Teoretyczne podstawy informatyki

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. letni 2025/26

inf.ug.edu.pl/~amb/TPI

Twierdzenie Myhilla-Nerode'a

Relacje równoważności na słowach

- dana jest relacja równoważności $R \subseteq A^* \times A^*$
  wyznacza ona podział A^* na klasy równoważności (rozłączne, niepuste)
  $$A = \bigcup_{a \in A} \{b \in A \mid b \; R \; a\}$$
  ozn.: $[a]_R = \{b \in A \mid b \; R \; a\}$, $A^*_R = \{[a]_R \mid a \in A\}$
  def.: indeks relacji to liczba klas równoważności, $|A^*_R|$
- dla języka $L \subseteq A^*$ definiujemy R_L jako
  $$x \; R_L \; y \iff (\forall \, z \in A^*. \, xz \in L \iff yz \in L)$$
- dla automatu deterministycznego $\mathcal{A}$ nad A^* definiujemy $R_{\mathcal{A}}$ jako
  $$x \; R_{\mathcal{A}} \; y \iff (\delta(q_0, x) = \delta(q_0, y))$$

Relacje prawostronnie niezmiennicze

- powyższe relacje są prawostronnie niezmiennicze:
  $x \; R_L \; y \Rightarrow \forall \, z \in A^*. \, xz \; R_L \; yz$
  $x \; R_{\mathcal{A}} \; y \Rightarrow \forall \, z \in A^*. \, xz \; R_{\mathcal{A}} \; yz \; x \; R \; y \Rightarrow x \; R_L \; y$
- przykłady relacji prawostronnie niezmienniczych:
  - oba słowa zaczynają się tą samą literą
  - oba słowa kończą się tą samą literą
  - słowa mają równą długość
- przykłady relacji które nie są prawostronnie niezmiennicze:
  - oba słowa są albo nie są poprawnym słowem języka polskiego
    np. słowa „kot" i „pies" są oba poprawne więc są równoważne
    ale „kotem" jest poprawnym słowem a „piesem" nie jest
    nie jest więc spełniony warunek $\forall \, z \in A^*. \, xz \; R \; yz$

Twierdzenie Myhilla-Nerode'a

- tw.: następujące warunki są równoważne
  1. język $L \subseteq A^*$ jest regularny
  2. relacja R_L ma skończony indeks
  3. L jest sumą pewnej liczby klas abstrakcji pewnej relacji równoważności na A^*, prawostronnie niezmienniczej, skończonego indeksu

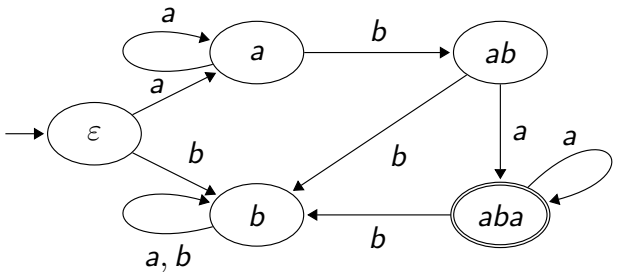
Twierdzenie Myhilla-Nerode'a, dowód

- $1 \Rightarrow 3$: L jest akceptowane przez automat deterministyczny $\mathcal{A}$, relacja $R_{\mathcal{A}}$ jest prawostronnie niezmiennicza,
  słowa, które trafiają do tego samego stanu, są równoważne czyli klas równoważności jest nie więcej niż stanów
  L jest sumą tych klas, które opisują przejście do stanów akceptujących
- $3 \Rightarrow 2$: L jest sumą pewnych klas relacji R prawostronnie niezmienniczej, a więc jeśli $x\ R\ y$, to $xz\ R\ yz$ dla dowolnego z,
  stąd $xz \in L \iff yz \in L$ czyli $x\ R_L\ y$; relacja R_L najwyżej skleja klasy relacji R czyli jest ich nie więcej
- $2 \Rightarrow 1$: definiujemy automat, Q klasy abstrakcji R_L, q_0 klasa ε, $F = \{[w] \mid w \in L\}$, funkcja przejścia (deterministyczna, całkowita) $\delta([u], a) = [ua]$
  definicja δ jest poprawna, jeśli $[u] = [v]$ to $[ua] = [va]$
  ten automat akceptuje L

Twierdzenie Myhilla-Nerode'a, przykład

- dany język $\{a^i b a^j \mid i, j \geq 1\} \subseteq \{a, b\}^*$
- słowa ε, a, b, ab, aba są wszystkie nierównoważne ze sobą
  np. $aa \notin L$ ale $aba \in L$ czyli $\neg(a\ R_L\ ab)$
  lub $ab \notin L$ ale $aba \in L$ czyli $\neg(ab\ R_L\ aba)$
- nie ma innych klas równoważności
  np. $a^i b\ R_L\ ab$, $i > 0$, ponieważ $a^i bw \in L \Leftrightarrow abw \in L$

q	a	b
ε	a	b
a	a	ab
ab	aba	b
aba	aba	b
b	b	b



- stanem początkowym jest ε, jedynym końcowym aba

Twierdzenie Myhilla-Nerode'a, wnioski

- tw.: dany język $L \subseteq A^*$, automat zbudowany z klas abstrakcji relacji R_L jest minimalnym automatem deterministycznym akceptującym L
- dw.: dla każdego innego automatu Q akceptującego L definiujemy funkcję $q \mapsto [\{w \in A^* \mid \delta(q_0, w) = q\}]$
  jest ona dobrze określona, tzn. jeśli różne słowa prowadzą do tego samego stanu, są one równoważne
  każda z klas abstrakcji automatu Myhilla-Nerode'a będzie w obrazie tej funkcji
  być może różne stany Q prowadzą do tego samego stanu automatu Myhilla-Nerode'a
  czyli jest on minimalnym

Zastosowania wyrażeń regularnych

Wyrażenia regularne w językach programowania

- oprócz gwiazdki $*$ – dowolnego powtórzenia, są jeszcze wersje
  $+$ – powtórzenie co najmniej jeden raz
  ? – element opcjonalny, wystąpienie 0 lub 1 raz
  $\{ile\}$ – dokładna liczba powtórzeń
  $\{min,\}$, $\{min, max\}$, $\{, max\}$ – zakres liczby powtórzeń
- oznaczenia na zbiory symboli
  . – (kropka, czasami gwiazdka), oznacza dowolny symbol z alfabetu
  [...] – wylicza możliwy zbiór symboli
  [... – ...] – wylicza zakres ASCII
  [^ ...] – uzupełnienie zbioru symboli
  są oznaczenia na tylko litery, tylko cyfry, litery lub cyfry, znaki białe
- konieczny jest znak ucieczki \ by móc opisać \., \[, itd.
- kotwice: poszukiwanie wzorca na początku ^ lub końcu $ wiersza, na granicy wyrazu, poszukiwania całego wyrazu

Przykłady

- szukaj wierszy, które sa w całości komentarzem, tzn. zaczynają się od spacji ale potem występuje #

```
grep -E '^[[:space:]]*#' test.ini
```

- wypisz wiersze, w których znaleziono *key* = *value*

```
grep -E '^[[:space:]]*[A-Za-z0-9_.-]+[[:space:]]*=[[:space:]]*[^[:space
    :]].*$' test.ini
```

- wypisz wszystkie adresy emailowe z plików tekstowych

```
grep -Eo '[[:alnum:]._%+-]+@[[:alnum:].-]+\.[[:alpha:]]{2,}' *.txt
```

  nie jest to precyzyjne, np. podwójne kropki są niedopuszczalne, nazwa dziedziny po @ nie może zaczynać się od myślnika, itd., z kolei formalna specyfikacja e-maili (RFC 5322) dopuszcza więcej możliwości
- znajdź pliki, których nazwa zawiera cyfry i kończą się na .log

```
find . -regextype posix-extended -regex '.*/[^/]*[0-9][^/]*\.log$'
```


Wyrażenia regularne w językach programowania, c.d.

- zastosowania wyrażeń regularnych nie ograniczają się do sprawdzenia czy słowo pasuje do wzorca
- ale również rozpoznania części składowych słowa, nazwania tych części i ich ponownego użycia
- nawiasy () służą do zaznaczenia grupy, mogą być zagnieżdżone, numerowane są od lewej
- rozróżniane mogą być dopasowania zachłanne (domyślnie) i leniwe, np. słowo *abbc* pasuje do wzorca $((a|b)^*)((b|c)^*)$ z \$1 = *abb* i \$3 = *c* ale wzorzec $((a|b)^*?)((b|c)^*)$ powoduje \$1 = *a* i \$3 = *bbc*
  (\$2 odpowiada za drugi nawias otwierający, niepotrzebny nam tutaj \$2 = *b* w dopasowaniu zachłannym, \$2 = *a* w dopasowaniu leniwym)
- zawsze trzeba przestudiować dokumentację, różne języki mają pewne możliwości a pewnych nie mają

Przykłady c.d.

- zamiana formatu daty MM/DD/YYYY → DD.MM.YYYY

```
sed -E 's/([0-9]{2})\/([0-9]{2})\/([0-9]{4})/\2.\1.\3/g' *.txt
```

- ujednolicenie numeracji plików (np. test-1.txt → test-001.txt)

```
rename -E 's/test-(\d+)\.txt/sprintf("test-%03d.txt",$1)/e' test-*.txt
```