Automaten en Berekenbaarheid

Pieter Vanderschueren

Academiejaar 2023-2024

Inhoudsopgave

1 Talen en automaten

1.1 Wat is een taal?

Definitie 1.1: String over een alfabet Σ

Een **string** over een alfabet Σ is een eindige opeenvolging van nul, één of meer elementen van Σ.

Toepassing 1.1: ϵ-compressie van een string

Als we uit een string $w \in \Sigma_\epsilon$ elk voorkomen van ϵ schrappen, wat resulteert in een nieuwe string s, dan noemen we s de ϵ-compressie van w.

Definitie 1.2: Taal L over een alfabet Σ

Een **taal** L over een alfabet Σ is een verzameling van strings over Σ.

1.2 Een algebra van talen

Definitie 1.3: Een algebra- of algebraïsche structuur

Een algebra- of algebraïsche structuur is een verzameling met daarop een aantal inwendige operaties: dikwijls binaire operaties, maar unair of met grotere ariteit kan ook. Zo wordt de verzameling van alle talen over een alfabet Σ een algebra als we als operaties unie, doorsnede, complement, etc. definïeren. Meer concreet: als L_1 en L_2 twee talen zijn, dan is

- de unie ervan een taal: $L_1 \cup L_2$
- de doorsnede ervan een taal: $L_1 \cap L_2$
- het complement ervan een taal: $\overline{L_1}$

Eigenschap 1.1: Concatenatie van twee talen

Gegeven twee talen L_1 en L_2 over hetzelfde alfabet Σ, dan noteren we de concatenatie van L_1 en L_2 als L_1L_2 en definïeren we:

$$L_1L_2 = \{xy \mid x \in L_1,\ y \in L_2\}$$

Eigenschap 1.2: Kleene ster van een taal

De Kleene ster van een taal wordt gedefinieerd als volgt:

$$L^* = \cup_{n \geq 0} L^n$$

Opmerking: $L^+ = LL^*$

Toepassing 1.2: Kleene ster van een alfabet

De verzameling van alle strings over een alfabet Σ is de kleene ster van het aflabet Σ^*, en volgt dus dat

$$L \in \mathcal{P}(\Sigma^*)$$

1.3 Reguliere expressies en reguliere talen

Definitie 1.4: Reguliere Expressie (RE) over een alfabet Σ

E is een **reguliere expressie** over een alfabet Σ indien E van de vorm is

- ϵ
- ϕ
- a waarbij $a \in \Sigma$
- $(E_1 E_2)$ waarbij E_1 en E_2 reguliere expressies zijn over Σ
- (E_1^*) waarbij E_1 een reguliere expressies is over Σ
- $(E_1 | E_2)$ waarbij E_1 en E_2 reguliere expressies zijn over Σ

Definitie 1.5: Reguliere taal

Een reguliere expressie E bepaalt een **reguliere taal** L_E over hetzelfde alfabet Σ als volgt:

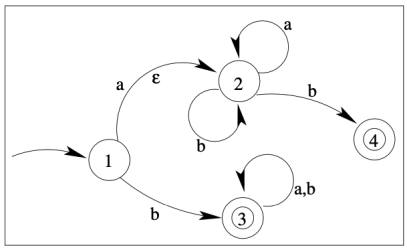
- als $E = a$ (met $a \in \Sigma$) dan is $L_E = \{a\}$
- als $E = \epsilon$ dan is $L_E = \{\epsilon\}$
- als $E = \phi$ dan is $L_E = \emptyset$
- als $E = (E_1 E_2)$ dan $L_E = L_{E_1} L_{E_2}$
- als $E = (E_1)^*$ dan $L_E = L_{E_1}^*$
- als $E = (E_1 | E_2)$ dan $L_E = L_{E_1} \cup L_{E_2}$

1.4 Niet-deterministische eindge toestandsautomaten

Definitie 1.6: Niet-deterministische eindige toestandsautomaat (NFA)

Een **niet-deterministische eindige toestandsautomaat** is een 5-tal $(Q, \Sigma, \delta, q_s, F)$ waarbij

- Q een eindige verzameling toestanden is
- Σ is een eindig alfabet
- $\delta \subseteq Q \times \Sigma_\epsilon \times Q$ is de overgangsrelatie van de automaat, wat kan worden voorgesteld in een transitietabel
- q_s is de starttoestand
- $F \subseteq Q$ is de verzameling eindtoestanden



Opmerking: In de literatuur wordt δ soms ook als een functie gedefinieerd die een toestand en een symbool afbeeldt op de verzameling van toestanden waarnaar een overgang mogelijk is:

$$\delta(q, a) = \{q' \in Q \mid (q, a, q') \in \delta\}.$$

We schrijven $p \xrightarrow{a} q$ om aan te geven dat (p, a, q) een overgang is in δ.

Definitie 1.7: Aanvaarden van strings in een NFA

Een string s wordt aanvaard door een NFA $(Q, \Sigma, \delta, q_s, F)$ indien er een sequentie $q_s = q_0 \xrightarrow{a_0} \dots \xrightarrow{a_{n-1}} q_n$ van overgangen bestaat met $q_n \in F$ zodat s de ϵ-compressie is van $a_0 \dots a_{n-1}$.

Dus: Voor toestanden p,q en string $w \in \Sigma^*$ schrijven we $p \overset{w}{\rightsquigarrow} q$ indien er een sequentie van overangen $p \xrightarrow{a_0} \dots \xrightarrow{a_{n-1}} q$ bestaat zodat w de ϵ-compressie is van $a_0 \dots a_{n-1}$.

Definitie 1.8: Taal L bepaald door een NFA M

Een taal L wordt bepaald door een NFA M, indien L de verzameling van strings is die M aanvaardt. We noteren de taal van M als L_M.

Definitie 1.9: Equivalentie van twee NFA's

Twee NFA's worden **equivalent** genoemd als ze dezelfde taal bepalen, m.a.w. elke equivalentieklassen van deze equivalentierelatie komt overeen met een taal.

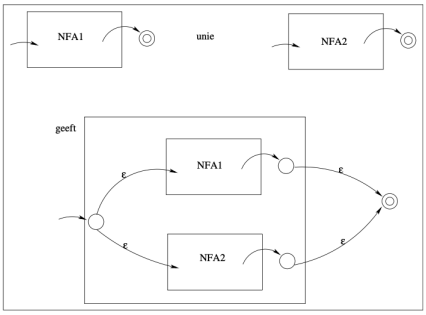
1.5 De algebra van NFA's

Eigenschap 1.3: Unie van twee NFA's

Gegeven: $i \in \{1,2\}: \ \text{NFA}_i = (Q_i, \Sigma, \delta_i, q_{s_i}, \{q_{f_i}\})$

De unie $\text{NFA}_1 \cup \text{NFA}_2$ is de $\text{NFA} = (Q, \Sigma, \delta, q_s, F)$ waarbij

- $Q = Q_1 \cup Q_2 \cup \{q_s, q_f\}$, $F = \{q_f\}$
- δ is gedefnieerd als:
  - $\forall x \in \Sigma : \delta(q_s, x) = \emptyset \ \wedge \ \delta(q_{f_i}, x) = \emptyset$
  - $\forall q \in Q_i \backslash \{q_{f_i}\}, \ x \in \Sigma_\epsilon : \ \delta(q, x) = \delta_i(q, x)$
  - $\delta(q_s, \epsilon) = \{q_{s_1}, q_{s_2}\}$
  - $\delta(q_{f_i}, \epsilon) = \{q_f\}$

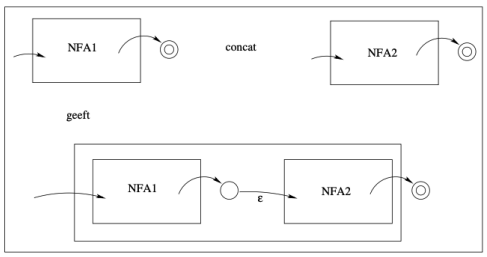


Eigenschap 1.4: Concatenatie van twee NFA's

Gegeven: $i \in \{1,2\}: \ \text{NFA}_i = (Q_i, \Sigma, \delta_i, q_{s_i}, \{q_{f_i}\})$

De concatenatie NFA_1NFA_2 is de $\text{NFA} = (Q, \Sigma, \delta, q_s, F)$ waarbij

- $Q = Q_1 \cup Q_2$, $q_s = q_{s_1}$, $F = \{q_{f_2}\}$
- δ is gedefnieerd als:
  - $\forall x \in \Sigma : \ \delta q_{f_1}, x = \emptyset$
  - $\forall q \in Q_i \backslash \{q_{f_1}\}, \ x \in \Sigma_\epsilon : \ \delta(q, x) = \delta_i(q, x)$
  - $\delta(q_{f_1}, \epsilon) = \{q_{s_2}\}$

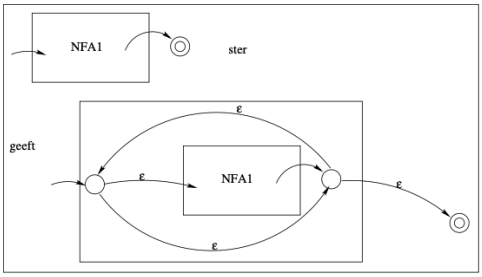


Eigenschap 1.5: Kleene ster van een NFA

Gegeven: $\text{NFA}_1 = (Q_1, \Sigma, \delta_1, q_{s_1}, \{q_{f_1}\})$

De Kleene ster NFA_1^* is de $\text{NFA} = (Q, \Sigma, \delta, q_s, F)$ waarbij

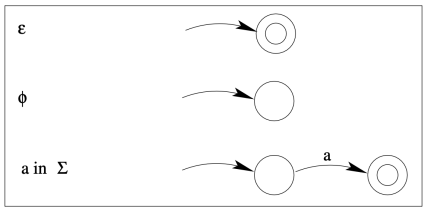
- $Q = Q_1 \cup \{q_s, q_f\}$, $F = \{q_f\}$
- δ is gedefnieerd als:
  - $\forall x \in \Sigma : \delta(q_s, x) = \emptyset \ \wedge \ \delta(q_{f_1}, x) = \emptyset$
  - $\forall q \in Q_1 \backslash \{q_{f_1}\}, \ x \in \Sigma_\epsilon : \ \delta(q, x) = \delta_1(q, x)$
  - $\delta(q_s, \epsilon) = \{q_{s_1}, q_{f_1}\}$
  - $\delta(q_{f_1}, \epsilon) = \{q_s, q_f\}$



1.6 Van RE naar NFA

Definitie 1.10: RE → NFA

We hebben alle ingrediënten om van een reguliere expressie RE een NFA te maken, en zodanig dat de $L_{RE} = L_{NFA}$. Vermits reguliere expressies inductief gedefinieerd zijn zullen we voor elk lijntje van die definitie een overeenkomstige NFA definiëren. De drie basisgevallen ϵ, ϕ en $a \in \Sigma$ zijn triviaal te modeleren als NFA. We illusteren hieronder:



De drie recursieve gevallen beschrijven we als volgt: laat E_1 en E_2 twee reguliere expressies zijn, dan is

- $NFA_{E_1E_2} = \text{concat}(NFA_{E_1}, NFA_{E_2})$
- $NFA_{E_1^*} = \text{ster}(NFA_{E_1})$
- $NFA_{E_1|E_2} = \text{unie}(NFA_{E_1}, NFA_{E_2})$

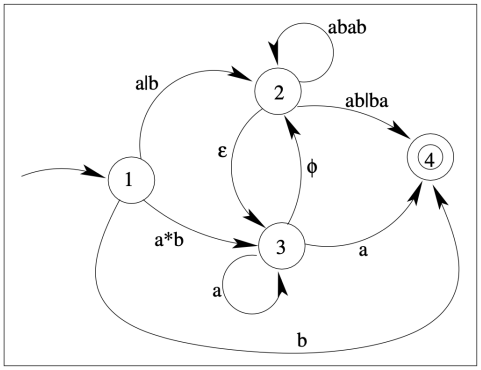
De constructie hierboven bewaart de taal, t.t.z. $L_{NFA_E} = L_E$.

1.7 Van NFA naar RE

Definitie 1.11: GNFA

Een **GNFA** is een eindige toestandsmachine met de volgende wijzigingen en beperkingen:

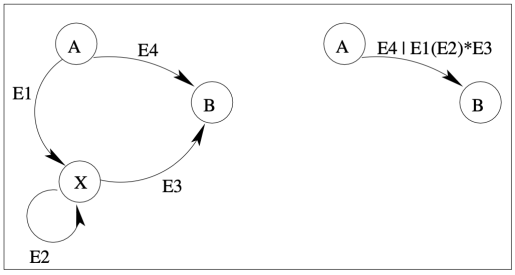
- er is slechts één eindtoestand en die is verschillend van de starttoestand
- vanuit de starttoestand vertrekt er juist één boog naar elke andere toestand; er komen geen bogen aan in de starttoestand
- in de eindtoestand komt juist één boog aan vanuit elke andere toestand; uit de eindtoestand vertrekken geen bogen
- voor paar p,q (let op: $p = q$ is geldig) van andere toestanden (geen start- of eindtoestand) is er juist één boog $p \to q$ en één boog $q \to p$.
- de bogen hebben als label een reguliere expressie



Algoritme 1.1: NFA → RE

1. Maak van de NFA een GNFA
   - Behoud alle toestanden en bogen van de NFA
   - Als er meerdere bogen zijn tussen twee toestanden gelabeld met symbolen $a_1 \dots a_n$ vervang deze door één boog met als label $a_1 | \dots | a_n$
   - Voer een nieuwe starttoestand in en een ϵ-boog naar de oude starttoestand
   - Voer een nieuwe eindtoestand in en ϵ-bogen vanuit elke oude eindtoestand
   - Voor elke boog die ontbreekt tussen twee toestanden om een GNFA te bekomen, voer een ϕ-boog in
2. Reduceer de GNFA:

   Kies een willekeurige toestand X verschillend van de start- of eindtoestand, ga naar stap 3 als dit niet mogelijk is. Voor elk paar toestanden A en B (let op: $A = B$ is geldig) verschillend van X bevat de GNFA een unieke boog $A \to B$ met label E_4, $A \to X$ met label E_1, $X \to X$ met label E_2 en $X \to B$ met label E_3. Vervang het label op de boog $A \to B$ door $E_4 | E_1 E_2^* E_3$. Doe dit voor alle keuzes voor A en B. Verwijder daana de knoop X en herhaal.



3. Bepaal RE: de boog van de GNFA heeft als label de gezochte RE

Stelling 1.1: Taal herkend door een NFA

De taal herkend door de NFA is de taal van de berekende reguliere expressie, en dus bepalen de twee geziene formalismen NFA en RE precies dezlfde klasse van talen: de reguliere talen.

1.8 Deterministische eindige toestandsmachines

Definitie 1.12: Deterministische eindige toestandsmachines

Een NFA is een DFA indien δ geen ϵ-overgangen bevat en indien voor elke $p \in Q$ en elke $a \in \Sigma$ een unieke $q \in Q$ bestaat zodat $p \xrightarrow{a} q$. Het komt erop neer dat in een DFA, δ een totale functie $Q \times \Sigma \to Q$ is. Voor DFA's zullen a we de unieke toestand q zodat $p \xrightarrow{a} q$ dan ook noteren als $\delta(p, a)$.

Algoritme 1.2: NFA → DFA

Neem voor een willekeurige string $w \in \Sigma^*$ de verzameling $\{q \mid q_s \overset{w}{\rightsquigarrow} q\}$ van toestanden vereikbaar met w. We noteren deze verzameling als q'_w. Deze verzameling dient in een DFA een singleton te zijn, dit sugereert de volgende werkwijze: we bouwen een nieuwe automaat $(Q', \Sigma, \delta', q'_s, F')$ waarvan de toestandenovereenkomen met verzamelingen van toestanden van de NFA. We construeren die zodanig dat voor elke string w geldt dat $q'_s \overset{w}{\rightsquigarrow} q'_w$ (in de DFA), i.e. q'_w is de set van toestanden in de NFA die bereikbaar zijn vanaf de oorspronkelijke begintoestand q_s met w. We definiëren:

- Q' als de verzameling van alle deelverzamelingen q' van Q die gesloten zijn onder ϵ-bogen, dus
$$p \in q' \ \wedge \ p \overset{\epsilon}{\rightarrow} q \ \Rightarrow \ q \in q'$$
- $\delta' : Q' \times \Sigma \to Q'$ als
$$\delta'(q', a) = \{q \mid \exists p \in q' : \ p \overset{a}{\rightsquigarrow} q\}$$
- q'_s als de verzameling van q_s en toestanden die bereikbaar zijn vanuit q_s met ϵ-bogen, dus
$$q'_s = \{q_s, q \mid q_s \overset{\epsilon}{\rightsquigarrow} q\}$$
- F' als de verzameling van alle q' zodat $q' \cap F \neq \emptyset$, dus
$$F' = \{q' \in Q' \mid q' \cap F \neq \emptyset\}$$

Uit constructie volgt dat deze automaat $(Q', \Sigma, \delta', q'_s, F')$ een DFA is.

Stelling 1.2: DFA en NFA equivalentie

De geconstrueerde DFA $(Q', \Sigma, \delta', q'_s, F')$ is equivalent met de NFA $(Q, \Sigma, \delta, q_s, F)$.

Bewijs 1.1: DFA en NFA equivalentie

We moeten verifiëren dat
$$\forall w \in \Sigma^* : \ q'_s \overset{w}{\rightsquigarrow} F' \ \Leftrightarrow \ q_s \overset{w}{\rightsquigarrow} F$$
De essentie van dat bewijs is dat
$$\forall w \in \Sigma^* : \ q'_s \overset{w}{\rightsquigarrow} q' \text{ (in de DFA)} \ \Leftrightarrow \ q' = q_w = \{q \mid q_s \overset{w}{\rightsquigarrow} q\} \text{ (in de NFA)}$$
Dit is eenvoudig inductief te bewijzen gebruik makend van het feit dat $q' = q'_w \Rightarrow \delta'(q', a) = q'_{wa}$. Dan geldt dat de DFA een string w aanvaardt als voor de unieke toestand q' zodat $q'_s \overset{w}{\rightsquigarrow} q'$ geldt dat $q' \cap F \neq \emptyset \ \Leftrightarrow \ q'_w \cap F \neq \emptyset \ \Leftrightarrow \ q_s \overset{w}{\rightsquigarrow} F$. □

Stelling 1.3: Taal bepaald door een DFA

De taal bepaald door een DFA is een reguliere taal.

Eigenschap 1.6: Doorsnede, verschil en complement van DFA's

Gegeven: $i \in \{1,2\}: \ \text{DFA}_i = (Q_i, \Sigma, \delta_i, q_{s_i}, \{q_{f_i}\})$

We maken een generische product DFA $(Q, \Sigma, \delta, q_s, F)$ als volgt:

- $Q = Q_1 \times Q_2$
- $\delta(p \times q, x) = \delta_1(p, x) \times \delta_2(q, x)$
- $q_s = (q_{s_1}, q_{s_2})$
- Om tot een volledig definitie te komen, moeten we nog F bepalen:
  - $F = F_1 \times F_2$: de DFA is de doorsnede van de twee talen
  - $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$: de DFA is nu de unie van de twee talen
  - $\forall i \neq j \in [1,2]: \ F = F_i \times (Q_j \times F_j)$: de DFA bepaalt nu de strings die tot L_i behoren, maar niet tot L_j
  - $F = (Q_1 \backslash F_1) \times (Q_2 \backslash F_2)$: de DFA bepaalt nu de strings die tot geen van beide talen behoren.

Hieruit volgt ook dat de unie, doorsnede en complement van twee reguliere talen ook regulier zijn. Daaruit volgt ook dat het complement van een reguliere taal ook regulier is, want $\overline{L} = \Sigma^* \backslash L$.

Definitie 1.13: f-string

We noemen een string s een f-string vanuit q van de DFA indien $\delta^*(q, s) \in F$, t.t.z. indien er een pad is van q naar een toestand van F die s genereert. F-gelijke toestanden zijn dan toestanden met dezelfde f-strings.

Opmerking: $q \in F \Leftrightarrow \epsilon$ is een f-string vanuit q

Definitie 1.14: f-gelijk

Twee toestanden q_1, q_2 zijn f-gelijk indien

$$\{s \in \Sigma^* \mid \delta^*(q_1, s) \in F\} = \{s \in \Sigma^* \mid \delta^*(q_2, s) \in F\}$$

In woorden, als q_1 en q_2 exact dezelfde f-strings hebben.

Eigenschap 1.7: f-gelijk

- De relatie f-gelijk is een equivalentie-relatie.
- Als p, q f-gelijk zijn dan geldt voor elk symbool a dat $\delta(p, a)$ en $\delta(q, a)$ ook f-gelijk zijn.
- Als p, q f-gelijk zijn dan geldt $p \in F \Leftrightarrow q \in F$.

Bewijs 1.2: Eigenschappen van de f-gelijk relatie

- Het is triviaal om te bewijzen dat f-gelijkheid een equivalentie-relatie is. Dit kan je doen door de reflexiviteit, symmetrie en transitiviteit van de relatie na te gaan.
- Veronderstel dat p, q f-gelijk zijn en veronderstel voor een willekeurig symbool a dat $\delta(p, a) = p', \delta(q, a) = q'$. De f-strings van p en q zijn gelijk, en dus ook hun f-strings van de vorm as. De f-strings van p' zijn de strings s zodat as een f-string is van p. Hetzelfde geldt voor q'. Bijgevolg hebben p', q' dezelfde f-strings en zijn ze dus f-gelijk.
- Als p en q f-gelijk zijn, en $p \in F$ dan is ϵ een f-string van p en dus ook van q. Aangezien er in een DFA geen ϵ-bogen zijn, is $q \in F$. Hetzelfde geldt in de andere richting.

□

Stelling 1.4: DFA_{min}

- DFA_{min} is een DFA, equivalent met DFA, en alle toestanden zijn f-verschillend.
- Als een DFA $N = (Q_1, \Sigma, \delta_1, q_s, F_1)$ een DFA is zonder onbereikbare toestanden en waarin elke twee toestanden f-verschillend zijn, dan bestaat er geen machine met strikt minder toestanden die dezelfde taal bepaalt. De DFA N is dan een minimale DFA of DFA_{min} voor deze taal.

Bewijs 1.3: DFA_{min}

- DFA_{min} is een DFA omdat f-gelijkheid van toestanden p en q de f-gelijkheid van $\delta(p, a)$ en $\delta(q, a)$ impliceert. Het gevolg is dat verschillende bogen met hetzelfde symbool vanuit f-gelijke p en q versmelten. Om aan te tonen dat DFA en DFA_{min} equivalent zijn, is de essentiële eigenschap dat elke equivalentie-klasse Q_i en elk element $p \in Q_i$ dezelfde f-strings heeft. Dus heeft $\tilde{q}_s$ dezelfde f-strings als q_s, en deze verzameling is dus de taal die beide DFA's bepalen. Aantonen dat een string w een f-string is van Q_i als en slechts als w een f-string is van $q \in Q_i$ gebeurt door inductie op de lengte van w, gebruikmakend van het feit dat voor elke Q_i, Q_j

$$p \in Q_i,\ q \in Q_j,\ \forall a \in \Sigma :\ Q_i \xrightarrow{a} Q_j \ \Leftrightarrow\ p \xrightarrow{a} q$$

  Tenslotte, twee verschillende toestanden Q_i, Q_j bevatten f-verschillende toestanden. Aangezien de f-strings van Q_i en Q_j die van hun elementen zijn, zijn ze f-verschillend.
- Veronderstel dat $Q_1 = \{q_s, q_1, \ldots, q_n\}$ waarbij q_s de starttoestand is. Stel dat $N_2 = (Q_2, \Sigma, \delta_2, q_s, F_2)$ een DFA is met minder toestanden dan N. Vermits in N elke toestand bereikbaar is, bestaan er strings $\forall i \in \mathbb{N}_0^+ : s_i$ zodanig dat $\delta_1^*(q_s, s_i) = q_i$. Vermits N_2 minder toestanden heeft moet voor een $i \neq j : \delta_2^*(p_s, s_i) = \delta_2^*(p_s, s_j)$. Vermits q_i en q_j f-verschillend zijn, is er een string s zodat $\delta_1^*(q_i, s) \in F_1$ en $\delta_1^*(q_j, s) \notin F_1$ of omgekeerd. Dus ook $\delta_1^*(q_s, s_i s) \in F_1$ en $\delta_1^*(q_s, s_j s) \notin F_1$ of omgekeerd. Dit betekent dat DFA_1 van de strings $s_i s$ en $s_j s$ er juist één accepteert. Maar N_2 zal beide strings $s_i s$ en $s_j s$ accepteren of geen van beiden, aangezien het parsen van s_i en s_j naar dezelfde node leidt, waarna hetzelfde pad gevolgd wordt om v te parsen. Dus kunnen de DFA's N en N_2 niet dezelfde taal bepalen.

□

Definitie 1.15: DFA isomorfisme

Een DFA $N_1 = (Q_1, \Sigma, \delta_1, q_{s_1}, F_1)$ is **isomorf** met een DFA $N_2 = (Q_2, \Sigma, \delta_2, q_{s_2}, F_2)$ als er een bijectie $b : Q_1 \to Q_2$ bestaat zodanig dat

- $b(F_1) = F_2$
- $b(q_{s_1}) = q_{s_2}$
- $b(\delta_1(q, a)) = \delta_2(b(q), a)$

Twee isomorfe DFA's bepalen dus dezelfde taal.

1.9 Myhill-Nerode relaties op Σ^*

Definitie 1.16: Fijnheid van partities

Een partitie P_1 is **fijner** dan een partitie P_2 indien

$$\forall x \in P_1,\ \exists y \in P_2 :\ x \subseteq y$$

Het omgekeerde van fijn is **grof**.

Definitie 1.17: $\sim_{\text{DFA}}$

Voor een DFA $N = (Q, \Sigma, \delta, q_s, F)$ definiëren we de relatie $\sim_{\text{DFA}}$ op Σ^* als volgt:

$$x \sim_{\text{DFA}} y \ \Leftrightarrow\ \delta^*(q_s, x) = \delta^*(q_s, y)$$

In woorden: er geldt $x \sim_{\text{DFA}} y$ als en slechts als het parsen van x en y vanuit q_s leidt tot dezelfde toestand q, dus $x, y \in \text{reach}(q)$.

Eigenschap 1.8: $\sim_{\text{DFA}}$

- Rechts congruentie van $\sim$: $\forall x, y \in \Sigma^*,\ a \in \Sigma :\ x \sim y \ \Rightarrow\ xa \sim ya$
- het aantal equivalentieklassen van $\sim$ is eindig, m.a.w. $\sim$ heeft een eindige index
- $\sim$ verfijnt $\sim_L$, of: $x \sim y \ \Rightarrow\ x \sim_L y$

Definitie 1.18: Myhill-Nerode relatie

Een equivalentierelatie $\sim$ tussen strings is een **Myhill-Nerode relatie** voor een taal L als de equivalentieklasse voldoet aan bovenstaande eigenschappen. We schrijven: $\sim$ is MN(L)

Stelling 1.5: $\mathbf{DFA}^{\mathbf{L}}_{\sim}$

Gegeven een taal L over Σ en een MN(L)-relatie $\sim$ op Σ^*, dan is $\mathrm{DFA}^{\mathrm{L}}_{\sim} = (Q, \Sigma, \delta, q_s, F)$ een DFA die L bepaalt, waarbij

- $Q = \{x_\sim \mid x \in \Sigma^*\}$
- $\delta(x_\sim, a) = (xa)_\sim$
- $q_s = \epsilon_\sim$
- $F = \{x_\sim \mid x \in L\}$

Bewijs 1.4: $\mathbf{DFA}^{\mathbf{L}}_{\sim}$

Dat δ goed gedefinieerd is, kan je bewijzen door gebruik te maken van de rechtse congruentie van $\sim$. Verder zijn alle ingrediënten van de DFA duidelijk, in het bijzonder ook dat Q slechts een eindig aantal toestanden bevat. We moeten nog bewijzen dat $L_{\mathrm{DFA}^{\mathrm{L}}_{\sim}} = L$:

$$x \in L_{\mathrm{DFA}^{\mathrm{L}}_{\sim}} \;\Leftrightarrow\; \delta^*(q_s, x) \in F \;\Leftrightarrow\; x_\sim \in F \;\Leftrightarrow\; x \in L$$

De middelste overgang bekom je door met inductie op de lengte van de string x te bewijzen dat $\delta^*(\epsilon_\sim, x) = (x)_\sim$. □

Stelling 1.6: $\sim_{\mathbf{DFA}}$ en $\sim$ zijn elkaars inverse

Voor elke taal L geldt dat de functie die DFA's van L afbeeldt op de de overeenkomstige MN(L)-relatie $\sim_{\mathrm{DFA}}$ en de functie die MN(L)-relaties $\sim$ afbeeldt op de overeenkomstige $\mathrm{DFA}^{\mathrm{L}}_{\sim}$, elkaars inversen zijn op een DFA-isomorfisme na.

Stelling 1.7: Infimumrelatie van Myhill-Nerode relaties

Als E een niet lege verzameling van MN(L) relaties is, dan is ook het infimum $\sim_{\mathrm{inf}}$ van E een MN(L)-relatie. $\sim_{\mathrm{inf}}$ is de transitieve sluiting van de unie van E. Dit betekent dat

$$x \sim_{\mathrm{inf}} y \;\Leftrightarrow\; i \in [0, n-1],\ \sim_i \in E :\ x = x_0 \sim_0 x_1 \sim_1 \ldots \sim_{n-1} x_n = y$$

Bewijs 1.5: Infimumrelatie van Myhill-Nerode relaties

We zagen al dat het infimum van een niet lege verzameling E van equivalentierelaties zelf ook een equivalentierelatie is.

- **Eindigheid:** $\sim_{\text{inf}}$ is een superset van elke willekeurige $\sim \in E$ en heeft dus minder equivalentieklassen dan $\sim$. Elke $\sim$ heeft slechts een eindig aantal equivalentieklasse, zodoende ook $\sim_{\text{inf}}$.
- **Rechts congruentie:** Stel $x \sim_{\text{inf}} y$ dan bestaat de sequentie

  $$x = x_0 \sim_0 x_1 \sim_1 \ldots \sim_{n-1} x_n = y$$

  Aangezien elke $\sim_i$ rechts congruent is, geldt voor elke $a \in \Sigma$ dat

  $$xa = x_0a \sim_0 x_1a \sim_1 \ldots \sim_{n-1} x_na = ya$$

  Bijgevolg geldt dat $xa \sim_{\text{inf}} ya$ en dus dat $\sim_{\text{inf}}$ rechts congruent is.
- **Verfijnen van $\sim_L$:** Stel $x \sim_{\text{inf}} y$ zodat er een sequentie bestaat

  $$x = x_0 \sim_0 x_1 \sim_1 \ldots \sim_{n-1} x_n = y$$

  Aangezien elke $\sim_i$ een verfijning is van $\sim_L$, geldt

  $$x_0 \in L \Leftrightarrow x_1 \in L \Leftrightarrow \ldots \Leftrightarrow x_n \in L$$

  We bekomen $x \in L \Leftrightarrow y \in L$ en dus dat $\sim_{\text{inf}}$ een verfijning is van $\sim_L$.

□

Eigenschap 1.9: MN(L) toebehorend aan een mininale DFA

Het kan nu ook bewezen worden dat de MN(L) relatie die hoort bij de minimale DFA voldoet aan de volgende conditie:

$$x \sim_{\text{inf}} y \;\Leftrightarrow\; \forall s \in \Sigma^* : \; (xs \in L \Leftrightarrow ys \in L)$$

Het is een vorm van f-gelijkheid gedefinieerd op strings in plaats van op toestanden.

Stelling 1.8: Stelling van Myhill-Nerode

Laat $L \subseteq \Sigma^*$ een taal zijn over Σ. De volgende drie uitspraken zijn dan equivalent:

$\Leftrightarrow$ L is regulier

$\Leftrightarrow$ er bestaat een Myhill-Nerode relatie voor L

$\Leftrightarrow$ definieer $\sim$ op Σ^* als volgt:

$$x \sim y \;\Leftrightarrow\; \forall s \in \Sigma^* : \; (xs \in L \Leftrightarrow ys \in L);$$

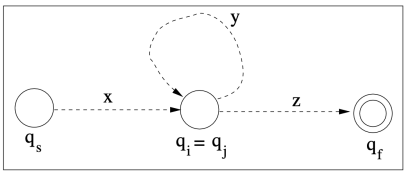
de relatie $\sim$ heeft een eindige index

1.10 Pompend lemma voor reguliere talen

Stelling 1.9: Pompend lemma voor reguliere talen

Voor een reguliere taal L bestaan een pomplengte d, zodanig dat als $s \in L$ en $|s| \geq d$, dan bestaat er een verdeling van s in stukken x, y en z zodanig dat $s = xyz$ en

- $\forall i \in \mathbb{N}_0^+ : \ xy^iz \in L$
- $|y| > 0$
- $|xy| \leq d$



Bewijs 1.6: Pompend lemma voor reguliere talen

Neem een DFA die L bepaalt, neem $d = \#Q$ en neem een willekeurige $s = a_1 \dots a_n \in L$ met $n \geq d$. Beschouw de accepterende sequentie

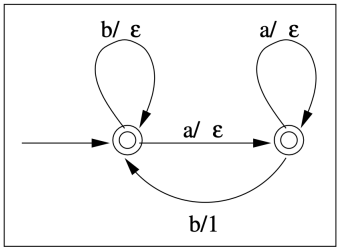
$$q_s = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} q_n \in F$$

Deze rij van toestanden heeft lengte $n+1$, wat strikt groter is dan d. Neem de eerste $d+1$ toestanden van de deze rij, dus $q_0, \dots, q_d$. Er zijn maar d verschillende toestanden, dus er zijn twee gelijke toestanden. Stel dat $q_i = q_j$ met $0 \leq i < j \leq d$, dan nemen we $x = a_1 \dots a_i$, $y = a_{i+1} \dots a_j$ en z de rest van de string. Alles volgt nu direct, zie desnoods te illustratie in de stelling. □

1.11 Varianten van eindige toestandsautomaten

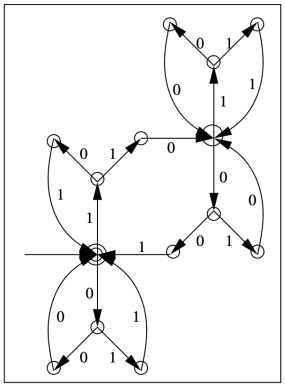
Definitie 1.19: Transducer

Een transducer zet een string om in een andere: we passen de definitie van een DFA een beetje aan, zodat ook output kan geproduceerd worden De labels zijn nu van de vorm a/x waarbij a in het inputalfabet zit en x in een outputalfabet (inbegrepen de lege string). Wat voor de / staat wordt gebruikt om de weg te vinden in de transducer alsof het een DFA was. Wat na de / staat wordt op de output gezet als die boog genomen wordt. De transducer hiernaast accepteert elke string en geeft als output een 1 voor elke b die vlak na een a komt.



Definitie 1.20: Optelchecker

Een DFA kan alleen gebruikt worden om te beslissen of een string behoort tot een taal. Zo zou je kunnen een taal definiëren van strings die correcte optellingen voorstellen en als die taal regulier is er een DFA voor bouwen. Hier is een poging: $\Sigma = \{0, 1\}$. De twee getallen die we willen optellen en het resultaat komen in binair, omgekeerd en we maken ze even lang door bij de kortste(n) wat leidende nullen toe te voegen. Dus als we 3 willen optellen bij 13, met resultaat 16, dan hebben we de drie bitstrings 11000, 10110 en 00001. Die mengen we nu systematisch, t.t.z. we maken groepjes van 3 bits die op de i-de plaats voorkomen en schrijven die groepjes achter elkaar. Dus we hebben



110 100 010 010 001

waarbij de blanco's enkel dienen om de groepering per drie te laten zien. Dus de string

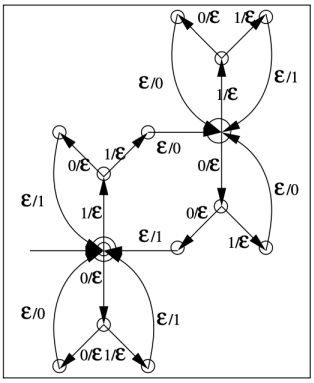
110100010010001

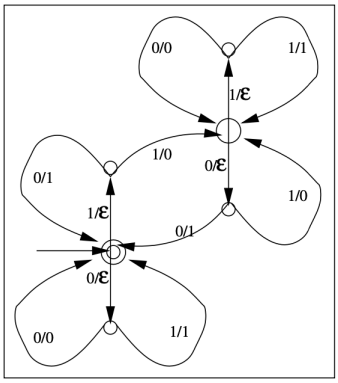
stelt de correcte optelling 3+13=16 voor. Een DFA voor de taal van correcte optellingen staat in de figuur hiernaast.

Definitie 1.21: Optel-transducer

Optellen bestaat eigenlijk in: gegeven twee getallen als input, output de som. Dat kan met een transducer: gebruik dezelfde voorstelling van de twee getallen die je wil optellen als hiervoor, en meng die op dezelfde manier, dus 3+13 wordt voorgesteld als de string

1110010100.





Definitie 1.22: Two-way finite automata

Een DFA staat ook wel gekend als een one-way finite automata, omdat de input van links naar rechts gelezen wordt en de automaat niet terug kan gaan. Een two-way finite automata 2DFA is een DFA die ook van rechts naar links kan lezen. De automaat kan dus terugkeren naar vorige toestanden.

Definitie 1.23: Büchi automaten

Büchi automaten trekken op NFA's maar je geeft er geen eindige strings aan, wel oneindige lange: er is dus geen moment waarop je string eindigt bij het doorlopen. De oneindige string s wordt aanvaard indien de rij toestanden waarlangs je passeert oneindig dikwijls een aanvaardende toestand heeft.

1.12 Contextvrije talen en hun grammatica

Definitie 1.24: Contextvrije grammatica - CFG

Een contextvrije grammatica is een 4-tal (V, Σ, R, S) waarbij

- V een eindige verzameling is van niet-eindsymbolen of **variabelen**
- Σ een eindige alfabet is van eindsymbolen of **terminals**, disjunct met V
- R is een eindige verzameling van regels of **producties**: een regel is een koppel van één niet-eindsymbool en een strings van elementen uit $V \cup \Sigma_\epsilon$. We schrijven $u \to v$ voor een regel (u, v).
- $S \in V$ is het **startsymbool**

Definitie 1.25: Afleiding m.b.v. een CFG

Gegeven een CFG (V, Σ, R, S). Een string f over $V \cup \Sigma_\epsilon$ wordt afgeleid uit een string b over $V \cup \Sigma_\epsilon$ m.b.v. de CFG als er een eindige rij strings $s_0, \ldots, s_n$ bestaat zodanig dat

- $s_0 = b$
- $s_n = f$
- $\forall i \in [0, n-1] : \ s_i \to s_{i+1}$

We noteren: $s_i \Rightarrow s_{i+1}$ en $b \Rightarrow^* f$.

Definitie 1.26: Taal bepaald door een CFG

De taal L_{CFG} bepaald door een CFG (V, Σ, R, S) is de verzameling strings s over Σ zodanig dat $S \Rightarrow^* s$. In formulevorm:

$$L = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

Definitie 1.27: Contextvrije taal - CFL

Een taal L is **contextvrij** indien er een CFG bestaat die L bepaalt, en dus $L = L_{\text{CFG}}$.

Definitie 1.28: Equivalente CFG's

Twee contextvrije grammatica's CFG_1 en CFG_2 zijn **equivalent** als ze dezelfde taal bepalen, en dus $L_{\text{CFG}_1} = L_{\text{CFG}_2}$.

Definitie 1.29: Chomsky normaalvorm

Een CFG (V, Σ, R, S) heeft de **Chomsky normaalvorm** als elke regel één van de volgende vormen heeft:

- $A, B,\ C \in V :\ A \to BC$
- $A \in V,\ \alpha \in \Sigma :\ A \to \alpha$
- $S \to \epsilon$

Stelling 1.10: Equivalente Chomsky normaalvorm

Voor elke CFG (V, Σ, R, S) bestaat er een equivalentie CFG in Chomsky normaalvorm.

Bewijs 1.7: Equivalente Chomsky normaalvorm

We vertrekken van een willekeurige CFG en transformeren hem terwijl we equivalentie bewaren naar Chomsky Normaalvorm.

1. Vervang alle voorkomens van S door een nieuw symbool X, en daarna voeg de regel $S \to X$ toe. Zo komt S enkel links voor. Zo wordt 1 van de voorwaarden voldaan. Deze stap is evident equivalentiebewarend.

2. Neem de verzameling van alle regels. Itereer de volgende operatie zo lang tot een fixpunt bereikt wordt:

   - Selecteer een regel $A \to \epsilon$ en een regel $B \to \alpha A \beta$.
   - Voeg de regel $B \to \alpha\beta$ toe.

   Deze stap is equivalentiebewarend. Daarvoor moet aangetoond worden dat elke iteratie equivalentiebewarend is. Kortom, als $B \to \alpha A \beta$ toegevoegd wordt, dan kan een string geschreven worden met die regel als en slechts als de strings geschreven kan worden zonder deze regel. Het is evident omdat het herschrijven met $B \to \alpha\beta$ kan gesimuleerd worden met $B \to \alpha a \beta$ en $A \to \epsilon$. Tenslotte: verwijder alle regels $A \to \epsilon$ voor $A \neq S$. Ook deze stap is equivalentiebewarend. Neem een parsing boom $S \to \ldots$; de wortel is S, elke node is gelabeld met een symbool A, elke node die geen blad is eheft een geordende rij kinderen gelabeld $BC \ldots D$ zodat $A \to BC \ldots D$ een regel is, en de bladeren van de bomen vormen een sequentie van terminale symbolen. Dan is het vrij duidelijk dat elke string van de CFL zo'n parsing boom heeft. Ook dat deze boom kan hervormd worden tot een boom die gebruik maakt van de toegevoegde regels en geen gebruik maakt van regels $A \to \epsilon$.

3. Nu willen we afgeraken van de regels van de vorm $A \to B$. Voor een regel van de vorm $\mathcal{E} = A \to B$ en een regel van de vorm $\mathcal{R} = B \to \gamma$, definieer de regel $\mathcal{U}(\mathcal{E}, \mathcal{R}) = A \to \gamma$. Zolang er regels van de vorm $\mathcal{E} = A \to B$ zijn (waaron B ook een niet-terminal is) en regels van de vorm $\mathcal{R} = B \to \gamma$ zijn, voeg de regel $\mathcal{U}(\mathcal{E}, \mathcal{R})$ toe. Nadat dit eindigt, verwijderen we uit de bekomen grammatica alle regels van de vorm $A \to B$. Deze stap is equivalentiebewarend. Neem een string s die kan afgeleid worden met de regels van de vorm $A \to B$. Dan kan s ook afgeleid worden zonder deze regels.

4. We hebben nu nog drie soorten regels te behandelen:

   (a) $A \to \gamma$ waar γ uit juist twee niet-eindsymbolen bestaat.

   (b) $A \to \gamma$ waar γ uit minstens twee symbolen bevat: vervang elke terminal a door een niet-terminaal A_a en voeg de regel $A_a \to a$ toe.

   (c) eventueel $S \to \epsilon$: die mag blijven.

5. De regels van de vorm
   $$n > 2 : A \to X_1 X_2 \ldots X_n$$
   vervang je door
   $$A \to X_1 Y_1,\ Y_1 \to X_2 Y_2, \ldots,\ Y_{n-2} \to X_{n-1} X_n$$

Hierbij is het duidelijk dat bij de transformatie naar de Chomsky Normaalvorm de grammatica equivalent blijft. □

1.13 Push-down automaat

Definitie 1.30: Aanvaarding van een string s door een PDA

Een string s wordt aanvaard door een PDA indien s kan worden opgesplitst in delen $i \in [1, m] : \ w_i \in \Sigma_\epsilon$, er toestanden $j \in [0, n] : \ q_j$ zijn en stacks $k \in [0, m] : \ \text{stack}_k \in \Gamma^*$, zodanig dat

- $\text{stack}_0 = \epsilon$: de stack is leeg in het begin
- $q_0 = q_s$: we vertrekken in de begintoestand
- $q_m \in F$: we komen aan in een eindtoestand met een lege string
- $x, y \in \Gamma_\epsilon : \ (q_{i+1}, y) \in \delta(q_i, w_{i+1}, x)$ en $x, y \in \Gamma_\epsilon, \ t \in \Gamma^* : \ \text{stack}_i = xt, \ \text{stack}_{i+1} = yt$

Definitie 1.31: Taal bepaald door een PDA

De taal L bepaald door een PDA bestaat uit alle strings die door de PDA aanvaard worden.

1.14 Equivalentie van CFG en PDA

Stelling 1.11: Equivalentie van CFG en PDA

Elke push-down automaat bepaalt een contextvrije taal en elke contextvrije taal wordt bepaald door een push-down automaat.

1.15 Pompend lemma voor contextvrije talen

Stelling 1.12: Pompend lemma voor contextvrije talen

Voor een contextvrije taal L bestaan een getal p (de pomplengte) zodanig dat elke string s van L met lengte strikt groter dan p kan opgedeeld worden in 5 stukken $u, v, x, y, z \in \Sigma^*$ zodanig dat $s = uvxyz$

- $\forall i \in \mathbb{N}_0^+ : \ uv^i xy^i z \in L$
- $|vy| > 0$
- $|vxy| \leq p$

Bewijs 1.8: Pompend lemma voor contextvrije talen

Als L eindig is dan is de stelling triviaal voldaan. Immers, dan heeft L een langste string s; neem $p = |s|$. Dan zijn er geen strings met lengte strikt groter dan p, en dan is de stelling triviaal voldaan. □

2 Talen en berekenbaarheid

3 Herschrijfsystemen

4 Andere rekenparadigmas

5 Talen en complexiteit