8. Konečný automat

BAB37ZPR – Základy programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Konečný automat

Konečný automat = výpočetní model, primitivní počítač

- Řídící jednotka s konečným počtem stavů
- Vstupní posloupnost — textový řetězec, fyzické vstupy
- Výstupní posloupnost, fyzické výstupy

Definice:

- konečná množina stavů Q
- počáteční stav $q_0 \in Q$
- konečná množina vstupních symbolů A
- přechodová funkce $f : Q \times A \to Q$

$$q_{t+1} = f(q_t, a_t)$$

t je čas nebo index do vstupní posloupnosti

- (někdy) množina koncových stavů $F \subseteq Q$
- (někdy) množina akcí G a výstupní funkce $g : Y \to G$ nebo $g : Y \times A \to G$

Konečný automat

Konečný automat = výpočetní model, primitivní počítač

- Řídící jednotka s konečným počtem stavů
- Vstupní posloupnost — textový řetězec, fyzické vstupy
- Výstupní posloupnost, fyzické výstupy

Definice:

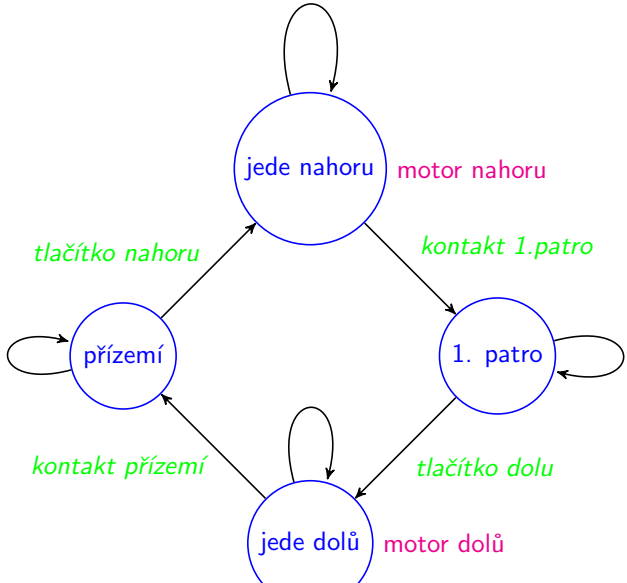
- konečná množina stavů Q
- počáteční stav $q_0 \in Q$
- konečná množina vstupních symbolů A
- přechodová funkce $f : Q \times A \to Q$

$$q_{t+1} = f(q_t, a_t)$$

t je čas nebo index do vstupní posloupnosti

- (někdy) množina koncových stavů $F \subseteq Q$
- (někdy) množina akcí G a výstupní funkce $g : Y \to G$ nebo $g : Y \times A \to G$

Příklad: řízení výtahu – stavový diagram



Příklad: řízení výtahu – přechodová a výstupní tabulka

vstupy				stavy		výstupy	
Tl.d.	Tl.n.	Příz.	1.p.	q_t	q_{t+1}	↑	↓
?	0	?	?	přízemí	přízemí	0	0
?	1	?	?	přízemí	jede nahoru	1	0
?	?	?	0	jede nahoru	jede nahoru	1	0
?	?	?	1	jede nahoru	1. patro	0	0
0	?	?	?	1. patro	1. patro	0	0
1	?	?	?	1. patro	jede dolů	0	1
?	?	0	?	jede dolů	jede dolů	0	1
?	?	1	?	jede dolů	přízemí	0	0

Automat přijímající jazyk

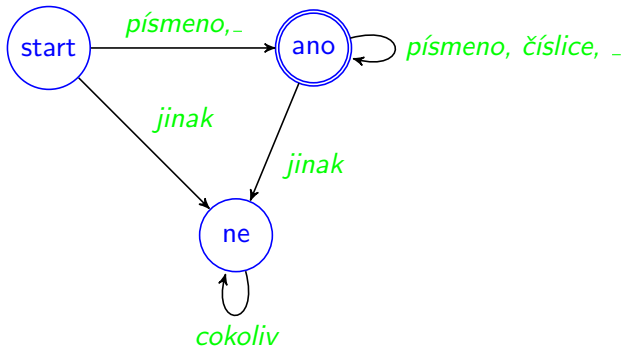
Jazyk = (i nekonečná) množina řetězců.

Rozhodovací konečný automat

- Vstupem je řetězec.
- Dává odpověď ano/ne, zda vstup patří do jazyka.
- V každém kroku čte automat jeden znak z řetězce.
- Vstupní abeceda A jsou všechny přípustné znaky.
- Automat přijímá řetězec $\Leftrightarrow$ na konci řetězce je automat v *přijímajícím (koncovém) stavu.*

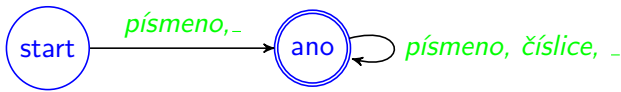
Příklad: jméno proměnné – totální automat

Automat přijímá platná jména proměnných v Pythonu.



- Automat je *totální* — přechodová funkce f je definována pro všechna $Q \times A$.

Příklad: jméno proměnné – částečný automat



- Částečný automat – nedefinovaný přechod znamená nepřijetí.

Jméno proměnné — implementace

```
1  def is_variable_name(s):
2    """ je 's' platne jmeno promenne v Pythonu? """
3    state="start"
4    for c in s:
5      if state=="start":
6         if c=="_" or is_letter(c):
7           state="ano"
8         else:
9           return False
10     else: # stav "ano"
11       if not (c=="_" or is_letter(c) or c.isdigit()):
12         return False
13   return state=="ano"
15 def is_letter(c):
16   return c.isupper() or c.islower()
```

Jméno proměnné – příklady

```
print(is_variable_name("moje_promenna12"))
```

```
ano
```

```
print(is_variable_name("moje{}_promenna"))
```

```
False
```

```
print(is_variable_name("12moje{}_promenna"))
```

```
False
```

Transformace textu

Transformační konečný automat

- Vstupem je řetězec.
- Výstupem je řetězec.
- V každém kroku čte automat jeden znak z řetězce.
- Vstupní abeceda A jsou všechny přípustné znaky.
- Součásti každého přechodu (nebo stavu) může být výstup jednoho či více znaků.

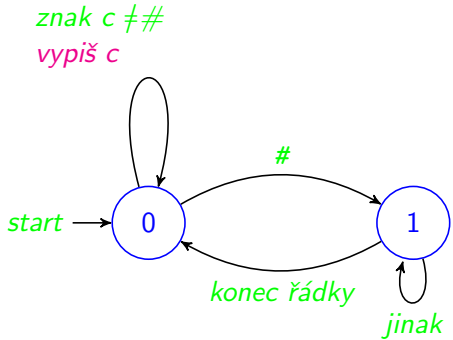
Příklad: přeskakování komentářů

Úkol:

- Vstupem je textový soubor
- Komentář je vše od znaku # (včetně) do konce řádky
- Vypište obsah souboru bez komentářů.

`preskoc_komentare.py`

Přeskakování komentářů (1)



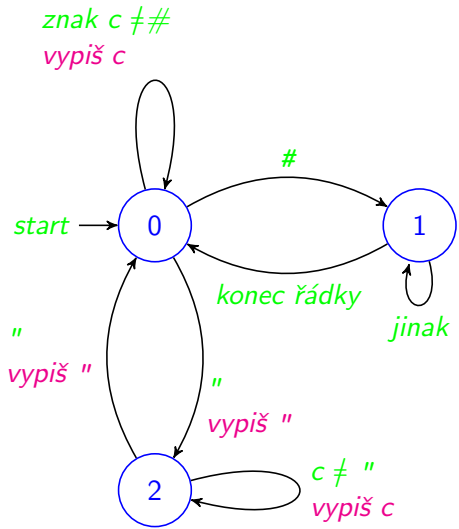
```
import sys

def preskoc_komentare(f):
  stav=0                # počáteční stav automatu
  while True:
    c=f.read(1)         # přečti jeden znak
    if c=="": return    # konec souboru
    if stav==0:         # počáteční stav
      if c=="#": stav=1 # začátek komentáře
      else:
        print(c,end="") # vytiskni znak
    else:               # stav 1="komentar"
      if c=="\n":       # konec řádky
        stav=0          # konec komentáře
        print(c,end="")
      else:             # vše ostatní ignorujeme
        pass
if __name__=="__main__":
  with open(sys.argv[1],'rt') as f: # otevři textový soubor
    preskoc_komentare(f)  # pokud se povedlo, přeskakuj
```

```
$ python3 preskoc_komentare.py preskoc_komentare.py

import sys

def preskoc_komentare(f):

  stav=0
  while True:
    c=f.read(1)
    if c=="": return
    if stav==0:
      if c=="
      else:
        print(c,end="")
...
```

Kde je problém?

Přeskakování komentářů (2)

Verze, kterou nezmate # v řetězci.



```
1  import sys
3  def preskoc_komentare(f):
4    stav=0                    # počáteční stav automatu
5    while True:
6      c=f.read(1)             # přečti jeden znak
7      if c=="": return        # konec souboru
8      if stav==0:             # počáteční stav
9        if c=="#": stav=1     # začátek komentáře
10       else:
11         if c=='"': stav=2   # začátek řetězce
12         print(c,end="")     # vytiskni znak
13     elif stav==1:           # 1="komentar"
14       if c=="\n": stav=0    # konec řádky, konec komentáře
15         print(c,end="")
16       else: pass            # vše ostatní ignorujeme
17     else:                   # stav = řetězec
18       if c=='"': stav=0
19       print(c,end="")       # vytiskni znak
```

Lexikální analýza (parsing)

- Vstupem je řetězec (posloupnost znaků)
- Výstupem je posloupnost symbolů *(tokens)* — číslo, operátor, identifikátor, klíčové slovo, ...
  - Symboly mohou mít atributy — řetězec, hodnota, ...
  - Reprezentujeme jako dvojice — typ symbolu + atribut
- Další operace
  - Vynechání komentářů
  - Odstranění mezer
  - Chybová hlášení

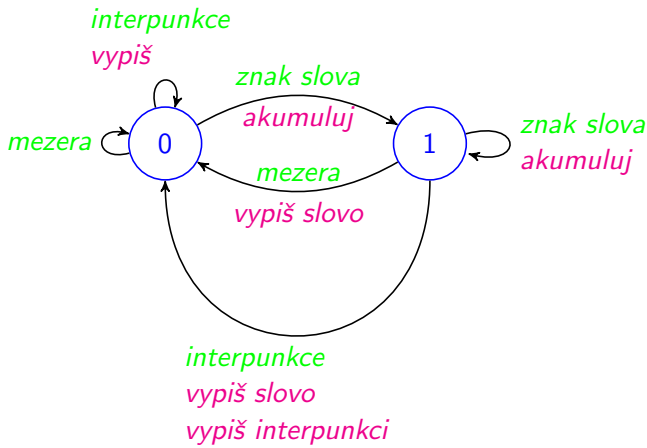
Příklad: Lexikální analýza textu

Rozdělte řetězec na posloupnost symbolů:

- *Interpunkce* — `.` `,` `;` `:` `?` `!` `"` `’` `(` `)` (jeden znak)
- *Slovo* — řetězec znaků, neobsahující interpunkci, mezery ani konce řádků.
- Konce řádků, mezery a tabulátory *(whitespace)* ignorujte.

`analyza_textu.py`

Analýza textu



Počáteční stav $q_0 = 0$.

```
def analyzuj(f):
  interpunkce="""".,;:?!"'()"""
  mezery=" \t\n"
  stav=0 # počáteční stav - mimo slovo
  while True:
    c=f.read(1)                # přečti jeden znak
    if c=="": return           # konec souboru
    if stav==0:
      if c in interpunkce: print("Interpunkce: ",c)
      elif c not in mezery:    # je to znak slova
        slovo=c; stav=1        # akumulátor slova
    else:                      # stav==1 - uvnitř slova
      if c not in interpunkce and c not in mezery:
        slovo+=c               # pořád uvnitř slova
      else:                    # slovo končí
        print("Slovo:       ", slovo); stav=0
        if c in interpunkce: print("Interpunkce: ",c)
if __name__=="__main__":
  analyzuj(sys.stdin)
```

```
$ python3 analyza_textu.py
```

```
Kam jdeš? Stůj!
```

```
Slovo:        Kam
Slovo:        jdeš
Interpunkce:  ?
Slovo:        Stůj
Interpunkce:  !
```

Nedeterministický automat

- Přechodová funkce vrací množinu stavů, $f : Q \times A \to 2^Q$, automat si jeden nedeterministicky ‘vybere’.
- Slovo je přijímáno automatem, pokud existuje posloupnost výběrů vedoucí do koncového stavu.
- Implementace:
  - Při simulaci si udržujeme množinu možných stavů.
  - Automaticky převedeme na deterministický automat s 2^n stavy.
  - Často lze nalézt ekvivalentní deterministický automat s méně stavy.

- Automat přijímající slova končící na *“ce”*
- Automat přijímající klíčová slova Pythonu.
- Automat přijímající celá nebo reálná čísla.

Nedeterministický automat

- Přechodová funkce vrací množinu stavů, $f : Q \times A \to 2^Q$, automat si jeden nedeterministicky 'vybere'.
- Slovo je přijímáno automatem, pokud existuje posloupnost výběrů vedoucí do koncového stavu.
- Implementace:
  - Při simulaci si udržujeme množinu možných stavů.
  - Automaticky převedeme na deterministický automat s 2^n stavy.
  - Často lze nalézt ekvivalentní deterministický automat s méně stavy.

- Automat přijímající slova končící na *"ce"*
- Automat přijímající klíčová slova Pythonu.
- Automat přijímající celá nebo reálná čísla.

Regulární výrazy v Pythonu

- Jazyk přijímaný konečným automatem = regulární jazyk.
- Regulární jazyk lze popsat *regulárním výrazem*.
- Pro regulární výrazy existují knihovny a nástroje.

Syntaxe

`.`	jakýkoliv znak
`[`*M*`]`	jakýkoliv znak z množiny *M*, lze `[a-z]`
`[^`*M*`]`	jakýkoliv znak mimo *M*
`*`	libovolný počet opakování předchozího
`+`	jedno a více opakování předchozího
`?`	žádné nebo jedno opakování předchozího
`()`	skupina
`|`	alternativy
`\`	následující znak ztrácí speciální význam

Další viz dokumentace

Příklad – jméno proměnné

```
import re
p=re.compile(r'[a-zA-Z_][a-zA-Z0-9_]*')
print(p.fullmatch("moje_promenna12"))
```

```
<re.Match object; span=(0, 15), match='moje_promenna12'>
```

```
print(p.fullmatch("moje{}_promenna"))
```

```
None
```

```
print(p.fullmatch("12moje{}_promenna"))
```

```
None
```

r'' znamená *raw* řetězec, bez interpretace

Příklad – jméno proměnné (2)

```
import re
variable_name_regexp=re.compile(r'[a-zA-Z_][a-zA-Z0-9_]*')
def is_variable_name(s):
   return variable_name_regexp.fullmatch(s) is not None
print(is_variable_name("moje_promenna12"))

print(is_variable_name("moje{}_promenna"))

print(is_variable_name("12moje{}_promenna"))
```

Příklad — reálné číslo

```
p=re.compile(r'[+-]?[0-9]+(\.[0-9]*)?([eE][+-]?[0-9]+)?')
print(p.fullmatch("-314"))

print(p.fullmatch("3.14"))

print(p.fullmatch("2341e-23"))

print(p.fullmatch("-2341e+23"))

print(p.fullmatch("-2341.e"))

print(p.fullmatch("278h"))
```

Přeskakování komentářů a regulární výrazy 1/2

```
1  import sys
2  import re
4  line_pattern=re.compile(r"([^#]*)(#.+)?\n")
6  def preskoc_komentare(f):
7    # vytiskne obsah souboru 'f' s vynechanymi komentari
8    for line in f.readlines(): # čte řádku po řádce
9      print(line_pattern.fullmatch(line).group(1))
11 if __name__=="__main__":
12   with open(sys.argv[1],'rt') as f: # otevři textový soubor
13     preskoc_komentare(f)  # pokud se povedlo, preskakuj
```

preskoc_komentare3.py

Přeskakování komentářů a regulární výrazy 2/2

Verze, kterou nezmate # v řetězci.

```
1  import sys
2  import re
4  line_pattern=re.compile(r'(([^#"]*("[^"]*")?)*)(#.+)?\n')
6  def preskoc_komentare(f):
7    # vytiskne obsah souboru 'f' s vynechanymi komentari
8    for line in f.readlines(): # čte řádku po řádce
9      print(line_pattern.fullmatch(line).group(1))
11 if __name__=="__main__":
12   with open(sys.argv[1],'rt') as f: # otevři textový soubor
13     preskoc_komentare(f)  # pokud se povedlo, preskakuj
```

preskoc_komentare4.py

Konečné automaty a regulární výrazy

Konečné automaty

- Jednoduchý výpočetní model
  - Rychlý, dobře teoreticky prozkoumaný.
  - Omezující – konečná paměť.
- Složitější implementace, existují nástroje pro její generování.
- Vhodné pro řízení jednoduchých systémů.
- Vhodné pro první krok analýzy textu i kódu v programovacích jazycích.
- Nedeterministické konečné automaty.

Regulární výrazy

- Teoreticky ekvivalentní konečným automatům.
- Snadné a rychlé použití, řada vyzkoušených a optimalizovaných knihoven.
- Některé složitější konstrukce jsou značně nepřehledné.
- Omezené množství výstupních akcí. Nelze použít pro řízení systému.