Practicum Algoritmen en Datastructuren

– voorjaar 2011 –

Opgave 5: ~~Beweren~~ *Parseren en bewijzen*

1 Achtergrond

Waterdichte bewijzen van niet triviale stellingen zijn moeilijk om te construeren. Ook al kunnen anderen het nalezen en zelf controleren, (herstelbare) fouten worden niet altijd opgemerkt. Een manier om bewijzen rigoureus te controleren is met zogenaamde *theorem provers*. Dit houdt echter wel in dat het hele bewijs en alles waarop het voortbouwt *geformaliseerd* moeten worden zodat alle informatie beschikbaar is voor het bewijsprogramma.

De methode die we hiervoor gaan gebruiken is gebaseerd op *sequenten* en *reductieregels* waarmee sequenten tot *semantische tableaus*[1] kunnen worden gereduceerd. Deze methode is in het college *beweren en bewijzen* uitvoerig aan de orde gekomen en staat duidelijk uitgelegd in hoofdstuk 3 van het boek[2]. Hoewel we de (mogelijk pas net behandelde) theorie als bekend veronderstellen kan het toch nuttig zijn om het betreffende hoofdstuk bij de hand te houden. Voor wie het boek niet meer in z'n bezit heeft (gehad) hebben we op Blackboard een deel van het dictaat *Inleiding Logica* van Jan Jaspars neergezet, waarin in hoofdstuk 3 semantische tableaus worden uitgelegd.

2 Leerdoelen

Na afloop van deze opdracht ben je in staat om:

- parsers te schrijven voor (eenvoudige) niet-natuurlijke talen, bijvoorbeeld beschreven door (BNF) grammatica's.
- boomstructuren en bijbehorende operaties te ontwerpen, voor bijvoorbeeld logische proposities, en deze te implementeren.
- geautomatiseerde afleidingen/redeneringen/berekeningen te doen met logische proposities.

3 Instructie

Bestudeer allereerst de sheets over *parsers* zoals die gebruikt zijn op het college en blader hoofdstuk 3 van het boek voor beweren en bewijzen/dictaat of Blackboard nog eens door. Begin pas daarna aan de volgende deelopdrachten. Zorg ervoor, dat je (op papier en aan de hand van een of meer plaatjes) al een ontwerp van je datastructuur en afleidingsalgoritmen hebt gemaakt, vóórdat je aanschuift achter de pc om je code in te voeren en uit te testen.

4 Probleemschets

In deze opdracht ga je een programma schrijven waarmee je gevolgtrekkingen in de propositielogica automatisch kunt controleren. Voor het controleren van bewijzen of stellingen heb je een formalisme/taal nodig waarmee zowel de gebruikers als het bewijsprogramma kunnen werken. Het eerste deel van deze opgave bestaat dan ook uit het inlezen van sequenten. Het tweede deel van deze

[1] Zie ook `http://nl.wikipedia.org/wiki/Semantisch_tableau/`.

[2] J.F.A.K. van Benthem, H.P. van Ditmarsch, J.Ketting, W.P.M.Meyer-Viol. *Logica voor informatici.* Addison-Wesley, 1993.

opgave bestaat uit het controleren van een ingelezen sequent. Dit kan door het reduceren van het sequent tot een semantisch tableau waaruit nagenoeg direct de (on)geldigheid volgt.

Deelopdracht 1a

Het gebruik van een formalisme vereist dat je een (eenvoudige) taal ontwerpt waarin je sequenten kunt opschrijven. Geef een definitie van deze taal in de vormt van een formele syntaxbeschrijving, bijvoorbeeld als *BNF*-grammatica[3].

Deelopdracht 1b

Om sequenten in je bewijsprogramma te krijgen, dien je een *parser* te implementeren die een sequent als tekst inleest en omzet in een geschikte datastructuur (de *syntaxboom*). De syntaxboom is een verzameling van Java-klassen die vrijwel rechtstreeks zal volgen uit de gekozen grammatica.

Deelopdracht 2

Het controleren van de geldigheid van een sequent kan op verschillende manieren. Welke methode je ook kiest, de basis hiervoor wordt gevormd door de reductieregels op pagina 37 en 38 van het boek voor beweren en bewijzen (of pagina 48 en 49 van het dictaat).

Je kunt eerst expliciet een sequent reduceren tot een semantisch tableau. Dit houdt o.a. in dat je geschikt datatype om semantische tableaus te representeren moet bedenken. Hier ligt het voor de hand om een of andere boomstructuur te introduceren. Daarna doorloop je het tableau om te bepalen of het gegeven sequent geldig is of niet. Echter, als je het probleem wat beter analyseert zul je tot de conclusie komen dat voor het bepalen van de geldigheid het niet nodig is om een boom van sequenten op te bouwen. Je kunt volstaan met het bijhouden van de formules die links en rechts in een sequent voorkomen.

Indien een sequent ongeldig is dien je een tegenvoorbeeld te construeren. Probeer in ieder geval de volgende sequenten uit: $p \vee q \vdash p, q$ en $p, q \vdash p \wedge q$ en $p \rightarrow (q \rightarrow r) \vdash (p \rightarrow q) \rightarrow r$.

5 Producten

Als producten heb je je uitgeteste Java-code, van zowel onderdeel 1, de parser (neem de gekozen grammatica op als commentaar in je parser-klasse) als onderdeel 2, de evaluator. Zorg er zoals altijd voor dat je programma voldoet aan de gevraagde specificaties en aan de kwaliteitscriteria zoals die gesteld worden.

6 Inleveren van je producten

Vóór maandag 21 maart, 8.30 uur, en wel door de uiteindelijke inhoud van je eigen Java-bestanden op tijd als assignment in te leveren via Blackboard (als *losse* attached files). Plaats in de 'Comment'-box, bij dat inleveren via Blackboard, de namen en studentnummers maar ook jullie studierichting van zowel jezelf als je practicumpartner en doe dat ook in alle meegestuurde bestanden.

[3]Extended BNF [ISO/IEC 14977:1996(E)]: `http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf`