Theorie-III

Formale Sprachen und Automatentheorie 1+2

Selbst-Studium

Walter Hower

… Recurrence Relation

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

$e_{\dots}$:= … , e_n := e_{n-1} …

backward substitution: …

forward substitution: …

statement: e_n = …

proof: induction on n …

… Recurrence Relation

2) d_n := diameter in a grid (q.; n := # nodes, … := …)

$d_{\dots}$:= … , $d_{\dots}$:= $d_{\dots -1}$ …

backward substitution: …

forward substitution: …

statement: $d_{\dots}$ = …

proof: induction on …

3) c_h := # connections in an h-dimensional hyper-cube

… Complexity

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

[*parallel* time compl.: polyn.] $[NP_{\mathrm{TIME}} \overset{?}{\setminus} (D)P_{\mathrm{TIME}}]$

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 …

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

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

$[NP_{\mathrm{TIME}} \setminus (D)P_{\mathrm{TIME}} \neq \emptyset$?]

… Complexity

– illustration

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

… Languages

- "type 2" grammar

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

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

variables | alphabet := {symbols} | *P*roductions

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

ex. (G): S, $N := \{S\}$, $\Sigma := \{a, b, +, -, (,)\}$

$P := \{S \rightarrow a, S \rightarrow b, S \rightarrow S+S, S \rightarrow S-S, S \rightarrow (S)\}$

$L(G)$ = … not "regular", but c.-f.

… Languages

- "type 3" (grammars)

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

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

right-linear …

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

regular

exercise: L_x = all strings over $\Sigma := \{0,1\}$ with exactly one "0": 1*01*; …
$L_y = \{a^k b^l \mid k, l \geq 0\}$ …

… Type 3

– illustration

- $L(\mathrm{DFA}) = \{p01s \mid p, s \in \Sigma^*\}$
  - 01, 010, 1001, 00011 $\in L(\mathrm{DFA})$
  - ε, 0, 1, 10, 11, 100 $\notin L(\mathrm{DFA})$
  - construction $(q_0, \Sigma, \delta, Q, F)$
    - » q_0 := start state
    - » $\Sigma := \{0, 1\}$
    - » 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

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

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

» q_0: # 0_s … , # 1_s … (start state)
» q_1: # 0_s even , # 1_s odd
» q_2: # 0_s odd , # 1_s even
» q_3: # 0_s odd , # 1_s odd

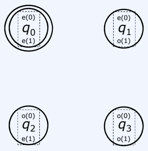
DFA: $(q_0, \{0,1\}, \delta, \{q_0,q_1,q_2,q_3\}, \{\ \dots\ \})$

exercise: transition diagram/graph and table

... Type 3

– transition diagram/graph

final state(s) – ?



→ ... ☺ blackboard

... Type 3

– illustration: w_0 := 1001, w_1 := 010; $w_i \in L$(DFA)?

» $\bar{\delta}(q_0,1001)$ =

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

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

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

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

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

$\rightarrow w_0 \in L$(DFA)

... Type 3

» $\bar{\delta}(q_0,010)$ =

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

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

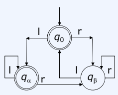
$\delta(\delta(\delta(\ \bar{\delta}(q_0,\varepsilon)\ ,\ 0)\ ,\ 1)\ ,\ 0\ )$ =
$\delta(\delta(\delta(\quad q_0 \quad ,\ 0)\ ,\ 1)\ ,\ 0\ )$ =

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

$\rightarrow w_1 \notin L$(DFA)

... Type 3

application: game design (“nest – food”)



Σ := {r, l}

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

<u>exercise</u>: minimization

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

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

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

... Type 2

– $L(D\mathrm{PDA}) \subset L(N\mathrm{PDA})$

- ex.: ***r***eversal: $L_r := \{w := w_p \cdot \boldsymbol{r}(w_p) \mid w_p \in \Sigma^*\}$
  ($\Rightarrow |w| = 2k,\ k \in \mathcal{N}$ (:= {0, 1, 2, 3, ...}); reversal → palindrome ↛ r.)
- corresponding *g*rammar $G_\rho := (S, N, \Sigma, P)$
  $S \in N := \{\ \ldots\ \}$, $\Sigma := \{0, 1\}$,
  $P := \{S \rightarrow \varepsilon \mid \ldots \qquad \}$
  [...]
- source of *n*on-determinism: no prefix property

... Type 2

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

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

  prefix-free $L(D\mathrm{PDA})$

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

... Introduction

– reduction of *PCP* to *DCF*

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

4.3.1.2.1 Notions

- *s*et S: collection of unique elements
- cardinality, cardinal number $|S|$
  – S finite : # elements
  – S infinite : order of infinity
- *p*ower *s*et: $\wp(S) =: 2^S := \{s \mid s \subseteq S\}$
  cardinality: $|\wp(S)| = |2^S| \underset{\text{finite}}{=} 2^{|S|} > |S|$

4.3.1.2.2 Order of Infinity

- $|A| = |B|$ <u>iff</u> $\exists$ bijection f: $A \rightarrow B$
- *d*enumerable (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