

Bakalářská práce: Konverze modelů regulárních jazyků

Autor: David Navrkal
Email: navrkald@gmail.com

Zkrácené zadání

- 1) Seznámit se s modely regulárních jazyků a nastudovat konverzní algoritmy mezi těmito modely.
- 2) Navrhnout a implementovat aplikaci na didaktickou demonstraci konverzí. Grafické uživatelské rozhraní přizpůsobit využitelnosti ve výuce.
- 3) Připravit sadu minimálně 10 úloh pro studenty.
- 4) Zhodnotit aplikaci a navrhnout další vylepšení.

Motivace: pomoci studentům Formálních jazyků a překladačů.

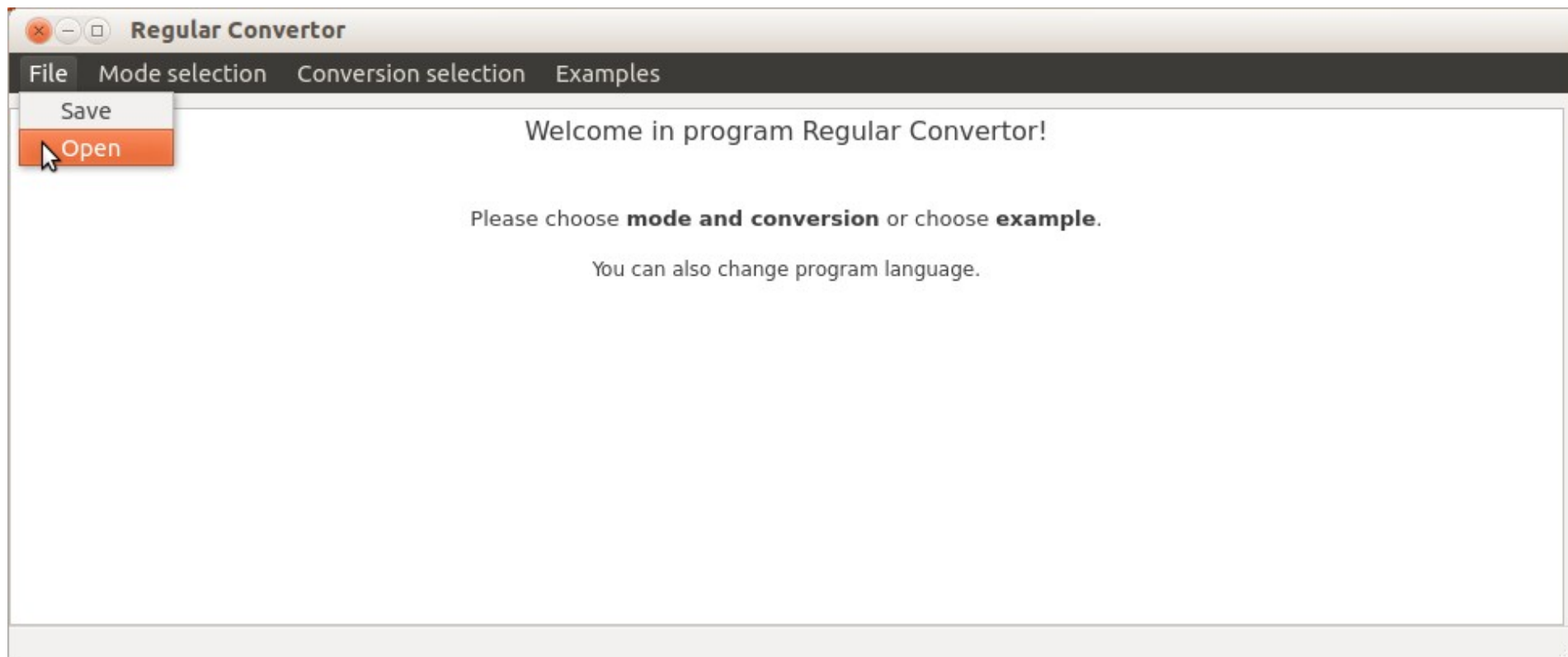
Implementované převodní algoritmy

- Převod regulárního výrazu na konečný automat (KA).
- Odstranění epsilon pravidel z KA.
- Determinizace KA.

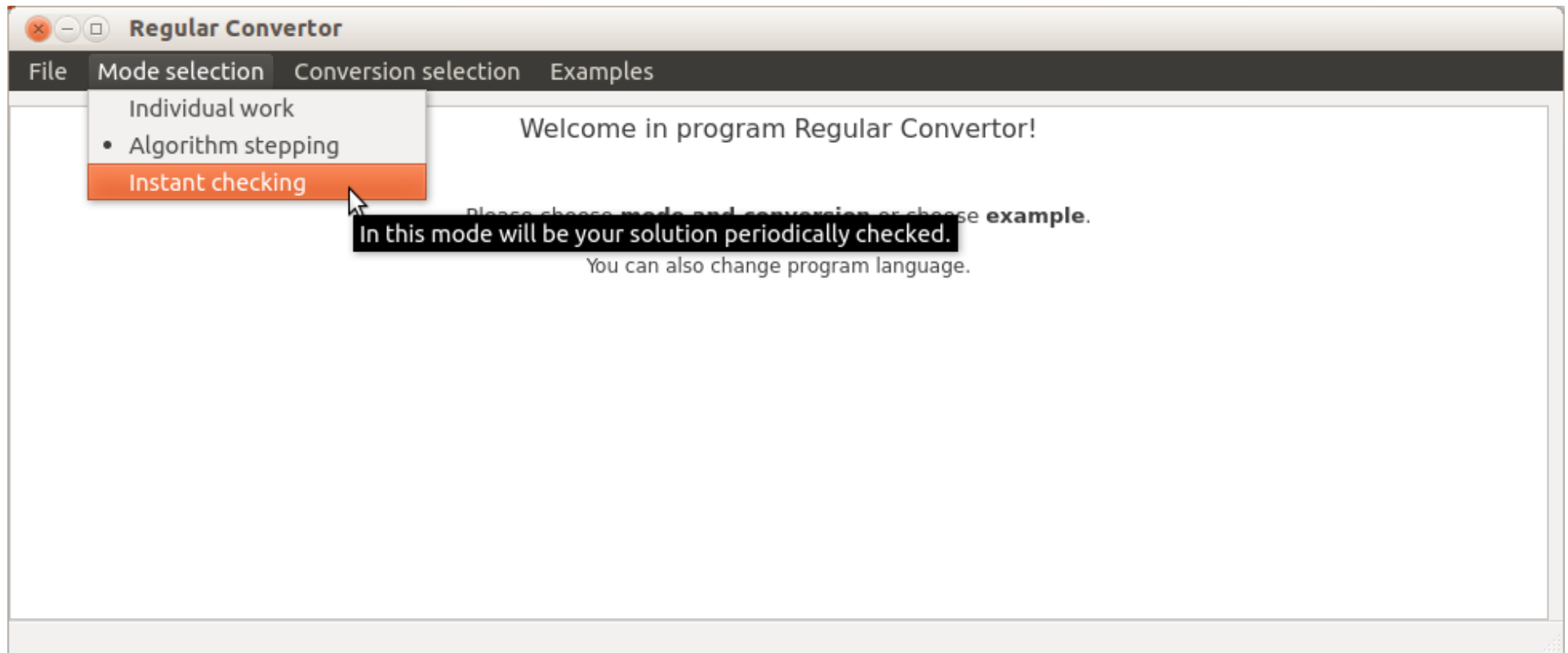
Návrh a implementace

- Grafický návrh aplikace (tzv. mockupy) jsem tvořil v programu **Balsamiq Mockups**.
- Vlastní aplikaci jsem implementoval v programovacím jazyce C++ za použití grafické knihovny Qt5.
- Aplikaci jsem vyvíjel v Linuxu a testoval v Microsoft Windows 7.

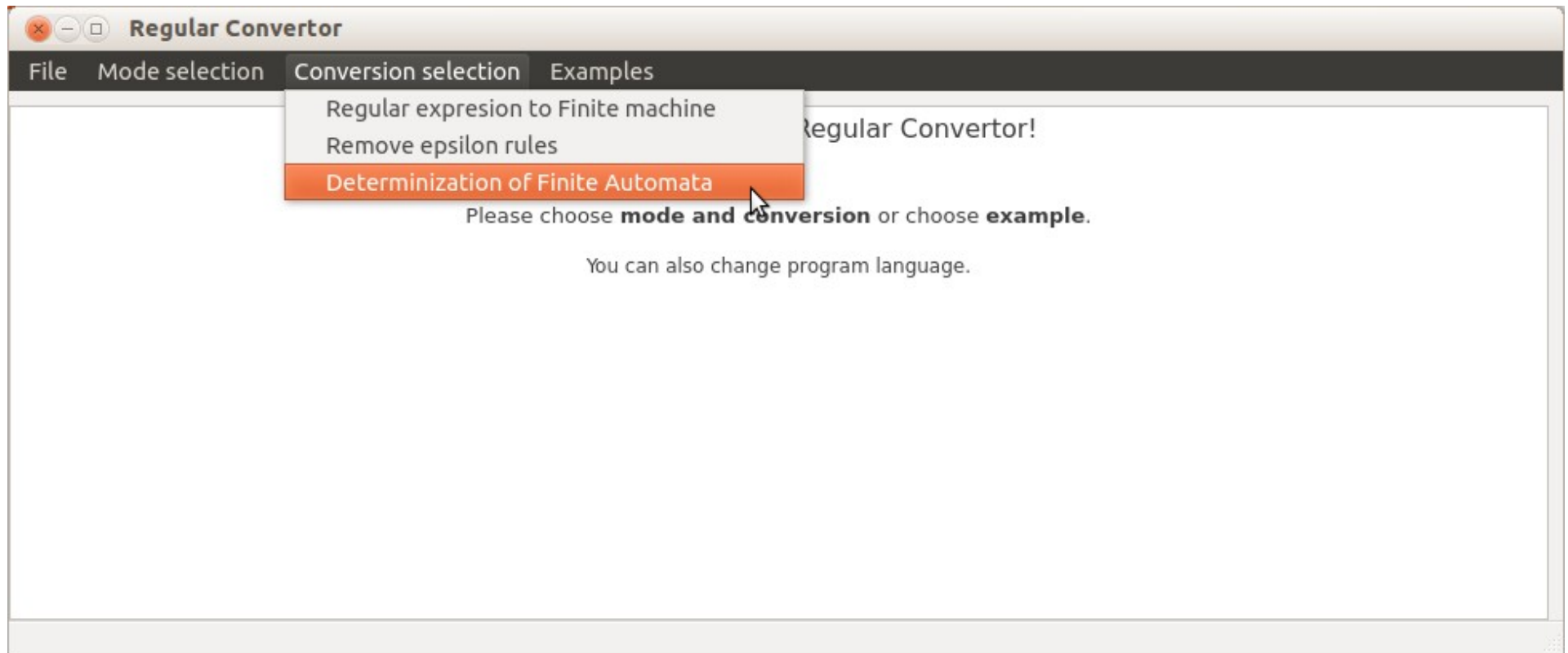
Hlavní okno - soubor



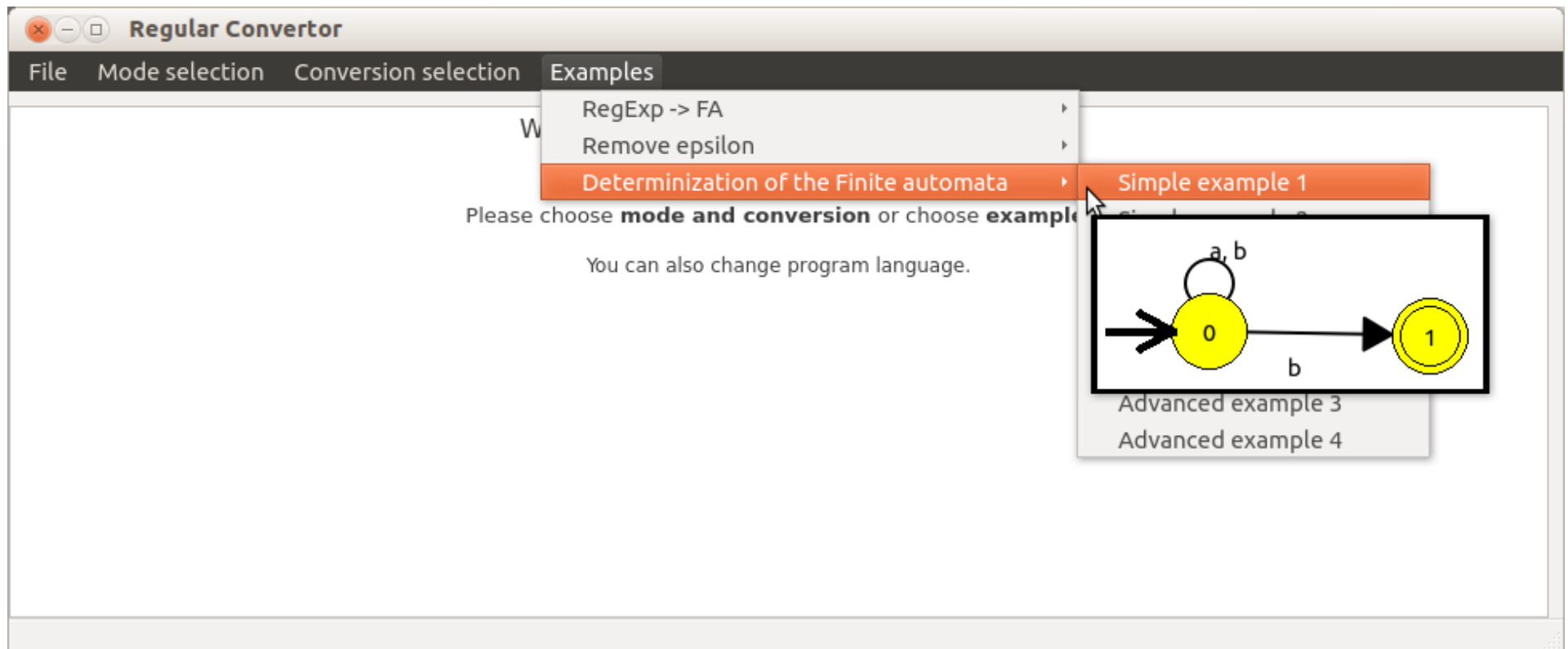
Hlavní okno - módy



Hlavní okno - konverze



Hlavní okno - příklady



- Strom pro RV.
- Uzly stromu (ikony).
- Editor KA.
- Formální popis KA.

Regular Converter - RegExp to Finite automata - Mode: individual work

File Mode selection Conversion selection Examples

Regular expression $a+b^*$ Algorithm RegExp to FA

Insert ϵ

Check solution Show solution

formal

"From inside" RegExp r repeatedly use this rules for construction of the finite automata M :

- For RegExp \emptyset create FA M_{\emptyset} :
- For RegExp ϵ create FA M_{ϵ} :
- For RegExp $a \in \Sigma$ create FA M_a :

Let for RegExp r a t exists FA M_r a M_t Then:

- For RegExp $r.t$ create FA $M_{r.t}$:
- For RegExp $r+t$ create FA M_{r+t} :
- For RegExp r^* create FA M_{r^*} :

left son

FA - graphic FA - formal

chosen node

FA - graphic FA - formal

right son

FA - graphic FA - formal

start state

final state

Odstranění epsilon pravidel – krokovací mód

- Krokování (ovládání).
- Pomocná okna.
- Breakpointy.

Regular Convertor - Remove epsilon rules - Mode: algorithm stepping

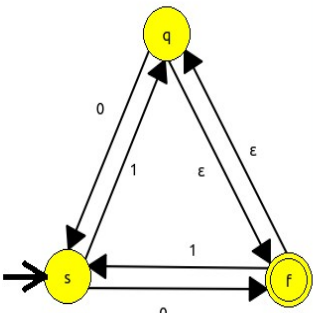
File Mode selection Conversion selection Examples

Input FA

Algorithm: Remove epsilon rules

begin prev stop play next end Delay: 0 ms

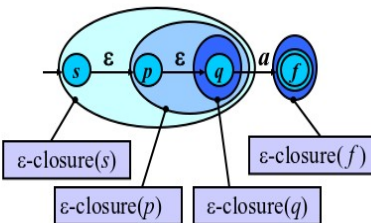
FA - graphic FA - formal



formal

Input: FA $M=(Q, \Sigma, R, s, F)$
Output: FA without ϵ -rules $M'=(Q, \Sigma, R', s, F')$

- for each $p \in Q$ do
- $\epsilon\text{-closure}(p) = \{q; q \in Q, p \vdash^* q\}$
- for each $p \in Q$ do
- for each $p' \in \epsilon\text{-closure}(p)$ do
- for each $r = \{p'a \rightarrow q\} \in R$ where $a \in \Sigma$ do
- $R' \leftarrow R' \cup r' = \{pa \rightarrow q\}$
- for each $p \in Q$ do
- if $\epsilon\text{-closure}(p) \cap F \neq \emptyset$ then
- $F' \leftarrow F' \cup \{p\}$



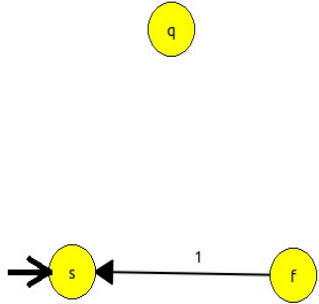
variables

$p = f$
 $\epsilon\text{-closure}(f) = \{f, q\}$
 $p' = f$
 $a = 1$
 $q = s$
 $r = \{f \xrightarrow{1} s\}$
 $r' = \{f \xrightarrow{1} s\}$

3-closer(f) = { f, q } ✓
3-closer(q) = { f, q } ✓
3-closer(s) = { s } ✓

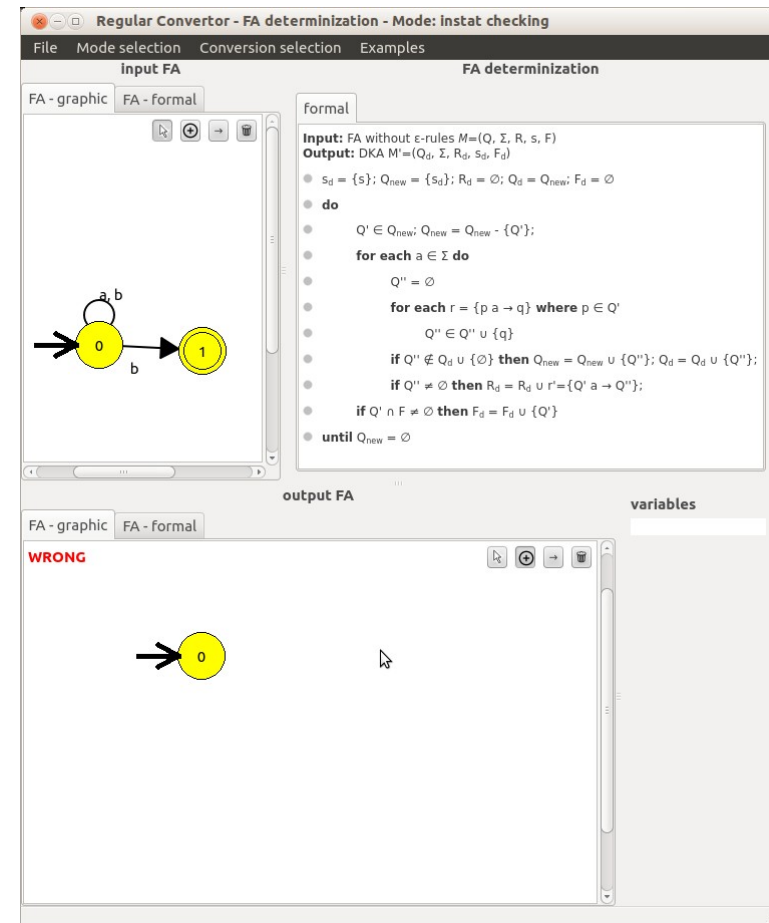
output FA

FA - graphic FA - formal



Determinizace KA – mód průběžné kontroly

- Zobrazuje se aktuální stav (špatně / správně).



Zdrojové kódy

- Zdrojové kódy jsou veřejně přístupné na serveru GitHub na url:
[*<https://github.com/navrkald/regularConvertor>*](https://github.com/navrkald/regularConvertor)

Možná vylepšení

- Implementace dalších převodů:
 - Minimalizace KA.
 - KA na regulární výraz (RV).
 - Převody regulárních gramatik na KA a na RV a zpět.
- Implementace v rámci diplomové práce:
 - Převody modelů jazyků, složitějších než regulárních.

Děkuji za pozornost