10. Konečný automat, regulární výrazy

BAB37ZPR – Základy programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Přehled témat

- Část 1 – Konečný automat
  - P10.1 Jméno proměnné
  - P10.2 Přeskakování komentářů
  - P10.3 Lexikální analýza textu
- Část 2 – Regulární výrazy
  - P10.4 Jméno proměnné
  - P10.5 Reálné číslo
  - P10.6 Přeskakování komentářů

Část I

Konečný automat

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:

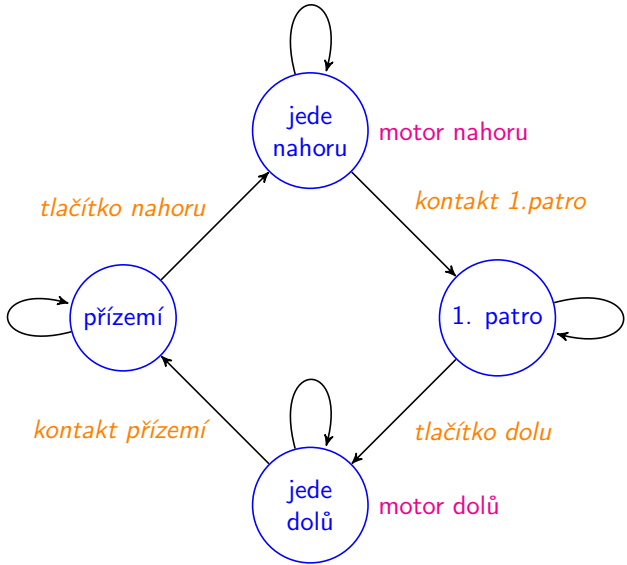
- konečná množina stavů Q
- počáteční stav $q_0 \in Q$
- konečná množina vstupních symbolů A
- přechodová funkce $f : Q \times A \to Q$

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

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.*

I. Konečný automat

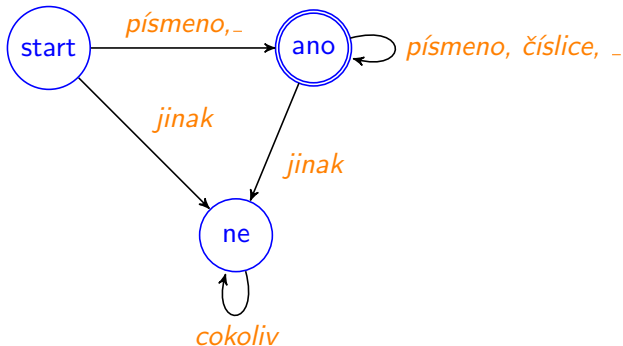
P10.1 Jméno proměnné

P10.2 Přeskakování komentářů

P10.3 Lexikální analýza textu

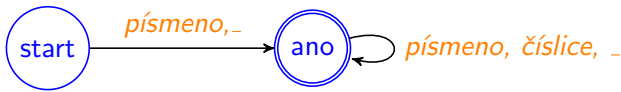
P10.1 Jméno proměnné – totální automat

Automat přijímá platná jména proměnných v Pythonu.



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

P10.1 Jméno proměnné – částečný automat



- Částečný automat – nedefinovaný přechod znamená nepřijetí.

P10.1 Jméno proměnné — implementace

```
>>> def is_variable_name(s):
...     """ je 's' platne jmeno promenne v Pythonu? """
...     state="start"
...     for c in s:
...       if state=="start":
...          if c=="_" or is_letter(c):
...            state="ano"
...          else:
...            return False
...       else: # stav "ano"
...         if not (c=="_" or is_letter(c) or c.isdigit()):
...           return False
...     return state=="ano"
...
```

automaton_examples.py.

P10.1 Jméno proměnné – příklady

```
>>> def is_letter(c):
...     return c.isupper() or c.islower()
...
>>> print(is_variable_name("moje_promenna12"))
True

>>> 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.

I. Konečný automat

P10.1 Jméno proměnné

P10.2 Přeskakování komentářů

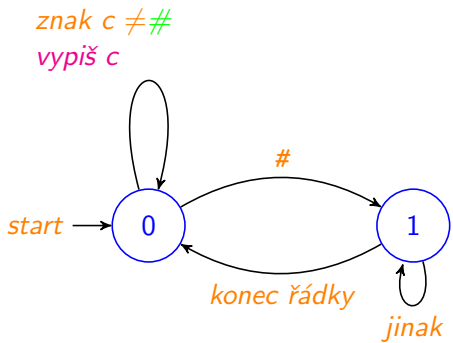
P10.3 Lexikální analýza textu

Problém

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

Problém

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



```
>>> 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=="#":              # začátek komentáře
...                 stav=1
...             else:
...                 print(c, end="") # vytiskni znak
...         else:                       # stav 1 -> komentar
...             if c=="\n":             # konec řádky
...                 stav=0              # konec komentáře
...                 print(c,end="")
...             else: pass              # vše ostatní ignorujeme
...
```

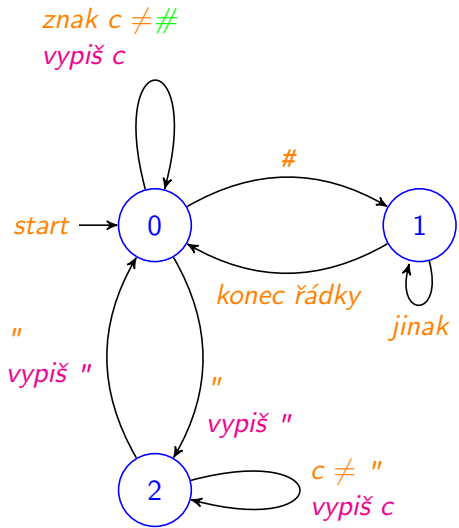
```
>>> # Tento program voláme s argumentem jméno souboru
>>> # python3 preskoc_komentare.py preskoc_komentare.py
>>> # Vypíše obsah souboru bez Pythonovských komentářů
>>> # vytiskne obsah souboru 'f' s vynechanymi komentari

>>> import sys

>>> if __name__=="__main__":
...     with open(sys.argv[1],'rt') as f: # otevři textový soubor
...         preskoc_komentare(f)  # pokud se povedlo, přeskakuj
...
```

preskoc_komentare.py

Je výstup programu očekávaný? Kde je problém?

Vylepšení

Verze, kterou nezmate # v řetězci.



Přeskakování komentářů – vylepšení


```
>>> def preskoc_komentare(f):
...     stav=0                          # počáteční stav automatu
...     while True:
...         c=f.read(1)                 # přečti jeden znak
...         if c=="":                   # konec souboru
...             return
...         if stav==0:                 # počáteční stav
...             if c=="#":              # začátek komentáře
...                 stav=1
...             else:
...                 if c=='"':          # začátek řetězce
...                     stav=2
...                 print(c,end="")     # vytiskni znak
```

```
...         elif stav==1:                   # 1="komentar"
...             if c=="\n":                 # konec řádky, konec komentáře
...                 stav=0
...                 print(c, end="")
...             else:                       # vše ostatní ignorujeme
...                 pass
...         else:                           # stav = řetězec
...             if c=='"':
...                 stav=0
...                 print(c,end="")         # vytiskni znak
...
```

I. Konečný automat

P10.1 Jméno proměnné

P10.2 Přeskakování komentářů

P10.3 Lexikální analýza textu

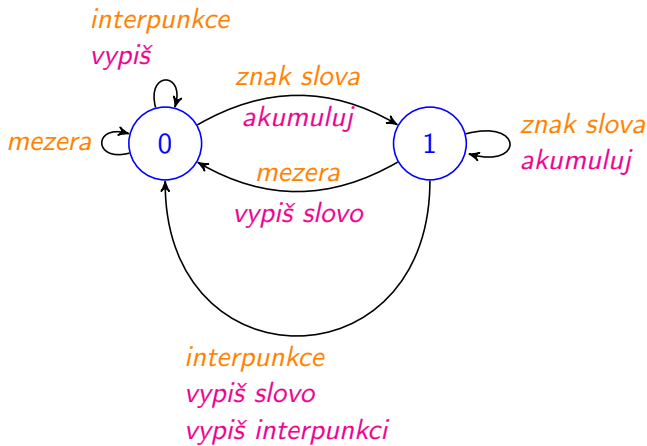
- Vstup: řetězec (posloupnost znaků)
- Výstup: posloupnost symbolů *(tokens)* – číslo, operátor, identifikátor, klíčové slovo, ...
  - Symboly mohou mít atributy — řetězec, hodnota, ...
  - Reprezentujeme jako dvojice — typ symbolu + atribut
- Další operace
  - Vynechání komentářů
  - Odstranění mezer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

Problém

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`



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.

Část II

Regulární výrazy

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

II. Regulární výrazy

P10.4 Jméno proměnné

P10.5 Reálné číslo

P10.6 Přeskakování komentářů

```
>>> import re
>>> # r' znamená raw řetězec, bez interpretace
>>> 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

>>> print(p.match("moje{}_promenna"))
<re.Match object; span=(0, 4), match='moje'>
```

```
>>> 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"))
True

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

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

II. Regulární výrazy

P10.4 Jméno proměnné

P10.5 Reálné číslo

P10.6 Přeskakování komentářů

P10.5 Reálné číslo

```
>>> p=re.compile(r'[+-]?[0-9]+(\.[0-9]*)?([eE][+-]?[0-9]+)?')
>>> print(p.fullmatch("-314"))
<re.Match object; span=(0, 4), match='-314'>
>>> print(p.fullmatch("3.14"))
<re.Match object; span=(0, 4), match='3.14'>
>>> print(p.fullmatch("2341e-23"))
<re.Match object; span=(0, 8), match='2341e-23'>
>>> print(p.fullmatch("-2341e+23"))
<re.Match object; span=(0, 9), match='-2341e+23'>
>>> print(p.fullmatch("-2341.e"))
None
>>> print(p.fullmatch("278h"))
None
```

II. Regulární výrazy

P10.4 Jméno proměnné

P10.5 Reálné číslo

P10.6 Přeskakování komentářů

P10.6 Přeskakování komentářů a regulární výrazy 1/2

```
>>> import sys
>>> import re

>>> line_pattern=re.compile(r"([^#]*)(#.+)?\n")

>>> def preskoc_komentare(f):
...   # vytiskne obsah souboru 'f' s vynechanymi komentari
...   for line in f.readlines(): # čte řádku po řádce
...     print(line_pattern.fullmatch(line).group(1))
...
>>> if __name__=="__main__":
...   with open(sys.argv[1],'rt') as f: # otevři textový soubor
...     preskoc_komentare(f)  # pokud se povedlo, preskakuj
...
```

preskoc_komentare3.py

Verze, kterou nezmate # v řetězci.

```
>>> import sys
>>> import re

>>> line_pattern=re.compile(r'(([^#"]*("[^"]*")?)*)(#.+)?\n')

>>> def preskoc_komentare(f):
...   # vytiskne obsah souboru 'f' s vynechanymi komentari
...   for line in f.readlines(): # čte řádku po řádce
...     print(line_pattern.fullmatch(line).group(1))
...
>>> if __name__=="__main__":
...   with open(sys.argv[1],'rt') as f: # otevři textový soubor
...     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.