Posudek bakalářské práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Maxmilian Kulikov
Název práce	Emulátor zvukových syntezátorů
Rok odevzdání	2022
Studijní program	Informatika
Studijní obor	IOI
Autor posudku	Miroslav Kratochvíl Oponent
Pracoviště	Luxembourg Centre for Systems Biomedicine

K celé práci

	lepší	OK	horší	nevyhovuje
Obtížnost zadání		X		
Splnění zadání			X	X
Rozsah práce *… textová i implementační část, zohlednění náročnosti*			X	

Autor v práci popisuje konstrukci jazyka pro implementaci zvukových generátorů a efektů. Hlavním problémem práce je zcela minimální překryv se zadáním, autor v práci navíc několikrát dodává, že ji nestihl dokončit, což je na kvalitě výsledku bohužel poznat. Práci by pravděpodobně bylo vhodnější odevzdat v dokončenějším stavu v pozdějším termínu, současná podoba je na hranici obhajitelnosti.

Návrh odevzdané části (tvořené kompilátorem malého specifického jazyka pro implementaci DSP efektů) je nedostatečně opodstatněný. Konkrétně není jasné, proč je konstrukce poměrně složitého kompilátoru jazyka nutná ke splnění zadání, když je na výrobu zvukových efektů k dispozici velké množství existujících frameworků (Ladspa/Nyquist pluginy, SOX pluginy, JACK framework, Timidity sequencer, …) i specializovaných jazyků s existujícími kompilátory (např. LISPX v případě Nyquist pluginů, některé varianty Lua, různé jednoduché podmnožiny FORTHu nebo Scheme, …), v některých jazycích jsou pak pro rychlé streamové výpočty k dispozici celé knihovny (např. Haskellovský `conduit`). V určitém smyslu interpretace zadání by opravdu stačil i jednoduchý, ryze konfigurační jazyk, zakódovaný např. JSONu, YAMLu nebo Dhallu. Napojení na jakékoliv zpracování zvuku v práci chybí a v textu je zmíněné jen okrajově.

Ve zbytku posudku tuto odchylku od zadání přehlížím a soustředím se jen na odevzdanou část, tj. implementaci kompilátoru. Práce je i tak poněkud neúplná, a to především kvůli nedostatečné analýze problému a absenci demonstrovatelných výsledků.

Textová část práce

	lepší	OK	horší	nevyhovuje
Formální úprava *… jazyková úroveň, typografická úroveň, citace*		X	X	
Struktura textu *… kontext, cíle, analýza, návrh, vyhodnocení, úroveň detailu*				X
Analýza				X
Vývojová dokumentace		X	X	
Uživatelská dokumentace			X	X

Hlavním problémem textové části práce je úplně chybějící analýza. Dal by se očekávat mírný úvod do problematiky zpracování zvuku, sekvencování MIDI a 'big picture' pohled na celé zadání a cíle práce. Chybí analýza vstupních dat pro výsledné syntezátory (jakým způsobem by se MIDI eventy daly reprezentovat v čistě bufferovém jazyce, který popisuje autor?). Chybí systematický popis modelu kompilace a spouštění jazyka (Vypadá to, že jazyk se bude spouštět jako 'kernel' na bufferových oknech určité velikosti; jakým způsobem je definováno jak velké budou výstupní buffery? Jde spouštění paralelizovat? Jakým způsobem se zachycené kontexty funkcí (tj. uzávěry/closures, viz. konec sekce 2.2.10) konvertují na statické struktury? Je tato konverze úplná? Co se stane v případech kdy program potřebuje dynamické množství těchto kontextů?) a modelu za prezentovaným typovým systémem (Popis v sekci 2.2.2 je extrémně neformální a náhodně míchá koncepty z několika druhů typových systémů. Po přečtení by mi přišlo logické, že jde vyrobit konstantní reference na 'IO typ', jaká by byla sémantika takového typu? Kde je specifikovaná sémantika zmíněných 'IO' typů?). Chybí formální vyhodnocení výsledku (např. rigorózní měření) nebo jakákoliv netriviální ukázka výstupu kompilátoru.

Z návrhu není jasné, jakým způsobem by bylo možné vyrobit poměrně běžný zdroj zvukového signálu, který by měl vlastní paměť (např. aby si pamatoval stav generátoru náhodných čísel nutného pro generování nějakého pokročilého druhu šumu). Kromě chybějící úvahy o podpoře MIDI signálu chybí i potenciální podpora pro jiné typy signálu než běžný 'waveform', např. signál přenesený jako série spekter.

Nedostatečná analýza je pravděpodobně důsledkem nedostatečného seznámení autora se současnými výsledky a metodami konstrukce překladačů, což je možné dedukovat z *úplné absence* jakéhokoliv odkazu na odbornou literaturu.

Styl a jazyk práce by si zasloužily dodatečnou korekci (a to i po rozsáhlých erratech dodaných po odevzdání práce).

Uživatelská dokumentace prakticky chybí. Text práce by se celkově dal považovat za neformální vývojovou dokumentaci — kapitoly ostatně popisují 'seznámení s jazykem', velmi neformální 'specifikaci' a zevrubný 'popis implementace'.

Výsledky uvedené v závěru nejsou v práci nijak demonstrované: Autor zmiňuje, že (nějak) otestoval schopnost jazyka vypočítávat "potřebné vzorky signálu v rozumném rozsahu" (bohužel jsem nenašel ani generátor sinusoidy); "rychlost překladu" (chybí jakýkoliv náznak benchmarku, tj. minimálně metrika a metodologie srovnání, a ideálně vhodně zvolená baseline); "expresivita k popisu základních operací při zpracování signálu" (základní operací většiny DSP pluginů je nějaká varianta Fourierovy transformace, která není implementovaná nebo demonstrovaná ani v základní podobě) a "vytváření komponent syntezátoru" (to je demonstrované velmi povrchně v sekci 1.7, potřebná sémantika bufferů a signálu je popsána jen implicitně pomocí několika příkladů). Autor neuvádí zdroje informací pro zmíněná "tři nově nastudovaná témata".

Implementační část práce	lepší	OK	horší	nevyhovuje
Kvalita návrhu *… architektura, struktury a algoritmy, použité technologie*			X	
Kvalita zpracování *… jmenné konvence, formátování, komentáře, testování*		X	X	
Stabilita implementace		X		

Implementační částí práce je transpiler specifického jazyka připomínajícího embedované C++ do C++. Program na dodaných příkladech funguje (tj. generuje *nějaký* kód). Kvůli nedostatku dokumentace je prakticky nemožné ho vyzkoušet na praktičtějších příkladech. Zreprodukovat výstup příkladu 'Hello world' ze sekce 1.1 se mi nepodařilo.

Autor v programu implementuje tradiční lexer a parser (flex/bison) který vstupní jazyk překládá do komplikovaného AST, ze kterého se poměrně jednoduchým transpilerem emituje C++ kód. Při návrhu AST by bylo vhodné se inspirovat existujícími návrhy jazyků, např. použít nějaký zjednodušený formální model pro snížení komplexity datových struktur (jednotlivých 'typů' uzlů v AST je opravdu hodně) a pro zjednodušení fungování transpileru (systém několika kontextů pro různé úkoly vypadá dost neuchopitelně).

Dokumentace kódu je minimální, struktura kódu je komentovaná jen řídce. Použité C++ je relativně moderní, bohužel trpí některými tradičními neduhy zbytečně objektového návrhu (detaily viz. níže). Vzhledem ke stylu návrhu programu by se vyplatilo použít programovací jazyk vhodnější pro transformace stromových struktur, ideálně dovolující pattern matching: např. OCaml nebo Haskell, případně nějakou verzi Prologu embedovatelnou/přeložitelnou do C++. Velká část kódu kompilátoru je takto tvořena 'boilerplatem' nutným pro správu stromové struktury.

Hlavní nedostatky implementace jsou následující: Některé části kódu vypadají neudržovatelně (např. dlouhé bloky všech možných implementací funkcí nad AST pro různé typy, hlavně instance `component_allocator` a `component_deleter`), především pokud uvážíme další plánovaná rozšíření kompilátoru. Některé C++ konstrukce jsou neopodstatněně krkolomné (např. není jasné, proč by kompilátor potřeboval compile-time konstanty pro mocniny desítky implementované výběrem z `constexpr` statické arraye (!)). Občas se vyskytují běžné chyby návrhu datových struktur, především v souvislosti se správným využitím RAII (časté použití manuálního `new/delete` a zbytečné indirekce). Zajímavostí zůstává nevysvětlená (ale viditelně zbytečná) nutnost alokovat i `std::string` (soubor `src/esl/component.cpp`, ř. 24–27).

Styl použití C++ je obecně pro kompilátorový projekt poměrné nevhodný: Separace interfaců od implementací není nijak systematická, a autor kvůli místy zbytečnému šablonování tříd používá "include everything" headery. V důsledku mě překvapil poměrně nepříjemný kompilační čas projektu (občas přes 30 sekund na jeden C++ modul).

Celkové hodnocení Neprospěl

Práci navrhuji na zvláštní ocenění Ne

Datum Podpis