SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Softvérové inžinierstvo, Informačné systémy

ISI TP Cup - Priebežná správa

Tím číslo 2 - Automaty

Vedenie projektu: Mgr. Daniela Chudá, PhD.

Bc. Robin Bábíček
Bc. Matúš Coranič
Bc. Matúš Čelko
Bc. Celestín Černák
Bc. Daniela Miloňová
Bc. Katarína Poláková

26.2.2009

Úvod

Tento dokument obsahuje priebežnú správu o riešení tímového projektu na účely súťaže TP cup. V stručnej forme v ňom hodnotíme doterajšiu prácu na projekte, dosiahnuté výsledky a budúce smerovanie projektu.

Náplň projektu a ciele

V prvej fáze riešenia tímového projektu sme sa venovali prehľadu existujúcich riešení, na základe ktorých sme boli schopní definovať požiadavky na výsledný produkt. Preskúmali sme 12 rôznych simulátorov, jednalo sa buď o študentské projekty alebo iné voľne dostupné simulátory. Po zhodnotení všetkých vlastností sme vytvorili porovnávaciu tabuľku, na základe ktorej sme definovali najdôležitejšie vlastnosti, na ktoré sa pri vytváranonm simulátore chceme zamerať. Keďže simulátor má slúžiť ako učebná pomôcka, dôraz je kladený na čo najväčšiu názornosť, prehľadnosť a jednoduché používanie s minimálnou potrebou učenia sa. Z pohľadu používateľa je systém rozdelený na dva ucelené nástroje, editor a simulátor.

Ako hlavnú pridanú hodnotu nášho simulátora v porovnaní s existujúcimi riešeniami sme zvolili nasledovné aspekty:

- jednoduché a používateľsky príjemné rozhranie
- možnosť navrhnúť si vlastný automat
- rozšírenie o prácu s operáciami nad výpočtovými zariadeniami
- prehľadné zobrazenie nedeterministického správania
- možnosť testovania študentov, generovanie problémových situácii nad zariadením aj s kontrolným riešením

Prehľad riešenia

Na základe týchto požiadaviek sme vytvorili návrh systému. Vytváraná aplikácia má byť aplikáciou modulárnou a ľahko rozšíriteľnou. Z tohto dôvodu sme ju rozdelili na 2 samostatné celky: moduly používateľského rozhrania a na výpočtovú logiku. Moduly používateľského rozhrania predstavujú tie časti aplikácie, ktoré prídu do priameho kontaktu s používateľom. Poskytnú mu rozhranie na editáciu a následnú simuláciu automatu. Tieto moduly neobsahujú žiadnu výpočtovú logiku.

Za výpočtovú logiku je zodpovedný samostatný modul, ktorého funkcionalitu využívajú jednotlivé moduly používateľského rozhrania. Tento modul má na starosti všetky operácie súvisiace so simuláciou automatu. Dostatočne všeobecné výpočtové jadro umožňuje jednotlivé moduly používateľského rozhrania podľa potreby jednoducho obmieňať.

Z pomerne široko špecifikovaného zadania sa naša doterajšia práca zameriavala na konečný automat, zásobníkový automat a Turingov stroj, samozrejme s možnosťou dodefinovať si aj vlastný

typ automatu. Ostatným typom automatov (napr. RAM stroj a výpočtový stroj) sme sa doteraz nevenovali, keďže ich výpočtové jadro bude odlišné od súčasného.

Ako skúška správnosti návrhu nám poslúžil vytvorený prototyp. Jeho hlavným cieľom bolo dokázať, že návrh riešenia predstavuje správny postup, ktorý povedie k želaným výsledkom. Zároveň nám umožnil odhaliť prípadné problémy, s ktorými sa počas implementácie môžme stretnúť.

Keďže ako techniku vývoja aplikácie sme zvolili agilné programovanie, prototyp má pre našu nasledujúcu prácu ďalší prínos. Je ním možnosť postupne pridávať funkcionalitu a v prípade problémov nebude predstavovať problém, ak sa odkloníme od správneho riešenia.

Pri implementácii prototypu sme sa teda zamerali na aspekty, ktoré nám určia ďalšie smerovanie práce a v prípade problémov by mohli spôsobiť zmenu použitých technológií a prístupov. Konkrétne sa jednalo o vizualizáciu simulátora a celkové grafické používateľské rozhranie. Nasledujúcim krokom bolo overenie použitých knižníc pre načítavanie XML súboru, generovania tried a vizualizácie jednotlivých stavov automatu. Ďalším dôležitým bodom, od ktorého závisí ďalšie smerovanie práce, je zvolený algoritmus prehľadávania stromu. Tieto aspekty boli dôkladne otestované, výsledky sú zhrnuté v nasledujúcej kapitole.

Keďže cieľom vytvorenia prototypu nebolo implementovať všetky časti zadania, implementovali sme iba kľúčové aspekty návrhu. Ostatné časti budú dopracované v ďalších etapách práce na projekte.

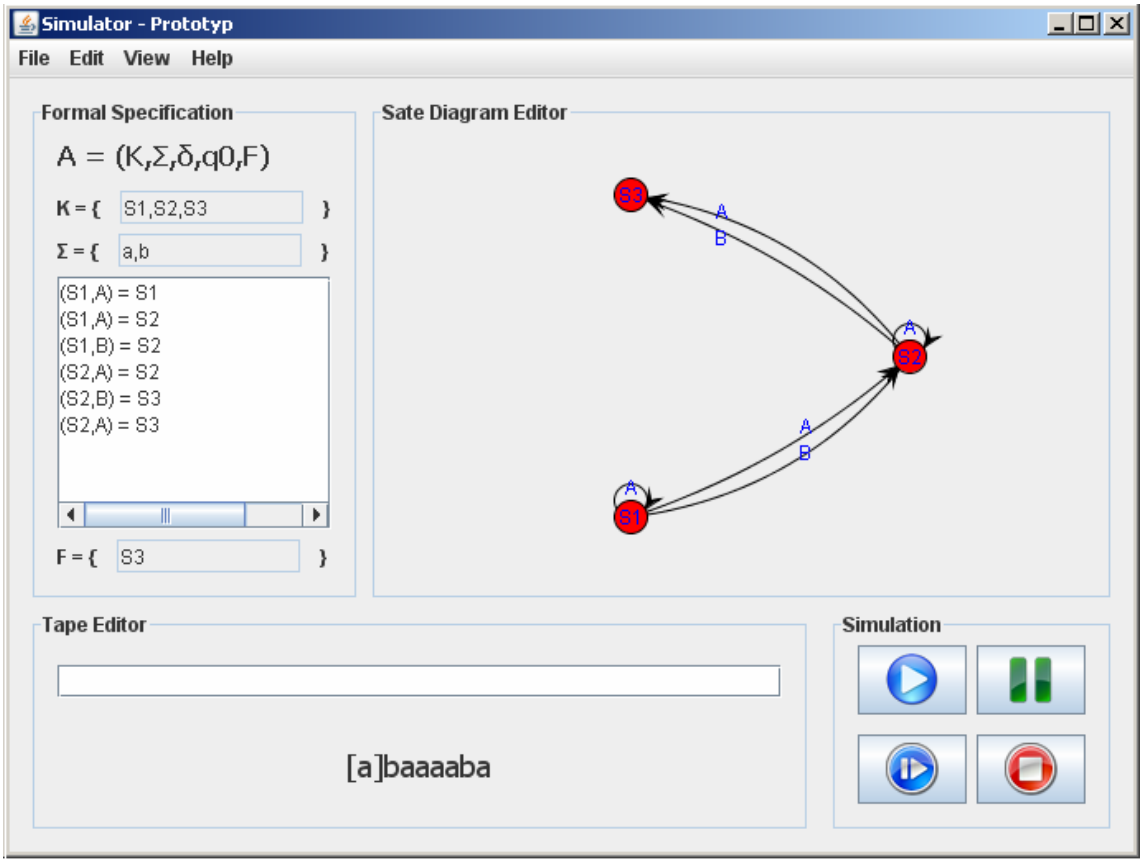
Dosiahnuté výsledky

Na overenie dôležitých aspektov prototypu sa ukázalo ako najefektívnejšie implementovať simulátor konečného automatu, ktorý je schopný načítať vstupný XML súbor, vizualizovať jednotlivé prvky simulácie, a čo je najdôležitejšie, túto simuláciu vykonať. Keďže jedným z hlavných prínosov finálnej práce bude prehľadné zobrazenie nedeterminizmu, tomuto aspektu sme sa venovali aj vo vytvorenom prototype. Podarilo sa nám implementovať a overiť dva spôsoby zobrazenia nedeterminizmu. V prvom je voľba nasledujúceho kroku ponechaná na používateľovi, pri druhom spôsobe simulátor sám vyhľadá prvú cestu, ktorá povedie k správnemu výsledku a vypíše ju používateľovi. Samotné vyhľadávanie je realizované pomocou prehľadávania stromu do šírky. Konkrétne dosiahnuté výsledky testovania sme rozdelili podľa aspektov, ktorých sa týkajú. Bližšie sú predstavené v nasledujúcich podkapitolách.

Grafické používateľské rozhranie

Prvá oblasť predstavuje komunikáciu aplikácie s používateľom. V tejto oblasti sme si v rámci prototypu vyskúšali, či nami špecifikované metódy prezentácie automatu, jeho jednotlivých častí, ako aj metódy zobrazenia simulácie a nedeterminizmu, sú pre používateľa dostatočne intuitívne. Práca s prototypom je intuitívna a jednoduchá. Obmedzená funkcionalita prototypu ale nedovoľuje overiť všetky zvolené princípy. Tie, ktoré bolo možné overiť, sa ukázali ako správne a pre používateľa ľahko

pochopiteľné. Princípy komunikácie s používateľom použité v prototype budú použité aj pri finálnej implementácii simulátora. Na obr. č. 1 je zobrazené okno simulátora, ktoré s menšími úpravami použijeme aj vo finálnej verzii simulátora.



Obr 1. Okno prototypu simulátora

Technická stránka

Druhá oblasť výsledkov predstavuje technickú stránku prototypu. Za jeho pomoci sme boli schopní overiť, či na zvolenej platforme a za použitia zvolených technológií je možné zostrojiť fungujúci simulátor. Platforma Java sa ukázala pre implementáciu ako vhodná. Návrh za pomoci XML schém sa ukázal ako efektívny spôsob definície vlastností simulátora. S použitím XML schém úzko súvisí použitie knižnice JAXB v aplikácií. JAXB umožňuje generovanie Java tried z XML schém a následne poskytuje funkcionalitu na deserializácie XML súborov do generovaných tried. Týmto spôsobom je možné definíciu automatu ukladať do XML súborov, ktoré sú relatívne dobre čitateľné aj za pomoci obyčajného textového editora. V prototype je použitá aj knižnica springframework. Springframework, mimo iných funkcií, umožňuje vytváranie inštancií tried s vlastnosťami nastavenými podľa preddefinovaných XML šablón. Prvotná idea použitia springframeworku bolo pri vytváraní konfigurácií jadra, kde by sa inštancia triedy, ktorá definuje vlastnosti simulátora, vytvárala za pomoci springframeworku. Od tejto metódy sa počas implementácie ustúpilo a bola použitá metóda za použitia XML schémy a generovaných tried, pričom parametre simulátora sú uložené v XML

šablónach, ktoré sú deserializované za využitia JAXB. Funkčnosť tohto riešenia bola overená na schéme konečného automatu, ktorý je prototyp schopný odsimulovať. Používateľské rozhranie bolo vytvorené za pomoci knižnice JUNG. Aj keď sa počas implementácie objavili problémy týkajúce sa nedostatočnej a chybnej dokumentácie tejto knižnice, jej použitie sa ukázalo ako prínos. Pomocou nej sú v prototype vizualizované jednotlivé stavy automatu.

Celkovo sa voľby platforiem a technológií ukázali ako správne a poskytujú dobrý predpoklad, že sa za ich pomoci podarí implementovať simulátor v plnom rozsahu.

Algoritmus prehľadávania stromu

Tretiu oblasť výsledkov predstavuje funkčnosť zvolených algoritmov. Prototyp reprezentuje postupnosť stavov ako stromovú štruktúru, pričom pri hľadaní úspešného riešenie prehľadáva strom do šírky. Zvolený algoritmus sa ukázal ako vhodný a bude použitý aj pri konečnej implementácii simulátora. Simulátor je rovnako schopný identifikovať jednotlivé vetvy stromu a nezávisle ich pri simulácií rozvetvovať. Táto možnosť nie je plne využitá v prototype, ale pri finálnej implementácií sa vďaka nej budú dať používateľovi poskytnúť širšie možnosti krokovania nedeterministických automatov, a to tým spôsobom, že umožní užívateľovi dynamicky prepínať jednotlivé vetvy stromu.

Zhrnutie stavu a plán ďalšej práce

Ako celok je vytvorený prototyp schopný načítať pripravený súbor obsahujúci definíciu konečného automatu. Tento automat je schopný graficky vizualizovať. Následne je používateľ schopný odsimulovať automat, a to jednak spôsobom krokovania, ako aj rýchlou simuláciou automatu, kedy simulátor používateľa priamo informuje, či automat vstup akceptoval. Riešene nedeterminizmu je závislé od zvolenej metódy simulácie. Pri krokovaní užívateľ určí, do ktorého stavu sa automat prepne. Pri rýchlej simulácií simulátor sám zvolí tú cestu, ktorá vedie k akceptovaniu vstupu automatom.

Celkovo je implementovaný prototyp funkčný, ukazuje správnosť zvolených prístupov, technológií a algoritmov a predstavuje základný kameň pred implementáciou plnohodnotného simulátora.

Po implementácii prototypu simulátora automatov sme sa začali venovať analýze požiadaviek a súvisiacich riešení simulátorov RAM a počítadlových strojov. Preskúmali sme niekoľko existujúcich simulátorov a definovali sme predbežné požiadavky. Keďže ich výpočtová logika je odlišná od simulátora bežných automatov, nasledujúcim krokom bolo definovanie návrhu ich simulácie. Na tejto etape stále pracujeme, plánujeme ju dokončiť v 3.-4. týždni semestra. Zároveň súbežne optimalizujeme vytvorený prototyp na stavové automaty a implementujeme plnohodnotný simulátor.