AutoGen

Multimediální simulátor pro kurz Teorie automatů

Bakalářská práce

Martin Němeček

Vedoucí práce: Doc. RNDr. Václav Nýdl, CSc.
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
2009

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejnění své bakalářské práce, a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 19. 4. 2009

Anotace

Tato bakalářská práce představuje doprovodný materiál projektu AutoGen, který slouží pro podporu výuky Teorie Automatů. Čtenář získá základní povědomí o Teorii Automatů a naučí se používat a ovládat všechny aspekty programu AutoGen, který můžete nalézt na přiloženém médiu.

Abstract

This paper represents cover manual for AutoGen project, which is supposed to enhance the learning experience of automata theory. The reader will gain elementary knowledge of automata theory and learn to use and control all aspects of AutoGen program, which can be found on the cover medium.

Obsah

1.Úvod

Můj motiv k tvorbě této práce byl zefektivnění výuky předmětu Teorie automatů. Při absolvování tohoto kurzu jsem si všimnul, že i přes špičkovou úroveň a připravenost přednášejícího docházelo k velkým zdržením kvůli kreslení názorných ukázek na tabuli a následnému čekání, až studenti informace zpracují. Protože věřím, že systém ukázek je pro účely výuky perfektní, rozhodl jsem se nahradit tabuli a fixu za počítačem řízenou simulaci.

Cílem mého projektu bylo vytvořit ucelené prostředí pro výuku fundamentálních aspektů Teorie automatů, které by bylo vhodné jak pro účely prezentace, tak pro domácí studium a experimenty. Začal tedy vývoj simulátoru automatů, který jsem nazval AutoGen. Mé požadavky na tento projekt byly poměrně náročné.

AutoGen musel dosáhnout následujících standardů:

- Simulovat operace na slovech a umožnit uživatelům přidávat vlastní operace.
- Tvořit automaty pomocí matice direktiv namísto visuálního modelování, jednak kvůli efektivnosti a také aby donutil studenta se s tímto procesem seznámit.
- Použít vytvořené automaty k provádění simulací včetně grafické reprezentace aktuálního stavu automatu.
- Možnost tvoření a simulace Mealyho a Moorova automatů.
- Možnost provádět nejpoužívanější operace s maticemi automatů.

Dále bylo třeba napsat tento doprovodný dokument, který by zmapoval možnosti a použití projektu AutoGen, dal by neznalému alespoň okrajově nahlédnout do teorie automatů a v neposlední řadě by zdokumentoval možnosti alternativních a volně dostupných projektů, které v některých oblastech mohou AutoGenu konkurovat.

2.Základy teorie [1]

2.1. Abeceda

Většina lidí si pod pojmem abeceda představí notoricky známou posloupnost od A do Z, tato abeceda je ale pouze jedna z nekonečného množství posloupností složených ze znaků, čísel nebo symbolů. V odborné literatuře se jakákoliv abeceda obecně označuje řeckým písmenem Σ.

Příkladem abecedy může být binární $\Sigma = \{0,1\}$ nebo $\Sigma = \{a,b\}$. Příklad složitější konstrukce může být hexadecimální abeceda:

$$\Sigma = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$$

2.2. Slova

Na problém abecedy logicky navazují slova. Slova nejsou nic jiného, než jakékoliv konečné posloupnosti písmen z dané abecedy. Písmena abecedy se ve slově mohou libovolně opakovat. V odborné literatuře se množina všech možných slov nad abecedou Σ označuje Σ^*. Pokud chceme vyjádřit slovo, které nemá žádná písmena, použijeme znak ϵ (prázdné slovo).

Nad slovy je možné provádět algebraické operace, tou nejzákladnější operací je reflexe (neboli reverzní slovo). Pokud $W = abc$ => $W^R = cba$. Jiným příkladem operace může být délka slova $\#(W)$ nebo zřetězení slov $W = ab$; $U = cd$; $W.U = abcd$.

2.3. Jazyky

Pod pojmem jazyk si většina lidí představí souhrn slovní zásoby a gramatických pravidel, které nám umožňují komunikovat. Z pohledu teorie automatů je nazýváme přirozené jazyky a z matematického hlediska je velice obtížné je definovat. Proto zavádíme pojem formální

[1] Základní terminologie odpovídá materiálu Křetínský a kol. **Automaty a formální jazyky I**. (2008)

jazyk, na který se díváme jako na jakoukoliv podmnožinu množiny Σ. Jazyk formálně označujeme písmenem L.

Jazyky obecně dále dělíme na jazyky nekonečné a konečné podle toho, zda mají konečný počet slov.

2.4. Automaty

Konečný automat je abstraktní model pro práci systémů, u kterých je možné rozlišit konečný počet stavů a pravidla pro zpracování vstupních podnětů. Aktuální stav takového systému se mění pouze na základě vnějšího podnětu a každý stav a podnět nese direktivu s informací, do jakého stavu systém přejde. Konečné automaty lze rozdělit do několika skupin, zde se zaměříme na následující:

2.4.1. Deterministické

V případě deterministického automatu má každý stav právě jednu direktivu pro každé písmeno jazyka automatu. Pokud je automat postaven na jazyce $\Sigma = \{a,b\}$ znamená to, že každý stav bude mít právě 2 direktivy – pro písmeno ‚a' a pro písmeno ‚b'. Deterministický automat také nesmí obsahovat epsilon direktivy.

2.4.2. Nedeterministické

V případě nedeterministického automatu může mít každý stav libovolný počet direktiv v rámci abecedy automatu. Znamená to, že v případě jednoho podnětu se může aktivovat libovolný počet stavů. Nedeterministické automaty mohou využívat také epsilon-kroky. epsilon krok se aktivuje při nabuzení stavu, který ho obsahuje, není tedy přímo vázaný na vnější podnět.

2.4.1. S výstupem

Tyto speciální typy automatů se liší tím, že vrací textový nebo číselný výstup při vnějším podnětu. Hlavní zástupci, kterými se zabývám, jsou Mealyho a Moorův automat.

U Moorova automatu každý stav obsahuje určitý výstup, který se provede po nabuzení tohoto stavu.

U Mealyho automatu každá direktiva obsahuje určitý výstup, který se provede při použití této direktivy.

2.5. Turingův Stroj

Turingův stroj je zařízení, které pracuje se vstupní páskou, vnitřními stavy a předpisy chování. Turingův stroj pracuje tak, že přečte znak na pásce a podle toho, v jakém je stavu, pásku na daném místě přepíše nebo nepřepíše jiným znakem. Poté posune nebo neposune čtecí hlavu doprava nebo doleva. Takto pracuje do té doby, dokud nedosáhne konce pásky a neukončí výpočet. V případě, že se stroj zastaví v průběhu výpočtu, znamená to, že vstupní slovo není automatem rozpoznáno. Mechaniky Turingova stroje nejsou zahrnuty v programu AutoGen, pro jednoduché simulace těchto strojů je však možné použít některý z alternativních nástrojů popsaných dále.

3.Úvod do programu AutoGen

3.1. Úvod do programu

AutoGen je grafický výukový nástroj pro oblast Teorie konečných automatů, je napsaný v objektově orientovaném programovacím jazyku C# z rodiny Microsoft .NET. Hlavní důraz je kladen na práci s automaty pomocí matice direktiv a na pokrytí hlavních disciplín běžné výuky. Pomocí programu AutoGen je možné navrhnout a provádět simulace na deterministických nebo nedeterministických automatech. AutoGen dále podporuje provádění složitých operací na slovech i celých automatech, což umožňuje studentovi lépe pochopit tyto metody, popřípadě ověřit správnost svých výsledků. Simulační jádro AutoGenu podporuje také práci s epsilon-kroky včetně jejich kaskád. Mimo grafické zobrazení simulací je možné použít i dodatečnou zvukovou reprezentaci jednotlivých aktivních stavů a to v polyfonním i sekvenčním režimu. Program je distribuován s bohatou

paletou ukázkových automatů pro použití při výuce i domácím studiu, stejně jako s řadou vestavěných skriptů pro operace na slovech.

3.2. Operace se slovy

AutoGen nativně podporuje provádění operací na slovech. V této kapitole se zaměříme na tento systém, vysvětlení jednotlivých přiložených operací a v neposlední řadě na postup při vytváření vlastních operací.

3.2.1. Systém operací

Hlavním cílem při vývoji bylo vytvořit modulární skriptovací nadstavbu bez obětování kompatibility se staršími verzemi .NET Frameworku a přitom umožnit uživateli co nejširší škálu použitelných konstruktů. Systém i skriptovací jazyk je tedy rovněž založen na jazyku C# se vším, co nám tento jazyk může nabídnout. Při spuštění systému se přeparsují všechny moduly ve složce `functions` a uživatel vidí v programu seznam dostupných funkcí. Uživatelská operace může obsahovat jakýkoliv počet operací, včetně nadstandardních C# konstruktů, uživatelský dotaz tedy nemusí být pouze jednořádkový, ale je zpracováván jako samostatný skript. Výstup se provede buď přiřazením do vestavěné proměnné `result` pro výpis do programového logu, nebo použitím funkce `message(text)` pro výstup do standardního dialogového okna.

Při exekuci dotazu se provedou následující akce:

1) Vytvoří se dynamické kompilační prostředí
2) Zavede se podpora těchto modulů, které uživatel může volně používat:
`System.Collections.Generic;System.Data;System.Windows.Forms;System.IO`
3) Zavedou se všechny dostupné moduly operací
4) Uživatelský dotaz se zavede ve formě metody `main()` virtuálního prostředí
5) Virtuální aplikace se spustí a výsledky a chybová hlášení jsou poslány zpět

3.2.2. Obecná syntaxe

Při zadávání dotazu platí všechna pravidla aplikovatelná na jazyk C#, je tedy třeba dodržovat datové typy a základní syntaxi. Tato pravidla nejlépe pochopíte na jednoduchých ukázkách dotazů.

V první ukázce chci získat reverzní slovo ke slovu `brk`, tedy `krb` (obr001)

```
result = reverse("brk");
```

V druhé ukázce uvidíte skládání více operací. Chci získat 3 první znaky z reverzního slova `atrap`, tedy `par`. (obr002)

```
result = cut( reverse("atrap") , 0, 3);
```

V poslední složitější ukázce naplno využijeme jazyka C# a zadáme dotaz v podobě delšího skriptu s využitím pomocných proměnných. Chci vybrat všechna slova ze seznamu `seznam`, která obsahují písmeno `S`. Dotaz vrátí `slon, písmeno`. (obr003)

```
string[] seznam = {"slon", "text", "písmeno", "citace"};
for(int i=0; i<seznam.Length; i++)
{
   if( num_count(seznam[i], "n") > 0 )
      result += (result == "") ? seznam[i] : ", " + seznam[i];
}
```

3.2.3. Tvorba vlastních funkcí

Kromě operací dodávaných spolu s programem, je možné napsat svůj vlastní skript operace. Vše, co potřebujete, je alespoň minimální znalost jazyka C# a textový editor typu Notepad. Všechny moduly operací jsou standardně uschovány ve složce functions. Vytvořte zde nový

textový soubor a pojmenujte ho například `mojeOperace.fce`. Je důležité, aby soubor měl koncovku `.fce`.

První řádka modulu musí obsahovat hlavičku funkce tak, jak se ukáže v seznamu operací v programu AutoGen. Zbytek modulu je obyčejná statická metoda s návratovým typem `string`.

Jako ukázku si vezmeme operaci `left(string word,int length)`, kde je světlejším pozadím vyznačena hlavička modulu a tmavým funkční metoda operace. (obr004)

```
left(string word, int length)
static string left(string word, int length)
{
  return word.Substring(word.Length - length, length);
}
```

Jako nepsané pravidlo: operace s návratovým typem `int` by měly mít v názvu prefix `num_` a hlavička modulu by se měla shodovat s názvem a parametry její metody.

3.3. Správa automatů

Program AutoGen nativně podporuje vytváření, ukládání a načítání automatů ve vlastním formátu AGA (AutoGen Automat). Tento formát drží informace o matici direktiv, jednotlivých stavech automatu, direktivy a epsilon-kroky každého stavu. Formát AGA také uchovává grafické souřadnice jednotlivých stavů pro použití v grafickém simulátoru. Uložené automaty můžete použít nejen v simulátoru, ale také k provádění operací na automatech, jako třeba synchronně paralelní kompozice dvou automatů.

3.3.1. Tvorba nového automatu

Vytváření automatu v AutoGenu je, na rozdíl od konkurenčních aplikací, založeno na definování matice direktiv a ne na grafickém skládání. Před tvorbou nového automatu potřebujeme znát počet písmen a stavů (obr005). V druhém kroku se zobrazí formulářová reprezentace matice (obr006). První řádka tabulky definuje písmena pro jednotlivé sloupce

matice a první sloupec tabulky určuje, které stavy automatu budou přijímací. Tělo tabulky odpovídá klasické matici direktiv s řádky jako stavy a sloupci jako písmeny, každá buňka tedy obsahuje sadu direktiv, které se provedou, pokud v daném stavu přijde na řadu dané písmeno. Protože AutoGen podporuje deterministické automaty pouze genericky (neověřuje jejich deterministickou konzistenci), je možné v matici direktiv použít také epsilon-kroky (obr007).

Klasické direktivy se do tabulky zadávají jako soubor stavů oddělených čárkami a to ve formátu `QX` nebo zkráceném `X`. Pro epsilon-kroky nezáleží na umístění ve sloupci, pokud máme zvolenou abecedu `w:{a, b}, můžeme` epsilon-direktivu přidat buď do direktiv pro písmeno `a` nebo `b` se shodným efektem. Formát zápisu pro epsilon-kroky je `eX`.

Po zadání matice direktiv je třeba přejít do modu `Arrange & Save`, tedy do grafického aranžéru (obr008). V něm se zobrazí lineárně uspořádané stavy automatu, které je třeba táhnutím myši rozmístit na požadované pozice. Tato grafická reprezentace automatu je použita k vizuálnímu zobrazení v simulátoru a je součástí formátu AGA. Při zpětné editaci automatu se však vizuální informace ztratí z důvodu nemožnosti ověření konformity s nově upravenou maticí direktiv.

Jakmile je automat naaranžován podle našich potřeb, kliknutím na `Save as` se zobrazí dialogové okno pro uložení souboru AGA automatu. Pokud je třeba provést dodatečné změny v matici direktiv, je možné použít funkci `Refine matrix`, veškeré informace o vizuální pozici stavů budou při další editaci matice ztraceny.

3.3.2. Úprava stávajícího automatu

Jediná změna oproti vytváření nového automatu spočívá v tom, že místo funkce `New` použijeme funkci `Load` a načteme dříve uložený automat ze souboru AGA. Okamžitě uvidíme matici direktiv pro náš automat, ale jak již bylo vysvětleno, grafické rozložení automatu se v tomto případě ztratí a je třeba znovu použít aranžér. Náš soubor automatu zůstane beze změn, dokud neprojdeme fází aranžéru a v dialogu `Save as` nezvolíme

přepsání starého automatu, až po tomto kroku jsou provedené změny nezvratné. Funkci úprav je tedy bezpečně možné využít i pro rychlé nahlédnutí na matici direktiv.

3.4. Mealyho a Moorovy automaty

Program AutoGen mimo běžných deterministických a nedeterministických konečných automatů obsahuje také nativní podporu pro prvky Mealyho a Moorova automatů včetně hybridů mezi všemi podporovanými typy automatů. Je tedy možné vytvořit automat, který bude využívat kostru nedeterministického automatu s epsilon-kroky a přitom bude obsahovat informace o zpětné vazbě jak stavů, tak direktiv.

3.4.1. Podpora Moorových automatu

Nativní Moorův automat je podle definice schopný zpětné vazby při nabuzení stavu. Každý stav může mít definován textový výstup (znak nebo slovo), který automat vrátí v případě nabuzení tohoto stavu. AutoGen plně podporuje nativní Moorův automat, je ale možné některé zpětné vazby ponechat prázdné a vytvořit tak hybridní chování, kde pouze některé stavy vrací Mealyho hodnoty. V průběhu simulace uvidíte výstup zpětné vazby Moorových prvků v oblasti konzole.

3.4.2. Přidání Moorových prvků při tvorbě automatu

Pravidla při tvorbě Moorova automatu jsou stejné, jako u běžného automatu. Pouze v editoru matice direktiv vyberte volbu `Specify Moore Sequence`, ke každému stavu vyplňte hodnotu jeho zpětné vazby, nebo nechte pole prázdné (obr009). Nakonec klikněte na tlačítko `OK` a pokračujte k uložení automatu jako obvykle.

3.4.3. Podpora Mealyho automatů

Nativní Mealyho automat je podle definice schopný zpětné vazby při použití direktivy. Každá direktiva může mít definován textový výstup (znak nebo slovo), který automat vrátí v případě použití této direktivy. AutoGen plně podporuje nativní Mealyho automat, je ale možné

některé zpětné vazby ponechat prázdné a vytvořit tak hybridní chování, kde pouze některé direktivy vrací Mealyho hodnoty. V průběhu simulace uvidíte výstup zpětné vazby Mealyho prvků v oblasti konzole.

3.4.4. Přidání Mealyho prvků při tvorbě automatu

Pravidla při tvorbě Mealyho automatu jsou stejné, jako u běžného automatu. Je možné využít epsilon-kroky, ale epsilon-krok nesmí obsahovat Mealyho zpětnou vazbu. Zpětná vazba se definuje pomocí znaku lomítka za číslem stavu, do kterého direktiva směřuje. Při použití více zpětných vazeb u nedeterministického hybridního automatu dojde ke složení písmen v pořadí výskytu v direktivě (obr010).

Příklady standardních direktiv:

```
1
0, 1, 3
0, e1
```

Možná použití Mealyho zpětné vazby:

`1/A` – napíše „A“ pokud dojde na tuto direktivu

`0, 1/R, 3` – napíše „R“ pokud dojde na tyto direktivy

`0/R, 1/T, 3/Z` – napíše složení „RTZ“ pokud dojde na tyto direktivy

`0/RTZ, 1, 3` – napíše slovo „RTZ“ pokud dojde na tyto direktivy

`0, e1/R` – chybné, epsilon-krok nesmí obsahovat Mealyho zpětnou vazbu

Po zadání zpětných vazeb pokračujte k uložení automatu jako obvykle. Je možné simultánně použít Moorovy zpětné vazby.

3.5. Simulace automatů

AutoGen obsahuje modul pro testování a prezentaci automatů zvaný Simulátor. Tento modul dokáže použít dříve uložený automat ve formátu AGA, určit jeho počáteční stavy a simulovat na něm zpracování slova. Samotná simulace poté využívá textový, grafický a zvukový výstup pro názorné zobrazení procesu.

3.5.1. Načtení automatu

Jak již bylo řečeno, simulátor funguje jako platforma pro „rozhýbání“ námi vytvořeného automatu, jako první krok tedy zvolíme funkci `Load automat` pro načtení požadovaného souboru AGA. Po načtení se okamžitě vykreslí grafická reprezentace našeho automatu, ve které jsou všechny stavy neaktivní.

3.5.2. Layout prostředí simulátoru

Grafické rozložení okna simulátoru (obr011) se dělí na 4 hlavní oblasti, těmi jsou Konzole, Ovládací prvky, Grafická prezentace a Stavová oblast. Funkce jednotlivých oblastí vysvětlím v následujících odstavcích.

Konzole zaznamenává všechny podstatné operace simulátoru v podobě textu. V této oblasti můžete vidět průběh načítání automatu a tvoření simulačního obalu pro automat, v průběhu simulace konzole vypisuje přijmutí/nepřijetí jednotlivých znaků slova, stejně jako konečný stav automatu. Konzole vypisuje také chybová hlášení v případě nekonzistentních automatů nebo nesmyslných vstupů pro inicializaci simulátoru. Konzole slouží také pro výstup zpětných vazeb Mealyho a Moorova automatů.

Ovládací prvky jsou textová pole a tlačítka ovládající vytvoření simulačního obalu pro automat a samotný průběh simulace.

- Textové pole `List starting states` obsahuje seznam počátečních stavů oddělených čárkami. V základním nastavení se jako startovní stav uvažuje Q0.
- Textové pole `Word to be parsed` obsahuje slovo, které chcete na automatu simulovat, v případě inkonsistence vstupu (zadání znaků mimo abecední doménu automatu atp.) budete na tento fakt upozorněni při vytváření obalu. Pokud nezadáte žádné slovo, simulátor uvidí vstup jako epsilon a tedy pouze provede nasazení do počátečních stavů.
- Další ovládací prvek je tlačítko `Create Simulator`, které efektivně vytvoří simulační obal pro současný automat, tento obal obsahuje data automatu, počáteční stavy a simulované slovo. Mějte na paměti, že při změně počátečních stavů, nebo slova je třeba

znovu použít toto tlačítko pro vytvoření nové verze simulačního obalu, do té doby platí staré nastavení.

- Tlačítka `STEP` a `FINISH` řídí samotné kroky simulace. Zatímco `STEP` nařídí simulátoru zpracovat jedno další písmeno ze zadaného slova, `FINISH` postupně provede všechny kroky až do zpracování celého slova.

Grafická prezentace je oblast, která ukazuje dynamicky se měnící vizuální reprezentaci simulovaného automatu. Přijímací stavy jsou vyznačeny dvojitým obrysem oproti nepřijímacím. Jednotlivé direktivy jsou zobrazeny jako čárkované šipky s vyznačenými písmeny, pro které tato direktiva platí. epsilon kroky jsou zobrazené jako tečkované šipky označené velkým písmenem epsilon. Smyčky nebo chcete-li „ouška“, kdy direktiva směřuje zpět do počátečního stavu, jsou vyznačeny jako čárkované smyčky u stavu směřující vždy od těžiště automatu a samozřejmostí je seznam písmen, pro která smyčka platí. Aktivní stavy jsou v průběhu simulace označeny výraznou barvou.
Jednotlivé barvy v programu jsou snadno změnitelné a v dalších verzích programu AutoGen se mohou měnit, proto uvádím pouze obecnou grafickou reprezentaci bez explicitního použití barev.

Stavová oblast reprezentuje aktuální stav aktivního obalu simulátoru, obsahuje ukazatel průběhu představující průběh zpracování daného slova a textovou oblast s momentálním stavem simulační vrstvy.

3.5.3. Provedení jednoduché simulace

K provedení simulace potřebujeme mít otevřený automat, nastavené počáteční stavy a zvolené slovo. Jako první je třeba vytvořit simulační obal kolem otevřeného automatu kliknutím na `Create Simulator`. V grafické prezentaci se zvýrazní počáteční stavy jako aktivní. Dále máme dvě možnosti, použít funkci STEP pro krokování simulace po jednotlivých písmenech, nebo použít FINISH pro skok na konec simulace. Po každém kroku se grafická prezentace překreslí, aby zrcadlila aktuální stav automatu uvnitř simulátoru.

3.5.4. Rychlý náhled matice direktiv

Jakmile načteme automat do simulátoru, můžeme kdykoliv rychle zobrazit matici direktiv do oblasti grafické prezentace a to funkcí `Visualize matrix`, matici skryjeme funkcí `Hide matrix`.

3.5.5. Zvuková reprezentace

Simulační modul podporuje také režim zvukové reprezentace aktivních stavů a to v sériovém nebo paralelním (polyfonním) režimu. Zvukovou podporu aktivujete funkcí `Enable sound`, budete dotázání, zdali chcete zvuky přehrávat polyfonně. Zvukovou podporu deaktivujete opětovným použitím stejné funkce.

Pokud nyní budete provádět simulaci, zvukový modul přiřadí ke každému stavu jeden z 21 možných tónů a při každém kroku simulace zahraje tóny aktivních stavů. V sériovém režimu se zvuky aktivních stavů přehrají postupně jeden po druhém, kdežto v paralelním režimu se přehrají všechny aktivní stavy najednou a vytvoří tak akord.

V průběhu hraní tónu není možné provést další krok simulace. Podpora zvuku vyžaduje přítomnost knihoven DirectX 9.0c v systému a je plně kompatibilní pouze s operačními systémy Windows XP, Vista, 7 a novějšími, v případě nekompatibility by nemělo dojít k pádu aplikace.

3.6. Operace s automaty

Program AutoGen podporuje mimo simulací i provádění složitých operací na jednom nebo více automatech, jako je synchronně paralelní kompozice a převod nedeterministického automatu na deterministický včetně optimalizací výsledku.

3.6.1. Layout prostředí operací

Prostředí operací na automatech se skládá ze 4 hlavních částí. 3 šedivé boxy představují 3 sloty pro automaty, 2 menší sloty v horní polovině slouží pro vstupní automaty A a B, spodní velké pole slouží pro vyzvednutí výsledku operace. Prostřední rozbalovací pole obsahuje seznam všech vestavěných operací, každá operace má v závorce formálně naznačeno, kolik automatů je potřeba na její provedení (obr012).

3.6.2. Provedení jednoduché operace

Jako první musíme určit vstupní automat(y). To provedete kliknutím na šedivý box, po kterém se zobrazí zjednodušená verze manažera automatů. Zde můžete buď vytvořit nový automat a ten použít, nebo načíst již připravený automat funkcí `Load`. Nově vytvořený automat by byl pouze dočasný a ztratí se po zavření okna operací, proto doporučuji si automat před operacemi připravit v klasickém správci automatů.

Pokud pro požadovanou operaci potřebujete dva automaty, udělejte to samé i s pravým boxem pro Automat B. Pokud naplníte oba sloty, ale provádíte operaci, která vyžaduje pouze jeden automat, použije se vždy pouze Automat A.

Jako druhý krok musíme vybrat požadovanou operaci z rozbalovacího seznamu, poté již stačí pouze kliknout na `Final Automat` slot ve spodní části okna a otevře se nám manažer s maticí direktiv a možností aranžovat a uložit výsledný automat.

3.6.3. Možné operace

AutoGen dokáže nad automaty provést tyto operace:

- Sériová Kompozice
- Synchronně Paralelní Kompozice (4 varianty určující složení přijímacích stavů)
- Co-Automat (reverze přijímání stavů)
- Reverzní automat
- Převod nedeterministického automatu na deterministický s optimalizací redundantních stavů

4. Dodávané příklady

4.1. Vestavěné a přiložené operace se slovy

AutoGen má již v základní distribuční verzi celou řadu operací, které je možná ihned používat i bez znalosti programování. V některých funkcích jsou použity indexy pro vymezení části slova. Index je počítán, jako pomyslná mezera mezi písmeny ve slově, kde před prvním písmenem je index 0 a za posledním písmenem je index rovný počtu písmen ve slově. Některé operace mají dvě varianty podle návratového typu. Operace, které vracejí číselný návratový typ `int`, mají v názvu prefix `num_` a lze je tedy bez převodu využít pro další výpočty.

4.1.1. Vestavěné funkce

`message(string text)`

- vypíše text do dialogového okna

`result`

- jakýkoliv text v této proměnné bude po dokončení skriptu vypsán do konzole

4.1.2. Přiložené funkce

`concat(string word1, string word2)`

- spojí dvě slova a vrátí výsledné slovo (zřetězení dvou slov)

`cut(string word, int start, int stop)`

- vrátí část slova mezi startovním a cílovým indexem (výřez ze slova)

`pow(string word, int times)`

- umocní slovo daným počtem a vrátí výsledné slovo (mocnina slova)

`pump(string word, int start, int stop, int times)`

- umocní část slova mezi startovním a cílovým indexem daným počtem

- část slova mimo vybraný interval je vrácena beze změn („napumpování“ slova)

`reverse(string word)`

- vrátí reverzní slovo (reverze slova)

`count(string word, string pattern)`

- vrátí slovo obsahující počet výskytů vzoru uvnitř daného slova (počet výskytů podslova)
- pokud potřebujete číselný návratový typ, je možné použít prefix `num_`

`length(string word)`

- vrátí slovo obsahující délku daného slova
- pokud potřebujete číselný návratový typ, je možné použít prefix `num_`

`left(string word, int length)`

- vrátí určitý počet znaků na začátku slova (získání prefixu)

`right(string word, int length)`

- vrátí určitý počet znaků na konci slova (získání sufixu)

4.2. Přiložené automaty

Program AutoGen je distribuován s malou sbírkou doprovodných automatů pro jednotlivé kapitoly Teorie automatů. Tyto příklady jsou rozdělené do několika složek pojmenovaných podle kapitol. Složka každé kapitoly obsahuje také textový dokument s popisem jednotlivých automatů.

5. Alternativní řešení

Jako součást práce jsem se rozhodl uvést také některé alternativní aplikace, které mohou sloužit ke studijním účelům a prezentacím. Zaměřil jsem se pouze na volně dostupná freeware řešení.

5.1. Automaton Simulator (Carl Burch, 2001)

Tento jednoduchý, volně dostupný nástroj poskytuje jen základní funkcionalitu. Podporuje pouze visuální modelování automatu bez možnosti náhledu na matici direktiv. Lze ho použít pro tvorbu deterministických a nedeterministických konečných automatů a Turingových strojů pro jednoduché simulace. Program podporuje epsilon-kroky, textové komentáře v automatu a poskytuje možnost tisku grafického náhledu. Program je napsaný v jazyce Java.

5.2. Visual Automata Simulator (Jean Bovet, 2006)

Toto řešení poskytuje velice zdařilé visuální modelování automatů, podporu epsilon-kroků, možnost převodu nedeterministického automatu na deterministický, práce s více otevřenými automaty. Pokud však chceme textový výpis matice direktiv, musíme se spokojit s ne příliš přehledným seznamem přechodů. VAS obsahuje také velice povedenou nativní podporu Turingova stroje. Program je napsaný v jazyce Java.

6.Závěr

Všechny cíle, které jsem si na začátku projektu položil, jsem se pokusil splnit. Program je sice stále ve fázi beta verze, ale již nyní je úctyhodnou konkurencí k ostatním volně dostupným aplikacím, jednak díky nadstandardním modulům pro operace se slovy a automaty a také díky snaze dívat se na automat nejen jako na obrázek, ale také jako na sadu stavů a direktiv s rozšiřujícími funkcemi. Díky nativní implementaci epsilon kroků, Mealyho a Moorova automatu se podařilo vytvořit prostředí takřka bez omezení, které umožňuje striktně se držet látky, nebo využít kreativitu studentů s možností tvořit zajímavé hybridní automaty. Stejně tak ověření správnosti složité synchronně paralelní kompozice, nebo převedení nedeterministického automatu na deterministický je s využitím mocných algoritmů AutoGenu otázkou vteřin.

7. Použité zdroje

Automaton Simulator homepage, [online]. [4. 4. 2009].
Dostupné na www:
http://ozark.hendrix.edu/~burch/proj/autosim/

Lawson, M., V. **Finite Automata**, [online]. [cit. 2008-04-20].
Dostupné na WWW:
http://www.ma.hw.ac.uk/mark1/preprints/Lawson.pdf

Implementing Deep Cloning via Serializing objects, [online]. [12. 1. 2009].
Dostupné na www:
http://www.codeproject.com/KB/tips/SerializedObjectCloner.aspx

Jančar, P. **Informace o kurzu Teorie jazyků a automatů**, [online]. [2008-04-20].
Dostupné na WWW:
http://www.cs.vsb.cz/jancar/TJAA/tjaa.2004.htm

Visual Automata Simulator homepage, [online]. [4. 4. 2009].
Dostupné na www:
http://www.cs.usfca.edu/~jbovet/vas.html

Černá, O., Křetínský, M., Kučera, A**. Automaty a formální jazyky I.** [online]. [2008-04-01].
Dostupné na WWW:
http://www.fi.muni.cz/usr/kretinsky/afj_I.ps

Chytil, M. **Automaty a gramatiky**.
Praha: SMTL, 1984. ISBN 04-012-84.

Grace, R. **Hudba a zvuk na počítači**.
Praha: Grada, 1999. ISBN 80-7169-519-X.

Hopcroft, J.E, et all. **Incroduction to Automata Theory, Languages, and Computation.**
Reading. Addison-Wesley, 2001. ISBN 0-201-44124-1.

8. Obsah přiloženého média

Obrázky

Obsahuje všechny obrázky, na které odkazuje tato práce.

AutoGen

Obsahuje aktuální funkční verzi programu AutoGen včetně vestavěných příkladů.

Práce

Obsahuje tuto práci ve formátu PDF.