PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Stavové automaty

2012 Petr Jančík

Anotace

Tento text je doprovodný dokument ke grafickému editoru stavových automatů s výstupem. Obsahuje hlavně popis uživatelského rozhraní a vnitřního fungování programu. Program je napsán v jazyce C# v prostředí Microsoft Visual Studio 2008.

Chtěl bych poděkovat rodině za psychickou podporu a vedoucímu práce RNDr. Arnoštu Večerkovi za pomoc při vypracovávání.

Obsah

Seznam obrázků

1. Úvod

Stavový automat, zkráceně KA (konečný automat) nebo FSM (finite state machine) je matematická abstrakce, používaná pro studium teorie vyčíslitelnosti a formálních regulárních jazyků, které jej popisují. Výstupem takového automatu je buď "přijal" pokud skončí v konečném stavu nebo "nepřijal". Teorie stavových automatů vychází z definice automatů o konečném počtu stavů, kterou položili Warren S. McCulloch a Walter S. Pitts v roce 1943.

Stavové automaty s výstupem jsou rozšířením tohoto modelu, kde výstup přijal/nepřijal je nahrazen výstupní abecedou a výstupní funkcí, která určuje, jaký bude výstup. S těmito rozšířeními přišli pánové George H. Mealy a Edward F. Moore v letech 1956 a 1955, po kterých jsou tyto automaty pojmenovány. Automaty lze využívat pro návrh parserů formálních jazyků, šifrovacích strojů, popis lidských jazyků a dalších. V dnešní době je důležitý rychlý návrh programů, a proto je vhodné mít nástroje, jako tyto automaty, které nám umožňují tento proces urychlit. Nástroje pro vývoj těchto automatů by měly umožňovat simulaci průběhu výpočtu automatu a nabízet ergonomické a efektivní rozhraní pro jeho sestavení.

Můj přístup zahrnuje navrhnout grafický editor, který umožňuje návrh takovýchto strojů a simulaci průběhu jejich výpočtu. Jádro tvoří pracovní plocha, na které lze rychle a přehledně vytvořit stavy, přechody automatu a určit počáteční stavy výsledného automatu. Program obsahuje možnost zadat vstupní slovo a sledovat chování automatu na jeho přechodovém diagramu. Tento editor výrazně usnadní návrh programů založených na stavových automatech a usnadní tak efektivitu práce.

1.1. Formální definice

1.1.1. Mooreův stavový automat

Mooreův stavový automat s výstupem je šestice $(S, S0, \Sigma, \Lambda, T, G)$ kde:

- S je neprázdná konečná množina stavů
- $S0$ je počáteční stav z množiny S
- Σ je konečná vstupní abeceda
- Λ je konečná výstupní abeceda
- T je přechodová funkce ($T\colon S \times \Sigma \to S$) další stav závisí na předchozím stavu a znaku na vstupu
- G je výstupní funkce ($G\colon S \to \Lambda$) výstup závisí na stavu, ve kterém se automat nachází

1.1.2. Mealyho stavový automat

Mealyho stavový automat s výstupem je šestice $(S, S0, \Sigma, \Lambda, T, G)$ kde:

- S je neprázdná konečná množina stavů
- $S0$ je počáteční stav z množiny S
- Σ je konečná vstupní abeceda
- Λ je konečná výstupní abeceda
- T je přechodová funkce ($T\colon S \times \Sigma \to S$)
- G je výstupní funkce ($G\colon S \times \Sigma \to \Lambda$) výstup závisí na stavu, ve kterém se automat nachází a na vstupním znaku

1.2. Zadání bakalářské práce

Cílem práce je sestavit aplikaci pro návrh stavových automatů (Mealyho automat, Mooreův automat). Návrh by měl být graficky (sestavením diagramu automatu) nebo formálním popisem automatu. Další funkcí aplikace by měla být simulace činnosti sestaveného automatu zobrazující jednotlivé kroky přijímání vstupního slova (přechody mezi stavy) a zobrazení výstupu automatu.

2. Uživatelská část

2.1. Požadavky na spuštění

Ke spuštění aplikace je potřeba Windows XP, Vista nebo Windows 7 s nainstalovanou .NET Framework 3. Pro nápovědu je třeba mít také nainstalován Adobe Reader.

Windows 7 obsahuje .NET Framework 3.5 a Windows Vista obsahuje .NET Framework 3, tudíž jen uživatelé Windows XP si ji musí nainstalovat z přiloženého DVD nebo stáhnout nejnovější ze stránek Microsoftu. Adobe Reader je také na DVD, nebo na stránkách Adobe.com.

Aplikace se spouští souborem `StateAutomata.exe`. Pro nápovědu je potřeba mít v umístění souboru také tento dokument.

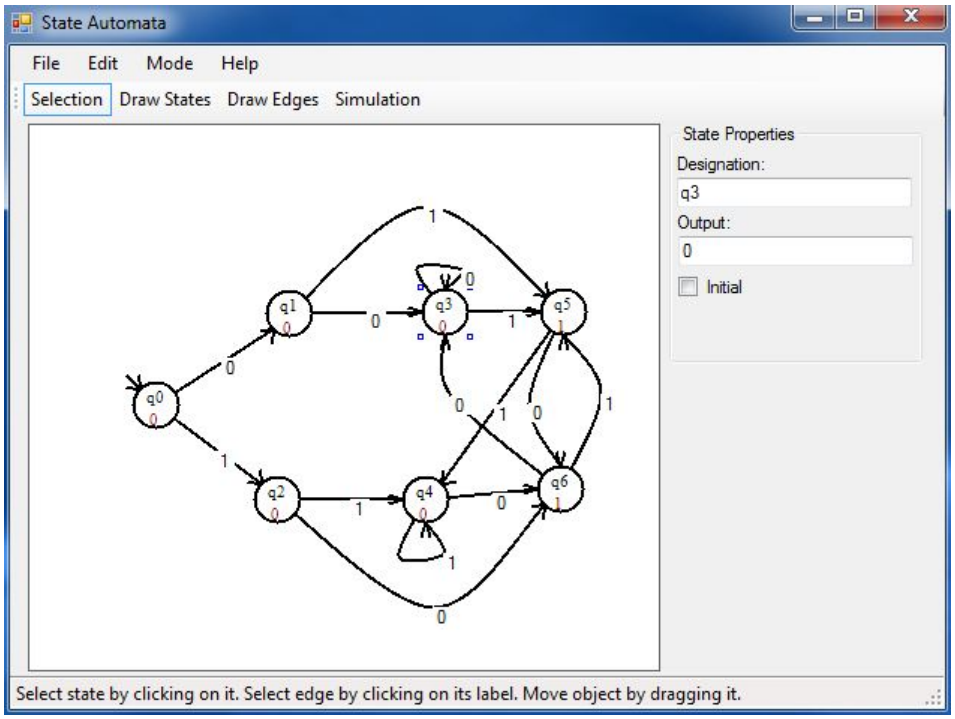
2.2. Popis programu

Tento program slouží k designu stavových automatů s výstupem. Umožňuje pohodlné a přehledné sestavení Mooreova nebo Mealyho stavového automatu a definici vstupních a výstupních znaků. Designované automaty je možné kdykoli načíst nebo uložit.

2.3. Ovládání

2.3.1. Popis okna

- *Menu*
  V horní části okna je hlavní menu aplikace.
  - V položce File jsou možnosti vytvoření nového Mooreova nebo Mealyho automatu, možnost načíst automat ze složky disku. Dále umožňuje uložit automat buď pod stávajícím názvem, pokud už byl jednou uložen, nebo s novým názvem. Poslední položka slouží k uzavření aplikace.
  - V položce Edit je možno smazat označený stav/hranu v hlavním okně.
  - Položka menu Mode obsahuje možnosti módů aplikace.
  - Poslední položkou je Help, která obsahuje odkaz na tento dokument.
- *Nabídka módů*
  Pod menu se nachází snadno přístupná lišta s módy aplikace.
- *Panel nastavení objektu / ovládání simulace*



Obrázek 1. Uživatelské rozhraní, Mooreův automat

  - Pokud je označen objekt, tato nabídka zobrazuje a umožňuje měnit jeho vlastnosti.
  - Při označení stavu se zde objeví jeho název, možnost zvolit jej jako počáteční a pokud je právě designován Mooreův automat, tak i výstupní znak.
  - Při označení hrany se zobrazí její vstupní znak a pokud je automat Mealyho, tak také její výstup.
  - Do polí vstupu a výstupu je možné zadat libovolné znaky, pokud je zadán více než jeden znak, tak jsou všechny znaky mimo první ignorovány.
  - V módu simulace je tato oblast využita pro její ovládání. Je možné zadat vstupní slovo, po přečtení celého vstupu pak lze kopírovat výstupní slovo. Objeví se také 4 tlačítka pro posun na počátek simulace, krok zpět, krok vpřed a posun na konec simulace.

- *Pracovní plocha*
  Pracovní plocha zabírá v rozhraní největši prostor a je hlavním prvkem, se kterým uživatel pracuje. Podle módu aplikace umožňuje selekci objektů, přidávání stavů, hran a sledování průběhu simulace.

2.3.2. Módy aplikace

- SELECTION - umožňuje vybírat objekty.
  Vybrané objekty lze posouvat tažením myší a měnit jejich vlastnosti. Stav se označuje kliknutím na něj, hrana pak kliknutím na její popisku. Odznačení se provádí klikem pravého tlačítka do pracovní plochy.

- DRAWSTATES - umožňuje přidávat stavy k automatu.
  Stav se přidává kliknutím na volné místo na pracovní ploše. Je možné ihned měnit vlasnosti přidaného stavu. Při kliknutí na existující stav se daný stav označí.

- DRAWEDGES - umožňuje vytvářet přechodové hrany mezi stavy automatu. Kliknutím na existující stav se začne vytvářet hrana z tohoto stavu. Při vytváření hrany lze přidávat další body kliknutím na volnou plochu. Přidání stavu je dokončeno opětovným klikem na některý ze stavů. Kliknutím pravým tlačítkem do plochy lze zrušit právě zakreslovanou hranu.

- SIMULATION - v tomto módu lze simulovat výpočet automatu.

2.3.3. Klávesové zkratky

- Delete - Smaže označený objekt.

- F1 - Zobrazí tento dokument.

- Ctrl+1 - Přepne do módu SELECTION.

- Ctrl+2 - Přepne do módu DRAWSTATES.

- Ctrl+3 - Přepne do módu DRAWEDGES.

- Ctrl+4 - Přepne do módu SIMULATION.

- Ctrl+S - Uloží automat do souboru.

- Ctrl+O - Otevře automat ze souboru.

2.3.4. Průvodce

Při spuštění aplikace se automaticky vytvoří nový Mooreův automat. Pokud požadujeme jiný typ automatu, lze jej vybrat v menu File nebo můžeme načíst již hotový automat z disku.

Program začíná v módu `Selection`. Klikem na lištu módů nebo v menu Mode si vybereme mód kreslení stavů (`Draw States`) a na ploše vytvoříme stavy, které by měl námi požadovaný automat mít. U každého stavu nastavíme jeho označení a u Mooreova stroje také výstupní znak stavu. Stav je možno označit klinutím na něj a posouvat tažením na volnou pozici.

Poté se přepneme do módu kreslení hran (`Draw Edges`), a mezi stavy vytvoříme hrany. Lze vytvářet přímé hrany nebo při vytváření hrany pomocí klikání na prázdnou plochu přidat další body a vytvořit tak zaoblené hrany. U hran nastavíme vstup, při kterém bude hranou procházet výpočet. U Mealyho stroje navíc přidáme výstupní znak při průchodu hranou. Hrany je možné označit kliknutím na jejich popisku. Pokud není hrana přímá, pak se při označení objeví body, kterými lze tvarovat výslednou křivku.

Nyní, pokud jsme s automatem spokojeni, můžeme přejít do stavu simulace, jinak doplníme další stavy a hrany. V režimu simulace zadáme vstupní slovo a tlačítkem `>` provedeme první krok výpočtu. Pokud nechceme krokovat výpočet a rádi bychom zjistili, jaký bude výstup po skončení výpočtu, klikneme na tlačítko `>>` které nás přesune na konec simulace. Pokud je v automatu chyba, například chybí hrana pro daný vstup, výpočet nebude moci pokračovat a zůstane v posledním korektním stavu. Nyní můžeme buď nahradit vstupní slovo jiným a začít od začátku, nebo se přesunout do editačních módů a chybu opravit. Pokud výpočet proběhl v pořádku, je možné si výstupní řetězec zkopírovat. Pokud chceme vyzkoušet jiný vstup, lze se tlačítkem `<<` vrátit na začátek a změnit vstupní slovo.

Jakmile jsme spokojeni s funkčností stroje, lze jej uložit v menu File na požadované místo na disk.

3. Programátorská část

Program je napsán v jazyce C# za použití standardních knihoven .NET.

3.1. Popis tříd a enumerací

Aplikace se skládá ze tříd pro reprezentaci stavů (`State`) a hran (`Edge`), které dědí ze společné třídy prvku automatu (`GraphElement`). Většina funkcionality se vyskytuje ve třídě uživatelského rozhraní (`Form1`),která se stará o udržování informací o automatu, změnu stavů a simulaci automatu a dědí z obecného formuláře `Form`.

Program obsahuje 2 enumerace:

- `MachineType`, která má hodnoty pro oba typy automatů, `Mealy` a `Moore`.
- `Mode`, je enumerace módů a obsahuje hodnoty `Selection, DrawStates, DrawEdges, Simulation` pro jednotlivé módy.

3.2. Popis jednotlivých metod a fieldů

3.2.1. class GraphElement

- Fieldy:
  - `char Output`
    Uchovává výstupní znak hrany/stavu.
  - `Point Location`
    Bod udávající souřadnice objektu.
- Metody:
  - `virtual public bool BelongsTo(Point point)`
    Virtuální metoda přepisovaná v potomkovi, která zjišťuje, zda se daný bod nachází uvnitř objektu.
  - `virtual public void Move(int X, int Y)`
    Virtuální metoda, která posouvá daný objekt na určené souřadnice.

3.2.2. class State

- Fieldy:
  - `Point Center`
    Udává střed kruhu, reprezentujícího stav na pracovní ploše.

  - `int Radius`
    Udává poloměr kruhu stavu.
  - `string Designation`
    Jméno stavu, zobrazené na pracovní ploše.
  - `List<Edge> EdgesFrom`
    Seznam hran s počátkem v tomto stavu, používáno převážně při simulaci.
  - `List<Edge> EdgesTo`
    Seznam hran s koncem v tomto stavu, používáno při přesunu.

- Metody:
  - `override public bool BelongsTo(Point point)`
    Zjišťuje, zda se bod nachází uvnitř kružnice stavu.
  - `override public void Move(int X, int Y)`
    Metoda, která posouvá střed stavu na novou pozici a posouvá koncové/počáteční body hran vedoucích do/ze stavu.

3.2.3. class Edge

- Fieldy:
  - `List<Point> Path`
    Body udávající hranu na pracovní ploše.
  - `State From`
    Počáteční stav hrany.
  - `State To`
    Koncový stav hrany.
  - `char Input`
    Vstupní znak pro přechodovou funkci.
  - `Label Label`
    Objekt, který zobrazuje označení hrany na pracovní ploše.

- Metody:
  - `override public bool BelongsTo(Point point)`
    Zjišťuje, zda se bod nachází uvnitř okrajů popisky hrany.
  - `override public void Move(int X, int Y)`
    Metoda, která posouvá tažený bod hrany na jeho novou pozici a ověří, zda jsou koncové šipky v optimálních pozicích vzhledem k nové pozici.

3.2.4. class Form1

Třída `Form1` obsahuje informace o programu.

- Fieldy:
  - `Mode ProgramMode`
    Udržuje informaci o módu editoru.
  - `MachineType MType`
    Udržuje informaci o typu právě editovaného automatu.
  - `SaveFileDialog save`
    Uživatelské rozhraní pro volbu umístění při ukládání souboru.
  - `OpenFileDialog load`
    Uživatelské rozhraní pro načítání automatu z disku.
  - `int Index`
    Index pro vytváření automatického jména stavu.
  - `Edge CurrentPath`
    Uchovává vytvářenou hranu, dokud není dokončena.
  - `List<State> States`
    Seznam stavů v editovaném automatu.
  - `List<Edge> Edges`
    Seznam hran v automatu.
  - `State InitialState`
    Počáteční stav automatu pro simulaci.
  - `State CurrentState`
    Stav, ve kterém se automat nachází v určitém kroku výpočtu.
  - `List<State> History`
    Historie stavů při simulaci, umožňuje návrat na počátek.
  - `string simulationInput`
    Kopie vstupního řetězce, používaná pro simulování.
  - `int simulationIndex`
    Index do simulationInput, kde se právě nachází výpočet.
  - `GraphElement SelectedObject`
    Obsahuje právě označený objekt na pracovní ploše. Při zvolení objektu se jejich daty vyplňují textová pole vpravo od plochy.
- Metody:
  - `void UpdateProperties()`
    Po označení objektu vypíše jeho informace do editovacích políček.

- `void ModeBarModeChanged(object sender, EventArgs e)`
  Obsluha události kliknutí na položku módu na panelu módů nebo v menu. Aktivuje mód daný objektem sender.
- `void ChangeMode(Mode mode)`
  Provádí vlastní změnu módu a nastavuje jeho důležité parametry.
- `void InitializeSimulation()`
  Při přechodu do módu simulace inicializuje tlačítka a vyprázdní pole vstupu a výstupu. Pokud není vybrán počáteční stav automatu, vybere první stav.
- `void DeselectObject()`
  Při kliku pravým tlačítkem do plochy zruší označení objektu a ukončí kreslení hrany.
- `void Form1_Resize(object sender, EventArgs e)`
  Při změně velikosti hlavního okna mění zároveň velikost pracovní plochy pro dynamické zvětšování.
- `void panel1_MouseClick(object sender, MouseEventArgs e)`
  Obsluhuje klik do pracovní plochy, pokud není program v módu simulace. Při kliku na stav jej označí, v módu kreslení hran pak buď začne hranu nebo ukončí právě kreslenou hranu. Při kliku do plochy v módu kreslení stavů zakreslí nový stav, v módu kreslení hran přidá bod k právě kreslené hraně, pokud nějaká je.
- `GraphElement MouseOverObject(Point point)`
  Vrátí element grafu, nad kterým se vyskytuje kurzor myši.
- `void panel1_Paint(object sender, EventArgs e)`
  Obsluha události překreslení pracovní plochy. Pro každou hranu spočítá přibližný směr vstupu do stavu, na průnik zakreslí šipku a zbývajícími body hrany proloží křivku. Pro každý stav zakreslí kružnici a napíše jeho označení a u Mooreova automatu i výstupní znak. Pokud je označen některý stav nebo hrana, tak kolem stavu nebo kolem označení hrany a hrany samotné zakreslí selekční útvary.
- `void panel1_MouseDown(object sender, MouseEventArgs e)`
  Slouží pro začátek tažení bodu hrany nebo stavu.
- `void panel1_MouseUp(object sender, MouseEventArgs e)`
  Pokud bylo taženo, tak umístí tažený objekt na místo pod kurzorem.
- `void SelectObject(Point Location)`
  Slouži pro označení objektu pod kurzorem.
- `void textBox1_TextChanged(object sender, EventArgs e)`
  Pokud se změní text v poli a nejsme v módu simulace, přeneseme novou hodnotu na označený objekt. V módu simulace se při změně textu povolí tlačítka pro postup vpřed.

- `void textBox2_TextChanged(object sender, EventArgs e)`
  Pokud se změní text v poli a nejsme v módu simulace, přeneseme novou hodnotu na označený objekt.
- `void InitialCheck_CheckedChanged(object sender, EventArgs e)`
  Obsluha události změny zatržení u checkboxu. Změní počáteční stav automatu na právě označený stav. Pokud je odznačen, tak zruší statut počátečního stavu z označeného.
- `void Forward()`
  Při stisku tlačítka > nebo >> na počátku simulace překopíruje vstupní slovo. Při stisku tlačítka > provede jeden krok automatu, při stisku >> zpracuje vstupní slovo až do konce.
- `void Back()`
  Při stisku tlačítka < se automat vrátí o jeden krok zpět v historii, při stisku tlačítka << se vrátí na počátek výpočtu.
- `void RecoverFromEror()`
  Slouží k resetu simulace při chybách neúplného automatu, prázdného slova, neznámých znaků na vstupu nebo startu simulace bez jediného stavu.
- `void DeleteEdge(Edge edge)`
  Smaže hranu a její označeni na pracovní ploše.
- `void DeleteState(State state)`
  Smaže stav a všechny hrany, které do něj vstupují nebo z něj vycházejí. Pokud byl stav počáteční, tak změní počáteční stav.
- `void saveToolStripMenuItem_Click(object sender, EventArgs e)`
  Při kliknutí na Uložit (Save) v menu uloží automat, pokud byl již jednou uložen. Pokud ne, otevře dialog výběru názvu a umístění.
- `void saveToolStripMenuItem_Click(object sender, EventArgs e)`
  Při kliknutí na Uložit jako (Save As) otevře dialog výběru názvu a umístění a uloží automat.
- `void SaveGame()`
  Uloží soubor na disk ve formátu XML.
- `void openToolStripMenuItem_Click(object sender, EventArgs e)`
  Pokud je soubor v pořádku, načte ho do pracovní plochy. V případě chyby vyprázdní stávající automat.
- `void NewMachine()`
  Vytvoří nový automat. Jeho typ je nastaven volající funkcí.

– `void ScrollbarHelper(State state)`
  Pomocná funkce, která při přidání stavu posouvá pomocný prvek (Label), aby se dalo při zmenšení pracovní plochy posouvat.
– `void ShowHelp()`
  Zobrazí tento PDF soubor jako nápovědu.

Závěr

Výsledkem této práce je naprogramovaný editor stavových automatů s výstupem. Editor obsahuje všechny požadované funkce v grafickém uživatelském rozhraní, zpracovaném podle standardů Windows User Interface Guidelines.

Programová stránka aplikace je ve funkčním stavu, ale několik věcí by bylo možné vylepšit. Bylo by vhodné přidat vykreslování tažených objektů během tažení na novou pozici a vykreslování neuplné hrany při její tvorbě. Dále by bylo vhodné oddělit uživatelské rozhraní od vlasního automatu, aby editor mohl používat jiná rozhraní, například zadání automatu formálním popisem.

Program by take mohl být rozšířen přidáním lokalizovaných menu a možností komplexních Mooreových a Mealyových strojů s více vstupy a výstupy.

Conclusions

The result of this bachelor thesis is an editor of state automata with output, called finite-state transducers. It includes all the required functions in a graphical user interface, made acording to Windows User Interface Guidelines.

The programing side of the application is in working order, but things could be improved. It would be appropriate to redraw objects while they are dragged to their new position and to render incomplete edges during creation. Furthermore, it would be advisable to separate the interface from the machine itself, so the editor could use other interfaces, for example one where you can create the machine from a formal description.

The program could also be extended by adding localized menus and a possibility to create complex Mealy and Moore machines with multiple inputs and outputs.

Reference

[1] Pilgrim, Robert A. *Computing Machinery* Robot Crafters, LLC, 2006.

[2] *Wikipedia* Převážně články Moore machine a Mealy machine a jejich dostupné zdroje.

[3] Demlová, Marie; Koubek, Václav *Algebraická teorie automatů* SNTL, Praha, 1990.

[4] Ľudovít, Molnár; Češka, Milan; Melichar, Bořivoj *Gramatiky a jazyky* SNTL, Praha, 1987.

A. Obsah přiloženého CD

`bin/`
: Obsahuje program StateAutomata ve spustitelné podobě přímo z CD/DVD. Adresář obsahuje i všechny potřebné knihovny a další soubory pro bezproblémové spuštění programu.

`doc/`
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu), tj. zdrojový text dokumentace, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty programu se všemi potřebnými (převzatými) zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu.

`readme.txt`
: Instrukce pro instalaci a spuštění programu, včetně požadavků pro jeho provoz.

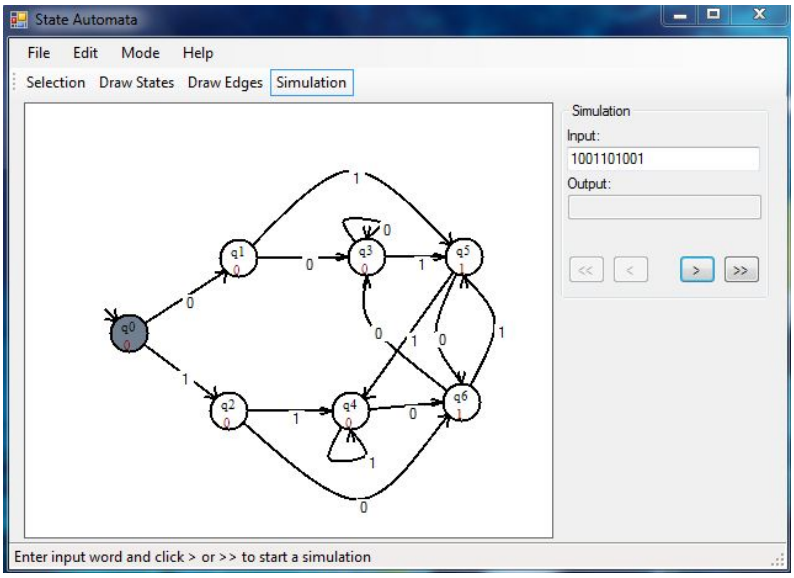
Navíc CD/DVD obsahuje:

`data/`
: Ukázková a testovací data použitá v práci a pro potřeby obhajoby práce.

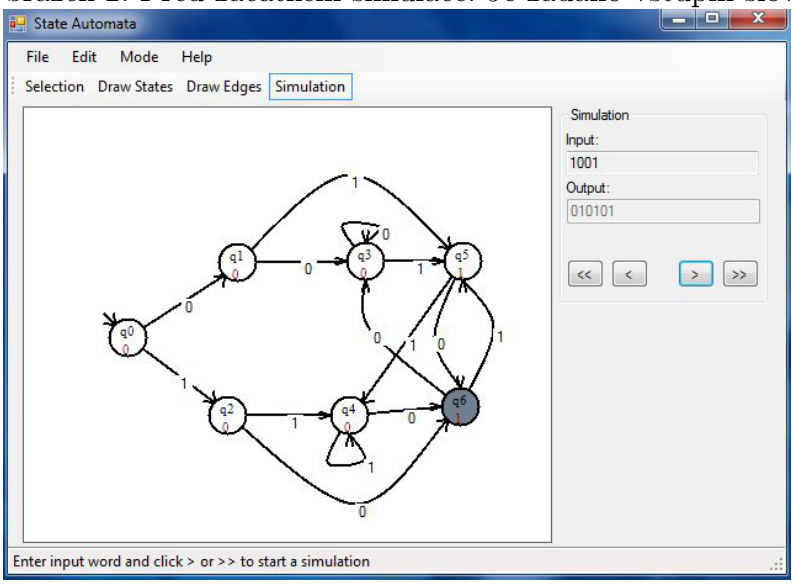
`install/`
: Instalátory aplikací, knihoven a jiných souborů nutných pro provoz programu / webové aplikace, které nejsou standardní součástí operačního systému.

U veškerých odjinud převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro materiály, u kterých toto není splněno, je uveden jejich zdroj (webová adresa) v textu dokumentace práce nebo v souboru `readme.txt`.

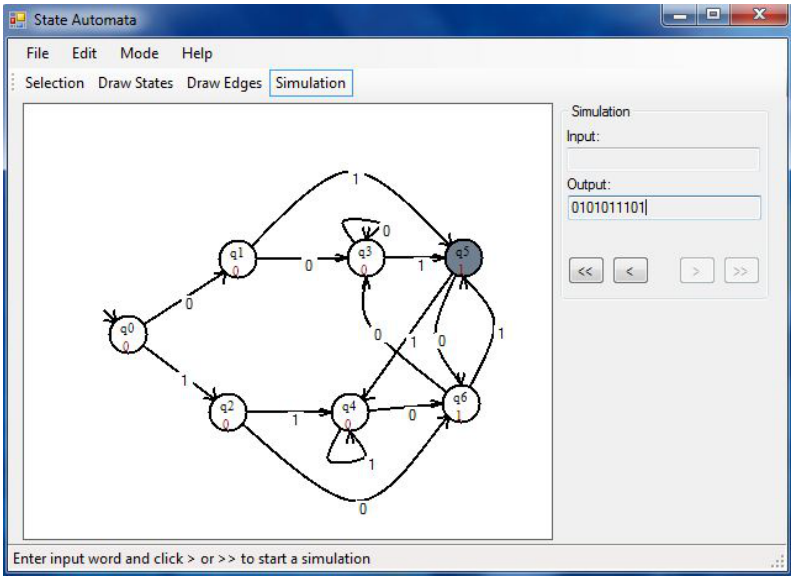
B. Ukázky simulace



Obrázek 2. Před začátkem simulace. Je zadáno vstupní slovo



Obrázek 3. V průběhu simulace jsou políčka pro vstup a výstup nedostupné.



Obrázek 4. Na konci simulace je pole výstupu otevřeno pro čtení.