Informatica — Universiteit van Amsterdam

Bachelor Informatica

Universiteit van Amsterdam

De Gebarende Wijs: Nederlandse zinnen met persoonlijk voornaamwoorden omzetten naar Nederlandse Gebarentaal

Nardi Lam,
10453555

8 juni 2018

Begeleider: Floris Roelofsen

Samenvatting

Tijdens dit project wordt onderzocht in hoeverre het Grammatical Framework (GF) kan dienen als basis voor een vertaalprogramma tussen een fragment van het Nederlands en de Nederlandse Gebarentaal (NGT). Er zijn voor fragmenten van deze talen formele grammatica's geschreven volgens het *multiple context-free grammar*-formalisme. Daaruit zijn programma's geïmplementeerd met behulp van het GF om tussen deze talen vertalingen uit te voeren.

Uit de evaluatie van deze programma's blijkt dat correcte inputzinnen correct vertaald worden, alleen dat de bestaande grammatica's overgenererend zijn. Daarnaast schieten de huidige implementaties vooral tekort in de reikwijdte van het fragment van de brontalen dat ze implementeren.

Het is niet duidelijk of het GF buiten de geïmplementeerde zinsvormen ook geschikt is om een vertaling van Nederlands naar NGT uit te voeren. Dit is omdat enkele ontbrekende constructies moeilijk te implementeren zijn door middel van formele grammaticaregels.

Er zijn wel andere aspecten van beide talen die goed binnen het GF geïmplementeerd zouden kunnen worden. Het is echter belangrijk dat een vertaalprogramma een goede dekking van de brontaal heeft en daarom is om te bepalen of het GF hiervoor geschikt is verder onderzoek nodig.

De vertaalprogramma's zijn online te testen via `http://nardilam.nl/ngtpv`.

Inhoudsopgave

HOOFDSTUK 1

Inleiding

De Nederlandse Gebarentaal (NGT) is een gebarentaal die in Nederland gebruikt wordt door dove en slechthorende mensen om (effectiever) te communiceren. Er zijn wereldwijd veel verschillende gebarentalen en de toepassing hiervan is pas recentelijk algemeen geaccepteerd (in Nederland vanaf de jaren '80) [1].

Deze talen zijn vaak ontstaan uit een bewust ontworpen basis die door sprekers ervan eigen is gemaakt, uitgebreid, en aangepast. Inmiddels zijn er mensen die vanaf hun geboorte zijn opgevoed met de NGT en deze als moedertaal spreken. Deze mensen hebben daardoor een sterke intuitie voor hoe iets in de NGT gezegd hoort te worden. Dit maakt de NGT een volwaardige taal, met een eigen lexicon (woordenschat), en bovenal een eigen grammatica.

In tegenstelling tot het Nederlands, bijvoorbeeld, is deze grammatica echter niet nauwkeurig genoeg onderzocht om er een sluitende beschrijving van te geven. Momenteel wordt hiernaar op de UvA onderzoek verricht onder het Europese project *SIGN-HUB*[1].

[1] Informatie hierover is te vinden op: http://www.uva.nl/disciplines/gebarentaal/onderzoek/sign-hub/sign-hub.html.

1.1 Onderzoekscontext

De voornaamste bronnen die beschikbaar zijn met informatie over de NGT zijn woordenboeken [2] en lesmateriaal voor scholen. In tegenstelling tot het merendeel van de gesproken talen, zijn er vrijwel geen hulpmiddelen voor geautomatiseerde vertaling tussen gebarentalen en gesproken talen. Dit maakt het lastig voor iemand die geen gebarentaal spreekt om met een spreker van een gebarentaal te communiceren, bijvoorbeeld als ouders van een kind dat doof of slechthorend is en gebarentaal is onderwezen. Daarnaast is voldoende blootstelling aan een taal op jonge leeftijd erg belangrijk voor de taalverwerving. Daarom zouden meer hulpmiddelen op het gebied van NGT erg nuttig zijn, vooral bij het opvoeden van dove kinderen die in een voornamelijk horende omgeving opgroeien.

Een mogelijke oorzaak voor het uitblijven van deze hulpmiddelen is, naast het ontbreken van gestandaardiseerde grammatica's, het visuele aspect van gebarentalen. Om gebarentaal naar gesproken taal te vertalen moet een video hiervan geanalyseerd worden, en daarnaast moet ook een vertaling naar gebarentaal toe op een uitgebreide manier gevisualiseerd worden om effectief te zijn.

Er zijn verschillende manieren bedacht om een taal als NGT toch op papier vast te leggen, maar welke het meest gepast is hangt erg van van de doelgroep en het niveau van detail dat vereist is (zie figuur 1.1 en 1.2).

Figuur 1.1: Omdat NGT een visuele taal is, is er geen ideale manier om gebaren weer te geven in een tekstueel formaat. In het woordenboek [2] wordt het gebaar TAAL als volgt weergegeven. Hier is te zien dat de vorm en de plek van de hand veranderen, en ook de positie van de mond speelt een rol. Sommige gebaren zijn echter te complex om nauwkeurig in één afbeelding te vatten.

Figuur 1.2: Een andere methode om NGT vast te leggen, is om een schrift te ontwikkelen dat alle informatie over een gebaar bevat. Een voorbeeld hiervan is de KOMVA-notatie. Dit is het gebaar voor TAAL in die notatie. Deze notatie legt wel precies de vorm van een gebaar vast, maar is op zijn beurt erg moeilijk intuitief te begrijpen.

Gegeven een toepasselijke notatie is er echter genoeg bekend over de grammatica van de NGT om een opzet voor een vertaalsysteem te proberen te maken. In dit project wordt een poging gedaan om dit te doen voor een fragment van de Nederlandse taal, beginnend bij simpele, mededelende zinnen. Later worden ook zinnen toegevoegd waarin persoonlijk voornaamwoorden gebruikt worden. De werking van persoonlijk voornaamwoorden verschilt in het algemeen erg tussen gesproken talen en gebarentalen, wat het interessant maakt om vanuit een computationeel opzicht te onderzoeken wat nodig is om deze twee aan elkaar te relateren.

Het computationele framework dat toegepast wordt is het bestaande *Grammatical Framework* (GF) [3]. Dit is een framework voor het structureren van taalkundige informatie op basis van formele grammatica's. Daarnaast is het ook in staat om vertalingen te genereren, vanuit deze formele representatie. Tijdens deze scriptie wordt onderzocht in hoeverre het mogelijk is om op basis van het GF een programma te maken dat Nederlandse zinnen met persoonlijk voornaamwoorden kan vertalen naar een equivalent in de NGT.

1.2 Onderzoeksvraag

In sectie 3.1.1 en verder zal een formele grammatica voor een beperkt fragment van het Nederlands gedefinieërd worden, die aangeeft wat voor zinnen dit programma precies moet kunnen vertalen. Voor de duidelijkheid wordt de taal die deze grammatica genereert in het vervolg **NLpv** genoemd. Omdat de NGT een visuele taal is, zal er een tekstuele representatie ontworpen worden waar deze zinnen naartoe vertaald worden. De definitie van deze representatie, **NGTg** genaamd, wordt gegeven in sectie 3.1.2. De taal die we daarmee willen vormen, een vertaling van de inputtaal NLpv, noemen we **NGTpv**. Een grammatica voor deze taal wordt ook in de loop van het project gedefinieerd.

Daarmee wordt de concrete onderzoeksvraag die in dit project behandeld wordt: *in hoeverre kan het GF dienen als framework voor een vertaalprogramma van de taal NLpv naar NGTpv?* De hypothese is dat

het een goede basis biedt voor de grammaticale aspecten van deze vertaling, maar dat er ook belangrijke semantische aspecten zijn die niet in het GF behandeld kunnen worden. Deze aspecten ontstaan vanwege de verschillen in de manier van verwijzen in gebarentalen ten opzichte van gesproken talen.

HOOFDSTUK 2

Achtergrond

2.1 Natuurlijke talen en formele grammatica's

Natuurlijke talen worden in de taalkunde beschreven op verschillende niveaus: op het niveau van woorden, op het niveau van zinnen en op het niveau van teksten of dialogen. Het gebied dat gaat over de vorm van worden heet morfologie, en de analyse van de structuur van dialogen en teksten wordt ook wel discourse analysis genoemd. Op het zinsniveau zijn twee belangrijke aspecten van toepassing: de syntaxis en de semantiek. De syntaxis bestudeert de letterlijke structuur van zinnen: welke woorden kunnen elkaar opvolgen om tot een geldige zin te komen en welke niet? De semantiek bestudeert de betekenis van zinnen: gegeven een aantal woorden die elkaar opvolgen, welke betekenis vormen ze samen[1]?

Het aspect van taal wat in dit project behandeld wordt is hoofdzakelijk de **grammatica**. De grammatica van een taal is een set regels die specificeren hoe zinnen in een taal syntactisch gevormd kunnen worden.

Er zijn twee belangrijke varianten waarin over grammatica gedacht wordt: de zogenaamde ***constituency grammars*** (**constituentiegrammatica's**) en ***dependency grammars*** (**dependentiegrammatica's**). In *constituency grammars* bestaat een zin uit **woordgroepen** of **constituenten**, die gevormd worden door andere woordgroepen of woorden te combineren, en waarvan slecht bepaalde combinaties geldige zinnen opleveren. De grammatica specificeert daarom regels die omschrijven welke woorden tot welke groepen behoren, en hoe vanuit deze groepen nieuwe groepen gevormd kunnen worden. In de *dependency grammars* staan niet (abstracte) woordgroepen maar woorden centraal, en heeft iedere zin één woord als ***head*** (**hoofd**). Meestal is dit de persoonsvorm. Verder is ieder woord **afhankelijk** van een ander woord, en zijn verschillende afhankelijkheidsrelaties mogelijk.

Een voordeel van *dependency grammars* is dat ze flexibeler zijn: ze

[1] Belangrijk is dat dit de betekenis is op zinsniveau. Dit staat los van de betekenis van woorden op zich, wat bijvoorbeeld blijkt door op te merken dat een zin met dezelfde woorden afhankelijk van de volgorde verschillende betekenissen kan hebben. Ook heeft bijvoorbeeld een woord als *stoel* op zichzelf al betekenis, maar veel andere woorden hebben dit niet, of hebben een betekenis die verschilt afhankelijk van de context.

kunnen op zinsbasis worden gevormd door de relaties tussen zinnen te beschrijven, zonder voor de hele taal een omvattende grammatica op te hoeven stellen. Daarom zijn ze uitermate geschikt voor het aanbrengen van enige structuur in databases van zinnen, zogenaamde *treebanks*. Ook kunnen hiermee efficiënte parsers op basis van machine learning-algoritmen getraind worden. Deze flexibiliteit kan echter ook een nadeel zijn: omdat er geen 'globale' grammatica bestaat, is het moeilijk om te bepalen of een zin tot een bepaalde taal behoort[2]. Daarom zijn ze ongeschikt voor het bewerkstellingen van vertaalmechanismen, in zoverre dat deze geen garantie voor de grammaticale juistheid van de output kan geven. Daarom wordt in dit project gewerkt met *constituency grammars*, waarvoor het Grammatical Framework een basis levert.

[2] Oftewel: *dependency grammars* beschrijven, terwijl *constituency grammars* voorschrijven.

De twee belangrijkste woordgroepen die nodig zijn in de grammatica's die wij opstellen, zijn als volgt:

1. NP (N): *noun phrase* (naamwoordgroep). Een woordgroep die *iets* (een object, persoon of concept) beschrijft. Het vervult dezelfde functie als een zelfstandig naamwoord, maar kan arbitrair ingewikkelde constructies bevatten[3].

[3] Denk aan iets als: "De man die de vrouw zag die gisteren in de winkel een hond bij zich had die..."

2. VP (VC): *verb phrase* (verbale constituent). Een woordgroep die de persoonsvorm van de zin bevat, en alle objecten die er aan verbonden zijn, zoals het lijdend voorwerp (direct object) en meewerkend voorwerp (indirect object). Daarnaast kunnen er ook meerdere werkwoorden in aanwezig zijn (bijvoorbeeld hulpwerkwoorden).

3. S (Z): *sentence* (zin). De groep die volledige zinnen bevat. Dit is over het algemeen de (begin- of) eindgroep van onze grammatica's: we willen volledige zinnen ontleden (of genereren).

Een grammatica bestaat over het algemeen uit **symbolen** en **productieregels**. Er zijn twee soorten symbolen, **terminale symbolen** en **non-terminale symbolen**. Terminale symbolen zijn letterlijke woorden (of woordgroepen), deze vormen uiteindelijk de taal (bijvoorbeeld: `'huis'`). In het algemeen noemen we een reeks terminale symbolen ook wel een **string**. Non-terminale symbolen zijn symbolen zoals S, NP en VP die een abstracte woordgroep voorstellen, en daarmee alleen dienen om de structuur van de zin correct te bepalen. De productieregels geven aan hoe vanuit een bepaald **beginsymbool** er een string terminale symbolen gevormd kan worden. Deze regels worden als volgt genoteerd:

1 $S \rightarrow NP\ VP$

2 $NP \rightarrow$ `'de jongen'`

3 $VP \rightarrow$ `'pakt de bal'`

Dit voorbeeld betekent dat een zin S bestaat uit een *noun phrase* NP en een *verb phrase* VP. Daarnaast is er slechts één NP en één VP: respectievelijk 'de jongen' en 'pakt de bal'.

Omdat we door het toepassen van deze regels vanuit S de string 'de jongen pakt de bal' kunnen maken[4], betekent dat dat deze zin in de **taal**[5] van deze grammatica zit. Belangrijk is dat formeel een grammatica dus een taal *voorschrijft*, niet andersom[6]. Om voor een gegeven zin te bepalen of deze tot de taal van een bepaalde grammatica behoort is dus niet triviaal, en afhankelijk van de eigenschappen van de grammatica kan de complexiteit van dat probleem erg variëren.

[4] Achtereenvolgend regel 1, 2 en 3 toepassen op S geeft 'de jongen pakt de bal'.

[5] De taal die een grammatica beschrijft is de verzameling van alle terminale symbolenreeksen die verkregen kunnen worden door bij een bepaald symbool te beginnen (in dit geval S) en de productieregels toe te passen.

[6] Het is niet zo dat een grammatica een taal *beschrijft*, zoals we gewoonlijk over natuurlijke taal denken.

2.1.1 Context in grammatica's en taal

Een belangrijke categorie van grammatica's zijn de **contextvrije grammatica's**. Dit zijn grammatica's met regels zoals hierboven beschreven, waar de beperking geldt dat aan de linkerkant van een regel slechts één non-terminaal symbool mag staan (verder niets). Dit kan geïnterpreteerd worden als het ontbreken van context: een symbool S kan altijd vervangen worden als er een regel '$S \rightarrow$ ... ' is, ongeacht waar deze gevonden wordt. Door deze beperking is het voor contextvrije grammatica's mogelijk om efficient (met polynomiale tijdscomplexiteit) te bepalen of een string zich in de taal bevindt, wat nodig is om een zin te kunnen ontleden.

Contextvrije grammatica's schieten echter in sommige situaties tekort om natuurlijke talen te beschrijven. Één reden hiervoor is dat gerelateerde woordgroepen altijd naast elkaar geplaatst moeten worden om door een contextvrije regel gegenereerd te kunnen worden. Dit kunnen we visualiseren door in een zin de woorden die een directe relatie tot elkaar hebben met haakjes te omgeven. Een voorbeeld is

$(^1$ $(^2$ Jan $)^2$ $(^3$ liep $(^4$ zonder te stoppen $)^4$ $(^5$ naar de supermarkt $)^5$ $)^3$ $)^1$

Hier zijn indices gebruikt om aan te geven welke haakjes bij elkaar horen. Het is te zien dat in deze zin alle haakjes correct genest zijn, wat betekent dat hij uit een contextvrije taal kan komen.

Een voorbeeld waar dit niet mogelijk is is als volgt:

...dat $(^1$ Jan $(^2$ Piet $(^3$ Maartje zag $)^1$ helpen $^2)$ zwemmen $^3)$

Hier gebeuren drie dingen tegelijk: Maartje zwom, Piet hielp haar, en Jan zag dat. Wanneer we deze echter in een constructie als deze zetten, lopen de relaties tussen de werkwoorden en degene die ze verricht door elkaar heen, te zien aan de indices. Dit zijn zogenaamde **cross-serial dependencies**, en deze zijn niet contextvrij[7]. Daarom is het verstandig om voor een programma dat natuurlijke taal moet verwerken een sterker formalisme toe te passen, dat ook kan omgaan met niet-contextvrije grammatica's, oftewel contextgevoelige grammatica's. Er zijn hier een aantal

[7] Zo een constructie is niet in alle talen mogelijk. Neem bijvoorbeeld de vertaling van het voorbeeld naar het Engels:
...that $(^1$ Jan saw $(^2$ Piet help $(^3$ Maartje swim $)^3$ $)^2$ $)^1$
Hier lopen de haakjes niet door elkaar, en deze zin is dus wel contextvrij.

alternatieven voor die ook praktisch inzetbaar zijn, en één hiervan wordt in de volgende paragraaf besproken.

2.1.2 Contextgevoelige grammatica's

Een variant op de contextvrije grammatica's die hiervoor ontworpen zijn zijn de ***multiple context free grammars*** of MCFG's [4]. Het idee hierachter is dat de regels hetzelfde zijn als voor contextvrije grammatica's wat non-terminale symbolen betreft, maar dat een non-terminaal symbool nu een string terminale symbolen bevat. We noemen dit de **inhoud** van een symbool. Een regel als $NP \rightarrow$ `'de jongen'` wordt dan

$$NP(\texttt{'de jongen'}).$$

Dit is een regel die stelt dat een NP-symbool dat de string 'de jongen' bevat een geldig symbool is.

Regels die non-terminale symbolen produceren, worden omgedraaid. In plaats daarvan geven non-terminale symbolen hun inhoud aan elkaar door. Een regel als $S \rightarrow NP\ VP$ wordt daarom:

$$S(uv) \leftarrow NP(u)\ \ VP(v).$$

Dit betekent dat, gegeven een NP met inhoud u, en een VP met inhoud v, er een geldige S bestaat met inhoud u gevolgd door v.

Tot hier is dit alleen een andere notatie voor het beschrijven van contextvrije grammatica's. Wat MCFG's echter ook mogelijk maken, is om een non-terminaal symbool méér dan één string als inhoud te geven (vandaar *multiple*). Op deze manier kunnen ook sommige niet-contextvrije grammatica's worden gedefinieërd. Een voorbeeld is de volgende grammatica:

$$A(\texttt{'a'}, \texttt{'b'}, \texttt{'c'})$$
$$A(\texttt{'a'}u, \texttt{'b'}v, \texttt{'c'}w) \leftarrow A(u, v, w)$$
$$S(uvw) \leftarrow A(u, v, w)$$

Hier is S het startsymbool, of in deze notatie toepasselijker het **eindsymbool**. Deze grammatica kan de taal $a^n b^n c^n$ voor $n \geq 1$ genereren, wat een abstract voorbeeld is van een taal die niet door een contextvrije grammatica gegenereerd kan worden. Een MCFG kan dit wel, doordat het symbool A zowel de a's, b's, als c's los van elkaar bevat, en er alleen een regel is om ze alle drie in één keer uit te breiden. Daardoor blijft het aantal opeenvolgende letters n altijd gelijk. Als laatste stap worden de a's, b's en c's samengevoegd tot één string, wanneer A naar S wordt omgezet.

In het geval van MCFG's hebben we dus niet alleen regels die 'de andere kant op' werken, maar zijn ze ook tot meer in staat: de verschillende inhoudelijke componenten (omsloten door non-terminale symbolen) kunnen wél of niet worden samengevoegd, en in een willekeurige volgorde.

Daarom bestaat een regel nu uit twee delen. Het eerste deel gaat slechts over welke non-terminale symbolen in de regel voorkomen, dit noemen we het **inferentiedeel**. Dit betekent dat bijvoorbeeld

$$S \leftarrow A$$

aangeeft dat een S afgeleid (of geïnfereerd) kan worden uit een A. Hóé de inhoud van de A omgezet wordt tot een inhoud voor de S wordt hier niet genoemd. Dit deel (wat tussen haakjes komt te staan) noemen we vanaf nu het **transformationele** deel. Bijvoorbeeld

$$S(uvw) \leftarrow A(u, v, w)$$

geeft aan dat de strings u, v en w uit A achter elkaar geplakt moeten worden om de inhoud van S te genereren. In het geval waar links en rechts slechts één symbool staat, spreken we af dat weglaten van het transformationele deel zoals in

$$S \leftarrow T$$

hetzelfde betekent als:

$$S(x) \leftarrow T(x).$$

2.2 Structuur van de NGT

2.2.1 Algemene syntactische eigenschappen

Zoals in de inleiding genoemd, is de NGT een visuele taal. Hij wordt geuit door het lichaam te bewegen, waarbij voornamelijk gebaren met de handen gemaakt worden, maar waarbij ook de verdere houding van het lichaam een betekenis heeft. Sommige aspecten van de lichaamshouding zullen vergelijkbaar zijn met iets als intonatie in gesproken talen, maar bepaalde componenten zijn wel essentieel voor de betekenis van een zin, en komen ook op een regelmatige manier voor. Er kunnen dus meerdere betekenisvolle handelingen tegelijkertijd gemaakt worden die met elkaar gecöordineerd zijn om een bepaalde zin te vormen. Dit noemen we **simultaneïteit**, en is vooral erg belangrijk in gebarentalen[8] [2].

De voornaamste vormen waarin simultaneïteit voorkomt in de NGT zijn in **non-manuele modificeerders** en in **ruimtegebruik**. Non-manuele modificeerders zijn de handelingen die gecombineerd met een manueel (met de handen) gebaar extra informatie verschaffen. Deze worden in dit project niet behandeld en daarom zullen we hier verder niet op ingaan.

Ruimtegebruik is wél een belangrijk aspect van de NGT voor dit project. De ruimte rondom de gebaarder wordt in de NGT gebruikt om gebaren in te plaatsen, zodat de relaties tussen gebaren verduidelijkt kunnen

[8] Een voorbeeld van simultaneïteit in gesproken talen zijn toontalen (zoals het Thai), waar niet alleen de klank maar ook de intonatie de betekenis van een woord verandert. Dit is een andere functie dan intonatie heeft in het Nederlands, want hier kan een andere intonatie een totaal ander woord vormen.

worden. Dit kan een (letterlijke) plaatselijke relatie zijn ('de bal ligt op de kast'), maar ook een manier om entitieten te onderscheiden ('die bal' vs. 'deze bal').

Wat het betekent om het te hebben over de syntax van de NGT, ligt dus voor een groot deel in de mate van simultaneïteit waar we rekening mee houden in onze representatie, en hoe we die vastleggen. In het geval van dit project worden weinig simultane aspecten van de NGT behandeld, en daarom behandelen we de taal voornamelijk als een reeks sequentieel opvolgende gebaren. Wel verwerken we het ruimtegebruik hierin, voorzover dat nodig is, als een annotatie dat op een gebaar 'geplakt' wordt om aan te geven dat het locatiegevoelig is.

De syntax voor manuele gebaren kan als volgt beschreven worden:

- De NGT heeft in de basis een SOV-volgorde: dit betekent dat de naamwoorden (subject/onderwerp, object/lijdend of meewerkend voorwerp) vooraan in de zin staan, en de werkwoorden achteraan. Dit verschilt met de SVO-volgorde van het Nederlands: in de zin 'ik gooi de bal' is de volgorde onderwerp, dan het werkwoord, gevolgd door het lijdend voorwerp.
- Er is geen duidelijke vervorming van gebaren die woordgroepen onderscheidt: werkwoorden worden niet vervoegd behalve als dit nodig is, en ook bijvoeglijk naamwoorden veranderen niet in vorm afhankelijk van het naamwoord waar ze op slaan.
- Er zijn ook geen lidwoorden.
- De vorige twee punten maken dat er geen zichtbaar verschil is in het 'geslacht' van woorden, zoals dat in het Nederlands er wel is (mannelijk/vrouwelijk/onzijdig).

Over het algemeen worden gebarenzinnen in tekst weergegeven door middel van **glossen**. Hierbij wordt aan ieder gebaar een symbool toegekend (een glos), zodat de volgorde van gebaren genoteerd kan worden. Bijvoorbeeld: voor het begrip 'jongen' bestaat een bepaald gebaar, waar door middel van een glos JONGEN uniek naar verwezen kan worden. Zo is een voorbeeldzin in de NGT:

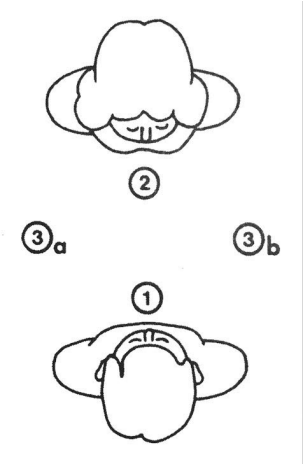
JONGEN BAL PAKKEN

Hier worden drie gebaren gebruikt, JONGEN, BAL en PAKKEN om de informatie van 'de jongen pakt de bal' over te brengen.

2.2.2 Persoonlijk voornaamwoorden en verwijzingen

Om in de NGT naar personen of objecten te verwijzen, wordt in plaats van verwijswoorden over het algemeen gebruik gemaakt van de plek van gebaren in de ruimte rondom de gebaarder.

Om deze ruimte iets concreter te bespreken geven we namen aan de verschillende plekken waar een persoon of object zich kan bevinden. Twee belangrijke plekken zijn plek 1 en 2, deze verwijzen respectievelijk naar de gebaarder zelf (de eerste persoon) en degene tegen wie de gebaarder praat (de tweede persoon). Daarnaast kunnen willekeurige naamwoordelijke gebaren (derde personen) op een andere, vanaf de gebaarder schuin georiënteerde plek gelocaliseerd worden.



Figuur 2.1: Een weergave van de locaties in de gebarenruimte. De gebaarder (locatie 1) staat rechts tegenover degene aan wie de gebaren gericht zijn (locatie 2), en rechts en links van diegene bevinden zich locaties 3a en 3b. Overige locaties (3c, 3d enz.) hebben geen vaste plek, maar dienen wel uniek te zijn. Uit [5].

Deze plekken worden aangegeven door het cijfer 3 gevolgd door een letter: 3a, 3b, 3c enzovoort. De meest gebruikte locaties, 3a en 3b, bevinden zich rechts en links van de tweede persoon. Wanneer een gebaar niet gelocaliseerd wordt, plaatst de gebaarder het tussen zichzelf en de tweede persoon: dit wordt de neutrale ruimte genoemd.

In plaats van verwijswoorden te gebruiken, kan een spreker van NGT een gebaar op een plek in de ruimte **localiseren**. Daarna kan deze letterlijk wijzen naar de locatie van een eerder gemaakt gebaar om ernaar te verwijzen. Dit gebeurt door middel van een indexgebaar, genoteerd door middel van IX$_{\dots}$. Een indexgebaar als IX$_{3a}$ betekent: wijzen met de indexvinger van de dominante hand naar locatie 3a.

Een zin waarin in het Nederlands een persoonlijk voornaamwoord gebruikt zou worden, kan door middel van indexgebaren geuit worden. Neem de zin ‘de jongen pakt de bal en hij gooit’. Na het voegwoord ‘en’ wordt ‘hij’ gebruikt om naar ‘de jongen’ te verwijzen, om herhaling en verbositeit te verminderen. In de NGT zou dit gedaan kunnen worden door het gebaar behorende bij ‘de jongen’ bij de eerste uiting te localiseren, bijvoorbeeld op locatie 3a, en daarna met een indexgebaar IX$_{3\text{A}}$ er naar terug te verwijzen. Het gelocaliseerde gebaar waarnaar verwezen wordt, wordt de **referent** genoemd. Op deze manier nemen de indexgebaren dus de functie van persoonlijk voornaamwoorden en overige verwijswoorden (die, dat) over.

Twee belangrijke verschillen ontstaan hierdoor met het Nederlands, en met gesproken talen in het algemeen. Omdat er geen aparte verwijswoorden zijn voor bijvoorbeeld geslacht (zij of hij) hoeft hier ook geen rekening mee te worden gehouden tijdens het verwijzen. Alleen de locatie van de referent doet ertoe, maar dit wordt door de gebaarder bepaald en is geen vaste eigenschap van het gebaar zelf, in tegenstelling tot het geslacht van een woord in het Nederlands. Dit geeft een uitdaging bij het vertalen, want er is dus geen vertaling voor het woord ‘zij’, zonder te weten waar dat woord naar verwijst.

Een voordeel van deze manier van verwijzen is dat minder makkelijk ambiguïteiten ontstaan. In het Nederlands is het mogelijk om met een woord als ‘hij’ of ‘die’ een onduidelijke verwijzing te construeren, als er aanvoorafgaand meerdere mogelijke referenten zijn. In de NGT is dit

niet mogelijk, want er is altijd precies één referent waar een gebaar IX$_3$A naar verwijst: dat wat eerder op 3a gelocaliseerd is. De enige andere mogelijkheid is dat er niets daar gelocaliseerd is en de verwijzing ongeldig is. Dit vormt echter een uitdaging voor de vertaling van Nederlands naar NGT, omdat deze ambiguïteiten dus moeten worden 'opgelost'. Dit is een probleem dat buiten deze scriptie valt, en de gekozen oplossing hiervoor wordt later besproken.

2.3 Verwijzingen in het Nederlands

Voor het correct uiten van verwijzingen in de NGT, is het van belang dat bekend is op welke locatie de referent zich bevindt, en daarmee is het ook nodig om te weten wát de referent is die bij een bepaalde verwijzing hoort. Om een zin te kunnen vertalen moet dus van iedere verwijzing in de bron duidelijk zijn wat de bijbehorende referent is. In Nederlandse zinnen is het echter mogelijk dat er ambigue situaties ontstaan. Een (flauw) voorbeeld is een incorrecte verwijzing als

> Ik zag Jan en Piet lopen en toen struikelde hij.

Het is hier niet te zeggen naar wie 'hij' verwijst. Een subtieler voorbeeld is echter een zin als de volgende:

> Jan gaf Piet de bal en hij gooide hem.

Een spreker van de taal zou deze zin eenduidig interpreteren, 'hij gooide hem' betekent: 'Piet gooide de bal'. Dat dit de juiste interpretatie is volgt echter niet uit de grammaticale structuur van de zin. De betekenis van de zin is hier essentieel. Om tot de meest logische interpretatie te komen, is een zekere **wereldkennis** nodig:

1. 'gooien' is een werkwoord dat alleen door een persoon uitgevoerd kan worden. Het onderwerp van de tweede deelzin is dus 'Jan' of 'Piet'.
2. Het is waarschijnlijk dat het lijdend voorwerp in de tweede deelzin 'de bal' is, want het is heel normaal om een bal te gooien, maar heel vreemd om een ander mens te gooien.
3. In het eerste zinsdeel geeft Jan de bal aan Piet. Dit betekent dat Piet in bezit van de bal is, en dus alleen hij hem kan gooien. Daarom moet 'Piet' het onderwerp van de tweede deelzin zijn.

Op deze manier is tot bovenstaande interpretatie te komen, maar dit staat los van de grammatica. Immers is

> Jan gaf Piet de bal en de bal gooide Jan.

ook een grammaticaal correcte zin.

Dit betekent dat het koppelen van verwijswoorden aan referenten een semantisch vraagstuk is dat buiten de mogelijkheden van een grammaticaal vertaalproces ligt. Dit probleem wordt in de literatuur **coreference resolution** genoemd, en er zijn algoritmes voor ontworpen die dit voor Nederlandse tekst proberen op te lossen[9]. Daarom gaan we er van uit dat de Nederlandse input eerst met zo een algoritme geannoteerd is om de referenten en de verwijzingen aan elkaar te koppelen. Een meevaller is dat coreference resolution binnen de NGT wél regelmatig kan gebeuren, omdat er een vast 'register' is van referenten (de locaties in de ruimte). Daardoor heeft deze aanname geen effect op verdere uitbreiding van het project met betrekking tot andere aspecten van de NGT.

[9] Een voorbeeld hiervan is het COREA-project [6].

2.4 Het *Grammatical Framework*

2.4.1 *Grammars* en structuur van programma's

Het Grammatical Framework is een programmeertaal waarin grammatica's gedefinieerd worden die dezelfde mogelijkheden hebben als MCFG's. Daarnaast biedt het GF een aantal operaties die op deze grammatica's uitgevoerd kunnen worden. De belangrijkste twee zijn parsen en linearisatie. Parsen (via het commando 'parse') neemt een string als input en beslist of deze in de grammatica van een gegeven taal zit. Zo ja, dan geeft dit commando de grammaticaregels die toegepast worden om deze string te vormen als output. Deze regels worden geïmplementeerd als functies tussen non-terminale symbolen en hebben een unieke naam. Bijvoorbeeld:

```
fun mkS : NP -> VP -> S
```

Dit definieert een functie `mkS` die van een NP en een VP een S maakt, en dus als implementatie kan dienen voor de regel $S(uv) \leftarrow NP(u)\,VP(v)$. De regels die toegepast worden om een bepaalde zin te parsen kunnen dus worden weergegeven als een reeks functies die gecompositioneerd worden. Gegeven een grammatica waar de regels namen hebben, zoals

```
mkS : NP -> VP -> S = np ++ vp
jongenNP : NP = "de jongen"
paktBalVP : VP = "pakt de bal"
```

kan de string `"de jongen pakt de bal"` geparsed worden als

```
mkS jongenNP paktBalVP
```

wat gelezen dient te worden als: roep `mkS` aan met als argumenten de output van `jongenNP` en van `paktBalVP`. Dit noemen we ook wel de **parse tree** van de zin. Dit omdat het een boomstructuur heeft[10] en omdat het laat zien hoe een zin afgeleid kan worden uit de grammaticaregels en dus waarom het in de taal zit. Ook geeft het de structuur van de zin weer: we

[10] Functies kunnen andere functies als argumenten meekrijgen, en die weer andere functies, enzovoorts. Maar het werkt één kant op: een 'binnenste' functie weet niets van een 'buitenste' functie. We zeggen dat de buitenste functie een ouder is van de binnenste, en de argumenten van een functie zijn de kinderen. Ook is er uiteindelijk één functie die als laatste aangeroepen wordt: die komt bovenaan de boom en is de *root*.

zien hier duidelijk dat de zin bestaat uit een NP en een VP en waar deze over gaan.

Linearisatie (via het commando 'linearize') is het tegenovergestelde van parsen. Hier wordt vanuit een parse tree de bijbehorende zin in een bepaalde taal gegenereerd. Dit komt neer op het uitvoeren van de bijbehorende functies in de juiste volgorde, zoals hierboven besproken.

Met deze functionaliteit kan in het GF een MCFG direct geïmplementeerd worden. Hiervoor moet de overstep gemaakt worden van concepten in de theorie van grammatica's naar de constructen in het GF. Dit gaat als volgt:

- Non-terminaal symbool → `cat`
- Regel in abstracte grammatica (inferentiedeel) → `fun`
- Regel in concrete grammatica (inhoudelijk deel) → `lin`

Daarnaast wordt het **linearisatietype** van een non-terminaal symbool gespecificeerd via `lincat`. Het type komt over met de vorm van de inhoud. Voor een MCFG is dit een verzameling van één of meerdere strings.

2.4.2 Computationele eigenschappen

Zoals hierboven besproken, kunnen in het GF een equivalent van MCFG's geïmplementeerd worden [7]. Daarnaast zijn er een aantal extra mogelijkheden. Zo kan een MCFG alleen werken met één of meer strings per non-terminaal symbool. Het GF heeft een uitgebreider type-systeem. Naast het basistype `Str` voor strings is het ook mogelijk ***tuples*** te maken. Een tuple is een verzameling van een aantal objecten. Het type `Str * Str` bestaat bijvoorbeeld uit twee strings, zoals `<"a","b">`. Daarnaast zijn er ook ***records***. Een record is ook een combinatie van (een of) meerdere objecten, maar hier worden de objecten aangeduid met labels: het record `{ s = "a" }` heeft als type `{ s : Str }`. Hier is `s` een veld van het record. Deze tuples en records kunnen genest worden om zo verschillende datastructuren op te bouwen.

Daarnaast heeft het GF een tweede soort type, de zogenaamde **parametertypes**. Parametertypes zijn bijzonder omdat ze niet herleid worden tot een string, maar een keuze representeren. Om bijvoorbeeld het onderscheid te maken of een woord enkelvoud of meervoud is, kan een parametertype gebruikt worden:

```
param Aantal = Enkelvoud | Meervoud ;
lincat Woord = { s : Str ; a : Aantal } ;
```

Hier geeft het veld `a` op een `Woord` aan of het enkelvoud of meervoud is. Kenmerkend van een parametertype is dat het een vast, eindig aantal waarden heeft, dat bij compiletijd gedefiniëerd moet zijn.

Parametertypes worden nuttig omdat ze samen gebruikt kunnen worden met ***tables***. Kort gezegd zijn tables functies die alleen parametertypes als argumenten hebben. Een table kan bijvoorbeeld op basis van een parameter voor enkelvoud of meervoud, de juiste persoonsvorm kiezen dat bij een onderwerp hoort:

```
Lopen = table Aantal {
    Enkelvoud => "loopt" ;
    Meervoud => "lopen"
} ;
LoopZin = \ow -> ow.s ++ (Lopen ! ow.a) ;
```

Hier vult `Lopen ! ow.a` de waarde van `ow.a` in in de tabel voor `Lopen`.

Het idee achter parametertypes en tables is dat ze bij de compilatie uitgewist kunnen worden, omdat ze een beperkt aantal waarden en uitkomsten hebben. In dit geval kunnen er bijvoorbeeld twee regels gegenereerd worden voor `LoopZin`, een voor enkelvoud en een voor meervoud, en kan de categorie `Woord` in tweeën gesplitst worden, `Woord_e` voor woorden in het enkelvoud en `Woord_m` voor het meervoud. Zo ontstaat er een groter aantal regels en symbolen, maar is het gebruik van parameters wel terug te herleiden tot een MCFG. Toch is het wel belangrijk om op te letten in het gebruik van parametertypes, omdat het aantal geproduceerde regels hierdoor flink kan toenemen [7].

Daarnaast zijn in het GF ook mogelijkheden tot *logisch programmeren*. Dit betekent dat predicaten gevormd kunnen worden met behulp van het type-systeem. Er kunnen inferentieregels opgesteld worden (mogelijk zonder linearisatie) zodat de aanwezigheid of afwezigheid van een regel tussen types een logisch predicaat formaliseert. Hier wordt verder in dit project geen gebruik van gemaakt, dus daarom wordt er niet op ingegaan. Wel is van belang dat ook deze functionaliteit terug te herleiden is naar een implementatie in MCFG's (waarbij de gegenereerde grammatica mogelijk complexer is dan het programma in GF).

HOOFDSTUK 3

Theoretische uitwerking

3.1 Het vertaalsysteem

3.1.1 Beperkte grammatica voor het Nederlands

In deze paragraaf wordt een formele grammatica gedefinieërd voor de beperkte variant van het Nederlands die als input geaccepteerd wordt. Er wordt begonnen met een basistaal, NLb genoemd, die bestaat uit zinnen met een onderwerp en persoonsvorm, en mogelijk ook een lijdend voorwerp (als het werkwoord dit toestaat). Voor het onderwerp en het lijdend voorwerp accepteren we alleen zelfstandig naamwoorden, omdat deze (handmatig) in een beperkt lexicon opgenomen kunnen worden.

Om de grammatica voor NLb te schrijven gebruiken we de woordgroepen uit sectie 2.1, en een aantal extra woordgroepen die de woorden die gebruikt worden bevatten:

1. ZN: zelfstandig naamwoord. Een woord dat verwijst naar een object, persoon of concept, en dat in het woordenboek opgenomen kan worden (geen eigennamen). Voor het gemak nemen we het lidwoord (de/het) mee in de definitie. Bijvoorbeeld: “de jongen”, “het huis”.

2. W2: tweeplaatsig werkwoord. Ook wel transitief werkwoord genoemd. Dit is een werkwoord dat samen met een lijdend voorwerp in een verbale constituent geplaatst kan worden. Bijvoorbeeld: “gooien” (want: “Ik gooi de bal”) maar niet “slapen” (“Ik slaap de bal” kan niet).

3. W1: éénplaatsig werkwoord. Ook wel intransitief werkwoord genoemd. Dit is een werkwoord zónder lijdend voorwerp, dus wel “slapen”.

4. W: Dit is de categorie voor werkwoorden die zowel mét als zonder lijdend voorwerp gebruikt kunnen worden. Bijvoorbeeld: “koken” (“Ik kook” is prima, maar “Ik kook een maaltijd” ook) of “eten”.

Hiermee definiëren we de volgende grammatica voor NLb:

```
Z(ab) ← N(a) VC(b)
N ← ZN
VC ← W1
VC(ab) ← W2(a) N(b)
W1 ← W
W2 ← W
ZN('de jongen' | 'de bal')
W1((vervoeg 'lopen') | (vervoeg 'werken'))
W2((vervoeg 'pakken') | (vervoeg 'gooien'))
W(vervoeg 'koken')
```

Hier staat de speciale notatie (vervoeg `'...'`) voor de disjunctie van alle vervoegde vormen (in de tegenwoordige tijd) van het gegeven werkwoord. Bijvoorbeeld: (vervoeg `'lopen'`) dient vervangen te worden door (`'loop'` | `'loopt'` | `'lopen'`).

3.1.2 Abstracte representatie van de NGT

Omdat de NGT een visuele taal is, is er een symbolische representatie voor ontwikkeld, waar de vertaling naartoe kan plaatsvinden. Het doel van deze representatie is om de inhoud van gebaarde zinnen in NGT goed genoeg te kunnen omvatten, dat de uiting ervan in gebaren systematisch gevormd kan worden. Met andere woorden: als we een persoon aanwijzen als 'gebaarder', en deze een 'woordenboek' geven waarin de symbolen gekoppeld worden aan handelingen, moet deze persoon in staat zijn om zonder verdere kennis van de taal een symbolische zin correct te gebaren naar iemand die wél NGT spreekt.

Het concept *glossen* in de NGT is hier sterk aan verwant. Bij het glossen worden unieke gebaren aan unieke symbolen gekoppeld, maar is de vorm van een gebaar over het algemeen niet heel gedetailleerd beschreven. Glossen is dus en notatie die erg toepasselijk wanneer we p syntactisch niveau naar de taal kijken. In bestaande databases van gebarentaalconversaties zoals het RU-corpus[1], zijn deze glossen ook gebruikt om gesprekken te annoteren. Dit geeft daarom een goede basis voor een symbolische representatie voor de taal, aangezien deze ontworpen is om fragmenten NGT correct en volledig te beschrijven, en omdat het een duidelijke maatstaf geeft voor wat juiste output is en hoe deze geïnterpreteerd moet worden [8].

[1] Dit is de grootste verzameling data van (geannoteerde) NGT-conversaties, verzameld door de Radboud Universiteit (zie: https://www.ru.nl/corpusngt).

Om deze redenen noemen we het 'schrift' dat we gebruiken voor NGT **NGTg**, en definiëren dit als volgt:

$$\text{NGTg} = (G, L, M, g).$$

G is een verzameling glossen (tekstuele symbolen, we noemen dit ook het

symbolische lexicon van NGTg), L een verzameling van instructies die een bepaalde handbeweging, gezichtsuitdrukking of lichaamshouding voorschrijven (ook wel het **visuele lexicon van NGTg**), M een verzameling **modificeerders** die op gebaren toegepast kunnen worden, en g een functie $G_n \to L$ die voor een bepaalde combinatie van een glos en modificeerders de bijbehorende instructies genereert. Hier is

$$G_n = G \times \bigcup_{i=1}^{n} M^n,$$

oftewel de verzameling paren van een glos met 1 tot n modificeerders. Om te beginnen nemen we $n = 1$, dus een glos heeft slechts één modificeerder.

Deze modificeerders zullen later nodig zijn om gebaren in een bepaalde context een specifiekere vorm te geven[2]. Belangrijk is dat M in ieder geval m_0 bevat: de nul-modificeerder die de meest neutrale vorm van het gebaar genereert, ook wel de **citeervorm**[3] genoemd.

Om een voorbeeld te geven zou L zou de instructies voor gebaren in KOMVA-notatie kunnen bevatten. Als we het gebaar voor TAAL (figuur 1.2) als voorbeeld gebruiken:

$$G = \{\text{TAAL}\}$$
$$L = \{\varnothing\ \text{b0}\ \updownarrow\ \updownarrow\ \twoheadrightarrow\ \text{L}\}$$
$$g(\text{TAAL}, m_0) = \varnothing\ \text{b0}\ \updownarrow\ \updownarrow\ \twoheadrightarrow\ \text{L}$$

[2] Denk aan het vervoegen van een werkwoord in het Nederlands: “loopt” en “lopen” zijn twee verschillende woorden, maar kunnen ook gezien worden als hetzelfde begrip, gecombineerd met een andere vervoegingsmodificeerder.

[3] Dit is het gebaar zoals deze in het woordenboek beschreven wordt [2].

3.1.3 Abstracte grammatica en vertalen naar NGTb

Om de eerder gedefinieërde taal NLb te vertalen, ontwerpen we ook een tegenhanger van deze taal als een subset van de NGT (uitgedrukt in glossen). Deze taal noemen we **NGTb**. We kunnen een grammatica voor deze taal definiëren op eenzelfde manier als voor NLb, er op lettend dat de glossen uit NGTg gebruikt worden als terminale symbolen.

```
1   Z( ab )  ← N( a )  VC( b )
2   N ← ZN
3   VC ← W1
4   VC( ab )  ← N( a )  W2( b )
5   W1 ← W
6   W2 ← W
7   ZN(‘JONGEN’ | ‘BAL’)
8   W1(‘LOPEN’ | ‘WERKEN’)
9   W2(‘PAKKEN’ | ‘GOOIEN’)
10  W(‘KOKEN’)
```

Deze grammatica verschilt slechts op een paar punten van die voor NLb. Ten eerste is de woordvolgorde anders (SOV in plaats van SVO),

wat leidt tot een andere regel voor VC's met een tweeplaatsig werkwoord en lijdend voorwerp. Daarnaast zijn de terminale symbolen vervangen door glossen uit NGTg. Hierbij valt op dat de speciale notatie (vervoeg `'...'`) niet meer nodig is, aangezien (deze) werkwoorden in de NGT niet van vorm veranderen afhankelijk van geslacht en meervoudigheid van het onderwerp.

Het is opvallend dat er een grote gemeenschappelijke basis is tussen de twee grammatica's. Sterker nog, als we regel 4 in de grammatica voor NGTb herschrijven als

$$4 \qquad VC(ba) \leftarrow W2(a)\ N(b),$$

zijn alle regels zonder terminale symbolen identiek, op de volgorde waarin strings van terminale symbolen toegevoegd worden na. Van deze gedeelde regels maakt het Grammatical Framework gebruik om zinnen te vertalen, door het gemeenschappelijke deel te extraheren als een abstracte, taalonafhankelijke grammatica. In deze abstracte grammatica missen slechts twee onderdelen: de terminale symbolen en de transformationele delen van de regels. Deze twee komen overeen met wat in het GF een linearisatieregel genoemd wordt.

Om de abstracte grammatica te vormen zijn twee stappen nodig:

1. In de oorspronkelijke grammatica's moeten alle terminale symbolen die te onderscheiden moeten zijn tijdens de vertaling, een eigen regel krijgen die ze omzet in een non-terminaal symbool.[4].

2. Alle regels die terminale symbolen bevatten, en de transformationele delen van de regels, moeten uit de abstracte grammatica verwijderd worden. Deze worden voor beide grammatica's apart opgeslagen en vormen zo de concrete grammatica's voor beide talen.

Als we deze stappen volgen komen we uit op de volgende abstracte grammatica[5], die we AGb noemen:

1	$Z \leftarrow N\ VC$
2	$N \leftarrow ZN$
3	$VC \leftarrow W1$
4	$VC \leftarrow W2\ N$
5	$W1 \leftarrow W$
6	$W2 \leftarrow W$
7,8	ZN
9,10	$W1$
11,12	$W2$
13	W

Met daarnaast de concrete invulling voor NLb:

[4] Anders gaat de betekenis verloren, aangezien alleen non-terminale symbolen en de regels om ze te vormen in de abstracte grammatica overblijven. Een regel als ZN(`'de jongen'|'de bal'`) zorgt ervoor dat na abstractie, de begrippen 'de jongen' en 'de bal' niet meer te onderscheiden zijn.

[5] Het valt op dat hierin veel 'herhaalde' regels lijken te staan, met slechts één symbool. Dat komt omdat deze regels later ingevuld worden met woorden voor verschillende begrippen, en dus is het toepassen van bijvoorbeeld regel 7 niet hetzelfde als het toepassen van regel 8.

```
1       Z(ab) ← N(a) VC(b)
3       VC(a) ← W1(a)
4       VC(ab) ← W2(a) N(b)
7       ZN('de jongen')
8       ZN('de bal')
9       W1(vervoeg 'lopen')
10      W1(vervoeg 'werken')
11      W2(vervoeg 'pakken')
12      W2(vervoeg 'gooien')
13      W(vervoeg 'koken')
```

en die voor NGTb:

```
1       Z(ab) ← N(a) VC(b)
3       VC(a) ← W1(a)
4       VC(ba) ← W2(a) N(b)
7       ZN('JONGEN')
8       ZN('BAL')
9       W1('LOPEN')
10      W1('WERKEN')
11      W2('PAKKEN')
12      W2('GOOIEN')
13      W('KOKEN')
```

Om zinnen te vertalen van de ene naar de andere taal, kan nu het volgende proces doorlopen worden:

1. Begin met een string van terminale symbolen in één taal, bijvoorbeeld NLb.
2. Pas een reeks inferentieregels toe uit de abstracte grammatica en de concrete grammatica van die taal, tot een beginsymbool te komen (hier *Z*) dat de inputstring als inhoud bevat. Onthoud de reeks regels die hiervoor toegepast moeten worden.
3. Pas dezelfde regels toe, maar nu met de concrete grammatica van een andere taal, om opnieuw het beginsymbool af te leiden.
4. De string die bij het beginsymbool hoort is de vertaalde zin.

Een voorbeeld is als volgt:

1. Begin met: 'de jongen pakt de bal' in NLb.
2. Reduceer dit tot een Z als volgt:
   (a) Input: 'de jongen pakt de bal'
   (b) Toepassen van 7, 11, 8 geeft: ZN('de jongen') W2('pakt') ZN('de bal')
   (c) Toepassen van 2, 2 geeft: N('de jongen') W2('pakt') N('de bal')

(d) Toepassen van 4 geeft: N('de jongen') VC('pakt de bal')

(e) Toepassen van 1 geeft: Z('de jongen pakt de bal').

De inhoud van deze Z is hetzelfde als de input, dus dit is een geldige zin.

De totale reeks regels is: 7,11,8 → 2,2 → 4 → 1.

3. Pas dezelfde regels toe, maar nu met de concrete grammatica van NGTb:

(a) Toepassen 7, 11, 8: ZN('JONGEN') W2('PAKKEN') ZN('BAL')

(b) Toepassen 2, 2: N('JONGEN') W2('PAKKEN') N('BAL')

(c) Toepassen 4: N('JONGEN') VC('BAL PAKKEN')

(d) Toepassen 1: Z('JONGEN BAL PAKKEN')

De vertaling naar NGTb is de zin: 'JONGEN BAL PAKKEN'.

3.2 Verwijzingen

3.2.1 Localisatie in NGTg

Om verwijzingen te kunnen uiten in de NGT, zijn twee stappen nodig:

1. Bij het gebaren van de referent moet deze gelocaliseerd worden[6].
2. Er moet op het moment van verwijzen kennis zijn van de locatie van de referent, en hiernaar wordt verwezen door middel van een indexgebaar.

[6] Voor gebaren op het lichaam kan dit door middel van een indexgebaar erna.

Ieder willekeurig gebaar dat een persoon of object representeert, kan gelocaliseerd worden. We kunnen bijvoorbeeld een gelocaliseerde variant van het gebaar dat 'jongen' betekent, noteren als $\textsc{jongen}_{3a}$. Dit betekent: het gebaar 'JONGEN', gelocaliseerd op locatie 3a.

Formeel gezien zijn hiervoor de modificeerders als gedefiniëerd in sectie 3.1.2 nodig. We breiden deze verzameling uit met de verzameling $M_L \subseteq L$ van **locatiemodificeerders**, gegeven door

$$M_L = \{l_x \mid x \in A\} = \{l_\mathrm{a}, l_\mathrm{b}, l_\mathrm{c}, l_\mathrm{d}, ...\}$$

waar A de verzameling is van alle letters in het alfabet. In het lexicon L worden dan voor ieder gebaar $\|A\| = 26$ aparte sets instructies opgenomen, zodat wanneer $g(x, m_0)$ de citeervorm van een gebaar genereert, $g(x, l_y)$ het gebaar gelocaliseerd op locatie $3y$ genereert. Bijvoorbeeld: $g(\text{JONGEN}, l_a)$ geeft het gebaar behorende bij 'JONGEN', gelocaliseerd op locatie 3a. Daarnaast definiëren we dat de notatie GLOS_{3x} (een glos met een subscript $3x$) geïnterpreteerd moet worden als een GLOS $\in G$, samen met de modificeerder $l_x \in M_L$.

Indexgebaren definiëren we soortgelijk: er is een speciaal gebaar IX dat ook gelocaliseerd kan worden. Het indexgebaar heeft echter geen

citeervorm, dus maken we hiervoor een uitzondering in de definitie van G_n:

$$G_n = \left(G \times \bigcup_{i=1}^{n} M^n \right) \cup \left(\{\text{IX}\} \times \bigcup_{i=1}^{n} (M \setminus \{m_0\})^n \right).$$

3.2.2 Uitbreiding naar NLpv: indexering en verwijzing

Om referenten in de input te kunnen herkennen en persoonlijk voornaamwoorden toe te voegen, breiden we NLb uit naar NLpv. Hierin komen twee belangrijke verschillen aan bod:

1. Naamwoordgroepen kunnen nu een referent zijn, en dienen met een unieke **index** geannoteerd te kunnen worden. Dit is een geheel getal groter of gelijk aan 1. Zo kan een verwijswoord later deze index gebruiken om naar deze naamwoordgroep terug te verwijzen.

   Voor deze indexering van referenten gebruiken we de volgende notatie: '[[4 de jongen]]' betekent dat 'de jongen' een referent is met index 4.

2. Er worden persoonlijk voornaamwoorden toegevoegd. Daarvan zijn er drie soorten:

   (a) De eerste persoon enkelvoud: 'ik'.

   (b) De tweede persoon enkelvoud: 'je' of 'jij'.

   (c) De derde persoon enkelvoud: 'hij', 'zij', 'hem', 'haar' of 'het'.

   De meervoudsvormen laten we buiten beschouwing. Ook wordt er geen onderscheid gemaakt tussen mannelijk/vrouwelijk en de grammaticale functie (zij/haar), omdat dit onderscheid niet relevant voor de taalonafhankelijke representatie van betekenis is.

   Voor de persoonlijk voornaamwoorden wordt ook aangegeven naar welke referent ze verwijzen. Dat gebeurt door middel van de volgende notatie: '[3 hij]' betekent dat 'hij' verwijst naar de referent met index 3.

Daarnaast is één extra uitbreiding nodig: er kan niet verwezen worden wanneer de input slechts bestaat uit zinnen met één onderwerp en één lijdend voorwerp. Daarom worden ook **samengestelde zinnen** toegevoegd, die gevormd door twee hoofdzinnen[7] te koppelen met het nevenschikkend voegwoord 'en'.

[7] Een hoofdzin is een zin zoals in NLb, dus met een onderwerp, persoonvorm, en mogelijk een lijdend voorwerp.

Een voorbeeld van een geldige zin in NLpv moet dus zijn:

[[1 de jongen]] pakt [[2 de bal]] en [1 hij] gooit [2 hem]

Hiertoe definiëren we de volgende grammatica voor NLpv:

$1*$ $\quad Z \leftarrow SZ$

$2*$ $\quad Z \leftarrow EZ$

```
3*      SZ(a 'en' b) ← SZ(a) H(b)
4*      SZ(a 'en' b) ← H(a) H(b)
5*      EZ ← H
6*      H(ab) ← N(a) VC(b)
7*      N('[[' i n ']]') ← N(n) I(i)
8       N ← ZN
9*      N('ik')
10*     N('je' | 'jij')
11*     N('[' i ('hij' | 'zij' | 'hem' | 'haar' | 'het') ']') ← I(i)
12      VC ← W1
13      VC(ab) ← W2(a) N(b)
14      W1 ← W
15      W2 ← W
16,17   ZN('de jongen' | 'de bal')
18,19   W1((vervoeg 'lopen') | (vervoeg 'werken'))
20,21   W2((vervoeg 'pakken') | (vervoeg 'gooien'))
22      W(vervoeg 'koken')
```

De regels met een sterretje achter het nummer zijn toegevoegd of aangepast. Regels 1 t/m 6 gaan over samenstellen van zinnen en definieren de nieuwe symbolen H, EZ, en SZ voor respectievelijk hoofdzinnen, enkelvoudige zinnen en samengestelde zinnen. Regel 7 gaat over het indexeren van referenten, deze zegt dat een N en het nieuwe indexsymbool I samengevoegd mogen worden tot een nieuwe N.

Regels 9 t/m 11 voegen de persoonlijk voornaamwoorden toe. Voor de eerste en tweede persoon zijn dit gewoon nieuwe waarden voor N. In de derde persoon ligt dit iets ingewikkelder: deze kunnen alleen samen met een index gebruikt worden. Dit wordt gespecificeerd in regel 11, op een zelfde soort manier als in regel 7.

De regels voor het vormen van een indexsymbool I zijn niet toegevoegd aan de eerdere grammatica, omdat deze nog wat toelichting verdienen. Laten we veronderstellen dat we getallen gebruiken als indices. Als we de getallen 1 tot en met n gebruiken, betekent dit dat er n verschillende waarden zijn waaruit een indexsymbool gevormd kan worden. Dit resulteert uiteindelijk in n verschillende regels[8]. Om een zo uitgebreid mogelijke input te accepteren, willen we een grote waarde voor n nemen, maar dit leidt er toe dat het aantal grammaticaregels flink toeneemt.

[8] Iedere unieke index moet een eigen regel krijgen, anders gaat de waarde verloren tijdens de vertaling.

Om het aantal regels zo klein mogelijk te houden, kunnen we compositioneel te werk gaan. Getallen boven de 9 bestaan uit meerdere cijfers, die samen het hele getal vormen. Zo kunnen we ook een grammatica schrijven voor getallen: de cijfers zijn de 'woorden', en deze vormen samen een 'zin', het getal. Om te beginnen ondersteunen we indices van twee cijfers,

zodat $n = 99$ (de grootst mogelijke index). Daarmee komen we uit op de volgende grammatica:

23 $I \leftarrow D$
24 $I(ab) \leftarrow D(a)\ D(b)$
25−34 $D(\texttt{`0'} \mid \texttt{`1'} \mid \texttt{`2'} \mid \texttt{`3'} \mid \texttt{`4'} \mid \texttt{`5'} \mid \texttt{`6'} \mid \texttt{`7'} \mid \texttt{`8'} \mid \texttt{`9'})$

Op deze manier is het aantal regels slechts $2 + 10 = 12$ in plaats van $n = 99$, maar toch is er een unieke reeks regels voor iedere index[9]. Bijvoorbeeld: 26, 27 → 24 geeft $D(\texttt{`1'})D(\texttt{`2'}) \rightarrow I(\texttt{`12'})$, oftewel index 12.

[9] Hierbij geldt wel dat indices 0 t/m 9 twee representaties hebben: 0, 1, 2, ... en 00, 01, 02, enzovoort. Een oplossing hiervoor zou zijn om regel 23 te laten vallen en altijd het maximale aantal cijfers te gebruiken, of om '0' als een apart symbool te definiëren dat niet als eerste cijfer gebruikt mag worden in regel 24. Hier wordt er voor gekozen om af te spreken dat getallen in de input niet met een 0 mogen beginnen.

3.2.3 Locatietoewijzing

De uitbreiding van NGTb naar NGTpv wordt ook gekenmerkt door het toevoegen van indices op naamwoordgroepen en door het toevoegen van persoonlijk voornaamwoorden. Het verschil is dat een naamwoordgroep met een index in de NGT een gelocaliseerde naamwoordgroep wordt, en dat persoonlijk voornaamwoorden worden geuit als indexgebaren die wijzen naar een bepaalde locatie. Daarom is een correspondentie tussen indices en locaties nodig. De correspondentie die hier gebruikt wordt is enumeratief, zo dat:

$1 \rightarrow$ 3a
$2 \rightarrow$ 3b
$3 \rightarrow$ 3c
$4 \rightarrow$ 3d
...
$25 \rightarrow$ 3y
$26 \rightarrow$ 3z

Iedere index i wordt dus gekoppeld aan de i-de letter van het alfabet.

Daarmee kunnen we de grammatica voor indices in NGTpv als volgt uitwerken:

23 $I(\texttt{`?'}) \leftarrow D(d0)$
$I(\texttt{`3a'}) \leftarrow D(d1)$
$I(\texttt{`3b'}) \leftarrow D(d2)$
...
$I(\texttt{`3i'}) \leftarrow D(d9)$
24 $I(\texttt{`3j'}) \leftarrow D(d1)\ D(d0)$
$I(\texttt{`3k'}) \leftarrow D(d1)\ D(d1)$
$I(\texttt{`3l'}) \leftarrow D(d1)\ D(d2)$
...
$I(\texttt{`3z'}) \leftarrow D(d2)\ D(d6)$

$I(\text{`?'}) \leftarrow D(d2)\ D(d7 \mid d8 \mid d9)$

$I(\text{`?'}) \leftarrow D(d3 \mid d4 \mid \ldots \mid d9 \mid d0)\ D(b)$

25−34 $D(d0 \mid d1 \mid d2 \mid d3 \mid d4 \mid d5 \mid d6 \mid d7 \mid d8 \mid d9)$

In deze grammatica heeft D een nogal andere betekenis. D heeft weliswaar 10 mogelijke waarden (productieregels), maar deze waarden komen niet voor in de inhoud van I. Dat is omdat er in een locatie in de NGT geen compositie mogelijk is, er komen geen losse getallen in voor. Daarom dient D slechts als een identificatie van welke index in de input aanwezig is geweest, en is er voor iedere combinatie die bij een bepaalde locatie hoort een regel die een I vormt met als inhoud de locatie[10]. Om hetzelfde voorbeeld als bij NLpv te nemen: 26, 27 → 24 geeft $D(d1)D(d2) \rightarrow I(\texttt{`3l'})$, dus index 12 wordt afgebeeld op de locatie 3l.

Een minder gewenste bijkomstigheid is dat de numerieke indices die geen locatie toegewezen hebben gekregen wel te uiten moeten zijn. Deze worden geuit door het plaatsvervangend symbool '?'. Het zou idealiter op een bepaalde manier onmogelijk gemaakt moeten worden om deze indices in NGT te gebruiken.

[10] Het lijkt in deze notatie er op alsof er een tal van regels zijn om indices te vormen, er zijn bijvoorbeeld 10 verschillende regels die onder 24 vallen. De reden waarom dit mogelijk is is omdat al deze regels slechts een verschillend transformationeel deel hebben. Dit maakt het mogelijk om deze regels in de abstracte grammatica als één regel te beschouwen.

Een mogelijke andere notatie zou zijn:

$$I(\texttt{`3a'} \leftarrow d = d1\ ;\ \ldots\ ) \leftarrow D(d)$$

Uiteindelijk moeten er meerdere regels gegenereerd worden om een I te kunnen afleiden in NGTpv, maar dit maakt voor het vertaalproces niet uit, zolang er voor iedere combinatie van D's slechts één sub-regel toe te passen is.

3.2.4 Uitbreiding naar NGTpv

De volledige uitbreiding naar NGTpv kan gemaakt worden door met de grammatica voor indices gelocaliseerde naamwoordgroepen en indexgebaren toe te voegen. De resulterende grammatica is als volgt:

1∗ $Z \leftarrow SZ$

2∗ $Z \leftarrow EZ$

3∗ $SZ(a\ \text{`/'}\ b) \leftarrow SZ(a)\ H(b)$

4∗ $SZ(a\ \text{`/'}\ b) \leftarrow H(a)\ H(b)$

5∗ $EZ \leftarrow H$

6∗ $H(ab) \leftarrow N(a)\ VC(b)$

7∗ $N(n\ \text{`\_\{'}\ i\ \text{`\}'}) \leftarrow N(n)\ I(i)$

8 $N \leftarrow ZN$

9∗ $N(\texttt{`IX\_\{1\}'})$

10∗ $N(\texttt{`IX\_\{2\}'})$

11∗ $N(\texttt{`IX\_\{'}\ i\ \texttt{`\}'}) \leftarrow I(i)$

12 $VC \leftarrow W1$

13 $VC(ab) \leftarrow W2(a)\ N(b)$

14 $W1 \leftarrow W$

15 $W2 \leftarrow W$

16,17 $ZN(\texttt{`JONGEN'} \mid \texttt{`BAL'})$

18,19 $W1(\texttt{`LOPEN'} \mid \texttt{`WERKEN'})$

20,21 $W2(\texttt{`PAKKEN'} \mid \texttt{`GOOIEN'})$

22 *W*(‘KOKEN’)

De verschillen in deze grammatica ten opzichte van NLpv zijn dat deelzinnen niet met een voegwoord gekoppeld worden (ze worden gewoon achter elkaar gezet), en dat de notatie van ‘indexering’ (localisatie) anders is[11].

[11] De notatie ‘IX_{1}’ is gekozen om overeen te komen met de broncode voor de glos IX$_1$.

De grammatica’s voor NLpv en NGTpv kunnen we weer omzetten tot een abstracte grammatica, AGpv:

1∗ $Z \leftarrow SZ$
2∗ $Z \leftarrow EZ$
3∗ $SZ \leftarrow SZ\ H$
4∗ $SZ \leftarrow H\ H$
5∗ $EZ \leftarrow H$
6∗ $H \leftarrow N\ VC$
7∗ $N \leftarrow N\ I$
8 $N \leftarrow ZN$
9∗ N
10∗ N
11∗ $N \leftarrow I$
12 $VC \leftarrow W1$
13 $VC \leftarrow W2\ N$
14 $W1 \leftarrow W$
15 $W2 \leftarrow W$
16,17 ZN
18,19 $W1$
20,21 $W2$
22 W
23 $I \leftarrow D$
24 $I \leftarrow D\ D$
25−34 D

en de concrete aanvulling voor NLpv:

3∗ $SZ(a\ \text{‘en’}\ b) \leftarrow SZ(a)\ H(b)$
4∗ $SZ(a\ \text{‘en’}\ b) \leftarrow H(a)\ H(b)$
6∗ $H(ab) \leftarrow N(a)\ VC(b)$
7∗ $N(\text{‘[[’}\ i\ n\ \text{‘]]’}) \leftarrow N(n)\ I(i)$
9∗ $N(\text{‘ik’})$
10∗ $N(\text{‘je’} \mid \text{‘jij’})$
11∗ $N(\text{‘[’}\ i\ (\text{‘hij’} \mid \text{‘zij’} \mid \text{‘hem’} \mid \text{‘haar’} \mid \text{‘het’})\ \text{‘]’}) \leftarrow I(i)$
13 $VC(ab) \leftarrow W2(a)\ N(b)$
16 $ZN(\text{‘de jongen’})$
17 $ZN(\text{‘de bal’})$
18 $W1(\text{vervoeg ‘lopen’})$

19 *W1*(vervoeg ‘werken’)
20 *W2*(vervoeg ‘pakken’)
21 *W2*(vervoeg ‘gooien’)
22 *W*(vervoeg ‘koken’)
24 $I(ab) \leftarrow D(a)\ D(b)$
25–34 *D*(‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’)

en voor NGTpv:

3∗ $SZ(a\ \text{`/'}\ b) \leftarrow SZ(a)\ H(b)$
4∗ $SZ(a\ \text{`/'}\ b) \leftarrow H(a)\ H(b)$
6∗ $H(ab) \leftarrow N(a)\ VC(b)$
7∗ $N(n\ \text{`\_\{'}\ i\ \text{`\}'}) \leftarrow N(n)\ I(i)$
9∗ *N*(‘IX_{1}’)
10∗ *N*(‘IX_{2}’)
11∗ $N(\text{`IX\_\{'}\ i\ \text{`\}'}) \leftarrow I(i)$
13 $VC(ab) \leftarrow W2(a)\ N(b)$
16 *ZN*(‘JONGEN’)
17 *ZN*(‘BAL’)
18 *W1*(‘LOPEN’)
19 *W1*(‘WERKEN’)
20 *W2*(‘PAKKEN’)
21 *W2*(‘GOOIEN’)
22 *W*(‘KOKEN’)
23 $I(\text{`?'}) \leftarrow D(d0)$
$I(\text{`3a'}) \leftarrow D(d1)$
...
$I(\text{`3i'}) \leftarrow D(d9)$
24 $I(\text{`3j'}) \leftarrow D(d1)\ D(d0)$
...
$I(\text{`3z'}) \leftarrow D(d2)\ D(d6)$
$I(\text{`?'}) \leftarrow D(d2)\ D(d7 \mid d8 \mid d9)$
$I(\text{`?'}) \leftarrow D(d3 \mid d4 \mid \ldots \mid d9 \mid d0)\ D(b)$
25–34 $D(d0 \mid d1 \mid d2 \mid d3 \mid d4 \mid d5 \mid d6 \mid d7 \mid d8 \mid d9)$

3.2.5 Herhalen van onderwerp

Een veel voorkomende constructie in de NGT is het herhalen van het onderwerp aan het einde van een zin. Een zin als ‘de jongen pakt de bal’ wordt dan:

JONGEN$_{3a}$ BAL PAKKEN IX$_{3a}$

Hier wordt aan het einde van de zin opnieuw naar ‘de jongen’ verwezen met een indexgebaar. Hier is van belang dat ‘JONGEN’ een locatie toegewezen krijgt, omdat er anders niet naar verwezen kan worden.

Dit fenomeen heet in het algemeen **right dislocation**, wat er naar verwijst dat het onderwerp aan het einde van de zin (rechts) opnieuw geplaatst wordt [9]. Daarnaast komt ook de term **pronoun copy** voor om dit herhalen van persoonlijk voornaamwoorden te beschrijven [10].

Het is niet volledig duidelijk in hoeverre deze herhaling noodzakelijk is [9]. Wel geeft het nadruk op en verduidelijking over de focus van de zin. Daarom is er voor gekozen om in ieder geval wanneer het onderwerp van de zin een persoonlijk voornaamwoord of gelocaliseerde naamwoordgroep is, aan het einde van de zin opnieuw naar het onderwerp te verwijzen.

Daarvoor breiden we de grammatica enigzins uit. Om het mogelijk te maken naar een gelocaliseerde naamwoordgroep te verwijzen, geven we het symbool N twéé strings als inhoud: de uiting van de naamwoordgroep met locatie, en een verwijzing naar de naamwoordgroep om later te gebruiken. De grammatica voor NGTpv wordt dan als volgt aangepast:

6∗ $H(n\ v\ m) \leftarrow N(n,m)\ VC(v)$

7∗ $N(n\ \texttt{`\_\{'}\ i\ \texttt{`\}'},\ \texttt{`IX\_\{'}\ i\ \texttt{`\}'}) \leftarrow N(n,m)\ I(i)$

8 $N(n,\ \texttt{''}) \leftarrow ZN(n)$

9∗ $N(\texttt{`IX\_\{1\}'},\ \texttt{`IX\_\{1\}'})$

10∗ $N(\texttt{`IX\_\{2\}'},\ \texttt{`IX\_\{2\}'})$

11∗ $N(\texttt{`IX\_\{'}\ i\ \texttt{`\}'},\ \texttt{`IX\_\{'}\ i\ \texttt{`\}'}) \leftarrow I(i)$

13 $VC(w\ n) \leftarrow W2(w)\ N(n,m)$

Hier is te zien dat N altijd twee strings als inhoud heeft. Voor een ongelocaliseerd zelfstandig naamwoord, is de waarde van de verwijzing de lege string, oftewel geen verwijzing. Wanneer een N een locatie krijgt toegewezen door middel van regel 7, wordt als verwijzing een indexgebaar opgeslagen dat naar dezelfde locatie wijst. Wanneer een hoofdzin gevormd wordt, wordt nu deze verwijzing naar het onderwerp aan het einde van de zin herhaald. In het geval van een tweeplaatsige verbale constituent wordt deze verwijzing genegeerd.

Verder hoeft niks aangepast te worden, want de inferentieregels van AGpv gelden nog steeds. Met de nieuwe grammatica voor NGTpv wordt een zin als:

[[1 de jongen]] pakt de bal

nu vertaald naar:

```
JONGEN_{3a} BAL PAKKEN IX_{3a}
```

HOOFDSTUK 4

Implementatie

4.1 Basis van het vertaalsysteem

Een GF-programma is opgedeeld in modules die samen de abstracte en concrete grammatica's definiëren. Om de basisgrammatica AGb uit sectie 3.1.3 te definiëren gebruiken we twee modules:

- Lex.gf (lexicon): hier worden functies ('fun') gedefinieerd voor het genereren van terminale symbolen. Deze module bevat alle woorden/begrippen in de taal. De categorieën ('cat') voor non-terminale symbolen waar deze woorden toe behoren zijn ook hier gedefinieerd (ZN, W1, W2, en W).
- Gram.gf (grammatica): uitbreiding van Lex.gf. Hier worden categoriëen voor de overige constituenten gedefinieerd (Z, N, VC) en de grammaticaregels die erbij horen. Dit is ook de hoofdmodule die gebruikt wordt voor het vertalen, aangezien deze de 'startcat' (beginsymbool) bevat.

Daarnaast zijn concrete modules voor beide talen gedefiniëerd: GramNLb.gf en LexNLb.gf, en daarnaast GramNGTb.gf en LexNGTb.gf. Deze modules geven invulling aan de eerder gedefiniëerde functies (via 'lin'). De Gram-modules lijken voor beide talen erg op elkaar, alleen het verschil in woordvolgorde is hier geïmplementeerd. Omdat veel regels erg simpel zijn (bijvoorbeeld: $N(u) \leftarrow ZN(u)$) zijn een aantal hulpfuncties gedefineerd in module Types.gf, voor onder andere het kopiëren van de inhoud van een symbool of het achter elkaar plakken van de inhoud van twee symbolen. In de Lex-modules zijn alle functies constant: ze geven de uiting van een begrip in een bepaalde taal aan.

Het GF heeft uitgebreide support voor hulpfuncties (*operaties* genoemd, via 'oper'). Dit zijn functies die bij het compilen van de grammatica's worden uitgevoerd en geëlimineerd. Daarom kunnen ze meer gedrag

uitvoeren dan normaal gezien een MCFG kan, maar wel alleen op input die al bij compile-time bekend is (dus, constanten). Dit maakt de code echter wel een stuk modulairder en minder redundant, aangezien regelmaat in de vorm van woorden gebruikt kan worden om voor een gegeven woord automatisch regels voor alle varianten te genereren. Bijvoorbeeld: de functie 'vervoeg' uit de grammatica voor NLb kan als een operatie gedefiniëerd worden die alle vervoegingen van het gegeven werkwoord genereert. Op deze manier komt de implementatie van de grammatica vrijwel letterlijk overeen met de theoretische variant.

4.2 Indexering

Om de varianten NLpv en NGTpv te implementeren, zijn extra symbolen en regels nodig. In het bijzonder zijn er regels nodig om indices te produceren. Deze zijn gedefiniëerd in een extra module, Indices.gf, en twee concrete aanvullingen IndicesNL.gf en IndicesNGT.gf.

Om de grammatica van de indices in NLpv te definiëren is geen bijzondere fuctionaliteit nodig. Voor NGTpv ligt dat anders: de categorie D heeft daar geen bijbehorende linearisatie naar tekst. Daarvoor kunnen we D als linearisatietype een parametertype geven, dat de waarden d0 tot d9 kan aannemen. Op deze manier hebben we 10 unieke waarden voor D waarmee we indices kunnen vormen. Omdat D een parametertype is, kunnen in de regel voor een index conditionele statements uitgevoerd worden op basis van de waarde de D die als argument meegegeven wordt. In dit geval zijn dat `case`-statements, die er als volgt uitzien:

```
\d -> case d of {
  d1 => l3 "a" ; d2 => l3 "b" ;
  d3 => l3 "c" ; d4 => l3 "d" ;
  d5 => l3 "e" ; d6 => l3 "f" ;
  d7 => l3 "g" ; d8 => l3 "h" ;
  d9 => l3 "i" ; d0 => NONE
}
```

Hier is `l3 "x"` een operatie die de string `"3x"` genereert, en `NONE = "?"` (of een ander plaatsvervangend symbool). Op deze manier kan het transformationele deel (`lin`) van een linearisatieregel afhangen van de waarde van de argumenten[1].

[1] Wat hier in de praktijk gebeurt, is dat D opgesplitst wordt in 10 verschillende categoriëen met slechts één element. Daarnaast wordt de linearisatieregel opgesplitst in één regel per mogelijkheid in het case-statement [7].

4.3 Voorbeelden van de implementatie

In de basisgrammatica worden zinnen ondersteund met alleen een onderwerp, persoonsvorm en lijdend voorwerp. Een aantal willekeurig gegenereerde voorbeelden van zinnen die hiermee vertaald kunnen worden

is weergegeven in tabel 4.1. Daarnaast zijn ook voorbeelden gegenereerd voor de zinnen die in NLpv geuit kunnen worden. Dit zijn samengestelde zinnen met geindexeerde naamwoordgroepen en persoonlijk voornaamwoorden. De vertalingen hiervan naar NGTpv zonder en met herhaling van het onderwerp zijn weergegeven in respectievelijk tabel 4.2 en 4.3.

Input (NLb)	Output (NGTb, in glos-notatie)
de bal loopt	BAL LOPEN
de bal gooit de jongen	BAL JONGEN GOOIEN
de jongen pakt de bal	JONGEN BAL PAKKEN
de jongen kookt	JONGEN KOKEN
de jongen gooit de jongen	JONGEN JONGEN GOOIEN
de jongen kookt de bal	JONGEN BAL KOKEN
de bal kookt	BAL KOKEN
de bal pakt de jongen	BAL JONGEN PAKKEN
de bal kookt de jongen	BAL JONGEN KOKEN
de bal kookt de bal	BAL BAL KOKEN

Tabel 4.1: Een aantal zinnen uit de taal NLb, vertaald naar NGTb met het vertaalprogramma. De inputzinnen zijn gegenereerd door middel van het GF-commando `generate_random`.

Input (NLpv)	Output (NGTpv, zonder herhaling van onderwerp, in glos-notatie)
jij kookt en ik loop	IX_2 KOKEN / IX_1 LOPEN
de jongen pakt jou	JONGEN IX_2 PAKKEN
jij kookt en ik gooi jou en jij werkt en het boek kookt jou en de jongen gooit jou en de bal pakt mij	IX_2 KOKEN / IX_1 IX_2 GOOIEN / IX_2 WERKEN / BOEK IX_2 KOKEN / JONGEN IX_2 GOOIEN / BAL IX_1 PAKKEN
ik werk en ik werk en het boek werkt en de bal kookt mij en ik pak jou en de bal loopt	IX_1 WERKEN / IX_1 WERKEN / BOEK WERKEN / BAL IX_1 KOKEN / IX_1 IX_2 PAKKEN / BAL LOPEN
ik gooi [[1 de bal]] en de jongen pakt [1 hem]	IX_1 BAL_{3a} GOOIEN / JONGEN IX_{3a} PAKKEN
[[1 de bal]] pakt [1 hem]	BAL_{3a} IX_{3a} PAKKEN
[1 zij] kookt de bal	IX_{3a} BAL KOKEN
[[1 [[2 jij]]]] kook [3 hij]	IX_{23b3a} IX_{3c} KOKEN
ik kook [[2 de jongen]] en [1 hij] pakt [2 hem] en jij kookt het boek	IX_1 JONGEN_{3b} KOKEN / IX_{3a} IX_{3b} PAKKEN / IX_2 BOEK KOKEN
[[1 de bal]] werkt en jij kookt [1 hem] en [1 hij] kookt [1 hem] en ik kook [[2 mij]]	BAL_{3a} WERKEN / IX_2 IX_{3a} KOKEN / IX_{3a} IX_{3a} KOKEN / IX_1 IX_{13b} KOKEN

Tabel 4.2: Een aantal zinnen uit de taal NLpv, vertaald naar NGTpv met het vertaalprogramma. Hier vindt nog geen herhaling van het onderwerp plaats. De inputzinnen zijn gegenereerd door middel van het GF-commando `generate_random`.

Input (NLpv)	Output (NGTpv, mét herhaling van onderwerp, in glos-notatie)
jij kookt en ik loop	IX$_2$ KOKEN IX$_2$ / IX$_1$ LOPEN IX$_1$
de jongen pakt jou	JONGEN IX$_2$ PAKKEN
jij kookt en ik gooi jou en jij werkt en het boek kookt jou en de jongen gooit jou en de bal pakt mij	IX$_2$ KOKEN IX$_2$ / IX$_1$ IX$_2$ GOOIEN IX$_1$ / IX$_2$ WERKEN IX$_2$ / BOEK IX$_2$ KOKEN / JONGEN IX$_2$ GOOIEN / BAL IX$_1$ PAKKEN
ik werk en ik werk en het boek werkt en de bal kookt mij en ik pak jou en de bal loopt	IX$_1$ WERKEN IX$_1$ / IX$_1$ WERKEN IX$_1$ / BOEK WERKEN / BAL IX$_1$ KOKEN / IX$_1$ IX$_2$ PAKKEN IX$_1$ / BAL LOPEN
ik gooi [[1 de bal]] en de jongen pakt [1 hem]	IX$_1$ BAL$_{3a}$ GOOIEN IX$_1$ / JONGEN IX$_{3a}$ PAKKEN
[[1 de bal]] pakt [1 hem]	BAL$_{3a}$ IX$_{3a}$ PAKKEN IX$_{3a}$
[1 zij] kookt de bal	IX$_{3a}$ BAL KOKEN IX$_{3a}$
[[1 [[2 jij]]]] kook [3 hij]	IX$_{23b3a}$ IX$_{3c}$ KOKEN IX$_{3a}$
ik kook [[2 de jongen]] en [1 hij] pakt [2 hem] en jij kookt het boek	IX$_1$ JONGEN$_{3b}$ KOKEN IX$_1$ / IX$_{3a}$ IX$_{3b}$ PAKKEN IX$_{3a}$ / IX$_2$ BOEK KOKEN IX$_2$
[[1 de bal]] werkt en jij kookt [1 hem] en [1 hij] kookt [1 hem] en ik kook [[2 mij]]	BAL$_{3a}$ WERKEN IX$_{3a}$ / IX$_2$ IX$_{3a}$ KOKEN IX$_2$ / IX$_{3a}$ IX$_{3a}$ KOKEN IX$_{3a}$ / IX$_1$ IX$_{13b}$ KOKEN IX$_1$

Tabel 4.3: Een aantal zinnen uit de taal NLpv, vertaald naar NGTpv met het vertaalprogramma. Hier vindt nu ook herhaling van het onderwerp plaats, zoals in de zin IX$_1$ LOPEN IX$_1$ = 'ik loop'. De inputzinnen zijn gegenereerd door middel van het GF-commando `generate_random`.

HOOFDSTUK 5

Discussie

5.1 Evaluatie vertaalmogelijkheden

Het programma dat ontwikkeld is is in staat om zinnen in de taal NLpv correct te vertalen naar de taal NGTpv. De reikwijdte van deze talen ten opzichte van het Nederlands en de NGT is echter erg beperkt. De zinnen die vertaald kunnen worden zijn samengestelde zinnen, waar iedere deelzin een onderwerp, persoonsvorm en (eventueel) een lijdend voorwerp bevat. De soorten werkwoorden die gebruikt kunnen worden zijn zelfstandige werkwoorden, en als onderwerp of lijdend voorwerp kunnen zelfstandig naamwoorden gebruikt worden, of persoonlijk voornaamwoorden die naar een eerder gebruikt zelfstandig naamwoord verwijzen.

Één punt waarop het programma nog erg beperkt is, is het (symbolisch) lexicon. Het aantal woorden wat gebruikt kan worden dient slechts om de grammaticale correctheid te illustreren en is daarom erg klein gehouden. Het lexicon kan systematisch uitgebreid worden met andere woorden in dezelfde woordgroepen, door productieregels hiervoor toe te voegen. Om dit makkelijk te maken is het lexicon in de implementatie in een aparte module gedefiniëerd.

In de voorbeeldzinnen in tabel 4.2 en 4.3 vallen een aantal dingen op. Dit zijn voornamelijk gevallen waarin zinnen geaccepteerd worden die ook als input niet correct zijn. Dit betekent dat de grammatica **overgenereert**: de taal is groter dan gewenst is. Een voorbeeld is dat eerste en tweedepersoons voornaamwoorden als 'ik' en 'jij' ook gelocaliseerd kunnen worden. Dit is niet de bedoeling, omdat 'jij' al inherent een locatie heeft, en in de output genereert dit dan ook dubbel gelocaliseerde gebaren (zoals IX_{23a}). Daarnaast kunnen ook geïndexeerde naamwoordgroepen nogmaals geïndexeerd worden. Dit levert soortgelijke resultaten op. Dit betekent dat het correcter zou zijn om een aparte categorie te hebben voor geïndexeerde naamwoordgroepen, waar ook 'ik' en 'jij' deel van uit-

maken. Zo kunnen deze niet dubbel geïndexeerd worden.

Ook valt op dat het mogelijk is om in een zin het onderwerp als lijdend voorwerp te gebruiken, zoals in: '[[1 de bal]] pakt [1 hem]'. Dit is niet correct, aangezien in dit geval een wederkerend voornaamwoord gebruikt dient te worden: '[[1 de bal]] pakt [1 zichzelf]'. Ook in de NGT wordt dit anders geuit: door middel van het gebaar ZELF . Anders conflicteert het immers met het herhalen van het onderwerp, zoals ook te zien is in tabel 4.3 waar soms twee keer hetzelfde indexgebaar wordt gemaakt aan het einde van een zin.

Daarnaast komt er soms een verwijzing voor waarvoor er geen referent is. Dit is echter een goede constructie om mogelijk te hebben, omdat mogelijk in een eerdere zin deze referent genoemd wordt. Op deze manier kan het programma ook gebruikt worden om sequentieel meerdere zinnen te vertalen. Het verifiëren van de aanwezigheid van referenten is dus een eventuele preprocessing-stap, maar gebeurt niet tijdens de vertaling zelf.

5.2 Mogelijke uitbreidingen

5.2.1 Variatie in naamwoorden

Een uitbreiding die nog toegevoegd kan woorden is een goede manier om met samengestelde gebaren om te gaan. Dit zijn gebaren die zijn opgebouwd uit de aaneenschakeling van andere gebaren, ongeveer net als samenstellingen in het Nederlands, maar niet altijd hetzelfde: een samenstelling in de NGT is niet altijd een samenstelling in het Nederlands en andersom. Dit kan toegevoegd worden door middel van extra notatie, bijvoorbeeld

VERJAARDAG^CADEAU

om 'verjaardagscadeau' te uiten. Daarvoor moet wel iets extra's toegevoegd worden in de definitie van NGTg: een mogelijkheid om een samengestelde glos GLOS1^GLOS2 te vertalen naar 'gebaar 1 + overgang 1 → 2 + gebaar 2'. Natuurlijk kunnen samengestelde gebaren nu in principe al ook toegevoegd worden als aparte begrippen, maar dan gaat wel de relatie tussen de delen van het woord verloren.

Om nieuwe woordgroepen toe te voegen, varieert de hoeveelheid werk die nodig is. Voor bijvoeglijk naamwoorden en bijwoorden is in de NGT vaak een extra component van de gebaren nodig, zoals een non-manuele modificeerder (bijvoorbeeld een gezichtsuitdrukking) of een aanpassing in hoe het gebaar gemaakt wordt. Dit vraagt om een uitgebreide systematische analyse van de structuur van gebaren en de componenten waaruit ze bestaan, om een systematische notatie en regels hiervoor te ontwerpen.

5.2.2 Directionele en locatieve werkwoorden

Voor sommige werkwoorden zijn ook extra regels nodig. Dit zijn bijvoorbeeld werkwoorden met een meewerkend voorwerp (drieplaatsige werkwoorden), die in de NGT vaak directionele werkwoorden zijn[1]. Dit betekent dat ze een richting hebben, bijvoorbeeld:

ik geef jou het boek

Dit zou in de NGT geuit kunnen worden als:

IX_1 BOEK $_1\text{GEVEN}_2$

Hier wordt het werkwoord 'GEVEN' gericht door de handen van locatie 1 naar locatie 2 te bewegen. Zo krijgt het de betekenis 'ik geef aan jou'.

Het lastige aan het implementeren van deze werkwoorden is dat het onderwerp of meewerkend voorwerp mogelijk geen locatie heeft. Bijvoorbeeld in de zin 'de jongen geeft het boek aan mij':

JONGEN_{3a} BOEK $_{3a}\text{GEVEN}_1$

Hier moet JONGEN gelocaliseerd worden, omdat GEVEN anders geen punt heeft om vanuit gericht te worden. Dat betekent dat de input 'de jongen geeft het boek aan mij' bij vertaling een locatie voor 'de jongen' moet genereren, zonder dat deze locatie al in gebruik is. Dit is dus een contextgevoelig probleem. Dat betekent dat tenminste informatie nodig is over welke locaties al in gebruik zijn, om een ongebruikte locatie te genereren. Daarnaast moet ook opgelet worden met de structuur van het programma, om niet onnodig veel regels te genereren[2].

Het is nog maar de vraag of dit genereren van indices wel een redelijk oplosbaar probleem is binnen het GF[3].

Daarom zou het misschien beter zijn om vanuit de Nederlandse zin een incomplete of ambigue *parse tree* te genereren, waardoor ontbrekende informatie ná het parsen toegevoegd zou kunnen worden. In dat geval wordt het vertalen een proces dat uit meerdere stappen bestaat. De taak van de formele grammatica is dan het genereren van een (mogelijk incomplete) *parse tree* vanuit een NL-zin, en andersom het genereren van een NGT-zin vanuit een complete *parse tree*. Daarnaast is er dan een proces om een incomplete *parse tree* correct aan te vullen. Hiervoor is wel grammaticale informatie nodig, dus mogelijk kan het logische framework van het GF hierbij van pas. Daarmee kunnen onder andere 'berekeningen' op *parse trees* gedefinieerd worden, die gebruikt zouden kunnen worden om onvolledigheden regelmatig op te vullen.

Naast directionele werkwoorden zijn er ook locatieve werkwoorden. Dit zijn werkwoorden waarvoor het gebaar op een bepaalde plek in de ruimte gemaakt dient te worden. Voor bijvoorbeeld het werkwoord 'VINDEN' geeft de plek aan wat er gevonden wordt. Bijvoorbeeld:

[1] Ook zijn er drieplaatsige werkwoorden die niet gericht zijn. Dan wordt gebruik gemaakt van het gerichte hulpwerkwoord $_a\text{OP}_b$.

[2] In principe zijn er gegeven de zin JONGEN$_?$ BOEK $_?$GEVEN$_1$ evenveel variaties als dat er locaties bestaan. Zolang de twee locaties overeenkomen, is de zin correct.

[3] Een reden om hieraan te twijfelen, is dat de abstracte representatie van de output niet eenduidig wordt. Stel dat een implementatie vanuit de Nederlandse zin correct JONGEN_{3a} BOEK $_{3a}\text{GEVEN}_1$ weet te genereren. Als we deze output opnieuw parsen, kunnen we geen onderscheid maken tussen 'JONGEN' met een expliciete of een impliciete index. Daarom is het mogelijk gepaster om deze locatie toe te voegen in een extra stap in het vertaalproces.

JONGEN BOEK$_{3b}$ VINDEN$_{3b}$

voor ‘de jongen vindt het boek’. Hier komen dezelfde moeilijkheden naar voren als bij directionele werkwoorden, al hoeft alleen op het lijdend voorwerp gelet te worden, in plaats van het onderwerp én meewerkend voorwerp.

5.2.3 Overige aspecten

Een andere categorie van werkwoorden zijn de modale werkwoorden (moeten, kunnen, willen). Ook deze zijn niet triviaal om naar de NGT te vertalen. Ze hebben zowel meer variatie in hun syntactische positie, als semantische verschillen met hun Nederlandse tegenhangers: ‘MOETEN’ is bijvoorbeeld in de NGT altijd dwingend of urgent, maar een zin als ‘moet je eens kijken’ is niet zo bedoeld en kan beter niet met ‘MOETEN’ vertaald worden.

Ten slotte is ook tijdsaanduiding in de NGT van belang. Bij expliciete tijdaanduiding (‘gisteren ging ik naar de bieb’) is dit vrij letterlijk te vertalen, maar bij tijdsaanduiding door werkwoordsvervoeging (‘ik ging naar de bieb’, dus: in het verleden) zijn er gebaren AL en GEWEEST om aan te geven dat het in de verleden tijd gebeurd is. De werkwoordsvervoeging moet dus naar een extra zinsdeel worden vertaald, dat aan het begin van de zin komt te staan.

Deze aspecten zijn echter niet grammaticaal onhandelbaar, want het aantal modale werkwoorden en tijdsaanduidingen is beperkt. Daarom wordt er verwacht dat deze wel binnen de huidige opzet toe te voegen zijn aan de programma’s.

5.3 Verder onderzoek

Op basis van de huidige resultaten en de mogelijke uitbreidingen kunnen een aantal suggesties voor verder onderzoek geformuleerd worden.

Ten eerste is het toevoegen van **impliciete localisatie** een logisch vervolg. Hiermee wordt bedoeld: localisatie van begrippen die niet expliciet geannoteerd zijn, maar vanwege hun grammaticale functie gelocaliseerd moeten zijn. Het is sterk verwant aan de geïmplementeerde vormen van localisatie, aangezien het hiervan gebruik kan maken (zoals bij de geïmplementeerde *pronoun copy*) maar voor andere vormen ervan is de implementatie ingewikkelder, bijvoorbeeld voor het meewerkend voorwerp. Hoewel het een grammaticale kwestie is, is het lastig te implementeren aangezien er geen ‘vaste’ regel is: er moet tijdens de vertaling een locatie gekozen worden, en deze locatie moet uniek zijn. Om deze keuze te maken is dus (vrijwel) globale context vereist. De vraag is: hoe veel context is nodig, en hoe kan dit in een grammaticale structuur geformaliseerd worden?

Een ander aspect is het correct en efficiënt indexeren van referenten in inputtekst. Er wordt voor de vertaling van uit gegaan dat deze indices correct zijn aangebracht, maar zoals eerder genoemd is dit geen triviale taak. Voor een eind-tot-eind vertaling zal een input in het Nederlands eerst geannoteerd moeten worden via een toepasselijk *coreference resolution*-algoritme[4]. Daarnaast zou het goed zijn als er efficiënt gebruik wordt gemaakt van de locaties te beschikbaar zijn. Het zal voor een NGT-spreker natuurlijker zijn om gedurende een lang verhaal verschillende personen of objecten op locaties 3a en 3b te plaatsen wanneer dit zonder verwarring mogelijk is dan om 8 verschillende locaties te gebruiken. Een proces voor het zo mogelijk hergebruiken van locaties zou ook een aanwinst zijn voor de begrijpelijkheid en natuurlijkheid van de vertaling.

[4] Een project dat dit probeert voor Nederlandse tekst is COREA [6], en simpelere algoritmen zoals [11, 12] kunnen ook geïmplementeerd en mogelijk beter geïntegreerd worden.

Daarnaast zou het goed zijn om de beschikbare data in het RU-corpus[5] te gebruiken om een vertaalsysteem als dit op een systematische en experimentele wijze te testen. Veel van het materiaal is geannoteerd met zowel glossen als een 'enge' vertaling[6] naar het Nederlands. Met de aanname dat deze handmatige vertalingen goed de betekenis van de zin overbrengen, kunnen ze voor het programma als input gebruikt worden, waarna de output in glossen vergeleken kan worden in betekenis met het origineel. Om dit redelijk uit te voeren, zal echter de reikwijdte van het programma eerst uitgebreid moeten worden, omdat het merendeel van de zinnen anders eerst sterk aangepast moet worden om aan de inputtaal van het programma te voldoen, wat de testdata alsnog synthetisch maakt.

[5] Zie https://www.ru.nl/corpusngt.

[6] Een vertaling die niet alle nuances van het origineel bevat, maar vooral letterlijk de betekenis overbrengt.

HOOFDSTUK 6

Conclusie

Tijdens dit project is onderzocht in hoeverre het Grammatical Framework (GF) kan dienen als basis voor een vertaalprogramma tussen een fragment van het Nederlands en de Nederlandse Gebarentaal (NGT). Daarin zijn twee aparte fragmenten behandeld: een minimale basis (met de talen NLb en NGTb) en deze basis uitgebreid met persoonlijk voornaamwoorden (de talen NLpv en NGTpv).

Voor deze formele talen zijn formele grammatica's geschreven volgens het *multiple context-free grammar*-formalisme (MCFG). Dit is het achterliggende formalisme voor het GF. Daaruit zijn programma's geïmplementeerd met behulp van het GF om tussen deze talen vertalingen uit te voeren, in de richting van NLb naar NGTb en NLpv naar NGTpv.

Uit de evaluatie van deze programma's blijkt dat correcte inputzinnen correct vertaald worden, alleen dat de bestaande grammatica's overgenererend zijn. Dit betekent dat er in de inputtaal zinnen zitten die volgens de brontaal (Nederlands) niet correct zijn. De grammatica's zouden verder uitgewerkt moeten worden om deze traditioneel ongrammaticale gevallen uit te sluiten.

Daarnaast schieten de huidige implementaties vooral tekort in de reikwijdte van het fragmenten van de brontalen (Nederlands en NGT) die ze implementeren. Er zijn nog relatief weinig zinsconstructies mogelijk, wat het niet praktisch maakt om het programma op niet-synthetische[1] input te testen. Ook voor inputzinnen die alleen de ondersteunde constructies bevatten, is nog voorverwerking nodig om *coreference resolution* plaats te laten vinden. Dit vindt plaats buiten het GF, omdat het niet met louter grammaticale informatie uit te voeren is.

[1] Hiermee wordt bedoeld: input niet niet speciaal voor dit doeleinde gegenereerd is.

Het is niet duidelijk of het GF buiten de geïmplementeerde zinsvormen ook geschikt is om een vertaling van Nederlands naar NGT uit te voeren. Dit is omdat enkele ontbrekende constructies die voortbouwen op de huidige resultaten moeilijk te implementeren zijn door middel van formele

grammaticaregels. Het verschil met *coreference resolution* is dat dit wel grammaticale eigenschappen zijn, maar met grote contextgevoeligheid. Mogelijk is er daarom een geschiktere methode dan één formele grammatica implementeren die al deze constructies kan genereren. Hiermee zou het vertalen een proces met meerdere stappen worden, waarbij de meer geavanceerde *features* van het GF mogelijk wel uitkomst zouden kunnen bieden.

Op deze ingewikkelde constructies na zijn er wel veel aspecten van beide talen die goed binnen het GF geïmplementeerd zouden kunnen worden. Het is echter belangrijk voor een vertaalprogramma dat het een bijna complete dekking van de brontaal heeft, en daarom zou het verstandig zijn om ook op basis van de ingewikkeldere constructies een beslissing te maken. Om te bepalen of het GF hiervoor geschikt is is daarom verder onderzoek nodig. Een belangrijke stap hierin zou zijn om te onderzoeken of het GF met de hier gedefiniëerde 'impliciete localisatie' om kan gaan.

De vertaalprogramma's zijn online te testen via `http://nardilam.nl/ngtpv`. Daar is ook de code te downloaden.

Bibliografie

[1] Beppie van den Bogaerde and Trude Schermer. Deaf studies in the Netherlands. *Forest*, 23:1–2, 2007.

[2] Corline (eds.) Schermer, Trude; Koolhof. *Van Dale Basiswoordenboek Nederlandse Gebarentaal*. Van Dale, 2009.

[3] Aarne Ranta. Grammatical framework, a type-theoretical grammar formalism. *The Journal of Functional Programming*, 14(2):145–189, 2004.

[4] Alexander Clark. An introduction to multiple context free grammars for linguists. http://www.cs.rhul.ac.uk/home/alexc/lsa2015/mcfgsforlinguists.pdf, 2015.

[5] G.M. Schermer. *De Nederlandse gebarentaal*. De Nederlandse Stichting voor het Dove en Slechthorende Kind, 1991.

[6] Iris Hendrickx, Gosse Bouma, Frederik Coppens, Walter Daelemans, Veronique Hoste, Geert Kloosterman, Anne-Marie Mineur, Joeri Van Der Vloet, and Jean-Luc Verschelde. A Coreference Corpus and Resolution System for Dutch. In *LREC*. Citeseer, 2008.

[7] Krasimir Angelov. *The mechanics of the Grammatical Framework*. Chalmers University of Technology, 2011.

[8] Onno Crasborn, Richard Bank, Inge Zwitserlood, Els Van der Kooij, Anne De Meijer, and Anna Safar. Annotation conventions for the Corpus NGT. *Ms., Radboud University Nijmegen*, 2015.

[9] Ingeborg Caren van Gijn. *The quest for syntactic dependency: Sentential complementation in Sign Language of the Netherlands*. PhD thesis, LOT Utrecht, 2004.

[10] Heleen et al. Bos. Pronoun copy in Sign Language of the Netherlands. 1995.

[11] Jerry R Hobbs. Resolving pronoun references. *Lingua*, 44(4):311–338, 1978.

[12] Shalom Lappin and Herbert J Leass. An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561, 1994.