Programavimo kalbų plėtros mechanizmų tyrimas ir tobulinimas

2015 - 2019

Doktorantas: Audrius Šaikūnas
Darbo vadovas: Albertas Čaplinskas

Tyrimo objektas ir tikslai

Objektas: refleksyviai plečiamos programavimo (REP) kalbos.

Tikslas: realizuoti sintaksinės analizės algoritmą, kuris būtų tinkamas refleksyvioms kalbos analizuoti.

Ataskaita

- Disertacijos eksperimentinė dalis parašyta:
  - Realizuotas algoritmas tinkantis analizuoti REP kalbas.
  - Algoritmas palygintas su kitais egzistuojančiais sintaksinės analizės algoritmais.
- Priduotas straipsnis „*Parsing with Scannerless Earley Virtual Machines*“ į BJMC žurnalą.

Plečiamumas (priminimas)

- **Plečiamos kalbos** – nėra aišku ką reiškia žodis „plečiamos“.

- **Sintaksiškai plečiamos prog. kalbos** – nėra tiksliai įvardintas plėtimo mechanizmas. Plėtimas galimas per:
  - Išorinius įrankius
  - Kompiliatoriaus įskiepius (*plugins*)
  - Specialus meta-kalbų aprašus, kurie pateikiami kompiliatoriui prieš kompiliuojant
  - Makrokomandas

- **Refkelsyviai plečiamos prog. kalbos** (REP) – naujas terminas:
  - Kalba, kurios bekonteksčiai plėtiniai yra aprašomi metakalba.
  - Kalba, kur metakalba ir kalba sutampa.
    - Plėtimas vyksta kompiliavimo metu.
    - Metakalbą ir kalbą galima maišyti viename faile.

Reikalavimai analizatoriui

Pagrindiniai reikalavimai:
- Turi veikti be dedikuoto leksinio analizatoriaus.
- Turi atpažinti visas bekontekstes kalbas.
- Turi palaikyti adaptyvias/refleksyvias gramatikas.
- Turi veikti pakankamai greitai.

Papildomi reikalavimai:
- Turi palaikyti nuo duomenų priklausomus apribojimus.
- Turi palaikyti reguliarius operatorius gramatikose.

EVM vs SEVM

SEVM patobulinimai:

- Labiau tinkamas analizei be dedikuoto lekserio: būsenos/įrašai kuriamos tik ties neterminalinių simbolių pradžiomis.
- Palaiko dalinai inkorporuotas redukcijas.
- Palaiko JIT, kad sukompiliuoti gramatikas į mašininį kodą.
- Determinuotų fragmentų iškėlimas.
- Palaiko papildomus dviprasmybių eliminavimo mechanizmus:
  - Neigiamas redukcijas.
  - Leksemų lygmens dviprasmybių eliminavimą.

Sintaksinė analizė su SEVM

- Vartotojas aprašo *pradines gramatikas*.
- SEVM transliuoja *pradines gramatikas* į *kompiliuotas gramatikas*, sudarytas iš instrukcijų.
- Optimizatorius atlieka poaibių konstrukciją ir generuoja optimizuotų gramatinių taisyklių instrukcijas.
- Instrukcijas panaudojant LLVM yra verčiamos į mašininį kodą.

- Gramatikos sudarytos iš gramatinių taisyklių.
- Vieną gramatinę taisyklę ties konkrečia įvesties vieta įvykdo viena *užduotis*.

SEVM našumo įvertinimas

Kad įvertinti SEVM našumą, buvo atlikti šie žingsniai:

- Sukurta SEVM realizacija `north`.
- Pasirinktos dvi programavimo kalbos: ANSI C ir Rust; jų gramatikos buvo realizuotos.
- Pasirinkti keli analizės algoritmai palyginimui:
  - bison (+ flex)
  - dparser
  - syn
  - yaep (+ flex)

SEVM našumo įvertinimo rezultatai

Metodas	Kalba	IQR	Išskirtys (%)	Laikas (s)	Rel. laikas
bison	ANSI C	0.0008	20.0	0.4974	1.00
dparser	ANSI C	0.0104	20.0	16.1007	32.36
north	**ANSI C**	**0.0162**	**0.0**	**4.6132**	**9.27**
yaep	ANSI C	0.0737	0.0	1.7231	3.46
north	**Rust**	**0.0197**	**0.0**	**6.3258**	**1.14**
syn	Rust	0.0346	0.0	5.5434	1.00

- ANSI C įvestis: 470 000 eilučių, 14.8 MB
- Rust įvestis: 650 000 eilučių, 22.3 MB
- N = 10

Reikalavimai analizatoriui

Pagrindiniai reikalavimai:

- Turi veikti be dedikuoto leksinio analizatoriaus.
  - **SEVM veikia be leksinio analizatoriaus.**
- Turi atpažinti visas bekontekstes kalbas.
  - **Gramatikoms nėra taikomi specialūs apribojimai.**
- Turi palaikyti adaptyvias/refleksyvias gramatikas.
  - **JIT + optimizatorius.**
- Turi veikti pakankamai greitai.
  - **Atliktas palyginimas su kitais analizės metodais.**

Kitų metų darbo planas

- Sudalyvauti tarptautinėje konferencijoje.
- Baigti rašyti ir apsiginti disertaciją.

Ačiū už dėmesį!