Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Demonstrace Turingova stroje

2015

Miroslav Dajč

Vedoucí práce:
Mgr. Tomáš Kühr, Ph.D.

Studijní obor:
Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Miroslav Dajč
Název práce:	Demonstrace Turingova stroje
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2015
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	Mgr. Tomáš Kühr, Ph.D.
Počet stran:	41
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Miroslav Dajč
Title:	Demonstration of Turing Machines
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2015
Study field:	Applied Computer Science, combined form
Supervisor:	Mgr. Tomáš Kühr, Ph.D.
Page count:	41
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Aplikace Simulátor Turingova stroje vytvořena pro výukové účely v jazyce C#, svým řešením umožňuje simulovat výpočet deterministického Turingova stroje s jednou páskou. Simulace výpočtu lze uživatelsky řídit, krokovat, ladit a vyhodnocovat konfigurace pomocí nabízených nástrojů a statistik. Aplikace umožňuje ukládat a načítat definice Turingových strojů do XML souborů.

Synopsis

The Turing machine simulator application, designed for educational purposes in the C# language, enables the simulation of the calculation of the deterministic Turing machine with one tape, through its conception. The simulation of calculations allows users to control, tune and analyze configurations by using the offered tools and statistics. Application allows you to store and retrieve the definitions of Turing machines into XML files.

Klíčová slova: Turingův stroj; simulace; výpočet; příklady; demonstrace

Keywords: Turing Machine; simulation; calculation; examples; demonstration

Děkuji vedoucímu práce Mgr. Tomáši Kührovi, Ph.D. za velmi vstřícný přístup, odbornou pomoc a za velmi užitečné informace a rady. Dále děkuji své rodině, přátelům a známým za podporu a trpělivost při mém studiu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Jednou ze základních oblastí studia teoretické informatiky je oblast nazývající se **vyčíslitelnost**. Vyčíslitelnost se zabývá studiem řešitelnosti problémů pomocí algoritmů a využívá k tomu některých prostředků, na kterých lze snadno většinu algoritmů zpracovávat a zkoumat jejich vlastnosti. Jedním z těchto prostředků je model teoretického výpočetního stroje nazvaný **Turingův stroj**.

Turingův stroj (dále jen „TS“) je pojmenován podle **Alana Turinga**[1], který je považován za jednoho ze zakladatelů moderní informatiky.

Tento teoretický model si lze představit jako primitivní stroj, který pro svůj výpočet má k dispozici jednoduché instrukce a paměťové médium v podobě nekonečné pásky, na kterou lze zapisovat jednotlivé symboly. Stroj může měnit vnitřní stavy a obsah pásky v závislosti na použitých instrukcích. Na základě vstupních dat je proveden výpočet, který probíhá v jednotlivých krocích a může končit, popř. může cyklit, s různým výsledkem vyhodnocení.

Tato práce popisuje aplikaci **Simulátor Turingova stroje** (dále jen „Simulátor TS“), která byla vytvořena v jazyce C# jako součást bakalářské práce. Tento softwarový nástroj je navržen především pro názornou demonstraci výpočtu TS. Jednotlivé kroky výpočetního procesu jsou přehledně zobrazovány uživateli, který může celý proces výpočtu řídit, krokovat, ladit a vyhodnocovat jednotlivé konfigurace TS. Aplikace umožňuje vytvářet definice TS, které lze ukládat a znovu načítat pomocí datových souborů XML.

Po zadání vstupního slova je možné spustit demonstraci výpočtu, kterou lze pozastavit nebo úplně zastavit a sledovat tak podrobně změny v konfiguraci TS po jednotlivých výpočetních krocích. Aplikace nabízí i možnost zobrazení historie výpočtu včetně detailních informací. Kvalitní informovanost uživatele zajišťuje rozsáhla statistika, která je v průběhu výpočtu pravidelně aktualizována. Součástí aplikace jsou i ukázkové definice TS, na kterých je prakticky možné demonstrovat různé výpočetní možnosti TS.

Celá aplikace byla navržena zejména pro pochopení problematiky výpočtu TS, tomu je přizpůsoben i vzhled a jednoduchost celého řešení. To ale současně není ochuzeno o možnosti, které znalým uživatelům umožní využívat aplikaci i pro složitější výpočty a práci s definicemi nad rámec základního pochopení celé problematiky. K dispozici je rozsáhlá nápověda, která popisuje celé řešení včetně návodů jak s aplikací pracovat a jak efektivně využívat její možnosti.

[1] Alan Mathison Turing (1912 - 1954) - významný britský matematik, logik, kryptograf a zakladatel moderní informatiky

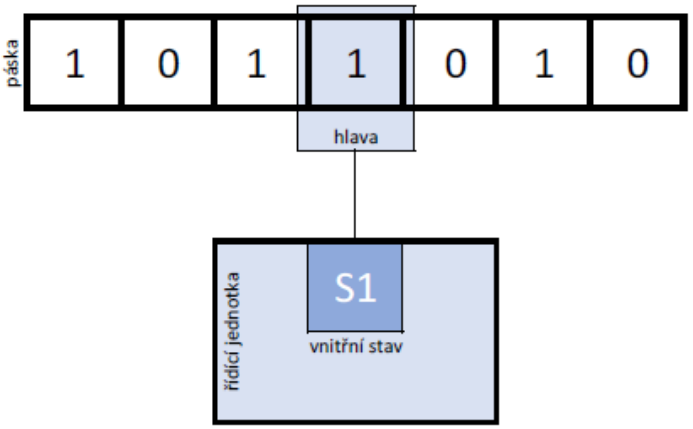
2 Teoretický model Turingova stroje

2.1 Neformální popis Turingova stroje

Pro snadné pochopení Turingova stroje a jeho výpočtu si nejprve neformálně popišme z čeho se skládá a jak probíhá výpočet.

Jak již bylo naznačeno v úvodu, pod pojmem **Turingův stroj** si lze představit primitivní stroj, který se skládá z řídící jednotky, čtecí a zapisovací hlavy a pásky. Takový model popisuje i *Pavel Martinek* v práci *Základy teoretické informatiky* [1].

Řídící jednotka se vždy nachází v jednom z předem definovaných interních stavů. Aktuální stav jednotky se v průběhu výpočtu muže měnit. Čtecí a zapisovací hlava provádí změny (čte a zapisuje) na pásce. Hlava se může po pásce pohybovat vpravo a vlevo, popř. může zůstat v místě, na kterém se nachází. Páska je rozdělena do jednotlivých buněk, ze kterých lze číst anebo zapisovat hodnoty. Tato páska je pravostranně nekonečná a čtecí (zapisovací) hlava se po ní může libovolně pohybovat od první buňky až do nekonečna. Zjednodušené schéma TS je na obrázku 1.



Obrázek 1: Zjednodušené schéma TS

Pohyb po pásce je závislý na instrukcích, na základě kterých probíhá výpočetní proces TS.

Výpočet začíná nastavením řídící jednotky do počátečního stavu. Čtecí (zapisovací) hlava je nastavena na první buňku pásky. Na pásce je zapsáno vstupní slovo skládající se ze symbolů zapsaných v jednotlivých buňkách pásky a to od první buňky směrem vpravo.

Čtecí (zapisovací) hlava přečte hodnotu buňky, na které se nachází a řídící jednotka na základě této hodnoty a aktuálního interního stavu provede přechod podle předem definovaných instrukcí. Přičemž nastaví interní stav řídící jednotky

na nový (popř. ponechá stávající), dále zapíše na pásku novou hodnotu a posune čtecí (zapisovací) hlavu buď vlevo, vpravo nebo hlava zůstane na místě.

Výpočet probíhá v jednotlivých krocích, dokud se řídící jednotka nedostane do jednoho z koncových stavů. Koncové stavy jsou interní stavy, ve kterých výpočet TS končí. Vstupní slovo je pak buď přijato, nebo zamítnuto podle toho jakého koncového stavu stroj dosáhnul. Pokud stroj neukončí výpočet po provedení konečného počtu kroků, říkáme, že cyklí. Podrobněji je tento proces popsán v kapitole 2.3.

Nyní jsme popsali teoretický model počítače, který odpovídá deterministickému TS. U deterministického TS je každý krok vždy jednoznačně definován na rozdíl od nedeterministického TS, který může obsahovat více možných instrukcí které odpovídají aktuálnímu vnitřnímu stavu řídící jednotky a přečtené hodnotě z pásky. Použitá instrukce se v pak vybere náhodně.

Následující text bude pojednávat výhradně o deterministickém TS, pokud nebude uvedeno jinak. Další možné varianty TS jako jsou např. vícepáskové, v tuto chvíli pomineme a budeme vždy předpokládat jednopáskovou variantu deterministického TS.

2.2 Formální definice Turingova stroje

Formální definice Turingova stroje se v závislosti na použitém informačním zdroji nebo použité literatuře může mírně lišit.

Podle definice, která je uvedena v práci *Pavla Martinka Základy teoretické informatiky* [1] je vyjádřen TS jako uspořádaná sedmice
(Q, Σ, Γ, δ, q_0, q_+, q_-), kde

- Q - je konečná neprázdná množina stavů,
- Σ - je vstupní abeceda,
- Γ - je pásková abeceda, Σ ⊂ Γ, Γ - Σ obsahuje speciální (prázdný) symbol,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times$ {L, P} je přechodová funkce,
- $q_0 \in$ Q - je počáteční stav,
- $q_+ \in$ Q - je přijímající stav,
- $q_- \in$ Q - je zamítající stav, přičemž $q_+ \neq q_-$.

Podívejme se nyní podrobněji na jednotlivé položky definice.

Q je označena konečná množina všech **vnitřních stavů** TS. V těchto stavech se může TS nacházet v průběhu výpočtu. Množina stavů musí obsahovat alespoň jeden stav[2].

[2]pokud definice TS obsahuje pouze jeden interní stav, musí být tento stav počáteční

Písmenem Σ je označena množina všech symbolů, kterou nazýváme **vstupní abeceda** (někdy také zkráceně abeceda). Abeceda obsahuje všechny symboly, které mohou tvořit vstupní slova TS. Rozšířením abecedy o speciální (prázdný) symbol dostaneme množinu symbolů nazvanou **pásková abeceda**.

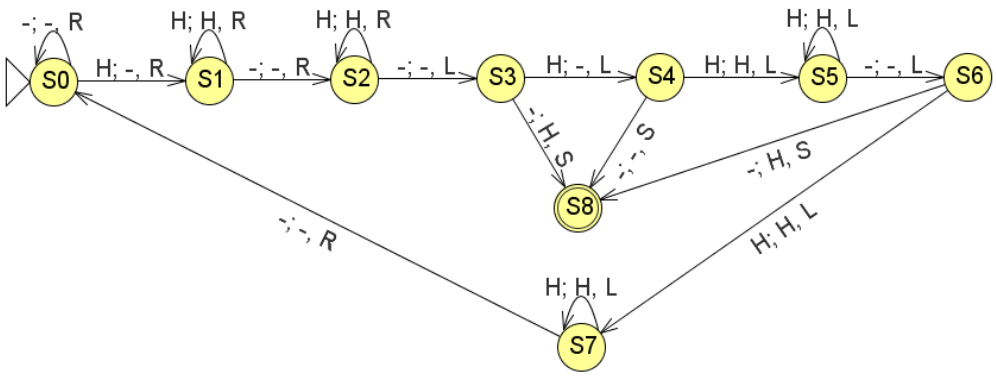
Prázdný symbol, který obsahuje pásková abeceda (označena v definici písmenem Γ), nemůže být součástí vstupního slova. Jedná se o symbol, který je při zahájení výpočtu zapsán ve všech buňkách pásky mimo ty, které obsahují vstupní slovo.

Přechodová funkce v definici TS označena δ je množina definovaných pravidel (přechodů) podle kterých probíhá výpočet. Tyto pravidla určují na základě aktuálního stavu a symbolu přečteného z pásky do jakého nového stavu TS přejde a jaký symbol bude zapsán na pásku. Po zápise může dojít k posunu čtecí (zapisovací) hlavy vlevo nebo vpravo, popř. hlava zůstane na místě. Více informací naleznete v kapitole 2.3.

Interní stav označen v definici jako q_0 nazýváme **počáteční stav**. V tomto stavu se nachází TS při zahájení výpočtu.

Koncové stavy jsou rozděleny do dvou skupin na **stavy přijímající**, které jsou označeny q_+ a **stavy zamítající**, které označujeme q_-. V takových stavech výpočet TS končí. Zvláštním případem je možnost, kdy počáteční stav je současně koncovým stavem. V takovém případě výpočet končí již v počátečním stavu bez jediného výpočetního kroku.

Definici TS lze znázornit i graficky pomocí diagramu (grafu). Obrázek 2 ukazuje definici TS, který simuluje výpočet rozdílu dvou hodnot zapsaných na pásce. Tato definice je součástí přílohy práce a naleznete ji na přiloženém CD/DVD.



Obrázek 2: Diagram definice TS

Pro konstrukci diagramů (grafů) jsou definována určitá pravidla a zvyklosti. Vytváření diagramů definic TS popisuje *Petr Jančar aj.* v učebním textu *Úvod do teoretické informatiky* [3]. Zobrazení definice TS pomocí diagramu (grafu) se

využívá pro svou přehlednost, na druhé straně u složitých definic se může uživatel v diagramu orientovat s obtížemi.

2.3 Výpočet Turingova stroje

V předešlé kapitole 2.2 jsme definovali TS. Nyní se podívejme podrobněji na průběh výpočtu TS.

Stroj se na začátku výpočtu nachází v počátečním stavu q_0. Zvláštní situace může nastat, pokud počáteční stav je současně i koncovým. V takovém případě výpočet končí bez toho, aniž by byl proveden jediný výpočetní krok.

Pokud není počáteční stav koncový, bude výpočet probíhat v jednotlivých krocích (někdy se uvádí i taktech). Počet kroků výpočtu nelze předem určit. V krajním případě může být počet kroků nekonečný. Tento případ nastane, pokud stroj cyklí. Více informací je dále v této kapitole.

Výpočet tedy začíná v počátečním stavu. Pomineme-li výše uvedenou možnost, že počáteční stav je současně koncový, probíhá výpočet dalšími kroky. Čtecí (zapisovací) hlava je nastavena na první buňce pásky a přečte první symbol vstupního slova zapsaný na pásce. Stroj podle aktuálního stavu, ve kterém se právě nachází a přečteného symbolu z pásky nalezne přechodovou funkci, která vyhovuje těmto podmínkám.

Přechodová funkce určuje další postup (přechod). Nejprve TS dle přechodové funkce změní interní stav, ve kterém se nachází, popř. zůstane ve stejném stavu. Čtecí (zapisovací) hlava zapíše na pásku nový symbol. Symbol se zapíše na pozici (do buňky) pásky na které se hlava nachází. Poté se hlava posune ve směru, který určí přechodová funkce (vlevo nebo vpravo, popř. hlava zůstane na místě). Pokud se nachází čtecí hlava na první buňce pásky a přechodová funkce požaduje posun hlavy vlevo, nedojde k žádnému posunu a hlava zůstane na místě. Není možné posunout hlavu mimo pásku. Posun hlavy vpravo lze provést vždy, protože páska je pravostranně nekonečná.

Pokud se nyní nachází TS v koncovém stavu, výpočet končí. Koncové stavy mohou být buď přijímající, nebo zamítající. V případě, že se TS dostane do přijímajícího koncového stavu, říkáme, že TS vstupní slovo přijal. Pokud se TS dostane do zamítajícího koncového stavu, v tom případě vstupní slovo zamítnul.

Pokud se TS nenachází v koncovém stavu, výpočet probíhá další krokem. Toto se opakuje, dokud stroj nedosáhne koncového stavu. Pokud TS končí výpočet po konečném počtu kroků, znamená to, že dosáhnul při výpočtu koncového stavu. Pokud výpočet nekončí po konečném počtu kroků, říkáme, že TS cyklí. V takovém případě přechází stroj mezi několika interními stavy a výpočet nikdy nekončí.

V každém kroku se TS nachází v určité **konfiguraci**. Konfigurace jednoznačně popisuje situaci, ve kterém se TS nachází. K tomu postačuje aktuální stav, konečná část pásky, na které je zapsáno vstupní slovo, popř. část kde dosud probíhal výpočet a pozice čtecí (zapisovací) hlavy.

Výpočet Turingova stroje lze tedy vyjádřit posloupností jednotlivých konfi-

gurací. Podrobněji popisuje tuto problematiku *Petr Jančar aj.* v učebním textu *Úvod do teoretické informatiky* [3].

3 Programátorská dokumentace

3.1 Použité technologie

Pro vývoj celé aplikace byl použit objektově orientovaný programovací jazyk **C#**. Programovací jazyk C# byl vyvinutý společností *Microsoft* a vychází nepřímo z jazyka C. Syntaxe tohoto programovacího jazyka je přehledná a snadno pochopitelná i pro začínající programátory. Celý projekt aplikace byl zpracován ve vývojové prostředí *Microsoft Visual Studio Express 2013 for Windows Desktop*. Informace o tomto vývojovém prostředí lze nalézt na internetovém portále společnosti *Microsoft* [4].

Toto vývojové prostředí nabízí mnoho užitečných nástrojů pro snadnou správu projektu, možnosti ladění a disponuje také rozsáhlou nápovědou. Projekt byl založen jako formulářová aplikace Windows. Základem této aplikace je hlavní okno, ve kterém se uživatel pohybuje po celou dobu běhu aplikace. Další okna se využívají pouze jako dialogová anebo jsou podřízená hlavnímu oknu.

V aplikaci je využita sada ikon, která byla vybrána na základě tématické příbuznosti. Sada byla čerpána z internetového portálu *pasnox/oxygen-icons-png* [5]. Kromě sady ikon nejsou v aplikaci použity žádné další komponenty z externích zdrojů.

Aplikace je určena pro platformu *Microsoft Windows* a využívá rozhraní *.NET Framework 4*. Před spuštěním aplikace je nutné ověřit, zdali je toto rozhraní v operačním systému dostupné. Více informací naleznete v kapitole č. 4.6

3.2 Návrh aplikace

Před samotným vývojem aplikace byla provedena analýza požadavků, které by měla výsledná aplikace splňovat. Jednalo se v podstatě o **Use Case**[3] analýzu. Nejprve bylo nutné stanovit omezení software a zamyslet se, pro koho je aplikace určena.

Navrhovaná aplikace bude mít několik omezení, které je nutné dodržet, pokud má správně fungovat. Jedná se o požadavky na prostředí, které jsou popsány v kapitole 4.6, dále je nutné, aby uživatel měl dostatečné oprávnění v operačním systému, ve kterém bude aplikaci instalovat. Pokud bude chtít uživatel využít i možnosti načítání a ukládání dat je nutné, aby měl i na tyto operace v daném souborovém systému oprávnění. Pokud budou splněny výše uvedené podmínky, bude moci uživatel nainstalovat a spustit aplikaci. Tím jsme stanovili podmínky, za jakých má být aplikace provozována.

[3]Use Case - česky případ užití je seznam kroků, který definuje interakci mezi actorem a systémem

Nyní je na čase analyzovat, pro koho je aplikace určena a jaké bude mít využití.

V zadání této práce je uvedeno, že aplikace bude sloužit pro výukové účely. Tím byla již částečně popsána role uživatele, pro kterého má být aplikace vytvořena. Vycházejme tedy z předpokladu, že uživatelem může být student nebo člověk zajímající se o problematiku TS. Označme je pro tuto chvíli jednotně **Actor**. Actor vstupuje do interakce se softwarem a software reaguje na jeho podněty. Aplikace by měla pro tyto účely splňovat základní případy užití, které vychází částečně ze zadání této práce, ale jsou rozšířeny autorem projektu. Soupis základních případů užití aplikace je v tabulce 1.

Tabulka 1: Případy užití

Pořadí	Popis
1.	Vytvoření nové definice TS
2.	Otevření definice TS ze souboru
3.	Uložení definice TS do souboru
4.	Simulace výpočtu TS
5.	Pozastavení a krokování v průběhu simulace výpočtu TS
6.	Získání informací o průběhu simulace výpočtu TS
7.	Ladění definice TS
8.	Generování dat definice TS

Podrobný popis všech případů užití by byl velmi rozsáhlý, proto je v příloze A zpracován jeden vzorový případ užití, aby čtenář mohl nahlédnout do této problematiky.

3.3 Syntaxe zdrojového kódu

Snahou bylo v celém projektu, aby byl zdrojový kód na první pohled přehledný a dobře srozumitelný. Z těchto důvodů je ve zdrojovém kódu použito některých pravidel, které tomu měly napomoci.

Jeden ze základních předpokladů dobře čitelného a srozumitelného zdrojového kódu je dostatečný popis zdrojového kódu pomocí komentářů. V aplikaci je použit popis u všech metod a tříd vždy před názvem třídy nebo metody, včetně popisu parametrů a návratových hodnot. Tento způsob zápisu komentářů napomáhá již samotnému programátorovi při psaní dalšího kódu. V některých vývojových prostředích, jako je např. *Microsoft Visual Studio* [4], je tento komentář při dalším použití metody zobrazován jako kontextová nápověda. Další výhodou takového popisu zdrojového kódu je možnost exportu popisu do XML[4] souboru.

Tento popis ovšem nemusí dostačovat v případě, že metoda má rozsáhlejší charakter a na první pohled není zřejmý význam některých operací. Z tohoto

[4]XML (Extensible Markup Language) - česky rozšiřitelný značkovací jazyk, je obecný značkovací jazyk

důvodu jsou v projektu u některých metod ještě jednořádkové podrobnější popisy.

Lepší orientaci ve zdrojovém kódu napomáhá i způsob označení metod, konstant, proměnných a dalších součástí zdrojového kódu. Autor projektu si dal za cíl, pokud je to možné, udržet v celém projektu následující pravidla, které si sám stanovil. Tyto pravidla nejsou ovšem dogma a je možné, že je v některých výjimečných případech porušuje.

Všechny metody v projektu jsou označeny prefixem[5]. Výjimku tvoří pouze metody, které obsluhují události uživatelského rozhraní. Tyto metody jsou generovány automaticky vývojovým prostředím a jsou pro programátory intuitivní, proto označení těchto metod nebylo měněno. Popis prefixů metod a jejich význam je v tabulce 2.

Tabulka 2: Seznam prefixů metod

Prefix	Popis
g_	metoda která pouze čte data a vždy má návratovou hodnotu
s_	metoda která mění data, nemusí mít návratovou hodnotu

Dále jsou ve zdrojovém kódu dodržována určitá pravidla pro označování konstant, proměnných a dalších prvků. Tyto prvky jsou označeny před názvem prefixem, který usnadňuje čitelnost a lepší orientaci ve zdrojovém kódu. Seznam prefixů a jejich význam je uveden v tabulce 3.

Tabulka 3: Seznam prefixů konstant, proměnných a dalších prvků

Prefix	Popis
c_	označení konstanty
d_	označení interní proměnné nebo pole v rámci třídy
del_	označení delegáta
g_	globální instance třídy, většinou použita v celém projektu
n_	označení proměnné nastavení aplikace
p_	parametr metody
i_	interní proměnná, která má platnost uvnitř metody

Celý zdrojový kód projektu je standardně strukturován podle zvyklostí používaných v jazyce C#.

3.4 Struktura aplikace

Aplikace Simulátor TS a všechny její třídy jsou umístěny ve jmenném prostoru[6] **SimulatorTS**. Tento jmenný prostor obsahuje několik tříd, které jsou rozděleny

[5]prefix - česky předpona, je část slova, která se vkládá před kořen slova

[6]jmenný prostor - umožňuje uspořádat zdrojový kód do celku, který spolu logicky souvisí

do několika zdrojových souborů. Seznam zdrojových souborů aplikace naleznete v příloze D.

Seznam tříd včetně jejich stručného popisu a popisu případných vazeb je následující:

Třída: **`Soubory`**
: Statická třída *Soubory* byla vytvořena z důvodu oddělení části kódu, který pracuje se souborovým systémem. Důvodem bylo docílení přehlednosti zdrojového kódu a snadná údržba v případě potřeby měnit či rozšiřovat datové struktury souborů.

Třída: **`Stroj`**
: Nejdůležitější třída, která představuje v aplikaci definici TS. Po spuštění aplikace se založí nová instance této třídy a ta je používána po celou dobu běhu aplikace jako výchozí pro práci s TS. Nová instance se vytváří pouze v případě, kdy dochází k načítání definice TS ze souboru. Tato třída udržuje všechny informace o aktuálním stavu TS.

Třída: **`FormAbeceda`**
: Třída představuje podřízené (MDI[7]) okno *Abeceda* v uživatelském rozhraní aplikace. Při spuštění aplikace dojde k založení instance tohoto okna, která je aktivní po celou dobu běhu aplikace.

Třída: **`FormAbecedaDialog`**
: Třída *FormAbecedaDialog* představuje v uživatelském rozhraní dialogové okno určené pro editaci položek abecedy. Třída obsahuje pouze konstruktor, ve kterém jsou nastaveny editační položky před otevřením dialogu. Další zpracování dat probíhá až po uzavření dialogu ve třídě *FormAbeceda*.

Třída: **`FormFunkce`**
: *FormFunkce* je další třídou představující okno v uživatelském rozhraní. Jedná se o okno *Přechodová funkce*. Instance této třídy je vytvořena ihned po spuštění aplikace a je po celou dobu běhu aplikace aktivní.

Třída: **`FormFunkceDialog`**
: Tato třída představuje dialogové okno v uživatelském rozhraní, které je určené pro editaci položek přechodové funkce. Kromě konstruktoru ve kterém se nastavují editační položky, obsahuje třída pouze jednu metodu, která je určena pro kontrolu správnosti vyplněných údajů.

Třída: **`FormGenerovaniDat`**
: Třída *FormGenerovaniDat* je určena k hromadnému generovaní dat TS. V uživatelském rozhraní představuje dialogové okno *Hromadné generování dat*. Toto dialogové okno je využíváno pro generování dat abecedy, stavů a pásky.

[7]MDI (Multiple Document Interface) - česky rozhraní více dokumentů, je typ aplikace umožňující zobrazit více dokumentů současně, každý dokument je zobrazen ve vlastním okně

Třída: **`FormHlavni`**

Hlavní okno aplikace v uživatelském rozhraní je reprezentováno třídou *FormHlavni*. Instance okna je vytvořena ve vstupním bodě aplikace. Třída obsahuje mimo metod pro obsluhu událostí uživatelského rozhraní i metody pro paralelní zpracování informací v samostatném vlákně. Podrobněji je samostatné vlákno a související metody popsány v kapitole 3.5.

Třída: **`FormOprogramuDialog`**

Třída *FormOprogramuDialog* reprezentuje v uživatelském rozhrani dialogové okno *O programu*, které zobrazuje informace o autorovi a verzi aplikace.

Třída: **`FormParametryDialog`**

Další třídou reprezentující objekt v uživatelském rozhraní je *FormParametryDialog*. Jedná se o editační dialogové okno *Parametry*, které je určeno pro nastavení parametrů definice TS.

Třída: **`FormPaska`**

Jednou ze složitějších tříd je třída *FormPaska*, která reprezentuje v uživatelském rozhraní podřízené (MDI) okno *Páska*. Instance okna je vytvořena při spuštění aplikace a zůstává aktivní po celou době běhu aplikace. Třída obsahuje metody, které zabezpečují grafické znázornění pásky.

Třída: **`FormPaskaDialog`**

V uživatelském rozhraní je editační dialog, ve kterém je možné měnit hodnoty zapsané na pásce, reprezentován třídou *FormPaskaDialog*. Instance této třídy je vytvářena jako součást obsluhy události při změně pásky.

Třída: **`FormStatistika`**

Jednou ze zajímavých tříd je *FormStatistika*. Tato třída reprezentuje v uživatelském rozhraní podřízené (MDI) okno *Statistika*. Instance této třídy je vytvořena ihned po spuštění aplikace. Okno zobrazuje aktuální statistické ukazatele o definici a stavu TS, popř. o právě probíhající demonstraci výpočtu TS.

Třída: **`FormStavy`**

Třída *FormStavy* reprezentuje poslední z podřízených (MDI) oken v uživatelském rozhraní. Okno *Stavy* slouží v aplikaci k zobrazení a údržbě všech vnitřních stavů TS. Instance této třídy je vytvořena ihned po spuštění aplikace.

Třída: **`FormStavyDialog`**

Editační dialog v uživatelském rozhraní pro změnu položek stavů je reprezantován třídou *FormStavyDialog*. Instance třídy je vytvářena vždy ze třídy *FormStavy* v okamžiku uživatelského požadavku na editaci položky stavu.

Třída: **`Program`**
: Základní třída *Program* obsahuje hlavní vstupní bod aplikace, metodu *Main*. V této metodě je vytvořena instance třídy *FormHlavni* reprezentující hlavní okno aplikace.

Třída: **`Settings`**
: Tato třída obsahuje metody, které umožňují měnit nastavení vlastností aplikace. Tyto metody a vlastnosti jsou využity v aplikaci pro ukládání pozic a velikosti oken.

Není záměrem této práce podrobně popisovat všechny třídy a provádět jejich analýzu. Čtenáře by ale mohly zaujmout některé třídy a jejich struktury nebo metody.

Podrobněji se nyní podívejme na některé zajímavé části zdrojového kódu třídy **FormHlavni**. Z důvodů nutnosti strukturovat některé údaje bylo žádoucí definovat nové struktury, které budou využívány v celé aplikaci. Jedná se o následující struktury:

- *StrukturaPolozkyStatistika*
- *StrukturaPolozkyAbeceda*
- *StrukturaPolozkyStavy*
- *StrukturaPolozkyPrechFunkce*
- *StrukturaPolozkyPaska*
- *StrukturaPolozkyHistorie*

Tyto struktury jsou určeny k práci s daty definice TS. Obsahově se odlišují pouze dvě struktury. Jedná se o *StrukturaPolozkyStatistika*, která je určena pro práci s daty statistiky a o *StrukturaPolozkyHistorie*, která je určena pro práci s historii výpočtu TS.

Podívejme se nyní podrobněji na strukturu *StrukturaPolozkyAbeceda*. Zdrojový kód této struktury vidíme v ukázce kódu 1. Nebudeme se nyní zabývat způsobem zápisu komentářů, který je podrobně popsán v kapitole 3.3, ale pouze samotným kódem. Struktura obsahuje parametry *poradi*, *nazev* a *popis*, které jsou určeny pro definici položky abecedy.

Parametr *poradi*, udržuje při vytváření položky abecedy samotná aplikace a to inkrementálním[8] způsobem. Tato položka je pak použita v dalších strukturách jako odkaz na požadovaný symbol abecedy. Tím se lépe udržuje konzistence dat a současně aplikace umožňuje měnit název symbolu a popis bez nutnosti měnit

[8] inkrementální - česky přírustkový, je způsob kdy aplikace zvyšuje nejvyšší nalezenou hodnotu o jedničku

data v ostatních strukturách (přechodové funkci nebo na pásce). Obdobné řešení je použito ve všech strukturách a vazbách mezi nimi.

Parametr *nazev*, obsahuje název symbolu. Aplikace nevyžaduje, aby symbol byl chápán jako jedno písmeno abecedy, ale může se skládat z více znaků, např. „AA“, „Xc“, „V12“.

Parametr *popis*, může obsahovat libovolný text. Jedná se o popis symbolu, který je zobrazován v seznamu položek v okně *Abeceda* a v historii použití symbolu, při zapnutí *Informačního průvodce*. Více o *Informačním průvodci* je popsáno v kapitole 4.4

```
1  /// <summary>Struktura položky abeceda</summary>
2  /// <remarks>kontrola kódu: 31.1.2015</remarks>
3  public struct StrukturaPolozkyAbeceda
4  {
5     /// <summary>Pořadí položky</summary>
6     public int poradi;
7     /// <summary>Počet použití položky pro čtení</summary>
8     public int pouziti_R;
9     /// <summary>Počet použití položky pro zápis</summary>
10    public int pouziti_W;
11    /// <summary>Název položky</summary>
12    public string nazev;
13    /// <summary>Popis položky</summary>
14    public string popis;
15
16    /// <summary>Konstruktor struktury</summary>
17    /// <remarks>kontrola kódu: 31.1.2015</remarks>
18    public StrukturaPolozkyAbeceda(int p_poradi, string p_nazev,
          string p_popis, int p_pouziti_R, int p_pouziti_W)
19    {
20       this.poradi = p_poradi;
21       this.nazev = p_nazev;
22       this.popis = p_popis;
23       this.pouziti_R = p_pouziti_R;
24       this.pouziti_W = p_pouziti_W;
25    }
26 }
```

Zdrojový kód 1: Struktura *StrukturaPolozkyAbeceda*

Ještě jsme se nezmínili o posledních dvou parametrech této struktury. Jedná se o parametr *pouziti_R* a *pouziti_W*. Tyto parametry jsou určeny ke zpracování statistických dat o daném symbolu abecedy. Parametr *pouziti_R* obsahuje údaj, který představuje hodnotu, kolikrát byl daný symbol přečten z pásky. Analogicky parametr *pouziti_W* hodnotu vyjadřující kolikrát byl symbol zapsán na pásku. Statistická data použití se vyskytují v různých formách i v ostatních strukturách definice TS.

Podívejme se ještě podrobněji na jednu zajímavou metodu třídy **Stroj**. Pokud jsme se v předešlém textu podrobněji věnovali struktuře položky abecedy, podívejme se nyní na to jak je použita tato struktura pro změnu položky abecedy. Metoda, která zabezpečuje změnu položky abecedy se nazývá *s_abeceda_zmenit* a její zdrojový kód je v ukázce kódu 2.

```
1  /// <summary>Změní položku abecedy dle pořadí</summary>
2  /// <remarks>kontrola kódu: 31.1.2015</remarks>
3  /// <param name="p_poradi">Pořadí</param>
4  /// <param name="p_nazev">Název</param>
5  /// <param name="p_popis">Popis</param>
6  /// <param name="p_pouziti_R">Použití - čtení</param>
7  /// <param name="p_pouziti_W">Použití - zápis</param>
8  public void s_abeceda_zmenit(int p_poradi, string p_nazev, string
       p_popis, int p_pouziti_R, int p_pouziti_W)
9  {
10    int i;
11    bool i_update = false;
12    StrukturaPolozkyAbeceda i_polozka = new StrukturaPolozkyAbeceda(
          p_poradi, p_nazev, p_popis, p_pouziti_R, p_pouziti_W);
13
14    for (i = 0; i < d_abeceda.Count; i++)
15    {
16       if (i_polozka.poradi == ((StrukturaPolozkyAbeceda)d_abeceda[i
             ]).poradi)
17       {
18          d_abeceda[i] = i_polozka;
19          i_update = true;
20       }
21    }
22
23    if (!i_update)
24    {
25       d_abeceda.Add(i_polozka);
26    }
27    d_zmenadat = true;
28 }
```

Zdrojový kód 2: Metoda *s_abeceda_zmenit* třídy *Stroj*

Metoda se volá s parametry odpovídající parametrům struktury *StrukturaPolozkyAbeceda*. Pomineme-li definici proměnných *i* a *i_update*, definuje se na začátku metody lokální proměnná, která je typu *StrukturaPolozkyAbeceda* a při inicializaci se předávají konstruktoru struktury parametry metody *s_abeceda_zmenit*. Tím dojde k vytvoření lokální proměnné ve struktuře, která odpovídá nové (změněné) položce abecedy. Záměrně uvádíme nové popř. změněné položky abecedy, protože nyní ještě není rozhodnuto, jak bude s položkou naloženo.

Další kód obsahuje cyklus, který projde všechny položky abecedy uložené v proměnné *d_abeceda*, která je typu *ArrayList*. V cyklu je podmínka, která

porovnává hodnotu *i_polozka.poradi* s hodnotou *poradi* ve struktuře *StrukturaPolozkyAbeceda*, která je uložena v *ArrayList* s indexem *i*.

Pokud je nalezena shoda, to znamená, že položka abecedy již existuje, následným příkazem bude nahrazena. Současně se změní i proměnná *i_update* na hodnotu *true*. Po ukončení cyklu zkontroluje podmínka proměnnou *i_update* a pokud není *true*, což odpovídá stavu, kdy položka s daným pořadím nebyla v předešlém cyklu nalezena, přidá do proměnné *d_abeceda* novou položku v předem připravené struktuře.

V obou případech dojde ke změně dat položky abecedy a potažmo celé definice TS. Tato skutečnost je zaznamenána změnou hodnoty proměnné *d_zmenadat* na *true*. Tato proměnná v aplikaci udržuje informaci o změně definice TS, což je využito např. při ukládání dat definice TS do souboru.

Obdobně jsou navrženy i metody pro změny ostatních položek definice TS.

Čtenář může další zdrojový kód najít na přiloženém CD/DVD, kde si může nastudovat další řešení a návrh aplikace.

3.5 Práce s vlákny

Z hlediska zdrojového kódu je samotný výpočet TS prováděn v samostatném vlákně[9]. Toto řešení umožnilo oddělit samotný výpočet TS, který běží v aplikaci na pozadí, od grafické demonstrace na popředí.

Použití samostatného vlákna je v aplikaci řešeno třídou **BackgroudWorker**, která je k tomuto účelu určena a poskytuje i potřebné metody a události. Hlavní událost třídy se nazývá *DoWork*. Tato událost je aktivována při spuštění demonstrace výpočtu TS pomocí metody *RunWorkerAsync*. Zdrojový kód obsluhy události *DoWork* obsahuje hlavní smyčku výpočtu TS, který probíhá na pozadí. Výpočet běží po jednotlivých krocích a po každém kroku probíhá kontrola, zdali se TS nedostal do některého z koncových stavů. Pokud je kontrola negativní (aktuální stav není koncový), pokračuje výpočet dalším krokem.

Při každém kroku dojde k volání metody *ReportProgress* z výše uvedené třídy. Tím vlákno oznamuje hlavnímu vláknu, že došlo k postupu při zpracování a v hlavním vlákně se vyvolá událost *ProgressChanged*. Ve zdrojovém kódu obsluhy této události je část, která kontroluje, zdali nebyl průběh výpočtu pozastaven.

Z důvodu pravidelné aktualizace uživatelského rozhraní se ve vlákně na pozadí volají metody tříd, které provádějí tyto aktualizace. Tyto metody jsou volány pomocí delegátů[10], vytvořených v hlavním vlákně programu. Použitím delegátů jsou ošetřeny možné kolize obou vláken při práci s uživatelským rozhraním hlavního okna. Metody se volají postupně v průběhu výpočtu, aby uživatel mohl vidět, jak probíhá jednotlivý krok výpočtu. Pokud by aplikace aktualizovala všechny prvky uživatelského rozhraní v obsluze události *ProgressChanged*, což se na první

[9] vlákno umožňuje paralelní spouštění samostatného kódu pomocí tzv. multithreadingu

[10] delegát slouží k reprezentaci reference na metodu, někdy je označován jako bezpečný ukazatel na funkci

pohled nabízí, byl by výpočet zobrazován skokově po jednotlivých krocích. Při větším zpomalení aplikace by pak takový výpočet nebyl plynulý.

Tím se dostáváme k další části zdrojového kódu, který je součástí metody *DoWork*. Jedná se o část zdrojového kódu zabezpečující zpomalení demonstrace výpočtu TS. Důvodem je možnost měnit rychlost demonstrace výpočtu TS i v průběhu výpočtu. Způsob je popsán v kapitole 4. Vlákno, které běží na pozadí zpomaluje výpočet pomocí metody *Sleep*, která je součástí třídy *Thread*. Pokud by nedocházelo ke zpomalení, aplikace by ztratila vypovídající hodnotu.

Průběh výpočtu vlákna na pozadí lze přerušit nebo pozastavit z hlavního vlákna. Dochází k tomu v případě uživatelského požadavku. Výpočet je buď pozastaven, v tom případě vlákno na pozadí cyklí v určitém zdrojovém kódu a čeká na další příkazy, nebo je zastaven a vlákno je ukončeno.

3.6 Datové soubory

Aplikace *Simulátor TS* umožňuje jednotlivé definice TS ukládat do souborů XML. Tyto soubory jsou strukturovány podle potřeb aplikace do několika elementů. Základní element **<stroj>** obsahuje tyto další elementy:

Element: **<parametry>**
: Element obsahuje další elementy popisující základní parametry definice TS jako jsou **<nazev>**, kde je uveden název TS, **<autor>**, kde je uveden autor a **<popis>** obsahující popis TS.

Element: **<stavy>**
: Obsahuje opakovaně elementy **<stav>**, které obsahují parametry interní stavů.

Element: **<abeceda>**
: Obsahuje opakovaně elementy **<symbol>**, které obsahují parametry všech symbolů.

Element: **<prechodovafunkce>**
: Obsahuje opakovaně elementy **<funkce>**, které obsahují parametry všech přechodových funkcí.

Element: **<paska>**
: Obsahuje opakovaně elementy **<bunka>**, které obsahují parametry všech použitých buněk pásky.

Jednoduchá definice TS uložená v XML souboru je ukázána v příloze B. Uložené definice TS v XML souborech lze opětovně načítat do aplikace. Před načtením je soubor nejprve zkontrolován. Kontrola probíhá ve třídě *Soubory*. Metoda, která pracuje se soborem nejprve zkouší načíst všechny elementy souboru XML do lokální instance třídy *Stroj* a pokud vše proběhne bez chyb, načte metoda data definice TS znovu do výchozí instance třídy *Stroj*.

Tento typ struktury datových souborů *Simulátoru TS* byl vybrán autorem aplikace záměrně z důvodu snadného pochopení a možnosti uživatelsky modifikovat uložené definice mimo aplikaci.

3.7 Kontrola kódu a testování

Zdrojový kód aplikace obsahuje mimo jiné i informace o kontrole kódu. Tento údaj je součástí komentářů, které jsou uvedeny před každou metodou všech tříd. Tento komentář můžete vidět dobře i v ukázce kódu 2, který je uveden v kapitole 3.4. Datum kontroly kódu poskytuje informaci, kdy došlo k revizi kódu. Při vývoji této aplikace byl kód několikrát kontrolován z důvodu snahy odstranit co nejvíce možných chyb ještě před testováním aplikace.

Testování aplikace proběhlo podle předem definovaných testovacích scénářů. Testovací scénáře vycházely z *Use Case* analýzy a zadání práce. Tabulka 4 ukazuje seznam vytvořených testovacích scénářů.

Tabulka 4: Testovací scénáře

Pořadí	Název scénáře
1	spuštění aplikace a výchozího rozvržení okna
2	rozvržení všech dialogových oken
3	založení definice TS
3a	nastavení parametrů TS
3b	založení, změna, odstranění symbolu
3c	založení, změna, odstranění stavu
3d	založení, změna, odstranění přechodové funkce
3e	změna pásky
4	uložení definice TS
4a	uložení definice TS - chybějící oprávnění
4b	uložení definice TS - existující soubor
4c	uložení definice TS - existující soubor, chybějící oprávnění
5	načtení definice TS
5a	načtení definice TS - chybějící oprávnění
5b	načtení definice TS - poškození datového souboru
6	simulace výpočtu TS
7	krokování výpočtu TS
8	pozastavení výpočtu TS
9	zastavení výpočtu TS
10	správná funkčnost zarážek
11	správná funkčnost informačního průvodce
12	správná funkčnost nápovědy
13	ukončení aplikace - neuložená definice TS
14	ukončení aplikace - běžící simulace

Tyto scénáře byly použity pro testování aplikace na těchto operačních systémech:

- *Microsoft Windows XP CZ*
- *Microsoft Windows Vista CZ*
- *Microsoft Windows 7 CZ*

Výsledky testování byly velmi dobré. V průběhu testování se nevyskytla žádná chyba aplikace. Vzhledem k rozsahu testovacích scénářů není možné je v této práci publikovat v plném rozsahu. Ukázka jednoho testovacího scénáře se záznamem výsledků testu, je uveden v příloze C.

4 Uživatelská dokumentace

4.1 Přehled základních vlastností

Kapitola popisuje základní vlastnosti aplikace *Simulátor TS* z uživatelského hlediska. Popis ovládání a uživatelského rozhraní naleznete v kapitole 4.2.

Základní vlastnosti

- Založení nového simulátoru
- Otevření, uložení a změny v definici simulátoru
- Simulace výpočtu a jeho řízení včetně rychlosti běhu
- Ladění simulátoru pomocí zarážek
- Krokování (vpřed i vzad) při simulaci výpočtu TS
- Možnost spuštění informačního průvodce
- Hromadné generování a hromadné odstranění dat

Vlastnosti definice TS

- Jednoduchá definice abecedy a vnitřních stavů
- Kontrola vazeb při definici TS
- Kontrola správné definice TS před spuštěním simulace
- Možnost změny názvů symbolů a stavů bez nutnosti měnit přechodovou funkci a pásku
- Přehledná, vždy aktuální statistika

Další nástroje

- Možnost nastavení zarážek a jejich snadné užití
- Řízení automatického umístění oken

`Uživatelské rozhraní`

- Jednoduché a přehledné uživatelské rozhraní
- Panel rychlého spuštění přehledně uspořádán v horní části aplikace
- Přehledně uspořádané jednotlivé okna definice TS
- Rozsáhlá nápověda popisující aplikaci i téma **Turingův stroj**

`Ostatní`

- Aplikaci není nutné instalovat
- Aplikace nevyžaduje žádné speciální nároky na hardware ani software

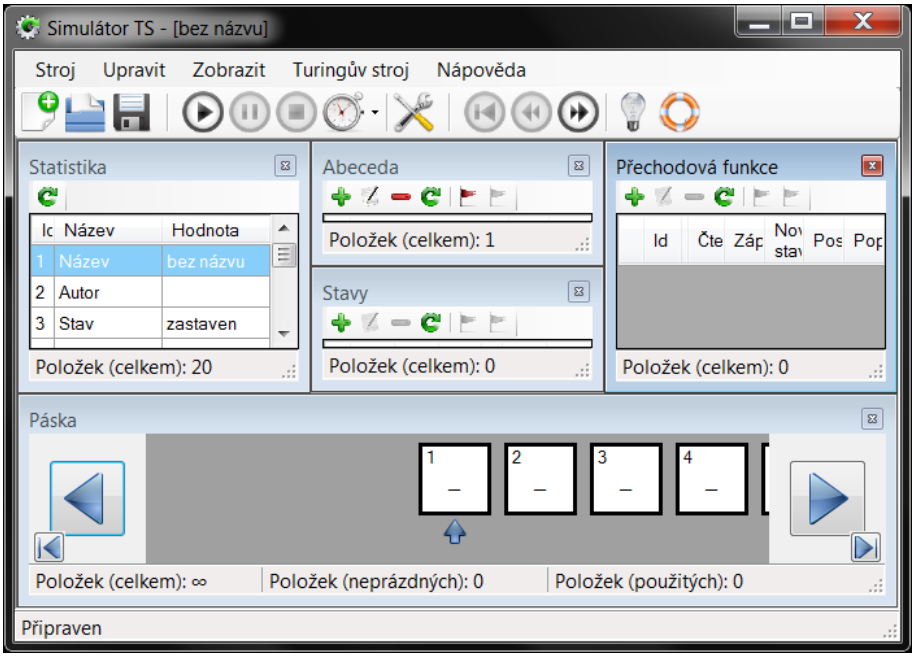
4.2 Uživatelské rozhraní

Uživatelské rozhraní aplikace *Simulátor TS* je podrobně popsáno v této kapitole. Součástí jsou i obrázky, aby čtenář měl lepší přehled, o které části aplikace text pojednává. Uživatelské rozhraní obsahuje tyto základní objekty:

Objekt: **`Hlavní okno`**
: Hlavní okno aplikace obsahuje základní prvky uživatelského rozhraní. Součástí je hlavní menu v horní části okna, dále pod hlavním menu program nabízí panel rychlého spuštění, který sdružuje základní příkazy programu. Všechny tyto příkazy je možné použít i z hlavního menu. V hlavní části okna jsou rozmístěna podřízená okna *Statistika*, *Abeceda*, *Stavy*, *Přechodová funkce* a *Páska*. Tyto okna jsou při spuštění aplikace zobrazena ve výchozím umístění. Ve spodní části hlavního okna je zobrazen stavový řádek. Ukázka hlavního okna je na obrázku 3.

Objekt: **`Hlavní menu`**
: Hlavní menu obsahuje základní příkazy programu rozdělené dle oblastí. Některé volby jsou v průběhu simulace nebo pozastavení simulace nedostupné.

Objekt: **`Panel rychlého spuštění`**
: Na panelu rychlého spuštění jsou umístěny tlačítka tak, aby uživatel měl snadný a rychlý přístup k nejpoužívanějším příkazům aplikace. Panel je rozdělen do několika oblastí, které se týkají založení nového TS a práce se soubory, dále pak simulace výpočtu TS a jeho řízení (včetně krokování) a poslední oblastí je nápověda včetně informačního průvodce.



Obrázek 3: Hlavní okno aplikace

Objekt: **`Stavový řádek`**

Stavový řádek je umístěn na spodním okraji hlavního okna. Aplikace zde zobrazuje informace o právě probíhajících nebo dokončených akcích. Informace ve stavovém řádku mají pouze informační charakter. Pokud aplikace oznamuje uživateli varovnou nebo kritickou informaci, nezobrazuje ji ve stavovém řádku, ale modálním dialogem. Stavový řádek kromě toho zobrazuje v průběhu simulace i animaci znázorňující průběh simulace.

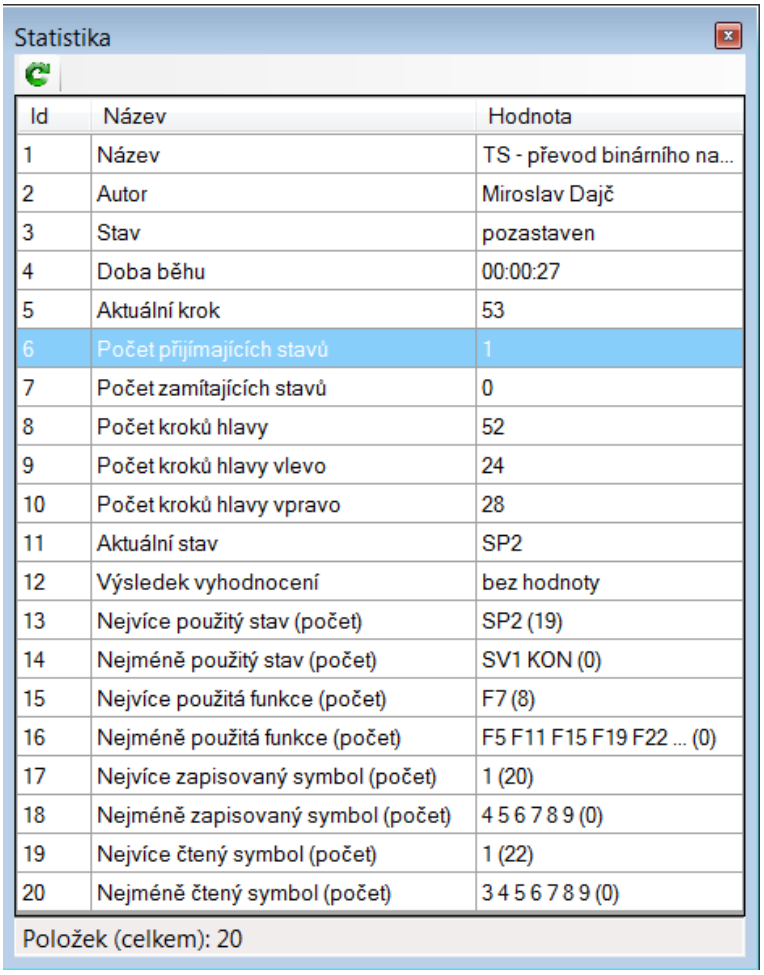
4.3 Základní okna

Okno: **`Statistika`**

Toto okno v aplikaci je určeno pro zobrazení statistiky výpočtu a stavu TS. Okno je rozděleno do tří částí, panel rychlého spuštění, datovou část a stavový řádek. Datová část obsahuje 20 základních statistik, které se při výpočtu TS pravidelně aktualizují. Řešení je koncipováno jako řádkový seznam, který obsahuje vždy pořadí, název statistického ukazatele a hodnotu. Celé okno je pouze informativní a kromě tlačítka aktualizace dat neobsahuje žádné aktivní funkce. Okno *Statistika* je na obrázku 4.

Okno: **`Abeceda`**

Podřízené okno *Abeceda* slouží pro definici a údržbu abecedy TS. Okno je rozděleno do tří částí, panel rychlého spuštění, datovou část a stavový řádek. Jednotlivé symboly jsou zobrazeny v datové části okna jako řádkový seznam, který obsahuje vždy pořadí, název symbolu a popis. Zvláštní



Obrázek 4: Okno *Statistika*

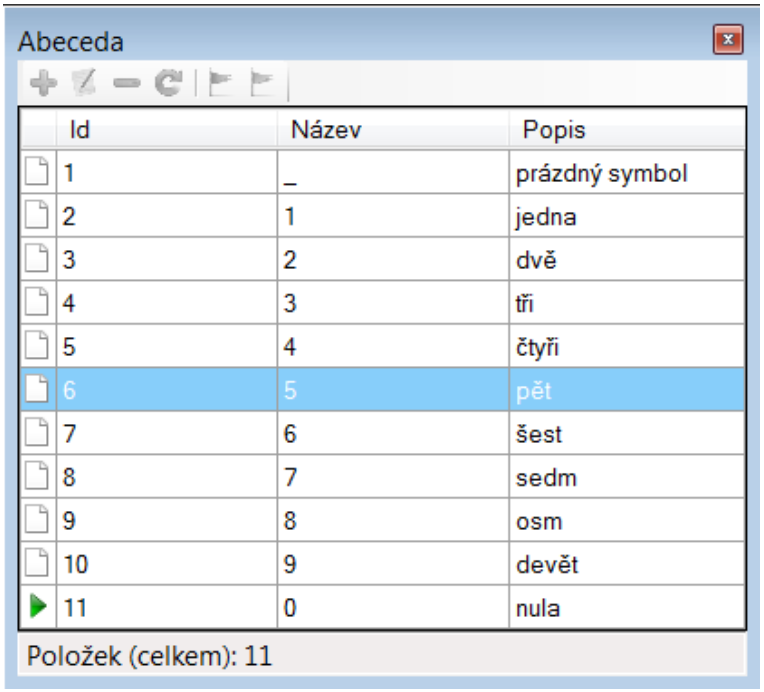
postavení mezi symboly abecedy má prázdný symbol. Ten je vždy v nové definice TS založen automaticky a má pořadí 1. Tento symbol nelze vymazat, pouze je možné změnit název a popis. Okno *Abeceda* je na obrázku 5.

Okno: **`Stavy`**

Aplikace *Simulátor TS* využívá pro definici a údržbu vnitřních stavů podřízené okno *Stavy*. Okno je rozděleno do tří částí, panel rychlého spuštění, datovou část a stavový řádek. Každá položka vnitřního stavu TS obsahuje pořadí, název vnitřního stavu a popis. Okno *Stavy* je na obrázku 6.

Okno: **`Přechodová funkce`**

Podřízené okno *Přechodová funkce* je v aplikaci určeno pro definici a údržbu přechodové funkce TS. Okno je rozděleno do tří částí, panel rychlého spuštění, datovou část a stavový řádek. Jednotlivé řádky seznamu v datové části okna představují jednotlivé definice přechodové funkce. Položka definice



Obrázek 5: Okno *Abeceda*

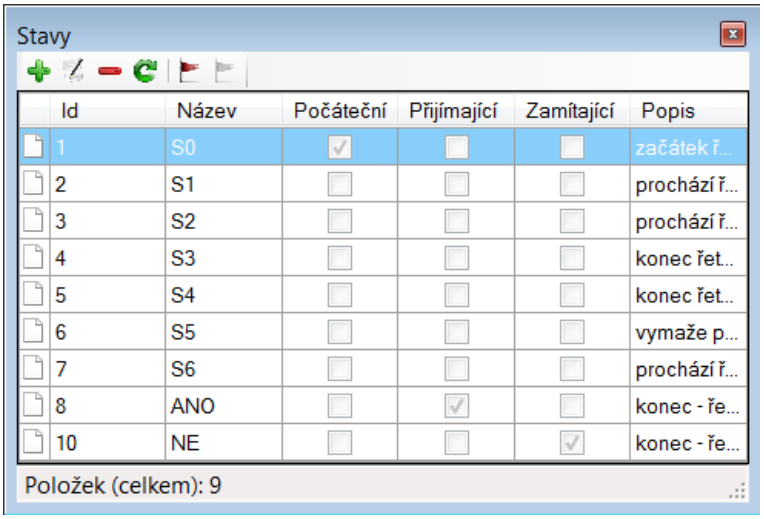
přechodové funkce se skládá z pořadí, (původního) stavu, čtení (symbolu z pásky), zápis (symbolu na pásku), nový stav, posun (směr posunu hlavy) a popis. Okno *Přechodová funkce* je na obrázku 7.

Okno: **`Páska`**

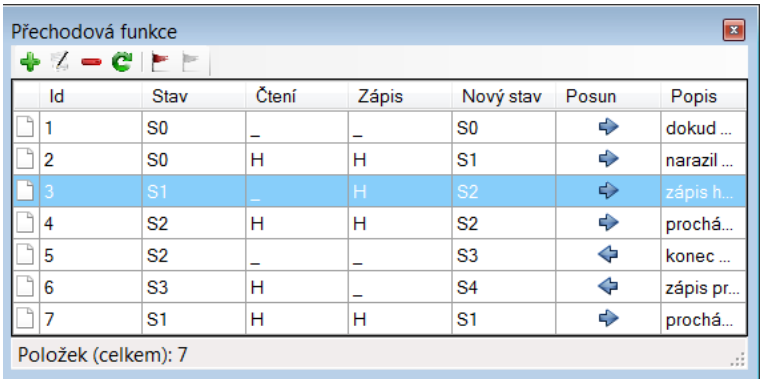
Okno Páska je jedním z nejdůležitějších oken aplikace Simulátor TS. Jedná se o okno představující pásku TS, které zobrazuje graficky jednotlivé buňky pásky včetně pozice čtecí (zapisovací) hlavy. Okno obsahuje v levé a pravé části panely pro posun zobrazované pásky. Hlavní část okna obsahuje grafické znázornění pásky. Jednotlivé buňky pásky jsou znázorněny čtvercovým panelem, který obsahuje pořadí buňky na pásce a hodnotu (symbol) zapsaný na pásce. Ve výchozím stavu je na pásce zapsán ve všech buňkách prázdný symbol a páska je pravostranně nekonečná. Počáteční buňka pásky má pořadí 1 a čtecí (zapisovací) hlava je nastavena na tuto první buňku. Hlavu znázorňuje šipka, která je zobrazena pod buňkou. Okna *Páska* je na obrázku 8.

Aplikace v průběhu simulace automaticky posunuje zobrazovanou část pásky tak, aby byla čtecí (zapisovací) hlava vždy viditelná. Podrobný popis tlačítek a jejich funkce je obsažen v nápovědě aplikace.

V případě, že simulace TS je zastavena, lze na každé buňce dvojklikem vyvolat dialog pro editaci položky. Poslední částí okna je stavový řádek, který je ve spodní části okna a zobrazuje několik hodnot:



Obrázek 6: Okno *Stavy*



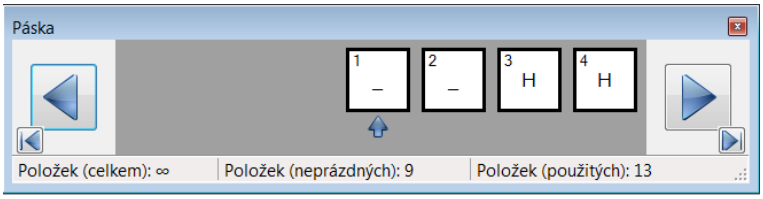
Obrázek 7: Okno *Přechodová funkce*

Hodnota: **`Položek (celkem)`**
: Hodnota je vždy nekonečno z důvodu pravostranně nekonečné pásky.

Hodnota: **`Položek (neprázdných)`**
: Hodnota zobrazuje údaj o počtu buněk na pásce, které obsahují jiný než prázdný symbol. Údaj je dobře použitelný při výpočtových simulacích.

Hodnota: **`Položek (použitých)`**
: Hodnota zobrazuje počet buněk, které jsou na pásce použité. Jsou to buňky, které obsahují vstupní slovo nebo je použil TS při simulaci. Počet se vždy počítá od první buňky po poslední použitou buňku.

Panely rychlého spuštění v podřízených oknech *Abeceda*, *Stavy* a *Přechodová funkce* sdružují tlačítka pro práci s položkami daného okna a tlačítka pro práci se zarážkami. Tlačítka na panelu rychlého spuštění jsou dostupná v těchto oknech,



Obrázek 8: Okno *Páska*

pouze pokud je simulace výpočtu TS zastavena. V tom případě lze na položkách dvojklikem vyvolat i dialogy pro editaci položek.

Pokud je zapnutý informační průvodce a simulace výpočtu TS je pozastavena, je na všech podřízených oknech (kromě okna *Statistika*), při najetí myši na každé z těchto oken zobrazován informační text. Více informací o *Informačním průvodci* naleznete v kapitole 4.4.

Pozice a velikost všech podřízených oken je udržována automaticky, pokud je zapnuta funkce *Automatické umístění oken*.

4.4 Informační průvodce

Informační průvodce je funkce aplikace, kterou lze zapnout buď v hlavním menu, nebo na panelu rychlého spuštění. Po zapnutí bude aplikace zobrazovat dodatečné informace o abecedě, stavech, přechodové funkci a pásce. Tato funkce zobrazuje vždy po najetí myši na datovou část příslušného okna informační text v „bublině“. Text se mění v závislosti na tom, zdali je simulace TS zastavena nebo pozastavena. Pokud simulace výpočtu TS běží, funkce informačního průvodce není aktivní.

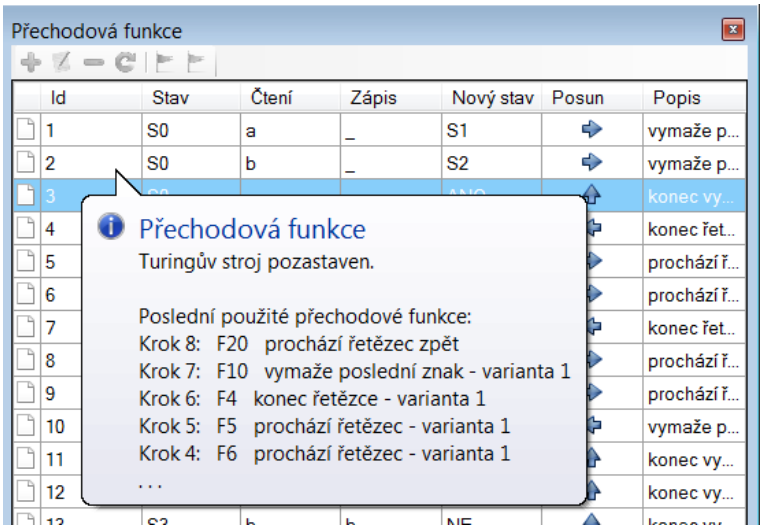
Pokud je simulace TS zastavena, zobrazuje informační průvodce vždy nad příslušným oknem (*Abeceda*, *Stavy*, *Přechodová funkce*, *Páska*) informační text vysvětlující funkci okna v aplikaci, popř. funkci dané části v definici TS.

V případě, že je simulace TS pozastavena, zobrazuje informační průvodce vždy nad příslušným oknem informace z průběhu simulace. Jedná se o zobrazení historie posledních pěti kroků simulátoru. Konkrétně je tato situace vidět na obrázku 9.

4.5 Nástroje

Nástroje aplikace slouží uživateli pro usnadnění některých postupů, např. ladění simulace TS nebo pro práci s definicí TS. Jedním z těchto nástrojů v aplikaci jsou **zarážky**.

Zarážky v aplikaci slouží k ladění simulace výpočtu TS. Lze je použít v podřízených oknech *Abeceda*, *Stavy* a *Přechodová funkce*. V každém z těchto oken je na panelu rychlého spuštění pro práci se zarážkami dvojice tlačítek (přidat zarážku, uvolnit zarážku). Pokud na některé položce je nastavena zarážka, dojde



Obrázek 9: Informační průvodce (zobrazení historie)

v průběhu simulace výpočtu TS na této položce k pozastavení. Průběh simulace bude pokaždé pozastaven pokud narazí aplikace v libovolném okně (*Abeceda*, *Stavy*, *Přechodová funkce*) na zarážku. Počet zarážek není nijak omezen. Zarážky lze jednotlivě přidávat a uvolňovat. Pro hromadné uvolnění všech zarážek má aplikace funkci *Odstranit všechny zarážky*. Tato funkce je dostupná v hlavním menu v části *Zobrazit*. Následuje popis tlačítek pro práci se zarážkami:

`Přidat zarážku`
: Na položce, která je označena kurzorem, bude nastavena zarážka. Položka je v levé části označena grafickým symbolem zarážky. Na položce, která je označena zarážkou, je simulace výpočtu vždy pozastavena.

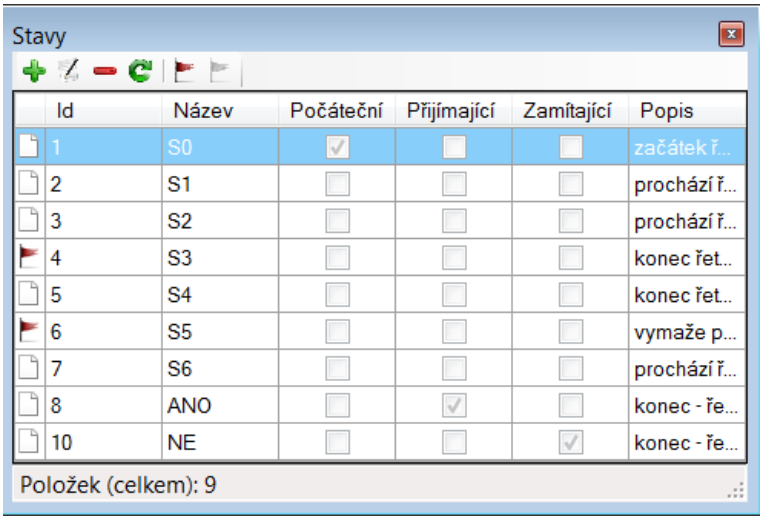
`Uvolnit zarážku`
: Na položce, která je označena kurzorem, bude uvolněna zarážka. Při nastavení zarážky je položka v levé části označena grafickým symbolem. Při uvolnění zarážky dojde k odstranění grafického symbolu v levé části položky.

Použití zarážek je zobrazeno na obrázku 10.

Dalším užitečným nástrojem aplikace jsou funkce pro hromadné generování a odstranění dat.

Funkce *Hromadné generování dat* je určena pro hromadné generování dat abecedy, stavů nebo pásky. Tato funkce je určena ke zjednodušení zakládání opakujících se položek. Vstupní údaje pro generování dat uživatel zadává v dialogu, který má odlišné vstupní parametry pro různé data generování.

Pro generování položek je nutné ve vstupním dialogu, který se otevře po spuštění funkce, zadat počet položek, kde uživatel určí kolik položek má aplikace



Obrázek 10: Použití zarážek v okně *Stavy*

vygenerovat. Tento vstupní parametr je shodný pro generování položek abecedy, stavů i pásky.

U generování položek abecedy je volitelným parametrem prefix, který má být na začátku v názvu jednotlivého symbolu. Např. pokud uživatel zadá počet položek `5` a prefix `Z`, budou vygenerovány položky abecedy s názvy `Z1, Z2, Z3, Z4,` a `Z5`. Pořadí položek abecedy určí aplikace automaticky.

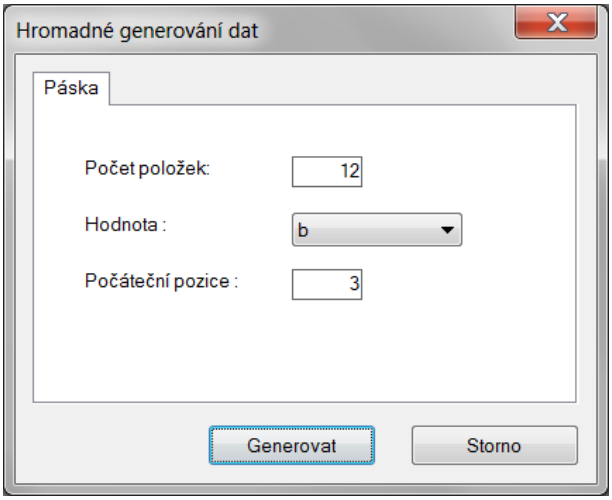
Obdobně u generování položek stavů je prefix, který má být na začátku v názvu jednotlivého stavu. Dalším vstupním parametrem u generování položek stavů je volba, zdali mají být stavy přijímající nebo zamítající.

Pro generování položek pásky je nutné zadat další povinné parametry. Jedná se o symbol z abecedy, který má být na pásku zapsán a pozice na pásce od které mají být data generována. Např. pokud uživatel zadá počet položek `4`, vybere symbol z abecedy `Z1` a zadá, že počáteční pozice je `1`, budou vygenerovány 4 buňky pásky od začátku pásky s hodnotou `Z1`. Generovat položky pásky lze i na pozice kde již hodnoty existují, v tomto případě aplikace původní hodnoty přepíše. Ukázka dialogu pro zadání vstupních parametrů pro generování dat pásky je na obrázku 11.

V souvislosti s hromadným generováním dat musíme zmínit i funkci *Hromadné odstranění dat*, která slouží k hromadnému odstranění položek abecedy, stavů, přechodové funkce nebo pásky.

Po spuštění funkce *Hromadné odstranění dat→Abeceda* aplikace zkontroluje zdali některý symbol z abecedy (kromě prázdného symbolu) nebyl dále použit v definici TS (v přechodové funkci nebo na pásce). Pokud kontrola proběhne v pořádku, vyzve aplikace znovu uživatele k potvrzení této akce. Pokud aplikace zjistí při kontrole, že některé symboly jsou dále v definici TS použity, zobrazí se varovné hlášení a položky nebudou odstraněny.

Všechny položky stavů lze odstranit obdobně pomocí funkce *Hromadné od-*



Obrázek 11: Dialog generování pásky

stranění dat→Stavy. Aplikace opět zkontroluje zdali některý stav nebyl dále použit v definici TS (přechodové funkci). Pokud kontrola proběhne v pořádku, vyzve aplikace znovu uživatele k potvrzení této akce. Pokud aplikace zjistí při kontrole, že některé stavy jsou dále v definici TS použity, zobrazí se varovné hlášení a položky nebudou odstraněny.

Pomocí funkce *Hromadné odstranění dat→Přechodová funkce* lze hromadně odstranit všechny položky přechodové funkce. Položky přechodové funkce lze odstranit vždy.

Data pásky lze hromadně odstranit pomocí funkce *Hromadné odstranění dat→Páska*. Data na pásce budou odstraněna, to znamená, že ve všech buňkách pásky bude zapsán prázdný symbol.

4.6 Systémové požadavky a instalace

Aplikace *Simulátor TS* je určena pro platformu *Microsoft Windows*. Podmínkou správné funkce aplikace je instalace rozhraní *.NET Framework 4*. Instalační balíček rozhraní je volně dostupný na webových stránkách společnosti *Microsoft* nebo jej lze nainstalovat z přiloženého CD/DVD které je součástí této práce.

Aplikaci není nutné instalovat, dostačuje pouze všechny součásti umístit do jednoho adresáře. Toto umístění (adresář) musí obsahovat soubory, které naleznete v příloze D. Mimo tyto soubory jsou v příloze uvedeny i datové soubory, které obsahují příklady ukázkových definic TS. Tyto soubory nevyžaduje aplikace ke správné funkci.

Aplikace je testována a plně funkční na těchto operačních systémech.

- *Microsoft Windows XP CZ*
- *Microsoft Windows Vista CZ*

- *Microsoft Windows 7 CZ*

Na každém z uvedených systémů byla aplikace testována podle testovacích scénářů, které naleznete v kapitole 3.7.

Závěr

Turingův stroj jako obecný model abstraktního výpočetního stroje byl v aplikaci *Simulátor Turingova stroje* znázorněn pomocí grafických komponent a bylo tak vytvořeno prostředí vhodné pro názorné simulace výpočtů toho stroje. Při vývoji aplikace nebylo ani tak důležité dosáhnout co nejefektivnější paměťové náročnosti nebo snad nejvyšší výkonosti, ale důraz byl kladen na přehlednost a jednoduchost aplikace, která by uživatelům dávala možnosti snadno pracovat s definicemi TS, studovat tak algoritmy a jejich vlastnosti, nahlížet v průběhu výpočtu na změny v konfiguracích a lépe tak pochopit problematiku Turingových strojů.

Samotná aplikace splňuje základní nároky a požadavky, které byly stanoveny v zadání této práce, ale autor aplikace již vidí další možnosti, které může znalejší uživatel postrádat. Velký prostor pro další vývoj aplikace je hlavně v možnostech importu dat definic v různých formátech, které by pomohly dosáhnout lepší přenositelnosti definic TS. Další příležitost pro rozšíření celého řešení je i možnost simulace nedeterministického TS, popř. vícepáskové verze TS. Tyto funkcionality by jistě zkompletovaly prostředí, které by poté poskytovalo uživatelům mnohostrannou využitelnost.

Conclusions

The Turing machine as a general model of an abstract computing machine was depicted in the Turing machine simulator application using graphical components and an environment suitable for the visual simulation of calculations of the machine was thus created. When developing the application neither achieving the most efficient memory consumption or perhaps the highest performance was so important, but emphasis was placed on the clarity and simplicity of an application, which would give users the possibility to easily work with TM definitions, to thereby study algorithms and their properties, to view the changes in the configurations during the calculation and thus better understand the issue of Turing machines.

The application itself meets the basic demands and requirements that were set in the assignment of work, but the designer of the application already sees other possibilities that the knowledgeable user may miss. Large space for further development of the application is mainly in the possibilities of importing data definitions in different formats, which would help achieve a better portability of TM definitions. The possibility of non-deterministic TM simulation, or the multiple-tape version of TM is also another opportunity for the expansion of the solution. Such functionality would certainly complete an environment, which would then provide users with versatile usability.

A Příloha - Use Case specifikace a diagram

Název

Otevření definice TS ze souboru

Krátký popis

Umožňuje otevření definice TS ze souboru XML včetně možných alternativ

Aktéři

- Uživatel
- Systém

Podmínky pro spuštění

Aplikace musí být spuštěna a musí být dostupná uložená definice TS v XML souboru.

Základní tok

1. Uživatel spustí funkci otevření definice TS.

2. Systém otevře dialogové okno pro výběr XML souboru s definicí TS.

3. Uživatel vybere soubor XML a potvrdí jeho otevření.

4. Systém zavře dialog pro otevření definice TS.

5. Systém validuje a načte definici TS.

Alternativní tok 2

2.1 Pokud není aktuální definice TS uložena, systém upozorní uživatele varovným hlášením, že dojde ke zrátě dat.

2.2 Uživatel potvrdí, že chce pokračovat.

2.3 Systém zavře varovné hlášení a otevře dialogové okno pro výběr XML souboru s definicí TS.

Alternativní tok 2.2

2.2.1 Uživatel ukončí funkci otevření definice TS.

2.2.2 Systém zavře varovné hlášení a ukončí funkci otevření definice TS.

Alternativní tok 3

3.1 Uživatel ukončí funkci otevření definice TS.

3.2 Systém zavře dialog pro otevření definice TS a ukončí funkci otevření definice TS.

Alternativní tok 5

5.1 Systém upozorní uživatele chybovým hlášením, že nelze načíst definici TS.

5.2 Uživatel potvrdí, chybové hlášení.

5.3 Systém zavře chybové hlášení a ukončí funkci otevření definice TS

Podmínky pro dokončení

Definice TS bude ze souboru XML načtena do aplikace.

Obrázek 12: Use Case diagram

Use Case diagram byl vytvořen v online aplikaci **draw.io** [6].

B Příloha - Ukázka XML souboru

```
1  <?xml version="1.0" encoding="windows-1250"?>
2  <!--datový soubor TS-->
3  <stroj>
4    <parametry>
5      <nazev>Příklad TS</nazev>
6      <autor>Miroslav Dajč</autor>
7      <popis>TS nahradí na pásce všechny symboly A symbolem B</popis>
8    </parametry>
9    <stavy>
10     <stav poradi="1" nazev="S1" pocatecni="True" prijimajici="False
           " zamitajici="False" popis="" />
11     <stav poradi="2" nazev="S2" pocatecni="False" prijimajici="True
           " zamitajici="False" popis="konec" />
12   </stavy>
13   <abeceda>
14     <symbol poradi="1" nazev="_" popis="prázdný symbol" />
15     <symbol poradi="2" nazev="A" popis="" />
16     <symbol poradi="3" nazev="B" popis="" />
17   </abeceda>
18   <prechodovafunkce>
19     <funkce poradi="1" stav="1" cteni="2" zapis="3" novystav="1"
          posun="1" popis="" />
20     <funkce poradi="2" stav="1" cteni="3" zapis="3" novystav="1"
          posun="1" popis="" />
21     <funkce poradi="3" stav="1" cteni="1" zapis="1" novystav="2"
          posun="0" popis="" />
22   </prechodovafunkce>
23   <paska>
24     <bunka poradi="1" hodnota="3" popis="symbol B" />
25     <bunka poradi="2" hodnota="3" popis="symbol A" />
26     <bunka poradi="3" hodnota="3" popis="symbol B" />
27     <bunka poradi="4" hodnota="1" popis="prázdný symbol"" />
28   </paska>
29 </stroj>
```

Zdrojový kód 3: Definice TS v XML souboru

C Příloha - Testovací scénář

Id

TC 1

Název

Otevření definice TS ze souboru

Krátký popis

Ověření postupu otevření definice TS ze souboru XML

Podmínky pro spuštění

Aplikace musí být spuštěna a musí být dostupná uložená definice TS v XML souboru.

Typ

Frontend

Potřebný čas

5 minut

Průběh testu

Kroky	Očekávání	Výsledek
1. Kliknout na hlavní Stroj	Rozbalení menu	OK
2. Kliknout na položku Otevřít	Otevření dialogového okna	OK
3. Vybrat platný XML soubor	Označí se vybraný soubor	OK
4. Kliknout na OK	Zavření dialogového okna	OK
5.	Načtení definice Abecedy	OK
6.	Načtení definice Stavů	OK
7.	Načtení definice Přechodové funkce	OK
8.	Načtení definice Pásky	OK
9.	Aktualizace dat statistiky	OK
10.	Informace ve stavovém řádku	OK
11.	Nastavení ovládacích prvků	OK

Výsledek testu

OK

D Obsah přiloženého CD/DVD

bin/
: Program SimulatorTS, spustitelný přímo z CD/DVD. Adresář obsahuje i všechny potřebné soubory pro bezproblémový běh programu z CD/DVD.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu).

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

readme.txt
: Instrukce pro spuštění programu SimulatorTS, včetně všech požadavků pro jeho bezproblémový provoz.

Navíc CD/DVD obsahuje:

data/
: Ukázková data definic TS.

Literatura

[1] MARTINEK, Pavel. *Základy teoretické informatiky* : učební text určený posluchačům bakalářského studijního programu Aplikovaná informatika, provozovanému v kombinované formě na Přírodovědecké fakultě Univerzity Palackého v Olomouci [online].
Dostupný z: http://phoenix.inf.upol.cz/esf/ucebni/zti.pdf.

[2] WIKIPEDIE. *Turingův stroj* : popis Turingova stroje, definice, problémy, možné modifikace [online].
Dostupný z: http://cs.wikipedia.org/wiki/Turing%C5%AFv_stroj.

[3] JANČAR, Petr aj. *Úvod do teoretické informatiky* : učební text určený pro studijní obor Informatika a výpočetní technika fakulty elektrotechniky a informatiky na Vysoké škole báňské – Technická univerzita Ostrava [online].
Dostupný z: http://www.cs.vsb.cz/kot/down/uti2007/uti-text-new.pdf.

[4] MICROSOFT. *Microsoft Visual Studio Express 2013 for Windows Desktop* : informace o vývojovém prostředí [online].
Dostupný z: https://www.visualstudio.com/.

[5] GITHUB.COM. *pasnox/oxygen-icons-png* : sada icon a PNG obrázků [online].
Dostupné z: https://github.com/pasnox/oxygen-icons-png

[6] DRAW.IO. *draw.io* : online aplikace pro vytváření diagramů [online].
Dostupné z: https://www.draw.io

[7] BRATKOVÁ, Eva (sest.). *Metody citování literatury a strukturování bibliografických záznamů podle mezinárodních norem ISO 690 a ISO 690-2* : metodický materiál pro autory vysokoškolských kvalifikačních prací [online].
Dostupný z: http://www.evskp.cz/SD/4c.pdf.