Theorie-III

Formale Sprachen und Automatentheorie 1+2

Walter Hower

Preliminary

- introduction (↔, hower@hs-albsig.de)

- organization
  - lectures (+ self study)
    - times: according to announcement(s)
    - locat.: announcement, currently: online
  - examination (in writing, 80 min.; as part of …
    – no support material)

Literature

- Walter Hower
  Diskrete Mathematik
  – Grundlage der Informatik
  2., erweiterte und verbesserte Auflage
  2021
  De Gruyter Oldenbourg
  978-3-11-069554-0 (Broschüre)
  978-3-11-069567-0 (eBook)
  https://doi.org/10.1515/9783110695557

… Literature

- Pavel Pudlák
  Logical Foundations of Mathematics
  and Computational Complexity
  – A Gentle Introduction
  Springer International Publishing
  Switzerland
  2013
  978-3-319-3426-8-9 (softcover)
  978-3-319-0011-8-0 (hardcover)
  10.1007/978-3-319-0011-9-7 (DOI)

… Literature

- Mikhail J. Atallah, Marina Blanton (eds.)
  Algorithms and Theory of Computation
  Handbook
  2nd edition
  Volume 1:
  General Concepts and Techniques
  978-1-13811-393-0 (paperback), 2017
  Volume 2 [1st ed.: 978-1-58488-820-8 (hardback), 2010]:
  Special Topics and Techniques
  2nd edition: October 18, 2019
  Chapman & Hall / CRC / Taylor & Francis

… Literature

- C. H. Papadimitriou, K. Steiglitz
  Combinatorial Optimization:
  Algorithms and Complexity
  2nd edition
  Dover
  1998
  978-0-486-40258-1

Thanks to Ken S. for discussion!

... Literature

- J. E. Hopcroft, R. Motwani, J. D. Ullman

  Thanks to JDU for discussion!

  - Introduction to Automata Theory, Languages, and Computation
    3rd, new internat. edition, Pearson, 2014
    978-1-2920-3905-3 (paperback) / 978-1-2920-5015-7 (eBook)
    978-1-2920-5616-6
  - Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit
    3., aktual. Aufl., Pearson Studium, 2011
    978-3-86–894082-4 (printed)
    978-3-86–326509-0 (e-book)

Fach-Lektor...

... Literature

- H. R. Lewis, C. H. Papadimitriou
  Elements of the Theory of Computation
  2nd, international, edition
  Pearson
  1998
  978-0-13272-741-9
  978-0-13262-478-7 (hardback)

… Literature

- Arindama Singh
  Elements of Computation Theory
  Springer-Verlag London
  2009
  978-1-4471-6142-4 (soft)
  978-1-84882-496-6 (hard)
  10.1007/978-1-84882-497-3 (DOI)

… Literature

- Juraj Hromkovič
  Theoretische Informatik
  Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie
  5. Auflage
  Springer Vieweg Fachmedien
  2014
  978-3-658-06432-7 (softcover)
  10.1007/978-3-658-06433-4 (DOI)

…

... Literature

- International Journal of
  Foundations of Computer Science
  World Scientific
  0129-0541 (print)
  1793-6373 (online)

... Literature

- Walter Hower
  Informatik-Bausteine
  – Eine komprimierte Einführung
  978-3-658-01279-3 (Softcover)
  https://doi.org/10.1007/978-3-658-01280-9
  Springer Nature Vieweg Fachmedien
  2019
  Buch-Reihe *Studienbücher Informatik*
  2522-0640 (paper), 2522-0659 (el.)

Overview

0 Discrete Maths

0.-1 Basis - Natürl. Zahlen, Notationen

- $\mathcal{N}$:= (Menge der) *natürlichen* Zahlen
- [Guiseppe] Peano-Axiome [1858 – 1932]
  - $0 \in \mathcal{N}$ [$\in$: Element von]
  - 0 [kleinste Zahl] $\neq s(n)$ [Nachfolger($n_{\in \mathcal{N}}$)]
  - $\forall n \in \mathcal{N}\ \exists s(n)$ [$\forall$ „für alle" All-Quantor, $\exists$ „es gibt" Existenz-Q.]
  - $n_1 \neq n_2 \Rightarrow s(n_1) \neq s(n_2)$ [s „injektiv"]
  - $(0 \in T_{\subseteq \mathcal{N}}) \wedge ((\forall n \in \mathcal{N})\ n \in T \rightarrow s(n) \in T)$ $\Rightarrow$ $T = \mathcal{N}$
- *F*kt./Index-Notation: $f(e) =: f_e$ [...]

... Basis - Mengen

- Definitions-Menge: Input-Menge auf die eine Funktion wirkt
- Werte-Menge [$\supseteq$ Bild-Menge]: Menge aus der die Fkt. Output-Werte nimmt
- Bild-Menge [$\subseteq$ Werte-Menge]: Menge der Werte die echt produziert werden
- Einschränkung $f^{D \rightarrow C}{}_{|S}$: $(D \supseteq)\ S \rightarrow C$
  f mit Definitions-Menge S $(\subseteq D)$

... Basis - Funktionen

- (totale) *Funktion* f: $D \rightarrow C$
  weist jedem Element x der Definitions-Menge („*domain*") D genau ein Element $f(x)$ der Werte-Menge („*co-domain*") C *zu*
  [$\forall\ x \in D\ \ \exists!\ f(x) \in C$]
  Üb.: ...
  genau 1
- partielle *Funktion*:
  nicht zwingend alle $x_{\in D}$ haben $f(x)_{\in C}$
  [Def.-Lücken erlaubt]
  CT-H., X-Prod.

... Basis - Funktionen

- Injektion : „1-to-1" (injektive) F.
  $x_1 \neq x_2 \quad \Rightarrow \quad f(x_1) \neq f(x_2)$
  # 1-to-1-F. aus k- in $n_{\geq k}$-Menge: ... $n^{\underline{k}}$
  ∄ Injekt. mit $|D| > |C|$ (cf. *Schubfach-Prinzip* in „Zähl-Techniken")
  $\Rightarrow |D| \leq |C|$ (nicht die Umkehrung) Kontrapositivum
- Surjektion : „onto" (surjektive) F.
  $\forall\ y \in C\ \ \exists\ x \in D$: $f(x) = y$
  Bild-Menge(F) = Werte-M.(F)
  # Surjektionen aus n- in $k_{\leq n}$-Menge: ... $k! \cdot s_2(n,k)$
  ∄ Surjekt. mit $|D| < |C|$
  $\Rightarrow |D| \geq |C|$ (nicht die Umkehrung)

... Basis - Funktionen

- Bijektion : inj. und surj. (bij.) F.

  „1-to-1-Korrespondenz" $\Rightarrow |D| = |C|$ (<u>nicht</u> die Umkehrung)

  Üb.: ... $|D| = |C| \Rightarrow \exists$ Bijektion

- Inverse $f^{-1}: Y \to X$
  s. d. für jedes $y \in Y$ das eindeutige $f^{-1}(y) =: x \in X$ existiert mit $f(x) = y$ für die Bijektion $f: X \to Y$

  f invertierbar: $\exists$ Inverse

... Basis - Funktionen

- Komposition („nach"): $g \circ f\,(x) := g(f(x))$
  $f: X \to Y,\ f(x) =: y;\ g: Y \to Z,\ g(y) =: z$
  $h := g \circ f: X \to Z,\ h(x) := g(f(x)) = g(y)$
  $[\forall\ x \in X] \quad \in \quad Z$

  üblicherweise: $f(g(x)) \neq g(f(x))$; Übung: ...

- inverse Komposition: $h^{-1}{}_{[\text{nicht „ } 1/h\text{"}]} := (g \circ f)^{-1}$
  [f, g Bijektionen: $f: X \to Y$, $g: Y \to Z$;
  $h := g \circ f: X \to Z;\ f^{-1}: Y \to X,\ g^{-1}: Z \to Y$]
  $h^{-1} := (g \circ f)^{-1} = f^{-1} \circ g^{-1}: Z \to X$
  $h^{-1}(z) := f^{-1}(g^{-1}(z)) := f^{-1}(y) := x$
  $[\forall\ z \in Z] \in X$ Ü.: $f(x) := 2x =: y$, $g(y) := y+1$; $h^{-1}(z) := \ldots \quad x$

… Basis - Rundung auf natürl. Zahl

- floor : $\mathbf{R}^{+}_{(0)} \to \mathcal{N}$:
  $\lfloor x \rfloor$:= größte natürliche Zahl $\leq x$
- ceiling : $\mathbf{R}^{+}_{(0)} \to \mathcal{N}$:
  $\lceil x \rceil$:= kleinste natürliche Zahl $\geq x$
- round : $\mathbf{R}^{+}_{(0)} \to \mathcal{N}$:
  $\lfloor x \rceil$:= „wähle"($\lfloor x \rfloor$,$\lceil x \rceil$)

… Basis - Kardinalität

- $\mathcal{N}^{+} := \mathcal{N} \setminus \{0\} = \{1, 2, 3, \ldots\}$ $[=: \mathcal{N}_1]$

  „ohne"
- $\mathcal{N}^{-} := \{-n \mid n \in \mathcal{N}^{+}\} = \{-1, -2, -3, \ldots\}$

  Menge aller „Negativ-*n*" <u>für die gilt</u> dass das „Eingabe-*n*" echt positiv natürlich ist
- $|\mathcal{N}^{+}| = |\mathcal{N}^{-}|$? Ja! Beweis: Bijektion

$\{(1;-1), (2;-2), (3;-3), \ldots\}$ $f_{+}(n) := -n$

$\{(-1;1), (-2;2), (-3;3), \ldots\}$ $f_{-}(n) := -n$

… Basis - Kardinalität

- $|\mathcal{N}| = |\mathcal{N}^+|$? Ja! Beweis: Bijektion

{(0;1), (1;2), (2;3), …}

$\boxed{f_{\mathcal{N}}(n) := n + 1}$

$\boxed{f_{\mathcal{N}+}(n) := n - 1}$

{(1;0), (2;1), (3;2), …}

… Basis - Kardinalität

- $|\mathcal{N}| = |\mathcal{Z}|$? Ja! Beweis: Bijektion

	0	1	2	3	4	5	6	7	8	9	…
…											

{(0;0), (1;1), (2;−1), (3;2), (4;−2), (5;3), (6;−3), (7;4), (8;−4), (9;5), …}

$f_{\mathcal{N}} : \mathcal{N} \rightarrow \mathcal{Z}$ Übung: $f(n) :=$ …

… Basis - Kardinalität

$f_Z : Z \to N$

{(0;0), (1;1), (−1;2), (2;3), (−2;4), (3;5), (−3;6), (4;7), (−4;8), (5;9), …}

Üb.: $f_Z : Z \to N$

$f(z) := \ldots$

Üb.: …; $|Q|$; Achtung: $|\Re|$, $|\wp(N)|$ ☺

0.0 Structural Induction
0.0.0 Foundations

- induction (Peano axiom 5)
  - basis (initial case)
  - hypothesis (given)
  - step (next)

- structure (recursively defined)
  - initial (basis)
  - given (hypothesis)
  - next (step)

0.0.1 Structures

- expression
  - initial : variable, number
  - given : (sub-)expression/s
  - next : compound expression
- tree
  - initial : root …
  - given : (sub-)trees
  - next : compound tree

… Structures

- <u>e</u>xpression (recursively defined)
  - recursion basis : $E \in \{\text{var., number}\}$
  - recursion step : $\underline{E}_1 + \underline{E}_2$, $\underline{E}_1 \cdot \underline{E}_2$, $(\underline{E})$
- statement : $|(| =_{e.} |)|$
  - basis : $|(|_E = 0 = |)|_E$
  - hypothesis : $|(| =_{e.\_given} |)|$
  - step (e._given → e._comp.) : $|(|_{next} =_{e.\_compound} |)|_{next}$

... Structures

- case analysis
  - $\underline{E}_1 + \underline{E}_2$: $|(|_{\underline{E_1}+\underline{E_2}} = |(|_{\underline{E_1}} \boxed{+} |(|_{\underline{E_2}}$

    $=_! |)|_{\underline{E_1}} \boxed{+} |)|_{\underline{E_2}} = |)|_{\underline{E_1}+\underline{E_2}}$
  - $\underline{E}_1 \cdot \underline{E}_2$: $|(|_{\underline{E_1}\cdot\underline{E_2}} = |(|_{\underline{E_1}} \boxed{+} |(|_{\underline{E_2}}$

    $=_! |)|_{\underline{E_1}} \boxed{+} |)|_{\underline{E_2}} = |)|_{\underline{E_1}\cdot\underline{E_2}}$
  - $(\underline{E})$: $|(|_{(\underline{E})} = |(|_{\underline{E}} \boxed{+ 1}$

    $=_! |)|_{\underline{E}} \boxed{+ 1} = |)|_{(\underline{E})}$

... Structures

- $\underline{t}$ree (recursively defined)
  - recurs. basis : R[oot]
  - recurs. step : R_{next} connected to $\underline{T_1}, \ldots, \underline{T_k}$ (with e_i and n_i, $1 \le i \le k$)

    \# edges \# nodes
- statement : $e = n-1$
  - basis : $e_R = 0 = 1-1 = n_R-1$
  - hypothesis : $e =_{t.\_given} n-1$
  - step$_{(t._given \to t._comp.)}$: $e_{\text{next}} =_{t.\_compound} n_{\text{next}} - 1$

… Structures

$$
\begin{aligned}
e_{\text{next}} &= \sum_{i:=1}^{k} e_i + k =_! \sum_{i:=1}^{k} (n_i - 1) + k \\
&= \sum_{i:=1}^{k} n_i - k + k = \sum_{i:=1}^{k} n_i \\
&= n_{\text{next}} - 1
\end{aligned}
$$

0.0.2 Finale

- recursively defined structures
- proof technique: induction
- structural induction
  - common principle
    - recurs./ind. basis
    - given structure
    - rec./ind. step
  - principally like for $\mathcal{N}$ (:= {0, 1, 2, …})
  - creative part: (structure) enlargement

0.1 Recurrence Relation

- increm. procedure $\rightarrow$ closed formula
- structure
  - initial value
  - recurrence principle
    - based on 1 step before
    - (creative) construction step
  - development (2 alternatives)
    - backward substitution
    - forward substitution
- proof (of the statement created)

… Recurrence Relation

- illustration: n-ary bit string (vector)
  - incrementing effect
  - adding 1 leading bit ($n{-}1 \stackrel{+1}{\rightarrow} n$)
    doubles the # input combinations:
    $c_n = 2 \cdot c_{n-1} = 2^1 \cdot 2^{n-1} = 2^{[1+(n-1)]} = 2^n$
    (# rows in a truth table for n bool. var.)
  - example: z_n := # possibilities that $\exists!$ zer0 (*false*)

… Recurrence Relation

$n := 0$ $\quad \boxed{}$ $\quad z_0 = 0$

$n := 1$

0	←	(1 x)
1		(+ z_0)

$z_1 = 1$

$n := 2$

0 0		(*true*)	
0 1	←	formerly "1" everywhere	(1 x)
1 0	←	formerly ∃! "0"	(+ z_1)
1 1			

$z_2 = 2$

… Recurrence Relation

$n := 3$

0 0 0			
0 0 1			
0 1 0			
0 1 1	←	formerly "1" everywhere	(1 x)
1 0 0			
1 0 1	←	formerly ∃! "0"	(+ z_2)
1 1 0	←		
1 1 1			

$z_3 = 3$

⋮

Form. Spr. u. Autom.-Th. 1+2 Theor. Inform. III Okt.–Dez. 2022

… Recurrence Relation

$z_0 := 0$ ← initial value

$$z_{n(>0)} := z_{n-1} + \binom{n-1}{0} = z_{n-1} + 1$$

new leading *true* — new leading *false* — recurrence principle

backward substitution: ← development

$$\begin{aligned} &= z_{n-2} + 1 + 1 = z_{n-2} + 2 \\ &= z_{n-3} + 1 + 2 = z_{n-3} + 3 \\ &\vdots \\ &= z_{n-n} + n = z_0 + n = n \end{aligned}$$

… Recurrence Relation

forward substitution: ← development

$$\begin{aligned} z_1 &:= z_0 + 1 = 0 + 1 = 1 \\ z_2 &:= z_1 + 1 = 1 + 1 = 2 \\ z_3 &:= z_2 + 1 = 2 + 1 = 3 \\ &\vdots \\ z_n &:= n \end{aligned}$$

… Recurrence Relation

statement: $z_n = n$ $\left[= \binom{n}{1}\right]$ (cf. Combinations)

proof: induction on n

i) $n := 0$

principle: $z_0 = 0 \leftarrow$
formula : $z_0 = 0 \leftarrow$ } =

ii) $n_{(>0)}-1 \rightarrow n$

princ.: $z_n := z_{n-1} + 1 \overset{!}{=} n-1 + 1 = n$

form.: $z_n = n$ = …

exercise: r-1/2/3: …

… Recurrence Relation

1) e_n := # edges in a complete graph (n := # nodes)

$e_{…}$:= … , $e_n := e_{n-1}$ …

backward substitution: …

forward substitution: …

statement: e_n = …

proof: induction on n …

… Recurrence Relation

2) d_n := *d*iameter in a grid (q.; n := # *n*odes, … := …)

$d_{…}$:= … , $d_{…}$:= $d_{…}$ $_{-1}$ …

backward substitution: …

forward substitution: …

statement: $d_{…}$ = …

proof: induction on …
…

3) c_h := # *c*onnections in an h-dimensional hyper-cube

1 Complexity

- parameters
  - space (logarithmic cost)
  - time (uniform cost)
- application
  - algorithm
  - problem (via best [*worst*-case] solver)
  - classes of problems (of similar complexity)

… Complexity

- units of cost

– *l*ogarithmic cost: # bits: space complexity

integer ≥ 0: $l(n) := \begin{cases} 1 & ;\ n \leq 1 \\ 1 + \lfloor \mathrm{ld}(n) \rfloor & ;\ n \geq 2 \end{cases}$

$\mathrm{ld}(n) := \log_2(n)$

– uniform cost: cost(operation) := 1

operations (in principle): time complexity

… Complexity

- estimation possibilities

notation for "$g \in$ complexity class f" [no "="]

– order (of magnitude) – upper bound:
$g = \mathrm{O}(f)$ iff $\exists c > 0\ \exists n_0 \in N\ \ \forall n \geq n_0$: $|g(n)| \leq c \cdot |f(n)|$

– order (of magnitude) – lower bound:
$g = \Omega(f)$ iff $\exists c > 0\ \exists n_0 \in N\ \ \forall n \geq n_0$: $|g(n)| \geq c \cdot |f(n)|$

– order (of magnitude) – rate of growth:
$g = \Theta(f)$ iff $\exists$ $(0 <)$ $c_1 \leq c_2$ $\exists n_0 \in N\ \ \forall n \geq n_0$:
$c_1 \cdot |f(n)| \leq |g(n)| \leq c_2 \cdot |f(n)|$

re- uest level				
	f_{O}	•	•	•
	f_{Θ}	•	•	•
	f_{Ω}	•	•	•
		best	avg.	worst case

… Complexity

- classes of probl. [formally: languages]
  - $(D)P_{\mathrm{TIME}}$: (deterministically) *p*olynomial ("efficient")

    examples:

    - 2-SAT (max. 2 literals/clause), Horn-SAT (≤ 1 positive lit.) [$\Theta(n)$] …
    - shortest paths ([Dijkstra: $\Theta(n^2)$], [Floyd-Warshall: $\Theta(n^3)$])
    - L$_{\text{inear}}$ P$_{\text{rogramming}}$ ([Khachiyan-1979], [Karmarkar-1984])
      (ellipsoid) (interior point)
    - Euler cycle (closed path visiting each *edge* once) …

[Digression]

- vertex degree g_v: # edges incident to v
- isolated vertex v: $g_v = 0$
- in-degree(v) := # edges w. end vertex v
- out-degree(v) := # edges with start v

[… Digression]

(di-graph) G eulerian

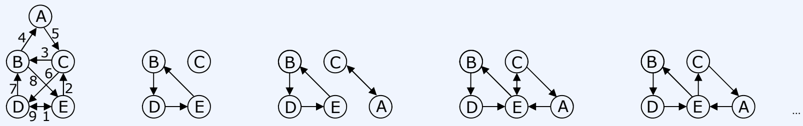
iff

$\forall\ u,v \in V$ with $g_{u/v} > 0$: $\exists$ path $u \rightarrow v$ ("reachability")

and

$\forall\ v \in V$: in-degree(v) = out-degree(v)

exercise:



… Complexity

– NP_{TIME} : *n*on(-determ.) *p*olyn. ([det.] exponential)

[*parallel* time compl.: polyn.] $NP_{\text{TIME}} \setminus (D)P_{\text{TIME}} =: AE$ [≠ { } ?]

[apparently exponential] ∗

(inofficial – own definition)

examples:

- (job-shop) scheduling […]
- Graph Isomorphism [GrIs] […]
- Hamilton Cycle [HC] (closed path via each *node* once)

a given *yes* instance can (determ.) be certified in polyn. time …

$(D)P_{\text{TIME}} \subseteq NP_{\text{TIME}}$ $[\Rightarrow NP_{\text{TIME}} \not\subset (D)P_{\text{TIME}}]$!

$(D)P_{\text{TIME}} \neq NP_{\text{TIME}}$ $[\ (D)P_{\text{TIME}} \subset NP_{\text{TIME}}\ ]$?

∗ https://www.claymath.org/sites/default/files/***p***vs***np***.pdf

- polynomial reducibility (of probl. by alg., f_r.)
  - $p_1 \leq_p p_2$ iff $a_2 \cong f_r^p(a_1)$ — (input length), computes, difficulty, computes — similar. via red. funct. f_r: $w \in L_1 \Leftrightarrow f_r(w) \in L_2$ (Integer LP)
- *NP-completeness* (TSP, [3-]SAT, ILP, CSP) e.g.
  - (new) problem $p \in NP$ (S. A. Cook's theorem [1971])
  - each probl. in *NP* polynom. reducible to p (transitive) …

    (when $p \in P$ could be shown would imply $P = NP$) …
- *NP-hardness* (e.g.: k^{th} heaviest subset)
  - $p \in NP$?
  - each problem in *NP* polynom. reducible to p

k^{th} heaviest subset:

given: $c_1, \ldots, c_n, k, l \in N$

decide: $\exists$ k distinct subsets

$s_1, \ldots, s_k \subseteq \{1, \ldots, n\}$ s.t.

$\sum_{j \in s_i} c_j \geq l$ for $i = 1, \ldots, k$ [?]

…

… Complexity

- *co-NP*$_{\text{TIME}}$

  – introduction

    - HC "complement" [HCc] (no such tour)
    - HCc $\in$ *NP* ? (HC $\in$ *NP*)
    - in contrast to *P* (= *Co-P* $\subseteq$ *Co-NP*)
      A $\in P \rightarrow$ Ac $\in P$ $\Rightarrow$
      Ac $\in P \rightarrow$ Acc (= A) $\in P$ $\Rightarrow$
      A $\in P \leftrightarrow$ Ac $\in P$ $\Rightarrow$
      (A $\notin P \rightarrow$ Ac $\notin P$) $\wedge$ (Ac $\notin P \rightarrow$ A $\notin P$)

… Complexity

– *NP*$_{\text{TIME}}$ recall

  - a given *yes* instance can be certified in polynomial time
  - HC $\in$ *NP*$_{\text{TIME}}$

– *co-NP*$_{\text{TIME}}$

  - class of complements of *NP*$_{\text{TIME}}$-problems
  - HCc $\notin$ *NP*$_{\text{TIME}}$?

… Complexity

– illustration

- HCc $\in$ *co-NP* (HC *NP-complete*)
- *NP* $\neq$ *co-NP* ? (HCc $\notin$ *NP* ?)
- complem. of *NP-complete* pr. $\in$ *NP* iff *NP* = *co-NP*
  $\notin$ [?] $\neq$
- [if] (*D*)*P* = *NP* $\underset{P \text{ closed w.r.t. complement}}{\rightarrow}$ *NP* = *co-NP* (= *P*)

  contrapositive: …
- (*D*)*P* [= *Co*-(*D*)*P*] $\subseteq$ *NP* $\cap$ *co-NP*

… Complexity

- (*D*)*P-SPACE*

  – polynomial *space* (w.r.t. input size)
  – *P* $\subseteq$ *NP* $\cap$ *co-NP* $\subseteq$ *NP* $=^?$ *co-NP* $\subseteq$ *P-SPACE*
  – *P-SPACE* $\ni$ k^{th} heaviest subset $\notin_?$ *NP*
  – *P-SPACE-complete*
    - (new) problem *p* $\in$ *P-SPACE*
    - each pr. in *P-SPACE* polynomially *reducible* to *p*
  – = *NP-SPACE*
  – $\supseteq$ (*D*)*P-TIME* [*a-TIME* $\subseteq$ *a-SPACE*]

2 Languages

- Σ (non-empty, finite) alphabet
- Σ^k {strings of length k $_{(\geq 0)}$ | symbol $\in \Sigma$}
- $\Sigma^+ := \bigcup_{k:=1}^{\infty} \Sigma^k$
- $\Sigma^* := \bigcup_{k:=0}^{\infty} \Sigma^k = \Sigma^0 \cup \Sigma^+ = \{\varepsilon\} \cup \Sigma^+$ "Kleene closure/star"
- $0 < |\Sigma| < \infty$
- $0 < |\Sigma^k| = |\Sigma|^k < |\Sigma^+| = \omega = 1 + \omega = |\Sigma^*| = \infty$
- type i language: $\exists$ type $i_{\text{max.}}$ grammar $\rightarrow$

…

… Languages

- "type 0" (grammars)

$G := (S, N, \Sigma, P)$

$S \in N :=$ {non-terminals}, $\Sigma :=$ {terminals}, $A := N \cup \Sigma$ $[N \cap \Sigma = \varnothing]$,
$\underset{\dots}{P} := \{[A^*NA^* \ni]\ \alpha \longrightarrow \beta\ [\in A^*]\}$

evtl. re-labelling: $N :=$ …

$L(G) := \{w \in \Sigma^* \mid S \overset{+_{\dots}}{\underset{G}{\Longrightarrow}} w\}$

recursively enumerable

… Languages

- "type 1" (grammars)

$G := (S, N, \Sigma, P)$

$S \in N :=$ {non-terminals}, $\Sigma :=$ {terminals},
$A := N \cup \Sigma$; $u, v \in A^*$; $H \in N$; $z \in A^+$; $\alpha := uHv$; $\beta := uzv$;

$$P := \{\alpha \to \beta\} \,\dot\cup\, \begin{cases} \{S \to \varepsilon\} & ; S \notin \beta \\ \{\ \} & ; S \in \beta \end{cases}$$

$L(G) := \{w \in \Sigma^* \mid S \overset{+}{\underset{G}{\Longrightarrow}} w\}$ *context-sensitive*

$\Sigma_{ex.} := \{a, b, c\}$

(example: $L = \{a^k b^k c^k \mid k \geq 0\}$, not "context-free"$_{\to}$)

… Languages

- "type 2" grammar

$G := (S, N, \Sigma, P)$

$S \in N :=$ {non-term.}, $\Sigma :=$ {term.}, $P := \{[N \ni]\ H \to z\ [\in (N \cup \Sigma)^*]\}$

variables | alphabet := {symbols} | *Productions*

$L(G) := \{w \in \Sigma^* \mid S \overset{+}{\underset{G}{\Longrightarrow}} w\}$ *context-free*

ex. (G): S, $N := \{S\}$, $\Sigma := \{a, b, +, -, (,)\}$
$P := \{S \to a,\ S \to b,\ S \to S+S,\ S \to S-S,\ S \to (S)\}$
$L(G) = \ldots$ not "regular", but c.-f.

… Languages

- L context-free

  $\Rightarrow$ („Pumping Lemma for c.-f. l.“)

  $\exists\ n_{[>0]}\ :\quad n \leq |z|\ ,\ z \in L$

  $\Rightarrow$

  $z := uvwxy\ ,\quad 0 \overset{\text{no air}}{<} |vx| \leq |vwx| \leq n\ (\leq |z|)\ ,$

  pumping ☺

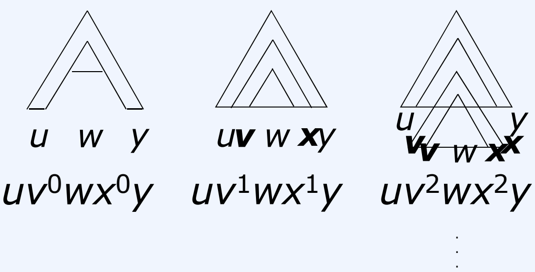
  ↓(pair)↓

  $\forall\ i_{[\geq 0]}\!:\ \ z_i := uv^iwx^iy \in L$

… Languages

- <u>proof</u>: induction on parse tree length
  - sketch ☺:

background ($L \neq \varnothing$):
binary parse trees +
Chomsky Normal Form:
$\underset{[\in N^2][\in \Sigma]}{N_0 \rightarrow N_1N_2 \mid a}$ $[\neq \varepsilon]$



  - en détail:
    - [HMU-2007, p. 281–283]
    - [HMU-2011, S. 321–323]

... Languages

- $\Sigma := \{b, e, s\}$, $L := \{b^p e^p s^p \mid p \geq 1\}$
- Illustration:
  - ε, be, s, $bbees$, $besbes \notin L \neq \emptyset$
  - bes, $bbeess$, $bbbeeesss$, $bbbbeeeessss \in L$
- L context-free $\Rightarrow$ (Pumping Lemma)

  $\exists\ 0 < n \leq |z|$, $z := uvwxy := b^n e^n s^n \in L$,

  $0 < |vx| \leq |vwx| \leq n < 3n \underset{\text{here}}{=} |z|$,

  $\forall i_{[\geq 0]}$: $z_i := uv^i wx^i y \in L$;

  indirect proof, contrapositive: $\neg\ \forall\ i$: $z_i \in L$

... Languages

$\kappa := vwx_{[\in \Sigma^+]}$, $\rho := uwy_{[\in \Sigma^+]}$; $z := b^n e^n s^n$

$|\kappa| \leq n \Rightarrow \kappa$ doesn't have b and s together

$|\kappa| \not> n \Rightarrow \kappa$ without any b or without an s

1) κ does not contain any $b \Rightarrow vx$ neither

   $\Rightarrow$ ρ has $n \cdot b$ $\wedge$ less than $n \cdot$ (s or e)

   $\Rightarrow$ # b, e, s not in balance $\Rightarrow z_0 \notin L_{\neg \text{ c.-f.}}$

2) κ does not contain any $s \Rightarrow vx$ neither

   $\Rightarrow$ ρ has $n \cdot s$ $\wedge$ less than $n \cdot$ (b or e)

   $\Rightarrow$ # b, e, s not in balance $\Rightarrow z_0 \notin L_{\neg \text{ c.-f.}}$

... Languages

- Pumping Lemma for context-free lang.
  - ¬ pumping lemma (dilemma ☺) ⇒ ¬ c.-f.
  - indirect proof technique, contrapositive
  - pair structure à la PDA; expressiveness:
    - higher than in regular languages (no stack)
    - lower compared to c.-sensitive l. (tripel ok)
  - indicator for the language complexity
  - background for prog. lang., compiler

... Languages

- "type 3" (grammars)

$$G := (S, N, \Sigma, P)$$

$S \in N :=$ {non-t.}, $\Sigma :=$ {t.}, $P := \{[N \ni]\ H \longrightarrow z\ [\in \{\varepsilon\} \cup \Sigma N^{0/1}]\}$

right-linear
...
(no mixture)

$$L(G) := \{w \in \Sigma^* \mid S \overset{+}{\underset{G}{\Rightarrow}} w\}$$

regular

exercise: L_x = all strings over $\Sigma := \{0,1\}$ with exactly one "0": 1*01*; ...

$L_y = \{a^k b^l \mid k, l \geq 0\}$ $(\Sigma_{ex.} := \{a, b\})$ ...

- closure properties of *regular* languages

…

- union
- intersection
- complement
- difference (intersection with complement)
- concatenation
- Kleene closure (“star”)

- (characteristic) feature of *regular* lang.

  - type-3 pumping lemma

L regular

$\Rightarrow$

$\exists\, n \geq 1 \;\; \forall\, w \in L$ with $|w| \geq n \;\; \exists\, xyz = w$
with $y \neq \varepsilon$, $|xy| \leq n$, $xy^iz \in L \;\; \forall\, i \geq 0$

[proof: via *DFA* …]

$\Sigma_{ex.} := \{a, b\}$

<u>exercise</u>: $L := \{a^kb^k \mid k \geq 0\}$ <u>not</u> regular …

[just the same: $\Sigma := \{(,)\}$]

… Languages

Pumping Lemma (type 3) boot camp ☺

- $L := \{a^k b^k \mid k \geq 0\}$ regular? $\Rightarrow_{(?)}$

  $w := a^n b^n = xyz;\ |w| \geq n \geq 1,\ y \neq \varepsilon,\ |xy| \leq n,\ |y| > 0$

  $w = xyz = a^n b^n = a^{n-j} a^j b^n,\ j > 0$

  $xy^0z = \underset{(n-j \neq n,\ j > 0)}{a^{n-j} b^n \notin L}$ $[i := 0 \Rightarrow \neg\,(xy^iz \in L \quad \forall\, i \geq 0)]$

  L not regular (but c.-f.)

… Languages

- # Chomsky hierarchy

$$|\Sigma^*| =_{\Sigma \neq \varnothing} |\mathcal{N}| \overset{\dots}{<} 2^{|\mathcal{N}|} = 2^{|\Sigma^*|} \cong |2^{\Sigma^*}|$$

$$\underline{L_{\text{type-3}} \subset L_{\text{type-2}} \subset L_{\text{type-1}}} \subset \underline{L_{\text{decidable}}} \subset \underline{L_{\text{type-0}}} \subset L_{\text{general}}^{\,:= \{L \mid L \subseteq \Sigma^*\}}$$

$(L_0\text{-}C)$ $\neg$ semi-decidable
$\Rightarrow (L_0,\ L_0\text{-}C)$ $\neg$ decidable

$\boxed{L\text{-}C := \textit{l}\text{anguage-complement}}$

[L "decidable" $\leftrightarrow$ (L "*semi*-decidable" $\wedge$ L-C "*semi*-decidable")]

…
recursively enumerable

$$L_{\substack{\text{prefix-free}\\ \text{regular}}} \subset \left\{ \begin{array}{l} L_{\substack{\text{prefix-free}\\ \text{determinist.}\\ \text{context-free}}} \\ L_{\text{type-3}} \end{array} \right\} \subset L_{\text{det. c.-f.}} \underset{\dots}{\subset} L_{\text{type-2}}$$

... Languages

- "word problem" [WP]

  - given
    - G
    - w [$\in \Sigma^*$]

  - decide
    - $w \in L(G)$ [?]

... Languages

– complexity/(un-)computability – of $L(G)$:

- regular ("type 3") : WP $\in$ $(D)P$ [$O(|w|)$]
- determ. context-free : WP $\in$ $(D)P$ [$O(|w|)$]
- context-free ("type 2") : WP $\in$ $(D)P$ [$O(|w|^3)$]
- context-sensitive ("type 1") : WP *P-SPACE-complete*
- recursively enumerable ("type 0"): WP not computable [„just" ☺ in general]

3 Automata

3.1 Type 3

- *f*inite *a*utomaton [FA]
  - for regular language[s]/expression[s]
  - *D*FA/*N*FA: *d*eterministic/*n*on-determin. FA
    - DFA (Σ, Q, F, q_0, δ)
      - Σ : [finite] set of [input] symbols
      - Q : [finite] set of states (evtl. incl. *d*ead end)
      - F : [finite] set of *f*inal/accepting states [$\subseteq Q$]
      - q_0 : [initial] start state [$\in Q$]
      - δ : [simple] transition function: $Q \times \Sigma \rightarrow Q$
    - L(DFA) := $\{w \in \Sigma^* \mid$ DFA "accepts" $w\}$

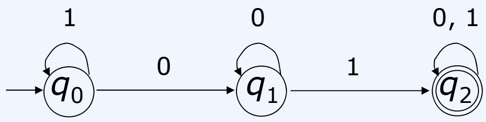
… Type 3

– illustration

- $L(\text{DFA}) = \{p01s \mid p, s \in \Sigma^*\}$
  - 01, 010, 1001, 00011 $\in L(\text{DFA})$
  - ε, 0, 1, 10, 11, 100 $\notin L(\text{DFA})$
  - construction $(\Sigma, q_0, \delta, Q, F)$
    - » $\Sigma := \{0, 1\}$
    - » q_0 := start state
    - » δ: [exercise …]
      $\delta(q_0, \ ) := q_0$, $\delta(q_0, \ ) := q_1$,
      $\delta(q_1, \ ) := q_1$, $\delta(q_1, \ ) := q_2$,
      $\delta(q_2, \ ) := q_{\dots}$, …
    - » $Q := \{q_0, q_1, q_2\}$
    - » $F := \{q_2\}$

… Type 3

– transition diagram/graph



– transition table

δ	**0**	**1**
$\rightarrow \boldsymbol{q_0}$	q_1	q_0
$\boldsymbol{q_1}$	q_1	$q_{2(F)}$
$\boldsymbol{q_{2(F)}}$	$q_{2(F)}$	$q_{2(F)}$

... Type 3

- extended transition function
  $\bar{\delta}(q,w)$ [$\in Q$]: $Q \times \Sigma^* \to Q$

  – principle of induction [on $|w|$]

  – recursive construction

    » $\bar{\delta}(q,\varepsilon) := q$ [$|\varepsilon| = 0$]
    » $w := xa$ (prefix $x \in \Sigma^*$, 1-char. suffix $a \in \Sigma$) [$\in \Sigma^+$]
    » $\bar{\delta}(q,w) := \delta(\ \bar{\delta}(q,x)\ ,\ a\ )$ [$|w| > 0$]
    » $L(\mathrm{DFA}) := \{w \in \Sigma^* \mid \bar{\delta}(q_0,w) \in F\}$

... Type 3

– example [(double) counting *modulo* 2]:

$L(\mathrm{DFA}) := \{w \in \Sigma^* \mid w$ has even # 0s and even # 1s$\}$

» q_0: # 0s ... , # 1s ... [start state]
» q_1: # 0s even , # 1s odd
» q_2: # 0s odd , # 1s even
» q_3: # 0s odd , # 1s odd

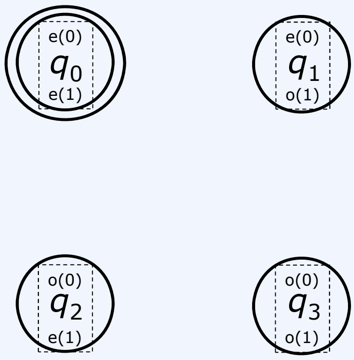
DFA: $(q_0, \{0,1\}, \delta, \{q_0,q_1,q_2,q_3\}, \{ \_ \})$

exercise: transition diagram/graph and table

… Type 3

– transition diagram/graph

final state(s) – ?



→ … © blackboard

… Type 3

– transition table

δ	**0**	**1**
→ $\boldsymbol{q_{0(F)}}$	q_2	q_1
$\boldsymbol{q_1}$	q_3	$q_{0(F)}$
$\boldsymbol{q_2}$	$q_{0(F)}$	q_3
$\boldsymbol{q_3}$	q_1	q_2

… Type 3

- illustration: w_0 := 1001, w_1 := 010; $w_i \in L(\text{DFA})$?
  - » $\bar{\delta}(q_0,1001)$ =

    $\delta(\ \bar{\delta}(q_0,100)\ ,\ 1\ )$ =

    $\delta(\delta(\ \bar{\delta}(q_0,10)\ ,\ 0)\ ,\ 1\ )$ =

    $\delta(\delta(\delta(\ \bar{\delta}(q_0,1)\ ,\ 0)\ ,\ 0)\ ,\ 1\ )$ =

    $\delta(\delta(\delta(\delta(\ \bar{\delta}(q_0,\varepsilon)\ ,\ 1)\ ,\ 0)\ ,\ 0)\ ,\ 1\ )$ =
    $\delta(\delta(\delta(\delta(\ \ q_0\ \ ,\ 1)\ ,\ 0)\ ,\ 0)\ ,\ 1\ )$ =

    $\delta(\delta(\delta(q_1,0),0),1) = \delta(\delta(q_3,0),1) = \delta(q_1,1) = q_0 \in F$

    $\rightarrow w_0 \in L(\text{DFA})$

… Type 3

» $\bar{\delta}(q_0,010)$ =

$\delta(\ \bar{\delta}(q_0,01)\ ,\ 0\ )$ =

$\delta(\delta(\ \bar{\delta}(q_0,0)\ ,\ 1)\ ,\ 0\ )$ =

$\delta(\delta(\delta(\ \bar{\delta}(q_0,\varepsilon)\ ,\ 0)\ ,\ 1)\ ,\ 0\ )$ =
$\delta(\delta(\delta(\ \ q_0\ \ ,\ 0)\ ,\ 1)\ ,\ 0\ )$ =

$\delta(\delta(q_2,1),0) = \delta(q_3,0) = q_1 \notin F$

$\rightarrow w_1 \notin L(\text{DFA})$

… Type 3

- NFA $(\Sigma, Q, F, q_0, \delta)$
  - Σ: [finite] set of [input] symbols
  - Q: [finite] set of states
  - F: [finite] set of final/accepting states [$\subseteq Q$]
  - q_0: start state [$\in Q$]
  - δ: transition function: $Q \times \Sigma \rightarrow 2^Q$ [$\delta(q,a) \subseteq Q$]

- $L(\text{NFA}) := \{w \in \Sigma^* \mid \text{NFA "accepts" } w\}$

… Type 3

- extended transition function
  $\bar{\delta}(q,w)$ [$\subseteq Q$]: $Q \times \Sigma^* \rightarrow 2^Q$
  - preliminaries
    - » $w := xa$ [$\in \Sigma^+$; prefix $x \in \Sigma^*$, 1-char. suffix $a \in \Sigma$]
    - » $\bar{\delta}(q,x) := \{p_1, p_2, \ldots, p_k\}$
    - » $\bigcup_{i:=1}^{k} \delta(p_i, a) := \{r_1, r_2, \ldots, r_m\}$
  - recursive construction
    - » $\bar{\delta}(q,\varepsilon) := \{q\}$ [$|\varepsilon| = 0$]
    - » $\bar{\delta}(q,w) := \{r_1, r_2, \ldots, r_m\}$ [$|w| > 0$]
    - » $L(\text{NFA}) := \{w \in \Sigma^* \mid \bar{\delta}(q_0,w) \cap F \neq \emptyset\}$

… Type 3

- $n := |Q_{\text{NFA}_{\text{min.}}}| \leq |Q_{\text{DFA}_{\text{min.}}}| \leq 2^n$
- $L(\text{DFA}) = L(\text{NFA})$
- TM which moves only to the right

… Type 3

- minimization of a DFA [min. # states]
- "equivalent": symmetric "pre-order"
  - reflexive
  - transitive
  - symmetric [info: also "anti-symmetric"]
- equivalent states $p \neq q$
  - $\forall\, w \in \Sigma^*: \quad \bar{\delta}(p,w) \in F \quad \Leftrightarrow \quad \bar{\delta}(q,w) \in F$
  - $p, q \quad =: \quad s$

… Type 3

- statements
  - final state ≠ non-final state
  - $\delta(p,a) =: r \neq s := \delta(q,a)$; $a \in \Sigma,\ p \neq q$
    $r \neq s \Rightarrow \exists\ w \in \Sigma^*\!:\ \bar{\delta}(r,w) \in F \oplus \bar{\delta}(s,w) \in F$
    $\Leftrightarrow\ \bar{\delta}(p,aw) \in F \oplus \bar{\delta}(q,aw) \in F$
  - $\bar{\delta}(q_0,x) = \bar{\delta}(q_0,y) \Rightarrow \bar{\delta}(q_0,xz) = \bar{\delta}(q_0,yz)$

… Type 3

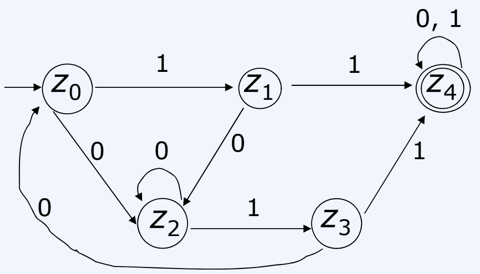
- algorithm: $\text{DFA}_o\ A \ \rightarrow\ \text{DFA}_m\ Z$ $(\exists!)$
  - delete unreachable states (from q_0)
  - find all pairs of equiv. states (*table-filling*)
  - partition the set Q of states in equivalence classes (of equivalent states); this minimally, i. e. pairs of states of different equivalence classes are not equivalent
  - each equiv. cl. corresponds to set of states
  - # equivalence classes (sets) =: “index”
  - each of these sets represents 1 state

… Type 3

- … alg.
  - $S_{[\subseteq Q]} := \{q^S{}_1, \ldots, q^S{}_{|S|}\}$ equiv. states in A
  - $T_{[\subseteq Q]} := \{q^T{}_1, \ldots, q^T{}_{|T|}\}$ equiv. states in A
  - $a \in \Sigma$
  - $\exists\ T \quad \forall\ q_i \in S \quad : \qquad \delta(q_i,a) \in T$
  - $\delta_{\mathrm{m}}(S,a) = T \qquad \in \qquad Q_Z \quad \ni \quad S$
  - $q_{0\_\mathrm{m}} := \{\ldots, q_0, \ldots\} \subseteq Q_A$
  - $q_{F\_\mathrm{m}} := \{\ldots, q_F, \ldots\} \subseteq F_A$
  - $k := |\Sigma|,\ n := |Q|;\ \mathrm{O}(k{\cdot}n^2) \supset \Theta(k{\cdot}n{\cdot}\mathrm{ld}(n))$

…

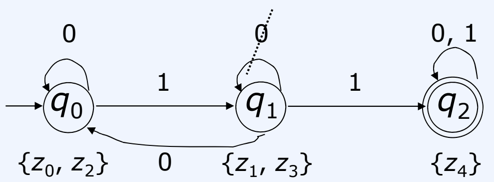
… Type 3



$\Sigma := \mathrm{B} := \{0, 1\}$

$|\{z_0, z_1, z_2, z_3, z_4\}| = 5$

$L(\mathrm{A}_z) := \{w \in \mathrm{B}^+ \mid w = p11s;\ p, s \in \mathrm{B}^*\}$

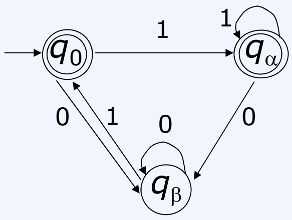


$|\{q_0, q_1, q_2\}| = 3 \leq 5$

$L(\mathrm{A}_\mathrm{m}) = L(\mathrm{A}_z)$

3 equivalence classes of states

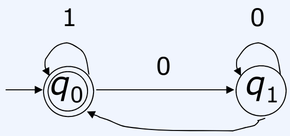
… Type 3



$\Sigma := \mathrm{B} := \{0, 1\}$

$|\{q_0, q_\alpha, q_\beta\}| = 3$

$L(\mathrm{A_g}) := \{\varepsilon\} \cup \{w \in \mathrm{B}^+ \mid w = p1,\ p \in \mathrm{B}^*\}$

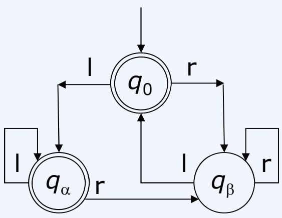


$|\{q_0, q_1\}| = 2 \leq 3$

$L(\mathrm{A_m}) = L(\mathrm{A_g})$

… Type 3

application: game design ("nest – food")



$\Sigma := \{r, l\}$

$|\{q_0, q_\alpha, q_\beta\}| = 3$

<u>exercise</u>: minimization …

$L(\mathrm{A_{nf}}) :=$

$|\{q_0, q_1\}| = 2 \leq 3$

$L(\mathrm{A_m}) = L(\mathrm{A_{nf}})$

3.2 Type 2

- *p*ush *d*own *a*utomaton (PDA)
  - recognizes context-free language(s)
  - NFA with auxiliary stack
  - *N*PDA: *n*on-deterministic PDA
    - *N*PDA $(\Sigma, \Gamma, Q, F, q_0, \delta)$ [+ bottom el. $\perp$ …]
      - Σ: [finite] set of input symbols
      - Γ: [finite] set of stack symbols [$\ni \perp$, in case]
      - Q: [finite] set of states
      - F: [finite] set of final/accepting states [$\subseteq Q$]
      - q_0: start state [$\in Q$]
      - δ: transition function: $Q \times \Sigma^* \times \Gamma^* \rightarrow 2^{(Q \times \Gamma^*)}$
    - $L(NPDA) := \{w \in \Sigma^* \mid NPDA$ "accepts" $w\}$

… Type 2

- $L(NPDA)_{\text{empty stack}} = L(NPDA)_{\text{final state(s)}}$
- *D*PDA: *d*eterministic PDA
  - $L(DPDA)_{\text{empty stack}} \subset L(DPDA)_{\text{final state(s)}}$
  - illustration
    - prefix property: $p, s \in \Sigma^+$; $w := ps \in L \Rightarrow p \notin L$ [-free]
    - absence of pref. prop.; ex.: expressions with "(…)"
    - problem: not knowing when p is just a proper prefix

… Type 2

– $L(DPDA) \subset L(NPDA)$

- ex.: ***r**eversal*: $L_r := \{w := w_b \cdot \boldsymbol{r}(w_b) \mid w_b \in \Sigma^*\}$
  [$\Rightarrow |w| = 2k,\ k \in \mathcal{N}$ (:= {0, 1, 2, 3, …}); reversal → palindrome ↛ r.]

- corresponding *grammar* $G_\rho := (S, N, \Sigma, P)$
  $S \in N := \{\ \ \ \}, \Sigma := \{0, 1\}$,
  $P := \{S \rightarrow \varepsilon \mid\ \ \ \}$
  […]

- source of *n*on-determinism: no prefix property

… Type 2

- ex.: *mirror*: $\Sigma_m := \Sigma \cup \{m_{\notin\Sigma}\}$

  $L_m := \{w := w_b \cdot m \cdot \boldsymbol{r}(w_b) \mid m \in \Sigma_m \setminus \Sigma,\ w_b \in \Sigma^*\}$
  [$\Rightarrow |w| = 2k+1,\ k \in \mathcal{N}$ (:= {0, 1, 2, 3, …}); mirror → ¬ rev., r. → ¬ m.]

  prefix-free $L(DPDA)$

  - no guessing concerning the centre
  - the middle ("*m*irror" ☺) is known
  - L_m not regular (¬ type-3)
  - prefix property

… Type 2

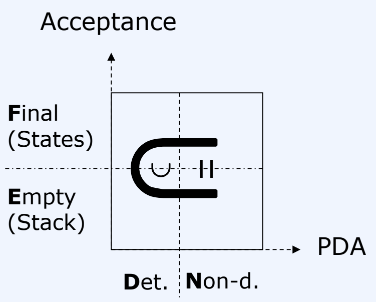
– $L(DPDA)_{\text{empty stack}} \quad \rightarrow \quad L$ prefix-free

– $L \neg$ prefix-free $\quad \rightarrow \quad L(DPDA)_{\neg \text{ empty stack}}$

– $L(DPDA)_{\text{empty stack}} \quad \leftrightarrow$

$$L(DPDA)_{\text{final state(s)}} \wedge L \text{ prefix-free}$$

… Type 2



3.3 Type 1

- ***l**inearly **b**ounded **a**utomaton (LBA)*
  - recognizes context-sensitive language(s)
  - *N*TM which does not leave its input area (bounded at 2 sides)
  - $L(N\text{LBA}) := \{w \in \Sigma^* \mid N\text{LBA "accepts" } w\}$
    ex.: $\{1^n \mid n = 2^l, l \geq 0\}$ recognizable by *N*LBA
  - $L(N\text{LBA}) = L(D\text{LBA})$?

3.4 Type 0

- *Turing Machine* (TM)
  - recognizes recursively enumerable lang.
  - *D*TM

    somewhere else b =: "#" as the separation symbol

    - DTM $(\Sigma, b, M, Q, F, q_0, \delta)$
      - Σ: finite set of input symbols [$\not\ni$ "*b*lank"; $\Sigma \cup \{b\} =: \Sigma_b$]
      - $M := \{$*l*eft, *r*ight, *n*owhere$\}$ set of "*m*ove" actions
      - Q: finite set of states
      - F: finite set of *f*inal/accepting states [$\subseteq Q$]
      - q_0: start state [$\in Q$]
      - δ: transition function: $Q \times \Sigma_b \to Q \times \Sigma_b \times M$
    - $L(\text{DTM}) := \{w \in \Sigma^* \mid \text{DTM "accepts" } w\}$

… Type 0

– *NTM*
  - Σ, b, M, Q, F, q_0
  - δ: $Q \times \Sigma_b \rightarrow 2^{(Q \times \Sigma_b \times M)}$
– L(NTM) = L(DTM)

- Church-Turing (hypo)thesis [=: *CT*-H]

  L(TM) $\cong$ {(semi-/) partially computable functions}

4 (Un-)Computability

4.1 Introduction

- undecidable problems
  - transitions in cellular automata
    - given
      - configurations c_0, c_1
    - decide
      - $c_0 \xrightarrow{+} c_1$ [?]
    - uncomputable …

… Introduction

  - equivalence of "type-2" grammars
    - given
      - G_1, G_2 context-free ["type 2"]
    - decide
      - $L(G_1) = L(G_2)$ [?]
    - uncomputable …

… Introduction

– *PCP* (*P*ost's *C*orrespondence *P*roblem)

- given
  - $|\Sigma| > 1$
  - 2 lists X, Y – each consisting of $k_{(>1)}$ parts
  - $X := x_1 \text{ ' } x_2 \text{ ' } x_3 \text{ '…' } x_k$; $Y := y_1 \text{ ' } y_2 \text{ ' } y_3 \text{ '…' } y_k$ (x_i, $y_i \in \Sigma^+$)
- decide
  - ∃ sequence [not a set] $i_1, …, i_{m(>1)} \in \{1, …, k\} =: I$ s. t. [*i*ndex set]

$$x_{i_1}…x_{i_m} = y_{i_1}…y_{i_m} \quad (\Rightarrow |x_{i_1}…x_{i_m}| = |y_{i_1}…y_{i_m}|) \quad [?]$$

- uncomputable …

… Introduction

- undecidable [in general]
- decidable instances [$\Sigma := \{0, 1\}$]
  - solvable [⇒ decidable, computable]
    - » $k := 3$; $X := 0 \text{ ' } 01 \text{ ' } 01000$; $Y := 000 \text{ ' } 1 \text{ ' } 01$
      solution: $m := 4$; $i_1 = 3$, $i_2 = 1 = i_3$, $i_4 = 2$
      $x_3 \ x_1 x_1 x_2 = y_3 \ y_1 \ y_1 \ y_2$
      01000'0'0'01 = 01'000'000'1
      010000001 = 010000001
  - unsolvable [⇒ decidable, computable]
    - » $k := 2$; $X := 0 \text{ ' } 1$; $Y := 1 \text{ ' } 0$; $x_{i_1}…x_{i_m} \neq y_{i_1}…y_{i_m}$
    - » $k := 2$; $X := 0 \text{ ' } 1$; $Y := 00 \text{ ' } 11$; $|x_{i_1}…x_{i_m}| \neq |y_{i_1}…y_{i_m}|$ ⇐
  - additionally given: w [$\in \Sigma^+$]
    - » decide: ∃ $x_{i_1}…x_{i_m} = y_{i_1}…y_{i_m} = w$
    - » computable …

... Introduction

– *d*isjunctiveness of “type-2” grammars

[=: *DCF*]

- given
  - G_1, G_2 context-*f*ree [“type 2”]
- decide
  - $L(G_1) \cap L(G_2) \; = \; \varnothing := \{ \}$ [?]
- uncomputable ... – proof:

(http://www.tcs.ifi.lmu.de/lehre/ss-2014/timi/handouts/handout-09)

... Introduction

– reduction of *PCP* to *DCF*

- given
  - X, Y
  - G_1, G_2
  - $I \cap \Sigma$ [:]= $\varnothing$
- define
  - $\Sigma' := \Sigma \cup I$
  - $G_1 := (S_1,\{S_1\},\Sigma',P_1)$, $P_1 := \{S_1 \rightarrow x_{\boldsymbol{i}}S_1\boldsymbol{i} \mid x_{\boldsymbol{i}}\boldsymbol{i}\}$
  - $G_2 := (S_2,\{S_2\},\Sigma',P_2)$, $P_2 := \{S_2 \rightarrow y_{\boldsymbol{i}}S_2\boldsymbol{i} \mid y_{\boldsymbol{i}}\boldsymbol{i}\}$

... Introduction

- $x_{i_1}...x_{i_j}i_j...i_1$ =: w_1 ; $y_{i_1}...y_{i_k}i_k...i_1$ =: w_2
- $x_{i_1}...x_{i_j}$ =: prefix(w_1) =: $p(w_1) \in \Sigma^+$
- $y_{i_1}...y_{i_k}$ =: $p(w_2) \in \Sigma^+$
- $i_j...i_1$ =: suffix(w_1) =: $s(w_1) \in I^+$
- $i_k...i_1$ =: $s(w_2) \in I^+$
- $w_1 := p(w_1)s(w_1) \in \Sigma'^+ \ni p(w_2)s(w_2) =: w_2$
- PCP : $(w_1 =? w_2) \Leftrightarrow_{[\Sigma^+ \cap I^+ = \varnothing]}$
  $([p(w_1) =? p(w_2)] \wedge_? [s(w_1) =? s(w_2)])$
  $\Leftrightarrow$ [$s(w_1) = s(w_2)$ solves PCP]
- $w_1 = w_2 =: w \in L(G_1) \cap L(G_2) \neq \varnothing \Leftrightarrow \exists$ PCP solution
- PCP unsolvable $\Leftrightarrow L(G_1) \cap L(G_2) = \varnothing$ [DCF]
- PCP uncomputable $\overset{\leq}{\underset{\text{red.}}{\Rightarrow}}$ DCF uncomputable

4.2 Diagonalization

– general halting problem [Alan M. Turing (1936)]
  - given
    – computer program [algorithm]
    – arbitrary input
  - decide
    – program halts [?]
  - uncomputable

– Georg Ferdinand Ludwig Philipp Cantor (1845 – 1918)

… Diagonalization

“diagonalization” argument, via *halting problem* table: algorithm A_i halts (“*yes*”) or not (“*no*”) on input j

	j: 0	1	2	3 …
A_0	y	n	n	y
A_1	y	n	y	y
A_2	n	y	n	y
A_3	y	y	n	y …
⋮				

abbrev.: $\neg n$:= *yes*: termination
$\neg y$:= *no* termination

$A := (a) :=$

$(\; a_0 \;, \; a_1 \;, \; a_2 \;, \; a_3, \ldots) :=$
$(\neg[A_0,0], \neg[A_1,1], \neg[A_2,2], \neg[A_3,3], \ldots)$
$(\; n \;, \; y \;, \; y \;, \; n \;, \ldots)$

$\exists i \in \mathcal{N} \;:\; A = A_i$?

no: $\forall i \in \mathcal{N} \;:\; A \neq A_i \,,\, \exists j_{=i}\!: a_j \neq [A_i,j]$

$\Rightarrow \forall j \in \mathcal{N}: \exists$ 2 alg. $A, A_k \;:\; a_j \neq [A_{k=j},j]$

$\Rightarrow n \in \{a_j, [A_j,j]\}$ (evtl. *no* termination)

$\Rightarrow$ $WP_{type\text{-}0}$ undecidable (uncomputable)

4.3 The Whole Lot

4.3.1 Basics

4.3.2 Computability/Decidability

4.3.3 Un-Computability/Un-Decidability

4.3.4 Conclusion

DHBW Duale Hochschule Baden-Württemberg Mannheim

4.3.1 Basics

4.3.1.1 Functions

4.3.1.2 Set Theory

DHBW Duale Hochschule Baden-Württemberg Mannheim

4.3.1.1 Functions

- (total) *function* f: $D \rightarrow C$
  assigns to every object x in the *domain* D exactly one object $f(x)$ in the *co-domain* C
  $[\forall\ x \in D\ \ \exists!\ f(x) \in C]$
  exactly 1
- bijective function: 1–1 correspondence
- partial *function*:
  not necessarily all $x_{\in D}$ have $f(x)_{\in C}$

4.3.1.2 Set Theory

4.3.1.2.1 Notions

4.3.1.2.2 Order of Infinity

4.3.1.2.1 Notions

- *set* S: collection of unique elements
- cardinality, cardinal number $|S|$
  - S finite : # elements
  - S infinite : order of infinity
- *power set*: $\wp(S) =: 2^S := \{s \mid s \subseteq S\}$

  cardinality: $|\wp(S)| = |2^S| \underset{\text{finite}}{=} 2^{|S|} \underset{\dots}{>} |S|$

4.3.1.2.2 Order of Infinity

- $|A| = |B|$ <u>iff</u> $\exists$ bijection $f: A \to B$
- *denumerable* (countably infinite) *set* S_d
  $|S_d| = |\mathcal{N}| =: \aleph_0$ $(= \infty)$ [bijection with $\mathcal{N}$]
- every infinite set contains a countably infinite subset
- A infinite <u>iff</u> $\exists\, S \subset A$ with $|S| = |A|$
- youtube.com/watch?v=zeCCMOHVS3w

… Order of Infinity

- ordinals (<u>ordinal</u> number<u>s</u>)
  $0 := \{\ \}$ $[=: \varnothing]$ (0^{th}) ordinal $[=: \omega_0]$
  $\alpha+1 =: \alpha^+ := \alpha\cup\{\alpha\}$; α ordinal, α^+ $s(\alpha)$
  $1 := 0+1 = \varnothing\cup\{\varnothing\} = \{\varnothing\} = \{0\}$
  $2 := 1+1 = \{\varnothing\}\cup\{\{\varnothing\}\} = \{\varnothing, \{\varnothing\}\}$
  $\vdots$ $= \{0, 1\}$
  $\omega := \{0, 1, 2, \ldots\}$ $\mathcal{N}$ 1^{st} infinite ordinal
  $=: \omega_1$ $(\alpha_1 < \alpha_2$ <u>iff</u> $\alpha_1 \in \alpha_2$ <u>iff</u> $\alpha_1 \subset \alpha_2)$
- $\alpha \cong |\alpha|$

… Order of Infinity

β *limit ordinal* [*lol* (☺)]: $\nexists\alpha$ with $\beta = s(\alpha)$

[no immediate predecessor]

ex.: ω_0, ω_1

$\omega_1+1 := \omega_1\cup\{\omega_1\} = \{\omega_1, \{\omega_1\}\}$

$\vdots$

$\omega_1+k = \{0, 1, 2, \ldots, \omega_1, \omega_1+1, \ldots, \omega_1+k-1\}$

ω_2 [2nd infinite *lol*] $= \omega_1 \cup \{\omega_1+n \mid n\in\omega\}$

$\vdots$

ω_i [ith *lol*] $= \omega_{i-1} \cup \{\omega_{i-1}+n \mid n\in\omega\}$

basis for *transfinite induction* …

… Order of Infinity

- $\alpha_1+\alpha_2$: α_1 <u>followed</u> by [disjoint] α_2 in order
  - addition of infinite ordinal
    [union with infinite set]
    not commutative
  - ex.: $1 + \omega$: $1 < \omega \Rightarrow 1 \subset \omega \Rightarrow 1 \cup \omega = \omega$
    $\omega + 1$: $1 < \omega < \omega + 1 = \omega\cup\{\omega\} \neq \omega$
- $|S_{\text{infinite}}| \overset{\text{(generalized)}}{\underset{\text{continuum hyp.}}{<}} |\wp(S)| = |\{s \mid s \subseteq S\}| > \omega$

  $\nexists$ surj.: $N \to 2^N$ "uncountable"
- $\to$ <u>uncomputability</u>

4.3.2 Computability/Decidability

- algorithms
  - terminate
  - $\text{decidable}_{\text{problems}}$
- L decidable <u>iff</u>
  L semi-decidable $\wedge$ L-C semi-decidable
- Chomsky hierarchy
  - $L_3 \subset L_2 \subset L_1 \subset L_{\text{decidable}} \subset \ldots$
  - proper subsets [of language classes]

4.3.3 Un-Computability/Un-Decidability

- … Chomsky hierarchy
  - computable | uncomputable !
  - $L_{\text{decidable}} \subset L_0 \; [\subset L_{\text{gen.}} := \{L \mid L \subseteq \Sigma^*\}]$
  - L_0 semi-decidable, L_0-C $\neg$semi-decidable
- $|\Sigma^*| =_{\Sigma \neq \emptyset} |\mathcal{N}| < 2^{|\mathcal{N}|} = 2^{|\Sigma^*|} \cong |2^{\Sigma^*}|$
  $\nexists$ surject.: $\Sigma^* \to 2^{\Sigma^*}$, $\mathcal{N} \to 2^{\mathcal{N}}$ $\exists$ incomparable elements
- $\exists$ only countably infinite many algorithms
- $\exists$ uncountably many languages/probl.
- $\exists$ uncomputable word problems

… Un-Computability/Un-Decidability

- Church-Turing (hypo)thesis
  - $L_0 \cong$ {semi-/partially computable funct.}
  - definition gaps allowed; uncomputable

- 3 proof ideas
  - Cantor's diagonalization
  - uncountable power-set cardinality
  - polynomial reducibility/reduction of probl. via alg.
    - $p_1 \leq_p p_2$ iff $a_2 \cong f_r^p(a_1)$ $\boxed{w \in L_1 \Leftrightarrow f_r(w) \in L_2}$
    - p_1 undecidable $\Rightarrow$ p_2 undecidable

4.3.4 Conclusion

- $L_{\text{decidable}} \subset L_0$
- countable $_{\text{\# alg.}}$ < uncountable $_{\text{\# probl.}}$
- ∃ uncomputable decision problem[s]
- $\{ \} \neq L_0 \setminus L_{\text{decidable}}$ really undecidable