LATVIJAS UNIVERSITĀTES
EKSAKTO ZINĀTŅU UN TEHNOLOĢIJU FAKULTĀTES
DATORIKAS NODAĻA

INTERPRETATORS PROGRAMMĒŠANAS VALODAI IMMUTANT

KVALIFIKĀCIJAS DARBS DATORZINĀTNĒS

Autors: Rolands Frīdemanis
Studenta apliecības Nr.: rf23009
Darba vadītājs: ...

RĪGA, 2025

Anotācija

Mūsdienās liela daļa no vispārīga pielietojuma programmēšanas valodām, satur sematiku, kas paredz, ka datu mainība ir ierasta lieta, tomēr šim pastāv būtisks trūkums, kas ir datu neparedzamība. Lai spriešanu par programmas stāvokli padarītu paredzamu, šī darba ietvaros tiek izstrādāta programmēšanas valoda, kuras gramatika un sintakse ierosina noteikta datu nemainību. Šis darbs satur valodas specifikāciju un interpretatora arhitektūras dokumentāciju lekserim, parsētājam un abstraktā sintakses koka pārstaigāšanas algoritmam. Rezultātā, izmantojot valodu C, tiek izveidota augsta līmeņa, intepretējama, dinamiski tipizēta valoda ar atomāriem datu tipiem.

Atslēgvārdi: interpretators, AST, funkcionālā programmēšana, C valoda, datu nemainība

Abstract

Most modern general-purpose programming languages use grammar and syntax that suggests mutable data being an ordinary matter, which in reality complicates reasoning about program state. To make such reasoning predictable, this work explores the design and delelopment of a programming language with explicit syntax and semantics of immutable data. This work contains specification for such language and documentation of the architecture for the lexer, parser and AST-walker of the underlying interpreter. As a result, a high-level, interpreted, dynamically typed programming language with only atomic data types is created. The language is implemented in C.

Keywords: interpreter, AST, functional programming, C language, data immutability

SATURS

APZĪMĒJUMU SARAKSTS

AST	Abstraktās sintakses koks
immutability, immutable	Pilnīga (datu) nemainība, pilnībā nemaināmi dati

IEVADS

Datu mainība un to nepārtraukta plūsma no viena mainīga uz otru ir neapstrīdama daļa no lielas daļas programmatūras. Pavisam noteikti var apgalvot, ka datorsistēmu arhitektūru atmiņas koncepcija ļauj rakstīt atmiņā ne vienu vien reizi. Atmiņu var relocēt, izdzēst, pieprasīt lielākus atmiņas gabalus u.t.t. No šī izriet datu mainības esence un līdz šai dienai tā pamata sastāvdaļa iekalta lielā daļā, ja ne visu, programmēšanas un skriptēšanas valodās.

No otras puses, mainīgiem datiem pastāv īpašība būt neparedzamiem. Kamēr mazas sistēmas spēj tikt galā ar nelielu daudzumu mainīgo, tad lielajās sistēma tas var kļūt par acīmredzamu problēmu. Atmiņa datoram ir viena, tomēr atsaukties uz to var no dažādām vietām dažādos laika brīžos, kas padara spriešanu par datu stāvokli daudz sarežģītāk.

Programmētāji izmanto iespēju datu mainību it īpaši jo tas ir tik viegli, izmantojot programmēšanas valodas un to abstrakcijas. Lai padarītu iepriekš minēto programmatūras uzvedību kontrolētāku, valodas no C saimes, Java, JavaScrip un citas valodas satur gramatikas, kas ierobežo mainīgu datu inicializāciju un to turpmāko vērtību maiņu. Kā piemēru var minēt `final` atslēgvārdu no valodas Java vai līdzīgu funkciju `Object.freeze` no valodas JavaScript, kas saldē jeb fiksē doto mainīgo un liedz pārrakstīt tā vērtību, bet tas nenotiek visos gadījumos. Ja mainīgā vērtība tips nav primitīvas, bet atsauces, piemēram kāds objekts vai masīvs, tas ir definēts pavisam citā atmiņas apgabalā, tāpēc realitātē tas gan būs maināms. No iepriekš minētiem piemēriem var secināt, ka arī datu nemainības implementācija programmēšanas valodās nav tikai saturiska no sintakses viedokļa, bet tai pastāv arī semantika jeb raksturojoša uzvedība pie dažādām operācijām.

Immutability is widely recognized in programming languages [**?**]

BIBLIOGRĀFIJA

[1] Leslie Lamport. *LaTeX: a Document Preparation System*. Addison Wesley, Massachusetts, 2 edition, 1994.