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: sukurti refkelsyviai plečiamos programavimo kalbos kompiliatorių.

Planuojami rezultatai

- ~~Išanalizuoti egzistuojantys sintaksinės analizės algoritmai.~~
- Naujas sintaksinės analizės algoritmas, kuris yra tinkamas REP kalbų analizei.
- Pavyzdinė REP kalba.
- Pavyzdinė REP kalbos realizacija.
- Tokios kalbos demonstraciniai atvejai (pvz. SQL užklausų, XML konstruktorių integravimas į bazinę kalbą).

Ataskaitinių metų darbo planas

- Parašyti disertacijos literatūros apžvalgą.
- Parašyti straipsnį
- Atsiskaityti kursą „*Informatikos inžinerijos matematiniai metodai*“ (dr. Gintautas Tamulevičius)
- Atsiskaityti kursą „*Informatikos ir informatikos inžinerijos tyrimo metodai ir metodika*“ (prof. dr. Albertas Čaplinskas)

Ataskaita

- Disertacijos literatūros apžvalga parašyta.
- Straipsnis „*Critical Analysis of Extensible Parsing Tools and Techniques*“ pateiktas į BJMC žurnalą.
- Atsiskaitytas kursas „*Informatikos inžinerijos matematiniai metodai*“ (įvertinimas: **10**)
- Atsiskaitytas kursas „*Informatikos ir informatikos inžinerijos tyrimo metodai ir metodika*“ (įvertinimas: **10**)

Plečiamumas

- **Plečiamos kalbos** – nėra aišku ką reikškia 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.

REP kalb. pavyzdys 1/2 (Katahdin)

```
method Test0() {
  //print(a % b); // Error
}

class ModExpression : Expression {
  pattern {
    option leftRecursive;
    a:Expression "%" b:Expression
  }

  method Get() {
    a = this.a.Get...();
    b = this.a.Get...();
    return a - (b * (a / b));
  }
}

method Test1() {
  print(a % b);
}
```

REP kalb. pavyzdys 2/2 (Katahdin)

```
import "fortran.kat";
import "python.kat";

fortran {
  SUBROUTINE ADD(A, B)
    INTEGER A
    INTEGER B
    A = A + B
    RETURN
  END
}

python {
  total = 0
  for i in range(10):
    ADD(total, i)
    print total
}
```

Reikalavimai sintaksiniam analizatoriui

R1. Turi palaikyti mutuojamas gramatikas
R2. Turi veikti be dedikuoto skanerio
R3. Turi atpažinti visas bekontekstes kalbas
• Įskaitant ir ε-nulines.
R4. Turi būti priimtinas našumas
• LR(k) gramatikoms turi būti būti $O(c)$ su *nedidele* konstanta.
• Nedviprasmiškoms – $O(n^2)$.
• Bekontekstėms – $O(n^3)$.
• Visoms kitoms – neribotas.

Nagrinėti sintaksinės analizės alg.

Nagrinėti algoritmai:

- LR(k)
- **GLR šeima (tradinicnis GLR/RNGLR/RIGLR)**
- Rekursyviai nusileidžiantis
- Packrat
- **APEG**
- Specificity
- Earley
- Efektyvios Earley variacijos
- **Yakker**

GLR šeimos sintaksiniai anaizatoriai

+ Tinka belekserinei analizei (R2).
+ Atpažįsta visas bekonstekstes kalbas (R3).
+ Geras našumas (R4).
- Nepalaiko mutuojamas gramatikas (R1).

Kad tiktų REP kalboms analizuoti, reikia pritaikyti [CV14] algoritmą, kuris skirtas LR(0) lentelėms generuoti, kad šis veiktų su RN lentelėmis, kurios naudojamos RNGLR analizatoriuje.

Yakker

+ Tinka belekserinei analizei (R2).
+ Atpažįsta visas bekontekstes gramatikas (R3).
+ Geras našumas (R4).
+ Atpažįsta gramatikas su nuo duomenų priklausančiais apribojimas (*data-dependent grammars*).
+ Palaiko atidėtus semantinius veiksmus.
- Nepalaiko mutuojamas gramatikas.

Kad tiktų REP kalboms analizuoti, reikia dinamiškai generuoti grafą, kuris naudojamas kaip vidinė gramatikos reprezentacija.

APEG

+ Palaiko mutuojamas gramatikas (R1).
+ Tinka belekserinei analizei (R2).
+ Geras našumas (R4).
+ Gera klaidų diagnozė.
+ Palaiko nuo konteksto priklausomus apribojimus.
- Atpažįsta ribotą gramatikų klasę (PEG) (R3).
    Nepalaiko kairiosios rekursijos.
    Nepalaiko dviprasmiškų gramatikų.

Kadanti algoritmas fundamentiškai paremtas PEG analize, jo praplėsti, kad atpažintų visas bekontekstes gramatiakas neišeis.

Nagrinėtos kalbos ir įrankiai

- Katahdin
- SugarJ
- Neverlang

Katahdin

- Naudoja rekurysviai nusileidžiantį sintaksinį analizatorių
- Semantiniam plėtimui naudojamas AST interpretavimas

Esminiai trūkumai:

- Ribota gramatinių plėtinių klasė
- Dinamiškai tipizuota – nėra kaip programas tikrinti kompiliavimo metu
- Itin mažas našumas:
  - Naudojamas nefektyvus rekursyviai nusileidžiantis analizatorius su dinaminėmis produkcijom
  - Programos interpretuojamos
- Globalūs sintaksiniai plėtiniai

Katahdin: pavyzdys

```
method Test0() {
  //print(a % b); // Error
}

class ModExpression : Expression {
  pattern {
    option leftRecursive;
    a:Expression "%" b:Expression
  }

  method Get() {
    a = this.a.Get...();
    b = this.a.Get...();
    return a - (b * (a / b));
  }
}

method Test1() {
  print(a % b);
}
```

SugarJ

- Naudoja SGLR analizatorių su naiviu gramatiniu plėtimu
- Semantika aprašoma per AST transformavimo taisykles

Esminiai trūkumai:

- Ribotas našumas – naivus sintaksinis plėtimas nėra efektyvus
- Ritobos galimybės aprašyti sudėtingesnius semantinius plėtinius – visa semantika dabar aprašoma per AST transformacijas
- Globalūs sintaksiniai plėtiniai

Apibendriminas & Išvados

- Nėra REP programavimo kalbos, kuri pilnai tenkintų visus poreikius
- Nėra sintaksinio analizės algoritmo, kuris tiktų REP kalbai analizuoti. Tačiau yra keli geri kandidatai:
  - **RNGLR** – tenkina visus reikalavimus, bet nepalaiko mutuojamų gramatikų (R1).
  - **Yakker** – tenkina visus reikalavimus, bet nepalaiko mutuojamų gramatikų (R1).
  - **APEG** – tenkina visus reikalavimus, bet yra apribota gramatinių plėtinių klasė (R3).

Kitų metų planas

- Sukurti analizės algoritmą, tenkinantį visus REP kalbos analizės reikalavimus
- Aprašyti sukurtą algoritmą disertacijoje
- Sudalyvauti tarptautinėje konferencijoje
- Parašyti straipsnį apie Earley virtualias mašinas (planujamas straipsnis „*Earley Virtual Machine*“)
- Atsiskaityti kursą „Programavimo kalbų teorija“
- Atsiskaityti kursą „Generatyviojo ir aspektinio programavimo metodai“

Ačiū už dėmesį!