldt $w \overset{*}{\underset{L'}{\Rightarrow}} u$ en dus $u \in \mathcal{L}(L')$ en $\mathcal{L}(L) \subseteq \mathcal{L}(L')$.

Andersom geldt hetzelfde: $\mathcal{L}(L') \subseteq \mathcal{L}(L)$. Dit kan op dezelfde manier worden laten zien.

Voor ieder 0IL-systeem is er een $0L^{\Sigma}$-systeem met dezelfde taal. Het bewijs hiervoor is net als het bewijs hierboven. □

stelling 5. *$2L^{\Sigma}$-systemen zijn equivalent aan 2IL-sytemen.*

Bewijs. Voor ieder $2L^{\Sigma}$-systeem $L = \langle \Sigma, w, \delta \rangle$ is er een 2IL-systeem $L' = \langle \Sigma, w, \delta' \rangle$ zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$. L' heeft hetzelfde alfabet Σ, beginwoord w en productieregels δ - maar dan uitgebreid volgens de inductieve definitie zoals hierboven beschreven - als L.

Neem een willekeurig woord $u \in \mathcal{L}(L)$. Omdat $w \overset{*}{\underset{L}{\Rightarrow}} u$ weten we dat er een afleiding $u_0 \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} u_n$ is met $n \geq 0$ zodanig dat $u_0 = w$ en $u_n = u$. Voor alle $i = 0, ..., n-1$ geldt dat $u_i = u_i^0 ... u_i^m$ met $m \geq 0$ en $u_i^j \in \Sigma$ voor $j = 0, ..., m$. Voor alle u_i geldt dat $\delta(\epsilon, u_i^0) * \delta(u_i^0, u_i^1) * ... * \delta(u_i^{m-1}, u_i^m) = u_{i+1}$, dus $\delta'(\epsilon, u_i, \epsilon) = u_{i+1}$, dus $u_i \underset{L'}{\Rightarrow} u_{i+1}$ en dus $u_0 \underset{L'}{\Rightarrow} .. \underset{L'}{\Rightarrow} u_{i+1}$. Omdat $u_{i+1} = u_n = u$ en $u_0 = w$ geldt $w \overset{*}{\underset{L'}{\Rightarrow}} u$ en dus $u \in \mathcal{L}(L')$ en $\mathcal{L}(L') \subseteq \mathcal{L}(L)$.

Andersom geldt hetzelfde: $\mathcal{L}(L') \subseteq \mathcal{L}(L)$. Dit kan op dezelfde manier worden laten zien.

Voor ieder 2IL-systeem is er een $2L^{\Sigma}$-systeem met dezelfde taal. Het bewijs hiervoor is net als het bewijs hierboven. □

Ook $1L^{\Sigma}$-systemen zijn equivalent aan 1IL-sytemen. Het bewijs hiervoor lijkt erg veel op het bewijs voor stelling 4 en stelling 5, dus laten we dit buiten beschouwing.

3.3 SL-systemen

Een Lindenmayer-systeem in het artikel van Van Dalen is een tupel $\langle \Sigma, \delta \rangle$. Hier mist dus het beginwoord en de taal is dus ook niet de verzameling van alle woorden die voortkomen uit het beginwoord. In de Lindenmayer-systemen van Van Dalen is de taal de verzameling van de woorden waarop het systeem stopt. We zullen deze systemen SL-systemen noemen.

Een SL-systeem stopt op een woord u als u leidt tot een woord v en v heeft geen nieuw opvolgend woord meer (of u heeft zelf al geen opvolgend woord meer). Voor een S0L-systeem bijvoorbeeld, betekent dit dus dat $u \stackrel{*}{\Rightarrow} v$ en $v \Rightarrow v$, dus $\delta(v) = v$. Dat betekent dus dat voor $v = v_0...v_n$ en $n \geq 0$ geldt dat $\delta(v_i) = v_i$ voor alle $i = 0, ..., n$. $v_0...v_n$ worden stopstates genoemd[7]. Het kan ook zijn dat v het lege woord ϵ is. Per definitie is ϵ ook een stopstate, omdat in een Lindenmayer-systeem 'niets' niet kan uitgroeien tot 'iets' - ϵ zal altijd ϵ blijven. De taal van een SL-systemen is $\{u \mid u \stackrel{*}{\Rightarrow} v \text{ en } v \Rightarrow v\}$. Merk op dat systemen zonder stopstates een lege taal hebben.

Deze SL-systemen lijken wezenlijk anders dan degenen die we hiervoor besproken hebben. Het is dan ook onduidelijk hoe en of deze systemen equivalent zijn aan de gebruikelijke Lindenmayer-systemen. Van Dalen geeft hierop geen antwoord en verder zijn dit soort systemen in de literatuur niet (of moeilijk) te vinden. Om de de equivalentie aan te tonen zouden we voor ieder SL-systeem L een IL-systeem L' moeten vinden zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$ en voor ieder IL-systeem L een SL-systeem L' zodanig dat $\mathcal{L}(L') = \mathcal{L}(L)$. Voor beide systemen L is het onduidelijk hoe L' eruit zou moeten zien.

[7] Van Dalen definieert de stopstates als symbolen uit Σ waarvoor δ niet is gedefinieerd. δ is dan een partiële functie. Echter moet dan ook de definitie van $u \Rightarrow v$ worden aangepast. Daarom kiezen we ervoor om een stopstate te definiëren als een state waarvoor geldt dat $\delta(a) = a$. Het effect is hetzelfde.

4 Turingmachines gebruikt door Van Dalen

De turingmachines die Van Dalen gebruikt zien er anders uit dan de gebruikelijke machines zoals bijvoorbeeld gedefinieerd door Sipser. In een aantal stappen zullen we laten zien hoe deze machines toch equivalent zijn.

4.1 Turingmachines

Een turingmachine zoals gedefinieerd in Sipser bestaat uit een tape en een tapehead die zich ergens op de tape bevindt. De machine bevindt zich altijd in een bepaalde state. Een turingmachine is een tupel met zeven elementen: $\langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$. Hierin is Q een eindige verzameling van states, Σ een eindig inputalfabet (zonder het symbool $\sqcup$) en Γ een eindig tape-alfabet waarvoor geldt dat $\sqcup \in \Gamma$ en $\Sigma \subseteq \Gamma$. δ is de transitiefunctie die er uit ziet als $Q' \times \Gamma \to Q \times \Gamma \times \{L, R\}$ waarbij Q' de verzameling Q is zonder $q_{accepts}$ and q_{reject}. $q_0 \in Q$ is de startstate en $q_{accept} \in Q$ en $q_{reject} \in Q$ zijn de eindstates waarvoor geldt dat $q_{accept} \neq q_{reject}$.

Een turingmachine heeft een tape die ingedeeld is in vakjes en oneindig lang is naar rechts. De machine begint met een input $u \in \Sigma^*$ die vanaf het meest linkse vakje op de tape is geschreven, één symbool per vakje. De rest van de tape is gevuld met $\sqcup$. De tapehead start op het meeste linkse vakje op de tape.

Een configuratie uqv betekent dat de machine in state q is, dat uv op de tape geschreven staat en dat de tapehead zich boven het eerste symbool van v bevindt. De rest van de tape, na v, is gevuld met $\sqcup$. Neem een configuratie $uaqbv$ met $q \in Q$, $a, b \in \Gamma$ en $u, v \in \Gamma^*$. $uaqbv$ leidt tot een configuratie $uq'acv$ ($uaqbv \Rightarrow uq'acv$)[8] dan en slechts dan als $\delta(q, b) = \langle q', c, L\rangle$ en $uaqbv \Rightarrow uacq'v$ dan en slechts dan als $\delta(q, b) = \langle q', c, R\rangle$. Wanneer de tapehead zich op het meest linkse vakje bevindt en uit de transitiefunctie volgt dat de tapehead naar links moet, blijft de tapehead op het meest linkse vakje staan, dus $qbv \Rightarrow q'cv$ dan en slechts dan als $\delta(q, b) = \langle q', c, L\rangle$. De eerste configuratie van een turingmachine, de startconfiguratie, met een input u is q_0u. De configuraties waarop een turingmachine eindigt zijn configuraties waarbij de state q_{accept} (accepterende configuratie) of q_{reject} is.

[8] Sipser gebruikt zelf geen pijltjesnotatie met betrekking tot turingmachines.

Een turingmachine accepteert input u als er een opeenvolging van configuraties bestaat zodanig dat $C_0 \Rightarrow ... \Rightarrow C_n$ voor $n \geq 0$ waar C_0 de startconfiguratie is en C_n een accepterende configuratie is. De taal van een turingmachine T is de verzameling van alle woorden die de machine accepteert, dus $\mathcal{L}(T) = \{u \mid T \text{ accepteert } u\}$ ofwel

$$\mathcal{L}(T) = \{u \mid q_0 u \stackrel{*}{\Rightarrow} v q_{accept} w \text{ met } v, w \in \Gamma^*\}$$

In onderstaande beschrijvingen van turingmachines zal het voorkomen dat we niet alle instanties van δ benoemen. Als er een instantie van δ niet genoemd wordt, gaan we ervan uit dat δ deze instantie naar q_{reject} verwijst. Als we bijvoorbeeld niet benoemen waar $\delta(q, a)$ op afgebeeld wordt, dan gaan we ervan uit dat $\delta(q, a) = \langle q_{reject}, a, L\rangle$. Merk op dat L en het tweede voorkomen van a willekeurig zijn, we hadden ook R kunnen zeggen en een andere letter uit het tape-alfabet kunnen kiezen in plaats van a.

4.2 Turingmachines met een dubbel oneindige tape

Van Dalen gebruikt een andere variant van de turingmachine dan Sipser. Allereerst hebben zijn turingmachines een dubbel oneindige tape. Dat betekent dat de tape van de machine niet alleen oneindig lang naar rechts strekt, maar ook oneindig lang is naar links. Er is dus niet meer zoiets als een eerste vakje van de tape.

Formeel gezien verandert er vrij weinig aan de turingmachine. De definitie, start- en eindconfiguraties zien er hetzelfde uit. Het enige verschil is dat de tapehead niet op hetzelfde vakje blijft staan wanneer de machine in een configuratie qbv is, waarbij $q \in Q$, $b \in \Gamma$ en $v \in \Gamma^*$, en $\delta(q, b) = \langle q', c, L\rangle$. In deze situatie geldt nu $qbv \Rightarrow q' \sqcup cv$ in plaats van $qbv \Rightarrow q'cv$.

stelling 6. *Turingmachines zijn equivalent aan turingmachines met een dubbel oneindige tape.*

Bewijs. Voor iedere turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$ is er een turingmachine $T' = \langle Q', \Sigma', \Gamma', \delta', q_0', q_{accept}, q_{reject}\rangle$ met een dubbel oneindige tape zodanig dat $\mathcal{L}(T) = \mathcal{L}(T')$.

Als de tapehead van T zich op het meest linkervakje bevindt en de tapehead moet naar links, blijft deze staan op dezelfde plek. Dit effect moeten we ook bereiken bij T'. T' markeert hiervoor het eerste symbool van de input en blijft het symbool op dit vakje markeren wanneer het verandert gedurende de berekening. Dit vakje staat symbool voor het eerste vakje van de tape van T. Wanneer T op het eerste vakje van de tape staat en naar links gaat en dus blijft staan, gaat T' daadwerkelijk naar links en vervolgens naar rechts om in dezelfde state terecht te komen al T.

Het inputalfabet van T' is gedefinieerd als $\Sigma' = \Sigma \cup \{\overset{\bullet}{a} \mid a \in \Gamma\}$. Er is een nieuwe startstate q_0' en $Q' = Q \cup \{q_0', q_1, q''\}$. Het eerste wat T' doet is het markeren van het eerste symbool van de input. Er geldt daarom dat $\delta'(q_0', a) = \langle q_1, \overset{\bullet}{a}, R\rangle$ en $\delta'(q_1, a) = \langle q_0, a, L\rangle$ voor alle $a \in \Gamma$. Vervolgens kan de berekening beginnen. Deze verloopt in principe hetzelfde als bij T, dus $\delta \subseteq \delta'$. Veder geldt voor alle $a \in \Gamma$ geldt dat $\delta'(q, \overset{\bullet}{a}) = \langle q'', \overset{\bullet}{b}, R\rangle$ en $\delta(q'', c) = \langle q', c, L\rangle$ wanneer $\delta(q, a) = \langle q', b, L\rangle$ voor alle $c \in \Gamma'$. Er geldt dat $\mathcal{L}(T) = \mathcal{L}(T')$.

Voor iedere turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$ met een dubbel oneindige tape is er een turingmachine $T' = \langle Q', \Sigma', \Gamma', \delta', q_0', q_{accept}, q_{reject}\rangle$ zodanig dat $\mathcal{L}(T) = \mathcal{L}(T')$.

T' doet in principe precies wat T doet, maar als T' daardoor vanaf het meest linkse vakje naar links moet, verschuift de machine eerst alles wat op de tape staat een vakje naar rechts waardoor een nieuw $\sqcup$ symbool ontstaat op het meest linkse vakje. Om bij te houden wat het meest linkse vakje is, zullen we het symbool wat hierop staat vanaf het begin af aan markeren. We voeren ook een markeersymbool in om de meest rechterkant van de input te herkenen en te weten wat er allemaal een vakje naar rechts moet worden geschoven wanneer T' op het meeste linkse vakje naar links gaat.

De verzameling states van T' is gedefinieerd als $Q' = Q \cup \{q^+, q^-, q^\#, q^t, q_0'\} \cup \{q_a \mid a \in \Gamma\}$, waarbij q_0' de startstate is van T'. Het inputalfabet van T' is gedefinieerd als $\Sigma' = \Sigma \cup \{\#\} \cup \{\overset{\bullet}{a} \mid a \in \Gamma\}$.

De machine begint met het aanbrengen van markeringen. $\delta'(q_0', a) = \langle q^+, \overset{\bullet}{a}, R\rangle$ voor alle $a \in \Gamma$ van T. Het symbool op het meest linkse vakje van de tape krijgt dus een markering. Vervolgens gaan we naar het einde van de input om de markering $\#$ te plaatsen aan het einde van de input. Dus

$\delta'(q^+, a) = \langle q^+, a, R\rangle$ voor alle $a \in \Sigma$ van T en $\delta'(q^+, \sqcup) = \langle q^-, \#, L\rangle$. We gaan dan weer terug naar het begin, dus $\delta'(q^-, a) = \langle q^-, a, L\rangle$ voor alle $a \in \Sigma$ van T en $\delta'(q^-, \overset{\bullet}{a}) = \langle q_0, \overset{\bullet}{a}, L\rangle$ voor alle $a \in \Sigma'$. De machine kan dan beginnen met de berekening. De berekening gaat in principe hetzelfde als bij T: $\delta \subset \delta'$. Daarbij komt dat $\delta'(q, \#) = \langle q^\#, \sqcup, R\rangle$ en $\delta'(q^\#, \sqcup) = \langle q, \#, L\rangle$. Dat zorgt ervoor dat de markering # bijhoudt welk gedeelte van de tape is gebruikt. Verder geldt dat wanneer $\delta(q, a) = \langle q', b, R\rangle$ in T dat $\delta'(q, \overset{\bullet}{a}) = \langle q', \overset{\bullet}{b}, R\rangle$ voor alle $a \in \Gamma'$ met een $\bullet$ erop. Als $\delta(q, a) = \langle q', b, L\rangle$, geldt voor $\delta'(q, \overset{\bullet}{a})$ dat alles een vakje naar rechts wordt geplaatst. Dat houdt in dat $\delta'(q, \overset{\bullet}{a}) = \langle q^a, \overset{\bullet}{\sqcup}, R\rangle$ en voor alle states $q^a \in Q'$ en alle $b \in \Gamma'$ geldt $\delta'(q^a, b) = \langle q^b, a, R\rangle$, behalve wanneer $q^a = q^\#$. Dan geldt $\delta'(q^\#, \sqcup) = \langle q^t, \#, L\rangle$ en gaan we weer helemaal terug naar het begin, dus $\delta'(q^t, a) = \langle q^t, a, L\rangle$, zolang er geen $\bullet$ boven a staat. In dat geval geldt $\delta'(q^t, \overset{\bullet}{a}) = \langle q', b, L\rangle$. Er geldt dat $\mathcal{L}(T) = \mathcal{L}(T')$

Omdat er voor iedere turingmachine een turingmachine is met een dubbeloneindige tape en andersom, geldt dat deze twee types turingmachines equivalent zijn.

□

We zullen vanaf nu steeds een turingmachine met een dubbel oneindige tape bedoelen als we het over een turingmachine hebben.

4.3 Turingmachines met endmarkers

De turingmachines van Van Dalen gebruiken endmarkers. Dit betekent dat de berekeningen op de tape plaats vinden tussen een begin- en eindmarker. De endmarkers zijn bedoeld om bij te houden hoeveel tape er gescand wordt tijdens het berekenen.

Het tupel $\langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$ is hetzelfde als hiervoor op Γ en δ na. Het tape-alfabet bevat namelijk ook, naast $\sqcup$ en Σ, de twee endmarkers $\$_0$ en $\$_1$. De startconfiguratie van de machine is ook anders. Een turingmachine met endmarkers begint namelijk met de configuratie $\$_0 q_0 w \$_1$ in plaats van $q_0 w$. De input is dus tussen de twee endmarkers gezet.

Iedere keer als de tapehead van de turingmachine op een endmarker staat, wordt de endmarker opgeschoven, zodat het aantal vakjes tussen de twee

endmarkers één groter wordt. De machine doet dit door de endmarker te vervangen door $\sqcup$. Vervolgens gaat de tapehead een vakje naar buiten en schrijft daar de endmarker terug. Het aantal vakjes wat gebruikt is, is het aantal vakjes tussen de twee endmarkers.

voorbeeld Neem een configuratie $q\$_0 abcd\$_1$ waarbij $q \in Q$ en $a, b, c, d \in \Sigma$ van een turingmachine T. De tapehead staat hier dus op de linkerendmarker. Dat betekent dat de volgende configuraties achtereenvolgens $q' \sqcup \sqcup abcd\$_1$ en $\$_0 q \sqcup abcd\$_1$ zijn. □

Formeel betekent dit dat er voor alle $q \in Q$ een $q' \in Q$ is waarvoor geldt $\delta(q, \$_0) = \langle q', \sqcup, L\rangle$ en $\delta(q', \sqcup) = \langle q, \$_0, R\rangle$. Voor de andere endmarker betekent dit dat er voor alle $q \in Q$ een $q' \in Q$ moet zijn waarvoor geldt $\delta(q, \$_1) = \langle q', \sqcup, R\rangle$ en $\delta(q', \sqcup) = \langle q, \$_1, L\rangle$.

Iedere turingmachine kan worden omgeschreven naar een equivalente machine die endmarkers gebruikt en andersom.

stelling 7. *Turingmachines zijn equivalent aan turingmachines met endmarkers*

Bewijs. Voor iedere turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$ is er een turingmachine $T' = \langle Q', \Sigma, \Gamma', \delta', q_0, q_{accept}, q_{reject}\rangle$ met endmarkers zodanig dat $\mathcal{L}(T) = \mathcal{L}(T')$. Het tape-alfabet Γ' bevat per definitie ook de twee endmarkers, maar blijft verder hetzelfde want we laten het inputalfabet Σ hetzelfde als bij T. Er komen twee extra states bij: $Q' = Q \cup \{q_{\$_0}, q_{\$_1}\}$. $\delta \subseteq \delta'$ en aan δ' wordt het volgende toegevoegd: $\delta(q, \$_0) = \langle q_{\$_0}, \sqcup, L\rangle$ en $\delta(q_{\$_0}, \sqcup) = \langle q, \$_0, R\rangle$ voor alle $q \in Q'$ en $\delta(q, \$_1) = \langle q_{\$_1}, \sqcup, R\rangle$ en $\delta(q_{\$_1}, \sqcup) = \langle q, \$_1, L\rangle$ voor alle $q \in Q'$. In de twee nieuwe states kan de machine dus alleen terechtkomen als er een $\$_0$ of $\$_1$ is gelezen.

Neem een woord $u \in \mathcal{L}(T)$. Dat betekent dat T het woord u accepteert, dus dat er opeenvolging van configuraties plaatsvindt die eindigt in een accepterende configuratie. De transitiefunctie van T' is de transitiefunctie van T met een paar aanvullingen met betrekking tot de endmarkers. De endmarkers staan per definitie altijd op plekken waar op een turingmachine zonder endmarkers $\sqcup$ staat. Dat betekent dat wanneer T een toepassing van $\delta(q, \sqcup) = \langle q', a, L\rangle$ of $\delta(q, \sqcup) = \langle q', a, R\rangle$ doet voor een $q, q' \in Q$ en $a \in \Gamma$, dat T' dan in sommige situaties een toepassing van $\delta(q, \$_0) = \langle q_{\$_0}, \sqcup, L\rangle$

of $\delta(q, \$_1) = \langle q_{\$_1}, \sqcup, R\rangle$ doet in plaats van hetzelfde als T. De toepassingen die per definitie volgen op de laatste twee zijn $\delta(q_{\$_0}, \sqcup) = \langle q, \$_0, R\rangle$ en $\delta(q_{\$_1}, \sqcup) = \langle q, \$_1, L\rangle$. Bij beide toepassingen komt de machine weer in een situatie terecht waarin, net als T, alsnog $\delta(q, \sqcup)$ wordt toegepast. Hieruit volgt dat T' in iedere state terecht komt waar T ook in terecht komt, dus dat T', net als T, ook in q_{accept} terecht komt op input u, dus dat voor $u \in \mathcal{L}(T')$ en $\mathcal{L}(T) \subseteq \mathcal{L}(T')$.

Neem een woord $u \in \mathcal{L}(T')$. Dan is er een opeenvolging van configuraties zodanig dat de configuratie $\$_0 q u \$_1$ tot een accepterende configuratie leidt door middel van een reeks toepassingen van δ. Zoals we net hebben gezien, doet T alle toepassingen van δ die T' doet. T' doet een aantal toepassingen meer, die betrekking hebben op de endmarkers. Deze δ toepassingen leiden niet tot een configuratie met q_{accept}, zoals in δ' is gedefinieerd. Het zijn dus niet deze extra toepassingen van T' ten opzichte van T die leiden tot een accepterende configuratie, dus zal T ook in q_{accept} terecht komen op input u en geldt dus dat $u \in \mathcal{L}(T)$ en $\mathcal{L}(T') \subseteq \mathcal{L}(T)$.

Voor iedere turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$ met endmarkers is er een turingmachine $T' = \langle Q, \Sigma, \Gamma', \delta', q_0, q_{accept}, q_{reject}\rangle$ zodanig dat $\mathcal{L}(T) = \mathcal{L}(T')$. Het inputalfabat Σ van T' is hetzelfde als het inputalfabet van T. Het tape-alfabet Γ' is wel anders dan Γ, omdat Γ niet meer de twee endmarkers $\$_0$ en $\$_1$ bevat. Ook de states, inclusief de start- en eindstates, blijven hetzelfde. In δ is voor iedere $q \in Q$ een $q' \in Q$ waarvoor geldt dat $\delta(q, \$_0) = \langle q', \sqcup, L\rangle$ met $\delta(q', \sqcup) = \langle q, \$_0, R\rangle$ en $\delta(q, \$_1) = \langle q', \sqcup, R\rangle$ met $\delta(q', \sqcup) = \langle q, \$_1, L\rangle$. δ' bevat deze gevallen niet, maar is verder precies hetzelfde als δ.

Neem een woord $u \in \mathcal{L}(T)$. Dat betekent dat $\$_0 q_0 u \$_1 \overset{*}{\Rightarrow} u q_{accept} a v$ met $a \in \Gamma$ en $u, v \in \Gamma^*$. De beginconfiguratie van T' is $q_0 u$. Op de plek van de endmarkers staat dus nu $\sqcup$. Door de werking van δ omtrent de endmarkers weten we dat de endmarkers altijd op de plek staat van een $\sqcup$ in configuraties van T'. Stel dat de tapehead op de linkerendmarker staat. Dan geldt voor alle $q \in Q$ dat $\delta(q, \$_0) = \langle q', \sqcup, L\rangle$ en vervolgens $\delta(q', \sqcup) = \langle q, \$_0, R\rangle$ voor een $q' \in Q$. Vervolgens wordt $\delta(q, \sqcup)$ uitgevoerd. In het geval van T' staat er geen endmarker maar een $\sqcup$ en dan wordt dus direct $\delta(q, \sqcup)$ uitgevoerd. T' komt dus in dezelfde states terecht als T, op q' na. Echter kan q' nooit q_{accept} zijn, want $\delta(q_{accept}, a)$ is per definitie niet gedefinieerd. Dus als $u \in T$, dan $u \in T'$ en $\mathcal{L}(T') \subseteq \mathcal{L}(T)$.

Neem een willekeurige $u \in \mathcal{L}(T')$ zit. Dan is er een opeenvolging van configuraties $C_0 \Rightarrow ... \Rightarrow C_n$ met $n \geq 0$. T' komt ook in deze configuraties terecht, met daarbij nog een aantal extra configuraties omtrent de endmarkers. Als T' dan in q_{accept} komt, komt T dat dus ook. Er geldt dus dat $u \in \mathcal{L}(T)$ en $\mathcal{L}(T') \subseteq \mathcal{L}(T)$

Omdat er voor iedere turingmachine een turingmachine met endmarkers is met dezelfde taal, en andersom, geldt dat turingmachines equivalent zijn aan turingmachines met endmarkers. □

We zullen het vanaf nu steeds hebben over turingmachines met endmarkers hebben als we praten over turingmachines.

4.4 Turingmachines zonder eindstates

De turingmachines van Van Dalen hebben geen eindstates - q_{reject} en q_{accept} ontbreken. Een woord wordt geaccepteerd door de machine wanneer de machine stopt. Op alle woorden die niet tot de taal behoren, berekent de machine oneindig lang door. Omdat deze machines geen eindstates hebben waarvoor δ per definitie niet is gedefinieerd zoals bij eerder genoemde turingmachines, is de δ functie van deze turingmachines partieel gedefinieerd. Dat betekent dat er voor sommige instanties van δ geen uitkomst zal zijn. Het kan bijvoorbeeld zo zijn dat $\delta(q, a)$ niet voorkomt in de definitie van δ. Dat betekent dan dat, als de turingmachine in state q is en een a leest, de turingmachine stopt. De input waarop deze machine begonnen is, behoort dan tot de taal van de machine. Neem $a \in \Gamma$, $u, v \in \Gamma^*$ en $q \in Q$. Een accepterende configuratie is dan een configuratie $uqav$ waarvoor $\delta(q, a)$ niet voorkomt in δ. Anders gezegd is een accepterende configuratie een configuratie C waarvoor geen configuratie C' is zodanig dat $C \Rightarrow C'$. De definitie van een turingmachine T zonder eindstates is $T = \langle Q, \Sigma, \Gamma, \delta, q_0 \rangle$, waarbij de transitiefunctie gedefinieerd is als $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, in plaats van $\delta : Q' \times \Gamma \to Q \times \Gamma \times \{L, R\}$, waarbij $Q' = Q$ is zonder $q_{accepts}$ en q_{reject}, zoals voorheen.

stelling 8. *Turingmachines zijn equivalent aan turingmachines zonder eindstates.*

Bewijs. Voor iedere turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\rangle$ is er een turingmachine $T' = \langle Q, \Sigma, \Gamma, \delta', q_0\rangle$ zonder eindstates zodanig dat $\mathcal{L}(T) = \mathcal{L}(T')$. De verzameling states van T' zijn dus hetzelfde als die van T, inclusief q_{accept} en q_{reject}. Ook de startstate blijft onveranderd, evenals Σ en dus ook Γ. Enkel de transitiefuncties van T en T' verschillen. Per definitie is de transitiefunctie van T gedefinieerd als $\delta : Q' \times \Gamma \to Q \times \Gamma \times \{L, R\}$, waarbij $Q' = Q$ is zonder $q_{accepts}$ en q_{reject} en de transitiefunctie T' is partieel gedefinieerd als $\delta' : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$. Voor alle $a \in \Gamma$ laten we $\delta'(q_{accept}, a)$ ongedefinieerd en definiëren we $\delta'(q_{reject}, a) = \langle q_{reject}, a, L\rangle$. Als de machine in q_{reject} terecht komt, zal die dus oneindig lang blijven berekenen.

Neem een willekeurig woord $u \in \mathcal{L}(T)$. Dat betekent dat de machine uiteindelijk in q_{accept} terecht komt en nooit in q_{reject}. De transitiefunctie δ' van T' is hetzelfde als δ, op het geval $\delta'(q_{reject}, a)$ met $a \in \Gamma$ na, maar daar komt T noch T' dus in terecht op input u. T' belandt dus ook in q_{accept} op input u, wat betekent dat de machine stopt en $u \in \mathcal{L}(T')$ en dus $\mathcal{L}(T) \subseteq \mathcal{L}(T')$.

Andersom geldt hetzelfde: als een woord u wordt geaccepteerd door T', betekent dat dat deze in een stopstate terecht komt. De enige stopstate is q_{accept}. Dat betekent dat ook T op input u in de q_{accept} terecht komt en u dus accepteert, dus $\mathcal{L}(T') \subseteq \mathcal{L}(T)$

Voor iedere turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0\rangle$ is er een turingmachine $T' = \langle Q', \Sigma, \Gamma, \delta', q_0, q_{accept}, q_{reject}\rangle$, waarbij $Q' = Q \cup \{q_{accept}, q_{reject}\}$. Σ en dus Γ blijven onveranderd. δ' is δ van T met daarbij voor alle gevallen $\delta(q, a)$ die bij T ongedefinieerd zijn: $\delta'(q, a) = \langle q_{accept}, a, L\rangle$. L is hierbij willekeurig; het gaat erom dat de gevallen die bij δ ongedefinieerd zijn, nu in q_{accept} terecht komen. Merk op dat het niet nodig is om te definieren wanneer de machine in q_{reject} terecht komt - als de machine oneindig blijft berekenen in T, doet die dat ook bij T', zodat de machine ook bij T' niet in q_{accept} terecht komt en dus de input niet tot de taal behoort.

Neem een willekeurig woord $u \in \mathcal{L}(T)$. Dan geldt dat $\$_0 q_0 u \$_1 \overset{*}{\Rightarrow} uqav$ met $q \in Q$, $a \in \Gamma$ en $u, v \in \Gamma^*$, waarbij $\delta(q, a)$ ongedefinieerd is, zodat de machine stopt. Dat betekent dat voor T' geldt dat $\delta'(q, a) = \langle q_{accept}, a, L\rangle$. De rest van δ' blijft onveranderd ten opzichte van δ, dus belandt T' op input u ook in de configuratie $uqav$ en vanuit daar gaat de machine dus naar q_{accept}, wat betekent dat $u \in \mathcal{L}(T')$ en $\mathcal{L}(T) \subseteq \mathcal{L}(T')$.

Andersom geldt hetzelfde. Stel dat $u \in \mathcal{L}(T')$. Dat betekent dat $\$_0 q_0 u \$_1 \overset{*}{\Rightarrow}$

$uq_{accept}av$ voor $a \in \Gamma$ en $u, v \in \Gamma^*$. Vanuit de definitie van δ' weten we dan dat de configuratie die voorafgaat aan $uq_{accept}av$ een configuratie was met een state q' en symbool b waar de tapehead op staat, waarvoor $\delta(q', b)$ niet was gedefinieerd in T. Dat betekent dat T op input u in een stopstate terecht zou komen en dus dat $u \in \mathcal{L}(T)$ en $\mathcal{L}(T') \subseteq \mathcal{L}(T)$.

Omdat er voor iedere turingmachine een turingmachine zonder eindstates is met dezelfde taal, en andersom, geldt dat turingmachines equivalent zijn aan turingmachines zonder eindstates. □

Van Dalen gebruikt dus de turingmachines zoals Sipser, maar dan zonder eindstates, met endmarkers en een dubbel oneindige tape. Uit bovenstaande bewijzen volgt dat de machines van Van Dalen equivalent zijn aan die van Sipser. We kunnen namelijk een turingmachine volgens Sipser omschrijven naar een turingmachine met een dubbel oneindige tape volgens stelling 6, die we vervolgens kunnen omvormen naar een turingmachine met endmarkers volgens stelling 7, en deze naar een turingmachine zonder eindstates volgens stelling 8, zoals Van Dalen ze definieert. Op dezelfde manier vinden we voor iedere turingmachine à la Van Dalen een turingmachine zoals we die gewend zijn van Sipser.

5 S2L-systemen en turingmachines

We gebruiken de definitie van een turingmachines zoals in het vorige hoofdstuk beschreven om te bewijzen dat S2L-systemen minstens zo sterk zijn als turingmachines. We hebben het dus over turingmachines met een dubbel oneindige tape, endmarkers en zonder eindstates. Allereerst zullen we de definitie van S2L-systemen aanpassen naar equivalente S2L+systemen. Het doel van deze aanpassing is om het bewijs dat S2L-systemen minstens zo sterk zijn als turingmachines te vereenvoudigen. Vervolgens zullen laten zien dat deze S2L+systemen minstens zo sterk zijn als turingmachines.

5.1 SL+systemen

Het verschil tussen SL-systemen en SL+systemen is de definitie van de taal van een systeem. De taal van een SL+systeem $L = \langle \Sigma, \delta \rangle$ is gedefinieerd als $\mathcal{L}(L) = \{u \mid L \text{ stopt op } f(\$_0 q_0 u \$_1)\}$ ofwel $\mathcal{L}(L) = \{u \mid f(\$_0 q_0 u \$_1)\} \stackrel{*}{\Rightarrow} v$ en $v \Rightarrow v\}$. f is een simpele functie die wordt toegepast op een string die een configuratie voorstelt. De functie maakt van de state q van de configuratie en het symbool a rechts van q, dus waar de tapehead van een machine op zou staan, een tupel $\langle q, a \rangle$. Voor een SL+systeem $L = \langle \Sigma, \delta \rangle$ is de functie dus als volgt gedefinieerd:

$$
\begin{array}{lcll}
f(\$_0 q_0 \$_1) & = & \$_0 \langle q_0 \$_1 \rangle & \\
f(\$_0 q_0 u_0 \$_1) & = & \$_0 \langle q_0, u_0 \rangle \$_1 & \text{met } u_0 \in \Sigma \\
f(\$_0 q_0 u_0 ... u_n \$_1) & = & \$_0 \langle q_0, u_0 \rangle u_1 ... u_n \$_1 & \text{met } u_i \in \Sigma, n \geq 1
\end{array}
$$

lemma 3. *S2L-systemen zijn equivalent aan S2L+systemen.*

Bewijs. Voor ieder S2L-systeem $L = \langle \Sigma, \delta \rangle$ is er een SL+systeem $L' = \langle \Sigma', \delta' \rangle$ zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$. Het alfabet van L' is $\Sigma' = \Sigma \cup \{\$_0, \langle q_0, a \rangle, \$_1\}$ voor alle $a \in \Sigma$. Voor de transitiefunctie van L' geldt dat $\delta \subseteq \delta'$ en voor alle

$a, c \in \Sigma'^{\epsilon}$ en $b, d \in \Sigma'$:

$$\begin{array}{lcl}
\delta'(a, \langle q_0, b\rangle, c) & = & \delta'(a, b, c) \\
\delta'(\langle q_0, d\rangle, b, a) & = & \delta'(d, b, a) \\
\delta'(a, b, \langle q_0, d\rangle) & = & \delta'(a, b, d) \\
\delta'(a, \$_0, c) & = & \epsilon \\
\delta'(a, \$_1, c) & = & \epsilon \\
\delta'(\$_0, b, \$_1) & = & \delta(\epsilon, b, \epsilon) \\
\delta'(\$_0, b, c) & = & \delta(\epsilon, b, c) \\
\delta'(a, b, \$_0) & = & \delta(a, b, \epsilon) \\
\delta'(\$_1, b, c) & = & \delta(\epsilon, b, c) \\
\delta'(a, b, \$_1) & = & \delta(a, b, \epsilon)
\end{array}$$

Een symbool $\langle q_0, b\rangle$ wordt dus beschouwd als b en de endmarkers als ϵ. In sommige gevallen zijn meerdere van de bovenstaande instanties van toepassing - in dat geval moet steeds de eerste instante die van toepassing is worden gekozen.

Neem een willekeurig woord $u \in \mathcal{L}(L)$. Stel $u = \epsilon$, dan:

$$\begin{array}{lcl}
\delta'(\epsilon, f(\$_0 q_0 u \$_1), \epsilon) & = & \delta'(\epsilon, \$_0\langle q_0, \$_1\rangle, \epsilon) \\
& = & \delta'(\epsilon, \$_0, \langle q_0, \$_1\rangle) * \delta'(\$_0, \langle q_0, \$_1\rangle, \epsilon) \\
& = & \delta'(\epsilon, \$_0, \$_1) * \delta'(\$_0, \$_1, \epsilon) \\
& = & \epsilon * \epsilon \\
& = & \epsilon
\end{array}$$

Hieruit volgt dat $f(\$_0 q_0 u \$_1) \underset{L'}{\overset{*}{\Rightarrow}} \epsilon$. ϵ is een stopstate, wat betekent dat $u \in \mathcal{L}(L')$. Stel $u = u_0 ... u_n$ met $n \geq 0$ en $u_i \in \Sigma$ voor $i = 0, ..., n$, dan:

$$\begin{array}{lcl}
\delta'(\epsilon, f(\$_0 q_0 u \$_1), \epsilon) & = & \delta'(\epsilon, \$_0\langle q_0, u_0\rangle u_1 ... u_n \$_1, \epsilon) \\
& = & \delta'(\epsilon, \$_0, \langle q_0, u_0\rangle) * \delta'(\$_0, \langle q_0, u_0\rangle, u_1) * \\
& & \delta'(\langle q_0, u_0\rangle, u_1, u_2) * ... * \delta'(u_{n-1}, u_n, \$_1) * \delta'(u_n, \$_1, \epsilon) \\
& = & \delta'(\epsilon, \$_0, u_0) * \delta'(\$_0, u_0, u_1) * \delta'(u_0, u_1, u_2) * ... * \\
& & \delta'(u_{n-1}, u_n, \$_1) * \delta'(u_n, \$_1, \epsilon) \\
& = & \epsilon * \delta(\epsilon, u_0, u_1) * \delta(u_0, u_1, u_2) * ... * \delta'(u_{n-1}, u_n, \epsilon) * \epsilon \\
& = & \delta(\epsilon, u, \epsilon)
\end{array}$$

Hieruit volgt dat $f(\$_0 q_0 u \$_1) \underset{L'}{\overset{*}{\Rightarrow}} \delta(\epsilon, u, \epsilon)$. Omdat L stopt op u weten we dat u uit stopstates bestaat, of leidt tot een string die uit stopstates bestaat.

Daarom geldt ook $u \in L'$. Er geldt dus dat $\mathcal{L}(L) \subseteq \mathcal{L}(L')$. Het bewijs voor $\mathcal{L}(L') \subseteq \mathcal{L}(L)$ lijkt hier erg veel op en laten we dus buiten beschouwing.

Voor ieder S2L+systeem $L = \langle \Sigma, \delta \rangle$ is er een S2L-systeem $\langle \Sigma', \delta' \rangle$ zodanig dat $\mathcal{L}(L) = \mathcal{L}(L')$. Het alfabet van L' is gedefinieerd als $\Sigma' = \Sigma \cup \{\overset{\bullet}{a} \mid a \in \Sigma\}$. De transitiefunctie van L' is voor alle $a, b, c \in \Sigma$ en $d, e \in \Sigma'^{\epsilon}$ gedefinieerd als volgt:

$$\begin{aligned}
\delta'(\epsilon, b, \epsilon) &= \overset{\bullet}{\$_0}\langle q_0, \overset{\bullet}{b}\rangle\overset{\bullet}{\$_1} \\
\delta'(a, b, \epsilon) &= \overset{\bullet}{b}\overset{\bullet}{\$_1} \\
\delta'(\epsilon, b, c) &= \overset{\bullet}{\$_0}\langle q_0, \overset{\bullet}{b}\rangle \\
\delta'(a, b, c) &= \overset{\bullet}{b} \\
\delta'(d, \overset{\bullet}{b}, e) &= \delta(d, \overset{\bullet}{b}, e)
\end{aligned}$$

Neem een willekeurig woord $u \in \mathcal{L}(L)$, dus L stopt op $f(\$_0 q_0 u \$_1)$. We willen bewijzen dat $u \in \mathcal{L}(L')$, dus dat L' stopt op u. Stel $u = \epsilon$, dan stopt L per definitie op u. Stel $u = u_0 ... u_n$ met $n \geq 0$ en $u_i \in \Sigma$ voor $i = 0, ...n$. Dan geldt $f(\$_0 q_0 u \$_1) = \$_0 \langle q_0 u_0 \rangle u_1 ... u_n \$_1$ en:

$$\begin{aligned}
\delta'(\epsilon, u, \epsilon) &= \delta'(\epsilon, u_0, u_1) * ... * \delta'(u_{n-1}, u_n, \epsilon) \\
&= \overset{\bullet}{\$_0}\langle q_0 \overset{\bullet}{u_0}\rangle * ... * \overset{\bullet}{u_n}\overset{\bullet}{\$_1} \\
&= \overset{\bullet}{\$_0}\langle q_0 \overset{\bullet}{u_0}\rangle \overset{\bullet}{u_1} ... \overset{\bullet}{u_n}\overset{\bullet}{\$_1}
\end{aligned}$$

De uitkomst van ϵ, u, ϵ is dus $f(\$_0 q_0 u \$_1)$, met op ieder symbool een $\bullet$. Omdat geldt dat $\delta'(d, \overset{\bullet}{b}, e) = \delta(d, \overset{\bullet}{b}, e)$ voor alle $b \in \Sigma$ en $d, e \in \Sigma'^{\epsilon}$, weten we dat L' precies hetzelfde zal doen als L en daarbij $\bullet$ zal plaatsen op ieder symbool. Ook L zal daarom stoppen op $f(\$_0 q_0 u \$_1)$ en dus op u. Er geldt dus $u \in L'$ en $\mathcal{L}(L) \subseteq \mathcal{L}(L')$. Het bewijs voor $\mathcal{L}(L') \subseteq \mathcal{L}(L)$ lijkt hier erg veel op en laten we dus buiten beschouwing.

Voor ieder S2L-systeem is er dus een S2L+systeem met dezelfde taal, en andersom, dus zijn S2L-systemen equivalent aan S2L+systemen.

□

5.2 S2L-systemen zijn minstens zo sterk als turingmachines

We laten zien dat S2L-systemen minstens zo sterk zijn als turingmachines door te bewijzen dat voor iedere turingmachine T er een S2L+systeem L is zodanig dat voor iedere configuratie C geldt dat $C \underset{T}{\Rightarrow} C'$ dan en slechts dan als $f(C) \underset{L}{\Rightarrow} f(C')$. Vervolgens laten we zien hoe hieruit volgt dat S2L-systemen minstens zo sterk zijn als turingmachines.

lemma 4. *Voor iedere turingmachines T is er een S2L+systeem L zodanig dat voor iedere configuratie C geldt $C \underset{T}{\Rightarrow} C'$ dan en slechts dan $f(C) \underset{L}{\Rightarrow} f(C')$.*

Bewijs. Beschouw een willekeurige turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0 \rangle$. Voor T definiëren we het S2L+systeem $L = \langle \Sigma', \delta' \rangle$. Het alfabet van L is gedefinieerd als $\Sigma' = \Gamma \cup (Q \times \Gamma)$. Omdat T een turingmachine met endmarkers is, bevat T per definitie voor alle $q \in Q$ een $q' \in Q$ zodanig dat $\delta(q, \$_0) = \langle q', \sqcup, L \rangle$. We voegen voor alle q en q' waarvoor dit geldt de volgende instanties toe aan δ' van L: $\delta'(\epsilon, \langle q, \$_0 \rangle, a) = \langle q', \sqcup \rangle \sqcup$ voor alle $a \in \Sigma'$. Voor de andere endmarker weten we dat er voor alle $q \in Q$ een $q' \in Q$ is zodanig dat $\delta(q, \$_1) = \langle q', \sqcup, R \rangle$. Voor al deze q en q' voegen we voor alle $a \in \Sigma'$ de instantie $\delta'(a, \langle q, \$_1 \rangle, \epsilon) = \sqcup \langle q', \sqcup \rangle$ toe. Voor iedere overige instantie $\delta(q, a) = \langle q', b, R \rangle$ van de turingmachine T voegen we voor alle $c, d \in \Sigma'^{\epsilon}$ het volgende toe aan δ' van L: $\delta'(c, \langle q, a \rangle, d) = b$ en $\delta'(\langle q, a \rangle, c, d) = \langle q', c \rangle$. Voor alle andere gevallen $\delta(q, a) = \langle q', b, L \rangle$ in δ voegen we voor alle $c, d \in \Sigma'^{\epsilon}$ de instanties $\delta'(c, \langle q, a \rangle, d) = b$ en $\delta'(c, d, \langle q, a \rangle) = \langle q', d \rangle$ toe. Ten slotte voegen we nog $\delta'(a, c, b) = c$ toe aan δ' voor alle $a, b \in \Sigma'^{\epsilon}$ en $c \in \Sigma'$ waarvoor δ' nog niet definieerd is. Merk op dat alle gevallen van δ' stopstates zijn op een aantal specifieke uitzonderingen na waarin het paar $\langle q, a \rangle$ van de huidige state q en het symbool a dat de tapehead leest een rol speelt.

Stel dat voor een willekeurige configuratie C geldt dat $C \underset{T}{\Rightarrow} C'$. We weten dat in C precies één $q \in Q$ voorkomt en dat de tapehead zich op een bepaalde $a \in \Gamma$ bevindt. Als we kijken naar hoe δ' van L is geconstrueerd op basis van δ van T, zien we dat er drie mogelijkheden zijn voor a die bepalen welke niet-stopstates van toepassing zijn op $f(C)$. We onderscheiden deze drie gevallen waarin $a = \$_0$, $a = \$_1$ en a is $\$_0$ noch $\$_1$.

1. Stel dat $a = \$_0$, dan $C = q\$_0 u$ en $f(C) = \langle q, \$_0\rangle u$ voor een $u = u_0...u_n \in \Gamma^*$ met $n \geq 0$ (want u bevat sowieso nog $\$_1$). We weten dat er een $q' \in Q$ is zodanig dat $\delta(q, \$_0) = \langle q', \sqcup, L\rangle$ voor T en dus $\delta'(\epsilon, \langle q, \$_0\rangle, a) = \langle q', \sqcup\rangle\sqcup$ voor iedere $a \in \Sigma'$ voor L. De overige states in $f(C)$ zijn stopstates. De configuratie die volgt na C is de configuratie $C' = q' \sqcup \sqcup u$. De string die volgt op $f(C)$ wordt als volgt berekend:

$$\begin{aligned}
\delta'(\epsilon, f(C), \epsilon) &= \delta'(\epsilon, \langle q, \$_0\rangle u, \epsilon) \\
&= \delta'(\epsilon, \langle q, \$_0\rangle, u_0) * \delta'(\langle q, \$_0\rangle, u, \epsilon) \\
&= \langle q', \sqcup\rangle \sqcup * u \\
&= \langle q', \sqcup\rangle \sqcup u \\
&= f(C')
\end{aligned}$$

   Hieruit volgt dus dat $f(C) \underset{L}{\Rightarrow} f(C')$.

2. Stel dat $a = \$_1$, dan $C = uq\$_1$ en $f(C) = u\langle q, \$_1\rangle$ voor een $u = u_0...u_n \in \Gamma^*$ met $n \geq 0$ (want u bevat sowieso nog $\$_0$). We weten dat er een $q' \in Q$ is zodanig dat $\delta(q, \$_1) = \langle q', \sqcup, R\rangle$ en dus $\delta'(a, \langle q, \$_1\rangle, \epsilon) = \sqcup\langle q', \sqcup\rangle$ voor iedere $a \in \Sigma'$ voor L. De configuratie die volgt na C is de configuratie $C' = u \sqcup q'\sqcup$. De string die volgt op $f(C)$ wordt als volgt berekend:

$$\begin{aligned}
\delta'(\epsilon, f(C), \epsilon) &= \delta'(\epsilon, u\langle q, \$_1\rangle, \epsilon) \\
&= \delta'(\epsilon, u_0, u_1) * \delta'(u_0, u_1...u_n\langle q, \$_1\rangle, \epsilon) \\
&= \delta'(\epsilon, u_0, u_1) * ... * \delta'(u_{n-1}, u_n, \langle q, \$_1\rangle) * \delta'(u_n, \langle q, \$_1\rangle, \epsilon) \\
&= u_0 * ... * u_n * \sqcup\langle q', \sqcup\rangle \\
&= u \sqcup \langle q', \sqcup\rangle \\
&= f(C')
\end{aligned}$$

   Hieruit volgt dus dat $f(C) \underset{L}{\Rightarrow} f(C')$

3. Stel a is $\$_0$ noch $\$_1$. In deze situatie zijn er drie mogelijkheden. De eerste mogelijkheid is dat de tapehead zich tussen de twee endmarkers bevindt. Er zijn maar twee situaties mogelijk waarin de tapehead zich niet tussen (of op) de endmarkers bevindt, namelijk de situaties die ontstaan wanneer de tapehead een endmarker heeft gelezen - er wordt dan een $\sqcup$ op de plek van de endmarker geschreven en de tapehead gaat een vakje naar buiten. We onderscheiden deze drie gevallen.

   3.1 Stel dat C de configuratie is nadat de linkerendmarker is gelezen. Het symbool a waar de tapehead op staat is dan gelijk aan $\sqcup$.

Rechts hiervan staat ook een $\sqcup$ - dit is waar eerst de endmarker stond. C is dan gelijk aan $q \sqcup \sqcup u$ waarbij $u = u_0...u_n \in \Gamma^*$ met $n \geq 0$ (want u bevat sowieso nog $\$_1$) en dus $f(C) = \langle q, \sqcup\rangle \sqcup u$. We weten dat er een $q' \in Q$ is zodanig dat $\delta(q, \sqcup) = \langle q', \$_0, R\rangle$ en dat daarom voor alle $c, d \in \Sigma'^{\epsilon}$ geldt dat $\delta'(c, \langle q, \sqcup\rangle, d) = \$_0$ en $\delta'(\langle q, \sqcup\rangle, c, d) = \langle q', c\rangle$. De configuratie die volgt na C is $C' = \$_0 q' \sqcup u$. De string die volgt op $f(C)$ wordt als volgt berekend:

$$\begin{aligned}
\delta'(\epsilon, f(C), \epsilon) &= \delta'(\epsilon, \langle q, \sqcup\rangle \sqcup u, \epsilon) \\
&= \delta'(\epsilon, \langle q, \sqcup\rangle, \sqcup) * \delta'(\langle q, \sqcup\rangle, \sqcup, u_0) * \delta'(\sqcup, u, \epsilon) \\
&= \$_0 * \langle q', \sqcup\rangle * u \\
&= \$_0\langle q', \sqcup\rangle u \\
&= f(C')
\end{aligned}$$

Dus geldt $f(C) \underset{L}{\Rightarrow} f(C')$.

3.2 Stel dat C de configuratie is nadat de rechterendmarker is gelezen. Het symbool a waar de tapehead op staat is dan wederom gelijk aan $\sqcup$ en links hiervan staat ook een $\sqcup$. C is dan gelijk aan $u \sqcup q \sqcup$ waarbij $u = u_0...u_n \in \Gamma^*$ met $n \geq 0$ (want u bevat sowieso nog de endmarker $\$_0$) en dus $f(C) = u \sqcup \langle q, \sqcup\rangle$. We weten dat er een $q' \in Q$ is zodanig dat $\delta(q, \sqcup) = \langle q', \$_1, L\rangle$ en dat daarom voor alle $c, d \in \Sigma'^{\epsilon}$ geldt dat $\delta'(c, \langle q, \sqcup\rangle, d) = \$_1$ en $\delta'(c, d, \langle q, \sqcup\rangle) = \langle q', d\rangle$. De configuratie die volgt na C is $C' = uq' \sqcup \$_1$. De string dievolgt op $f(C)$ wordt als volgt berekend:

$$\begin{aligned}
\delta'(\epsilon, f(C), \epsilon) &= \delta'(\epsilon, u \sqcup \langle q, \sqcup\rangle, \epsilon) \\
&= \delta'(\epsilon, u_0, u_1) * ... * \delta'(u_{n-1}, u_n, \sqcup) * \\
&\quad \delta'(u_n, \sqcup, \langle q, \sqcup\rangle) * \delta'(\sqcup, \langle q, \sqcup\rangle, \epsilon) \\
&= u_0 * ... * u_n * \langle q', \sqcup\rangle * \$_1 \\
&= u\langle q', \sqcup\rangle \$_1 \\
&= f(C')
\end{aligned}$$

Dus geldt $f(C) \underset{L}{\Rightarrow} f(C')$.

3.3 Stel dat a zich tussen de twee endmarkers bevindt. Dan $C = uqav$ en dus $f(C) = u\langle q, a\rangle v$, waarbij $u = u_0...u_n, v = v_0...v_m \in \Gamma^*$ met $n, m \geq 0$ (omdat u en v allebei sowieso een endmarker bevatten). We weten niet of $\delta(q, a) = \langle q', b, L\rangle$ of $\delta(q, a) = \langle q', b, R\rangle$, dus onderscheiden we de twee gevallen.

3.3.1 Stel $\delta(q,a) = \langle q', b, R\rangle$, dan weten we dus dat voor alle $c, d \in \Sigma'^{\epsilon}$ geldt dat $\delta'(c, \langle q,a\rangle, d) = b$ en $\delta'(\langle q,a\rangle, c, d) = \langle q', c\rangle$. De configuratie die volgt na C is $C' = ubq'v$. De string die volgt op $f(C)$ wordt als volgt berekend:

$$\begin{aligned}
\delta'(\epsilon, f(C), \epsilon) &= \delta'(\epsilon, u\langle q,a\rangle v, \epsilon) \\
&= \delta'(\epsilon, u_0, u_1) * ... * \delta'(u_{n-1}, u_n, \langle q,a\rangle) * \\
&\quad \delta'(u_n, \langle q,a\rangle, v_0) * \delta'(\langle q,a\rangle, v_0, v_1) * \\
&\quad \delta'(v_0, v_1, v_2) * ... * \delta'(v_{m-1}, v_m, \epsilon) \\
&= u_0 * ... * u_n * b * \langle q', v_0\rangle * v_1 * ... * v_m \\
&= ub\langle q', v_0\rangle v_1 ... v_m \\
&= f(ubq'v_0 ... v_m) \\
&= f(ubq'v) \\
&= f(C')
\end{aligned}$$

Dus geldt $f(C) \underset{L}{\Rightarrow} f(C')$.

3.3.2 Stel $\delta(q,a) = \langle q', b, L\rangle$, dan weten we dus dat voor alle $c, d \in \Sigma'^{\epsilon}$ geldt dat $\delta'(c, \langle q,a\rangle, d) = b$ en $\delta'(c, d, \langle q,a\rangle) = \langle q', d\rangle$. De configuratie die volgt na C is $C' = u_0 ... u_{n-1} q' u_n bv$. De string die volgt op $f(C)$ wordt als volgt berekend:

$$\begin{aligned}
\delta'(\epsilon, f(C), \epsilon) &= \delta'(\epsilon, u\langle q,a\rangle v, \epsilon) \\
&= \delta'(\epsilon, u_0, u_1) * ... * \delta'(u_{n-2}, u_{n-1}, u_n) * \\
&\quad \delta'(u_{n-1}, u_n, \langle q,a\rangle) * \delta'(u_n, \langle q,a\rangle, v_0) * \\
&\quad \delta'(\langle q,a\rangle, v_0, v_1) * ... * \delta(v_{m-1}, v_m, \epsilon) \\
&= u_0 * ... * u_{n-1} * \langle q', u_n\rangle * b * v_0 * ... * v_m \\
&= u_0 ... u_{n-1} \langle q', u_n\rangle bv \\
&= f(C')
\end{aligned}$$

Dus geldt $f(C) \underset{L}{\Rightarrow} f(C')$.

De situaties die hierboven zijn beschreven zijn alle mogelijke situaties van een turingmachine. Uit al deze situaties volgt dat $f(C) \underset{L}{\Rightarrow} f(C')$, dus kunnen we afleiden dat voor alle configuraties C geldt dat als $C \underset{T}{\Rightarrow} C'$ dan $f(C) \underset{L}{\Rightarrow} f(C)$.

Stel dat $f(C) \underset{L}{\Rightarrow} f(C')$ geldt voor een willekeurige configuratie C. Op dezelfde manier kan worden bewezen dat $C \underset{T}{\Rightarrow} C'$, dus $f(C) \underset{L}{\Rightarrow} f(C)$ dan en slechts dan als $C \underset{T}{\Rightarrow} C'$.

Hieruit dat voor iedere turingmachine T er een S2L+systeem L is zodanig dat voor iedere configuratie C geldt dat $C \underset{T}{\Rightarrow} C'$ dan en slechts dan als $f(C) \underset{L}{\Rightarrow} f(C')$. □

Met behulp van lemma 3 en 4 zullen we laten zien dat S2L-systemen minstens zo sterk zijn als turingmachine.

stelling 9. *S2L-systemen zijn minstens zo sterk als turingmachines.*

Bewijs. Om te bewijzen dat S2L-systemen minstens zo sterk zijn als turingmachines, moeten we laten zien dat er voor iedere turingmachine T een S2L-systeem L is zodanig dat $\mathcal{L}(T) = \mathcal{L}(L)$. Beschouw een willekeurige turingmachine $T = \langle Q, \Sigma, \Gamma, \delta, q_0 \rangle$. Laat L' het S2L+systeem zijn uit lemma 4 waarvoor geldt dat voor iedere configuratie C geldt dat $C \underset{T}{\Rightarrow} C'$ dan en slechts dan als $f(C) \underset{L'}{\Rightarrow} f(C')$. Laat L het S2L-systeem zijn dat L' simuleert zoals beschreven in lemma 3. Voor L geldt dan dus ook dat voor iedere configuratie C geldt dat $C \underset{T}{\Rightarrow} C'$ dan en slechts dan als $f(C) \underset{L}{\Rightarrow} f(C')$.

Neem een willekeurig woord $u \in \mathcal{L}(T)$. Er geldt dat T stopt op u. Dat betekent er een opeenvolging van configuraties is zodanig dat $C_0 \overset{*}{\underset{T}{\Rightarrow}} C_n$ voor $n \geq 0$ zodanig dat dat er geen C_{n+1} is waarvoor geldt dat $C_n \underset{T}{\Rightarrow} C_{n+1}$. Er geldt dat C_0 de startconfiguratie $\$_0 q_0 u \$_1$ is en $C_n = uqav$ is voor een zekere $q \in Q$, $a \in \Gamma$ en $u, v \in \Gamma^*$. Omdat we weten dat voor iedere configuratie C geldt dat $C \underset{T}{\Rightarrow} C'$ dan en slechts dan als $f(C) \underset{L}{\Rightarrow} f(C')$, weten we dus dat $f(C_0) \underset{L}{\Rightarrow} ... \underset{L}{\Rightarrow} f(C_n)$ waarbij $C_0 = f(\$_0 q_0 u \$_1)$ en $C_n = f(uqav)$ en dus $f(\$_0 q_0 u \$_1) \overset{*}{\underset{L}{\Rightarrow}} f(uqav)$. Omdat er geen configuratie C_{n+1} is zodanig dat $C_n \underset{T}{\Rightarrow} C_{n+1}$, weten we dat $\delta(q, a)$ ongedefinieerd is. Zoals we in lemma 4 hebben gezien zijn alle states van L stopstates, behalve een aantal waarbij het paar $\langle q, a \rangle$ van de huidige state q en het symbool a waarop tapehead staat betrokken zijn. Echter, omdat $\delta(q, a)$ ongedefinieerd is, zijn er ook geen niet-stopstates gedefinieerd voor δ' met betrekking tot het paar $\langle q, a \rangle$. Daarom geldt voor ieder symbool $c \in \Sigma'$ waar $f(uqav)$ uit is opgebouwd dat c een stopstate is. Er geldt dus dat $f(\$_0 q_0 u \$_1) \overset{*}{\underset{L}{\Rightarrow}} f(uqav)$ en $f(uqav) \underset{L}{\Rightarrow} f(uqav)$, dus $u \in \mathcal{L}(L)$ en $\mathcal{L}(T) \subseteq \mathcal{L}(L)$. We kunnen op dezelfde manier laten zien dat $\mathcal{L}(L) \subseteq \mathcal{L}(T)$. □

6 Conclusie

In deze scriptie hebben we het bewijs van Van Dalen uitgewerkt en laten zien dat bepaalde Lindenmayer-systemen, die we S2L-systemen hebben genoemd, minstens zo sterk zijn als turingmachines zoals gedefinieerd door Sipser. Vanwege de Church-Turing-hypothese kunnen we stellen dat deze S2L-systemen equivalent zijn aan turingmachines. Echter blijkt dat de gebruikelijke Lindenmayer-systemen geen S2L-systemen zijn. Het is nog de vraag hoe en of de S2L-systemen van Van Dalen equivalent zijn aan de gebruikelijke (contextsensitieve) Lindenmayer-systemen.

In de context van de kunstmatige intelligentie betekent dit dat het niet zeker is of en hoe alle mogelijke processen en berekeningen gesimuleerd kunnen worden op een Lindenmayer-systeem. Mogelijkerwijs zijn er dus algoritmen die niet te berekenen zijn op een Lindenmayer-systeem en is een Lindenmayer-systeeem dus geen geschikt model voor een computer. We weten wel dat S2L-systemen equivalent zijn aan turingmachines. Ook deze S2L-systemen bevatten het parallelisme dat Lindenmayer-systemen zo uniek maakt en geschikt voor het beschrijven van (imaginaire) levensvormen. S2L-systemen zouden dus een goed alternatief kunnen zijn voor Lindenmayer-systemen om computationeel ingewikkelde problemen mee op te lossen, zolang niet bekend is of Lindenmayer-systemen equivalent zijn aan turingmachines.

De nog openstaande vraag is dus hoe de S2L-systemen zich verhouden tot gebruikelijke Lindenmayer-systemen en uiteindelijk blijft de vraag dus wat de compuationele kracht is van gebruikelijke Lindenmayer-systemen en of we daadwerkelijk (contextsensitieve) Lindenmayersystemen kunnen gebruiken als een model voor alles wat berekenbaar is.

Referenties

[1] D. Van Dalen, *A Note on Some Systems of Lindenmayer*, 1970.

[2] A. Lindenmayer, Mathematical Models for Cellular Interactions in Development I, II, *J. Theoretical Biology* **18** (1968) 280-315.

[3] M. Sipser, *Introduction to the Theory of Computation*, 3rd edition, Cengage Learning, 2013.

[4] L.M.L. de Campos, M. Roisenberg en R.C.L. de Oliveira, Automatic design of neural networks with l-systems and genetic algorithms - a biologically inspired methodology, *International Joint Conference on Neural Networks* **1** (2011) 1199-1206.

[5] L. Kari, G. Rozenberg en A. Solomaa, L-systems, *Handbook of formal languages* **1** (1997) 253 - 328