FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Posudek oponenta závěrečné práce

Student:	Oleksandr Zaporozhchenko
Oponent práce:	Ing. Jan Trávníček, Ph.D.
Název práce:	Zpracování POSIX regulárních výrazů
Obor:	Teoretická informatika

Datum vytvoření: 15. 6. 2020

Hodnotící kritérium:	*Způsob hodnocení – následující škálou 1 až 4:*
1. Splnění zadání	***1=zadání splněno,*** *2=zadání splněno s menšími výhradami,* *3=zadání splněno s většími výhradami,* *4=zadání nesplněno*

Popis kritéria:
Posuďte, zda předložená ZP dostatečně a v souladu se zadáním obsahově vymezuje cíle, správně je formuluje a v dostatečné kvalitě naplňuje. V komentáři uveďte body zadání, které nebyly splněny, posuďte závažnost, dopady a případně i příčiny jednotlivých nedostatků. Pokud zadání svou náročností vybočuje ze standardů pro daný typ práce nebo student případně vypracoval ZP nad rámec zadání, popište, jak se to projevilo na požadované kvalitě splnění zadání a jakým způsobem toto ovlivnilo výsledné hodnocení.

Komentář:
Zadání bakalářské práce je jak teoreticky tak prakticky velmi zajímavé a možná i obtížností přesahující bakalářskou práci. Zadání bylo studentem splněno.

Hodnotící kritérium:	*Způsob hodnocení – bodové hodnocení 0 až 100 bodů (známka A až F):*
2. Písemná část práce	*95 (A)*

Popis kritéria:
Zhodnoťte přiměřenost rozsahu předložené ZP vzhledem k obsahu, tj. zda všechny části ZP jsou informačně bohaté a ZP neobsahuje zbytečné části. Dále posuďte, zda předložená ZP je po věcné stránce v pořádku, případně vyskytují-li se v práci věcné chyby nebo nepřesnosti. Zhodnoťte dále logickou strukturu ZP, návaznosti jednotlivých kapitol a pochopitelnost textu pro čtenáře. Posuďte správnost používání formálních zápisů obsažených v práci. Posuďte typografickou a jazykovou stránku ZP, viz Směrnice děkana č. 26/2017, článek 3. Posuďte, zda student využil a správně citoval relevantní zdroje. Ověřte, zda jsou všechny převzaté prvky řádně odlišeny od vlastních výsledků, zda nedošlo k porušení citační etiky a zda jsou bibliografické citace úplné a v souladu s citačními zvyklostmi a normami. Zhodnoťte, zda převzatý software a jiná autorská díla, byly v ZP použity v souladu s licenčními podmínkami.

Komentář:
Analýza časové složitosti algoritmu pro porovnání hodnoty referenčního slova s řetězcem by zasloužila Lemma.
Lemma 1.1 je dle práce nové, ale nachází se v kapitole, která je efektivně kapitolou rešeršní. Čekal bych ho v práci později.
Bod 3 příkladu 2.1: w' se má spíše rovnat baab.
Navržená gramatika pro regulární výrazy se zpětnými referencemi neumožňuje dvojí použití iterace. Zároveň je gramatika navržená jako by binární operace sjednocení a zřetězení byly pravě asociativní. Po převedení gramatiky na LL(1) gramatiku tento návrhový nedostatek korektně mizí a určení asociativity je dále závislé jen na návrhu sémantických pravidel pro vzniklou LL(1) gramatiku.
Algoritmus 2 je náchylný na zacyklení. Tento fakt je bohužel diskutován až v navazujících kapitolách.
Text práce je jinak dle mého názoru v pořádku a informačně bohatý.
Velmi pozitivně hodnotím nové poznatky ve formě alternativních důkazů a formulací týkajících se regulárních výrazů se zpětnými referencemi.

Hodnotící kritérium:	*Způsob hodnocení – bodové hodnocení 0 až 100 bodů (známka A až F):*
3. Nepísemná část, přílohy	*70 (C)*

Popis kritéria:
Dle charakteru práce se případně vyjádřete k nepísemné části ZP. Například: SW dílo – kvalita vytvořeného programu a vhodnost a přiměřenost technologií, které byly využité od vývoje až po nasazení. HW – funkční vzorek – použité technologie a nástroje, Výzkumná a experimentální práce
– opakovatelnost experimentů

Komentář:
Nepísemná příloha byla testována z repozitáře https://gitlab.fit.cvut.cz/zaporole/regex_matcher commitu e8f903a2e80a041e5fc727117d2e3907cff9df77.

Implementace přijímá na vstupu v regulárních výrazech epsilon a prázdný regulární výraz jako speciální znaky, ale tohoto se drží i v pozdějších fázích parsování regulárních výrazů, i když by je lexer dále mohl už předávat jako speciální tokeny. Takový přístup by byl vhodný především v reprezentaci AST, kde by tím pádem mohly vzniknout uzly EpsilonAST a EmptyAST.

Podobně by main funkce mohla rovnou přeložit uživatelský parametr na hodnotu z výčtu a následně by konstruktor třídy Matcher nemusel znát význam hodnot předávaných argumenty do programu.

V implementaci parseru je nevhodně využíváno volání std::move, které tak nedovoluje kompilátoru dostatečně optimalizovat. Celkově je move semantika v kódu používána neideálně.

I když je pro implementaci zvolený standard C++14, některé bloky kódu odpovídají C++11 nebo i dřívějším přístupům (časté nevyužívání for-each varianty for cyklu i když je to možné, ...), nebo dokonce ani nevyužívají vhodnějších jazykových konstrukcí (inicializace instančních proměnných mimo member initializer list).

Implementace vůbec nevyužívá neznaménkových celočíselných typů a to ani na místech, kde by k tomu mělo využití kontejnerů standardní knihovny vést.

Implementace využívá mapy pro reprezentaci navštívených stavů (proměnná visited) v metodě AvdFA::avdMemory, ale pracuje s ní jako se setem.

Metoda MemoryAutomaton<T>::accepts při simulaci automatu v každém kroku automat klonuje, a to včetně včetně například přechodové funkce. Rozdělením automatu na jeho definici a konfiguraci by se snížily požadavky na kopírování.

Podobně je zbytečně kopírováno i v metodě MemoryAutomaton<T>::accepts, kde by byla vhodnější const auto & trans místo jen auto. To je vidět i na obrázku 4.2.

V implementaci je často využíváno magických konstant.

Lexer nevyužívá zapouzdření a tedy není navržen v objektově orientovaném přístupu korektně.

Proměnným v konstruktoru třídy Matcher lze snížit rozsah platnosti.

Testování probíhalo podle textu po kompilaci bez optimalizací.
Podle textu by mělo volání make testTime <název_sady> spustit testování pro konkrétní sadu regexů, makefile bohužel tento cíl nedefinuje. Makefile podporuje cíl time, který zdá se jen vyvolává script testime.sh, jehož výstupem jsou časové údaje doby výpočtu zda slovo patří do jazyka regulárního výrazu, ale také velké množství informací, že "zadaný soubor neexistuje" a nebo "(standard_in) 1:syntax error".

Implementace by jistě mohla být rychlejší (viz výše) a pro porovnání na rychlost s nástrojem grep a regulárními výrazy jazyka PERL je potřeba využít optimalizací, jinak je měření neporovnatelné.

Místo volání exit bych raději viděl použití výjimek.

Hodnotící kritérium:	*Způsob hodnocení – bodové hodnocení 0 až 100 bodů (známka A až F):*
4. Hodnocení výsledků, jejich využitelnost	*85 (B)*

Popis kritéria:
Dle charakteru práce zhodnoťte možnosti nasazení výsledků práce v praxi nebo uveďte, zda výsledky ZP rozšiřují již publikované známé výsledky nebo přinášející zcela nové poznatky.

Komentář:
Implementace je funkční. Bohužel, pro reálné použití, byť i jako referenční implementace, by vyžadovala relativně velký přepis.

Hodnotící kritérium:	*Způsob hodnocení – nehodnotí se*
5. Otázky k obhajobě	

Popis kritéria:
Uveďte případné dotazy, které by měl student zodpovědět při obhajobě ZP před komisí (body oddělte odrážkami).

Otázky:
Proveďte měření při použití optimalizací a prezentujte jeho výsledky.

Hodnotící kritérium:	*Způsob hodnocení – bodové hodnocení 0 až 100 bodů (známka A až F):*

6. Celkové hodnocení *82 (B)*

Popis kritéria:
Shrňte stránky ZP, které nejvíce ovlivnily Vaše celkové hodnocení. Celkové hodnocení nemusí být aritmetickým průměrem či jinou hodnotou vypočtenou z hodnocení v předchozích jednotlivých kritériích. Obecně platí, že bezvadně splněné zadání je hodnoceno klasifikačním stupněm A.

Text hodnocení:

Text práce hodnotím výborně, bohužel nepísemná příloha velmi snižuje dojem z práce a musím tak celkově hodnotit výslednou známkou B (velmi dobře).

Podpis oponenta práce: