

Projectvoorstel Talen en Automaten
Lexical analyser generator: *Lexesis*

Thomas Avé, Robin Jadoul

In fase I werd door deze groep reeds de omzetting van een reguliere expressie naar een ϵ -NFA¹ en beide **table filling** algoritmes² geïmplementeerd.

Voor fase II stellen we onszelf tot doel een programmeertaal agnostische generator voor lexicale analysators te maken. In tegenstelling tot de bekende en frequent gebruikte *flex* is het ons doel hierbij in de feitelijke specificatie voor de analysator geen specifieke programmeertaal te laten voorkomen (bij *flex* moet men specifieke code schrijven in de gewenste programmeertaal). Dit heeft tot voordeel dat eenzelfde input bestand eenvoudig hergebruikt kan worden in meerdere projecten, zonder verdere compatibiliteitsproblemen. Zo kan bijvoorbeeld een standaardcomité van een programmeertaal een voorbeeldspecificatie in dit inputformaat vrijgeven, en kunnen verschillende toepassingen³ daarvan gebruik maken zonder op eender welke manier geresliceerd te zijn, of van elkaar afhankelijk te zijn.

Intern zal de generator werken door een collectie van *reguliere expressies*⁴ die de gewenste *tokens* beschrijven om te zetten naar een ϵ -NFA, vervolgens middels **mssc** hier een DFA van te maken die (optioneel, te specificeren door de eindgebruiker) ook nog met **table filling** geminimaliseerd kan worden. Wanneer er meerdere mogelijke *tokens* zijn voor een bepaalde tekenreeks, zal deze ambiguïteit opgevangen worden door een voorrangsregel: de tekstueel vroegste *reguliere expressie* in het inputbestand zal voorrang krijgen.

Voor een gegeven invoerbestand zal dan de mogelijkheid bestaan om dit om te zetten naar een sourcecode bestand naar keuze⁵. Dit bestand kan dan mee gecompileerd worden (of geïnterpreteerd, afhankelijk van de taal) en op die manier een interface beschikbaar stellen waarmee de uiteindelijke toepassing een tekenreeks naar een stroom tokens kan omzetten. Bovendien zal de generator in staat zijn (dankzij de programmeertaal-onafhankelijkheid) een idiomatische interface vrij te geven, per programmeertaal.

Enkele van de toepassingen die mogelijk gemaakt worden met het gebruik van **Lexesis** zijn de volgende:

- Compiler frontends
- Linting
- Syntax highlighting
- (Deel van) syntax checking

¹Robin Jadoul

²Thomas Avé en Robin Jadoul

³Voor enkele voorbeelden van toepassingen, zie onder

⁴De posix variant, niet de *perl compatible regex*

⁵Indien de gewenste taal geïmplementeerd is, initieel wordt enkel C++ ondersteuning gepland

Om verder ook het nut en de gebruiksvriendelijkheid van onze generator aan te tonen, zullen we verder ook trachten een *syntax highlighter* voor een voorlopig onbesliste taal te schrijven.