Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Softvérové jazyky

Syntaktický analyzátor - simpleURL

Obsah

Zadanie

simpleURL
Pre definovanú gramatiku simpleURL v BNF vytvorte tabul'kou riadený syntaktický analyzátor (SA) zhora nadol. Postupne realizujte nasledujúce kroky:

1. Vytvorte niekol'ko príkladov viet daného jazyka (korektné aj nekorektné).
2. Prepíšte gramatiku z BNF (Backus-Naur Form) do gramatických pravidiel s alternatívami. Vyznačte miesta, kde gramatika nespĺňa podmienky pre deterministickú analýzu.
3. Transformujte gramatiku tak, aby bola LL(1):
   a. odstráňte l'avú rekurziu (ak treba)
   b. urobte l'avú faktorizáciu (ak treba)
4. Vyšetrite, či je takto upravená gramatika je typu LL(1). Konkrétna nájdite:
   a. množinu FIRST1 pre každý neterminálny symbol v transformovanej gramatike
   b. množinu FOLLOW1 pre každý neterminálny symbol v transformovanej gramatike
   V prípade, že gramatika stále nie je typu LL(1), upravte pravidlá tak, aby ju bolo možné transformovat' na LL(1).
5. Pre transformovanú gramatiku typu LL(1) vytvorte tabul'ku prechodov.

Pravidlá BNF:

1. url → httpaddress | ftpaddress | telnetaddress | mailtoaddress
2. httpaddress → 'http : //' hostport ['/' path] ['?' search]
3. ftpaddress → 'ftp : //' login '/' path
4. telnetaddress → 'telnet : //' login
5. mailtoaddress → 'mailto ::' xalphas '@' hostname
6. login → [user [':' password] '@'] hostport
7. hostport → hostname [':' port]
8. hostname → xalphas { '.' xalphas }
9. port → digits
10. path → segment { '/' segment }
11. search → xalphas { '+' xalphas }
12. user → xalphas
13. password → xalphas
14. segment → { xalpha }
15. xalphas → xalpha { xalpha }
16. xalpha → alpha | digit
17. digits → digit{ digit }
18. alpha → A | .. | Z | a | .. | z .
19. digit → 0 | .. | 9 .

Príklady viet jazyka

Testovanie sme vykonávali na príkladoch, ktoré sa nachádzajú v súbore ***inputs.txt.*** Čiernou farbou sú vyznačené vstupy, ktoré analyzátor akceptoval a červenou vstupy, ktoré obsahovali chybu, či už tokenizátor nedokázal nájsť príslušný znak a priradiť ho niektorému z tokenov, alebo jednoducho neexistuje pravidlo, ktoré by tento sled znakov dokázalo spracovať.

1. **http://s0ftv3r0v3j4zyky.jv/path/to/file**
   - testujeme základnú http url, za ktorou nasleduje viacero za sebou idúcich /path
2. **http://s0ftv3r0v3j4zyky.jv/path/to/file?search1keyword**
   - testujeme rovnakú url ako v prípade 1 s pridaním search
   - reálny search by mal byť vo forme ?search=keyword, ale keďže nemáme znak = v gramatike tak sme ho nahradili 1 aby bol vstup akceptovaný
3. **ftp://cyprich@smrecek.sj/**
4. **ftp://student:cyprich@smrecek.sj/directory/file**
   - testujeme ftp adresu s loginom vo forme USER@PASSWORD (e.g.1) a USER:PASSWORD @ HOSTPORT (e.g.2) za ktorým musí ísť terminál "/" a môže/nemusí následovať "/" PATH
5. **ftp://:cyprich@smrecek.sj/directory/file**
   - neakceptovaný vstup po ftp terminále musí následovať LOGIN / PATH a nie terminál :
   - Terminál : nesmie následovať práve kvôli Neterminálu USER, ktorý negeneruje epsilon
6. **telnet://student:cyprich@smrecek.sj**
   - testujeme telnet adresu, za ktorou nasleduje LOGIN rovnako ako pri ftp s tým rozdielom, že následne nesmie následovať /PATH
7. **telnet://student:cyprich@smrecek.sj/directory/file/path?subject+HelloPERCENT20WorldANDbody+LoremPERCENT20Ipsum**
   - testujeme /PATH pre telnet, ale ako bolo vyššie spomenuté vstup je rejected, keďže telnet nesmie obsahovať /PATH
8. **telnet://student:cyprich@smrecek:0101**
   - testujeme telnet url, kde je USER s PASSWORD, za ktorým nasleduje @ HOSTNAME :PORT
9. **telnet://student:cyprich@smrecek:0101.sj**
   - v tomto prípade telnet utr nesmie mať príponu .xy teda by tento vstup analyzátor neakceptoval
10. **http://smrecek.sj/directory/file/path?subject+HelloPERCENT20WorldANDbody+LoremPERCENT20Ipsum**
    - testujeme http url, za ktorým ide PATH a následne SEARCH
    - znaky ako % ,= alebo &, ktoré sa zvyčajne v reálnom searchu nachádzajú sme nahradili symbolmi z našej gramatiky
11. **http://smrecek.sj/directory?subject+HelloPERCENT20WorldANDbody+LoremPERCENT20Ipsum**
    - testujeme http url, bez PATH ale s násobným SEARCHOM s terminálnymi symbolmi z našej gramatiky

12. **http://s0ftv3r0v3j4zyky.jv/path1/path2/file?search+query+filter+1**
    - testujeme správny formát http url, za ktorým môže následuje PATH aj SEARCH aj PLUSSEARCH
13. **http://s0ftv3r0v3j4zyky.jv/path1/path2/file?search=query&filter=1**
    - testujeme rovnakú url akurát so znakmi, ktoré nie sú v našej gramatike preto ich tokenizer nevie rozpoznať a vstup je tým pádom rejected
14. **http://s0ftv3r0v3j4zyky.jv:8080/path/to/file**
    - testujeme http url, pri ktorej je zadaný aj port a teda testujeme pravidlo PORT
15. **http://s0ftv3r0v3j4zyky.jv:2121/path/to/file?search+keyword**
    - testujeme http url, s definovaným portom, násobným PATH a násobným SEARCH
16. **mailto::cyprich@smrecek**
    - testujeme základny tvar pre mailto:: adresa bez prípony .xy
17. **mailto::cyprich@smrecek.sj**
    - testujeme tvar s príponou .xy
18. **mailto::cyprich@smrecek.sj?subject+HelloPERCENT20WorldANDbody+LoremPERCENT20Ipsum**
    - testujeme pridanie SEARCH za príponu .xy, vstup je zamietnutý, pretože pravidlo, ktoré by podporovalo tento tvar neexistuje

Prepis do gramatických pravidiel s alternatívami

Na začiatok sme odstránili medzery z terminálov e.g. ``http : //`` sme zmenili na ``http://`` a zátvorky symbolizujúce počet výskytov.

1. url → httpaddress| ftpaddress| telnetaddress| mailtoaddress.
2. httpaddress → 'http://' hostport |
   'http://' hostport '/' path|
   'http://' hostport '?' search|
   'http://' hostport '/' path '?' search.
3. ftpaddress → 'ftp://' login '/' path.
4. telnetaddress → 'telnet://' login.
5. mailtoaddress → 'mailto::' xalphas '@' hostname.
6. login → hostport |
   user '@' hostport|
   user ':' password '@' hostport.
7. hostport → hostname |
   hostname ':' port.
8. hostname → xalphas| xalphas '.' hostname.
9. port → digits.
10. path → segment| segment '/' path.
11. search → xalphas '+' search.
12. user → xalphas.
13. password → xalphas.
14. segment → xalpha segment | ε.
15. xalphas → xalpha | xalpha xalphas.
16. xalpha → alpha | digit.
17. digits → digit| digit digits.
18. alpha → 'A' | .. | 'Z' | 'a' | .. | 'z'.
19. digit → '0' | .. | '9'.

Úprava zápisu

Zápis gramatiky sme upravili tak, aby bolo možné gramatiku skontrolovať v nástroji http://smlweb.cpsc.ucalgary.ca/

Vzhľadom na to, že nástroj neakceptuje niektoré znaky, bolo potrebné ich upraviť. Terminál . (bodku) sme nahradili terminálom dot.

Výsledná úprava gramatiky, ktorú vie nástroj prečítať.

1. URL -> HTTPADDRESS | FTPADDRESS | TELNETADDRESS | MAILTOADDRESS.
2. HTTPADDRESS -> http:// HOSTPORT | http:// HOSTPORT / PATH | http:// HOSTPORT ? SEARCH | http:// HOSTPORT / PATH ? SEARCH.
3. FTPADDRESS -> ftp:// LOGIN / PATH.
4. TELNETADDRESS -> telnet:// LOGIN.
5. MAILTOADDRESS -> mailto:: XALPHAS @ HOSTNAME.
6. LOGIN -> HOSTPORT | USER @ HOSTPORT | USER : PASSWORD @ HOSTPORT.
7. HOSTPORT -> HOSTNAME | HOSTNAME : PORT.
8. HOSTNAME -> XALPHAS | XALPHAS dot HOSTNAME.
9. PORT -> DIGITS.
10. PATH -> SEGMENT | SEGMENT / PATH.
11. SEARCH -> XALPHAS | XALPHAS + SEARCH.
12. USER -> XALPHAS.
13. PASSWORD -> XALPHAS.
14. SEGMENT -> XALPHA SEGMENT | .
15. XALPHAS -> XALPHA | XALPHA XALPHAS.
16. XALPHA -> ALPHA | DIGIT.
17. DIGITS -> DIGIT | DIGIT DIGITS.
18. ALPHA -> a | b | c | d | e | f | g.
19. DIGIT -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.

Výstup:

- Všetky neterminály sú dosiahnuteľné a uskutočniteľné.
- Neterminály s možnosťou null sú: SEGMENT PATH.
- Ukončiteľné neterminály sú: MOREDIGITS DIGIT ALPHA MOREXALPHAS XALPHA PLUSSEARCH SLASHPATH SEGMENT DIGITS DOTHOSTNAME XALPHAS PORT HOSTNAME LOGIN SEARCH PATH HOSTPORT URL MAILTOADDRESS TELNETADDRESS FTPADDRESS HTTPADDRESS.
- Žiadne cykly.

Gramatika nie je LL(1), pretože:

- HTTPADDRESS má FIRST FIRST konflikt.
- LOGIN má FIRST FIRST konflikt.
- HOSTPORT má FIRST FIRST konflikt.
- HOSTNAME má FIRST FIRST konflikt.
- PATH má FIRST FIRST konflikt.
- SEARCH má FIRST FIRST konflikt.

- XALPHAS má FIRST FIRST konflikt.
- DIGITS má FIRST FIRST konflikt.

Konflikt:

1. FIRST(URL) = FIRST(HTTPADDRESS) ∪ FIRST(FTPADDRESS) ∪ FIRST(TELNETADDRESS) ∪ FIRST(MAILTOADDRESS)
2. FIRST(HTTPADDRESS) = {http://} ∪ {http://} ∪ {http://} ∪ {http://}

Keďže už pri tvorení množiny FIRST pre druhý neterminál gramatiky HTTPADDRESS nastal FIRST FIRST konflikt, ďalej nepokračujeme v tvorbe množín FIRST a FOLLOW, ale prechádzame ku kroku odstraňovania prefixov a rekurzií.

Odstránenie prefixov a rekurzií

Zohľadňujúc gramatiku uvedenú na predošlej strane:

1. Pravidlo sa nezmenilo.
2. Bol odstránení prefix ‚http:// HOSTPORT‘, následne bol odstránený prefix ‚/ PATH‘. Vznikli tak 2 nové neterminálne symboly: SUFPATHSEARCH a SUFSEARCH.
3. Pravidlo sa nezmenilo.
4. Pravidlo sa nezmenilo.
5. Pravidlo sa nezmenilo.
6. Bol odstránený prefix ‚USER‘. Vznikol tak nový neterminálny symbol SUFPASSHOST.
7. Bol odstránený prefix ‚HOSTNAME‘. Vznikol tak nový neterminálny symbol SUFPORT.
8. Bol odstránený prefix ‚XALPHAS‘. Vznikol tak nový neterminálny symbol DOTHOSTNAME.
9. Pravidlo sa nezmenilo.
10. Bol odstránený prefix ‚SEGMENT‘. Vznikol tak nový neterminálny symbol SLASHPATH.
11. Bol odstránený prefix ‚XALPHAS‘. Vznikol tak nový neterminálny symbol PLUSSEARCH.
12. Pravidlo sa nezmenilo.
13. Pravidlo sa nezmenilo.
14. Pravidlo sa nezmenilo.
15. Bol odstránený prefix ‚XALPHA‘. Vznikol tak nový neterminálny symbol MOREXALPHAS.
16. Pravidlo sa nezmenilo.
17. Bol odstránený prefix ‚DIGIT‘. Vznikol tak nový neterminálny symbol MOREDIGITS.
18. Pravidlo sa nezmenilo.
19. Pravidlo sa nezmenilo.

Gramatika neobsahovala ľavú rekurziu ani v jednom z pravidiel.

Gramatika po aplikovaní úprav:

1. URL -> HTTPADDRESS | FTPADDRESS | TELNETADDRESS | MAILTOADDRESS.
2. HTTPADDRESS -> http:// HOSTPORT SUFPATHSEARCH.
3. SUFPATHSEARCH -> / PATH SUFSEARCH | ? SEARCH | ε.
4. SUFSEARCH -> ? SEARCH | ε.
5. FTPADDRESS -> ftp:// LOGIN / PATH.
6. TELNETADDRESS -> telnet:// LOGIN.
7. MAILTOADDRESS -> mailto:: XALPHAS @ HOSTNAME.
8. LOGIN -> HOSTPORT | USER SUFPASSHOST.
9. SUFPASSHOST -> @ HOSTPORT | : PASSWORD @ HOSTPORT.
10. HOSTPORT -> HOSTNAME SUFPORT.
11. SUFPORT -> : PORT | ε.
12. HOSTNAME -> XALPHAS DOTHOSTNAME.
13. DOTHOSTNAME -> dot HOSTNAME | ε.
14. PORT -> DIGITS.
15. PATH -> SEGMENT SLASHPATH.
16. SLASHPATH -> / PATH | ε.
17. SEARCH -> XALPHAS PLUSSEARCH.

18. PLUSSEARCH -> + SEARCH | ε.
19. USER -> XALPHAS.
20. PASSWORD -> XALPHAS.
21. SEGMENT -> XALPHA SEGMENT | ε.
22. XALPHAS -> XALPHA MOREXALPHAS.
23. MOREXALPHAS -> XALPHAS | ε.
24. XALPHA -> ALPHA | DIGIT.
25. DIGITS -> DIGIT MOREDIGITS.
26. MOREDIGITS -> DIGITS | ε.
27. ALPHA -> a | b | c | d | e | f | g.
28. DIGIT -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.

Výstup:

- Všetky neterminály sú dosiahnuteľné a uskutočniteľné.
- Neterminály s možnosťou null sú: SUFPATHSEARCH SUFSEARCH SUFPORT DOTHOSTNAME SLASHPATH PLUSSEARCH SEGMENT MOREXALPHAS MOREDIGITS PATH.
- Ukončiteľné neterminály sú: MOREDIGITS DIGIT ALPHA MOREXALPHAS XALPHA PLUSSEARCH SLASHPATH SEGMENT DIGITS DOTHOSTNAME XALPHAS PORT SUFPORT SUFPASSHOST HOSTNAME LOGIN SEARCH SUFSEARCH PATH SUFPATHSEARCH HOSTPORT URL MAILTOADDRESS TELNETADDRESS FTPADDRESS HTTPADDRESS.
- Žiadne cykly.

Gramatika nie je LL(1), pretože:

- LOGIN má FIRST FIRST konflikt.

Množina FIRST s konfliktom

Z nástroja vieme, že pri neterminále LOGIN je prítomný FIRST FIRST konflikt. Avšak pre jeho vizualizáciu si vypočítame množinu FIRST manuálne.

V gramatike máme 28 neterminálov, uvádzame ich v rovnakom poradí ako v upravenej gramatike na predošlej strane. Výpočet FIRST pre tieto neterminály je nasledovný:

1. FIRST(URL) = FIRST(HTTPADDRESS) ∪ FIRST(FTPADDRESS) ∪ FIRST(TELNETADDRESS) ∪ FIRST(MAILTOADDRESS) = {http://, ftp://, telnet://, mailto::}
2. FIRST(HTTPADDRESS) = {http://}
3. FIRST(SUFPATHSEARCH) = {/} ∪ {?} ∪ {ε} = {/, ?, ε}
4. FIRST(SUFSEARCH) = {?} ∪ {ε} = {?, ε}
5. FIRST(FTPADDRESS) = {ftp://}
6. FIRST(TELNETADDRESS) = {telnet://}
7. FIRST(MAILTOADDRESS) = {mailto::}
8. FIRST(LOGIN) = FIRST(HOSTPORT) ∪ FIRST(USER) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9} ∪ {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
9. FIRST(SUFPASSHOST) = {@} ∪ {:} = {@, :}
10. FIRST(HOSTPORT) = FIRST(HOSTNAME) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
11. FIRST(SUFPORT) = {:} ∪ {ε} = {:, ε}
12. FIRST(HOSTNAME) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
13. FIRST(DOTHOSTNAME) = {dot} ∪ {ε} = {dot, ε}
14. FIRST(PORT) = FIRST(DIGITS) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
15. FIRST(PATH) = FIRST(SEGMENT) \ {ε} ∪ FIRST(SLASHPATH) = {/, a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
16. FIRST(SLASHPATH) = {/} ∪ {ε} = {/, ε}
17. FIRST(SEARCH) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
18. FIRST(PLUSSEARCH) = {+} ∪ {ε} = {+, ε}
19. FIRST(USER) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
20. FIRST(PASSWORD) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
21. FIRST(SEGMENT) = FIRST(XALPHA) ∪ {ε} = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
22. FIRST(XALPHAS) = FIRST(XALPHA) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
23. FIRST(MOREXALPHAS) = FIRST(XALPHAS) ∪ {ε} = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
24. FIRST(XALPHA) = FIRST(ALPHA) ∪ FIRST(DIGIT) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
25. FIRST(DIGITS) = FIRST(DIGIT) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
26. FIRST(MOREDIGITS) = FIRST(DIGITS) ∪ {ε} = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
27. FIRST(ALPHA) = {a, b, c, d, e, f, g}
28. FIRST(DIGIT) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Vidíme, že v pravidle 8 nastal FIRST FIRST konflikt. Množiny FOLLOW preto zatiaľ nepočítame a prechádzame ku kroku úpravy zadania.

Úprava pravidla

Keďže sme nevedeli upraviť pravidlá na LL(1) tvar tak sme modifikovali zadanie.

Pôvodné pravidlo:

login := [user [':' password] '@'] hostport.

sme zmenili na

login := user [':' password] '@' hostport.

Tým krokom sme docielili, že sme dostali gramatiku LL(1):

1. URL -> HTTPADDRESS | FTPADDRESS | TELNETADDRESS | MAILTOADDRESS.
2. HTTPADDRESS -> http:// HOSTPORT SUFPATHSEARCH.
3. SUFPATHSEARCH -> / PATH SUFSEARCH | ? SEARCH | ε.
4. SUFSEARCH -> ? SEARCH | ε.
5. FTPADDRESS -> ftp:// LOGIN / PATH.
6. TELNETADDRESS -> telnet:// LOGIN.
7. MAILTOADDRESS -> mailto:: XALPHAS @ HOSTNAME.
8. LOGIN -> USER SUFPASSHOST.
9. SUFPASSHOST -> @ HOSTPORT | : PASSWORD @ HOSTPORT.
10. HOSTPORT -> HOSTNAME SUFPORT.
11. SUFPORT -> : PORT | ε.
12. HOSTNAME -> XALPHAS DOTHOSTNAME.
13. DOTHOSTNAME -> dot HOSTNAME | ε.
14. PORT -> DIGITS.
15. PATH -> SEGMENT SLASHPATH.
16. SLASHPATH -> / PATH | ε.
17. SEARCH -> XALPHAS PLUSSEARCH.
18. PLUSSEARCH -> + SEARCH | ε.
19. USER -> XALPHAS.
20. PASSWORD -> XALPHAS.
21. SEGMENT -> XALPHA SEGMENT | ε.
22. XALPHAS -> XALPHA MOREXALPHAS.
23. MOREXALPHAS -> XALPHAS | ε.
24. XALPHA -> ALPHA | DIGIT.
25. DIGITS -> DIGIT MOREDIGITS.
26. MOREDIGITS -> DIGITS | ε.
27. ALPHA -> a | b | c | d | e | f | g.
28. DIGIT -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.

Túto gramatiku nám nástroj http://smlweb.cpsc.ucalgary.ca/ označil ako správnu LL(1) gramatiku.

Výstup:

- Všetky neterminály sú dosiahnuteľné a uskutočniteľné.
- Neterminály s možnosťou null sú: SUFPATHSEARCH SUFSEARCH SUFPORT DOTHOSTNAME SLASHPATH PLUSSEARCH SEGMENT MOREXALPHAS MOREDIGITS PATH.
- Ukončiteľné neterminály sú: MOREDIGITS DIGIT ALPHA MOREXALPHAS XALPHA PLUSSEARCH SLASHPATH SEGMENT DIGITS DOTHOSTNAME XALPHAS PORT SUFPORT SUFPASSHOST HOSTNAME LOGIN SEARCH SUFSEARCH PATH SUFPATHSEARCH HOSTPORT URL MAILTOADDRESS TELNETADDRESS FTPADDRESS HTTPADDRESS.
- Žiadne cykly.

Množiny FIRST a FOLLOW

V gramatike máme 28 neterminálov, uvádzame ich v rovnakom poradí ako v upravenej gramatike na predošlej strane. Výpočet FIRST pre tieto neterminály je nasledovný:

First:

1. FIRST(URL) = FIRST(HTTPADDRESS) ∪ FIRST(FTPADDRESS) ∪ FIRST(TELNETADDRESS) ∪ FIRST(MAILTOADDRESS) = {http://, ftp://, telnet://, mailto::}
2. FIRST(HTTPADDRESS) = {http://}
3. FIRST(SUFPATHSEARCH) = {/} ∪ {?} ∪ {ε} = {/, ?, ε}
4. FIRST(SUFSEARCH) = {?} ∪ {ε} = {?, ε}
5. FIRST(FTPADDRESS) = {ftp://}
6. FIRST(TELNETADDRESS) = {telnet://}
7. FIRST(MAILTOADDRESS) = {mailto::}
8. FIRST(LOGIN) = FIRST(USER) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
9. FIRST(SUFPASSHOST) = {@} ∪ {:} = {@, :}
10. FIRST(HOSTPORT) = FIRST(HOSTNAME) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
11. FIRST(SUFPORT) = {:} ∪ {ε} = {:, ε}
12. FIRST(HOSTNAME) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
13. FIRST(DOTHOSTNAME) = {dot} ∪ {ε} = {dot, ε}
14. FIRST(PORT) = FIRST(DIGITS) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
15. FIRST(PATH) = FIRST(SEGMENT) \ {ε} ∪ FIRST(SLASHPATH) = {/, a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
16. FIRST(SLASHPATH) = {/} ∪ {ε} = {/, ε}
17. FIRST(SEARCH) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
18. FIRST(PLUSSEARCH) = {+} ∪ {ε} = {+, ε}
19. FIRST(USER) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
20. FIRST(PASSWORD) = FIRST(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
21. FIRST(SEGMENT) = FIRST(XALPHA) ∪ {ε} = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
22. FIRST(XALPHAS) = FIRST(XALPHA) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
23. FIRST(MOREXALPHAS) = FIRST(XALPHAS) ∪ {ε} = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
24. FIRST(XALPHA) = FIRST(ALPHA) ∪ FIRST(DIGIT) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
25. FIRST(DIGITS) = FIRST(DIGIT) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
26. FIRST(MOREDIGITS) = FIRST(DIGITS) ∪ {ε} = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε}
27. FIRST(ALPHA) = {a, b, c, d, e, f, g}
28. FIRST(DIGIT) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Keďže v tomto prípade sa v gramatike nevyskytuje žiaden FIRST FIRST konflikt, pokračujeme výpočtom množiny FOLLOW pre každý neterminál.

Follow:

1. FOLLOW(URL) = {$}
2. FOLLOW(HTTPADDRESS) = FOLLOW(URL) = {$}
3. FOLLOW(SUFPATHSEARCH) = FOLLOW(HTTPADDRESS) = {$}
4. FOLLOW(SUFSEARCH) = FOLLOW(SUFPATHSEARCH) = {$}
5. FOLLOW(FTPADDRESS) = FOLLOW(URL) = {$}
6. FOLLOW(TELNETADDRESS) = FOLLOW(URL) = {$}
7. FOLLOW(MAILTOADDRESS) = FOLLOW(URL) = {$}
8. FOLLOW(LOGIN) = {/} U FOLLOW(TELNETADDRESS) = {/} U{$} = {/, $}
9. FOLLOW(SUFPASSHOST) = FOLLOW(LOGIN) = {/, $}
10. FOLLOW(HOSTPORT) = FIRST(SUFPATHSEARCH) / {ε} U FOLLOW (HTTPADDRESS) U FOLLOW(SUFPASSHOST) U FOLLOW(SUFPASSHOST) = {/, ?} U {$} U {/, $} U {/, $} = {/, ?, $}
11. FOLLOW(SUFPORT) = FOLLOW(HOSTPORT) = {/, ?, $}
12. FOLLOW(HOSTNAME) = FOLLOW(MAILTOADDRESS) U FIRST(SUFPORT) / {ε} U FOLLOW(HOSTPORT) U FOLLOW(DOTHOSTNAME) = {$} U {:} U {/, ?, $} = {:, /, ?, $}
13. FOLLOW(DOTHOSTNAME) = FOLLOW(HOSTNAME) = {:, /, ?, $}
14. FOLLOW(PORT) = FOLLOW(SUFPORT) = {/, ?, $}
15. FOLLOW(PATH) = FIRST(SUFSEARCH) / {ε} U FOLLOW(SUFPATHSEARCH) U FOLLOW(FTPADDRESS) U FOLLOW(SLASHPATH) = {?} U {$} U {$} = {?, $}
16. FOLLOW(SLASHPATH) = FOLLOW(PATH) = {?, $}
17. FOLLOW(SEARCH) = FOLLOW(SUFPATHSEARCH) U FOLLOW(SUFSEARCH) U FOLLOW(PLUSSEARCH) = {$} U {$} = {$}
18. FOLLOW(PLUSSEARCH) = FOLLOW(SEARCH) = {$}
19. FOLLOW(USER) = FIRST(SUFPASSHOST) = {@, :}
20. FOLLOW(PASSWORD) = {@}
21. FOLLOW(SEGMENT) = FIRST(SLASHPATH) / {ε} U FOLLOW(PATH) U FOLLOW(SEGMENT) = {/} U {?, $} = {/, ?, $}
22. FOLLOW(XALPHAS) = {@} U FIRST(DOTHOSTNAME) / {ε} U FOLLOW(HOSTNAME) U FIRST(PLUSSEARCH) / {ε} U FOLLOW(SEARCH) U FOLLOW(USER) U FOLLOW(PASSWORD) U FOLLOW(MOREXALPHAS) = {@} U{dot} U {:, /, ?, $} U {+} U {$} U {@, :} U {@} = {:, @, dot, +, /, ?, $}
23. FOLLOW(MOREXALPHAS) = FOLLOW(XALPHAS) = {:, @, dot, +, /, ?, $}
24. FOLLOW(XALPHA) = FIRST(SEGMENT) / {ε} U FOLLOW(SEGMENT) U FIRST(MOREXALPHAS) / {ε} U FOLLOW(XALPHAS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9} U {/, ?, $} U {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9} U {:, @, dot, +, /, ?, $} = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, @, dot, +, /, ?, $}
25. FOLLOW(DIGITS) = FOLLOW(PORT) U FOLLOW(MOREDIGITS) = {/, ?, $}
26. FOLLOW(MOREDIGITS) = FOLLOW(DIGITS) = {/, ?, $}
27. FOLLOW(ALPHA) = FOLLOW(XALPHA) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, @, dot, +, /, ?, $}
28. FOLLOW(DIGIT) = FOLLOW(XALPHA) U FIRST(MOREDIGITS) / {ε} U FOLLOW(DIGITS) = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, @, dot, +, /, ?, $} U {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} U {/, ?, $} = {a, b, c, d, e, f, g, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, @, dot, +, /, ?, $}

V prípadoch zvýraznených žltou sme sledovali výskyt FIRST FOLLOW konfliktov, avšak žiaden nenastal. Keďže nie je prítomný žiaden FIRST FIRST ani FIRST FOLLOW konflikt, môžeme konštatovať, že gramatika je LL(1).

Sumarizovanú tabuľku množín FIRST a FOLLOW uvádzame nižšie. Zástupný symbol dot, ktorý sme v úprave zápisu zaviedli a nahrádzal terminál . (bodka), bol spätne nahradený.

Neterminál	First	Follow
URL	http:// ftp:// telnet:// mailto::	$
HTTPADDRESS	http://	$
SUFPATHSEARCH	/ ? ε	$
SUFSEARCH	? ε	$
FTPADDRESS	ftp://	$
TELNETADDRESS	telnet://	$
MAILTOADDRESS	mailto::	$
LOGIN	a b c d e f g 0 1 2 3 4 5 6 7 8 9	/ $
SUFPASSHOST	@ :	/ $
HOSTPORT	a b c d e f g 0 1 2 3 4 5 6 7 8 9	/ ? $
SUFPORT	: ε	/ ? $
DOTHOSTNAME	. ε	: / ? $
HOSTNAME	a b c d e f g 0 1 2 3 4 5 6 7 8 9	: / ? $
PORT	0 1 2 3 4 5 6 7 8 9	/ ? $
SLASHPATH	/ ε	? $
PATH	/ a b c d e f g 0 1 2 3 4 5 6 7 8 9 ε	? $
PLUSSEARCH	+ ε	$
SEARCH	a b c d e f g 0 1 2 3 4 5 6 7 8 9	$
USER	a b c d e f g 0 1 2 3 4 5 6 7 8 9	@
PASSWORD	a b c d e f g 0 1 2 3 4 5 6 7 8 9	/ ?
SEGMENT	a b c d e f g 0 1 2 3 4 5 6 7 8 9 ε	/ ? $
MOREXALPHAS	a b c d e f g 0 1 2 3 4 5 6 7 8 9 ε	: @ . + / ? $
XALPHAS	a b c d e f g 0 1 2 3 4 5 6 7 8 9	: @ . + / ? $
XALPHA	a b c d e f g 0 1 2 3 4 5 6 7 8 9	a b c d e f g 0 1 2 3 4 5 6 7 8 9 : @ . + / ? $
MOREDIGITS	0 1 2 3 4 5 6 7 8 9 ε	/ ? $
DIGITS	0 1 2 3 4 5 6 7 8 9	/ ? $
ALPHA	a b c d e f g	a b c d e f g 0 1 2 3 4 5 6 7 8 9 : @ . + / ? $
DIGIT	0 1 2 3 4 5 6 7 8 9	a b c d e f g 0 1 2 3 4 5 6 7 8 9 : @ . + / ? $

Tabuľka prechodov

Tabuľka obsahuje všetky pravidlá, ktoré sme očíslovali a vypísali podseba. V zadaní sa nachádza pravidlo ***alpha → "A" | .. | "Z" | "a" | .. | "z".*** kde sme pre zjednodušenie rozsahu tabuľky používali len prvých 7 písmen aj kvôli tomu, že sa na ne aplikuje rovnaké pravidlo.

Očíslované pravidlá:

1. URL -> HTTPADDRESS.
2. URL -> FTPADDRESS.
3. URL -> TELNETADDRESS.
4. URL -> MAILTOADDRESS.
5. HTTPADDRESS -> http:// HOSTPORT SUFPATHSEARCH.
6. SUFPATHSEARCH -> / PATH SUFSEARCH.
7. SUFPATHSEARCH -> ? SEARCH.
8. SUFPATHSEARCH -> ε.
9. SUFSEARCH -> ? SEARCH.
10. SUFSEARCH -> ε.
11. FTPADDRESS -> ftp:// LOGIN / PATH.
12. TELNETADDRESS -> telnet:// LOGIN.
13. MAILTOADDRESS -> mailto:: XALPHAS @ HOSTNAME.
14. LOGIN -> USER SUFPASSHOST.
15. SUFPASSHOST -> @ HOSTPORT.
16. SUFPASSHOST -> : PASSWORD @ HOSTPORT.
17. HOSTPORT -> HOSTNAME SUFPORT.
18. SUFPORT -> : PORT.
19. SUFPORT -> ε.
20. HOSTNAME -> XALPHAS DOTHOSTNAME.
21. DOTHOSTNAME -> dot HOSTNAME.
22. DOTHOSTNAME -> ε.
23. PORT -> DIGITS.
24. PATH -> SEGMENT SLASHPATH.
25. SLASHPATH -> / PATH.
26. SLASHPATH -> ε.
27. SEARCH -> XALPHAS PLUSSEARCH.
28. PLUSSEARCH -> + SEARCH.
29. PLUSSEARCH -> ε.
30. USER -> XALPHAS.
31. PASSWORD -> XALPHAS.
32. SEGMENT -> XALPHA SEGMENT.
33. SEGMENT -> ε.
34. XALPHAS -> XALPHA MOREXALPHAS.
35. MOREXALPHAS -> XALPHAS.
36. MOREXALPHAS -> ε.
37. XALPHA -> ALPHA.
38. XALPHA -> DIGIT.
39. DIGITS -> DIGIT MOREDIGITS.
40. MOREDIGITS -> DIGITS.
41. MOREDIGITS -> ε.
42. ALPHA -> a | b | c | d | e | f | g.
43. DIGIT -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.

NETERMINÁL/ TERMINÁL	**a -> z, A -> Z**	**0 -> 9**	**http:// /**	**ftp:// /**	**telnet ://**	**mailto::**	**:**	**@**	**.**	**+**	**/**	**?**	**$**
URL			1	2	3	4							
HTTPADDRESS			5										
SUFPATHSEARCH											6	7	8
SUFSEARCH												9	10
FTPADDRESS				11									
TELNETADDRESS					12								
MAILTOADDRESS						13							
LOGIN	14	14											
SUFPASSHOST							16	15					
HOSTPORT	17	17											
SUFPORT							18				19	19	19
HOSTNAME	20	20											
DOTHOSTNAME							22		21		22	22	22
PORT		23											
PATH	24	24									24	24	24
SLASHPATH											25	26	26
SEARCH	27	27											
PLUSSEARCH										28			29
USER	30	30											
PASSWORD	31	31											
SEGMENT	32	32									33	33	33
XALPHAS	34	34											
MOREXALPHAS	35	35					36	36	36	36	36	36	36
XALPHA	37	38											
DIGITS		39											
MOREDIGITS		40									41	41	41
ALPHA	42												
DIGIT		43											

Implementácia

Tokenizácia

Pri spracovaní vstupov začíname tokenizáciou jednotlivých reťazcov. Zo súboru načítavame vstupy riadok po riadku. Každý načítaný reťazec analyzujeme podľa tokenizera, ktorý rozpoznáva niekoľko tokenov.

Pre uľahčenie parsovania vstupu sme sa rozhodli ho tokenizovať pomocou regulárnych výrazov:

- URLADDRESS: Označuje URL adresu. Regulárny výraz (http://|ftp://|telnet://|mailto::) hľadá výskyt týchto častí na začiatku vstupného reťazca.
- SYMBOL: Označuje symbol. Regulárny výraz [/?:@.\+] hľadá výskyt akýchkoľvek z týchto znakov na začiatku vstupného reťazca.
- SINGLEDIGIT: Označuje jednociferné číslo. Regulárny výraz \d hľadá výskyt jednej číslice na začiatku vstupného reťazca.
- SINGLELETTER: Označuje jedno písmeno. Regulárny výraz [a-zA-Z] hľadá výskyt jedného písmena (veľkého alebo malého) na začiatku vstupného reťazca.

Pri tokenizácii si jednotlivé tokeny ukladáme vo formáte (TOKEN_TYPE,VALUE).

Návod na použitie

Program je implementovaný ako konzolová aplikácia v jazyku Python. Je ho možné spustiť príkazom `python main.py`. V takomto prípade sa spustí v interaktívnom režime a čaká na vstup od používateľa. Ak chce používateľ čítať vstupy zo súboru, je možné program spustiť ako `python main.py cesta`, pričom potom sekvenčne číta vstupy zo súboru, kde každý riadok považuje za jeden vstup.

Program vypysuje kroky step by step, vrátane aktuálneho stavu zásobníka aj aktuálneho zoznamu nespracovaných tokenov. Prípade, že si tento výstup neželáte zobraziť, je ho možno stlmiť prepínačom -s.

Program podporuje tieto prepínače:

- -t, --print_tokens: Vypíše tokeny.
- -r, --print_rules: Vypíše pravidlá.
- -p, --print_parsing_table: Vypíše tabuľku syntaktickej analýzy.
- -n, --print_nonterminals: Vypíše nonterminály.
- -m, --print_terminals: Vypíše terminály.
- -s, --silent_mode: Spustí program v tichom režime.
- -f, --file: Cesta k súboru. Ak nie je špecifikovaná, program beží v interaktívnom režime.

Príklad spustenia programu so všetkými prepínačmi:

```
python script.py -t -r -p -n -m -s -f input.txt
```

Spracovanie

V kóde je definovaná tabuľka prechodov a tabuľka prechodov. Každé pravidlo má svoje číslo a každé číslo definuje práve jedno pravidlo. Vstup sa číta po riadkoch zo súboru, alebo keď je program spustený v interaktívnom móde, tak po riadkoch ako používateľský vstup z konzoly.

Program je postavený na myšlienke zásobníka, kedy sa overuje vrch zásobníka so štartovacím symbolom. V zásobníku je vstupný symbol. Vstup je akceptovaný v prípade, že pri jeho spracovaní nenastala žiadna chyba, v zásobníku sa nachádza len štartovací symbol a celú vstup je prečítaný. V opačnom prípade vstup nie je akceptovaný a tým pádom teda nepatrí do jazyka.

Zotavenie

Pokiaľ počas spracovania vstupu nastane chyba, program vypíše, po ktorú časť vstupu sa dostal, kým nastala. Následne je možné vizuálne porovnať pôvodný vstup so spracovanou časťou. Časť, ktorá sa nespracovala, je chybná a porušuje pravidlá jazyka. Používateľ má teda možnosť vidieť, čo chybu spôsobuje.

Výstup

Výstup iba s výsledkom:

```
H:\Môj disk\5 SJ\5 SJ Zadania\7.5 - Projekt\Solution>python main.py -f input.txt -s
Reading from file input.txt
************************************************
Input URL: http://s.j

Success - valid URL

Accepted tokens:
http://s.j
************************************************
Input URL: mailto::s@j

Success - valid URL

Accepted tokens:
mailto::s@j

H:\Môj disk\5 SJ\5 SJ Zadania\7.5 - Projekt\Solution>
```

Časť výstupu krok po kroku:

```
Stack: ['-START-', 'DOTHOSTNAME', 'MOREXALPHAS']
Tokens: []
Current state: MOREXALPHAS
Current token: $
Applying rule 36: ['ε']

Step: 16
Stack: ['-START-', 'DOTHOSTNAME', 'ε']
Tokens: []
Current state: ε
Current token: $
Epsilon detected, skipping...
Current state 16: DOTHOSTNAME
Applying rule 22: ['ε']

Step: 17
Stack: ['-START-', 'ε']
Tokens: []
Current state: ε
Current token: $
Epsilon detected, skipping...
Current state 17: -START-

Success - valid URL

Accepted tokens:
mailto::s@j

H:\Môj disk\5 SJ\5 SJ Zadania\7.5 - Projekt\Solution>
```

Záver

Nami navrhnuté riešenie je schopné úspešne rozpoznať rôzne typy URL, podľa definovaných pravidiel zo zadania.

Odhalili sme konflikt v zadanej gramatike. Po konzultácii sme si upravili jedno pravidlo, tak aby sme nemali žiaden konflikt. Následne sme definovali, množiny FIRST a FOLLOW, pomocou ktorých sme vyplnili tabuľku prechodov. Navrhnutý parser sme implementovali v jazyku Python a odovzdávame ho spolu s touto dokumentáciou.

Na overenie nášho riešenia sme vyskúšali viacero vstupov, ako správnych, tak aj nesprávnych. V implementácii sa nachádza inteligentné konzolové rozhranie riadené prepínačmi. Používateľ má možnosť jednoducho zobraziť jednotlivé množiny ako aj sledovať celý proces syntaktickej analýzy s následným finálnym vyhodnotením.