VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KONVERZE MODELŮ REGULÁRNÍCH JAZYKŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE David Navrkal
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

KONVERZE MODELŮ REGULÁRNÍCH JAZYKŮ

CONVERSION OF MODELS OF REGULAR LANGUAGES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE David Navrkal
AUTHOR

VEDOUCÍ PRÁCE Ing. Zbyněk Křivka, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

Navrkal David: Konverze modelů regulárních jazyků, bakalářská práce, Brno, FIT VUT v Brně, 2013

Konverze modelů regulárních jazyků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Zbyňka Křivky, PhD.
Další informace mi poskytli...
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
David Navrkal
Datum (!!!! DOPLŇTE DATUM !!!!)

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant, apod.).

Obsah

1 Úvod

Ihned na úvod se chci zmínit o tom, že předpokládám, že čtenář zná základní pojmy týkající se formálních jazyků, především pak jazyků regulárních, pojmů, definic, modelů a algoritmů jejich vzájemných převodů a další věcí s tím související. Přesto musím uvést alespoň základní definice, pro případ, že by toto práci četl i někdo jiný, který nestudoval na této fakultě a také proto, že při studiu teorie jsem narazil na některé mírně odlišné definice. Méně důležité pojmy a definice jsem do hlavního těla práce neuváděl a můžete je nalézt pouze jako přílohy.

V této práci jsem se soustředil především na návrh a implementaci.

2 Teorie

V této části si popíšeme teorii týkající se modelů regulárních jazyků. V případě, že dané modely dobře znáte, můžete tuto část přeskočit a přečíst si konverzní algoritmy mezi těmito modely.

2.1 Konečný automat

Následující definici jsem přebral z [1 str. 101] .

Definice:

Konečný automat (KA) je pětice:

$M=(Q,\Sigma,R,s,F)$, *kde:*

- Q *je konečná množina stavů*
- Σ *je vstupní abeceda*
- R *je konečná množina pravidel tvaru:* $pa \rightarrow q$, *kde* $q, p \in Q$, $a \in \Sigma \cup \{\varepsilon\}$.
- $s \in Q$ *je počáteční stav*
- $F \subseteq Q$ *je množina koncových stavů*

Tuto definici zde uvádím zejména proto, že jsem se setkal na internetu i v literatuře s odlišnými definicemi, které definovali například *s* ne jako počáteční stav, ale jako množinu počátečních stavů. (Pro KA podle definice z [1 str. 101] se dá nedeterministický výběr počátečního stavu simulovat ε přechody z počátečního stavu do „simulovaných počátečních stavů“.)

2.2 Regulární výrazy

Následující definici jsem přebral z [1 str. 33]

Definice:

Nechť Σ je abeceda. Regulární výrazy nad abecedou Σ a jazyky, které značí, jsou definovány následovně:

$\emptyset$ je RV značící prázdnou množinu (prázdný jazyk)

ε je RV značící jazyk $\{\varepsilon\}$

a, kde $a \in \Sigma$, je RV značící jazyk $\{a\}$

Nechť r a s jsou regulární výrazy značící po řadě jazyky L_r a L_s, potom:

(r.s) je RV značící jazyk $L = L_r L_s$

(r+s) je RV značící jazyk $L = L_r \cup L_s$

(r) je RV značící $L = L_r{}^*$*

V literatuře můžeme nalézt i definice r^+ značící rr* nebo r*r. My tohle rozšíření nebudeme používat, protože vyjadřovací síla regulárních výrazů podle předešlé definice zůstává stejná. Pro zjednodušení můžeme používat *rs* na místo notace *r.s*.

Kvůli redukci počtu závorek budeme uvažovat následující priority: „* > . > +“ („*“ má větší prioritu než „.“ a to má větší prioritu než „+“).

2.3 Regulární gramatika

Pro jistotu si zde uvedeme i obecnou definici gramatiky a následně uvedeme i definici pravé lineární gramatiky a pravé regulární gramatiky.

Definice gramatiky:

Gramatika G je čtveřice $G = (N, \Sigma, P, S)$, kde

- *N je konečná množina nonterminálních symbolů.*
- *Σ je konečná množina terminálních symbolů.*
- *P je konečná množina kartézského součinu $(N \cup \Sigma)^*N.(N \cup \Sigma)^*\Sigma$, nazývaná množinou přepisovacích pravidel.*
- *$S \in N$ je výchozí (startovací) symbol gramatiky G.*

Prvek $(\alpha, \beta) \in P$ je přepisovací pravidlo a zapisuje se ve tvaru $\alpha \rightarrow \beta$, kde α je levá strana, β je pravá strana pravidla, $\alpha \rightarrow \beta$.

Viz [2].

V předchozí části jsme si představili definici obecné gramatiky, nás však bude zajímat gramatika regulární, konkrétně nyní pak pravá lineární gramatika.

Definice pravé lineární gramatiky:

Pravá lineární gramatika je taková, která má přepisovací pravidla ve tvaru:

$A \rightarrow xB$, nebo $A \rightarrow x$ kde $A, B \in N$, $x \in \Sigma^*$ (kde Σ^* je množina řetězců nad abecedou Σ)

Viz [2].

Nyní si nadefinujeme pojem pravá regulární gramatika. Jak následně uvidíte, jedná se o speciální typ pravé lineární gramatiky kde x je řetězec délky jedna.

Definice pravé regulární gramatiky:

Pravá regulární gramatika je taková, která má přepisovací pravidla ve tvaru:

$A \rightarrow xB$, *nebo* $A \rightarrow x$ *kde* $A, B \in N, x \in \Sigma$

Viz [2].

3 Specifikace požadavků

Následující požadavky jsem získal opakovanými debatami s vedoucím mé bakalářské práce a mými vlastními úvahami. Postupně jsme dospěli k následujícím požadavkům.

- Má se jednat o aplikaci s důrazem na grafické uživatelské rozhraní.
- Aplikace mát být výukový nástroj, který má pomoci studentům formálních jazyků a teoretické informatiky pochopit a naučit se používat konverzní algoritmy modelů regulárních jazyků.
- Aplikace nemá být určená na konverzi velkých objemů dat, hlavní důraz se klade na didaktickou demonstraci převodních algoritmů.
- Implementace následujících typů převodů:
    - Obecný konečný automat (KA) na KA bez ε pravidel.
    - KA bez ε pravidel na deterministický KA bez nedostupných stavů (DKA).
    - DKA na dobře specifikovaný konečný automat (DSKA).
    - DSKA na minimální konečný automat.
    - KA na regulární výraz.
    - Regulární výraz na KA.
    - KA bez ε pravidel na pravou regulární gramatiku.
- Vizualizace toho, jak konečný automat zpracovává vstupní řetězce.
- Aplikace by měla mít následující módy převodů:
    - **Krokový režim**, ve kterém uživatel převádí jeden model na druhý po jednotlivých krocích algoritmu. Tento režim je nejdůležitější, protože v něm se uživatel učí krůček po krůčku používat daný konverzní algoritmus. V tomto režimu se postupně zvýrazňují jednotlivé kroky algoritmu, které má uživatel vykonat.
    - **Režim kontroly**, ve kterém uživatel samostatně převede jeden model na druhý a program mu pouze zkontroluje, zda převod provedl správně. Tento mód bude užitečný hlavně uživatelům, kteří převodní algoritmus již pochopili a chtějí si zkontrolovat svou samostatnou práci.

    - **Režim běhu**, ve kterém program bez zásahu uživatele převede jeden model na druhý. V tomto režimu si může uživatel algoritmus krokovat a pozorovat jak mají být správně provedeny jednotlivé kroky. Využití má především pro uživatele, kteří nepochopili převodní algoritmus a chtějí se podívat na správný postup řešení.
- Aplikace má být využitelná pro hromadnou demonstraci (tj. na přednášce, nebo na demo cvičení), z čehož vyplívá, že mají být vidět použité fonty i ze zadních řad. Zároveň by měla být použitelná na osobních počítačích pro samostatné procvičení přednášené látky studenty. Z obojího plyne, že by aplikace měla mít možnost měnit velikost použitých písmen a změnu měřítka grafických prvků.
- Primárně má aplikace běžet na operačním systému Microsoft Windows XP a novější.
- Výsledný produkt má být hlavně určen pro použití na Fakultě informačních technologii VUT v Brně, proto použité názvosloví, definice a barevné značení bude odpovídat tomu, které se používá v předmětech Formální jazyky a překladače a Teoretická informatika vyučovaných na naší fakultě.

4 Návrh

Kapitolu Návrh jsem záměrně rozdělil do dvou částí, toho co uvidí uživatelé (návrh grafického uživatelského rozhraní) a na návrh implementace (objektový návrh).

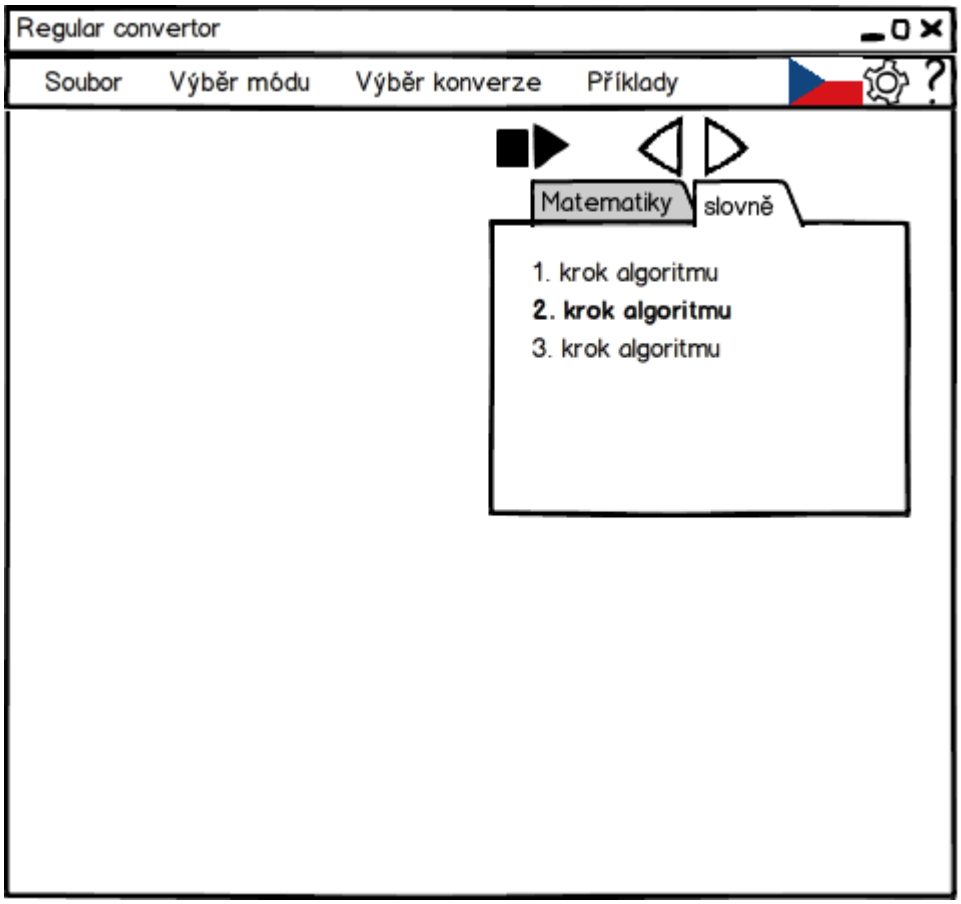
4.1 Návrh grafického uživatelského rozhraní

Při návrhu grafického uživatelského rozhraní (GUI) jsem vycházel z klasického rozložení a vzhledu ovládacích prvků. Mým cílem bylo vytvořit aplikaci s chováním a vzhledem, na které je uživatel zvyklý z obvyklých grafických aplikací z prostředí MS Windows a ne aplikaci s experimentální GUI. Účelem je, aby se uživatel rychle orientoval v nové aplikaci a mohl se tak soustředit více na demonstrované převodní algoritmy, než na otázky typu: „Na co mám kliknout, aby aplikace udělala to a to?“, nebo: „Kde najdu to a to abych mohl udělat to a to“. Pro usnadnění práce se po najetí myši na ovládací prvek zobrazí nápověda.

Následující obrázky jsem vytvořil pomocí programu Balsamic pro tvorbu návrhu uživatelských rozhraní, tzv. „mockupů“, proto se může vzhled výsledné aplikace mírně lišit.

4.1.1 Návrh zobrazení základních elementů

4.1.1.1 Hlavní okno



Obrázek 1: Hlavní okno programu před výběrem některé z konverzí

Na obrázku 1 v hlavním menu se popořadě nachází položky **Soubor**, pod kterou se nachází standardní volby jako, uložení výchozího modelu pro převod do souboru a načtení modelu ze souboru. Dále pak **výběr módu** a **výběr konverze**, obojí viz kapitola specifikace požadavků. Jako poslední položka menu obsahuje **příklady,** ve kterých najde uživatel předpřipravené příklady modelů a jejich konverzí, které si může sám vyzkoušet. Napravo od příkladů uživatel uvidí vlajku ukazující **aktuální jazyk**, po kliku na něj se zobrazí výběr ostatních lokalizací. Napravo od něj se nachází ikony pro **nastavení** a práci s **nápovědou**.

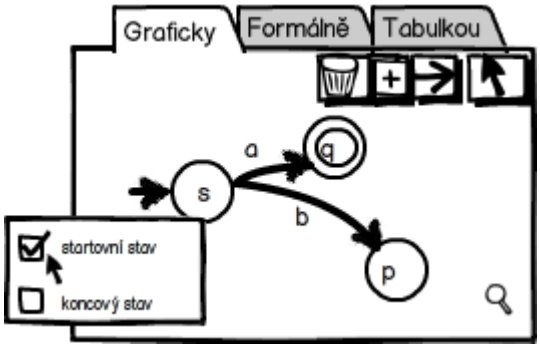
V pravé části obrazovky můžete vidět okno s dvěma záložkami, v obou se nachází algoritmus převodu. Jediný rozdíl je, že v záložce pojmenované **matematicky** se nachází algoritmus v zapsaný v matematické podobě a druhé pojmenované **slovně** jsou instrukce ve formě vět.

Nad tímto oknem můžete vidět tlačítko vypadající jako černý čtverec symbolizující příkaz stop. Když se uživatel nachází v krokovém módu, pak po stisku této ikony se přepne do modu editace původního převáděného modelu, krokový mód se tím ukončí. Napravo od tohoto tlačítka je druhé připomínající vybarvený černý trojúhelník symbolizující „play“ při přehrávání hudby. Má více

významů, v krokovém režimu zkontroluje uživateli, zda provedl daný krok algoritmu v pořádku a posune se v algoritmu na další krok. V režimu kontroly se zkontroluje, zdali uživatel provedl celou konverzi správně. A v módu běhu program zobrazí správný výsledek převodu.

4.1.1.2 Grafická reprezentace konečného automatu

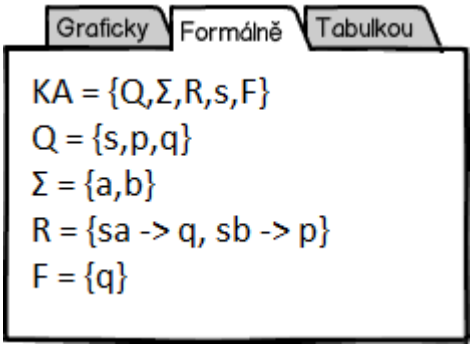
Na konečný automat můžeme reprezentovat třemi způsoby, první a to **grafický pohled** můžete vidět na obrázku 2.



Obrázek 2: Grafický pohled na KA po kliknutí pravým tlačítkem myši na uzel s

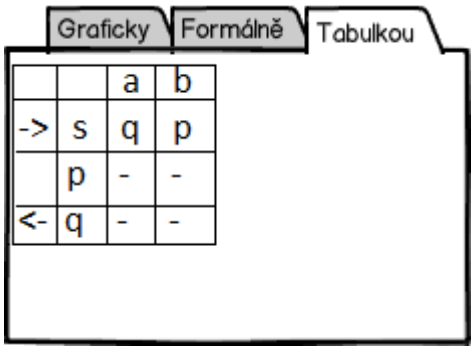
Ostatní dva představím následně. V grafickém pohledu se nachází v pravém horním rohu tlačítka na **mazání** uzlů a hran, dále pak tlačítko na **přidávání uzlů** a **přidávaní hran** a na **přemisťování** hran a uzlů. Pokud chceme zároveň přesouvat více uzlů, můžeme je označit klikem myši spolu s držením klávesy „ctrl“. Po kliknutí pravím tlačítkem na uzel, jak je vidět na obrázku, má uživatel možnost nastavit uzel jako koncový a jako startovní (aplikace dovolí uživateli nastavit jako startovní pouze jeden uzel, viz kapitola Teorie). Vpravo dole se nachází ikona pro přiblížení, nebo oddálení objektů v grafickém pohledu, stejného efektu se dá dosáhnout i stiskem klávesy „ctrl“ a pohybem kolečka myši.

Na následujícím obrázku 3 můžete vidět formální zápis KA z předchozího, kde byl tentýž automat zobrazen graficky. Kde Q je konečná množina stavů, Σ je abeceda, R je množina přechodů, s je startovní stav a F je konečná množina koncových stavů.



Obrázek 3: Formální popis KA z obrázku 2

Třetím možným zápisem je zapsat KA tabulkou. Tohoto zápisu nejvíce využijeme při algoritmu minimalizace KA.



		a	b
->	s	q	p
	p	-	-
<-	q	-	-

Obrázek 4: Tabulkový popis obrázku 2

Na obrázku 4 můžete vidět zápis KA tabulkou v prvním sloupci, šipka značí, jestli se jedná o počáteční stav (šipka doprava), nebo o koncový stav (šipka doleva). V druhém sloupci je jméno uzlu. V záhlaví další části tabulky jsou uvedeny symboly abecedy a pod nimi je naznačena množina přechodů. Pokud z místa nevede hrana, se symbolem abecedy, pak je v tabulce uveden znak pomlčky. V tabulce je zapsán opět stejný KA jako z předchozích obrázků.

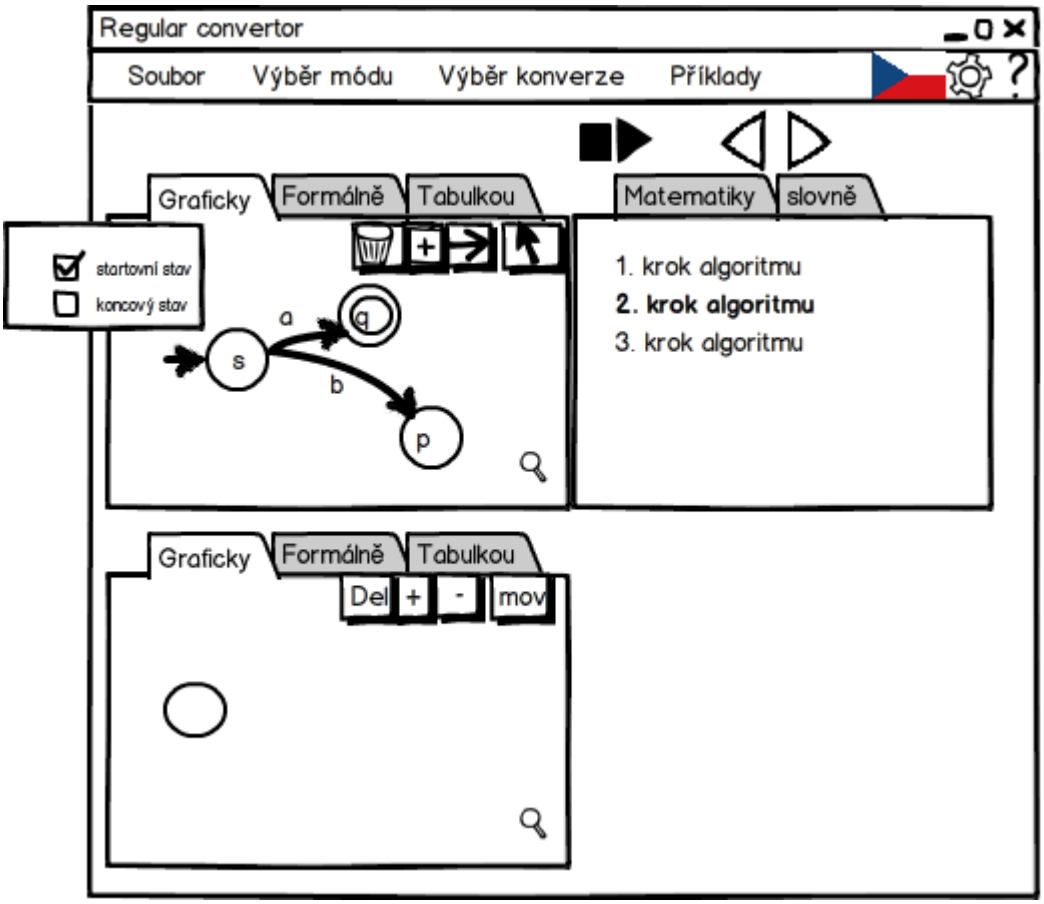
4.1.2 Návrh zobrazení převodů

Při návrhu GUI pro převody jsem se snažil držet schématu, že vždy vpravo nahoře bude převodní algoritmus, nalevo od něj bude výchozí model pro převod a pod ním výsledný model.

4.1.2.1 Konverze konečných automatů

Následující schéma zobrazené na obrázku 5 znázorňuje převod konečného automatu na jeho speciální typy a to:

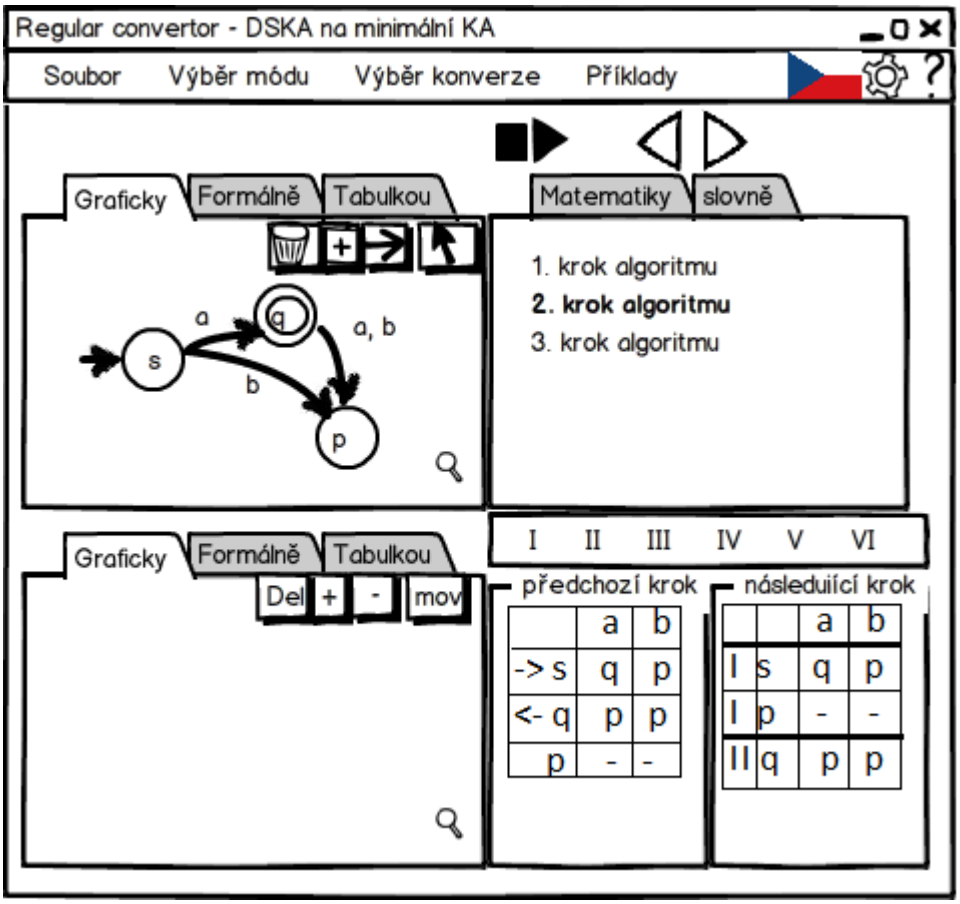
- Obecný konečný automat (KA) na KA bez ε pravidel.
- KA bez ε pravidel na deterministický KA bez nedostupných stavů (DKA).
- DKA na dobře specifikovaný konečný automat (DSKA).



Obrázek 5: Hlavní okno pro převod konečných automatů

Uživatel bude postupovat podle převodního algoritmu a postupně bude v levém dolním okně tvořit výsledný KA. Pokud bude uživatel převádět v *krokovém režimu*, pak nebude moci po dobu jeho spuštění moci upravovat výchozí KA.

Speciálním typem převodu konečných automatů je převod **DSKA** na **minimální konečný automat**. Při tomto převodu potřebujeme dvě pomocná okna, ve kterých bude uživatel hledat rozlišitelné stavy.



Obrázek 6: Obrazovka pro převod DSKA na minimální KA

Jak můžete vidět na obrázku 6, tyto dvě okna jsem umístil do volného prostoru vpravo dole pod okno s algoritmem. V levém okně pojmenovaném **předchozí krok** uživatel vidí předchozí krok algoritmu. Nový krok, který se následně stane krokem předchozím, tvoří uživatel v levém okně pojmenovaném **následující krok**. Jelikož je konvence v převodu na minimální KA označovat jednotlivé skupiny uzlů římskými číslicemi, je nad těmito dvěma okny prvek, ve kterém si může uživatel vybrat, kterou římskou číslicí chce označit kterou skupinu. Po rozdělení uzlů KA do finálních skupin musí uživatel výsledný automat překreslit do výstupního okna pro minimální automat.
Z tohohle pohledu lze okna *předchozí* a *následující krok* brát jako okna pomocné.

TODO: Návrh KA -> RV, RV -> KA,

4.2 Objektový návrh

TODO: diagram tříd a jeho popis.

5 Implementace

Celý projekt jsem implementoval v C++ za použití grafické knihovny Qt 4 verze 8.2. Zdrojové kódy byly vyvíjeny pod linuxovým operačním systémem a testovány v MS Windows 7 a XP.

TODO: Dokončit

6 Závěr

TODO: Závěr

Literatura

1. **Meduna, Alexandr.** *Automata and Languages: theory and applications.* London : Springer, 2000. ISBN 81-8128-333-3.

2. **Smrčka, Aleš, Vojnar, Tomáš and Češka, Martin.** *Teoretická informatika - studijní opora.* Brno : s.n., 2011. p. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/TIN/public/Texty/oporaTIN.pdf>.

Seznam obrázků

Seznam příloh

Konverzní algoritmy modelů regulárních jazyků

Na následujícím obrázku uvidíte diagram znázorňující převodní algoritmy mezi jednotlivými modely regulárních jazyků s očíslováním použitým v této kapitole.

TODO vložit obrázek.

Obecný KA na KA bez ε pravidel

ε uzávěr

V algoritmu převodu budeme používat pojmu *ε uzávěr* proto si jej nyní definujme.

Definice:

Pro každý stav $p \in Q$ je definován ε-uzávěr(p):

ε-uzávěr(p) = $\{q: q \in Q, p \vdash^* q\}$

Neformálně pak můžeme říct, že *ε uzávěr* stavu p obsahuje množinu všech takových uzlu, do kterých se můžeme dostat ze stavu p bez přečtení vstupního symbolu.

Algoritmus převodu

Vezmi startovní uzel *s* z původního KA a udělej jeho ε uzávěr. Všechny takto nalezené uzly vezmi a sjednoť je do nového startovního stavu s´ KA´ nez ε pravidel.

Pokud nový uzel q´ obsahuje alespoň jeden stav, který byl v původním KA stavem koncovým, pak označ q´ jako koncový uzel.

Pro všechny symboly abecedy Σ:

Vezmi symbol abecedy a podívej se, kam se dá dostat v původním KA z ε uzávěru stavu s.

Všechny takto nalezené uzly vezmi s sjednoť je spolu s jejich ε uzávěry do nového uzlu automatu KA´.

Pokud se již takový uzel v KA´ vyskytuje, nevytvářej nový uzel.

Vytvoř přechod se symbolem abecedy v automatu KA´.

Pokud takto vznikl alespoň jeden nový uzel, pokračuj bodem 2. pro všechny nově vzniklé uzly KA´, jinak skonči.

Formálně zapsáno:

$R' := \emptyset$

Pro všechny $p \in Q$:

$R' := R' \cup \{pa \rightarrow q: p'a \rightarrow q \in R, a \in \Sigma, p' \in \varepsilon\text{-uzávěr}(p), q \in Q\};$

$F' := \{p: p \in Q, \varepsilon\text{-uzávěr}(p) \cap F \neq \emptyset\}$

Citace 4. Přednáška IFJ

KA bez ε pravidel na DKA bez nedostupných stavů

Algoritmus převodu

Neformální zápis:

Vstupní stav původního automatu přidej do stavů DKA

Začínáme se vstupním místem a podíváme se, kam se z něj můžeme dostat. V případě nedeterminismu (jedním vstupním symbolem se dostaneme do více stavů) tyto různé stavy sloučíme do jednoho a spolu s relací přechodu vložíme do DKA.

Postupně bereme další neprozkoumané stavy DKA a díváme se kam složky nových stavů vedou a přidáváme nové stavy a přechodové funkce do té doby dokud mi zbývají neprozkoumané stavy v původním KA.

Všechny stavy DKA, které obsahují složky které byli v původním KA v množině koncových stavů, označím jako koncové.

Formální zápis:

Vstup: KA bez ε-přechodů: $M = (Q,\Sigma,R,s,F)$

Vystup: DKA: $M_d = (Q_d, \Sigma, R_d, s_d, F_d)$

$Q_d := \{Q': Q' \subseteq Q, Q' \neq \emptyset\}$

$R_d := \emptyset$

Pro všechny $Q' \in Q_d$ a všechny $a \in \Sigma$

$Q'' := \{q: p \in Q', pa \rightarrow q \in R\}$

Pokud $Q'' \neq \emptyset$ pak $R_d := R_d \cup \{Q'a \rightarrow Q''\}$

$s_d := \{s\}$

$F_d := \{F'': F' \in Q_d, F' \cap F \neq \emptyset\}$

Citace IFJ 4. Přednáška.

DKA bez nedostupných stavů na DSKA