A Programming Language for Designing Turing Machines

Leysen, Jens
Universiteit Antwerpen, België
jens.leysen@student.uantwerpen.be

De Maeseneer, Dries
Universiteit Antwerpen, België
dries.demaeseneer@student.uantwerpen.be

Marinus, Sander
Universiteit Antwerpen, België
sander.marinus@student.uantwerpen.be

Hamelink, Frederic
Universiteit Antwerpen, België
frederic.hamelink@student.uantwerpen.be

1 Onderwerpen Individuele Toepassingsopdracht

- CNF (Dries De Maeseneer, Frederic Hamelink)
- CYK (Jens Leysen, Dries De Maeseneer, Sander Marinus)
- PDA2CFG (Jens Leysen, Dries De Maeseneer, Sander Marinus)
- LL1 (Sander Marinus)

2 Onderwerp Groepsopdracht

2.1 Introductie

Ons project voor machines en Berekenbaarheid gaat over een gestructureerde taal om Turing machines te ontwerpen. Dit taaltje laat ons toe om snel Turing machines te designen, te testen en die visueel voor te stellen om toekomstige Informatica studenten te helpen de leestrof beter te doen begrijpen. Om dit

project te realiseren gaan we een context vrije taal opstellen dat de LALR parser zal gebruiken om te controleren of een inputbestand een correcte beschrijving bevat door middel van een AST(Abstract Syntax Tree). We gaan een Turing machine opstellen aan de hand van dat bestand en een interface aanmaken om de te simuleren Turing machine te visualizeren. Op deze manier zorgen we ervoor dat toekomstige informatica studenten op een correcte, snelle en handige manier kunnen bijleren over de werking van Turing machines. Ons project is gedeeltelijk gebaseerd op twee reeds bestaande projecten [1][2].

2.2 Geschiktheid Conferentie

Ons project werkt voornamelijk verdiepend op de reeds geziene theorie omtrent parsers en Turing machines. We denken dat we met dit project een mooie balans hebben gevonden tussen de theorie omtrent context vrije grammatica's, parsers en Turing machines. We denken dat dit project geschikt is voor de conferentie wegens volgende zaken:

- Verdiepend: We implementeren een nieuw parsing algoritme en error detectie via een AST.
- Maatschappelijk nut: Onze tool zou handig gebruikt kunnen worden om toekomstige informatica studenten te helpen bij het verwerken van automaten theorie.

2.3 Evaluatiecategorie

Wij denken dat ons voorstel binnen de 'goud' categorie valt. We implementeren een nieuw parsing algoritme en werken bovendien verdiepend op de reeds geziene theorie omtrent Turing machines.

2.4 Features

Het project zal volgende features bevatten:

1. Een CFL voor single- en multi-tape Turing machines.
2. Een parsing algoritme uit de volgende lijst: LR, SLR, LALR, Early.
3. Aanmaken van een single-tape Turing machine.
4. Simulatie van een single-tape Turing machine.
5. Dynamische visualisatie van de werking van een single-tape Turing machine.
6. Statische visualisatie van een single-tape Turing machine.
7. Het inlezen van een multi-tape Turing machine.
8. Het simuleren van een multi-tape tape Turing machine.

9. Code genereren aan de hand van AST.
10. Error detection AST.

Voor meer informatie omtrent de features, zie onderstaande sectie omtrent onze planning.

2.5 Planning

Alle sprints duren twee weken. Gedurende deze twee weken proberen we een set aan features te finaliseren. De eerste sprint begint 29 November. De laatste sprint eindigt op 9 Januari.

Sprint 1

- Het opstellen en bedenken van een CFL voor single- en multi-tape Turing machines. We willen een beschrijving van een context vrije taal die een correcte beschrijving van Turing machines geeft. Het doel is dus om een programmeertaal voor Turing machines te ontwerpen. (**Frederic**, 8 uur)
- De implementatie van een parsing algoritme uit de volgende lijst: LR, SLR, LALR, Early. Dit algoritme zal gebruikt worden om te testen of een beschrijving van een Turing machine geldig is volgens de beschreven programmeertaal. (**Dries**, 24 uur)

Sprint 2

- Het aanmaken van een virtuele one tape Turing machine gebaseerd op een geldige beschrijving in onze programmeertaal. (**Sander**, 16 uur)
- Het simuleren van een input op de Turing machine. We willen dus weten of een input geaccepteerd wordt door deze Turing machine. (**Sander**, 16 uur)
- Het genereren van C++ code aan de hand van een Abstract Syntax Tree (**Frederic**, 8 uur)
- Error detection van de AST (**Dries**, 6 uur)

Sprint 3

- Een dynamische visualisatie van de werking van een single-tape Turing machine. We willen stap per stap de uitvoering van onze single-tape Turing machine kunnen tonen. (**Frederic**, 28 uur)
- Een statische visualisatie van een Turing machine aan de hand van een diagram. (**Dries**, 12 uur)
- Het inlezen van een multi-tape Turing machine. (**Jens**, 12 uur)

- Simulatie van een multi-tape Turing machine (zie boven). (**Jens**, 24 uur)

Elk feature zal uitbunding getest worden gebruik makend van een testing framework (zoals Catch2). We zullen ook tijd steken in het maken van presentatie (2 uur per persoon), meetings (6 uur per persoon, 1 uur per week) en onderzoek. We denken ook dat we zeker 10 uur per persoon aan verbredend onderzoek zullen doen.

Dit geeft een schatting van:

- Jens: 54 uur.
- Dries: 60 uur.
- Sander: 50 uur.
- Frederic: 62 uur.

Referenties

[1] "Alan"(2021) https://github.com/kelvindecosta/alan, accessed 28-11-2021

[2] "turingmachine.io"(2021) https://turingmachine.io/, accessed 28-11-2021