Samenvatting van

Automaten en Berekenbaarheid

W. Van Onsem
KU Leuven
Departement Computerwetenschappen
Declaratieve Talen en Artificiële Intelligentie (DTAI)

Chomsky hiërarchie (§2-§3)

Orakel (§3.17)

O^L

Orakelmachine: Een Orakelmachine O^L is een TM met een orakel voor de taal L. Men kan dit voorstellen als een TM met een extra band (de orakel-tape) en drie extra toestanden $q_?$, q_ja en q_nee. De TM raadpleegt het orakel door de string op de orakelband te schrijven en in toestand $q_?$ te gaan. Het orakel brengt wanneer de string tot L behoort de machine in toestand q_ja en anders in toestand q_nee. L is mogelijk onbeslisbaar.

Reducties

$L_1 \leq_T L_2$

Turing-reductie van talen: Een taal L_1 is Turingreduceerbaar naar taal L_2, indien L_1 beslisbaar is relatief ten opzichte van L_2. dit wil zeggen: er bestaat een orakelmachine O^{L_2} die L_1 beslist. We schrijven $L_1 \leq_T L_2$.

Toepassingen:
- Filosofie
- Complexiteitstheorie

Turing (§3.1-3.9)

TM

Turingmachine: Een TM is een 7-tal $\langle Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r\rangle$ met:
- Q een eindige verzameling toestanden is;
- Σ een invoeralfabet dat niet # bevat;
- Γ een tape alfabet met $\# \in \Gamma$ en $\Sigma \subsetneq \Gamma$;
- $q_s \in Q$ de start-toestand, $q_a \in Q$ de accepterende eindtoestand, $q_r \in Q$ de verwerpende eindtoestand met $q_r \neq q_a$.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times L, S, R$ de totale transitiefunctie.

$s \in L_\text{TM}$

String door een TM geaccepteerd: De TM wordt geïnitialiseerd in toestand q_s met de invoerstring op de tape, op de overige plaatsen staat een #, en de leeskop op het meest linkse teken. Zolang de TM niet in q_a en q_r zit, wordt het karakter uitgelezen onder de leeskop. Op basis van de transitiefunctie δ wordt een karakter onder de leeskop geset, de leeskop naar links/rechts verplaatst en komt de machine in een nieuwe toestand. Als de TM na een eindig aantal stappen in q_a komt wordt de string aanvaard.

$s \in \infty_\text{TM}$

String waarvoor de TM niet stopt: Een string s behoort tot ∞_TM indien bij invoer van s, de TM nooit in q_a of q_r terecht komt.

Herk

Herkennen: Een Turingmachine TM herkent L_TM: Een taal L waarvoor zo'n TM bestaat noemt men een herkenbare taal.

Besl

Beslissen: Een TM beslist een taal L indien de TM L herkent en bovendien $\infty_\text{TM} = \emptyset$. Een taal L waarvoor zo'n TM bestaat noemt men een beslisbare taal.

Enumereerbaar

EM

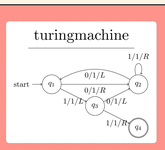
Enumeratormachine: Een EM is een TM met volgende uitbreidingen:
- Een enumeratortoestand q_e;
- Een uitvoerband met uitvoermarker;
- De transitiefunctie $\delta : Q \times \Gamma \to Q \times \Gamma \times \Gamma_\text{e} \times \{L, S, R\}$.

De machine start met lege band en lege uitvoerband, in de q_e toestand en werkt als een TM. Wanneer iets op de uitvoer wordt geschreven verschuift de schrijfkop naar rechts. Wanneer de machine in toestand q_e komt, wordt op de uitvoer de uitvoermarker geschreven en komt de EM opnieuw in q_s terecht.

Herk ≡ EM

Enumereren equivalent met herkennen: Een taal geënumereerd door een EM is herkennbaar en elke herkenbare taal wordt door een EM geënumereerd.

turingmachine



Rice

Stelling van Rice: Voor elke niet-triviale taalinvariante eigenschap P van Turingmachines geldt $L_P = \{\langle M\rangle \,|\, M \text{ voldoet aan } P\}$ is niet beslisbaar.

Turing-berekenbaar

Turing-berekenbare functie: Een functie f is Turing-berekenbaar indien er een Turingmachine bestaat die bij input s uiteindelijk stopt met $f(s)$ op de band.

$L_1 \leq_m L_2$

Reductie van talen: Een taal L_1 (over Σ_1) is reduceerbaar naar een taal L_2 (over Σ_2) indien er een afbeelding $f : \Sigma_1^* \to \Sigma_2^*$ bestaat zodat $f(L_1) \subseteq L_2$ en $f(\bar{L_1}) \subseteq \bar{L_2}$ en f Turing-berekenbaar is. We noteren dat door $L_1 \leq_m L_2$.

$\Sigma(n)$

Bezige bever: de bezige bever is een totale functie $\Sigma : \mathbb{N} \to \mathbb{N}$ die met invoer n, het maximale aantal 1'en uitgeschreven door een Turingmachine met n toestanden en een initieel lege invoer associeert.

Toepassingen:
- Programmeertalen
- Levens redden (WO-II)

Context-sensitief (§2.26)

CSG

Context-sensitieve grammatica over een alfabet Σ: Een CSG over Σ is een 4-tal $\langle V, \Sigma, R, S\rangle$ met:
- V een eindige verzameling niet-eindsymbolen;
- Σ een eindig alfabet van eindsymbolen disjunct met V;
- R een eindige verzameling regels; een regel is een koppel van $\alpha A\beta \to \alpha\gamma\beta$ met $\alpha, \beta, \gamma \in (\Sigma \cup V)^*$ en $A \in V$;
- $S \in V$ is het startsymbool.

CSL

Context-sensitieve taal: Een taal L is context-sensitief indien er een CSG bestaat zodat $L = L_{CSG}$. De verzameling van contextsensitieve talen duiden we aan met CSL.

LBA

Lineair begrensde automaat: Een LBA is een TM die niet leest of schrijft buiten het deel van de band dat initieel invoer bevat.

CSG ≡ LBA

Equivalentie van CSG en LBA: Elke LBA bepaalt een context-sensitieve taal en elke context-sensitieve taal wordt bepaald door een LBA.

$s \in L_\text{CSG}$ wordt beslist in

$$\mathcal{O}\left(c^n\right)$$

Toepassingen:
- Natuurlijke taalverwerking
- Webservers
- Gegevensbanken

Context-vrij (§2.19-§2.25)

CFG

Contextvrije grammatica over een alfabet Σ: $\langle V, \Sigma, R, S\rangle$ is een CFG over Σ indien:
- V een eindige verzameling niet-eindsymbolen is;
- Σ een eindig alfabet van eindsymbolen disjunct met V;
- R een eindige verzameling regels; een regel is een koppel van één niet-eindsymbool en een string van elementen uit $V \cup \Sigma$,; we schrijven de twee delen van zulk een koppel met een $\to$ ertussen;
- $S \in V$ is het startsymbool.

CFL

Contextvrije taal: Een taal L is contextvrij indien er een CFG bestaat zodat $L = L_{CFG}$. De verzameling van contextvrije talen duiden we aan met CFL.

$b \Rightarrow^\star_{CFG} f$

Afleiding met behulp van een contextvrije grammatica: Gegeven een CFG $\langle V, \Sigma, R, S\rangle$. Een string f over $V \cup \Sigma_\text{e}$ wordt afgeleid uit een string b uit $V \cup \Sigma_\text{e}$ met behulp van de CFG indien er een eindige rij strings $s_0, s_1, \ldots, s_n$ bestaat zodat:
- $s_0 = b \wedge s_n = f$;
- s_{i+1} verkregen wordt uit s_i (voor $i < n$) door in s_i een niet-eindsymbool X te vervangen door de rechterkant van een regel waarin X links voorkomt.

We noteren $s_i \Rightarrow s_{i+1}$ en $b \to^\star f$

CFG ∼ L_CFG

Een contextvrije grammatica bepaalt een taal: De taal L_{CFG} bepaald door een CFG $\langle V, \Sigma, R, S\rangle$ is gedefinieerd als: $L_{CFG} = \{s \in \Sigma^* | S \Rightarrow^* s\}$.

PDA

Push-down automaat: Een PDA is een 5-tal $\langle Q, \Sigma, \Gamma, \delta, q_s, F\rangle$ met:
- Q een eindige verzameling toestanden;
- Σ een eindig inputalfabet;
- Γ een eindig stapelalfabet;
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ de overgangsfunctie;
- $q_s \in Q$ de starttoestand;
- $F \subseteq Q$ de verzameling eindtoestanden.

$s \in L_\text{PDA}$

String aanvaard door een PDA: Een string s wordt aanvaard door een PDA $\langle Q, \Sigma, \delta, q_s, F\rangle$ indien s kan worden opgesplitst in delen $w_i, i = 1 \ldots m$ ($w_i \in \Sigma$), er toestanden $q_j, j = 0 \ldots m$ ($q_j \in Q$) zijn, en stapels $\text{stapel}_k, k = 0 \ldots m$ ($\text{stapel}_k \in \Gamma^*$) zodat:
- $\text{stapel}_0 = \varepsilon \wedge q_0 = q_s \wedge q_m \in F$;
- $(q_{i+1}, y) \in \delta(q_i, w_{i+1}, x)$ waarbij $x, y \in \Gamma_\varepsilon$ en $\text{stapel}_i = xt$, $\text{stapel}_{i+1} = yt$ met $t \in \Gamma^*$.

CFG ≡ PDA

Equivalentie van CFG en PDA: Elke PDA bepaalt een contextvrije taal en elke contextvrije taal wordt bepaald door een PDA.

$L \in \text{CFL}$

Pompend lemma voor CFL: Voor een contextvrije taal L bestaat er een pomplengte d zodat elke string $s \in L$ met $|s| \geq d$ kan opgedeeld worden in $s = uvxyz$ met $u, v, x, y, z \in \Sigma^*$ zodat:
- $\forall i \geq 0 : uv^i x y^i z \in L$;
- $|vy| > 0$;
- $|vxy| \leq d$.

$s \in L_\text{CFG}$ wordt beslist in

$$\mathcal{O}\left(n^3\right)$$

Toepassingen:
- Extraheren van informatie
- Compilers
- Natuurlijke taalverwerking

Regulier (§2.4-§2.18)

RE

Reguliere expressie over een alfabet Σ: E is een RE over Σ indien E van de vorm is:
- ε;
- ϕ;
- a met $a \in \Sigma$,
- $(E_1 E_2)$ met E_1 en E_2 RE's zijn over Σ;
- $(E_1 | E_2)$ met E_1 en E_2 RE's zijn over Σ, dan is $L_E = L_{E_1} L_{E_2}$.
- $(E_1 | E_2)$ met E_1 en E_2 Re's zijn over Σ, dan is $L_E = L_{1} \cup L_2$.
- $(E_1)^*$ met E_1 een RE is over Σ, dan is $L_E = L_{E_1}^*$.

RegLan

Reguliere taal: Een taal die door een reguliere expressie bepaald wordt is een reguliere taal. De verzameling van reguliere talen duiden we aan met RegLan.

RE ∼ L_RE

Een reguliere expressie bepaalt een taal: Een RE E bepaalt een taal L_E over hetzelfde alfabet Σ als volgt:
- Als $E = \varepsilon$, dan is $L_E = \{\epsilon\}$
- Als $E = \phi$, dan is $L_E = \emptyset$
- Als $E = a$ met $a \in \Sigma$, dan is $L_E = \{a\}$
- $(E_1 E_2)$ met E_1 en E_2 RE's zijn over Σ, dan is $L_E = L_1 L_2$.
- $(E_1 | E_2)$ met E_1 en E_2 RE's zijn over Σ, dan is $L_E = L_1 \cup L_2$.
- $(E_1)^*$ met E_1 een RE is over Σ, dan is $L_E = L_{E_1}^*$.

NFA

Niet-deterministische eindige toestandsautomaat: Een NFA is een 5-tal $\langle Q, \Sigma, \delta, q_s, F\rangle$ met:
- Σ een eindig alfabet;
- Q een eindige verzameling toestanden;
- $F \subseteq Q$ de verzameling eindtoestanden.

GNFA

Gegeneraliseerde niet-deterministische eindige toestandsautomaat: Een GNFA is een NFA $\langle Q, \Sigma, \delta, q_s, F\rangle$ met:
- slechts één eindtoestand $F = \{q_F\}$ zodat $q_F \neq q_s$;
- de bogen bevatten een reguliere expressie als label;
- $\forall q_i, q_j \in Q \setminus \{q_s, q_F\} : \exists ! \in \text{RegExp}(\Sigma) : \delta(q_i, \epsilon) = q_j$;
- $\forall q_i \in Q \setminus \{q_s, q_f\} : \exists ! \in \text{RegExp}(\Sigma) : \delta(q_s, \epsilon) = q_j$;
- $\forall q_i \in Q \setminus \{q_s, q_f\} : \exists ! \in \text{RegExp}(\Sigma) : \delta(q_i, \epsilon) = q_f$.

DFA

Deterministische eindige toestandsautomaat: Een DFA is een 5-tal $\langle Q, \Sigma, \delta, q_s, F\rangle$ met:
- Q een eindige verzameling toestanden;
- Σ een eindig alfabet;
- $\delta : Q \times \Sigma \to Q$ de overgangsfunctie van de automaat;
- $q_s \in Q$ de starttoestand;
- $F \subseteq Q$ de verzameling eindtoestanden.

$f^=$

f-gelijk: Twee toestanden $q_1, q_2 \in Q$ in een DFA zijn f-gelijk indien:

$$\forall w \in \Sigma^* : \delta^*(q_1, w) \in F \leftrightarrow \delta^*(q_2, w) \in F$$

Anders zijn ze f-verschillend.

$L \in \text{RegLan}$

Pompend lemma voor RegLan: Voor een reguliere taal L bestaat een pomplengte d zodat elke string $s \in L$ met $|s| \geq d$ kan opgedeeld worden in $s = xyz$ met $x, y, z \in \Sigma^*$ zodat:
- $\forall i \geq 0 : xy^i z \in L$;
- $|y| > 0$;
- $|xy| \leq d$.

Myhill-Nerode relaties

$\sim_\text{mn}$

Myhill-Nerode relatie: Een Myhill-Nerode relatie over een taal L is een equivalentie-relatie $\sim_\text{mn}$ die aan volgende eigenschappen voldoet:
- $\forall x, y \in \Sigma^*, a \in \Sigma : x \sim_\text{mn} y \to xa \sim_\text{mn} ya$;
- $\sim_\text{mn} \subseteq \sim_L$;
- $\sim_\text{mn}$ heeft een eindig aantal equivalentieklassen.

$\sim_\text{sup}$

Supremum van MN relaties: Het supremum van MN relaties $\sim_1$ en $\sim_2$ is de transitieve sluiting van de disjunctie:

$$x \sim_\text{sup} y \leftrightarrow (x \sim_1 y) \vee (x \sim_2 y)$$

$\sim_\text{mn} \to$ DFA

MN relatie naar DFA: Gegeven een taal L over een alfabet Σ en een MN-relatie $\sim$ over Σ^*. Dan is $\langle Q, \Sigma, \delta, q_s, F\rangle$ een DFA die L bepaalt met: $Q = \{x_- | x \in \Sigma\}$, $q_s = \epsilon_-$, $F = \{x_- | x \in L\}$ en $\delta(x_-, a) = (xa)_-$.

$\sim_\text{sup} \equiv \sim_\text{mn}$

Supremum van MN relaties is een MN relatie: Gegeven twee MN relaties $\sim_1$ en $\sim_2$, dan volgt het supremum aan alle eigenschappen van een MN relatie.

$\mathcal{P} \leq \mathcal{P}$

Fijnere partitie: Een partitie $\mathcal{P}_1$ is fijner dan $\mathcal{P}_2$ indien elk element van $\mathcal{P}_1$ vervat zit in een element van $\mathcal{P}_2$.

DFA ≡∼mn

MN relaties tegenover DFA: De overgangen van DFA naar de NM relatie, en van de MN relatie naar een DFA zijn elkaars inversen – op isomorfisme na.

Min DFA

Minimale DFA: Gegeven een DFA zonder onbereikbare toestanden en waarbij elke twee verschillende toestanden, f-verschillend zijn, hiervoor bestaat er geen DFA met strikt minder toestanden die dezelfde taal bepaalt.

DFA ≅ DFA

Isomorfe DFA: Twee DFA's $\langle Q_1, \Sigma, \delta_1, q_{s,1}, F_1\rangle$ en $\langle Q_2, \Sigma, \delta_2, q_{s,2}, F_2\rangle$ zijn isomorf indien er een bijectie $b : Q_1 \to Q_2$ bestaat met:
- $b(F_1) = F_2 \wedge b(q_{s,1}) = q_{s,2}$;
- $\forall q \in Q_1 : \forall a \in \Sigma : b(\delta_1(q, a)) = \delta_2(b(q), a)$

$s \in L_\text{RE}$ wordt beslist in

$$\mathcal{O}\left(n\right)$$

Toepassingen:
- Extraheren van informatie
- Compilers
- Internetprotocollen
- Digitale elektronica

NFA → DFA

NFA naar DFA: De gelijkheden geven voor elke NFA $\langle Q_n, \Sigma, \delta_n, q_{s,n}, F_n\rangle$ een equivalente DFA $\langle Q_d, \Sigma, \delta_d, q_{s,d}, F_d\rangle$:

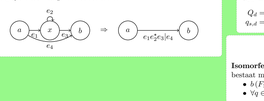
$$\begin{aligned} Q_d &= \mathcal{P}(Q_n) & F_d &= \{q | q \in Q_d \wedge Q \cap F_n \neq \emptyset\} \\ q_{s,d} &= \delta_n^*(q_{s,n}, \epsilon) & \delta_d(Q, a) &= \bigcup_{q \in Q} \delta_n^*(\delta_n(q, a), \epsilon) \end{aligned}$$

NFA ≡ NFA

Equivalentie van twee NFA's: Twee NFA's worden equivalent genoemd als ze dezelfde taal bepalen.

Reduce

Reductie-stap: Neem een toestand $x \in Q \setminus \{q_s, q_f\}$ en toestanden $a, b \in Q$ en voer volgende reductie uit:



$s \in L_\text{NFA}$

String aanvaard door een NFA: Een string s wordt aanvaard door een NFA $\langle Q, \Sigma, \delta, q_s, F\rangle$ indien s kan geschreven worden als $a_1 a_2 \ldots a_n$ met $a_i \in \Sigma_\varepsilon$, en er een rij toestanden $t_1 t_2 \ldots t_{n+1}$ bestaat zodat:
- $t_1 = q_s$;
- $t_{i+1} \in \delta(t_i, a_i)$;
- $t_{n+1} \in F$.

Herschrijfsystemen (Λ-calculus; §4)

$A \sim B$

Equivalente termen: Twee termen A en B zijn equivalent indien er een sequentie van herschrijfregels bestaat zodat $A \to \ldots \to X$ en $B \to \ldots \to X$.

$f\ x$

Currying: $f\ x$ lezen we als f toegepast op argument x. Elke functie in λ-calculus heeft maar één argument (maar resulteert soms in een nieuwe functie).

$(\lambda x\ .\ x)\ x$

Vrij voorkomen: Een voorkomen van de variabele x is vrij in de expressie E indien:
- E gelijk is aan x;
- E gelijk is aan $\langle A\ B\rangle$ en het voorkomen van x is vrij in A of B;
- E gelijk is aan $\langle \lambda\ y\ .\ A\rangle$ en $y \neq x$.

CR-1

Church-Rosser I: Gegeven twee expressies E_1 en E_2. Indien $E_1 \leftrightarrow^* E_2$, dan bestaat er een expressie E zodat $E_1 \to^* E$ en $E_2 \to^* E$. Bijgevolg kan geen expressie geconverteerd worden naar twee verschillende normaalvormen.

CR-2

Church-Rosser II: Gegeven twee expressies E en N met N in normaalvorm. Dan bestaat er een reductierij in normaalorde $E \to^* N$. De normaalorde bepaalt dat de meest linkse buitenste redex eerst moet worden gereduceerd.

Herschrijfregels λ-calculus

$\to^\alpha$

Alfa-conversie: Hernoem een variabele in een λ-expressie. Formeel:

$$\lambda\ x\ .\ E \to^\alpha \lambda\ y\ .\ E'$$

In E worden alle vrije voorkomens van x vervangen. y is geen variabele gebonden in E.

$\to^\beta$

Beta-reductie: Herschrijf naar een kopie van het lichaam van de λ-expressie waarin de vrije voorkomens van de parameter vervangen zijn door het argument. Formeel:

$$(\lambda\ x\ .\ E)\ y \to^\beta E'$$

$\to^\eta$

Eta-conversie: Herschrijf een λ-expressie die de parameter toepast op een functie naar de functie. Formeel:

$$\lambda\ x\ .\ F\ x \to^\eta F$$

Met F een functie waarin x niet vrij voorkomt.

$\leftarrow^\beta$

Beta-abstractie: Herschrijf een term naar een functie die wordt toegepast op een argument. β-abstractie wordt gebruikt om de equivalentie tussen termen aan te tonen. Formeel:

$$(\lambda\ x\ .\ E')\ y \leftarrow^\beta E$$

Ingebouwde functies λ-calculus

$\mathbb{N}$

Natuurlijke getallen: Een natuurlijk getal $n \in \mathbb{N}$ wordt meestal voorgesteld als $c_n = \lambda\ f\ x\ .\ f^n\ x$. Volgende operaties zijn gedefinieerd:
- optelling $A_+ = \lambda\ x\ y\ p\ q\ .\ x\ p\ (y\ p\ q)$
- vermenigvuldiging $A_* = \lambda\ x\ y\ z\ .\ x\ (y\ z)$
- exponent $A_{\text{expp}} = \lambda\ x\ y\ .\ y\ x$

$\vec{x}$

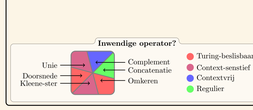
Lijsten: Lijsten worden meestal door het gelinkte-lijst-paradigma voorgesteld. Elke node heeft een Hoofd en een Staart. Men stelt een node voor met Cons:
- Hoofd $= \lambda\ c\ .\ c\ (\lambda\ a\ b\ .\ a)$
- Staart $= \lambda\ c\ .\ c\ (\lambda\ a\ b\ .\ b)$
- Cons $= \lambda\ a\ b\ f\ f\ a\ b$

$\mathbb{B}$

True, False en If: Booleaanse waarden $b \in \mathbb{B}$ worden meestal voorgesteld als volgt:
- **true** $= \lambda\ x\ y\ .\ x$
- **false** $= \lambda\ x\ y\ .\ y$
- **if** $= \lambda\ x\ y\ z.\ (x\ y\ z)$

Legende

Inwendige operator?



Taal-aspecten (§2.1-§2.3)

Σ

Alfabet: Een alfabet Σ is een set karakters die relevant zijn in een gegeven taal L.

$s \in \Sigma^\star$

String: Een string over een alfabet Σ is een sequentie van 0, 1 of meer karakters uit Σ. $\Sigma^\star$ is de verzameling van alle strings over Σ.

$L \subseteq \Sigma^\star$

Taal: Een taal over een alfabet Σ is een verzameling strings over Σ.

$L_1 L_2$

Concatenatie van talen: Gegeven twee talen L_1 en L_2 over eenzelfde alfabet Σ, dan is de concatenatie $L_1 L_2 = \{xy | x \in L_1, y \in L_2\}$. L^* is de concatenatie van n keer de taal L.

$L^\star$

Kleene-ster: De Kleene-ster van een taal L is de unie van nul, één of meer concatenaties van de taal: $L^* = \cup_{n=0}^\infty L^i$.

Andere paradigma's (§5)

PCP

Post Correspondence Problem: Gegeven een set S van tuples $\{\alpha, \beta\}$ met $\alpha, \beta \in \Sigma^*$ over een alfabet Σ. Bestaat er een sequentie elementen $s_0 s_1 \ldots s_n$, zodat de concatenatie van het eerste element van de tuples gelijk is aan de concatenatie van het tweede element van de tuples?

Primitief recursief

Primitief recursieve functies: Een primitief recursieve functie is inductief gedefinieerd als volgt:
- De nulfunctie: $\text{nul} : \mathbb{N} \to \mathbb{N} : x \mapsto 0$;
- De successorfunctie: $\text{succ} : \mathbb{N} \to \mathbb{N} : x \mapsto x + 1$;
- De projectiefunctie: $p_i^n : \mathbb{N}^n \to \mathbb{N} : (x_1, x_2, \ldots, x_n) \mapsto x_i$;
- De compositiefunctie: $\text{Cn}[f, g_1, g_2, \ldots, g_m] : \mathbb{N}^k \to \mathbb{N} : \vec{x} \mapsto f(g_1(\vec{x}), g_2(\vec{x}), \ldots, g_m(\vec{x}))$ met $f : \mathbb{N}^m \to \mathbb{N}$ en $g_i : \mathbb{N}^k \to \mathbb{N}$ primitieve functies;
- De primitieve recursie:

$$\Pr[f, g] : \mathbb{N}^k \to \mathbb{N} : (\vec{x}, y) \mapsto \begin{cases} f(\vec{x}) & \text{if } y = 0 \\ g(\vec{x}, y - 1, \Pr[f, g](\vec{x}, y - 1)) & \text{otherwise} \end{cases}$$

met $f : \mathbb{N}^k \to \mathbb{N}$ en $g : \mathbb{N}^{k+2} \to \mathbb{N}$ primitieve functies;

Ack

Ackermann-functie: Een functie die niet kan worden voorgesteld aan de hand van primitieve recursie is de Ackermann-functie:

$$\text{Ack} : \mathbb{N}^2 \to \mathbb{N} : (x, y) \mapsto \begin{cases} y + 1 & \text{if } x = 0 \\ \text{Ack}(x - 1, 1) & \text{if } y = 0 \\ \text{Ack}(x - 1, \text{Ack}(x, y - 1)) & \text{otherwise} \end{cases}$$

μ-recursief

Recursieve functies: Een μ recursieve functie is een uitbreiding op primitief recursieve functies met de onbegrensde minimalisatie:

$$\text{Mn}[f] : \mathbb{N}^k \to \mathbb{N} : \vec{x} \mapsto \begin{cases} y & \text{if } f(\vec{x}, y) = 0 \wedge \forall z < y : f(\vec{x}, z) \neq 0 \\ \textbf{undef} & \text{otherwise} \end{cases}$$

met $f : \mathbb{N}^{k+1} \to \mathbb{N}$ een μ-recursieve functies.

Cel

Cellulaire automaat: Een cellulaire automaat is een verzameling identieke eindige toestandsmachines die allen regelmatig van toestand veranderen. De toestand hangt af van de eigen toestand en deze van de buren. In het 2-dimensioneel geval hangt deze toestand dus van $Q_\text{self} \times Q_\text{left} \times Q_\text{right} \times Q_\text{up} \times Q_\text{down}$.

DNA

DNA rekenen: DNA rekenen is een rekenmechanisme dat werkt aan de hand van een set dominostenen. Onder bepaalde voorwaarden kunnen dominostenen samenklitten tot clusters. Door dit op grote schaal toe te passen en vervolgens te filteren, kan men sommige problemen uitrekenen.

Ant

Ant rekenen: We beschouwen een verzameling "mieren": objecten die een eerder toevallig gedrag vertonen. Wanneer een mier dichter bij de oplossing van een probleem komt, trekt deze andere mieren aan met behulp van "feromonen". Door evaporatie verdwijnt het effect van de feromonen met de tijd.

Problemen

A_ϕ

Acceptatie: $A_\phi = \{\langle M, s\rangle | \text{En } \phi \ M \text{ accepteert de string } s\}$.

H_ϕ

Stopt: $H_\phi = \{\langle M, s\rangle | \text{En } \phi \ M \text{ stopt bij invoer } s\}$.

E_ϕ

Leegheid: $E_\phi = \{\langle M\rangle | \text{En } \phi \ M \text{ herkent de lege taal}\}$.

EQ_ϕ

Gelijkheid: $EQ_\phi = \{\langle M_1, M_2\rangle | \text{Voor } \phi\text{'s } M_1 \text{ en } M_2\text{: } L_{M_1} = L_{M_2}\}$.

ES_ϕ

Lege string: $ES_\phi = \{\langle M\rangle | \text{En } \phi \ M \text{ accepteert de lege string}\}$.

Regular_ϕ

Regulier: $ES_\phi = \{\langle M\rangle | \text{En } \phi \ M \text{ herkent een reguliere taal}\}$.

ALL_ϕ

Alle strings: $ES_\phi = \{\langle M\rangle | \text{En } \phi \ M \text{ herkent alle strings } \Sigma^*\}$.

Beslisbaar?