

Zvýrazňovanie syntaxe na základe definície meta-modelu

Matej Gagyi

Úvod

- Mam definovanú gramatiku jazyka:

```
9 public class Transition extends Declaration {
10
11     private String label;
12
13     private State source;
14
15     private State target;
16
17     @Before("trans")
18     @After(";")
19     public Transition(
20         @Token("ID") String label,
21         @Before(":")
22         @Token("ID")
23         @References(value = State.class, field = "source") String sourceLabel,
24         @Before("->")
25         @Token("ID")
26         @References(value = State.class, field = "target") String targetLabel) {
27         this.label = label;
28     }
```

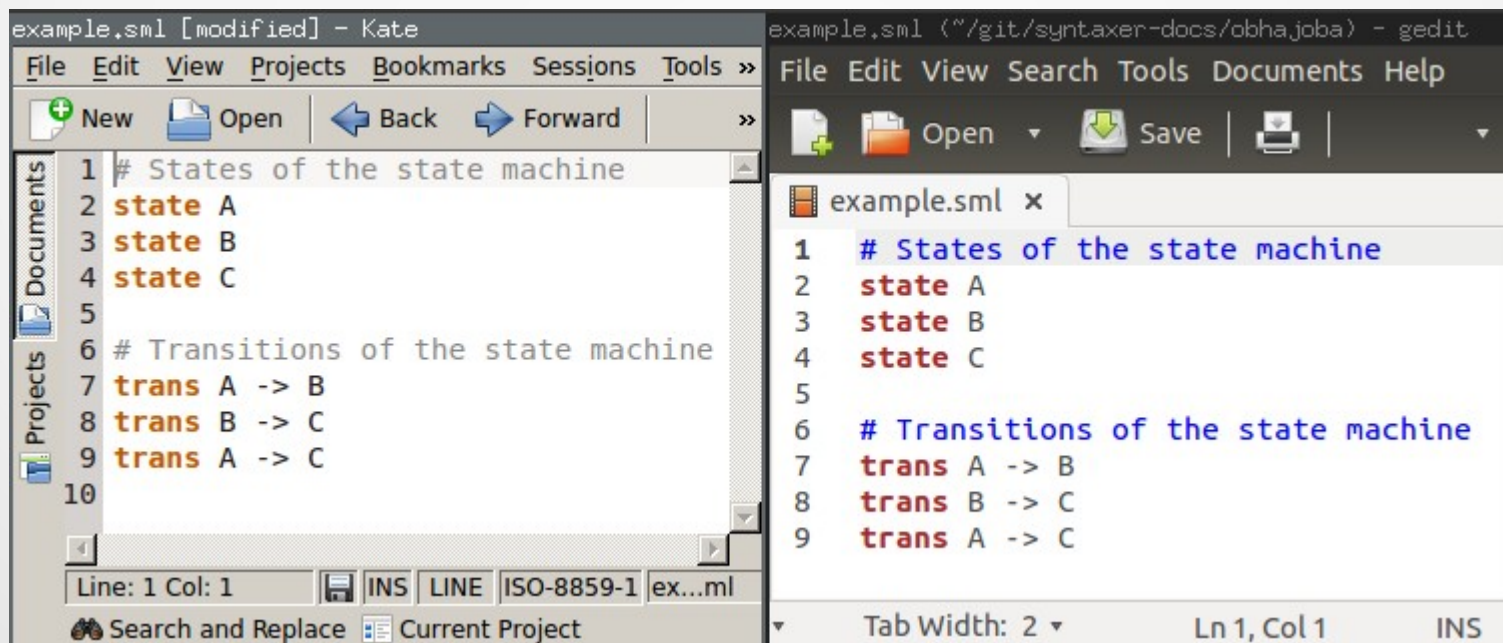
Cieľ práce

- Zdrojové kódy budú zvýraznené automaticky

```
1 # States of the state machine
2 state A
3 state B
4 state C
5
6 # Transitions of the state machine
7 trans A -> B
8 trans B -> C
9 trans A -> C
10
```

Cieľ práce

- ... a to v rozličných programoch



The image displays two side-by-side screenshots of text editors, illustrating the same code being viewed in different environments.

Left Screenshot (Kate): The window title is "example.sml [modified] - Kate". The menu bar includes File, Edit, View, Projects, Bookmarks, Sessions, and Tools. The toolbar shows icons for New, Open, Back, and Forward. The code is as follows:

```
1 # States of the state machine
2 state A
3 state B
4 state C
5
6 # Transitions of the state machine
7 trans A -> B
8 trans B -> C
9 trans A -> C
10
```

The status bar at the bottom shows "Line: 1 Col: 1", "INS", "LINE", "ISO-8859-1", and "ex...ml".

Right Screenshot (gedit): The window title is "example.sml (~/.git/syntaxer-docs/obhajoba) - gedit". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar shows icons for Open, Save, and Print. The code is as follows:

```
1 # States of the state machine
2 state A
3 state B
4 state C
5
6 # Transitions of the state machine
7 trans A -> B
8 trans B -> C
9 trans A -> C
```

The status bar at the bottom shows "Tab Width: 2", "Ln 1, Col 1", and "INS".

Agenda

1. Metodika práce

- 1. Cieľové editori

- 2. Implementačné výzvy

2. Anazylátor regulárnych výrazov

- 1. Regulárne výrazy

- 2. Deterministický Konečný Automat

3. Výsledky

Metodika práce

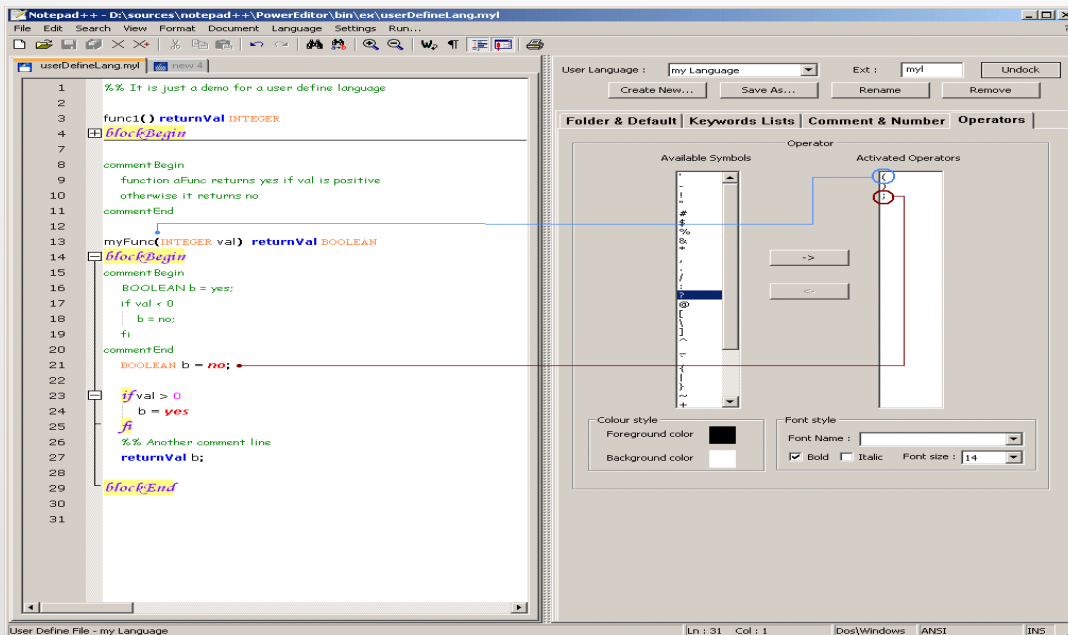
- Výber cieľových editorov
- Implementácia, riešenie problémov, implementácia
- Analýza nevyriešených problémov

Metodika práce – Cieľové editori I.

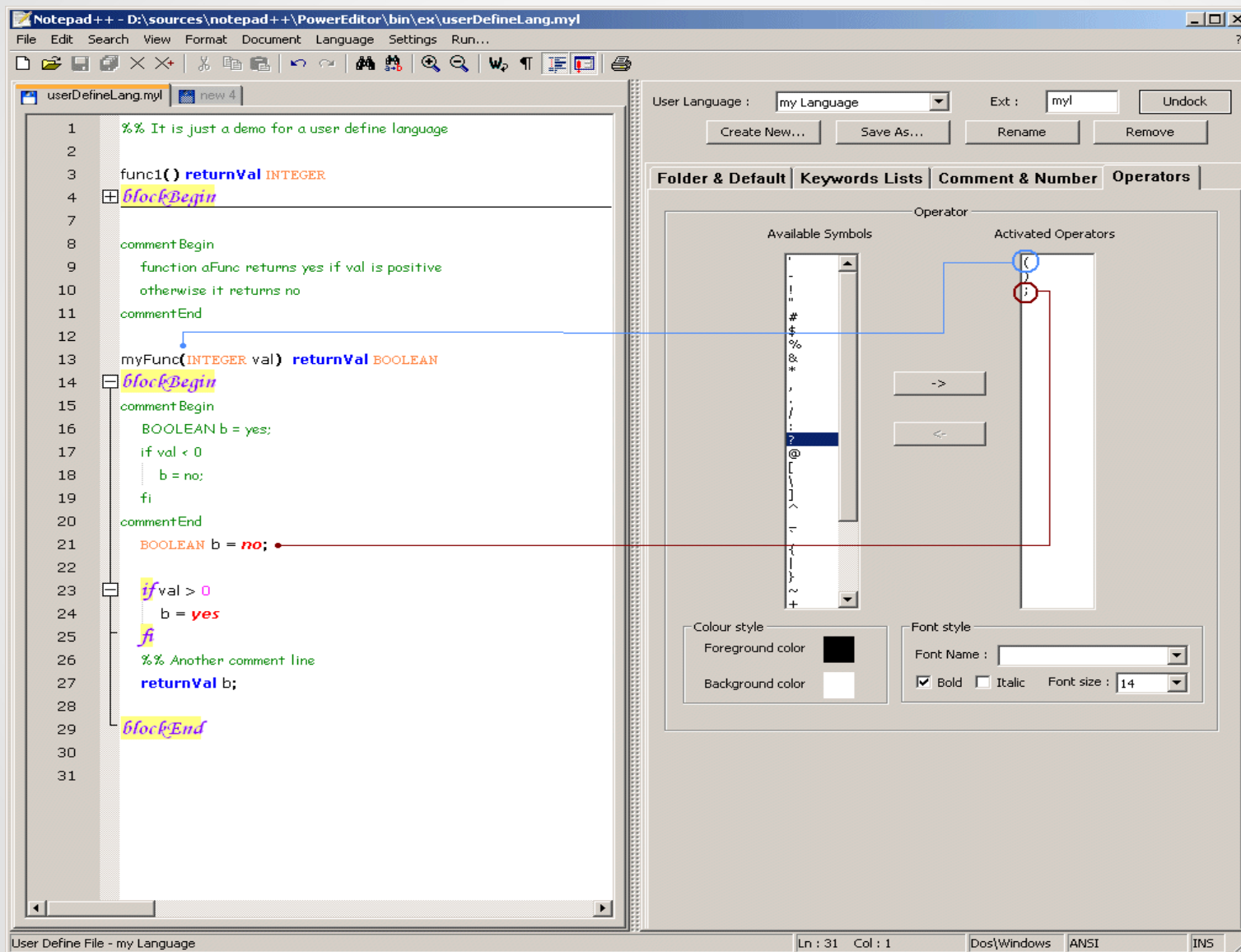
- Pôvodné očakávania:
 - **Notepad++** pre Windows
 - **Kate** a **Gedit** pre Linux
 - **Pygments** a **CodeMirror** pre Web

Metodika práce – Cieľové editori II.

- Notepad++
 - Chybajúca dokumentácia
 - Jednoduchá konfigurácia
 - Tvorba jazykov pomocou GUI



Ciel'ové editori II. - Notepad++



Metodika práce – Cieľové editori III.

- **Kate a Gedit**
 - Množstvo dokumentácie, IRC kanál
 - Deklaratívny zápis syntaxe
 - Sila

The image displays two side-by-side screenshots of text editors, Kate (left) and Gedit (right), both editing a file named 'example.sml'. The code in both editors is identical and represents a state machine definition.

Kate Editor (Left): The window title is 'example.sml [modified] - Kate'. The menu bar includes File, Edit, View, Projects, Bookmarks, Sessions, and Tools. The toolbar has buttons for New, Open, Back, and Forward. The left sidebar shows 'Documents' and 'Projects' views. The code is as follows:

```
1 # States of the state machine
2 state A
3 state B
4 state C
5
6 # Transitions of the state machine
7 trans A -> B
8 trans B -> C
9 trans A -> C
10
```

The status bar at the bottom shows 'Line: 1 Col: 1', 'INS', 'LINE', 'ISO-8859-1', and 'ex...ml'. There are also buttons for 'Search and Replace' and 'Current Project'.

Gedit Editor (Right): The window title is 'example.sml (/git/syntaxer-docs/obhajoba) - gedit'. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar has buttons for Open, Save, and a printer icon. The code is as follows:

```
1 # States of the state machine
2 state A
3 state B
4 state C
5
6 # Transitions of the state machine
7 trans A -> B
8 trans B -> C
9 trans A -> C
```

The status bar at the bottom shows 'Tab Width: 2', 'Ln 1, Col 1', and 'INS'.

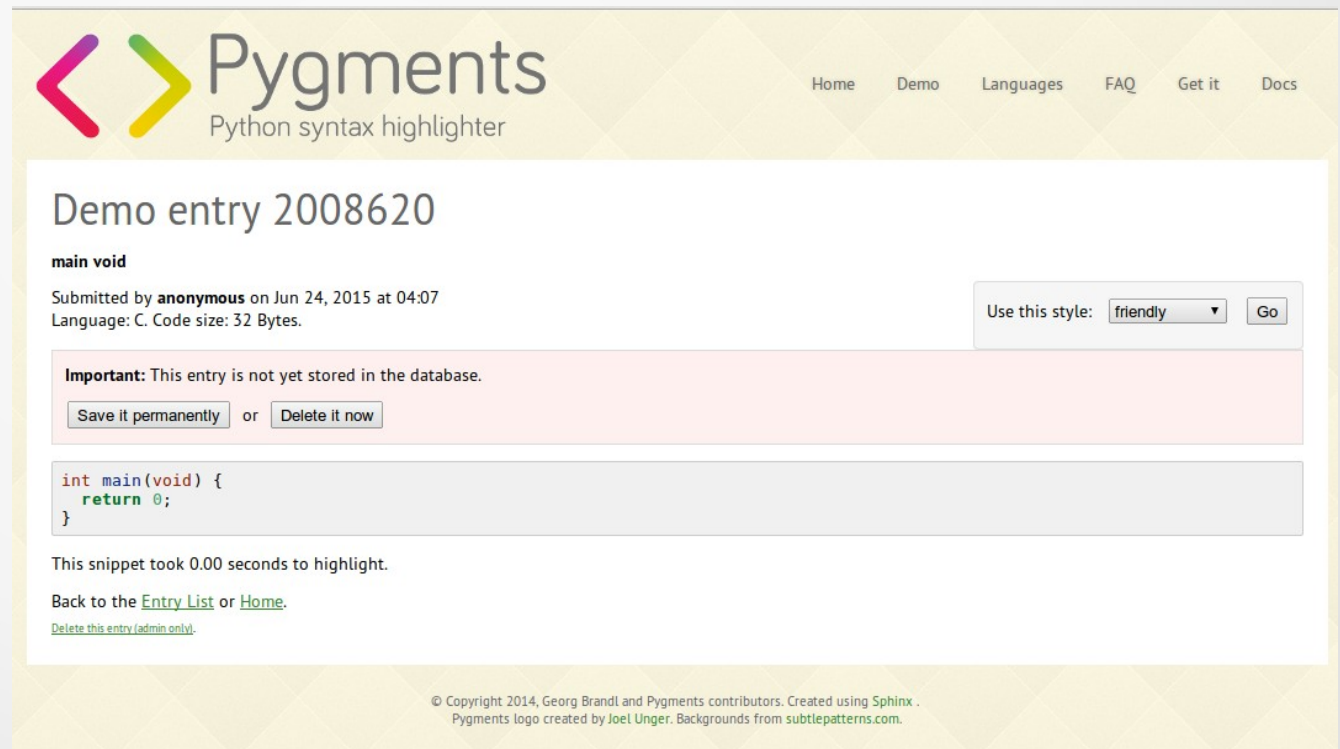
Metodika práce – Cieľové editori IV.

- **Pygments a CodeMirror**
 - Množstvo dokumentácie, mailing listy
 - Nutnosť JavaScriptu a Pythonu
 - Náročné odladovanie

Metodika práce – Cieľové editori V.

- **Pygments**

- Vyžaduje veľa zdrojového kódu
- Python
- Sila nástroja



The screenshot shows the Pygments website interface. At the top, there is a navigation bar with links: Home, Demo, Languages, FAQ, Get it, and Docs. The main heading is "Pygments" with the subtitle "Python syntax highlighter". Below this, there is a section titled "Demo entry 2008620" with the code snippet "main void". The submission information states: "Submitted by anonymous on Jun 24, 2015 at 04:07" and "Language: C. Code size: 32 Bytes." There is a dropdown menu for "Use this style:" set to "friendly" and a "Go" button. A red warning box says: "Important: This entry is not yet stored in the database." Below this, there are buttons for "Save it permanently" and "Delete it now". The code snippet is highlighted in a light gray box:

```
int main(void) {  
    return 0;  
}
```

 Below the code, it says: "This snippet took 0.00 seconds to highlight." At the bottom, there are links: "Back to the Entry List or Home." and "Delete this entry (admin only)." The footer contains copyright information: "© Copyright 2014, Georg Brandl and Pygments contributors. Created using Sphinx . Pygments logo created by Joel Unger. Backgrounds from subtlepatterns.com."

Metodika práce – Cieľové editori VI.

- **CodeMirror**

- API CodeMirror nie je kompatibilné s RE
- Obmedzenosť nástroja
- JavaScript

This is CodeMirror

```
1 <!-- Create a simple CodeMirror instance -->
2 <link rel="stylesheet" href="lib/codemirror.css">
3 <script src="lib/codemirror.js"></script>
4 <script>
5   var editor = CodeMirror.fromTextArea(myTextarea, {
6     lineNumbers: true
7   });
8 </script>
```

Other demos... ▼

Metodika práce – Implementačné výzvy I.

1. Meta-model syntaxe a API YajCo
2. Java AnnotationProcessor
3. Rozdielnosť editorov

Metodika práce – Implementačné výzvy II.

- **Meta-model syntaxe a API YajCo**
 - Anotácie YajCO: @TokenDef, @Before
 - CompilerGenerator vs. PrinterGenerator
 - API triedy language

```
9 public class Transition extends Declaration {
10
11     private String label;
12
13     private State source;
14
15     private State target;
16
17     @Before("trans")
18     @After(";")
19     public Transition(
20         @Token("ID") String label,
21         @Before(":")
22         @Token("ID")
23         @References(value = State.class, field = "source") String sourceLabel,
24         @Before("->")
25         @Token("ID")
26         @References(value = State.class, field = "target") String targetLabel) {
27         this.label = label;
28     }
```

Metodika práce – Implementačné výzvy III.

- **Java AnnotationProcessor**

- Hook javac pre spracovanie anotácií
- Napojenie na Maven

```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 2.783s
```


Metodika práce – Implementačné výzvy IV.

- **Rozdielnosť editorov**

- Deklaratívna konfigurácia vs. Imperatívny kód
- Rozdielne správanie
 - Regulárne výrazy
 - Elementy gramatiky

```
3 public class KateEditorStrategy extends GeneralEditorStrategy {
4     private static final String ID = "kate";
5     private static final String NAME = "Kate";
6     private static final String TEMPLATE = "kate.vm";
7     private static final String SUFFIX = "kate.xml";
8
9     public KateEditorStrategy() {
10         super(ID, NAME, TEMPLATE, SUFFIX);
11     }
12
13     @Override
14     protected String transformRegex(String regexp, String type) {
15         return regexp.replaceAll("\\\\n\\)?\\$?$", "\\$");
16     }
17 }
18
```

Analyzátor RE

- RE generujú jazyk reťazcov
 - “Ako rozoznáme rôzne elementy jazyka?”
 - “Ktorý RE reprezentuje komentár a ktorý operátor?”
- DFA čiastočne odpovedá tieto dopyty

Analyzátor RE – Regulárne výrazy I.

1. Syntax
2. Nevhodnosť reprezentácie

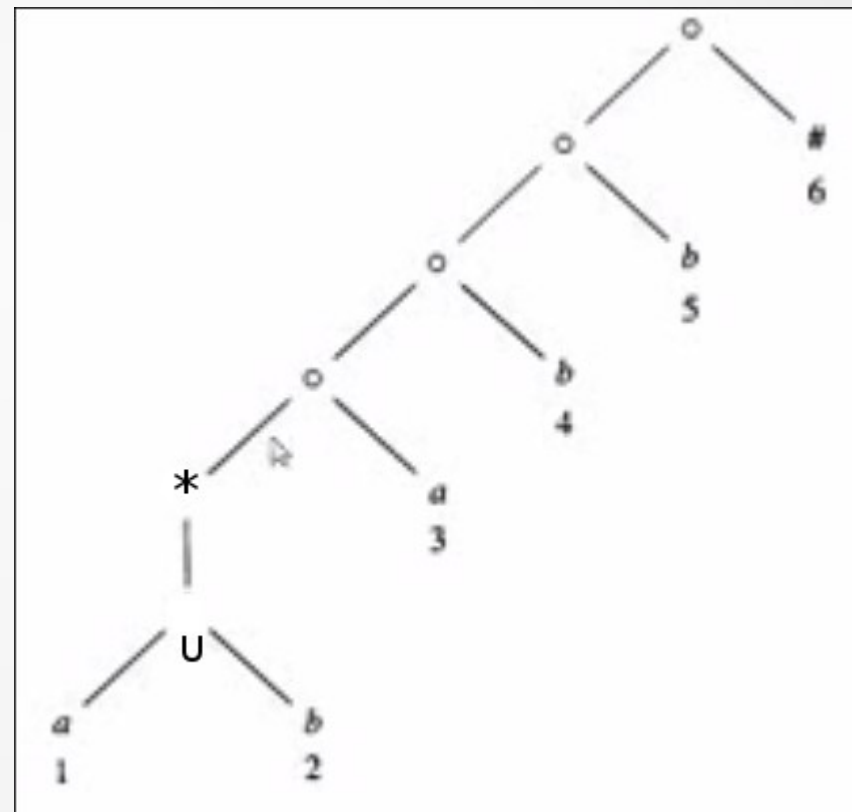
Analyzátor RE – Regulárne výrazy II.

- **Syntax**

- Java podporuje regulárne výrazy z OS UNIX
- Syntaktická analýza môže využiť zovšeobecnenia
 - Príklad: `ab+` → `abb*`

Analýzátor RE – Regulárne výrazy III.

- Syntax
 - Syntaktický strom
 - Příklad: $(a|b)^*abb\#$



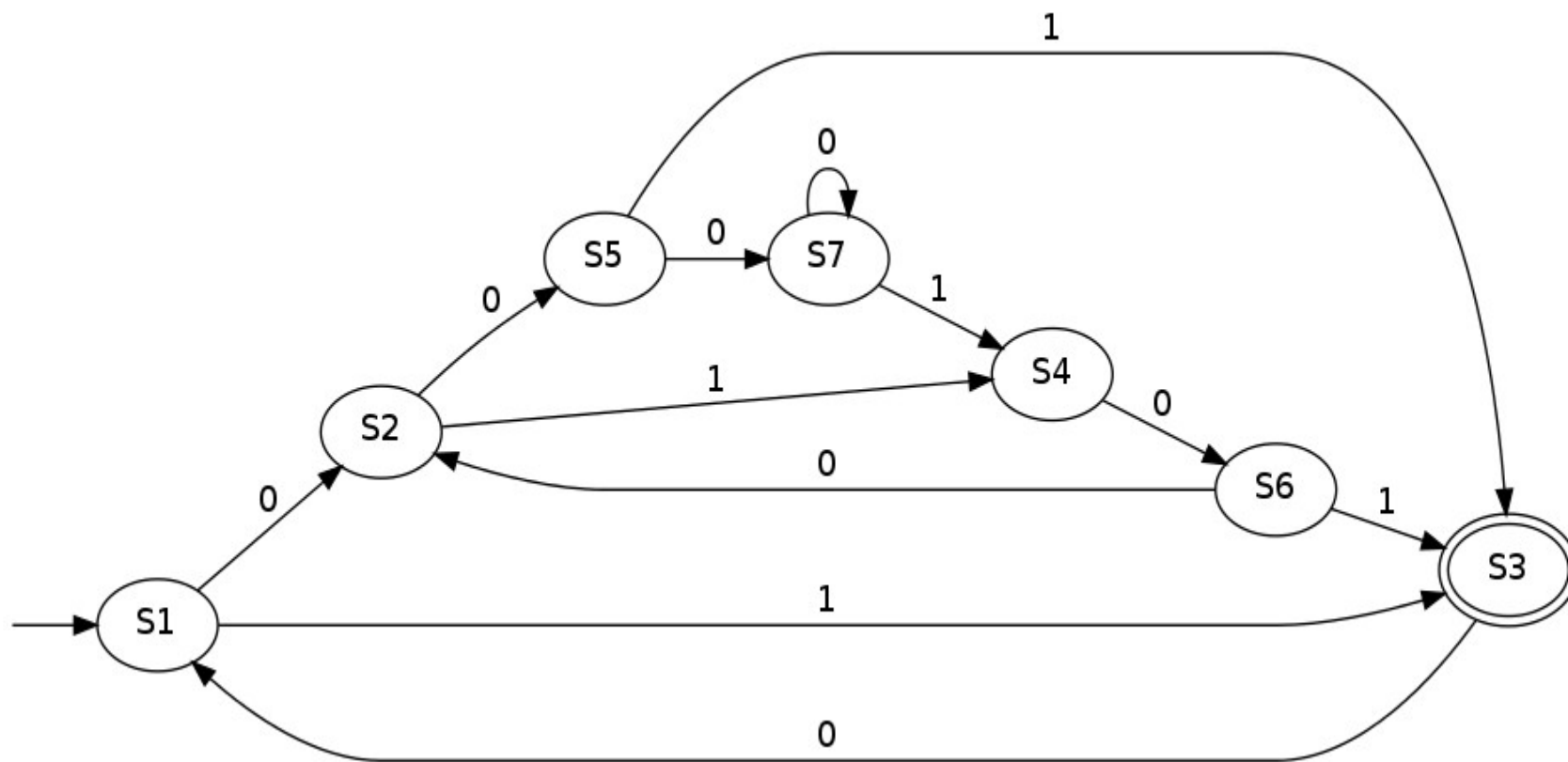
Analyzátor RE – Regulárne výrazy IV.

- **Nevhodnosť reprezentácie**
 - Je náročné hľadať vlastnosti celého jazyka
 - “Aké znaky môžu byť na indexe N?”
 - Traverz AST tento dopyt nerieší

Analyzátor RE – DFA I.

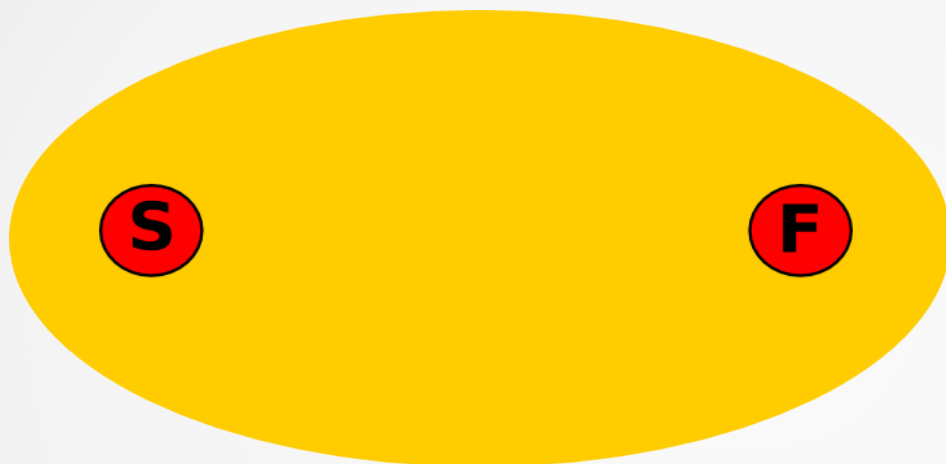
1. Príklad DFA
2. Transformácia RE na DFA

Analýzátor RE – Regulárne výrazy IV.



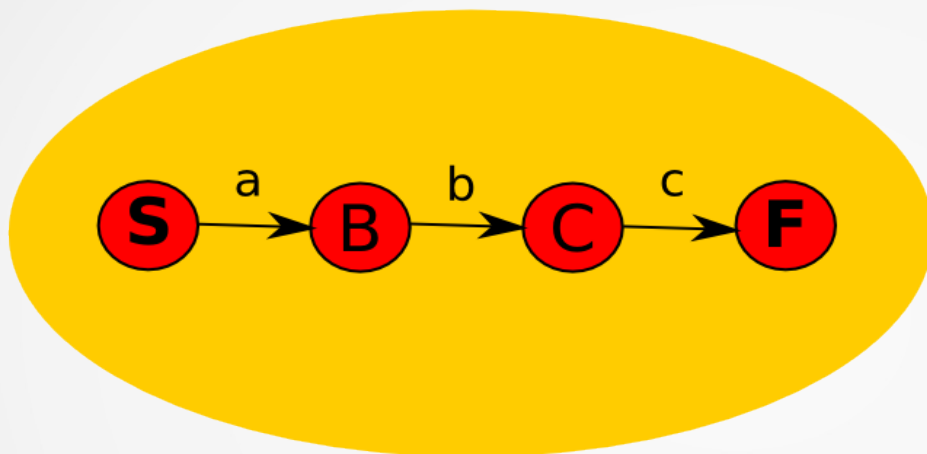
$((0((1|00+1)00)*(100+1)0)1|1)0)*0((1|00+1)00)*(1|00+1)0)1$

Analyzátor RE – Regulárne výrazy IV.



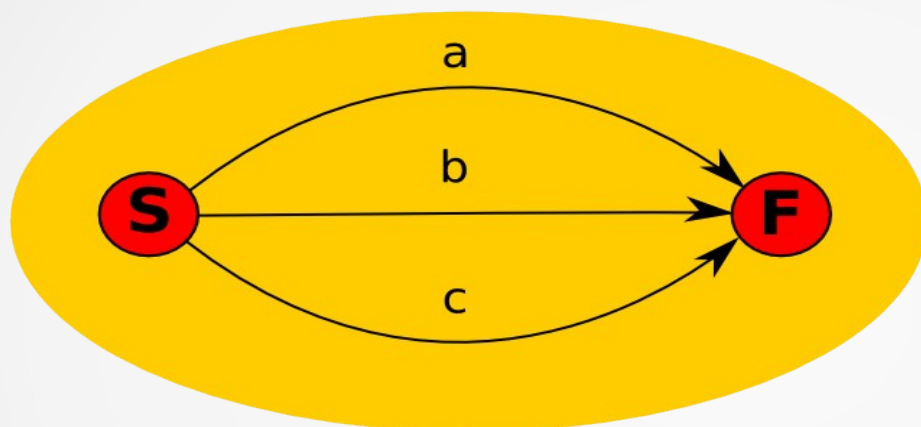
- 1) Každý uzol transformujeme na fragment NFA
- 2) Prechody môžu smerovať z vonku fragmentu len do štartovacieho stavu fragmentu
- 3) Von z fragmentu môžu smerovať len prechody z koncového stavu fragmentu

Analýzátor RE – Regulárne výrazy IV.



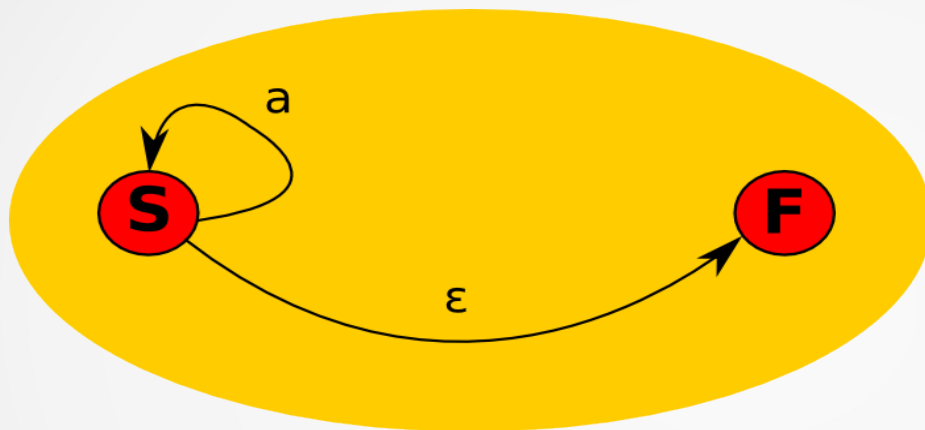
- 1) Prechody sekvencie sú označené znakmi v sekvencii
- 2) Potomka v AST začleníme pomocou predodov do jeho fragmentu

Analýzátor RE – Regulárne výrazy IV.



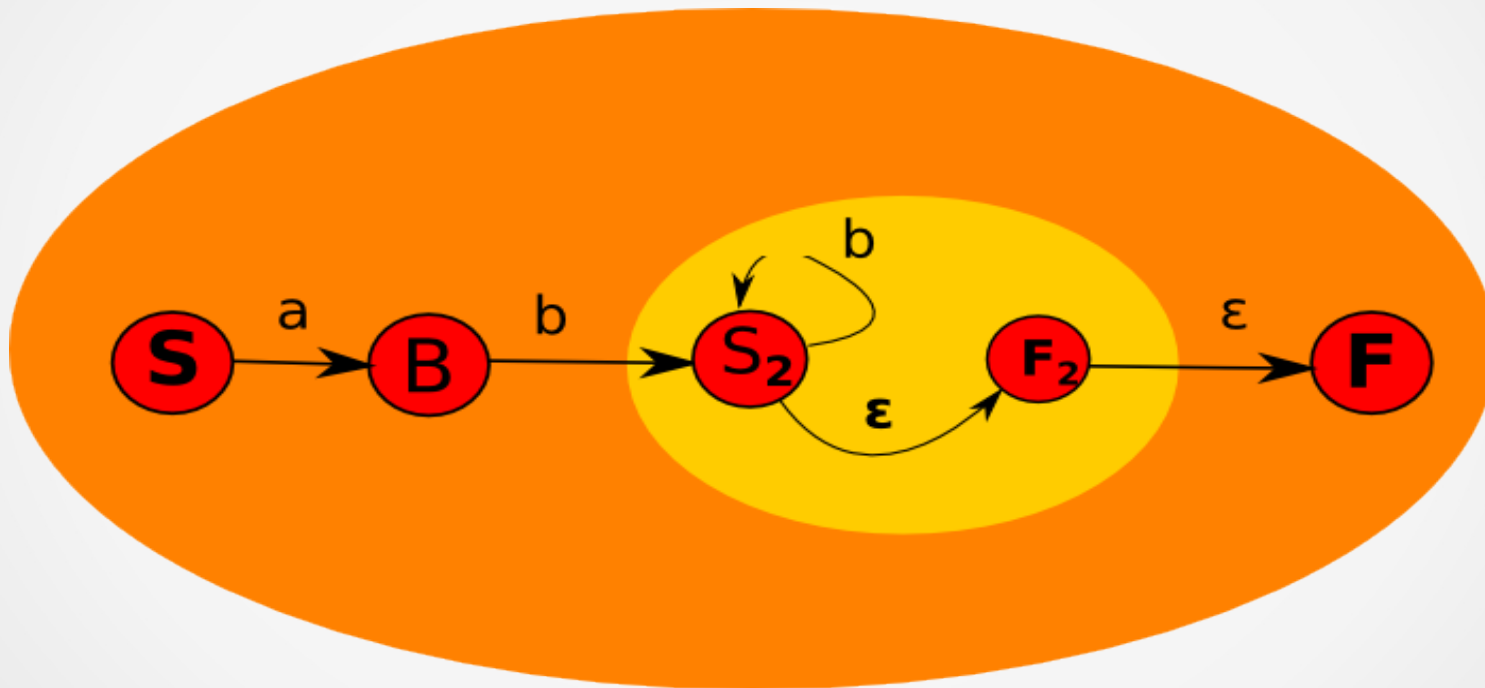
- 1) Prechody alternatívy sú paralelne
- 2) Pokomkov v AST zaradíme tiež paralelne k znakom

Analýzátor RE – Regulárne výrazy IV.



1) Kvantifikácia (**Kleene star*** operátor) má jediného potomka, alebo prehod

Analyzátor RE – Regulárne výrazy IV.



Príklad

Výsledky

- Sme schopný rozpoznať niekoľko elementov gramatiky:
 - kľúčové slová
 - operátori, výrazy a konštanty
 - komentáre

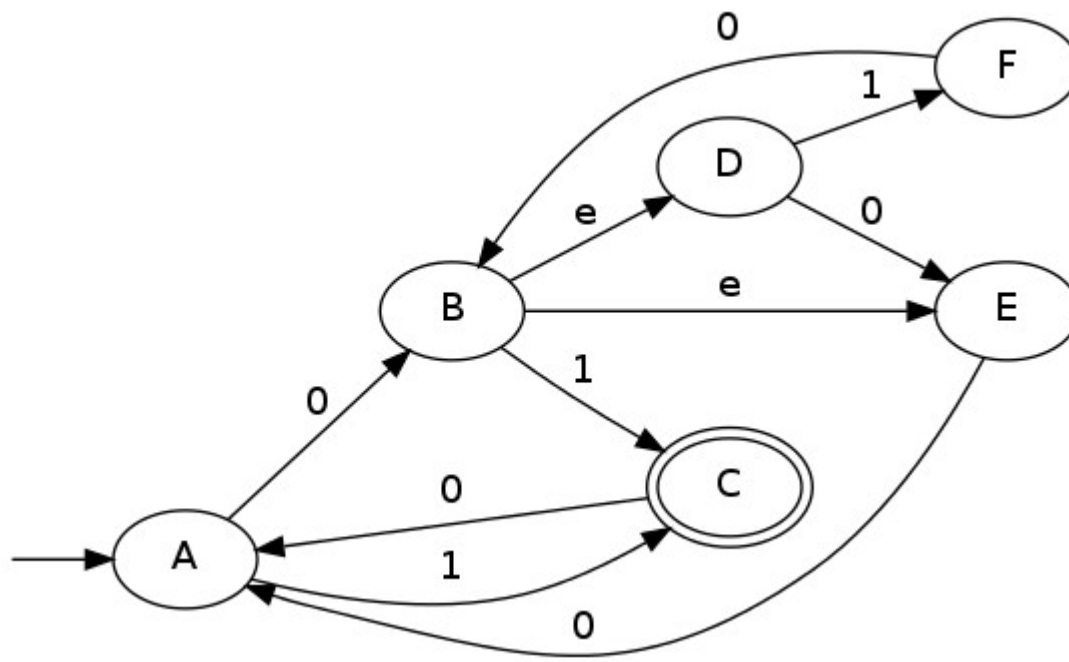
Výsledky

- Implementovali sme editori Kate, Gedit
- Naučil som sa viac ako obvykle (a chcem viac)
- STATS:
 - 61 commit-ov
 - 4327 riadkov kodu
 - 6 git branch-i



Dakujem za pozornost

Zadné vrátka – epsilon-NFA



Zadné vrátka – eNFA definícia

Definícia: Nedeterministický konečný automat s ε -prechodmi je päťica

$$A = (K, \Sigma, \delta, q_0, F) \quad \text{kde}$$

K je konečná množina stavov

Σ je konečná vstupná abeceda

$q_0 \in K$ je začiatkový stav

$F \subseteq K$ je množina akceptačných (koncových) stavov

$\delta : K \times \Sigma_\varepsilon \rightarrow P(K)$ je prechodová funkcia

a zároveň

$$\varepsilon \notin K$$

$$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$$

Zadné vrátka - NFA

