Asymmetrische afstanden in Optimaliteitstheorie

Dicky Gilbers & Petra Hendriks[1]

1. Introductie

Is het een wiskundige misvatting dat de afstand tussen A en B gelijk is aan de afstand tussen B en A? Elke Groninger weet dat de afstand Groningen-Amsterdam voor Groningers korter is dan de afstand Amsterdam-Groningen voor Amsterdammers. Ook in de taalkunde blijken afstanden asymmetrisch te kunnen zijn. Bijvoorbeeld de afstand tussen het Zweeds en het Deens. Denen en Zweden kunnen in hun eigen taal met elkaar communiceren, maar een Deen verstaat een Zweed beter dan een Zweed een Deen (Gooskens, 2006; Moberg, Gooskens, Nerbonne & Vaillette, 2006). Aan de hand van de lettergreepstructuren van verschillende talen zullen we in dit artikel laten zien wat voor consequenties asymmetrie in taal heeft voor taalverwerving in het algemeen, en voor leeralgoritmes binnen de optimaliteitstheorie van Prince en Smolensky (1993) (verder OT) in het bijzonder.

Allereerst zullen we in paragraaf 2 illustreren hoe verschillen tussen talen in OT gemodelleerd worden als verschillende rangschikkingen van dezelfde verzameling van universele constraints. Daarna bespreken we in paragraaf 3 de twee meest bekende leeralgoritmes binnen OT en laten we zien dat deze twee leeralgoritmes verschillende voorspellingen doen met betrekking tot het leren van een tweede taal op basis van een eerste taal. Het leeralgoritme van Tesar en Smolensky (1998) is asymmetrisch en staat daardoor toe dat de afstand tussen taal A en taal B verschilt van de afstand tussen taal B en taal A. Gebruikmakend van deze asymmetrie doen we in paragraaf 4 voorspellingen over de relatieve leerbaarheid van lettergreepstructuren in het Nederlands, Hawaiiaans en Berber, beginnend vanuit een van deze talen en eindigend bij een andere. In paragraaf 5 presenteren we ondersteuning voor onze hypothese van asymmetrische afstanden tussen talen op basis van de hierboven genoemde mogelijkheid dat de onderlinge verstaanbaarheid van twee talen asymmetrisch kan zijn. Paragraaf 6 bespreekt, uitgaande van deze asymmetrische afstand tussen talen, hoe het leren van een tweede taal bespoedigd zou kunnen worden. In paragraaf 7, tenslotte, keren we terug naar bovengenoemd verschil in afstand tussen Amsterdam en Groningen, maar dan vanuit taalkundig perspectief.

2. Mogelijke talen

Een taaltheorie met een zeer breed domein van toepassing is OT. OT wordt niet meer alleen toegepast binnen het domein van de fonologie, waarin het oorspronkelijk werd geïntroduceerd (Prince & Smolensky, 1993), maar ook binnen de syntaxis, semantiek en pragmatiek. Binnen deze domeinen doet OT voorspellingen over de structuur, verwerving en verwerking van verschillende aspecten van taal. De voorspellende kracht van OT zit in de mogelijke rangschikkingen van universele constraints.[2] Om een voorbeeld te geven: elke taal combineert de klinkers en medeklinkers, die samen de klankeninventaris van de taal vormen, tot lettergrepen, maar niet elke taal doet dat op dezelfde wijze. OT stelt vast wat universele constraints zijn die de vorm van lettergrepen bepalen. Als we die constraints in alle mogelijke rangschikkingen weergeven moeten ze de mogelijke en onmogelijke lettergreepstructuren in de talen

van de wereld beschrijven. De constraints zijn dus universeel, maar de rangschikkingen geven de variatiemogelijkheden tussen talen aan.

Talen als het Nederlands en het Engels kennen clusters van medeklinkers binnen een lettergreep: *streep, split, darts.* Daarnaast hebben deze talen ook lettergreepstructuren waarin een vocaal hoogstens door één medeklinker voorafgegaan wordt: *papa, zo, mee.* Deze laatste combinaties van medeklinker gevolgd door klinker komen in alle talen van de wereld voor en zijn daarom het minst gemarkeerd. Er bestaan talen die ongemarkeerde kenmerken hebben en geen gemarkeerde, maar er bestaan geen talen die gemarkeerde kenmerken hebben maar geen ongemarkeerde. Er bestaat dus geen taal die wel woorden als *stroop* heeft en geen woorden als *papa.* De relevante universele OT-constraint wordt daarom als volgt geformuleerd: *NoComplex: medeklinkerclusters zijn niet toegestaan*. Dat het Nederlands en het Engels toch clusters kennen, komt doordat in deze talen *NoComplex* relatief laag gerangschikt is en dus eerder geschonden kan worden.

In het Hawaiiaans komen clusters van medeklinkers niet voor: *kanaka* 'man'; *wahine* 'vrouw'; *alapine* 'vaak'. *NoComplex* is in deze taal een hoog gerangschikte constraint. Verder eindigen in het Hawaiiaans alle lettergrepen op een klinker: *NoCoda: medeklinkers zijn verboden aan het eind van een lettergreep.* De werking van deze constraints is te illustreren aan de hand van leenwoorden uit het Engels in het Hawaiiaans: *we**lew**eka* uit Engels *ve**l**v**e**t* 'fluweel' en ***pal**aoa* uit Engels ***fl**our* 'meel' (Archangeli, 1997).[3]

Betekent dit nu dat het Hawaiiaans een makkelijkere taal is dan het Nederlands? Ja en nee. De lettergreepstructuur is minder complex en het lijkt er daarom op dat voor een Nederlander de Hawaiiaanse lettergreepstructuur makkelijker te leren zal zijn dan de Nederlandse voor de Hawaiiaan, maar gemarkeerdheid is niet de enige drijfkracht in de woordvorming van een taal. Om communicatief te zijn moet je heel veel verschillende betekenissen kunnen uiten en ontkom je niet aan complexiteit. De manier waarop dat gebeurt, verschilt echter per taal. De trekkervis heet in het Hawaiiaans *humuhumunukunukuapua'a.* Complexiteit en daarmee de communicatief gezien nodige betekenisverschillen worden in de taal vaak gevonden in de morfologische opbouw en daarmee in de lengte van de woorden.

Ook het Mandarijn Chinees kent geen clusters van medeklinkers. De complexiteit wordt hier gevonden in de verschillen in toon: *ma* met een hoge toon betekent 'moeder', met een hoge stijgende toon 'hennep', met een hoge dalende toon 'feeks' en met een dalende en weer stijgende toon 'paard' (Katamba, 1989, p.188). Het Engels en het Nederlands daarentegen staan zowel in het midden als aan de uiteinden van woorden medeklinkerclusters toe: ***spr**ing, a**bstr**act*, *a**rts***.

Nog extremer is de lettergreepstructuur van het Imdlawn Tashlhiyt dialect van het Berber, een Marokkaans dialect (Dell & Elmedlaoui, 1985). In deze taal is het niet noodzakelijk dat de kern van een lettergreep een klinker bevat en kom je dus vormen tegen als *txdmt* 'hout sprokkelen' en *trglt* 'slot'.

OT laat zien dat het een goede theorie is, als het mogelijk is om met herschikking van een beperkt aantal universele constraints alle mogelijke taalvariatie in lettergreepstructuur te beschrijven en slechts alle. Ook dat laatste is van groot belang. Geen enkele taal kent woorden waarbij alle medeklinkers aan het begin van het woord staan en alle klinkers aan het eind. Woorden als *nmtpooaoee* komen niet voor, woorden als *onomatopee* wel. Ook al is de beschrijving van zo'n taal voor een taalkundige heel gemakkelijk, het mag niet zo zijn dat een mogelijke constraint-rangschikking een dergelijke structuur prefereert boven alle andere mogelijkheden. De taalwetenschapper wil weten welke patronen in natuurlijke talen voorkomen, hoe

we die patronen kunnen karakteriseren en hoe we patronen kunnen uitsluiten die we niet vinden in talen en waarvan we vermoeden dat ze niet bestaan.

3. Verwerving van de grammatica

We hebben gezien dat constraints meer of minder belangrijk kunnen zijn in een bepaalde taal. Om een taal te leren wordt aangenomen dat iemand die een taal leert de rangorde van de constraints moet leren. Met andere woorden, een taalleerder moet leren welke constraints belangrijk zijn, en welke constraints minder belangrijk zijn. Hiervoor zijn verschillende leeralgoritmes voorgesteld. We zullen hieronder het Error-Driven Constraint Demotion Algorithm van Bruce Tesar en Paul Smolensky (1998) en het Gradual Learning Algorithm van Paul Boersma (1998) bespreken. We zullen zien dat deze twee leeralgoritmes verschillende voorspellingen doen voor het leren van een tweede taal.

3.1 Het Error-Driven Constraint Demotion Algorithm

Een belangrijke vraag in de taalverwerving is hoe kinderen in staat zijn om zich in zo korte tijd een zo complex iets als een grammatica eigen te maken. Dit is geen eenvoudige taak, omdat kinderen alleen te horen krijgen welke taalvormen wel mogelijk zijn volgens de te leren grammatica. Ze krijgen niet te horen welke taalvormen *niet* mogelijk zijn volgens de grammatica. Omdat kinderen maar zelden gecorrigeerd worden door ouders en andere volwassenen, lijkt het erop dat de informatie die kinderen krijgen over de te leren taal eigenlijk te beperkt is om er de grammatica uit te kunnen destilleren. Dit wordt wel het logische probleem van taalverwerving genoemd.

Tesar en Smolensky (1998) stellen een leeralgoritme voor dat in staat is om uit een gehoorde taalvorm toch informatie af te leiden over welke taalvormen niet mogelijk zijn. Daarbij maken ze gebruik van het feit dat als in OT een vorm optimaal is, de concurrerende vormen blijkbaar minder goed zijn en dus minder goed voldoen aan de constraints van de grammatica. Dit levert het taalverwervende kind informatie op over de te leren rangschikking van de constraints. Op basis van deze informatie kan het kind de rangschikking van de constraints van de grammatica bijstellen.

Dit zullen we illustreren aan de hand van een eenvoudig voorbeeld. Stel dat het kind over drie constraints beschikt, in de rangschikking *NoComplex* >> *NoCoda* >> *Parse*, d.w.z. dat *NoComplex* het sterkst is, en dan *Nocoda*, en dan *Parse* (*alle klanksegmenten uit de input, d.w.z. de opgeslagen vorm, worden gerealiseerd in de output)*. Stel nu dat iemand *stroop* tegen het kind zegt. Op basis van dit woord kan het kind concluderen dat de constraint *NoComplex* blijkbaar te hoog gerangschikt is in zijn grammatica. Immers, volgens de door het kind gehanteerde rangschikking van de constraints verwachten we geen clusters van medeklinkers binnen een lettergreep, zoals in *stroop*, maar zouden we eerder iets als *seterope* te horen moeten krijgen. Maar omdat het kind *stroop* hoorde, kan het kind concluderen dat het verbod op clusters van medeklinkers door *NoComplex* blijkbaar minder belangrijk is dan hij op basis van zijn grammatica dacht. Het woord *stroop* levert het taalverwervende kind dus informatie op over de te leren rangschikking van de constraints: als dit woord een mogelijk woord is in de te leren taal, dan staat *NoComplex* te hoog in zijn rangschikking. Kortom, de gehanteerde constraint-rangschikking is fout en moet dus

worden aangepast. Als er geen fout wordt geconstateerd, wordt de rangschikking niet aangepast. In dit opzicht is het leeralgoritme 'error-driven'.

Volgens het leeralgoritme van Tesar en Smolensky gebeurt nu het volgende. Als een constraint te hoog gerangschikt staat, dan wordt deze constraint een stapje naar beneden verplaatst ('constraint demotion'). De resulterende rangschikking is dan: *NoCoda* >> *NoComplex* >> *Parse*. Deze rangschikking wordt aangenomen totdat eventueel opnieuw een fout wordt ontdekt die de taalleerder dwingt tot herrangschikking van de constraints. Op deze manier wordt stap voor stap de uiteindelijke rangschikking van de constraints bereikt. Belangrijk is dat een constraint alleen naar beneden wordt verplaatst als er bewijs is dat de constraint te hoog gerangschikt staat. Op deze manier wordt voorkomen dat het leeralgoritme 'doorschiet' en een of meer constraints te laag gerangschikt worden.

Volgens Tesar en Smolensky levert alleen constraint-demotie precies de juiste rangschikking op. Zou je op basis van het bovenstaande voorbeeld constraints omhoog gaan verplaatsen in plaats van naar beneden, dan weet je op basis van het gehoorde woord *stroop* nog niet of uiteindelijk *NoCoda* boven *NoComplex* moet belanden, of *Parse*, of wellicht beide constraints. Een verkeerde constraint-rangschikking wordt voorkomen door constraints naar beneden te verplaatsen in plaats van omhoog.

3.2 Het Gradual Learning Algorithm

Boersma's (1998) Gradual Learning Algorithm (zie ook Boersma & Hayes, 2001) verschilt in twee belangrijke aspecten van Tesar en Smolensky's leeralgoritme. Ten eerste wordt aangenomen dat er niet alleen sprake is van een hiërarchische rangschikking tussen constraints, maar tevens dat de constraints op een lineaire schaal geplaatst zijn, zodat het mogelijk is om de afstanden tussen constraints te bepalen. Sommige constraints liggen dichter bij elkaar dan andere constraints. Het is zelfs mogelijk dat er sprake is van overlap tussen constraints, waardoor variatie binnen een taal kan worden verklaard. Het leeralgoritme is om deze reden 'gradual'. Leren gebeurt niet stapsgewijs, maar gradueel. Een constraint die herrangschikking ondergaat, schuift een stukje op langs de lineaire schaal, maar passeert hierbij niet noodzakelijkerwijs altijd een andere constraint.

Een tweede verschil met Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm is dat constraints zowel naar beneden als naar boven verplaatst kunnen worden. Als er een verschil wordt ontdekt tussen de vorm die de taalleerder hoort en de vorm die optimaal zou zijn volgens de grammatica van de taalleerder, dan worden niet alleen de constraints die geschonden worden door de gehoorde vorm naar beneden verplaatst. Gelijktijdig worden alle constraints naar boven verplaatst waaraan voldaan wordt door de vorm die ten onrechte als optimaal wordt beschouwd volgens de grammatica van de taalleerder. Op deze manier wordt het waarschijnlijker gemaakt dat de gehoorde vorm de volgende keer optimaal is, en wordt het minder waarschijnlijk gemaakt dat de vorm die ten onrechte optimaal was de volgende keer weer optimaal is.

3.3 Verschillende voorspellingen

Hoewel de twee leeralgoritmes allebei kunnen verklaren hoe taalleerders de uiteindelijke rangschikking van hun moedertaal leren, doen ze in een aantal gevallen verschillende voorspellingen. Verschillende voorspellingen worden onder andere gedaan voor het leren van een tweede taal. Elke taal wordt bepaald door de unieke rangschikking van dezelfde universele verzameling van constraints. Het leren van een taal bestaat dus uit het leren van de specifieke rangschikking van de constraints voor die taal. Voor het volgende nemen we aan dat een taalleerder die een tweede taal leert, begint met de rangschikking in zijn moedertaal, en op basis hiervan de rangschikking van de tweede taal leert. Dit is geen oncontroversiële aanname, maar wel de sterkst mogelijke aanname. Een alternatieve aanname is dat taalleerders beginnen met dezelfde aanvankelijke rangschikking, waarin bijvoorbeeld gemarkeerdheidsconstraints zoals *NoComplex* hoger gerangschikt zijn dan Correspondence/Faithfulness constraints zoals *Parse* (zie paragraaf 4 voor een discussie van deze twee soorten constraints). Volgens deze aanname begin je voor elke te leren taal weer bij nul. Een nog zwakkere aanname is de aanname dat de aanvankelijke constraint-rangschikking volkomen willekeurig is. Tesar en Smolensky (1998) betogen overigens dat met behulp van hun leeralgoritme niet elke taal te leren is vanuit een volkomen willekeurige aanvankelijke rangschikking op basis van alleen positieve feedback (d.w.z. feedback over de vormen die mogelijk zijn in een taal, in tegenstelling tot negatieve feedback, waarbij expliciete informatie gegeven wordt over de vormen die niet mogelijk zijn in een taal).

Uitgaand van bovenstaande aannames doen de twee leeralgoritmes verschillende voorspellingen over het leren van een tweede taal op basis van een eerste taal. Dit zullen we illustreren aan de hand van een abstract voorbeeld, waarbij we een hypothetische taal Grotaal (beschreven door de constraint-rangschikking C1 >> C2 >> C3) en een andere hypothetische taal Amstertaal (beschreven door de constraint-rangschikking C2 >> C3 >> C1) aannemen. Stel dat een taalleerder Amstertaal spreekt, maar Grotaal wil leren. Neem daarnaast aan dat we beginnen met het naar beneden verplaatsen van de hoogste verkeerd gerangschikte constraint. In dit geval heeft de taalleerder volgens Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm vier stappen nodig om de uiteindelijke constraint-rangschikking te bereiken, waarbij elke stap resulteert in de omwisseling van de volgorde van twee constraints (in paragraaf 5 bespreken we een alternatieve, kortere, route). De constraint die een stap naar beneden is geplaatst ten opzichte van de voorgaande rangschikking, is telkens weergegeven in vet:

Van Amstertaal naar Grotaal:

0. C2 >> C3 >> C1
1. C3 **>> C2** >> C1
2. C3 >> C1 **>> C2**
3. C1 **>> C3** >> C2
4. C1 >> C2 **>> C3**

Maar als een taalleerder zou beginnen met Grotaal, en op basis hiervan Amstertaal zou willen leren, zou dat volgens Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm in slechts twee stappen gebeuren:

Van Grotaal naar Amstertaal:

0. C1 >> C2 >> C3
1. C2 **>> C1** >> C3
2. C2 >> C3 **>> C1**

Kortom, het Error-Driven Constraint Demotion Algorithm voorspelt dat het sneller gaat om Amstertaal te leren als je Grotaal spreekt, dan het omgekeerde. Van Grotaal naar Amstertaal hoeven we slechts twee stappen te nemen, terwijl we van Amstertaal naar Grotaal wel vier stappen nodig hebben. Deze asymmetrie ontstaat door de asymmetrie van het leeralgoritme. Omdat constraints alleen naar beneden verplaatst worden, is het leeralgoritme asymmetrisch en gevoelig voor de leerrichting (van taal A naar taal B of omgekeerd).

Het Gradual Learning Algorithm van Boersma, daarentegen, is niet asymmetrisch op deze manier. Constraints worden zowel naar boven als naar beneden verplaatst naar aanleiding van een geconstateerde fout in de rangschikking. De afstand van taal A naar taal B zal dan ook even groot zijn als de afstand van taal B naar taal A.

Het gevolg van dit verschil is dat de twee leeralgoritmes verschillende voorspellingen doen met betrekking tot het leren van een tweede taal. Voor Boersma's Gradual Learning Algorithm is het niet van belang of taal A geleerd wordt vanuit taal B, of omgekeerd. Volgens Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm, daarentegen, kan het uitmaken of je taal A leert vanuit taal B, of omgekeerd. In sommige gevallen is de afstand tussen A en B in de ene richting groter dan de afstand tussen A en B in de tegenovergestelde richting.

In de volgende paragraaf kijken we aan de hand van concrete talen en specifieke constraints in meer detail naar de voorspellingen die Tesar en Smolensky's algoritme doet voor het leren van lettergreepstructuur.

4. Afstanden tussen talen

Constraints op lettergreepstructuur geven uitdrukking aan de ongemarkeerde lettergreep: een medeklinker gevolgd door een klinker. Constraints als *Onset*: *elke lettergreep begint met een medeklinker* en de eerder genoemde *NoCoda: een lettergreep mag niet op een medeklinker eindigen* drukken dit inzicht uit. Ook *Peak: de kern van een lettergreep bestaat uit een klinker* drukt de ongemarkeerde structuur uit. Als aan al deze gemarkeerdheidsconstraints zou worden voldaan, zouden slechts lettergrepen als *pa* en *ma* in talen voorkomen. Correspondence/Faithfulness constraints zorgen ervoor dat variatie mogelijk is in de structuur om zo diversiteit in betekenis te krijgen. Talen moeten immers communicatief kunnen zijn. Het Nederlandse woord *broek* betekent iets anders dan *boek.* Om dit verschil in betekenis structureel mogelijk te maken, moeten de gemarkeerdheidsconstraints gedomineerd worden door een Correspondence/Faithfulness constraint die stelt dat zowel /b/ als /r/ gerealiseerd moeten worden, ook al betekent dat een schending van *NoComplex.* Deze constraint is de eerder genoemde *Parse: parseer elk inputsegment in de output.*

Met de OT-constraints moet het mogelijk zijn om de lettergreepstructuren van alle en slechts alle talen van de wereld te beschrijven. Verschillende rangschikkingen moeten de Nederlandse, Hawaiiaanse en Berberse structuur als optimale output geven, terwijl geen enkele volgorde van constraints vormen als *nmtpooaoee* als optimaal mag aangeven, omdat dergelijke vormen niet voorkomen (zie ook Archangeli, 1997).

Het Nederlands stelt de klinker als kern van de lettergreep verplicht, maar clusters en coda's zijn toegestaan en een onset niet verplicht. De constraintvolgorde is:

NL: *Peak >> Parse >> NoComplex >> NoCoda >> Onset*[4]

Het Hawaiiaans heeft als rangschikking:

HI: *Peak >> NoComplex >> NoCoda >> Parse >> Onset*

Om van de Nederlandse tot de Hawaiiaanse rangschikking te komen, moet *Parse* demotie ondergaan en onder *NoComplex* en *NoCoda* belanden. Dat kan in twee stappen.

NL: *Peak >> Parse >> NoComplex >> NoCoda >> Onset*
Peak >> NoComplex ***>> Parse*** *>> NoCoda >> Onset*
HI: *Peak >> NoComplex >> NoCoda* ***>> Parse*** *>> Onset*

Om van de Hawaiiaanse rangschikking tot de Nederlandse te komen, moeten zowel *NoComplex* als *NoCoda* onder *Parse* belanden. Dat zijn vier stappen. Het is dus een grotere inspanning om van het Hawaiiaans naar het Nederlands te gaan, dan andersom.

HI: *Peak >> NoComplex >> NoCoda >> Parse >> Onset*
Peak >> NoCoda ***>> NoComplex*** *>> Parse >> Onset*
Peak >> NoCoda >> Parse ***>> NoComplex*** *>> Onset*
Peak >> Parse ***>> NoCoda*** *>> NoComplex >> Onset*
NL: *Peak >> Parse >> NoComplex* ***>> NoCoda*** *>> Onset*

Het Berber staat medeklinkers toe als kern van de lettergreep. *Peak* staat dus laag:

Br: *Onset >> NoComplex >> Parse >> NoCoda >> Peak*

Om van de Nederlandse tot de Berberse rangschikking te komen, moeten alle constraints tot onder *Onset* verplaatst worden, te beginnen met de hoogst gerangschikte constraint *Peak*. In totaal zijn dat twaalf stappen:

NL: *Peak >> Parse >> NoComplex >> NoCoda >> Onset*
Parse ***>> Peak*** *>> NoComplex >> NoCoda >> Onset*
Parse >> NoComplex ***>> Peak*** *>> NoCoda >> Onset*
Parse >> NoComplex >> NoCoda ***>> Peak*** *>> Onset*
Parse >> NoComplex >> NoCoda >> Onset ***>> Peak***
NoComplex ***>> Parse*** *>> NoCoda >> Onset >> Peak*
NoComplex >> NoCoda ***>> Parse*** *>> Onset >> Peak*
NoComplex >> NoCoda >> Onset ***>> Parse*** *>> Peak*
NoCoda ***>> NoComplex*** *>> Onset >> Parse >> Peak*
NoCoda >> Onset ***>> NoComplex*** *>> Parse >> Peak*
Onset ***>> NoCoda*** *>> NoComplex >> Parse >> Peak*
Onset >> NoComplex ***>> NoCoda*** *>> Parse >> Peak*
Br: *Onset >> NoComplex >> Parse* ***>> NoCoda*** *>> Peak*

Volgens de demotiemethode kost het eveneens twaalf stappen om van het Berber tot het Nederlands te komen:

Br: *Onset >> NoComplex >> Parse >> NoCoda >> Peak*
NoComplex ***>> Onset*** *>> Parse >> NoCoda >> Peak*
NoComplex >> Parse ***>> Onset*** *>> NoCoda >> Peak*
NoComplex >> Parse >> NoCoda ***>> Onset*** *>> Peak*
NoComplex >> Parse >> NoCoda >> Peak ***>> Onset***
Parse ***>> NoComplex*** *>> NoCoda >> Peak >> Onset*
Parse >> NoCoda ***>> NoComplex*** *>> Peak >> Onset*
Parse >> NoCoda >> Peak ***>> NoComplex*** *>> Onset*
NoCoda ***>> Parse*** *>> Peak >> NoComplex >> Onset*
NoCoda >> Peak ***>> Parse*** *>> NoComplex >> Onset*
Peak ***>> NoCoda*** *>> Parse >> NoComplex >> Onset*
Peak >> Parse ***>> NoCoda*** *>> NoComplex >> Onset*
NL: *Peak >> Parse >> NoComplex* ***>> NoCoda*** *>> Onset*

Hoe is dit te verklaren? Anders dan bij het Nederlands ten opzichte van het Hawaiiaans is het niet helemaal duidelijk of het Berber complexer is dan het Nederlands. Enerzijds staat het Berber medeklinkers als peak toe, maar anderzijds kent het Nederlands clusters van medeklinkers. De complexiteit zit dus in verschillende aspecten van de lettergreepstructuur.

In de hierboven gegeven rangschikking van het Berber wordt de analyse van Prince en Smolensky (1993, p.17) aangehouden, met *Onset* hoger dan *Parse* en *Peak* (de vergelijkbare gemarkeerdheidsconstraint heet *Hnuc* bij Prince en Smolensky), om zo de realisatie [w̩] met een syllabisch liquida voor onderliggend /ul/ af te dwingen. Archangeli (1997, p.5) stelt echter dat klinkers wel degelijk ook woordinitieel voorkomen in het Berber: *ildi* ‘trekken’. Volgens de analyse van Prince en Smolensky zou dit woord als [j̩ldi] met een syllabische liquida moeten worden gerealiseerd. Stel nu dat Archangeli gelijk heeft en dat *Onset* helemaal niet dominant gerangschikt hoeft te worden in het Berber. In dat geval kost het slechts vijf stappen om van het Nederlandse systeem tot het Berberse te komen.

NL: *Peak >> Parse >> NoComplex >> NoCoda >> Onset*
Parse ***>> Peak*** *>> NoComplex >> NoCoda >> Onset*
Parse >> NoComplex ***>> Peak*** *>> NoCoda >> Onset*
Parse >> NoComplex >> NoCoda ***>> Peak*** *>> Onset*
Parse >> NoComplex >> NoCoda >> Onset ***>> Peak***
Br: *NoComplex* ***>> Parse*** *>> NoCoda >> Onset >> Peak*

Andersom blijkt de verwerving van het Nederlandse systeem vanuit het Berberse systeem nu ingewikkelder. De demotiemethode heeft hier vijftien stappen voor nodig, als we telkens weer uitgaan van aanpassing van de hoogst gerangschikte constraint:

Br: *NoComplex >> Parse >> NoCoda >> Onset >> Peak*
Parse ***>> NoComplex*** *>> NoCoda >> Onset >> Peak*
Parse >> NoCoda ***>> NoComplex*** *>> Onset >> Peak*
Parse >> NoCoda >> Onset ***>> NoComplex*** *>> Peak*
Parse >> NoCoda >> Onset >> Peak ***>> NoComplex***
NoCoda ***>> Parse*** *>> Onset >> Peak >> NoComplex*
NoCoda >> Onset ***>> Parse*** *>> Peak >> NoComplex*
NoCoda >> Onset >> Peak ***>> Parse*** *>> NoComplex*
Onset ***>> NoCoda*** *>> Peak >> Parse >> NoComplex*
Onset >> Peak ***>> NoCoda*** *>> Parse >> NoComplex*
Onset >> Peak >> Parse ***>> NoCoda*** *>> NoComplex*
Onset >> Peak >> Parse >> NoComplex ***>> NoCoda***
Peak ***>> Onset*** *>> Parse >> NoComplex >> NoCoda*
Peak >> Parse ***>> Onset*** *>> NoComplex >> NoCoda*
Peak >> Parse >> NoComplex ***>> Onset*** *>> NoCoda*
NL: *Peak >> Parse >> NoComplex >> NoCoda* ***>> Onset***

Een nadeel van de demotiemethode is dat de eis om telkens te beginnen met demotie van de hoogst gerangschikte constraint ertoe leidt dat *Onset* op den duur dominant wordt en vervolgens toch ook weer demotie moet ondergaan om zijn oorspronkelijk al lage positie in de rangorde in te nemen. Je zou toch zeggen dat de taalleerder geen evidentie nodig heeft om *Onset* te demoten als in beide talen Onsets niet verplicht zijn. Dit is een consequentie van de gehanteerde demotiemethode.

Een ander mogelijk nadeel vinden we in een vergelijking van de afstanden tussen taalsystemen. De afstand tussen het Hawaiiaans en het Berber lijkt groter dan die tussen het Nederlands en het Berber. Om van de Hawaiiaanse tot de Berberse rangschikking te komen, geeft de demotiemethode echter net als van het Nederlands naar het Berber vijf stappen:

HI: *Peak >> NoComplex >> NoCoda >> Parse >> Onset*
NoComplex ***>> Peak*** *>> NoCoda >> Parse >> Onset*
NoComplex >> NoCoda ***>> Peak*** *>> Parse >> Onset*
NoComplex >> NoCoda >> Parse ***>> Peak*** *>> Onset*
NoComplex >> NoCoda >> Parse >> Onset ***>> Peak***
Br: *NoComplex >> Parse* ***>> NoCoda*** *>> Onset >> Peak*

Betrekken we daar ook de terechte kritiek van Boersma (1998) op het gebrek aan robuustheid van het model van Tesar en Smolensky voor versprekingen en fouten

in het taalaanbod bij, dan kunnen we stellen dat het demotiemodel zeker nog een aantal nadelen heeft. Toch kunnen we concluderen dat alleen een model dat uitgaat van slechts demotie van constraints om van het ene systeem tot het andere te komen leidt tot mogelijk asymmetrische afstanden tussen de rangschikkingen, en dus tussen de taalsystemen.

In de volgende paragraaf zullen we zien dat de gebruikelijke wijze om dialectafstanden te meten met behulp van het Levenshtein-algoritme ons niet in staat stelt om asymmetrische afstanden tussen talen uit te drukken.

5. Het Levenshtein-algoritme

Kessler (1995) introduceert het Levenshtein-algoritme. Om de afstand tussen de uitspraakvarianten van een woord te berekenen, moeten de kosten bepaald worden in de vorm van toevoegingen, verwijderingen en vervangingen die minimaal vereist zijn om de ene uitspraak te wijzigen in de andere. Het volgende voorbeeld komt uit Heeringa (2004, p. 124).

afternoon gerealiseerd als [æəftənʉn] en als [æftərnun]

æəftənʉn	deletie van ə	1
æftənʉn	insertie van r	1
æftərnʉn	substitutie van ʉ/u	1
æftərnun		
Levenshtein-afstand		3

Nerbonne et al (1996), Heeringa (2004) en Heeringa en Joseph (2007) meten de Levenshtein-afstanden tussen Nederlandse dialecten. Gooskens en Heeringa (2004) doen hetzelfde voor Noorse dialecten. Het Levenshtein-algoritme is op verschillende manieren toe te passen. De veranderingen kunnen gemeten worden als veranderingen tussen klanksegmenten, zoals hierboven. Het verschil tussen *pak* en *bak* is dan even groot is als het verschil tussen *pak* en *lak*. Een alternatief is de veranderingen te meten op featureniveau, waarbij het verschil tussen *pak* en *bak* kleiner is dan het verschil tussen *pak* en *lak*. Heeringa (2004) maakt gebruik van logaritmische akoestische segmentafstanden. Op die manier voorkomt hij het probleem dat klanken qua features erg van elkaar kunnen verschillen, zoals bijvoorbeeld /l/ en /w/, terwijl ze akoestisch gezien erg op elkaar lijken. De afstand tussen de varianten *oude* en *olde* is ten onrechte erg groot als we een op (articulatorische) features gebaseerd Levenshtein-algoritme toepassen, terwijl de akoestische afstand tussen [l] en [w] slechts een klein verschil in de locusfrequentie van de tweede formant laat zien (zie Gilbers 2002). Heeringa's aanpak om akoestische afstanden te meten is dus te prefereren boven het gebruik van featureafstanden of klankafstanden.

De analyses die gebruik maken van Levenshtein-afstanden delen echter een groot nadeel als we dialecten met elkaar willen vergelijken: de afstanden zijn volstrekt symmetrisch. De afstand van [æəftənʉn] naar [æftərnun] is net zo groot als de afstand van [æftərnun] naar [æəftənʉn].

afternoon gerealiseerd als [æəftənʉn] en als [æftərnun]

æftərnun	substitutie van u/ʉ	1
æftərnʉn	deletie van r	1
æftənʉn	insertie van ə	1
æəftənʉn		
Levenshtein-afstand		3

Dit gegeven levert problemen op voor een beschrijving van de in de inleiding genoemde observatie dat Denen Zweden beter kunnen verstaan dan Zweden Denen, als ieder in zijn eigen taal communiceert (Gooskens 2007). Als de afstand tussen talen inderdaad asymmetrisch is, dan schiet het Levenshtein-algoritme tekort.

In OT worden verschillen tussen talen beschreven als verschillen in rangschikking van dezelfde constraints. Gebruik makend van het Constraint Demotion Algorithm van Tesar en Smolensky kan de afstand tussen talen beschreven worden als het aantal demotiestappen om van de ene rangschikking tot de andere te komen. We hebben hierboven al laten zien dat dit algoritme ons in staat stelt asymmetrische afstanden tussen talen te voorspellen.

6. Sluiproutes of omwegen in taalverwerving

Een interessante vraag die nu rijst betreft de praktische implicaties van bovenstaande hypothese van asymmetrische afstanden tussen talen. Wat kunnen we hier nu mee in de praktijk? Gemakshalve keren we daarvoor terug naar het abstracte en relatief eenvoudige voorbeeld uit paragraaf 3.3, de afstand van Amstertaal naar Grotaal en vice versa. In paragraaf 3.3 zagen we dat er vier stappen nodig waren om vanuit de constraint-rangschikking van Amstertaal die van Grotaal te bereiken:

Van Amstertaal naar Grotaal (optie 1):

0. C2 >> C3 >> C1
1. C3 **>> C2** >> C1
2. C3 >> C1 **>> C2**
3. C1 **>> C3** >> C2
4. C1 >> C2 **>> C3**

Behalve deze optie in vier stappen is er ook nog een andere, kortere, weg mogelijk, die slechts twee stappen vereist:

Van Amstertaal naar Grotaal (optie 2):

0. C2 >> C3 >> C1
1. C2 >> C1 **>> C3**
2. C1 **>> C2** >> C3

Bij optie 1 beginnen we met de hoogst gerangschikte constraint, C2, die we eerst onder C3 plaatsen, en vervolgens onder C1. Vervolgens dient C3 eerst onder C1 geplaatst te worden, en daarna onder C2, zodat C3 de laagst gerangschikte constraint wordt. Bij optie 2 beginnen we in omgekeerde volgorde, en plaatsen we eerst

constraint C3 naar beneden, en daarna C2. Deze volgorde levert een beduidend kortere route op. Onze stelling dat de afstand van Amstertaal naar Grotaal vier stappen beslaat was dus een simplificatie. Wanneer begonnen wordt met het naar beneden schuiven van de sterkste constraint C2 is de afstand inderdaad vier stappen, maar wanneer begonnen wordt met constraint C3 is de afstand slechts twee stappen. Andere opties dan de hierboven genoemde twee zijn er niet, aangezien constraints alleen naar beneden geplaatst mogen worden en niet naar boven, althans volgens Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm, en slechts één constraint per keer mag worden verschoven.

Hoe wordt nu bepaald welke constraint als eerste naar beneden wordt geplaatst, C2 of C3? Met andere woorden, welke factoren zorgen ervoor dat we tot de langere omweg worden gedwongen of een kortere sluiproute kunnen nemen? Volgens beide eerder genoemde leeralgoritmes (Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm en Boersma's Gradual Learning Algorithm) vindt leren plaats op 'error-driven' wijze. Alleen indien er een verschil is tussen de geobserveerde vorm en de vorm die optimaal zou moeten zijn volgens de grammatica van het kind, wordt de constraint-rangschikking aangepast. Dit houdt in dat het leerproces deels afhankelijk is van de vormen die het kind hoort.

Hoort de taalleerder een vorm die strijdig is met constraint C2, dan zal C2 naar beneden geplaatst worden (daarmee de omweg nemend in optie 1). In deze situatie mag de taalleerder er niet voor kiezen om constraint C3 naar beneden te plaatsen (zodat de sluiproute in optie 2 kan worden genomen). Voor de te hoge rangschikking van C3 is immers geen bewijs in de gehoorde vorm. Hoort de taalleerder daarentegen eerst een vorm die strijdig is met constraint C3, dan kan de sluiproute wel worden genomen en zal constraint C3 naar beneden worden geplaatst. Welke leerroute wordt genomen, de efficiënte sluiproute of de minder efficiënte omweg, wordt dus bepaald door de volgorde van de vormen die de taalleerder tegenkomt.

Hiermee hebben we een interessant theoretisch handvat om het leren van een tweede taal te versnellen. Kennen we de aanvankelijke en de te leren constraint-rangschikkingen (voor een bepaald taalverschijnsel), dan kunnen we vaststellen wat de kortste route is tussen die twee talen. Aan de hand van die kortste route kunnen we vervolgens bepalen welke constraint het eerst naar beneden geplaatst moet worden, en wat dus de aard moet zijn van de taaldata die de taalleerder in het eerste stadium van het leerproces tegenkomt. Willen we in bovenstaand voorbeeld de sluiproute bevorderen, dan zal de taalleerder dus zoveel mogelijk in aanraking dienen te komen met vormen die constraint C3 schenden. Is constraint C3 bijvoorbeeld de constraint *NoComplex*, dan zal de taalleerder zoveel mogelijk lettergrepen moeten tegenkomen met clusters van medeklinkers, zoals *stro* en *klei*. Op het moment dat duidelijk is dat C3 onder C1 is geplaatst, moet de taaldata juist zoveel mogelijk vormen bevatten die constraint C2 schenden. Indien C2 de constraint *NoCoda* is, dan zullen dat lettergrepen dienen te zijn die eindigen op een consonant, zoals *aap* en *boek*.

Een complicatie vormen lettergrepen die zowel C3 als C2 schenden, zoals het woord *stroop* indien we aannemen dat C3 inderdaad *NoComplex* is en C2 *NoCoda* (zoals in paragraaf 3.1 ter illustratie werd aangenomen). Welke constraint dient nu naar beneden verplaatst te worden, C3 of C2? Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm neemt (in tegenstelling tot Boersma's Gradual Learning Algorithm) aan dat in dit geval alleen de hoogst gerangschikte onterecht geschonden constraint naar beneden verplaatst wordt, in het geval van de route van Amstertaal naar Grotaal dus C2. Op deze manier zijn we dus op de minder efficiënte omweg terechtgekomen. Dit laat zien dat het efficiënt leren van een tweede taal niet

simpelweg een kwestie is van het aanbieden van vormen waarin de te leren taal zoveel mogelijk verschilt van de eigen taal. De precieze volgorde en de structuur van de aangeboden vormen zijn van essentieel belang voor de keuze voor de omweg of de sluiproute in het leerproces, en dus voor het efficiënt leren van een andere taal.

7. Amsterdam-Groningen v.v.

Laten we nu terugkeren naar het in de inleiding aangestipte verschil in afstand tussen Groningen-Amsterdam versus Amsterdam-Groningen. Zoals we eerder zagen, staan het Nederlands en het Hawaiiaans alleen klinkers toe als lettergreepkern. Het Imdlawn Tashlhiyt Berber sluit daarentegen geen enkele klank uit als kern van de lettergreep. Het lijkt erop dat per taal wordt vastgesteld welke mate van sonoriteit klanken moeten hebben om de kern van een lettergreep te kunnen innemen. Kabardian (Kuipers, 1960) staat alleen niet-hoge klinkers toe als kern, dus niet /i/ en /u/; Sanskrit (Whitney, 1889) staat alle klinkers toe als kern, net als het Nederlands en het Hawaiiaans, maar ook /r/; Lendu (Tucker, 1940) voegt /l/ als mogelijke kern daaraan toe en het Engels staat naast alle klinkers /r,l,n/ toe: *butter, bottle, button* (Blevins, 1995). Dit duidt op een glijdende sonoriteitsschaal tussen enerzijds Kabarian en anderzijds Berber met alle andere talen daartussen. In OT is dit te beschrijven door *Peak* op te delen in een gradueel schendbare constraint:

> *NoStopPeak >> NoFricPeak >> NoNasalPeak >> NoLateralPeak >> NoRhoticPeak >> NoHighVowelPeak*

Constraints als *Parse* (hier *Parseer Consonant als Peak)* kunnen op elk punt interveniëren om de lettergreeptypologie te beschrijven, maar de rangschikking van de opgedeelde *Peak*-componenten onderling blijft onveranderd.

Het Gronings kent syllabische nasalen (Bolognesi, p.c.): *moeten* /mutən/ [mutn̩], *pakken* /pɑkən/ [pɑkŋ̩], *kopen* /kopən/ [kopm̩] en mogelijk ook syllabische liquidae in woorden als *vader, bakker, boedel*.[5] Het Gronings realiseert *kopen* /kopən/ dus als [kopm̩], terwijl in bijvoorbeeld het Amsterdams de realisatie [kopə] optimaal is.[6]

> Gronings: *NoStopPeak >> NoFricPeak >>* ***Parse*** *>> NoNasalPeak >> NoLateralPeak >> NoRhoticPeak >> NoHighVowelPeak*
>
> Amsterdams: *NoStopPeak >> NoFricPeak >> NoNasalPeak >> NoLateralPeak >> NoRhoticPeak >>* ***Parse*** *>> NoHighVowelPeak*

De transformatie Gronings-Amsterdams kan worden gemaakt in drie stappen:

Gr.: *NoSP >> NoFP >> Parse >> NoNP >> NoLP >> NoRP >> NoHVP*
NoSP >> NoFP >> NoNP ***>> Parse*** *>> NoLP >> NoRP >> NoHVP*
NoSP >> NoFP >> NoNP >> NoLP ***>> Parse*** *>> NoRP >> NoHVP*
Am.: *NoSP >> NoFP >> NoNP >> NoLP >> NoRP* ***>> Parse*** *>> NoHVP*

De transformatie Amsterdams-Gronings daarentegen in negen:

Am.: *NoSP >> NoFP >> NoNP >> NoLP >> NoRP >> Parse >> NoHVP*
NoSP >> NoFP >> NoLP ***>> NoNP*** *>> NoRP >> Parse >> NoHVP*
NoSP >> NoFP >> NoLP >> NoRP ***>> NoNP*** *>> Parse >> NoHVP*
NoSP >> NoFP >> NoLP >> NoRP >> Parse ***>> NoNP*** *>> NoHVP*
NoSP >> NoFP >> NoRP ***>> NoLP*** *>> Parse >> NoNP >> NoHVP*
NoSP >> NoFP >> NoRP >> Parse ***>> NoLP*** *>> NoNP >> NoHVP*
NoSP >> NoFP >> NoRP >> Parse >> NoNP ***>> NoLP*** *>> NoHVP*
NoSP >> NoFP >> Parse ***>> NoRP*** *>> NoNP >> NoLP >> NoHVP*
NoSP >> NoFP >> Parse >> NoNP ***>> NoRP*** *>> NoLP >> NoHVP*
Gr.: *NoSP >> NoFP >> Parse >> NoNP >> NoLP* ***>> NoRP*** *>> NoHVP*

De demotiemethode laat opnieuw de asymmetrie tussen de systemen zien. De afstand van het Amsterdams naar het Gronings is groter dan andersom. Nadeel is wel dat het consequent toepassen van de regel om telkens de dominante constraint naar beneden te verplaatsen hier tussenfases oplevert die geen recht doen aan de sonoriteitshiërarchie.

8. Conclusie

In dit artikel hebben we laten zien dat de afstand tussen taal A en taal B groter kan zijn dan die tussen taal B en taal A. Daarvoor hebben wij gebruik gemaakt van de inherente asymmetrie van een binnen optimaliteitstheorie (OT) voorgesteld leeralgoritme, namelijk Tesar en Smolensky's Error-Driven Constraint Demotion Algorithm. Dit algoritme staat alleen toe dat constraints naar beneden verplaatst worden in de constraint-rangschikking, en niet naar boven. Dit heeft als gevolg dat het aantal leerstappen afhangt van de richting waarin geleerd wordt. We hebben dit idee toegepast op tweede-taalverwerving, en gekeken naar de voorspellingen die dit leeralgoritme doet met betrekking tot de relatieve snelheid van verwerving van een taal op basis van een andere taal. Daarnaast hebben we gesuggereerd dat asymmetrische afstanden wellicht ook een verklaring zouden kunnen bieden voor de geconstateerde asymmetrie voor wat betreft de onderlinge verstaanbaarheid van talen.

Noten

---

[1] Wij willen Herman Veenker bedanken voor commentaar op een eerdere versie van dit artikel. Petra Hendriks bedankt tevens NWO (projectnummer 277-70-005) voor de financiële ondersteuning.

[2] OT in een notedop: in OT bepalen constraints, een soort welgevormdheidscondities, wat grammaticaal is en wat niet, bijvoorbeeld of je een woord als *papa* nu als *pápa* of als *papá* moet realiseren. De mogelijke realisaties van een woord in een bepaalde taal worden geëvalueerd aan de hand van dergelijke constraints. Die constraints kunnen contrasterend zijn en tegengestelde claims leggen voor wat betreft de te prefereren realisatie. Conflicten worden daarbij opgelost door verschillen in belangrijkheid aan te nemen tussen de verschillende constraints. Vergelijk een en ander maar met verkeersregels. Rechts gaat voor, tenzij het verkeer van links op een voorrangsweg rijdt. Deze laatste regel wordt echter weer 'overruled' door de regel die stelt dat je moet wachten voor een rood stoplicht. In het verkeer is dus ook sprake van een verzameling hiërarchisch geordende regels die bepalen wie uiteindelijk voorrang heeft. Merk op dat de afzonderlijke verkeersregels niet hard maar zacht oftewel schendbaar zijn. Ze mogen echter alleen geschonden worden om aan een hoger gewaardeerde regel te kunnen voldoen.

De grote vernieuwing van OT in de taalkunde zit in het feit dat ook de taalkundige constraints zacht zijn. Een outputkandidaat kan grammaticaal zijn, zelfs als deze een of meer constraints schendt. Zo lang er geen betere kandidaat is, is de minst slechte de optimale. De op zich ongeordende universele constraints kennen een stricte dominantieverhouding in de grammatica van een taal. Talen verschillen van elkaar wat betreft de rangschikking van de constraints. Dat in het Nederlands *papa* als *pápa* wordt gerealiseerd en in het Frans als *papá* ligt dus aan het taalspecifieke verschil in belangrijkheid van in principe dezelfde constraints. Wat de taalleerder moet verwerven is dat in taal A constraint C1 prioriteit heeft over constraint C2, terwijl dat in taal B andersom kan zijn.

Alle constraints moeten wel universeel gemotiveerd worden. Stel je wilt een OT-analyse maken van hoe succesvol iemand in een bepaalde situatie is. De universele welgevormdheidscondities worden dan geformuleerd als "de sterkste wint", "de snelste wint", "de mooiste wint", "de slimste wint", "de rijkste wint", etc. De rangschikking zal per situatie verschillend zijn. Voor een universitaire studie zal wellicht de zwaartehiërarchie net iets anders liggen dan bij een auditie voor een film, en als je in een jungle achtervolgd wordt door een wild beest is de eigenschap de snelste te zijn misschien net iets belangrijker dan het feit dat je erg mooi bent, maar het kan nooit zo zijn dat een welgevormdheidsprincipe geformuleerd wordt als "de domste wint", ook al kan de domste er in een bepaalde situatie wel als optimale kandidaat uitkomen.

[3] Vergelijk ook het Japanse woord *garasu* dat afkomstig is uit het Nederlandse *glas*. De voor het Nederlands en het Engels laag gerangschikte constraint is blijkbaar dominant in talen als het Hawaiiaans en het Japans.

[4] Hier wordt geabstraheerd van eventueel gelijk geordende constraints (in OT aangegeven met ";").

[5] Een aantal talen/dialecten doorbreken deze sonoriteitsschaal volgens Blevins (1995). Central Carrier (Walker, 1979), bijvoorbeeld, staat klinkers, nasalen en fricatieven toe als kern van de lettergreep. De klankeninventaris van deze taal kent echter geen liquidae. Het Gronings kent wel liquidae, maar het is niet helemaal duidelijk of het dialect naast syllabische nasalen ook syllabische liquidae heeft. Zo niet, dan heeft het Gronings een defectief systeem dat niet op basis van de sonoriteitsschaal kan worden beschreven.

[6] Wij abstraheren hier van de progressieve plaatsassimilatie van de syllabische nasalen in het Gronings en de precieze onderliggende vorm van het meervoudssuffix (al dan niet met sjwa).

Bibliografie

Archangeli, Diana

1997 Optimality Theory: An Introduction to Linguistics. In: Archangeli, Diana & D. Terence Langendoen (eds). *Optimality Theory, An Overview*. Oxford: Blackwell: 1-32.

Blevins, Juliette

1995 The Syllable in Phonological Theory. In: Goldsmith, John A. (ed.) *The Handbook of Phonological Theory*. Oxford: Blackwell. 206-244.

Boersma, Paul

1998 *Functional phonology*. The Hague: Holland Academics Graphics.

Boersma, Paul & Bruce Hayes

2001 Empirical Tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 32, 45-86.

Dell, François & Mohamed Elmedlaoui

1985 Syllabic Consonants and Syllabification in Imdlawn Tashlhiyt Berber. *Journal of African Languages and Linguistics* 7: 105-130.

Gilbers, Dicky

2002 Conflicting phonologically based and phonetically based constraints in the analysis of /l/-substitutions. In: Beers, Mieke, Petra Jongmans & A. Wijnands (eds) *Netwerk Eerste Taalverwerving, Net-bulletin 2001*, Leiden 2002, p.22-40.

Gooskens, Charlotte

2006 Linguistic and extra-linguistic predictors of Inter-Scandinavian intelligibility. In: Weijer, Jeroen van de & Bettelou Los (eds.). *Linguistics in the Netherlands* 23, Amsterdam: John Benjamins, 101-113.

2007 The contribution of linguistic factors to the intelligibility of closely related languages. *Journal of Multilingual and multicultural development,* Vol. 28, No. 6: 445-467.

Gooskens, Charlotte & Wilbert Heeringa

2004 Perceptive evaluation of Levenshtein dialect distance measurements using Norwegian dialect data. In: *Language variation and Change*, 16, 3: 189-207.

Heeringa, Wilbert

2004 *Measuring Dialect Pronunciation Differences using Levenshtein Distance.* Doctoral Dissertation. University of Groningen. Groningen.

Heeringa, Wilbert & Brian Joseph

2007 *Identifying Conservative and Innovative Areas in the Dutch Dialect Landscape: a dialectometric study.* Ms. University of Groningen.

Katamba, Francis

1989 *An Introduction to Phonology*. Longman, London and New York.

Kessler, Brett

1995 Computational dialectology in Irish Gaelic. In: *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics.* EACL, Dublin, 60-67.

Kuipers, Aert H.

1960 *Phoneme and Morpheme in Kabardian.* The Hague: Mouton.

Moberg, Jens, Charlotte Gooskens, John Nerbonne & Nathan Vaillette
2006 Conditional entropy as a measure of linguistic remoteness between related languages. Ineke Schuurman et al. (eds.) *Proceedings of Computational Linguistics in the Netherlands 2006* Amsterdam: Rodopi.

Nerbonne, John, Wilbert Heeringa, Erik van den Hout, Peter van der Kooi, Simone Otten & Willem van de Vis
1996 Phonetic Distance between Dutch dialects. In: Gert Durieux, Walter Daelemans and Steven Gillis (eds.). *CLIN VI, Papers from the sixth CLIN meeting.* University of Antwerp, Center for Dutch Langiage and Speech, Antwerpen, 185-202.

Prince, Alan and Paul Smolensky
1993 *Optimality Theory: Constraint Interaction in Generative Grammar.* Technical Report CU-CS-696-93, Department of Computer Science, University of Colorado at Boulder, and Technical Report TR-2, Rutgers Center for Cognitive Science, Rutgers University, New Brunswick, NJ. Published by Blackwell in 2004.
2006 Optimality in phonology: I. Syllable structure. In: Smolensky, Paul & Geraldine Legendre (eds.). *The harmonic mind: From neural computation to optimality-theoretic grammar.* Vol. 2. Cambridge, MA: MIT Press: 3-25.

Tesar, Bruce and Paul Smolensky
1998 Learnability in Optimality Theory. *Linguistic Inquiry* 29, 229-268.

Tucker, A.N.
1940 *The Eastern Sudanic Languages,* vol. 1. London: Dawsons.

Walker, Richard
1979 Central Carrier phonemics. In: Zimmerly, D. (ed.). *Contributions to Canadian Linguistics*. Canadian Ethnology Service Paper No. 50. National Museum of Man Mercury Series. National Museums of Canada, Ottawa. 93-107.

Whitney, W.D.
1889 *Sanskrit Grammar.* Cambridge, MA: Harvard University Press.