PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Simulátor Turingova stroje

s oboustranně nekonečnou páskou

2011 Lukáš Ogoun

Anotace

Cílem této bakalářské práce bylo vytvořit aplikaci pro operační systém MAC OS X, která bude simulovat práci Turingova stroje s oboustranně nekonečnou páskou, kdy veškeré vstupy, jako je zadání vstupního slova či přechodové funkce, bude zadávat uživatel, u nehož se předpokládá principielní znalost problematiky.

Děkuji Mgr. Petrovi Osičkovi za vedení mé bakalářské práce a za konstruktivni připomínky při jejím vzniku. Největší dík však patří mé rodině, především jejichž čas jsem přípravě práce „obětoval“.

Obsah

1. Úvod **7**

2. Turingův stroj **8**
2.1. Teorie Turingova stroje 8
2.2. Definice Turingova stroje 9
2.3. Formální definice Turingova stroje 10
2.4. Konfigurace Turingova stroje 10
2.5. Přechodová funkce Turingova stroje 11
2.6. Krok výpočtu turingova stroje 11
2.7. "Busy Beaver", nejznámější příklad Turingova stroje 11
2.8. Detaily potřebné pro vizualizaci nebo implementaci Turingova stroje 12
2.9. Univerzální Turingovy stroje 12
2.10. Srovnání se skutečnými stroji 12
2.11. Omezení Turingových strojů 14

3. Alan Turing, život a dílo **15**

4. Apple, Objective-C a Cocoa Framework **18**
4.1. Mac OS X 18
4.2. Cocoa Framework 18
4.3. Objective-C 18

5. Uživatelská dokumentace aplikace TMS **20**
5.1. Instalace aplikace 20
5.2. Spuštění aplikace 20
5.3. Popis aplikace 20
5.3.1. State 20
5.3.2. Tape content 21
5.3.3. Initial configuration 21
5.3.4. Controls 21
5.3.5. Transition function input 21
5.3.6. Transition functions 22
5.3.7. Configuration history 22
5.4. Menu 22
5.4.1. File - New Turing Machine Simulator 23
5.4.2. File - Open Turing Machine Configuration 23
5.4.3. File - Save Turing Machine Configuration 25
5.5. Ukončení aplikace 25
5.6. Odinstalace aplikace 25

6. Programátorská dokumentace aplikace TMS **26**
6.1. Vývojové prostředí, použité nástroje 26
6.2. Use Case Diagram 27
6.3. Diagram tříd 28
6.4. Třídy a jejich metody, funkce 29
6.4.1. Třída “TMSimulator” 29
6.4.2. Třída “TMTape” 30
6.4.3. Třída “TMTransArray” 31
6.4.4. Třída “TMTransition” 32
6.4.5. Třída “TMConfiguration” 32
6.5. Třída “TMConfigArray” 32

Závěr **34**

Conclusions **35**

Reference **36**

A. Obsah přiloženého CD **38**

Seznam obrázků

1. Alan Turing kolem r. 1948 15
2. Hlavní okno aplikace TMS 20
3. Menu - File - New TMS 23
4. Menu - File - Open TMS 23
5. Menu - File - Save TMS 25
6. Use Case Diagram TMS 27
7. Class Diagram TMS 28

1. Úvod

Tato aplikace byla vytvořena za účelem pratického znázornění práce turingova stroje pro potřeby studentů teoretické informatiky, kteří jako primární operační systém používají MAC OS X, tedy byla vyvinuta ve vývojovém prostředí Xcode, v programovacím jazyce Objective-C 2.0 za použití frameworku Cocoa.

Aplikace je nazvaná „Turing Machine Simulator“, dále jen „TMS“. Jako „TMS“ je označena i u popisů v samotné aplikaci a to především z mnoha praktických důvodů.

Specifikace zadání, požadavky:

- Simulátor Turingova stroje s oboustranně neomezenou páskou,
- zadání vstupního slova uživatelem a jeho editace,
- zadání libovolného algoritmu pomocí přechodové funkce uživatelem a její editace,
- možnost volby z různých způsobů krokování výpočtu turingova stroje,
- uživatelská dokumentace,
- programátorská dokumentace.

2. Turingův stroj

2.1. Teorie Turingova stroje

Turingův stroj je teoretické zařízení, které manipuluje se symboly na části pásky v závislosti na uvedených přechodových funkcích - pravidlech. Navzdory své jednoduchosti se může Turingův stroj přizpůsobit a simulovat logiku jakéhokoliv počítačového algoritmu a částečně je využitelný i při výkladu funkce počítačového mikroprocesoru.[7]

„Turingův" stroj byl popsán Alanem Turingem v roce 1936, kdy jej Turing nazýval „a-machine" nebo „automatic-machine". Zámýšlenou funkcí Turingova stroje nebylo jeho praktické využití jako výpočetního zařízení, ale spíše myšlenkový experiment, reprezentující výpočetní stroj. Turingův stroj pomáhá počítačovým vědcům pochopit omezení mechanických výpočtů.[3]

Turing popsal stručnou definici tohoto experimentu ve své eseji „Intelligent Machinery" z r. 1948. Turing napsal, že Turingův stroj (zde nazývaný „Logický Výpočetní Stroj") se skládá:

...z nekonečné paměťové kapacity ve formě pásky sestávající se z jednotlivých polí, kdy na jakékoliv z nich může být zapsán symbol. V jakémkoliv okamžiku je ve stroji jeden symbol, nazývaný „čtený symbol". Stroj může změnit čtený symbol a jeho chování je zčásti tímto symbolem ovlivněno, ale symboly umístěné jinde na pásce aktuální chování stroje neovlivní. Páska se nicméně skrze stroj může posouvat zpět i dopředu, což je jedna ze základních činností stroje, může být tedy nakonec změněn jakýkoliv symbol na pásce.[5]

Turingův stroj, který je schopný simulovat jakýkoliv jiný Turingův stroj se nazývá „univerzálni Turingův stroj" (UTM). Více matematicky orientovaná definice s podobnou „univerzální" povahou byla představena Alonzem Churchem, jehož práce na lambda kalkulu se prolínala s Turingovou prací ve formální teorii výpočtů, známé dnes jako Church-Turingova teze. Tato teze uvádí, že Turingovy stroje zachycují představu účinné metody v logice a matematice a poskytují přesnou definici algoritmu neboli „mechanické procedury".

„Studium jejich abstraktních vlastností nabízí mnoho pohledů do počítačové vědy a teorie složitosti."[7]

Podle původního článku („On computable numbers, with an application to the Entscheidungsproblem") si Turing nepředstavoval mechanismy, ale člověka, kterého nazýval „počítač" a který otrocky vykonával tato deterministická mechanická pravidla:

„...Chování počítače v jakémkoliv okamžiku je určeno symboly, které čte, a jeho „stavem mysli" v daném okamžiku. Lze předpokládat, že existuje vazba B na určitý počet symbolů nebo polí, které může počítač číst v jednom okamžiku. Pokud jich chce číst více, musí je číst postupně. Budeme také předpokládat, že počet stavů mysli, které je potřeba brát v úvahu, je konečný. Důvody jsou stejné, jako ty, které omezují počet symbolů. Přijmeme-li nekonečný počet stavů

mysli, pak si budou některé náhodně blízké a budou se plést. Opět platí, že omezení není takové, aby vážně ohrozilo výpočet, protože se lze vyhnout použití složitějších stavů mysli tím, že na pásku zapíšeme větší počet symbolů. Každá taková operace znamená změnu fyzického systému, který je tvořen počítačem a jeho páskou. Stav systému známe tehdy, pokud známe posloupnost symbolů na pásce, které symboly jsou počítačem čteny (případně v konkrétním pořadí) a stav mysli počítače. Můžeme předpokládat, že v jedné oparaci se nezmění více, než jeden symbol. Jakékoliv další změny mohou být prováděny v samostatných operacích tohoto typu. Umístění polí, jejichž symboly mohou být tímto způsobem změněny, je stejné, jako těch polí, která jsou čtena. Můžeme tedy bez ztráty zobecnění předpokládat, že pole, jejichž symboly jsou čteny, jsou vždy ta pole, která byla přečtena.[6]

„...Představme si, že operace vykonávané počítačem jsou rozděleny na „jednoduché operace“, které jsou natolik elementární, že si lze jejich další dělení jen těžko představit...““[6]

2.2. Definice Turingova stroje

Přesněji se Turingův stroj sestává z:

1. **Pásky**, rozdělené na pole, poskládaná vedle sebe. Každá buňka obsahuje symbol z nějaké abecedy - konečné množiny znaků. Abeceda vždy obsahuje speciální prázdný symbol a jeden nebo více dalších symbolů. U pásky se předpokládá, že se může libovolně rozšiřovat vlevo i vpravo, Turingův stroj tedy vždy obsahuje tak velkou pásku, jak velkou potřebuje pro své výpočty. Pole, na která nebylo dříve zapisováno, jsou „naplněna“ prázdným symbolem. V některých modelech jsou pásky zleva ohraničené speciálním symbolem a páska se může nekonečně rozšiřovat pouze doprava.

2. **Hlavy**, která umí číst a psát symboly na pásce a posouvat pásku doleva a doprava, vždy o jednu buňku v jednom výpočetním kroku. V některých modelech se pohybuje hlava, páska je statická.

3. **Konečné tabulky instrukcí** (někdy nazývané tabulka akcí nebo přechodových funkcí). Sestávající z 5-tic, kde stav, ve kterém se stroj aktuálně nachází a symbol na pásce, který je aktuálně pod hlavou (hlava jej čte), určuje stroji učinit následující (v daném pořadí):

   - vymazat nebo zapsat symbol (namísto přečteného),
   - posunout hlavu doprava, doleva nebo zůstat na místě,
   - přejít do nového stavu (může být stejný nebo odlišný od původního).

   Ve 4-ticových modelech jsou kroky mazání nebo zapisování symbolu a posun hlavy doleva, doprava nebo ponechání na místě, určeny samostatnými

instrukcemi. Konkrétně tabulka říká stroji, aby smazal nebo zapsal symbol nebo posunul hlavu doleva, doprava, či nikam, ale nikoliv obě akce v jedné instrukci. V některých modelech je určeno, že pokud v tabulce neexistuje instrukce pro aktuální kombinaci symbolu a stavu, stroj zastaví. Jiné modely vyžadují, aby byly všechny položky vyplněny.[15]

4. **Stavového registru**, který ukládá stav Turingova stroje - jeden z konečně mnoha. Je zde jeden speciální počáteční stav, se kterým je stavový registr inicializován. Turing píše, že tyto stavy nahrazují „stav mysli“, ve kterém je osoba, provádějící výpočty.

2.3. Formální definice Turingova stroje

Turingův stroj M je sedmice:

$$\boldsymbol{M = \{Q, \Gamma, b, \Sigma, \delta, q_0, F\}}$$

, kde

- $\boldsymbol{Q}$ je konečná neprázdná množina stavů,
- $\boldsymbol{\Gamma}$ je konečná neprázdná množina páskové abecedy/symbolů,
- $\boldsymbol{b \in \Gamma}$ je prázdný symbol (jediný symbol, který se může na pásce vyskytovat nekonečně-krát v kterémkoliv kroku výpočtu),
- $\boldsymbol{\Sigma \subseteq \Gamma \setminus \{b\}}$ je množina vstupních symbolů,
- $\boldsymbol{q_0 \in Q}$ je počáteční stav,
- $\boldsymbol{F \subseteq Q}$ je množina konečných resp. přijímajících stavů,
- $\boldsymbol{\delta : Q \setminus F \times \Gamma \to Q \times \Gamma \times \{L, R\}}$ je přechodová funkce.

2.4. Konfigurace Turingova stroje

Konfigurací Turingova stroje nazýváme prvek $\boldsymbol{\langle q, s, n \rangle}$
z množiny $\boldsymbol{Q \times \{yb^{\omega} | y \in \Gamma^*\} \times \mathbb{N}_0}$, kde:

- **q** je aktuální stav,
- **s** je nejmenší souvislá část pásky, obsahující všechny neprázdné symboly,
- **n** je pozice čtecí hlavy.

2.5. Přechodová funkce Turingova stroje

$\boldsymbol{Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}}$, kde:

- $\boldsymbol{Q \setminus F}$ je aktuální stav,
- $\boldsymbol{\Gamma}$ je čtený znak na páskovém slově,
- $\boldsymbol{Q}$ je stav, do kterého TS přechází,
- $\boldsymbol{\Gamma}$ je symbol, který zapisuje na pole pod čtecí (zapisovací) hlavou,
- $\boldsymbol{\{L, R, N\}}$ je směr, jakým se posune ev. neposune čtecí (zapisovací) hlava.

2.6. Krok výpočtu turingova stroje

Na začátku výpočtu je Turingův stroj v počáteční konfiguraci a na pásce je zapsané vstupní slovo.
Dále pracuje v jednotlivých krocích:

1. Pokud je aktuální stav zároveň stavem koncovým, výpočet končí,
2. čtecí hlava přečte jeden vstupní symbol z buňky, na které se právě nachází,
3. pokud je v přechodové funkci pro aktuální stav a pro přečtený symbol definovaný přechod, provede se:
   - změní se stav,
   - na aktuální pozici hlavy se zapíše příslušný symbol,
   - hlava se příslušným způsobem posune (či neposune).

2.7. "Busy Beaver", nejznámější příklad Turingova stroje

V teorii vyčíslitelnosti je nazýván jako „Busy Beaver“ takový Turingův stroj, který dosahuje maximálního "provozního zatížení" (například měřeno počtem provedených kroků nebo počtem neprázdných symbolů na pásce v jejím konečném stavu) mezi všemi Turingovy stroji v dané třídě. Turingův stroj této třídy musí splňovat určité specifikace návrhu a musí se nakonec zastavit poté, co svou práci zahajoval s prázdnou páskou.

Funkce „Busy Beaver“ vyčísluje tyto horní limity pro dané typy „provozního zatížení“ a je nerekurzivní. Ve skutečnosti lze dokázat, že funkce „Busy Beaver“ roste asymptoticky rychleji, než jakákoliv rekurzivní funkce. Tento koncept poprvé představil Tibor Radó, jako „Busy Beaver game“ ve svém článku „On Non-Computable Functions“ z r. 1962.[12]

2.8. Detaily potřebné pro vizualizaci nebo implementaci Turingova stroje

„Teoreticky popsaný model Turingova stroje poskytuje pouze částečné informace o tom, jak se bude stroj chovat a jak budou vypadat jeho výpočty.“[13]

Je třeba například rozhodnout, jak budou symboly ve skutečnosti reprezentovány či způsob implementace nekonečného čtení a zapisování symbolů.

Posun doleva a doprava může pohybovat čtecí hlavou na pásce. Při modelování (implementaci) Turingova stroje je praktičtější posouvat pásku pod čtecí hlavou, namísto pohybu hlavou.

Páska může být konečná a automaticky se prodlouží použitím prázdných symbolů až v případě potřeby. Je to obvyklejší, než uvažovat, jak nekonečně prodloužit pásku na obou koncích a přeplňovat ji prázdnými symboly.

Páska nemůže mít stanovenu pevnou délku, protože by to neodpovídalo dané definici a vážně by tím byl omezen rozsah výpočtů, které stroj může vykonávat. K tomu se více hodí lineárně omezené automaty.

2.9. Univerzální Turingovy stroje

Univerzální Turingův stroj je takový stroj, který dokáže simulovat libovolný Turingův stroj na libovolném vstupu. Univerzální stroj toho v podstatě dosahuje tím, že čte jak popis stroje, který má simulovat, tak i vstupy na své vlastní pásce. Alan Turing tento stroj představil na přelomu let 1936 a 1937. Tento model je některými považován za původce počítačového programu uloženého v počítači, který v roce 1946 použil John von Neumann pro své „Elektronické Výpočetní Zařízení“ a který dnes nese Neumannovo jméno: von Neumannova architektura. Univerzální Turingův stroj je známý také jako univerzální výpočetní stroj, univerzální stroj, U stroj nebo U.[3]

Z pohledu výpočetní složitosti je vícepáskový Turingův stroj v porovnání se stroji, které simuluje, logaritmicky pomalejší.[3]

2.10. Srovnání se skutečnými stroji

Často se říká, že Turingovy stroje jsou narozdíl od jednodušších automatů stejně výkonné, jako skutečné stroje a jsou schopné vykonat jakoukoliv operaci, jako skutečný program. Co zde ale není zmíněno je fakt, že vzhledem k tomu, že se skutečný stroj může ocitnout v konečně mnoha konfiguracích, je tento „skutečný stroj“ pouze lineárně omezený automat. Na druhou stranu, Turingovy stroje jsou ekvivalentní strojům, které mají neomezené množství úložné kapacity pro své výpočty. Turingovy stroje ve skutečnosti nejsou určeny k modelování počítačů, ale jsou určeny k modelování samotných výpočtů; počítače, které počítaly pouze na svých vnitřních pamětech pevné velikosti, byly vyvinuty až později.[3]

Existuje celá řada způsobů, jak vysvětlit, proč jsou Turingovy stroje užitečné jako modely skutečných počítačů:

- Cokoliv, co může spočítat skutečný počítač, může spočítat i Turingův stroj. Například: „Turingův stroj může simulovat jakýkoliv typ podprogramu nacházející se v programovacích jazycích, včetně rekurzivních procedur a jakéhokoliv známého mechanismu, předávajícího parametry.“[9]

  Dostatečně velké konečné automaty také mohou modelovat skutečné počítače bez ohledu na vstupy a výstupy. Prohlášení o omezeních Turingova stroje platí tedy i o skutečných počítačích.[3]

- Rozdíl spočívá pouze ve schopnosti Turingova stroje manipulovat s neomezeným množstvím dat. Vzhledem k tomu, že je dáno konečné množství času, Turingův stroj (stejně jako ten skutečný) může manipulovat pouze s konečným množstvím dat.[3]

- Stejně jako Turingův stroj i skutečný stroj může mít svůj ukládací prostor rozšířen podle potřeby tím, že se mu přidá více disků nebo jiných paměťových médií. Pokud je jich nedostatek, nemusí být Turingův stroj jako model počítače dostatečně účelný. Faktem ale je, že ani Turingovy stroje, ani skutečné stroje, nepotřebují astronomické množství paměti k tomu, aby mohly vykonávat užitečné výpočty. Čas potřebný ke zpracování bývá obvykle daleko větší problém.[3]

- Popis programu reálného stroje pomocí jednodušších abstraktních modelů je často mnohem složitější, než popis pomocí Turingova stroje. Turingův stroj může například popisovat algoritmus mající několik stovek různých stavů, zatímco ekvivalentní determinstický konečný automat jich má pro daný skutečný stroj kvadrilion, což činí analýzu pomocí deterministického konečného automatu za nemožnou.[3]

- Turingovy stroje popisují algoritmy nezávisle na tom, kolik paměti potřebují. Je zde limit daný aktuální potřebou jakéhokoliv stroje, ale tento limit může dále v čase libovolně růst. Turingovy stroje nám umožňují vyslovovat o algoritmech tvrzení, která jsou (teoreticky) neměnná, bez ohledu na pokrok architektur konvenčních výpočetních strojů.[3]

- Turingovy stroje zjednodušují vyjádření algoritmů. Algoritmy na abstraktních strojích, ekvivalentních Turingovým strojům, jsou obvykle obecnější, než jejich protějšky běžící na skutečných strojích, protože mají k dispozici libovolné a přesné datové typy a nikdy nemusí řešit neočekávané situace (například běh mimo dostupnou paměť).[3]

Jeden případ, kdy jsou Turingovy stroje pro programy nevyhovující, je velké množství skutečných programů, jako např. operační systémy, textové procesory,

které jsou psány tak, aby přijímaly v průběhu času neomezené vstupy a proto nezastaví. Turingovy stroje nemodelují takové výpočty správně (stále ale mohou modelovat jejich části, jakými jsou například samostatné procedury).

2.11. Omezení Turingových strojů

Omezení Turingových strojů jsou taková, že nemodelují správně možnosti speciálních režimů. Například moderní počítače s uloženým programem jsou vlastně příklady konkrétnějších podob abstraktních strojů, známých jako programové stroje s náhodným přístupem.[3]

Stejně jako Univerzální Turingův stroj, i program s náhodným přístupem ukládá svůj „program“ do „paměti“, která je oddělená od instrukcí jejich konečněstavového stroje. Narozdíl od univerzálního Turingova stroje má program s náhodným přístupem nekonečný počet rozlišitelných, spočitatelných, ale neomezených „registrů“ - paměťových „buněk“, které mohou obsahovat libovolně velké celé číslo.[10].

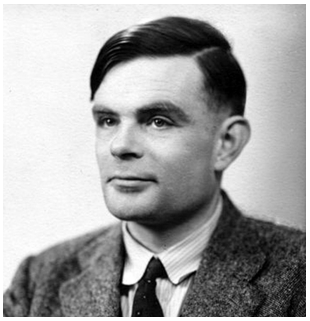
Konečněstavové stoje s programy s náhodným přístupem jsou vybaveny možností nepřímého adresování (například obsah jednoho registru může být použitý jako adresa jiného registru); tak může program s náhodným přístupem adresovat jakýkoliv registr v posloupnosti registrů. Výsledkem tohoto rozdílu je, že existují optimalizace výpočtů, které mohou být prováděny na základě ukazatelů paměti, které není možné v Turingově stroji realizovat; a tak, když jsou Turingovy stroje používány jako základ pro časově omezený běh výpočtu, může být na některých provozních časech algoritmů nalezena dolní mez (s ohledem na zjednodušený předpoklad Turingova stroje). Příklad takového binárního vyhledávání je algoritmus, u kterého lze prokázat, že běží rychleji, když používá namísto Turingova stroje program s náhodným přístupem.[3]

Dalším omezením Turingových strojů je, že neumí správně modelovat paralelismus. Máme-li například vazbu na velikost celého čísla, které může být vypočítáno vždy-zastavujícím nedeterministickým Turingovým strojem, začínajícím s prázdnou páskou. Naopak, existují vždy zastavující paralelní systémy bez vstupů, které mohou vypočítat celé číslo neomezené velikosti (takový proces může být vytvořen na lokálním úložišti, které je inicializováno 0 a dále takový proces sám sobě současně vysílá zastavující a spouštěcí zprávy. Jakmile přijme zastavující zprávu, zastaví s neomezeným číslem ve svém lokálním úložišti).[3]

3. Alan Turing, život a dílo

Alan Mathison Turing se narodil 23. června 1912 jako druhé dítě Julia Mathisona a Ethel Sáry Turing. Neobvyklé jméno Turing ho řadilo do významného rodokmenu anglické šlechty, i když ne bohaté, ale určením do vyšší střední třídy, ve smyslu anglického třídního systému.[8]

Dnes je Alan Turing známý jako zakladatel informatiky, původce dominantní vědy konce dvacátého století, ale toto mu v době jeho života nebylo připisováno. Stejně tak jeho dětství nenapovídalo, že by se tak někdy mohlo stát.[8]



Obrázek 1. Alan Turing kolem r. 1948

Jméno Turing bylo známo především díky práci bratra jeho otce Julia o muškaření, která neměla nic společného s vědeckou nebo akademickou činností. Vliv na Alana měla pravděpobně technicky založená rodina jeho matky s jejich zaměřením na aplikované vědy, ale to vše bylo podřízeno třídě, církvi a impériu, což naopak nepotvrzoval Alanův starší bratr John F. Turing, který se stal advokátem v Londýně. Příběh Alana Turinga nevychází z rodiny nebo tradice, ale z jeho izolované a samostatné mysli.[8]

Alan Turing sdílel se svým bratrem dětství, jasně určené požadavky třídy a exilem jeho rodičů do Indie. Až do roku 1926, kdy se jeho otec vrátil z Indie, byl Alan Turing spolu se svým bratrem vychováván v různých anglických domovech, kde nemohl rozvíjet své nadání, jedinečnost či schopnost objevování. Věda pro něj byla mimoškolním zájmem, který se poprvé projevil v podobě primitivních chemických pokusů. Dostal ale populární knihu "Natural Wonders Every Child Should Know" („Přírodní divy, které by mělo znát každé dítě"), která na něj měla dle jeho slov klíčový vliv.[8]

Vědecké zájmy v jeho dětství byly zkouškou pro jeho matku, jejíž trvalou hrůzou bylo, že nebyl přijatelný pro žádnou anglickou státní školu. Přesto byl nakonec přijat na školu Sherborne a brzy poté ředitel školy prohlásil: „Pokud má být jen vědecký specialista, státní škola je pro něj plýtváním času". Posouzení jeho založení bylo téměř správné.[8]

Zásadní podnět přišel k Turingovi od jeho spolužáka z vyššího ročníku - Christophera Morcoma, také velmi schopného člověka, ve kterém se Alan Turing v roce

1928 zhlédl a Morcom tím dal vzniknout - Turingově důležitému - období intelektuálního přátelství, které však skončilo náhlou Morcomovou smrtí v r. 1930 (zemřel na tuberkulózu).[8]

Turingovo přesvědčení, že nyní musí pokračovat v tom, v čem Morcom nemohl, ho uvrhlo do dlouhodobé krize. Tři roky za sebou, jak je známo z jeho dopisů Morcomově matce, se jeho myšlenky zaměřovaly na otázku, jak je lidská mysl - a Christopherova především - ztělesněna ve hmotě a zda může být z této hmoty vysvobozena smrtí.[8]

Tato otázka ho vedla hlouběji do oblasti fyziky dvacátého století, kde mu pomáhala kniha "The Nature of the Physical World" a přemýšlel, jestli kvantověmechanické teorie ovlivnily tradiční problémy mysli a hmoty.[8]

Jeho zásadní práci, založenou na vztahu mezi logickými instrukcemi, činnostmi mysli a strojem, který by mohl být ztělesněn v praktické fyzické formě, představil v dubnu 1936. V tu samou dobu ale došel k totožnému závěru americký logik Alonzo Church, čímž Turing přišel o vavříny za svůj objev. Ve svém článku "On Computable Numbers with an application to the Entscheidungsproblem" se musel odkazovat na Churchovu práci a vydání bylo zpožděno až do srpna 1936. Nicméně již tehdy bylo patrné, že Turingův přístup byl jiný. Church se spoléhal na vnitřní matematické předpoklady, spíše než by se odvolával na činnosti, které by mohly být ve skutečnosti prováděny reálnými věcmi nebo lidmi ve skutečném světě. Následně se stal koncept Turingova stroje základem moderní teorie výpočtu a vyčíslitelnosti.[8]

Turing se zásadní mírou zasloužil o vznik stroje zvaného „Bombe", který dokázal v roce 1940 rutinně dekódovat zprávy posílané mezi německými jednotkami Luftwaffe během 2. světové války. Daleko komplikovanější prostředek - Enigmu, kterou používalo německé námořnictvo - dokázal Turing prolomit již na konci roku 1939, ale pro rozluštění komunikace bylo potřeba získat další materiál. To se podařilo a rutinní dešifrování začalo v polovině roku 1941 a zásadním způsobem ovlivnilo vývoj konce druhé světové války.[8]

V roce 1948 proběhla v Manchesteru světově první praktická demonstrace principů Turingova stroje. Zasadil se o ní tehdejší radarový inženýr F. C. Williams, za přispění velkého grantu Královské Společnosti, který zajistil M. H. A. Newman, topologista v Cambridge.[8]

Ačkoliv Turing nikdy netajil svou homosexualitu, po svém návratu na Cambridge v roce 1948 byl záměrně ještě otevřenější a rozbujelejší a jeho milencem se stal student matematiky na Královské univerzitě.[8]

V březnu roku 1952 byl Alan Turing zatčen a souzen, dle tehdejších zákonů za vztah s mužem. Turing se nijak obvinění nebránil, naopak tvrdil, že na jeho skutcích není nic špatného. Zakládal si na otevřenosti o své sexualitě i v těžké a nepříjemné atmosféře strojírenského Manchesteru. Raději než vězení, podrobil se Turing chemické kastraci.[8]

Turing i nadále pokračoval ve své práci, ale dopady výkonu rozsudku jej po psychické stránce velmi ovlivnily. Zemřel 7. června 1954 ve svém bytě na otravu

kyanidem, s nedojedeným jablkem vedle jeho postele. Jeho matka věřila, že náhodně požil kyanid z prstů po amatérském chemickém pokusu, ale je pravděpodobnější, že důkladně naplánoval svou smrt tak, aby tomu sama mohla uvěřit. Verdikt koronera zněl: sebevražda.[8]

4. Apple, Objective-C a Cocoa Framework

4.1. Mac OS X

Mac OS X je operační systém počítačů Macintosh. První Mac OS X byl vydán 24. března 2001 ve verzi 10.0. Vznikl jako kombinace několika různých technologií. Základ systému se jmenuje Darwin a je složen z hybridního jádra unixového typu XNU spolu s množstvím BSD, GNU a dalších „open source“ nástrojů. Nad jádrem je množina knihoven, služeb a technologií, které jsou ve větší míře přejaty z NeXTSTEPu a předchozího operačního systému Mac OS. Grafické uživatelské rozhraní se jmenuje Aqua a bylo vyvinuto společností Apple.[11]

4.2. Cocoa Framework

Frameworky Cocoa a Cocoa Touch, které jsou motorem Mac OS X a iOS jsou úzce spjaty se zkušenostmi nabytými při vývoji prostředí Xcode. Vysokoúrovňová API Cocoa usnadňují přidávání animací, vytváření sítí a přirozený vzhled aplikací vůči systému spolu s chováním aplikací a přitom zaberou jen několik málo řádků kódu.[14]

Framework Cocoa obsahuje knihovny, aplikační rozhraní (API) a runtime moduly, které utvářejí vývojovou vrstvu pro všechno, co je obsaženo v Mac OS X. Vyvíjením pomocí Cocoa jsou vytvářeny aplikace stejným způsobem, jako je vytvořen samotný Mac OS X. Vytvořené aplikace automaticky dědí chování a vlastnosti Mac OS X, s plným přístupem ke skrytým možnostem operačního systému UNIX. Používání Cocoa spolu s vývojovým prostředím Xcode je ta nejlepší cesta, jak vytvářet nativní aplikace pro Mac.[14]

4.3. Objective-C

V Objective-C je implementováno mnoho věcí z Cocoa. Jde o objektově orientovaný jazyk, který je vytvořen tak, aby běžel co nejrychleji a díky využití dynamického modulu runtime je velmi flexibilní. Vzhledem k tomu, že je Objective-C nadstavbou jazyka C, je snadné začlenit do aplikace v Cocoa i kódy v C, částečně i C++.[14]

Jakmile je aplikace spuštěna, modul runtime Objective-C vytvoří objekty na základě výkonné logiky, nikoliv pouze jak je definováno při kompilaci. Například aplikace běžící v Objective-C může načíst rozhraní (soubor *.xib, vytvořený v Xcode resp. v části „Interface Builder“), připojit objekty Cocoa z rozhraní ke kódu aplikace a poté spustit danou metodu, například po stisku tlačítka v uživatelském rozhraní. Žádná rekompilace již není nutná.[14]

Dynamický modul runtime Objective-C je podobný mnoha moderním skriptovacím jazykům, takže je snadné mapovat vlastnosti Cocoa do jiných jazyků

pomocí Cocoa Bridge. S Cocoa Bridge může vývojář vytvořit provotřídní aplikaci pro Mac OS X za použití AppleScriptu, Ruby a Pythonu.[14]

Cocoa používá návrhové schéma typu Model-View-Controller, a to všude. Model - zapouzdřuje data aplikace, View - zobrazuje a upravuje data z Modelu, Controller - zprostředkovává logiku mezi Modes a View. Rozdělením zodpovědností tímto způsobem je možné vytvořit aplikaci se snažším návrhem, implementací a údržbou.[14]

Schéma MVC znamená, že Interface Builder nevyžaduje psaní kódu nebo jeho generování v okamžiku, kdy se soustředíme na vzhled aplikace. Vazby Cocoa na Mac odstraňují většinu „spojovacího“ kódu. Vytváření spojení mezi Controllers, kódovanými v Xcode a Views, navrhovanými prostřednictvím Interface Builderu je jednoduše záležitostí grafického „spojování“ dvou částí k sobě. Interface Builder spolupracuje s Cocoa pro zjednodušení lokalizace aplikací, což urychluje její rozšíření na různé trhy.[14]

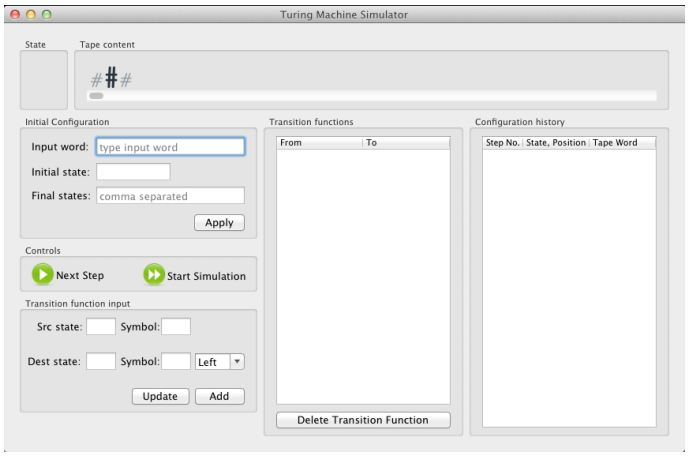
5. Uživatelská dokumentace aplikace TMS

5.1. Instalace aplikace

Instalace aplikace probíhá prostým zkopírováním aplikačního souboru do složky „Applications“. Aplikaci lze spustit i bez kopírování do složky „Applications“ a to z libovolného jiného dostupného umístění.

5.2. Spuštění aplikace

Aplikace se spouští otevřením souboru TMS.app, obecně spustitelný jako aplikace v MAC OS X. Po spuštění je na obrazovce okno aplikace a s aplikací lze pracovat, viz Obrázek 2.



Obrázek 2. Hlavní okno aplikace TMS

5.3. Popis aplikace

Aplikace "Turing Machine Simulator" slouží k demonstraci průběhu výpočtu turingova stroje. Tedy lze pomocí přechodových funkcí, jejichž popis je níže, implementovat libovolný algoritmus.

Pro práci s aplikací je třeba disponovat alespoň základní znalostí problematiky konečných automatů resp. jazyků a gramatik, s nimiž automaty pracují. Bez této znalosti nelze efektivně aplikaci využít.

5.3.1. State

Pole „State“ zobrazuje v aplikaci aktuální stav Turingova stroje. V průběhu výpočtu se hodnota, zobrazená v tomto poli, mění podle aktuální konfigurace Turingova stroje.

5.3.2. Tape content

Tape content je okno, které zobrazuje aktuální stav pásky Turingova stroje. Tato páska je pouze informativní, nelze do ní přímo zapisovat či z ní mazat jednotlivé symboly. Páska znázorňuje aktuální umístění čtecí (zapisovací) hlavy Turingova stroje. Toto umístětí hlavy je reprezentováno zvětšeným fontem čteného symbolu, který je navíc tučný. Symbol „#“ reprezentuje prázdný symbol na pásce, kterých je ve skutečnosti nekonečně mnoho, na pásce aplikace jsou graficky znázorněny pouze prázné symboly, které ohraničují vstupní slovo, popřípadě přepsaný symbol v průběhu výpočtu, pokud je přepisujícím symbolem právě prázdný symbol. Páska Turingova stroje je při inicializaci aplikace reprezentována třemi symboly „#“. Pokud vstupní slovo nebo slovo na pásce v průběhu výpočtu překročí zobrazovanou délku pásky, posouvá se páska pod čtecí (zapisovací) hlavou tak, aby byla hlava vždy viditelná, pokud možno v prostředku zobrazené části slova.

Při ručním zadání nemusí být zadáno žádné slovo, jako zadání je považováno „prázdné slovo“.

5.3.3. Initial configuration

Sekce „Initial configuration“ seskupuje všechny údaje, které je - mimo přechodových funkcí - třeba zadat před samotným spuštěním výpočtu Turingova stroje. Konkrétně se jedná o vstupní slovo, počáteční stav a množinu konečných tzv. přijímajících stavů. Všechny tyto parametry musí být vyplněny, jinak se inicializace Turingova stroje nezdaří. V případě, že některý z povinných parametrů zůstane nevyplněný, je považováno zadání za chybné.

Výjimkou pro zadání slova je znak , který je z abecedy vyloučen z důvodu využití tohoto znaku pro jiný účel.

5.3.4. Controls

V sekci Controls lze ovládat průběh výpočtu Turingova stroje. Lze volit buďto mezi spuštěním celého výpočtu stiskem zeleného tlačítka „Start Simulation“ nebo krokováním pomocí zeleného tlačítka „Next Step“. V případě volby tlačítka „Start Simulation“ se spustí výpočet Turingova stoje, kde je programově nastaven časový interval mezi jednotlivými kroky výpočtu 1s. V případě volby tlačítka „Next Step“ se při každém jeho stisku provede jeden krok výpočtu.

Pro pohyb průběhem výpočtu Turingova stroje, nad rámec možností popsaných tlačítek, lze využít okno „Configuration history“, jak je dále podrobně popsáno v kapitole 5.3.7..

5.3.5. Transition function input

Sekce „Transition function input“ slouží k ručnímu zadání přechodové funkce. Jsou zde všechna pole potřebná pro zadání kompletní přechodové funkce, tj. uspořádané pětice $\boldsymbol{\delta : Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}}$. Po vyplnění všech parametrů se přechodová funkce přidá do seznamu přechodových funkcí stisknutím tlačítka „Add“. Pokud zadáme přechodovou funkci špatně, vrátíme se k její editaci kliknutím kurzorem myši na danou přechodovou funkci v seznamu přechodových funkcí. Tím se nám její obsah vrátí do sekce „Transition function input“ a zde ji můžeme opravit. Opravu následně potvrdíme stiskem tlačítka „Update“, čímž se změna projeví v seznamu přechodových funkcí.

Namísto ručního vyplňování konfigurace Turingova stroje prostředníctvím sekcí „Initial Configuration“ a „Transition Function“ lze použít předdefinované konfigurace, o nichž podrobněji pojednává kapitola 5.4.2..

Výjimkou pro zadání přechodové funkce je znak , který je z abecedy vyloučen z důvodu využití tohoto znaku pro jiný účel.

5.3.6. Transition functions

Tabulka „Transition functions“ obsahuje všechny zadané nebo ze souboru načtené přechodové funkce. V okamžiku, kdy probíhá výpočet Turingova stroje, je aktuálně použitá přechodová funkce zvýrazněna, což značně zpřehledňuje sledování výpočtu Turingova stroje a případně i „ladění“ přechodových funkcí v průběhu výpočtu.

Jak bylo popsáno v kapitole 5.3.5., kliknutím kurzorem myši na konkrétní přechodovou funkci umožníme její editaci. Toto však není možné, pokud je Turingův stroj ve stavu vykonávání výpočtu.

Přechodovou funkci lze i smazat a to vybráním zvolené přechodové funkce kliknutím myši tak, aby byla aktivní a následným stiskem tlačítka „Delete Transition Function“.

5.3.7. Configuration history

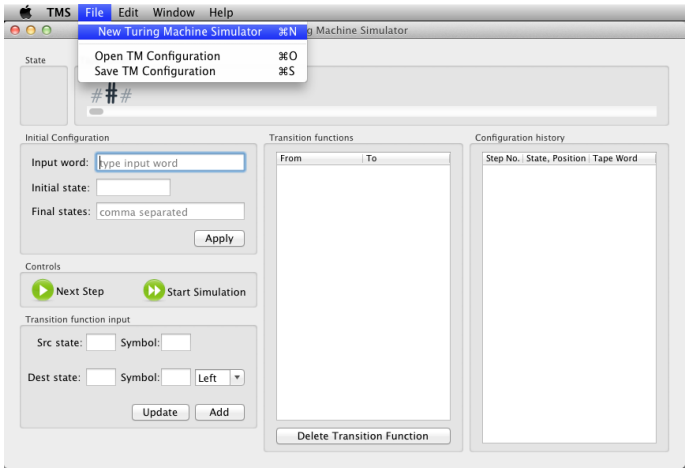
Tabulka „Configurations History“ obsahuje všechny konfigurace Turingova stroje, kterými doposud Turingův stroj prošel během aktuálního výpočtu. Kliknutím do tabulky konfigurací lze Turingův stroj zastavit a klikáním na různé konfigurace se lze libovolně pohybovat mezi všemi dosud existujícími konfiguracemi. Stiskem tlačítka „Play“ lze nechat Turingův stroj pokračovat ve výpočtu od naposledy zvolené (zvýrazněné) konfigurace.

5.4. Menu

Menu aplikace obsahuje několik položek, které mohou uživateli pomoci při práci s aplikací.

5.4.1. File - New Turing Machine Simulator

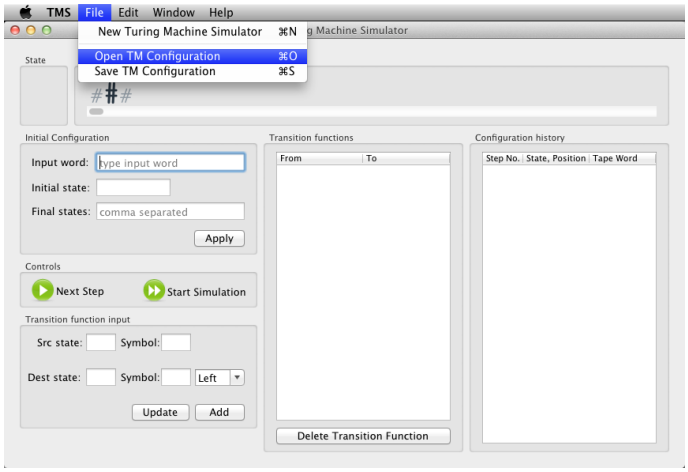
Volba „New Turing Machine Simulator“ umožní uživateli vytvořit novou konfiguraci Turingova stroje. Pokud je volba provedena, předchozí konfigurace je ztracena, viz Obrázek 3..



Obrázek 3. Menu - File - New TMS

5.4.2. File - Open Turing Machine Configuration

Volba „Turing Machine Configuration“ nabídne uživateli zvolit k načtení již dříve uloženou nebo vytvořenou koniguraci Turingova stroje, která je ve formě textového souboru, s příponou „tmc“. Příklady nejznámějších konfigurací jsou součástí složky se souborem aplikace, viz Obrázek 4..



Obrázek 4. Menu - File - Open TMS

V případě, že uživatel otevírá konfiguraci, kterou předtím neuložil z již existující konfigurace v aplikaci, ale vytvořil vlastní konfigurační soubor, je nutné dodržet předepsanou strukturu souboru, kdy jako šablonu lze využít již existující předpřipravené soubory, např. „empty-structure.tmc“. Tento soubor je potřeba upravit podle potřeby, tak jak je vytvořen, není aplikací akceptovatelný.

Pro sekci **„Machine“** jsou povinné parametry „tapeContent“ a „initialState“.

Pro sekci **„Transitions“** jsou povinné parametry „actualTapeSymbol“, „srcState“, „newTapeSymbol“ a „destState“.

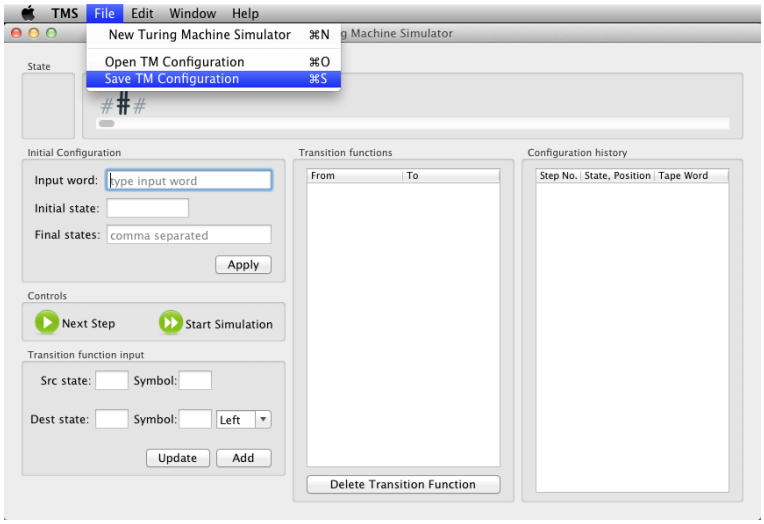
Pro sekci **„Configurations“** jsou povinné všechny parametry.

Struktura konfiguračního osuboru:

```
{
  "Machine": {
    "actualState": "",
    "finalStates": "",
    "tapeContent": "",
    "initialState": "",
    "headPosition": ""
  },
  "Transitions": [
    {
      "headMoveType": "",
      "actualTapeSymbol": "",
      "srcState": "",
      "newTapeSymbol": "",
      "destState": ""
    },
    {...
    }
  ],
  "Configurations": [
    {
      "tapeContent": "",
      "actualState": "",
      "headPosition": ""
    }
  ]
}
```

5.4.3. File - Save Turing Machine Configuration

Volba „Save Turing Machine Configuration“ umožňuje uživateli uložit aktuální konfiguraci Turingova stroje. Stejně jako u načtení konfigurace, i uložení je ve formě textového souboru s příponou „tmc“, kdy může uživatel zvolit pro uložení libovolné umístění, viz. Obrázek 5.. Po načtení konfiguračního souboru (soubor se načte pouze pokud obsahuje u povinných parametrů správně vyplněné hodnoty), se zobrazí aplikace, resp. simulátor Turingova stroje v dané konfiguraci a lze s Tutingovým strojem ním pracovat libovolně dle popisu v předchozích podkapitolách. Lze tedy načítat i takové konfigurace, které byly již dříve uloženy i v průběhu výpočtu Turingova stroje.



Obrázek 5. Menu - File - Save TMS

5.5. Ukončení aplikace

Ukončení aplikace lze provést dvěma způsoby:

- **Z menu** volbou „Turing Machine Simulator - Quit Turing Machine Simulator“,
- **z aplikace** zavřením jejího okna (červeně podbarvený symbol „+“ v levém korním rohu okna aplikace).

5.6. Odinstalace aplikace

Odinstalace se provádí smazáním souboru „TMS.app“ ze složky „Applications“.

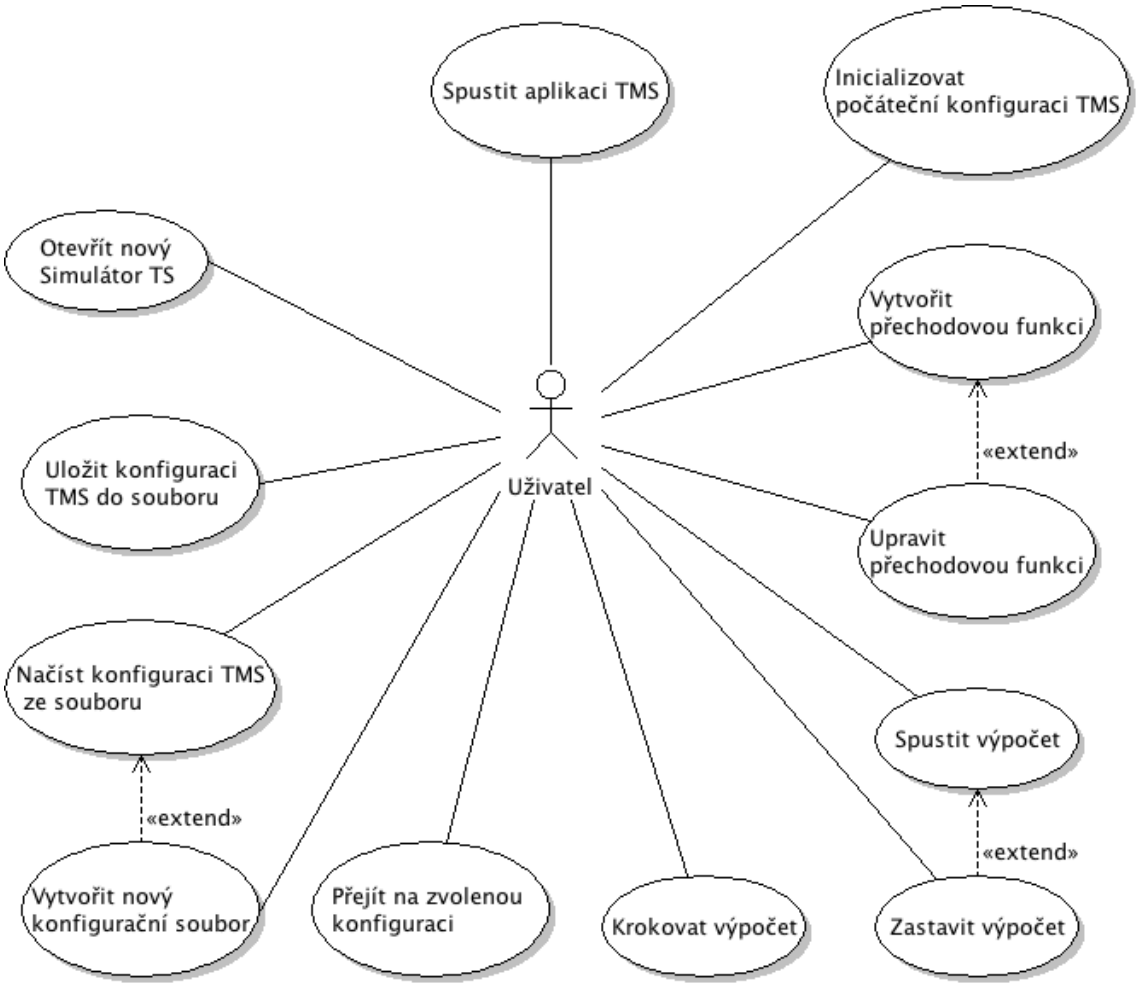
6. Programátorská dokumentace aplikace TMS

6.1. Vývojové prostředí, použité nástroje

Aplikace „Turing Machine Simulator“ byla vytvořena ve vývojovém prostředí Xcode 4.0, běžícím na operačním systému MAC OS X 10.6 (Snow Leopard) a Xcode 4.1, běžícím na operačním systému MAC OS X 10.7 (Lion), s použitím programovacího jazyka Objective-C a taktéž s využitím Cocoa Frameworku.

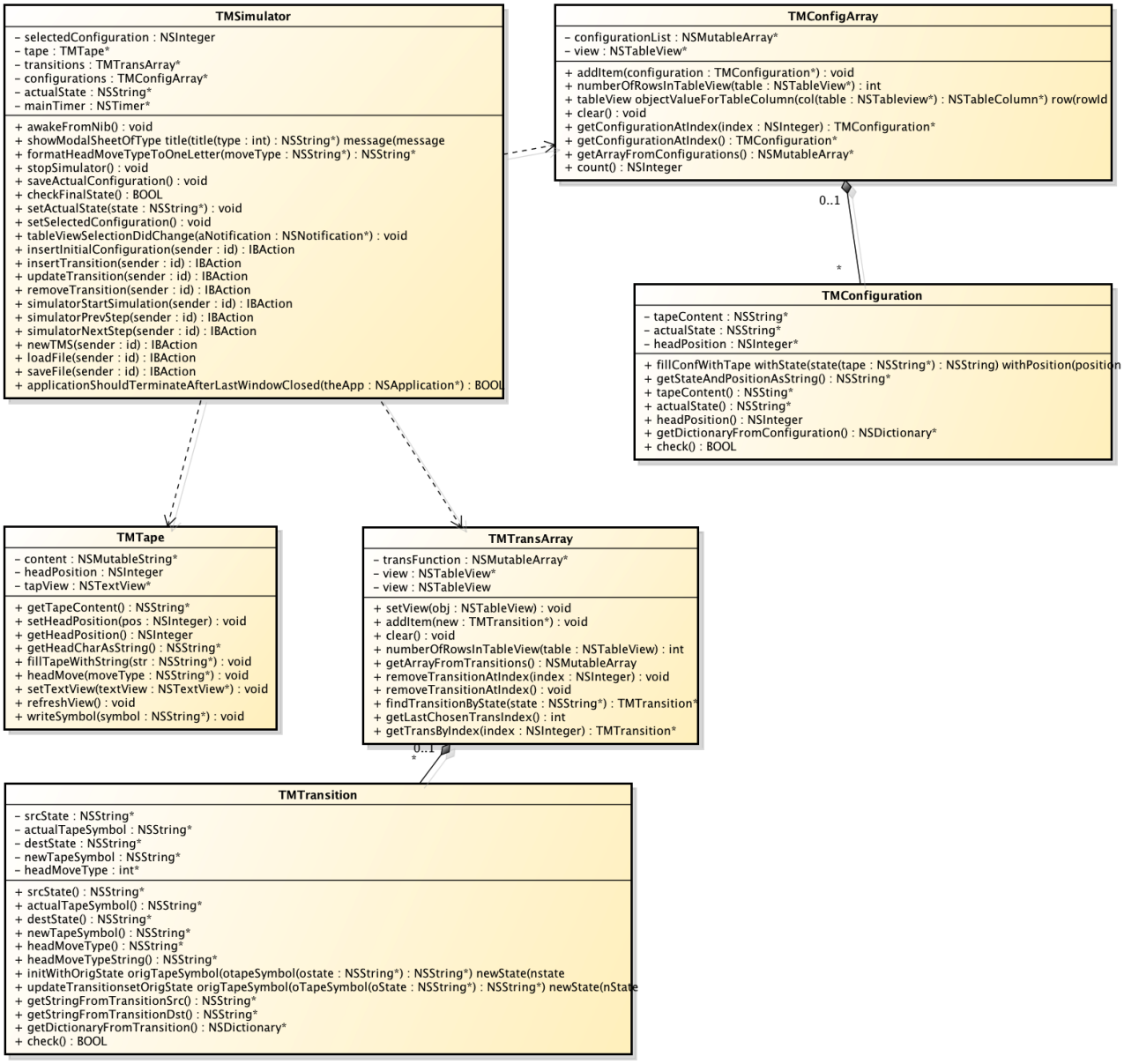
V aplikaci je pro funkce nahrávání dat ze souboru a ukládání dat do souboru použitý framework SBJSON. JSON je jednoduchý formát pro výměnu dat, který je snadno čitelný pro člověka i pro počítač. Pro počítače je snadné takové soubory analyzovat i vytvářet. Jeho základy tvoří programovací jazyk JavaScript. JSON je textový formát, zcela nezávislý na programovacím jazyce, ale používá konvence, známé programátorům velkého množství nejrozšířenějších programovacích jazyků. Tyto vlastnosti činí JSON ideálním jazykem pro výměnu dat.

6.2. Use Case Diagram



Obrázek 6. Use Case Diagram TMS

6.3. Diagram tříd



Obrázek 7. Class Diagram TMS

6.4. Třídy a jejich metody, funkce

6.4.1. Třída "TMSimulator"

- `awakeFromNib() : void`
  - nastavení počátečních hodnot všech polí, inicializace hodnot, nastavení scrolleru, nastavení delegátů
- `showModalSheetOfType(type : int) title(title : NSString*) message(message : NSString*) : void`
  - zobrazení modálního okna ve dvou režimech - alert a information
- `formatHeadMoveTypeToOneLetter(moveType : NSString*) : NSString*`
  - převod popisu pro pohyb hlavy (Left/L, Right/R, None/N)
- `stopSimulator() : void`
  - zastavení timeru pro průběh přechodové funkce
- `saveActualConfiguration() : void`
  - uložení aktuální konfugurace: aktuální stav, obsah pásky, tabulku přechodových funkcí a tabulku historie konfigurací
- `checkFinalState() : BOOL`
  - kontrola, zda je zadán koncový stav
- `setActualState(state : NSString*) : void`
  - nastavuje textové pole stavu a vnitřní proměnnou
- `setSelectedConfiguration() : void`
  - přesun Turingova stroje do zvolené konfigurace z tabulky historie konfigurací
- `tableViewSelectionDidChange(aNotification : NSNotification*) : void`
  - změna výběru v tabulce, pomocí NSNotification se zjišťuje, o jakou tabulku se při daném výběru jedná
- `insertInitialConfiguration(sender : id) : IBAction`
  - vložení počáteční konfigurace - dáno vstupem uživatele resp. konfiguračním souborem
- `insertTransition(sender : id) : IBAction`
  - vložení přechodové funkce do tabulky přechodů z uživatelského vstupu, tlačítko „Add“

- `updateTransition(sender : id) : IBAction`
  - oprava zvolené přechodové funkce z uživatelského vstupu, tlačítko „Update“
- `removeTransition(sender : id) : IBAction`
  - odstranění zvolené přechodové funkce z uživatelského vstupu, tačítko „Delete Transition Function“
- `simulatorStartSimulation(sender : id) : IBAction`
  - spuštění simulace výpočtu s intervalem 0,5s mezi jednotlivými kroky výpočtu, spouští timer
- `simulatorNextStep(sender : id) : IBAction`
  - krokování výpočtu, z uživatelského vstupu, tlačítko „Next Step“
- `newTMS(sender : id) : IBAction`
  - vymaže všechny hodnoty z GUI, TMS je „prázdný“
- `loadFile(sender : id) : IBAction`
  - načte soubor typu TMC, který obsahuje kompletni konfiguraci Turingova stroje
- `saveFile(sender : id) : IBAction`
  - uloží soubor typu TMC s definovanou strukturou
- `applicationShouldTerminateAfterLastWindowClosed(theApp : NSApplication*) : BOOL`
  - ukončení aplikace zavřením okna

6.4.2. Třída “TMTape”

- `getTapeContent() : NSString*`
  - získání aktuálního obsahu pásky
- `setHeadPosition(pos : NSInteger) : void`
  - nastaví čtecí/zapisovací hlavu na požadovanou pozici
- `getHeadPosition() : NSInteger`
  - získání aktuální pozice čtecí/zapisovací hlavy
- `getHeadCharAsString() : NSString*`
  - získání znaku na pásce jako řetězce
- `fillTapeWithString(str : NSString*) : void`
  - naplní obsah objektu TMTape řetězcem, volá se po kliknutí na tlačítko Apply

- `headMove(moveType : NSString*) : void`
  - pohyb čtecí/zapisovací hlavy po pásce
- `setTextView(textView : NSTextView*) : void`
  - nastaví interní ukazatel na textové pole, naplňuje se z hlavního programu, pomocí ukazatele je možné měnit pole
- `refreshView() : void`
  - nastavení všech znaků na pásce na definovaný font, znak pod čtecí/zapisovací hlavou na větší font
- `writeSymbol(symbol : NSString*) : void`
  - nahradí symbol pod hlavou řetězcem v parametru

6.4.3. Třída "TMTransArray"

- `addItem(new : TMTransition*) : void`
  - přidání přechodu do pole přechodů
- `clear() : void`
  - smaže všechny přechodové funkce, využití při volbě nového TMS
- `numberOfRowsInTableView(table : NSTableView) : int`
  - vrací počet řádků tabulky - funkce rozhraní pro tabulku
- `tableView(table : NSTableView*) objectValueForTableColumn (col : NSTableColumn*) row:(rowid id) : id`
  - funkce daného rozhraní, která vrací objekt na dané místo tabulky
- `getArrayFromTransitions() : NSMutableArray*`
  - vytvoří pole slovníků - pro uložení do souboru
- `removeTransitionAtIndex(index : NSInteger) : void`
  - smaže aktuálně vybranou přechodovou funkci
- `findTransitionByState(state : NSString*) : TMTransition*`
  - prohledá přechodovou funkci a vrátí první přechod, který odpovídá stavem a symbolem, nebo vrací nil, pokud nenalezne
- `getLastChosenTransIndex() : int`
  - vrátí index naposledy vybraného přechodu z přechodové funkce, používá se k obarvení řádku v tabulce
- `getTransByIndex(index : NSInteger) : TMTransition*`
  - získá přechodovou funkci na základě daného indexu

6.4.4. Třída "TMTransition"

- `initWithOrigState(ostate : NSString*) origTapeSymbol(oTapeSymbol : NSString*) newState(nstate : NSString*) newTapeSymbol(nTapeSymbol : NSString*) transition(transition : NSString*) : void`
  - přiřadí hodnoty dané přechodové funkce do proměnných
- `updateTransitionsetOrigState(oState : NSString*) origTapeSymbol(oTapeSymbol : NSString*) newState(nState : NSString*) newTapeSymbol(nTapeSymbol : NSString*) transition(transition : NSString*)`
  - přiřadí hodnoty dané přechodové funkce do proměnných v případě aktualizace přechodové funkce
- `getStringFromTransitionSrc() : NSString*`
  - získá levou stranu přechodové funkce
- `getStringFromTransitionDst() : NSString*`
  - získá pravou stranu přechodové funkce
- `getDictionaryFromTransition() : NSDictionary*`
  - vrácení přechodu jako slovníku - použito pro ukládání do souboru
- `check() : BOOL`
  - ověří správnost všech dat

6.4.5. Třída "TMConfiguration"

- `fillConfWithTape(tape : NSString*) withState(state : NSString) withPosition(position : NSInteger) : void`
  - přiřadí hodnoty dané konfigurace do proměnných
- `check() : BOOL`
  - ověří správnost všech dat

6.5. Třída "TMConfigArray"

- `addItem(configuration : TMConfiguration*) : void`
  - přidání konfigurace do pole konfigurací
- `numberOfRowsInTableView(table : NSTableView*) : int`
  - vrací počet řádků konfigurací

- `tableView(table : NSTableView*) objectValueForTableColumn(col : NSTableColumn*) row(rowId : id) : id`
  -tabulka historie konfigurací
- `clear() : void`
  - smaže všechny konfigurace, využití při volbě nového TMS
- `getArrayFromConfigurations() : NSMutableArray`
  - vytvoří z konfigurací pole konfigurací
- `getConfigurationAtIndex(index : NSInteger) : TMConfiguration*`
  - získání aktuálně zvolené konfigurace

Závěr

Aplikace, kterou tato práce popisuje, může sloužit nejen k demonstraci výpočtu Turingova stroje, ale i k seznámení se s programovacím jazykem Objective-C pro ty, kteří zvažují v tomto jazyce vytvářet své školní i budoucí programátorské počiny.

Jedná se o první verzi programu, která by mohla v příštích verzích doznat následujících vylepšení:

- Lokalizace programu do jiných jazyků (především do Češtiny)
- Přidat možnost volit hodnotu časovače pro interval mezi jednotlivými kroky výpočtu
- Vytvořit „Help“ aplikace

Conclusions

This text describes an application, which can be used not only to demonstrate Turing machine computation, but also can be used to introduction to Objective-C programming language for those who consider use this programming language to create their school or future applications programming.

This is the first version that could be varied in future versions of the following improvements:

- Localization of the application into other languages (priparilly in Czech),
- Add option to choose a value for the timer interval between each step of computation,
- Create an Application „Help“.

Reference

[1] Martínek, Pavel. *Základy teoretické informatiky*
Univerzita Palackého, Olomouc, 2006

[2] Rich, Elaine. *Automata, Computability and Complexity*
Pearson Prentice Hall, Upper Saddle River, 2008.

[3] Copeland, B. Jack. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*
Clarendon Press (Oxford University Press), Oxford UK, 2004, ISBN 0-19-825079-7

[4] Hopcroft J. E., Motwani R., Ullman J. *Introduction to Automata Theory, Languages, and Computation*
Addison-Wesley, Boston, 2006. ISBN 978-0321462251

[5] Turing, Alan. *Intelligent Machinery*
esej, 1948

[6] Turing, Alan. *ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM*
článek, 1936

[7] Vitanyi, Paul. *Turing Machine*
Scholarpedia, 2009. ISSN 1941-6016

[8] Hodges, Andrew. *Alan Turing: the enigma*
Vintage, Random House, London, 1992. ISBN 0-09-911641-3

[9] Hopcroft J., Ullmann J.

[10] Elghot,Robinson, Hartmanis, Cook-Rechow

[11] Singht, Amit. *Architecture of Mac OS X*
http://osxbook.com/book/bonus/ancient/whatismacosx//arch.html

[12] Radó, Tibor (1962). *On non-computable functions*
Bell System Technical Journal, Vol. 41, No. 3 (May 1962), pp. 877–884

[13] van Emde Boas, Peter/ *Machine Models and Simulations*
pp. 3–66, in Jan van Leeuwen, ed., Handbook of Theoretical Computer Science, 1990

[14] *The official Apple developers site*
http://developer.apple.com

[15] *The Free Worldwide Internet Encyclopedia*
http://www.wikipedia.org

A. Obsah přiloženého CD

`bin/`
: Aplikace TMS.APP spustitelná přímo z CD/DVD.

`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 TMS / MAC OS X aplikace TMS, se všemi potřebnými (převzatými) zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu / (v ZIP archivu).

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

Navíc CD/DVD obsahuje:

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

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