# Procedūrinio programavimo pagrindai

## Gramatikos

lekt. Irmantas Radavičius
irmantas.radavicius@mif.vu.lt
Informatikos institutas, MIF, VU

# Turinys

# Formaliosios gramatikos

Gramatikos naudojamos kalboms (pvz. C) apibrėžti.

Gramatika – tai ketvertas G = (N, T, P, S), kuriame

N – „kintamieji“ (neterminalinių simbolių aibė)

T – apibrėžiamos kalbos (terminaliniai) simboliai, $N \cap T = \emptyset$

P – generavimo (produkcijos) taisyklės, kurių kiekviena

$P : (N \cup T)^* N (N \cup T)^* \rightarrow (N \cup T)^*$

S – pradinis (starto) simbolis

Žodžiui gauti, taikomos generavimo taisykles (nuo S)
kol nebelieka neterminalinių simbolių.

Žodis priklauso kalbai L(G), jei jį galima gauti iš S taikant taisykles iš P.

# Chomskio hierarchija

Rekursyviai skaičios gramatikos (kalbas atpažįsta Turingo mašina)

Kontekstui jautrios gramatikos

$p : \alpha A \beta \rightarrow \alpha \gamma \beta$

Bekontekstės gramatikos

$p : A \rightarrow \gamma$

Reguliariosios gramatikos (kalbas atpažįsta baigtinis automatas)

$p : A \rightarrow a$

$p : A \rightarrow aB$ (dešinioji) arba $p : A \rightarrow Ba$ (kairioji)

# Reguliarūs reiškiniai

Regularūs reiškiniai/išraiškos (angl. regular expression, regex, regexp) – būdas teksto šablonui apibrėžti

Baigtinei simbolių aibei Σ regexp nusako:

- tuščia aibė Ø,
- aibė iš tuščio žodžio { ε }
- aibė { a ∈Σ }

Jei jei R ir S yra regexp, kiti regexp gaunami operacijomis:

- R*(kartojimas),
- RS (prijungimas),
- R|S (alternatyva)

Grupavimui, naudojami skliaustai.

Variantai... `/http[s]?:\/\/\w+\.com/`

# Bekaus-Nauro forma

Būdas užrašyti bekontekstę gramatiką.

Susideda iš taisyklių, turinčių pavidalą:

```
<symbol> ::= __expression__
```

Terminaliniai simboliai – tie, kurių nėra kairėje,
kiti neterminaliniai,
pradinis simbolis – pirmos taisyklės kairėje.

Pavyzdžiai:

```
<expr> ::= <term>|<expr><addop><term>
```

```
<integer> ::= <digit>|<integer><digit>
```

# BNF in BNF

```
 <syntax>         ::= <rule> | <rule> <syntax>
 <rule>           ::= <opt-whitespace> "<" <rule-name> ">" <opt-whitespace> "::=" <opt-
whitespace> <expression> <line-end>
 <opt-whitespace> ::= " " <opt-whitespace> | ""
 <expression>     ::= <list> | <list> <opt-whitespace> "|" <opt-whitespace>
<expression>
 <line-end>       ::= <opt-whitespace> <EOL> | <line-end> <line-end>
 <list>           ::= <term> | <term> <opt-whitespace> <list>
 <term>           ::= <literal> | "<" <rule-name> ">"
 <literal>        ::= '"' <text1> '"' | "'" <text2> "'"
 <text1>          ::= "" | <character1> <text1>
 <text2>          ::= "" | <character2> <text2>
 <character>      ::= <letter> | <digit> | <symbol>
 <letter>         ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
| "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" |
"o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
 <digit>          ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
 <symbol>         ::=  "|" | " " | "-" | "!" | "#" | "$" | "%" | "&" | "(" | ")" | "*"
| "+" | "," | "-" | "." | "/" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "[" | "\" |
"]" | "^" | "_" | "`" | "{" | "|" | "}" | "~"
 <character1>     ::= <character> | "'"
 <character2>     ::= <character> | '"'
 <rule-name>      ::= <letter> | <rule-name> <rule-char>
 <rule-char>      ::= <letter> | <digit> | "-"
```

<lecture> ::= end