Practical General Top-down Parsers

A. Izmaylova & A. Afroozeh

Samenvatting

Dit proefschrift presenteert het ontwerp en de implementatie van praktische, algemene top-down parsers. Top-down parsers, meestal in de vorm van recursive-descent parsers, zijn efficiënt, gemakkelijk te begrijpen en worden veel gebruikt om handmatig parsers voor echte programmeertalen te schrijven. Vanwege de moeilijkheid om links-recursie te behandelen, leiden top-down parsers altijd aan expressiviteitsproblemen in vergelijking met hun bottom-up tegenhangers.

In veel toepassingen van parseren, zoals het ontwerpen van domeinspecifieke talen, waarbij factoren zoals expressiviteit en gebruiksgemak belangrijk zijn, worden algemene parseringsalgoritmen gebruikt. Algemene parseringsalgoritmen ondersteunen alle context-vrije grammatica's en kunnen alle ambiguïteiten presenteren in de vorm van een compact bos van bomen. Het meeste onderzoek naar het ontwikkelen van algemene parsers is gericht op bottom-up parsers, in het bijzonder op het gegeneraliseerde LR-algoritme (GLR). GLR is een volwassen parseringsalgoritme, maar vanwege zijn bottom-up karakter, is het moeilijk te begrijpen en te modificeren. Er zijn in het verleden een aantal pogingen ondernomen om de expressieve kracht van recursive-descent parseren te vergroten door verschillende backtrackingtechnieken te gebruiken, maar een oplossing voor het probleem van links-recursie blijkt moeilijk te zijn.

Het gegeneraliseerde LL-algoritme (GLL), dat in 2010 door Scott en Johnstone is ontwikkeld, biedt een levensvatbaar alternatief voor GLR. GLL parsers ondersteunen alle context-vrije grammatica's (inclusief links-recursieve), ze lijken op recursive descent en hun runtime heeft een één-op-één relatie met de grammatica. Dit maakt GLL aantrekkelijk voor het parseren van programmeertalen. Ander, zeer belangrijk maar minder bekend werk, over het generaliseren van recursive-descent parsers, is het werk van Mark Johnson over CPS-herkenners gebaseerd op continuaties (Continuation Passing Style). Ons werk in dit proefschrift is geïnspireerd door zowel GLL als de CPS-herkenners van Johnson, en, zoals we hebben ontdekt, zijn deze twee algoritmen zo vergelijkbaar in de manier waarop ze links-recursie verwerken dat we ze in principe als hetzelfde algoritme kunnen beschouwen.

GLL is een relatief nieuw algemeen parseringsalgoritme dat in de praktijk nog niet veel is gebruikt. Het werken met de herkenners van Johnson heeft ons ertoe gebracht

om een efficiëntere Graph Structured Stack (GSS) voor GLL voor te stellen, die lijkt op het traditionele memoriseren van functieresultaten. Onze gewijzigde versie van GLL parseren is niet alleen sneller, maar ook gemakkelijker te begrijpen, en heeft de basis gelegd voor ons werk op het gebied van data-afhankelijk grammatica's.

We hebben ontdekt dat data-afhankelijk grammatica's een uitstekende abstracte tussentaal vormen voor het implementeren van verschillende disambigueringsconstructies zonder kennis van een specifieke parseringstechnologie. Omdat data-afhankelijke grammatica's van een nogal laag-niveau zijn, stellen we ook hoog-niveau disambigueringsconstructies voor, bijvoorbeeld voor prioriteitsregels van operatoren en indentatiegevoeligheid, die naar data-afhankelijke grammatica's kunnen worden vertaald. We bespreken de toepassing van onze techniek om verschillende ambiguiteiten op te lossen, bijvoorbeeld die gevonden in indentatiegevoelige talen zoals Haskell en Python, conditionele directieven in C#, de bekende `typedef` ambiguiteit in C, en ingewikkelde gevallen van operatorprioriteiten in OCaml.

Een belangrijk deel van dit proefschrift is gewijd aan de semantiek en implementatie van disambigueringsconstructies voor prioriteiten van operatoren. Disambigueringsconstructies voor operatorprioriteiten behoren tot de belangrijkste disambigueringsconstructies en zijn misschien het moeilijkste om correct te krijgen. We introduceren een afleidinggebaseerde semantiek voor operatorprioriteiten die onafhankelijk is van de onderliggende parseringstechniek, en veilig is, d.w.z. geen zinnen uit de taal verwijdert als er geen ambiguïteit is. Bovendien kan de semantiek voor operatorprioriteiten omgaan met zogenaamde *diepe* gevallen van operatorprioriteit die vaak voorkomen in functionele programmeertalen zoals OCaml. Onze veilige specificatie van regels voor operatorprioriteiten wordt geïmplementeerd door een automatisch grammaticaherschrijfproces dat de vorm van de afleidingsbomen behoudt, conform de oorspronkelijke ambigue grammatica. Deze herschrijving kan echter leiden tot zeer grote grammatica's, wat ons motiveerde om een alternatieve implementatie te ontwikkelen die is gebaseerd op data-afhankelijke grammatica's en die ambigue operatorprioriteiten kan oplossen tijdens runtime.

Onze laatste bijdrage bestaat uit algemene parsercombinators die alle context-vrije grammatica's kunnen verwerken en een binair Shared Packed Parse Forest (SPPF) in kubische tijd en ruimte kunnen produceren. Onze algemene parsercombinators hebben de flexibiliteit en expressiviteit van traditionele parsercombinators en de runtimegarantie van algemene parseringsalgoritmen. Onze algemene parsercombinators zijn gebaseerd op de CPS-herkenners van Johnson. We breiden Johnson's CPS-herkenners uit om herkenning in kubische tijd te bereiken en breiden de resulterende kubische CPS-herkenners uit naar parsers die een binair SPPF construeren. We hebben onze kubische CPS-parsers gebruikt als basis voor Meerkat, een algemene parsercombinatorbibliotheek voor Scala.

In de loop van dit proefschrift hebben we, tenslotte, Iguana ontwikkeld, een op GLL gebaseerde data-afhankelijk parseringsraamwerk. Met Iguana's API (Application Programmers Interface) voor uitbreidbare data-afhankelijke grammatica's kan de gebruiker gemakkelijk nieuwe disambigueringsconstructies toevoegen of bestaande constructies wijzigen. We hebben Iguana gebruikt om verschillende echte programmeertalen te parseren, zoals Java, C#, Haskell en een groot deel van OCaml. Onze

prestatieanalyses tonen aan dat Iguana praktisch bruikbaar is voor het parseren van echte programmeertalen en qua prestaties kan concurreren met een volwassen parseringstool zoals ANTLR.