Posudek oponenta bakalářské práce

Student: Polanský Ondřej
Téma: Srovnání efektivity různých programovacích jazyků při práci s automaty (id 22909)
Oponent: Lengál Ondřej, Ing., Ph.D., UITS FIT VUT

1. Náročnost zadání **obtížnější zadání**

Student měl v práci implementovat sadu algoritmů pro práci s konečnými automaty ve 4 programovacích jazycích, C++, C#, OCaml a Python, a srovnat jejich výkon a pracnost použití. Zadání hodnotím jako mírně obtížnější, jelikož se student musel naučit několik nových programovacích jazyků a taky nastudovat algoritmy z vědecké literatury.

2. Splnění požadavků zadání **zadání splněno**

Zadání bylo splněno.

3. Rozsah technické zprávy **je v obvyklém rozmezí**

Rozsah je v obvyklém rozmezí.

4. Prezentační úroveň předložené práce **75 b. (C)**

Práce má logickou strukturu, kapitoly na sebe navazují.

Asi nejdůležitější je Kapitola 4, která obsahuje naměřené výsledky a jejich diskuzi. V této části je ještě hodně prostoru na vylepšení, hlavně s ohledem na systematičnost. Autor například uvádí, že výsledky algoritmů pro stejnou operaci v různých jazycích porovnával "namátkově", což považuji za nedostatečné. Proč neporovnávat všechny výsledky, když je stejně máme?

Dále nechápu autorovu poznámku v sekci 4.2.1 o testování prázdnosti jazyka, že "některé jazyky jsou lepší pro menší automaty, jiné naopak". V této sekci by bylo lepší experimenty rozdělit na minimálně dvě části: automaty s prázdným jazykem (kde algoritmus musí projít všechny stavy) a automaty s neprázdným jazykem, kde algoritmus může skončit dříve. Dále nechápu poznámku o tom, že "OCaml příliš šetří pamětí na úkor rychlosti", když OCaml spotřeboval 10 MiB a C++ 13 MiB. Z takto malého rozdílu je těžké něco odvozovat (tipuji, že je naměřená pamět kterou alokovalo jádru procesu, což je daleko od skutečně používané paměti).

Proč jsou v tabulkách s časy uváděny hodnoty s přesností na až 10 číslic? Raději bych v tabulkách viděl směrodatnou odchylku nebo jiné indikátory. Místo tabulek by se mohly použít boxploty, ze kterých by se daly zjistit třeba kvartily.

Moc mi není jasná zmiňovaná problematičnost minimalizace deterministických automatů. Zatímco asymptotická složitost determinizace je exponenciální a minimalizace pouze O(n log n), ve výsledcích to vypadá přesně naopak. Kde je problém? Očekával bych nějakou diskuzi.

Dále jsou dost zarážející výsledky pro testování univerzality (4.2.7). Když je porovnám s výsledky pro testování prázdnosti jazyka automatu, jsou všechny skoro stejné, a to malé. Plus mínus platí, že maximální čas z testování univerzality je stejný jako minimální čas z testování prázdnosti, což z pohledu výpočetní složitosti těchto problémů (PSPACE-úplný s exponenciálním algoritmem pro univerzalitu a lineární pro prázdnost) nedává smysl. Buď je někde v algoritmu chyba, nebo je testovací sada automatu velmi nevhodně zvolená.

Některé výroky autora jsou v technickém textu zarážející, např. "Python splnil svůj pomalý standard" (4.2.7), případně výrok "Autorův názor je takový, že neexistuje žádné pozitivum dynamického typování" ukazuje studentův omezený rozhled. Věta "Dá se říci, že C++ má velmi Erbenovský charakter" sice rozesměje, ale v technickém textu nemá co dělat.

5. Formální úprava technické zprávy **78 b. (C)**

Práce je psána česky. Formální úprava práce je víceméně v pořádku, občas se v práci vyskytne nějaký překlep. Horší to je u matematických zápisů. Student například soustavně používá množinové závorky namísto kulatých závorek pro n-tici při definici složek konečného automatu.

6. Práce s literaturou **90 b. (A)**

V pořádku.

7. Realizační výstup **80 b. (B)**

Je dodáno několik zdrojových souborů s implementací automatů a operací nad nimi, vetšinou jeden soubor pro každý jazyk (implementace pro C# je dodána jak ve verzi, kde je vše v jednom souboru, tak ve verzi, kde jsou třídy v několika souborech.

Implementace algoritmů je víceméně přímočará bez snah o optimalizace. Příkladem může být použití řetězců pro reprezentaci stavu a symbolu => jakýkoliv algoritmus pracující nad automatem pak musí při práci se stavy a symboly provádět drahé porovnávání řetězců namísto jednoduchého porovnání dvou integerů pomocí jedné instrukce.

Zdrojové kódy jsou psány relativně slušně, komentáře jsou na vhodných místech. Je dodána programová dokumentace.

Není mi jasné, proč sekvenční implementace algoritmů v Pythonu používá frontu queue, která je určena pro synchronizaci ve vícevláknových programech; obyčejný list by úplně stačil a nejspíš byl mnohem rychlejší (otázka, jak by se toto promítnulo do výsledků).

Nepodařilo se mi v odevzdaných souborech najít oněch 200 automatů, na nichž byly experimenty provedeny. Adresář *Prelozene_programy/automaty* obsahuje jen několik málo automatů, které vypadají spíš jako vstupy pro testy než benchmarky.

8. Využitelnost výsledků

Výsledkem je srovnání čtyřech programovacích jazyků z hlediska rychlosti, efektivity práce s pamětí a uživatelské přívětivosti. Hrubé výsledky jsou víceméně očekávané, přesnější informace nemá smysl vzhledem k naivní neoptimalizované implementaci brát příliš v potaz.

9. Otázky k obhajobě

1. Máte představu, proč je minimalizace (asymptotická složitost O(n log n)) o tolik pomalejší, než determinizace (exponenciální složitost) a není tomu naopak?
2. Co je příčinou toho, že experimenty pro univerzalitu skončily tak rychle?

10. Souhrnné hodnocení **75 b. dobře (C)**

Ondřej Polanský ve své bakalářské práci zvládl naimplementovat algoritmy k práci s konečnými automaty ve čtyřech rozdílných programovacích jazycích a srovnat jejich výkonnost. Celkem hezkou práci kazí nedostatečné zpracování výsledků, zmíněno detailně výše. Celkově práci hodnotím stupňem **C (dobře)**.

Prohlášení: Uděluji VUT v Brně souhlas ke zveřejnění tohoto posudku v listinné i elektronické formě.

V Brně dne: 9. června 2020

Lengál Ondřej, Ing., Ph.D.
oponent