Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Syntaktický analyzátor

Simple URL

**Autori**: Michal Zajic, Tímea Nagy

**Predmet**: Softvérové jazyky

**Cvičenie**: Streda 8:00

**Cvičiaci**: doc. Ing Michal Kompan, PhD.

**Obsah**

[1. Analýza a návrh 1](#_Toc26255756)

[1.1. Príklady viet z jazyka 1](#_Toc26255757)

[1.1.1. Korektné príklady 1](#_Toc26255758)

[1.1.2. Nekorektné príklady 1](#_Toc26255759)

[1.2. Prepis gramatiky z BNF do gramatických pravidiel s alternatívami 2](#_Toc26255760)

[1.2.1. Gramatika v BNF 2](#_Toc26255761)

[1.2.2. Pravidlá s alternatívami 2](#_Toc26255762)

[1.3. Transformácia gramatiky do LL(1) 4](#_Toc26255763)

[1.4. Overenie LL(1) 7](#_Toc26255764)

[1.4.1. First1 7](#_Toc26255765)

[1.4.2. Follow1 7](#_Toc26255766)

[1.5. Tabuľka prechodov 9](#_Toc26255767)

[2. Implementácia 11](#_Toc26255768)

[2.1. Používateľské rozhranie 11](#_Toc26255769)

[2.2. Testy 13](#_Toc26255770)

[2.3. Tabuľka a pravidlá 14](#_Toc26255771)

[2.4. Postup implementácie 14](#_Toc26255772)

[2.5. Zotavenie z chýb 15](#_Toc26255773)

[2.6. Návod na spustenie 15](#_Toc26255774)

# Analýza a návrh

**Podiel práce**:

* Tímea Nagy – 50%
* Michal Zajic – 50%

## Príklady viet z jazyka

Pre analyzátor sme vytvorili testy a v nich sme snažili pokryť všetky typy URL adries, ktoré patria do jazyka a aj niekoľko chybných. V tejto časti sme vybrali niektoré z nich.

## Korektné príklady

**Príklad 1:** http://hostname:port/path?search

Napr. http://Youtube.com.foo:48484/Watch/Video?123456+foo+bar

**Príklad 2:** ftp://user:password@hostname:port/path

Napr. ftp://foo:fooBAR123456@foo:123456/foo/bar/foo/

**Príklad 3:** telnet://user:password@hostname:port

Napr. telnet://foo:123456@Foo:123456

**Príklad 4:** mailto::xalphas@xalphas.xalphas

Napr. mailto::foo@foo.BAR.foo.123

## Nekorektné príklady

**Príklad 1**: nesprávne poradie search a path

Napr. http://foo.com?foo/bar

**Príklad 2**: nesprávna ftp adresa

Napr. fpt://a@b/

**Príklad 3**: nesprávny port (port môže byť len číselný)

Napr. telnet://foo@foo:foo

**Príklad 4**: chýbajúci xalphas

Napr. mailto::

## Prepis gramatiky z BNF do gramatických pravidiel s alternatívami

## Gramatika v BNF

url := httpaddress | ftpaddress | telnetaddress | mailtoaddress

httpaddress := „http://“ hostport [ / path ] [ ? search ]

ftpaddress := „ftp://“ login / path

telnetaddress := „telnet://“ login

mailtoaddress := „mailto::“ xalphas @ hostname

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

hostport := hostname [ : port ]

hostname := xalphas { . xalphas }

port := digits

path := segment { / segment }

search := xalphas { + xalphas }

user := xalphas

password := xalphas

segment := { xalpha }

xalphas := xalpha { xalpha }

xalpha := alpha | digit

digits := digit {digit}

alpha := „A“ | .. | „Z“ | „a“ | .. | „z“

digit := „0“ | .. | „9“

## Pravidlá s alternatívami

Zadefinovanie neterminálov:

A = url

B = httpaddress

C = ftpaddress

D = telnetaddress

E = mailtoaddress

F = login

G = hostport

H = hostname

I = port

J = path

K = search

L = user

M = password

N = segment

O = xalphas

P = xalpha

R = digits

S = alpha

T = digit

Prepis gramatiky v BNF pomocou neterminálov definovaných vyššie:

A → B | C | D | E

B → „http://“ G | „http://“ G / J | „http://“ G / J ? K | „http://“ G ? K

C → „ftp://“ F / J

D → „telnet://“ F

E → „mailto::“ O @ H

F → L : M @ G | L @ G | G

G → H | H : I

H → O H`

H` → . O H` | ε

I → R

J → N J`

J` → / N J` | ε

K → O K`

K` → + O K` | ε

L → O

M → O

N → PN | ε

O → P O`

O` → P O` | ε

P → S | T

R → T R`

R` → T R` | ε

S → [A-Za-z]

T → [0-9]

Červenou farbou sú vyznačené miesta, kde gramatika nesĺpňa podmienky LL(1). Pri vyznačených miestach nastáva First-First konflikt.

## Transformácia gramatiky do LL(1)

Na odstránenie First-First konfliktov, ktoré sme opísali vyššie sme využili faktorizáciu:

A → B | C | D | E

B → „http://“ G B`

B` → / J B`` | ? K | ε

B`` → ? K | ε

C → „ftp://“ F / J

D → „telnet://“ F

E → „mailto::“ O @ H

F → F` G

F` → L F`` | ε

F`` → @ | : M @

G → H G`

G` → : I | ε

H → O H`

H` → . O H` | ε

I → R

J → N J`

J` → / N J` | ε

K → O K`

K` → + O K` | ε

L → O

M → O

N → PN | ε

O → P O`

O` → P O` | ε

P → S | T

R → T R`

R` → T R` | ε

S → [A-Za-z]

T → [0-9]

Červenou farbou je vyznačené miesto, kde gramatika naďalej nesĺpňa podmienky LL(1). Pri vyznačenom mieste nastáva First-First konflikt kvôli číslam a veľkým a malým písmenám. Výpočet First pre vybrané neterminálu pre znázornenie First-First konfliktu:

First(F) = First(F`) \ ε U First(G) = {[0-9], [A-Za-z] } U { [0-9], [A-Za-z]}

First(F`) = First(L) U {ε} = {[0-9], [A-Za-z]} U {ε} ={[0-9], [A-Za-z], ε}

First(G) = First(H) = {[0-9], [A-Za-z]}

First(L) = First(O) = {[0-9], [A-Za-z]}

First(H) = First(O) = {[0-9], [A-Za-z]}

First(O) = First(P) = {[0-9], [A-Za-z]}

First(P) = First(S) \ ε U First(T) = {[0-9], [A-Za-z]}

First(T) = {[0-9]}

First(S) = {[A-Za-z]}

Po viacerých pokusoch odstránenia First-First konfliktu, po lepšej analýze daného konfliktu a po konzultácií s cvičiacim sa prišlo na to, že hore uvedenú gramatiku nie je možné tranformovať na typ LL(1) prostriedkami obsiahnutými v predmete.

Problém nastáva pri riadkoch:

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

hostport := hostname [ : port ]

Ako príklad si zoberieme vstup, ktorý obsahuje „usera“ a vstup, ktorý neobsahuje „usera“, ale len „hostname. “User” aj “hostname” sa môžu skladať z čísel a z malých a z veľkých písmen. Gramatika nešpecifikuje spôsob, ako zistiť, že aktuálne spracovávaný znak patrí pre „usera“ alebo pre „hostname“.

Na riešenie tohto problému sme zaviedli pravidlo, že sa „user“ musí nachádzať v URL adrese práve raz.

Problémové riadky po zavedení pravidla vyzerajú nasledovne:

login := user [ : password ] @ hostport

hostport := hostname [ : port ]

Prepis zmenenej gramatiky v BNF pomocou neterminálov definovaných vyššie (zmeny oproti predošlej gramatike sú vyznačené farebne):

A → B | C | D | E

B → „http://“ G B`

B` → / J B`` | ? K | ε

B`` → ? K | ε

C → „ftp://“ F / J

D → „telnet://“ F

E → „mailto::“ O @ H

F → L F` @ G

F` → : M | ε

G → H G`

G` → : I | ε

H → O H`

H` → . O H` | ε

I → R

J → N J`

J` → / N J` | ε

K → O K`

K` → + O K` | ε

L → O

M → O

N → PN | ε

O → P O`

O` → P O` | ε

P → S | T

R → T R`

R` → T R` | ε

S → [A-Za-z]

T → [0-9]

V tejto gramatike už neboli nájdené žiadne miesta, ktoré by nespĺňali podmienky LL(1).

Rozhodli sme sa však gramatiku ďalej upraviť, zjednodušiť. Ako prvé sme odstránili zbytočné neterminály s jedným pravidlom, ktorý obsahuje jeden neterminál. To sú neterminály I, L a M. Všetky výskyty týchto neterminálov sme nahradili ich pravidlom. Gramatika po tejto zmene vyzerá nasledovne (zmeny oproti predošlej gramatike sú vyznačené farebne):

A → B | C | D | E

B → „http://“ G B`

B` → / J B`` | ? K | ε

B`` → ? K | ε

C → „ftp://“ F / J

D → „telnet://“ F

E → „mailto::“ O @ H

F → O F` @ G

F` → : O | ε

G → H G`

G` → : R | ε

H → O H`

H` → . O H` | ε

J → N J`

J` → / N J` | ε

K → O K`

K` → + O K` | ε

N → PN | ε

O → P O`

O` → P O` | ε

P → S | T

R → T R`

R` → T R` | ε

S → [A-Za-z]

T → [0-9]

Následne sme gramatiku prepísali na krajší tvar, kde sme zmenili len pomenovanie neterminálov tak, aby išli v abecednom poradí a neobsahovali apostrof:

A → „http://“ F B | „ftp://“ D / J | „telnet://“ D | „mailto::“ O @ H

B → / J C | ? L | ε

C → ? L | ε

D → O E @ F

E → : O | ε

F → H G

G → : R | ε

H → O I

I → . O I | ε

J → N K

K → / N K | ε

L → O M

M → + O M | ε

N → Q N | ε

O → Q P

P → Q P | ε

Q → U | T

R → T S

S → T S | ε

T → [0-9]

U → [A-Za-z]

## Overenie LL(1)

## First1

First(U) = {[A-Za-z]}

First(T) = {[0-9]}

First(S) = First(T) U {ε} = {[0-9]} U {ε} = {[0-9], ε}

First(R) = First(T) = {[0-9]}

First(Q) = First(T) U First(U) ={[0-9]} U {[A-Za-z]} = {[0-9], [A-Za-z]}

First(P) = First(Q) U {ε} = {[0-9], [A-Za-z]} U {ε} = {[0-9], [A-Za-z], ε}

First(O) = First(Q) ={[0-9], [A-Za-z]}

First(N) = First(Q) U {ε} = {[0-9], [A-Za-z]} U {ε} = {[0-9], [A-Za-z], ε}

First(M) = {+, ε}

First(L) = First(O) = {[0-9], [A-Za-z]}

First(K) = {/, ε}

First(J) = First(N) \ ε U First(K) ={[0-9], [A-Za-z]} U {/, ε} = {[0-9], [A-Za-z], /, ε}

First(I) = {., ε}

First(H) = First(O) ={[0-9], [A-Za-z]}

First(G) = {:, ε}

First(F) = First(H) ={[0-9], [A-Za-z]}

First(E) = {:, ε}

First(D) = First(O) = {[0-9], [A-Za-z]}

First(C) = {?, ε}

First(B) = {/, ? , ε}

First(A) = {„http://“, „ftp://“, „telnet://“, „mailto::“}

## Follow1

Follow(A) = {$}

Follow(B) = Follow(A) = {$}

Follow(C) = Follow(B) = {$}

Follow(D) = Follow(A) U {/} = {$} U {/} = {$, /}

Follow(E) = {@}

Follow(F) = First(B) \ ε U Follow(A) U Follow(D) = {/, ? } U {$} U {$} = {/, ?, $}

Follow(G) = Follow(F) = {/, ?, $}

Follow(H) = Follow(A) U First(G) \ ε U Follow(F) = {$} U {:} U {/, ?, $} = {$, :, /, ?}

Follow(I) = Follow(H) U Follow(I) = {$, :, /, ?}

Follow(J) = Follow(A) U First(C) \ ε U Follow(B) = {$} U {?} U {$} = {$, ?}

Follow(K) = Follow(J) U Follow(K) = {$, ?}

Follow(L) = Follow(B) U Follow(C) = {$} U {$} = {$}

Follow(M) = Follow(L) U Follow(M) = {$}

Follow(N) = First(K) \ ε U Follow(J) U Follow(K) U Follow(N) = {/} U {$, ?} U {$, ?}

= {/, $, ?}

Follow(O) = {@} U First(E) \ ε U Follow(D) U Follow(E) U First(I) \ ε U Follow(H) U Follow(I) U First(M) \ ε U Follow (L) U Follow (M)

= {@} U {:} U {$, /} U {@} U {.} U {$, :, /, ?} U {$, :, /, ?} U {+} U {$} U {$} = {@, :, $, /, ., ?, +}

Follow(P) = Follow(O) U Follow(P) = {@, :, $, /, ., ?, +}

Follow(Q) = First(N) \ ε U Follow(N) U First(P) \ ε U Follow(O) U Follow(P)

= {[0-9], [A-Za-z]} U {/, $, ?} U {[0-9], [A-Za-z]} U {@, :, $, /, ., ?, +} U {@, :, $, /, ., ?, +} ={[0-9], [A-Za-z], /, $, ?, @, :, ., +}

Follow(R) = Follow(G) = {/, ?, $}

Follow(S) = Follow(R) U Follow(S) = {/, ?, $}

Follow(T) = Follow(Q) U First(S) \ ε U Follow(R) U Follow(S) = {[0-9], [A-Za-z], /, $, ?, @, :, ., +} U {[0-9]} U {/, ?, $} U {/, ?, $}

={[0-9], [A-Za-z], /, $, ?, @, :, ., +}

Follow(U) = Follow(Q) ={[0-9], [A-Za-z], /, $, ?, @, :, ., +}

Po analýze množín FIRST1 a FOLLOW1 sme sa presvedčili, že gramatika je LL(1).

## Tabuľka prechodov

|  |  |
| --- | --- |
| Pravidlo A  1 : http://FB  2 : ftp://D/J  3 : telnet://D  4 : mailto::O@H  Pravidlo B  5 : /JC  6 : ?L  7 : epsilon  Pravidlo C  8 : ?L  9 : epsilon  Pravidlo D  10 : OE@F  Pravidlo E  11 : :O  12 : epsilon  Pravidlo F  13 : HG  Pravidlo G  14 : :R  15 : epsilon  Pravidlo H  16 : OI  Pravidlo I  17 : .OI  18 : epsilon  Pravidlo J  19 : NK | Pravidlo K  20 : /NK  21 : epsilon  Pravidlo L  22 : OM  Pravidlo M  23 : +OM  24 : epsilon  Pravidlo N  25 : QN  26 : epsilon  Pravidlo O  27 : QP  Pravidlo P  28 : QP  29 : epsilon  Pravidlo Q  30 : U  31 : T  Pravidlo R  32 : TS  Pravidlo S  33 : TS  34 : epsilon  Pravidlo T  35 : [0-9]  Pravidlo U  36 : [A-Za-z] |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **[A-Za-z]** | **[0-9]** | **+** | **/** | **.** | **:** | **@** | **http://** | **ftp://** | **telnet://** | **mailto::** | **?** | **$** |
| **A** | error | error | error | error | error | error | error | 1 | 2 | 3 | 4 | error | error |
| **B** | error | error | error | 5 | error | error | error | error | error | error | error | 6 | 7 |
| **C** | error | error | error | error | error | error | error | error | error | error | error | 8 | 9 |
| **D** | 10 | 10 | error | error | error | error | error | error | error | error | error | error | error |
| **E** | error | error | error | error | error | 11 | 12 | error | error | error | error | error | error |
| **F** | 13 | 13 | error | error | error | error | error | error | error | error | error | error | error |
| **G** | error | error | error | 15 | error | 14 | error | error | error | error | error | 15 | 15 |
| **H** | 16 | 16 | error | error | error | error | error | error | error | error | error | error | error |
| **I** | error | error | error | 18 | 17 | 18 | error | error | error | error | error | 18 | 18 |
| **J** | 19 | 19 | error | 19 | error | error | error | error | error | error | error | 19 | 19 |
| **K** | error | error | error | 20 | error | error | error | error | error | error | error | 21 | 21 |
| **L** | 22 | 22 | error | error | error | error | error | error | error | error | error | error | error |
| **M** | error | error | 23 | error | error | error | error | error | error | error | error | error | 24 |
| **N** | 25 | 25 | error | 26 | error | error | error | error | error | error | error | 26 | 26 |
| **O** | 27 | 27 | error | error | error | error | error | error | error | error | error | error | error |
| **P** | 28 | 28 | 29 | 29 | 29 | 29 | 29 | error | error | error | error | 29 | 29 |
| **Q** | 30 | 31 | error | error | error | error | error | error | error | error | error | error | error |
| **R** | error | 32 | error | error | error | error | error | error | error | error | error | error | error |
| **S** | error | 33 | error | 34 | error | error | error | error | error | error | error | 34 | 34 |
| **T** | error | 35 | error | error | error | error | error | error | error | error | error | error | error |
| **U** | 36 | error | error | error | error | error | error | error | error | error | error | error | error |

# Implementácia

**Podiel práce**:

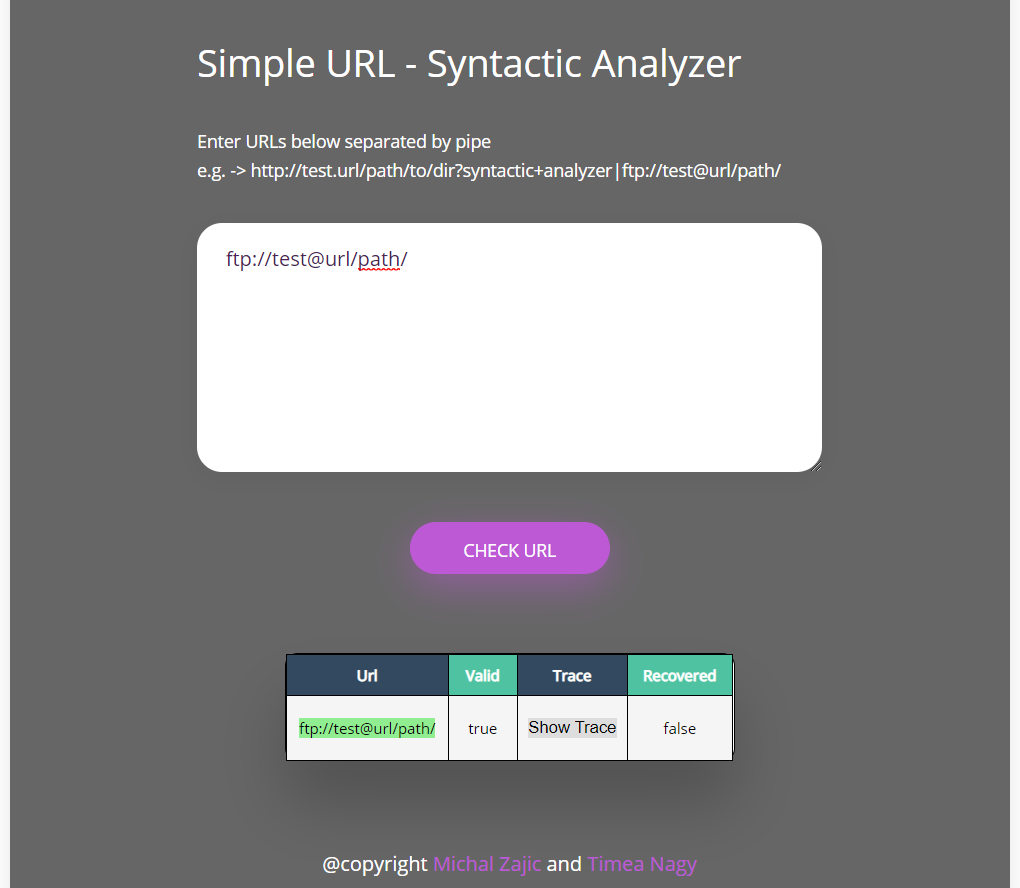
* Tímea Nagy – 50%
* Michal Zajic – 50%

Implementovali sme tabuľkou riadený syntaktický analyzátor v jazyku JavaScript spolu s používateľským rozhraním a testami. Analyzátor rozpoznáva vety v jazyku simple URL.

## Používateľské rozhranie

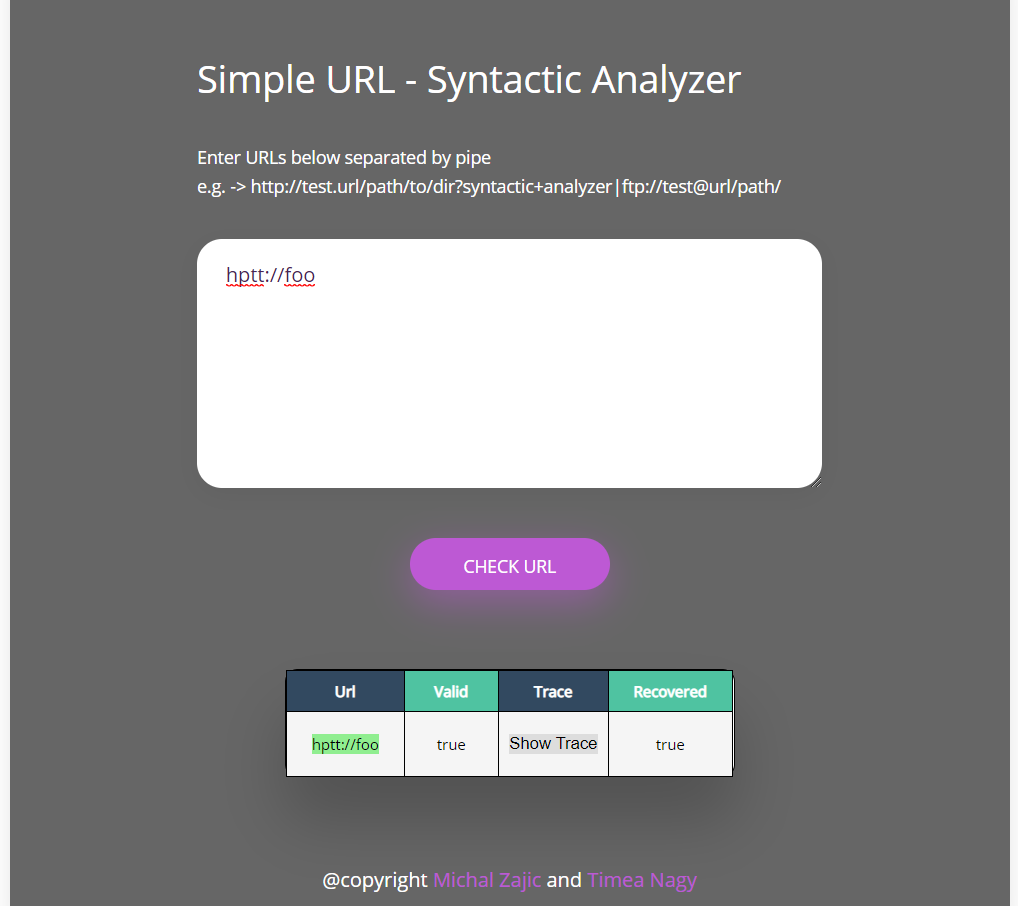
Analyzátor je možné spustiť spolu s používateľským rozhraním, ktorého základ je „index.html“ súbor. Používateľ môže zadať ľubovoľný počet URL adresies oddelených pomocou znaku pipe „|”. Po zadaní URL adresy analyzátor vypíše či je adresa platná alebo nie a či sa využilo zotavenie z chýb alebo nie. Ponúka aj možnosť zobrazenia krokov, ako bol vstup spracovaný.

Pri správnom vstupe a bez využitia zotavenia z chýb vyzerá výstup nasledovne:





Pri využití zotavenia chýb vyzerá výstup nasledovne:





## Testy

Pre analyzátor sme napísali testy tak, aby sme pokryli všetky možné URL adresy, ktoré sú platné a niekoľko neplatných adries. Nachádzajú sa v súbore „analyzer.test.js“. Testy sme rozdelili na štyri časti podľa štyroch pravidiel, ktoré sa nachádzajú pri štartovacom neterminály – URL adresy začínajúce na „http://“, „ftp://“, „telnet://“ a „mailto::“. V rámci všetkých týchto typov sme zadefinovali platné URL adresy, adresy, pri ktorých sa využilo zotavenie z chýb a chybné URL adresy. Spustiť ich je možné pomocou príkazu „npm test“.

K URL adresám máme informáciu či je adresa platná alebo nie a či bolo využité zotavenie z chýb alebo nie.

Ukážka platných URL adries:

    ["http://a", true, false],

    ["http://1", true, false],

    ["http://Z", true, false],

    ["http://5", true, false],

    ["http://foobarfoo", true, false],

    ["http://1234567", true, false],

    ["http://FOO", true, false],

    ["http://fooBARfoo123", true, false],

Ukážka URL adries, pri ktorých sa využilo zotavenie:

    ["tenet://foo:123456@foo:123456", true, true], // incorrect telnet address

    ["telnet://foo@foo:foo", true, true], // incorrect port

Ukážka neplatných URL adries:

    ["mailto::", false, false], // missing xalphas

    ["mailto::a@", false, false],  // missing hostname

    ["mailto::a", false, false],   // missing hostname and @ character

    ["mailto::a@b.", false, false],    // incorrect hostname

    ["mailto::a.foo.bar@b", false, false], // incorrect xalphas

## Tabuľka a pravidlá

Reprezentácia prechodovej tabuľky a pravidiel sa nachádza v súbore „table\_and\_rules.js“. Má formu *dictionary*. Pre každý neterminál si pamätáme terminály, ktoré vieme vygenerovať pomocou daného neterminálu a k terminálom pravidlo, ktorým musíme pokračovať, aby sme dostali daný terminál. Ak neterminál môže vygenerovať hocijaké písmená alebo čísla, v *dictionary* máme pre tieto terminály špeciálne označenie. Na základe tohto označenia analyzátor na začiatku vygeneruje *key-value* hodnoty do *dictionary* pre každé písmeno alebo číslo. Kód pre generovanie týchto hodnôt sa nachádza v súbore „table\_transform.js“.

Ukážka reprezentácia tabuľky pre neterminál S:

    "S" : {

        "number" : 33,

        "/" : 34,

        "?" : 34,

        "$" : 34

}

Ukážka reprezentácie pravidiel:

    1: "http://FB",

    2: "ftp://D/J",

    3: "telnet://D",

    4: "mailto::O@H",

    5: "/JC",

    6: "?L",

    7: '',

    8: "?L",

## Postup implementácie

Samotný proces spracovávania vstupu sa nachádza v súbore „simple\_url\_syntactic\_analyzer .js“. Algoritmus začína predspracovaním vstupu, v rámci ktorého pridáme na koniec vstupu znak „$“.

Po predspracovaní inicializujeme zásobník, t. j. pridáme štartovací neterminál.

Samotný algoritmus odvodenia prebieha ako rekurzia. Má nasledovný priebeh:

1. Vyberieme prvý znak zo zásobníka, ktorý môže byť terminál alebo neterminál a vyberieme aj prvý znak z nespracovanej časti URL adresy.
2. Ak sú tieto znaky rovnaké a nejedná sa o neterminál, tak odstraňujeme tento znak zo zásobníka a aj z URL adresy. Pri štartovacom pravidle odstraňujeme celý terminál napr. „http://“.
3. Ak sme rekurziu zavolali s regulárným výrazom napr. [A-Za-z] alebo [0-9], tak skontrolujeme či platí na prvý znak z nespracovanej časti URL adresy. Ak áno, odstránime daný znak.
4. Skontrolujeme či máme epsilonové pravidlo. Ak áno, zavoláme rekurziu znova so zvyšnými pravidlami.
5. Znova vyberieme prvý znak zo zásobníka a vyberieme aj prvý znak z nespracovanej časti URL adresy.
6. Ak narazíme na znak „$”, teda spracovali sme celú URL adresu a nemáme žiadne ďalšie pravidlá v zásobníku, adresa je validná a rekurziu skončíme.
7. Ak pre aktuálny neterminál nemáme žiadne pravidlo, s ktorým by sme sa dostali k terminálu na začiatku URL adresy, adresa nie je validná a končíme rekurziu.
8. Aplikujeme pravidlo (uložíme do zásobníka) pre aktuálny neterminál na základe terminálu, ktorý sa nachádza na začiatku nespracovanej časti URL adresy.
9. Ak pravidlo je regulárny výraz, musíme ju poslať do rekurzie.
10. Zavoláme rekurziu a celý proces začíname od znova.

## Zotavenie z chýb

Zotavenie chýb nastane, ak sa používateľ pomýli na začiatku URL adresy alebo pri porte.

URL adresa môže začínať len so štyrmi rôznymi reťazcami znakov podľa typu URL adresy - „http://“, „ftp://“, „telnet://“ a „mailto::“. Ak sa používateľ pomýli v tejto časti URL adresy, tak sa pomocou edit distance algoritmu zistí, na ktorú z tých štyroch typov sa URL adresa podobá a na základe toho zmeníme začiatok URL adresy.

Port sa môže skladať len z čísel. Chceme však uznať aj URL adresy, v ktorých sa používateľ pomýlil pri porte a namiesto čísel tam napísal aj písmená. Pridali sme nové pravidlá do tabuľky:

R → U S

S → U S

Kde R a S sú terminály, ktoré generujú port a U generuje veľké a malé písmená.

## Návod na spustenie

Program je vytvorený v jazyku JavaScript. Vyžaduje naištalovaný Node.js.

Ako prvé je potrebné nainštalovať všetky potrebné moduly, to zabezpečí príkaz „npm install“ spustený v hlavnom priečinku programu. Na spustenie testov slúži príkaz „npm test“. Program obsahuje aj používateľské rozhranie, ktoré je možné spustiť otvorením „index.html“ súboru v prehliadači. Pred tým však odporúčame zadať príkaz „npm run build“.