De Nederlandse afbreekpatronen voor TEX

Geschiedenis en werking

Abstract
Dit artikel beschrijft de totstandkoming van de Nederlandse afbreekpatronen voor TEX, alsmede hoe deze patronen binnen een TEX-systeem gebruikt worden.

Keywords
TEX, woordafbreking, patronen, Unicode, UTF-8, patgen, PDFTEX, LuaTEX, XeTEX, OpenTaal, Groene Boekje

Inleiding

Het oorspronkelijke TEX van Donald Knuth was helemaal op het Engels gericht. De fonts bevatten de karakters die in het Engels en in de wiskunde voorkwamen. Woorden werden afgebroken volgens Engelse afbreekregels. Opmerkelijk is deze zin in het proefschrift van Frank Liang, die het afbreekalgoritme voor TEX ontwikkeld heeft: "One necessary component of the system is a computer-based algorithm for hyphenating English words."[15, p. 1]. Let op het woord 'English'.

Er was een mogelijkheid om de afbreekregels te vervangen door die van een andere taal, maar dan had je een nieuw TEX-programma daarvoor nodig (strikt gesproken een nieuw *format*). Knuth ontwikkelde TEX in eerste instantie om zijn eigen boeken (The Art of Computer Programming) te kunnen zetten. Het is in eerste instantie populair geworden doordat de American Mathematical Society (AMS) het geadopteerd heeft, maar dat was nog steeds een Engelstalige omgeving.

En dat gaf natuurlijk aanleiding tot de vraag naar afbreekpatronen voor andere talen, waaronder het Nederlands.

Ik, als niet-historicus, ga nu proberen eerst een historisch verantwoord overzicht te maken van de ontwikkeling van deze Nederlandse afbreekpatronen. Het maakt het iets makkelijker dat ik hier ook aan meegewerkt heb.

Als we de MAPS doorspitten, dan komen we in 1988 in MAPS nummer 2 het verslag van de tweede NTG-vergadering[27] het volgende tegen:

"

5.5 Nederlandse TEX

Veel belangstelling blijkt er te bestaan in een public-domainversie van de Nederlandse afbreekpatronen.

N. Poppelier vermeldt dat men in Utrecht bezig is om op een officiële manier uit een Nederlandse lijst van afgebroken woorden, patronen te genereren. Echter de te gebruiken woordenlijst blijkt niet public domain te zijn. Er zijn pogingen ondernomen om van een vrij te gebruiken woordenlijst uit te gaan.

T. Jurriens blijkt ook pogingen te ondernemen om patronen in het public domain onder te brengen.

H. Brouwer heeft via TEXHax afbreekpatronen voor de Nederlandse taal verkregen. Hij gaat na of deze vrij zijn. Implementatie heeft inmiddels plaatsgevonden. Voor 90-95% blijken afbrekingen correct te gebeuren.

P. Tutelaers had via mail een lijst van Nederlandse afbreekpatronen uit België ontvangen. Ook enkele andere aanwezigen bleken patronen van diverse bronnen verkregen te hebben. Inventarisatie is hard nodig en zou een goede start voor een nieuw op te richten werkgroep kunnen zijn.

Meerdere afbreekpatronen tegelijkertijd kunnen via een multi-language TEX afgehandeld worden. In TEXHax blijkt meer hierover vermeld te staan. G. de Jong heeft van de multi-language TEX een VAX implementatie welke geen problemen geeft. Hij heeft contact met de auteur ervan voor een eventuele verspreiding ervan in Nederland.

"

In dezelfde bijeenkomst:

"

Werkgroep 13: Nederlandse TEX Een nieuw te vormen werkgroep. Grote behoefte was er naar zaken als Nederlandse afbreekpatronen en Nederlandse style files.

"

Maar, let op: in de vergadering van de werkgroep op 9 november 1989 wordt gesteld:[2]

“

2.2 Afbreekpatronen
Dit gaat buiten de werkgroep om.

”

TEXHax is een mailing list van TUG, die al sinds 1986 bestaat. Archieven vanaf april 2000 staan op de TUG mailing list site[1]. Oude archieven van 1968-2003 staan op CTAN[2]. De laatste zijn niet doorzoekbaar. Ze staan per jaar in een map; ongeveer 100 gecomprimeerde bestanden per jaargang. Om te zoeken moet je ze downloaden, decomprimeren, en dan kun je lokaal zoeken.

Ik vond hiermee o.a. dat er naast Nederlands gevraagd werd naar afbreekpatronen voor Hebreeuws (inclusief bijbehorend font), IJslands, Hongaars, Spaans, Italiaans, Portugees, Frans, Zweeds, Deens, Noors en Duits (al aanwezig).

Zie bijvoorbeeld het bericht van Theo de Klerk (DEC) op 31 augustus 1988:

“

```
DUTCH HYPHENATION PATTERNS

To implement a proper version of LaTeX on our
machines, I'd like to know if there is any
dutch hyphenation pattern for TeX available
anywhere in netland.
```

”

Enige tijd later (22 september 1988) komt er een antwoord van Peter Vanroose (K.U. Leuven):

“

```
Some time ago, I constructed the file
HYPHEN_NEDERLANDS.TeX, which works fairly
well. It is even a lot smaller than
HYPHEN.TeX; this is not so stange (sic!),
because the splitting rules in dutch are
rather strict and consistent. The only caveat
is, that compound words (and those are in
principle unlimited in dutch) are sometimes
not splitted right. This of course can be
cured by inserting '\-'s in the text itself.
Fortunately, this is not happening too much.
It can be remarked that newspapers seem
to have the same "compound word splitting"
problem with their automatic splitter.
  Installation:
  Replace the file HYPHEN.TEX by the one given
below.
Run initex on the preferred format (e.g.
PLAIN.TEX or LPLAIN.TEX).
Rename the output .FMT file :
e.g. PLAIN.FMT ---> PLAIN_NEDERLANDS.FMT
Replace the original HYPHEN.TEX.
Define the commands
nTeX as "TeX &plain_nederlands",
nLaTeX as "TeX &lplain_nederlands".
```

”

Het bestand `HYPHEN_NEDERLANDS.TeX` was bijgesloten.

Ik weet niet of dit de patronen uit België zijn waar Piet Tutelaers naar verwijst, maar het lijkt mij wel waarschijnlijk.

Uit de verwoording leid ik af dat dit handgemaakte patronen zijn, gebaseerd op de Nederlandse spellingsregels van die tijd.

Deze patronen zijn in oktober 1989 ook te vinden op de NTG-fileserver[18].

“

```
* Dutch hyphenation patterns.
* Hyphen1.TeX: short but satisfactory.
               (author: Peter Vanroose)
* Hyphen1.doc: some notes by Peter Vanroose.

* Hyphen2.TeX:
* long and powerful. (author: Henk Pennings) (sic!)
* note that this requires stretching the 'triesize'
* of both TeX and IniTeX!
```

”

Op 10 december 1989 meldt Don Hosek op TEXHax dat op een ftp-server in Claremont de volgende afbreekpatronen staan:

DEHYPH.TEX	German Hyphenation
DEHYPHA.TEX	German Hyphenation with Umlaut
ISHYPHA.TEX	Icelandic Hyphenation
ITHYPH.TEX	Italian Hyphenation
NEHYPH.TEX	Dutch Hyphenation
PTHYPH.TEX	Portuguese Hyphenation
TKHYPHA.TEX	Turkish Hyphenation

De opmerking van Nico Poppelier dat “men in Utrecht bezig is om op een officiële manier uit een Nederlandse lijst van afgebroken woorden, patronen te genereren.” slaat op het werk waar Henk Penning en ik mee bezig waren en waar bovenstaande `Hyphen2.TeX` het resultaat van is. Of dit het bovengenoemde `NEHYPH.TEX` is weet ik niet. Ik zal het resultaat van ons werk in dit artikel aanduiden als de CELEX-patronen, om redenen die verderop duidelijk gemaakt worden.

Henk Penning is helaas in 2019 veel te jong overleden. Ik moet het daarom met mijn herinnering doen.

Ook op de ftp-server van Informatica op de Universiteit Utrecht (`archive.cs.ruu.nl`) waren ze in 1990 te vinden in de map `TEX`[30]:

"The following listing is the INDEX file in the TEX subdirectory. For information contact Piet van Oostrum <piet@cs.ruu.nl>"

```
...
... henkp 30763 Jun 15 12:22 hyphen.dutch.Z
...
```

'henkp' was de gebruikersnaam van Henk Penning. Hij was systeembeheerder bij het Departement Informatica.

Verder werden de afbreekpatronen gedistribueerd op de NTG-diskettes[29]:
We hebben daar diskette `NTG006C` (346.112 bytes) met de map `HYPPAT`:

```
VANROOSE.NOT Informatie+Nederlandse Hyphenation file
VANROOSE.HYP van Peter Vanroose (2760 bytes)
CELEX.NOT    Informatie+Noord-Vlaams Hyphenation
CELEX.HYP    pattern CELEX/RUU (59758 bytes)
```

en verder Nederlandse stijlen en nog wat extra's.

Het lijkt mij dat hier Nederlands en Vlaams verwisseld is (hoewel Vlaams ook Nederlands is). En wat Noord-Vlaams moet betekenen is mij onduidelijk. Ik denk dat Noord-Nederlands bedoeld is. Al vraag ik mij af of er echt verschil is tussen Vlaamse en Noord-Nederlandse afbreekpatronen.

De oude afbreekbestanden blijken nog bewaard te zijn in een obscuur hoekje van CTAN[3]: (De kolom Notes heb ik zelf toegevoegd.)

```
Name        Size    Date          Notes
nehyph1.tex 38 kB 1991-07-03 Ingekorte CELEX-patronen
nehyph2.tex 51 kB 1991-07-03 Originele CELEX-patronen
nehyph3.tex  2 kB 1991-07-03 Patronen Peter Vanroose
```

Het verschil tussen `nehyph2.tex` (51749 bytes) en `CELEX.HYP` (59758) is te verklaren doordat de bestanden op de diskettes in MS-DOS-formaat zijn waarbij een regelovergang twee bytes is en `nehyph2.tex` in Unix-formaat met één byte per regelovergang.

Merk op dat hier `ne` gebruikt is in plaats van `nl`. Dit is niet in overeenstemming met de officiële talencodes van IETF (Internet Engineering Task Force)[4]. Hierin is `ne` de code voor Nepalees, en `nl` voor Nederlands. Latere bestandsnamen gebruiken `nl`.

Tenslotte heeft Gerard Kuiken nog afbreekpatronen voor het Nederlands gemaakt. Voor zover ik weet zijn die niet in een distributie terecht gekomen. Hij heeft ze gedeeld op de TEX-NL mailing list onder de naam `NLHYPHEN.MAX`, o.a. in een bericht op 28 juli 1998, waar hij versie 1.10 deelt. Het archief van TEX-NL uit die tijd is, voor zover ik weet, verdwenen, maar ik had zelf een kopie van dit bericht op mijn eigen computer gearchiveerd. Deze patronen heeft hij met de hand gemaakt.

Gerard schrijft:[5]

"

Ik had PCTeX zelf gekocht. Later bleek dat er geen afbreekpatronen voor het Nederlands bestonden. Ik was ook bezig om een collegedictaat over Irreversibele Thermodynamica in het Nederlands te schrijven. Later in het Engels, uitgebreid met toepassingen voor diffusie en rheologie, in 1994 verschenen bij Wiley & Sons, 2de druk februari 1995.
Ik kreeg een licentie voor PCTeX voor de Benelux en de toegenomen vraag naar Nederlandse afbreekpatronen noodzaakte mij om ze te maken. Woorden die met nieuwe patronen correct worden afgebroken werden verzameld en vormden mijn testbestand, dat ik nu kwijt ben.

Om `zo-iets` toe te voegen, kijk ik eerst naar het kortste patroon `o1i`, maar dat geeft fouten met `fooi`, `plooi`, `gooi` etc, 'o1ie' geeft fouten met `fooien`, `plooien`, `strooien` etc.; 'zo1i' is mijn keus en tevens kan 'zo1a' voor `zo-als` worden toegevoegd, zodat dan het totaal aantal van 5785 patronen wordt vermeerderd met 1.

Ik had een klein DOS programmaatje geschreven dat alle patronen gaf van de woorden in mijn testbestand die van invloed konden zijn en mogelijk een foute afbreking zou geven tengevolge van nieuwe patronen.

"

Voor de statistieken is er in tabel 1 een overzicht van de verschillende sets Nederlandse patronen. Ter vergelijking de Engelse erbij.

Bestand	Aantal patronen	Grootte [kB]
CELEX	7928	51
Tutelaers	12724	83
Vanroose	326	1,7
Kuiken	5786	46
TEX (Engels)	4447	27

Tabel 1: Overzicht van afbreekpatronen.

In de rest van het artikel zal ik eerst uitleggen hoe de afbreekpatronen eruit zien, hoe TEX's afbreekalgoritme werkt, en hoe de patronen gebruikt worden in dat algoritme. Dan hoe ze gemaakt kunnen worden. Daarna zal ik beschrijven hoe de afbreekpatronen van Henk Penning en mij (de CELEX-patronen) tot stand gekomen zijn, wat de beperkingen ervan waren, en hoe de verdere ontwikkeling tot de tegenwoordig gebruikte patronen is verlopen. De huidige patronen zoals die in de

TEX-distributies voorkomen, zijn door Piet Tutelaers gemaakt, en die zal ik dan ook als de Tutelaers patronen aanduiden. Daarna zal ik nog wat vertellen over de coderingen van de patronen. Om tenslotte te eindigen met enkele overpeinzingen over de toekomst van dit alles.

TEX's afbreekalgoritme

Het huidige afbreekalgoritme in TEX is in 1980–1982 ontworpen door Frank Liang, een PhD student van Knuth. Computers waren in die tijd langzaam en hadden niet veel geheugen, dus een voorwaarde was dat het algoritme snel was en niet veel geheugen nodig had. Liangs algoritme voldoet hieraan. Bovendien is het algoritme onafhankelijk van welke taal gebruikt wordt. Het algoritme is niet 100% correct; het kan afbreekpunten missen, of soms ongewenste afbreekpunten toevoegen, maar deze missers zijn redelijk zeldzaam. Voor dit soort gevallen heeft TEX een 'uitzonderingslijst' waar deze moeilijke gevallen in aangegeven kunnen worden.

In de huidige uitzonderingslijst staat o.a. `wa-ter-staats-in-ge-ni-eur`, waar '`-`' een mogelijk afbreekpunt aangeeft. Liangs algoritme met de Tutelaers afbreekpatronen zou de volgende afbreekpunten genereren: `wa-ter-staat-s-in-ge-ni-eur`, dus een onterecht afbreekpunt tussen '`t`' en '`s`'. Aan de andere kant staat het woord `staats-in-ge-ni-eur` niet in de uitzonderingslijst, en als je dit woord zou gebruiken bestaat de kans dat TEX het afbreekt tussen de '`t`' en de '`s`'. Nu is dit een woord dat niet zo gauw gebruikt zal worden, dus zo erg is dit niet. Trouwens, op de site van de Taalunie, waar je woorden kunt controleren[6] wordt dit woord niet herkend. Wel wordt een soortgelijk woord gesuggereerd, nl. `staatsingrijpen`, en de afbreking van dit woord in TEX is ook fout, nl. `staat-s-in-grij-pen`. Daarentegen wordt het woord `staatsinrichting` wel goed afgebroken als `staats-in-rich-ting`. Verderop zullen we onderzoeken waardoor dit verschil veroorzaakt wordt.

In mijn Van Dale-woordenboek[31] zijn de enige woorden die met `staats` beginnen en een afbreking tussen de '`t`' en '`s`' hebben, de woorden met `staatsie` erin. Het zou in principe mogelijk moeten zijn om hiervoor aparte afbreekpatronen toe te voegen. Alleen is het soms makkelijker om woorden aan de uitzonderingslijst toe te voegen dan de patronenlijst aan te passen. In een TEX-document kun je zelf extra uitzonderingen toevoegen met het commando `\hyphenation`, bijvoorbeeld

```
\hyphenation{staats-in-ge-ni-eur staats-in-grij-pen}
```

Na het controleren van een woord in de uitzonderingslijst (waarbij alleen het gehele woord exact gecontroleerd wordt), komt het eigenlijke algoritme van Liang in beeld. Ik zal hier een beknopte uitleg van geven.

We beginnen met de structuur van de patronen. Hier is zo'n patroon uit het Tutelaers patronenbestand voor Nederlands (`hyph-nl.tex`): `.ge5r4e`

De '`.`' aan het begin betekent dat dit patroon alleen van toepassing is aan het begin van een woord; op dezelfde manier kan er ook een '`.`' aan het eind van een patroon staan. De letters geven aan op welk woorddeel het patroon van toepassing is, hier dus '`gere`' aan het begin van een woord; en de cijfers geven de (on)wenselijkheid van een afbreking op die plaats aan – dit is een cijfer van 0 t/m 9, waarbij 0 i.h.a. weggelaten wordt. Een even cijfer verhindert afbreking op dat punt; een oneven cijfer bevordert afbreking op dat punt. Hoe hoger het cijfer, hoe sterker de (on)wenselijkheid, m.a.w. een hoger cijfer annuleert een lager cijfer. Als op een bepaalde plaats in één patroon een 3 staat, en in een ander patroon een 4, dan wint de 4, en vindt er op die plaats dus geen afbreking plaats.

Conceptueel werkt het afbreekalgoritme als volgt:

1. eerst worden begin- en eindmarkeringen aan het woord toegevoegd (d.w.z. een '`.`')
2. dan worden alle substrings van het woord vergeleken met de patronen (kijk of de letters kloppen)
3. als een patroon klopt dan worden de cijfers ervan onthouden op de juiste plaats in het woord; als er meerdere zijn op een plaats dan wordt het maximum genomen
4. als het resultaat op een bepaalde plaats oneven is, dan is dat een mogelijk afbreekpunt

Dit klinkt als veel werk, en dat zou het ook zijn als het algoritme niet een slimme datastructuur zou gebruiken om dit veel efficiënter te maken. Deze datastructuur heet een *trie*[7]. Dit is een boom, waarbij elke node geïndexeerd wordt met een waarde (in dit geval een letter of '`.`') om de volgende node te vinden. In de nodes staan de cijfers van de patronen opgeslagen. Door vanaf het begin (de beginmarkering '`.`'), en dan telkens vanaf de volgende letter door de boom te wandelen en de volgende letter als index te gebruiken, controleer je efficiënt welke patronen passen.

In de TEX-implementatie wordt de *trie* dan nog gecompactificeerd in een lineaire structuur, maar het gaat te ver voor dit artikel om dit te beschrijven en het is ook irrelevant voor de rest van het verhaal.

Een uitgebreidere uitleg kun je vinden in appendix H van het TEXbook[13]. En als je de details wilt weten dan moet je Frank Liangs proefschrift[15] bestuderen. En als je het nog gedetailleerder wilt, bekijk dan de originele TEX-code in 'TEX: The Program'[14, Part 42 en 43, sectie 919–966, pp. 386–402]. Er zijn ook andere implementaties van dit algoritme beschikbaar waarmee je kunt experimenteren[8].

In onderstaand voorbeeld heb ik het woord `afbreken` door het algoritme met de Tutelaers patronen gehaald. Dit geeft het volgende resultaat waarbij ik de alleen de patronen die passen weergeef. Ik heb spaties tussen de letters gezet om plaats te maken voor de cijfers.

```
. a f b r e k e n .
. a4
. a f3
   4f b
     1b
     3b r4
           1k e
               4n .
  ---------------
   4 3   4 1   4
  a f-b r e-k e n
```

Er zitten wat interessante patronen bij, bijvoorbeeld '`1b`': dit zegt dat er vóór een '`b`' afgebroken mag worden, maar niet heel sterk. Er zijn dan weer andere patronen die dat in specifieke gevallen verhinderen, zoals '`4bt4`', dat zegt dat voor en na '`bt`' niet afgebroken mag worden, tenzij er natuurlijk nog weer een passend patroon zou zijn met een 5 erin, maar die zitten er niet in.

Het volgende patroon is '`3br4`'; dit breekt af voor '`br`', en is zelfs nog iets sterker. Voor '`br`' wordt altijd afgebroken en erna nooit; er zitten geen patronen in die nog sterker zijn.

Let op: deze patronen zijn gegenereerd (zie verderop); er is dus niet over nagedacht, in tegenstelling tot de patronen van Peter Vanroose en Gerard Kuiken.

Dan nu de vraag waarom `staatsingenieur` niet goed wordt afgebroken, maar `staatsinrichting` wel. Ik zal niet alle patronen laten zien die toegepast worden (15 voor `staatsingenieur` en 16 voor `staatsinrichting`), maar alleen degene die relevant zijn voor dit probleem.

Er is een patroon '`1si`' dat in beide gevallen van toepassing is tussen de '`t`' en de '`s`'. Daarom komt daar bij `staatsingenieur` een afbreekpunt. Maar bij `staatsinrichting` is ook het patroon '`4sinr`' van toepassing, dat zegt dat voor '`sinr`' niet afgebroken moet worden. De 4 annuleert de 1. Bij `staatsingenieur` of `staatsingrijpen` is er niet een vergelijkbaar patroon.

```
. s t a a t s i n r i c h t i n g .
          1s i
          4s i n r
```

Overigens breken zowel de CELEX-patronen en die van Peter Vanroose `staatsingenieur` op dezelfde manier verkeerd af. De patronen van Gerard Kuiken doen het wel goed, maar die hebben dan ook een patroon '`2s3ingeni`'. De CELEX-patronen en die van Gerard Kuiken breken `staatsinrichting` goed af, maar die van Peter Vanroose maken hier foutief `staat-sin-rich-ting` van.

Het is enigszins triest dat het woord `afbreekalgoritme` afgebroken wordt als `af-breekal-go-rit-me`, waarbij het afbreekpunt tussen beide woorddelen `afbreek` en `algoritme` dus niet opgemerkt wordt. Het mag een schrale troost zijn dat de overige afbreekpunten wel goed zijn. De afbreekpatronen van Peter Vanroose maken er `af-bree-kal-go-rit-me` van. Maar deze patronen zijn ook veel kleiner (d.w.z. minder in aantal), en het is dan ook niet vreemd dat ze het minder goed doen op dit soort ongebruikelijke woorden.

Het moet opgemerkt worden dat er in het Nederlands (en bijv. ook in het Duits) gevallen van afbreking zijn, waarbij een woord veranderd wordt als het afgebroken wordt. Voorbeelden zijn:

`omaatje` → `oma-/tje`, `menuutje` → `menu-/tje`,
`cafeetje` → `café-/tje`, `AOW'er` → `AOW-/er`,
`beëindigen` → `be-/eindigen`,
waarbij '`/`' de regelovergang aangeeft.

TEX's afbreekalgoritme heeft niet de mogelijkheid om dit automatisch te doen, en het kan dus ook niet met behulp van patronen weergegeven worden. Wel kun je hiervoor `\discretionary` gebruiken, bijvoorbeeld

```
oma\discretionary{-}{}{a}tje
menu\discretionary{-}{}{u}tje
caf\discretionary{é-}{}{ee}tje
AOW\discretionary{-}{}{'}er
be\discretionary{-}{e}{ë}indigen
```

maar dat is lastiger dan het woord zelf te kunnen gebruiken.

Het `\discretionary` commando heeft 3 parameters `{pre}{post}{no}`. De eerste twee argumenten `pre` en `post` worden gebruikt in geval van een afbreking: resp. het gedeelte vóór en na de regelovergang; `pre` zal meestal met een afbreekteken eindigen. Het derde argument `no` wordt gebruikt als er geen afbreking op dat punt plaatsvindt.

Overigens, bij gebruik van het `babel` package kun je het laatste voorbeeld oplossen door '`be"eindigen`' te schrijven, maar dat heeft weer andere nadelen (zoals zoeken in je editor en spellingcontrole).

Met de huidige afbreekalgoritme en -patronen wordt `omaatje` helemaal niet afgebroken, `menuutje` als `me-nuutje`, `cafeetje` als `ca-feetje`, en `beëindigen` als `be-ëin-di-gen`, waarbij dus foutief een trema op de '`e`' kan blijven staan na de afbreking. Dit laatste komt door de aanwezigheid van het patroon '`3ë`'. Als we dit handmatig verwijderen wordt de afbreking `beëin-di-gen`, wat in ieder geval beter is.

LuaTEX heeft een aangepast afbreekalgoritme waarbij dit in de uitzonderingslijst aangegeven worden door daarin de parameters van `\discretionary` op te nemen[7, Chapter 5, pp. 73–89], bijvoorbeeld

```
\hyphenation{menu{-}{}{u}tje oma{-}{}{a}tje
             caf{é-}{}{ee}tje}
```

Overigens mag het commando `\discretionary` ook ingevoegd worden maar dit wordt door LuaTEX in deze context genegeerd. Jammer genoeg hebben PDFTEX en XeTEX deze uitbreiding niet.

Ook OpenOffice.org/LibreOffice gebruiken het afbreekalgoritme van TEX via de bibliotheek `libhyphen`[9]. Hierin is ook het afbreekalgoritme aangepast om de eerder genoemde speciale afbrekingen te kunnen doen[19]. Het gebruikt wel een andere syntax voor deze patronen dan LuaTEX, maar het idee is hetzelfde. Bijvoorbeeld

```
a1atje./a=t,1,3
e1etje./é=tje,1,5
```

Als zo'n patroon een winnende match geeft en de afbreking plaatsvindt vanwege dit patroon, dan wordt er een substitutie toegepast. Het eerste getal geeft het begin aan van het stuk wat vervangen wordt, relatief t.o.v. het patroon (in deze voorbeelden dus de eerste letter van het patroon); het tweede getal geeft de lengte van het te vervangen stuk aan. De tekst achter de '`/`' is de vervanging, waarbij '`=`' het afbreekteken is.

In het eerste voorbeeld wordt '`aat`' dus vervangen door '`a-t`'; in het tweede voorbeeld wordt '`eetje`' vervangen door '`é-tje`', indien hier afgebroken wordt.

Dit zijn in feite ook `\discretionary`s maar dat is een TEX-begrip dat `libhyphen` niet kent.

In `libhyphen` wordt geen aparte uitzonderingslijst gebruikt, maar door middel van een voorbewerking worden de uitzonderingen omgezet in patronen. Zo wordt in het Engels bijvoorbeeld `am-phet-a-mine` in het patroon `.am9p8h8e8t9a9m8i8n8e.` omgezet.

In de Nederlandse patronen wordt dit nog niet gebruikt, maar wel in de Hongaarse. Hunspell (waar `libhyphen` ondergebracht is) is van oorsprong Hongaars (vandaar de naam) en László Németh heeft de uitbreiding van het algoritme in eerste instantie voor het Hongaars ontworpen, omdat het gewone algoritme daar niet zulke goede resultaten gaf. Maar het is natuurlijk ook bruikbaar voor andere talen.

Hunspell en `libhyphen` worden in diverse applicaties gebruikt, vooral in opensourceprojecten, zoals Chrome, Opera en Firefox, maar bijvoorbeeld ook in InDesign. In dit kader is het vermeldenswaardig dat in 2021 – op verzoek van Annemarie Apple van Google – de MIT licentie aan de Nederlandse (en Griekse) patronen is toegevoegd zodat ze ook in Android gebruikt zouden kunnen worden[10].

Babel en LuaTEX

Het package `babel`[6] heeft voorzieningen om bijzondere afbrekingen te doen als LuaTEX gebruikt wordt, buiten de gewone patronen om. Dit gebeurt met z.g. *transforms*, transformaties die zowel voor als na het afbreken toegepast kunnen worden. Voor ons doel zijn vooral de transformaties ná het afbreken (*posthyphenation*) interessant, en ik zal me beperken tot degene die relevant zijn voor het Nederlands. In de `babel` documentatie zijn ook voorbeelden te vinden voor andere talen. Hierbij moet wel opgemerkt worden dat de huidige documentatie van deze functies niet heel erg helder zijn. Zie ook [1].

Om te beginnen is er in de Nederlandse `babel` ondersteuning een voorziening om een trema na een afbreekpunt weg te halen. Dit is de `diaeresis.hyphen` *transform*. Dit is onafhankelijk van de meer traditionele aanpak van `babel` waarbij '`"e`' zonder afbreken 'ë' geeft en met afbreken '`-e`'. Met de *transform* kun je gewoon het woord gebruiken zonder dergelijke fratsen.

```
\usepackage[dutch]{babel}
\babelprovide[transforms=diaeresis.hyphen]{dutch}
```

Met deze declaraties wordt `beëindigen` correct afgebroken als `be-ein-di-gen`. En analoog voor de andere klinkers.

Daarna is het mogelijk om eigen *posthyphenation transforms* te definiëren. Bijvoorbeeld voor de correcte afbreking van `cafeetje`:

```
\babelposthyphenation{dutch}{eetje}{
  { no = ee , pre = é-, penalty=50 },
  remove,
  {string = tje}
}
```

Het eerste argument van `\babelposthyphenation` is de taal waarvoor dit van toepassing is. Het tweede is een Lua-patroon (in feite een reguliere expressie). Het derde argument (de vervanging) is een lijst die zegt wat met elk onderdeel dat een match heeft in het patroon (in dit geval elke letter) moet gebeuren. Letters waarvoor niets staat worden weggelaten. Laten we kijken wat er in dit voorbeeld gebeurt.

Het patroon is hier `eetje`. Je zou misschien denken dat dit ook toegepast wordt op woorden als `beetje` en `feetje`. Maar dit wordt toegepast na het afbreken en dan staat er in deze woorden als een afbreekpunt (dit wordt in dit argument met een '`|`' aangegeven). Aangezien deze woorden afgebroken worden als `beet-je` en `fee-tje` zou het patroon `eet|je` of `ee|tje` moeten zijn om hier ook een match te geven, iets wat we natuurlijk niet willen.

De eerste letter wordt vervangen door `{ no = ee , pre = é- }`. Dit is in feite een `\discretionary{é-}{}{ee}`, waarvan de elementen aangegeven worden met `pre`, `post` en `no`, en lege elementen weggelaten mogen worden.

De tweede letter wordt weggelaten, omdat die al in

de eerste verwerkt zit. Tenslotte wordt de rest van het patroon vervangen door de letterlijke tekst `tje`. Het is ook mogelijk om iedere letter individueel over te nemen door drie keer `{}` te specificeren, want `{}` betekent dat de letter ongewijzigd overgenomen moet worden.

```
\babelposthyphenation{dutch}{eetje}{
  { no = ee , pre = é- },
  remove,
  {}, {}, {}
}
```

Met deze declaratie worden de woorden `cafeetje cafeetjes buurtcafeetje` afgebroken als `ca-fé-tje ca-fé-tjes buurt-ca-fé-tje`

Voor zover ik weet zijn er met de huidige afbreekpatronen geen andere woorden die een match hebben met `eetje` (dus zonder afbreking). Maar het is mogelijk om deze transformatie specifiek voor `cafeetje` (en afleidingen ervan) te maken door dit expliciet in het patroon op te nemen. Daarbij moeten we er rekening mee houden dat het wordt afgebroken als `ca-feetje` en het afbreekpunt moeten we aangeven met '`|`'.

```
\babelposthyphenation{dutch}{ca|feetje}{
{}, {}, {}, {},
{ no = ee , pre = é- },
remove,
{}, {}, {}
}
```

De eerste vier elementen (`ca|f`) worden overgenomen; de rest is hetzelfde als het vorige voorbeeld.

Twee andere woorden die in dit artikel genoemd worden, zijn `omaatje` en `menuutje`. Ik zal `omaatje` hier behandelen; `menuutje` is dan analoog. Om te beginnen zouden we het iets makkelijker kunnen doen met het `\hyphenation` commando, omdat dit in LuaTEX ook een notatie voor *discretionaries* heeft:

```
\hyphenation{oma{-}{}{a}tje}
```

Echter, het nadeel t.o.v. een patroon is dat dit alleen voor het woord zelf geldt en niet voor samenstellingen en afleidingen, zoals `omaatjes`. Een patroon kan dit wel. In [1] wordt het voorbeeld van `omaatje` ook gegeven, maar daar is het fout. Het patroon `aatje` is te ruim. Dan vallen ook `maatje` en `tomaatje` onder het patroon en worden dan resp. `ma-tje` en `toma-tje`[11]. En er zijn nog diverse andere woorden met `aatje`.

Daarom nemen we `omaatje` in zijn geheel en combineren dit gelijk met `opaatje`. Hier gebruiken we de kracht van reguliere expressies.

```
\babelposthyphenation{dutch}{o[mp]aatje}{
{}, {}, {},
{ no = a , pre= - },
{}, {}, {}
}
```

De notatie `[mp]` betekent dat daar een '`m`' of een '`p`' mag staan. Hierdoor worden zowel `omaatje` als `opaatje` en hun meervouden goed afgebroken. Woorden als `tomaatje` en `automaatje` vallen niet onder het patroon, omdat die een afbreekpunt na de '`o`' hebben en er geen '`|`' in het patroon staat.

`Patgen`: patroongeneratie

Naast het handmatig vervaardigen van de patronen (wat een monnikenwerk is) kunnen ze ook semiautomatisch gemaakt worden met het programma `patgen`. Dit is geschreven door Frank Liang en wordt beschreven in het tweede deel van Liangs proefschrift[15, hoofdstuk 4]. Je genereert de patronen dan vanuit een lijst met afgebroken woorden (d.w.z., woorden met afbreekstreepje op de juiste plaats). Dit is ook de manier waarop TEX's standaardafbreekpatronen voor het Engels gemaakt zijn.

Het oorspronkelijke programma `patgen` was bedoeld voor TEX82, dat nog met 7-bitkarakters werkte. Het kende alleen de letters '`a`' t/m '`z`' (hoofdletters worden naar kleine letters omgezet).

Vanwege de internationale vraag naar faciliteiten voor andere talen, bracht Knuth in TEX versie 3.0 de mogelijkheid om fonts met internationale alfabetten te ondersteunen, en ook om meerdere talen tegelijk aan boord te hebben en zelfs binnen een document te kunnen wisselen. Daarom werkt deze versie intern met 8-bitkarakters. Deze versie kwam in januari 1990 beschikbaar. Maar `patgen` kon hier nog niet mee werken, In November 1991 heeft Peter Breitenlohner `patgen` aangepast zodat het ook met 8-bitkarakters kan werken, en dus ook andere letters dan '`a`' t/m '`z`' aankan. Deze versie kan met diverse alfabetten werken zolang er niet meer dan 243 karakters in zitten. (256 karakters in 8 bits, minus de cijfers en de karakters '`-`', '`*`' en '`.`' of vervangende karakters voor de laatste drie).

Het algoritme is verder in principe hetzelfde.

De beschrijving van de oorspronkelijk versie staat in Frank Liangs proefschrift[15, hoofdstuk 4], waarvan ik vind dat die best lastig te begrijpen is, en voor de nieuwe versie is er een tutorial van Yannis Haralambous[10]. Dit tutorial vertelt alleen hoe je het programma moet gebruiken, niet hoe het van binnen werkt.

David Antoš schreef een herimplementatie van de `patgen` in C++[3], die ook Unicode aankan en zijn beschrijving over de werking ervan vind ik veel duidelijker. Zijn afstudeerverslag[4] legt de werking nog gedetailleerder en duidelijker uit. Zijn versie heeft wel wat extra uitbreidingen vergeleken met `patgen`, maar voor het begrijpen is dat geen probleem. Er is ook nog een conferentie-artikel over dit programma[5].

Ik zal hier in het kort de werking schetsen. Het gaat te ver om in dit artikel dit gedetailleerd te doen. Daarvoor kun je de hiervoor genoemde literatuur gebruiken.

Om de patronen te genereren met `patgen` heb je een woordenlijst nodig met afbreekpunten, op dezelfde manier als de uitzonderingslijst van het commando `\hyphenation`. Voor `patgen` is wel vereist dat er één woord per regel staat.

Je haalt dit bestand dan door `patgen` heen, en dit gaat hier patronen uithalen. Een patroon is in principe een substring van zo'n woord. Elk patroon wordt dan op de hele lijst losgelaten en de afbreekpunten die hieruit komen worden vergeleken met de ingevoerde afbreekpunten. Er zijn drie mogelijkheden:

- Een bestaand afbreekpunt wordt gevonden. Dit is de ideale situatie.
- Een bestaand afbreekpunt wordt gemist. Jammer.
- Het patroon vind een onjuist afbreekpunt. Dit wil je voorkomen.

Het programma telt nu hoeveel goede en slechte afbreekpunten het patroon genereert en met een formule waarvan de parameters door de gebruiker ingevoerd zijn wordt dan gekeken of het patroon geaccepteerd wordt of verworpen. Het criterium is:

$$w_g * g - w_b * b \geq D$$

Hierbij zijn g en b het aantal goede, resp. slechte afbreekpunten die het patroon oplevert; w_g en w_b zijn bijbehorende gewichtsfactoren en D is de drempelwaarde. Als aan de vergelijking voldaan is, wordt het patroon geaccepteerd.

Het kiezen van deze getallen is natte-vingerwerk. Meestal moet je maar wat proberen en dan proberen zo aan te passen dat de uitkomst beter wordt. Je kunt ook kijken wat anderen gedaan hebben en proberen of die getallen goed werken. Liang geeft de getallen die hij gebruikt heeft voor de standaard patronen voor TEX.

Sander van Geloven meldt dat hij een script heeft dat heel uitgebreid waardes probeert en rapporteert hoe goed het resultaat is.[12]

Ook het tutorial van Haralambous[10] geeft de parameters die hij gebruikt heeft.

Een aantal artikelen van Petr Sojka et al.[22, 25, 23, 24] geven o.a. de effecten van verschillende keuzes van de parameters op de gegenereerde patronen weer bij het genereren van Tsjecho-Slowaakse patronen. Dit kan ook nuttig zijn bij het kiezen van de parameters.

De parameters die gebruikt zijn voor het genereren van de Tutelaers patronen zijn te vinden in [26, Tabel 1].

Behalve de bovengenoemde parameters moeten ook nog de minimale en maximale patroonlengte opgegeven worden. En verder welke niveaus behandeld moeten worden (dit zijn de cijfers 1 t/m 9 in de patronen). Je geeft een minimum en een maximum niveau op.

De aangegeven niveaus worden dan één voor één behandeld, van laag naar hoog. Binnen een niveau worden dan een aantal passes over de woordenlijst gemaakt, voor de verschillende patroonlengtes en posities van het afbreekpunt in een patroon, waarbij de patronen verfijnd worden. Tussen de niveaus door worden opnieuw de waarden van de gewichtsfactoren, de drempelwaarde en de minimale en maximale patroonlengte gevraagd.

De even niveaus zijn bedoeld om verkeerde afbreekpunten te onderdrukken, zoals in het begin van dit artikel (sectie TEX's afbreekalgoritme) beschreven is. De betekenis van 'goed' en 'slecht' wordt in deze niveaus omgedraaid: een goed patroon is een patroon dat verkeerde afbreekpunten onderdrukt, en een patroon is slecht als het een geldig afbreekpunt onderdrukt.

Aan het eind worden de goede patronen naar een bestand geschreven. Het is ook mogelijk om de woordenlijst zoals die uit het algoritme komt ook naar een bestand te laten schrijven. Hierin worden de terecht gevonden afbreekpunten met een '*' aangegeven, de foutieve met een '.' en de gemiste met een '-'.

Het is gebruikelijk dat de patronen in deze eerste run van `patgen` nog niet goed zijn. Daarom kan een volgende run gemaakt worden waarbij de patronen uit de vorige run als invoer gebruikt worden. Dit levert dan (hopelijk) verbeterde patronen op. Dit proces kan een aantal keren herhaald worden. Bij opeenvolgende runs moeten de niveaus oplopend zijn.

Het voordeel van het maken van verschillende runs in plaats van `patgen` alles in één run (met verschillende niveaus) te laten doen, is dat je dan tussendoor de gegenereerde woordenlijst met de indicatie van foute en gemiste afbreekpunten kunt inspecteren. Als je maximale controle wilt, kun je het beste in elke run één niveau doen.

Een testrun van `patgen`

In deze sectie geef ik de resultaten van een testrun, met een speelgoedvoorbeeld. Omdat ik nog geen Nederlandse woordenlijst met afbreekpunten heb, heb ik hiervoor het kleine woordenlijstje van de uitzonderingen bij de Nederlandse patronen genomen. Je vindt deze achter het commando `\hyphenation` in het Nederlandse afbreekbestand[13]. Dit bevat 40 afgebroken woorden, o.a. het hier al eerder besproken `wa-ter-staats-in-ge-ni-eur`.

Ik gebruik als voorbeeld de parameters zoals Petr Sojka die gebruikt heeft en die de meest correcte patronen voor het Tsjechisch opleverde[25, Table 6][24, Table 2]. Alleen geef ik niveaus 1 en 2 op, terwijl Sojka, één niveau per run doet.

De aanroep van `patgen` krijgt vier bestandsnamen mee:

1. de woordenlijst met afbreektekens
2. een patronenbestand om mee te beginnen (mag leeg zijn)
3. het bestand waar de gegenereerde patronen naar geschreven worden
4. een '*translate*' bestand dat het alfabet aangeeft

Het laatste is nodig als je andere letters dan 'a' t/m 'z' gebruikt, bijvoorbeeld letters met diakritische tekens of Griekse letters. Ons bestand bevat alleen 'a' t/m 'z' dus laten we dit leeg. Omdat we in de run met niveau 1 beginnen, starten we ook met een leeg patronenbestand.

```
PATGEN-RUN $ patgen dutch.hyp /dev/null pat.out /dev/null
This is PATGEN, Version 2.4 (TeX Live 2025)
left_hyphen_min = 2, right_hyphen_min = 3, 26 letters
0 patterns read in
pattern trie has 256 nodes, trie_max = 256, 0 outputs
hyph_start, hyph_finish: 1 2
pat_start, pat_finish: 1 3
good weight, bad weight, threshold: 1 5 1
processing dictionary with pat_len = 1, pat_dot = 0

0 good, 0 bad, 106 missed
0.00 %, 0.00 %, 100.00 %
0 patterns, 256 nodes in count trie, triec_max = 256
2 good and 12 bad patterns added (more to come)
finding 4 good and 0 bad hyphens, efficiency = 2.00
pattern trie has 256 nodes, trie_max = 256, 2 outputs
processing dictionary with pat_len = 1, pat_dot = 1

4 good, 0 bad, 102 missed
3.77 %, 0.00 %, 96.23 %
0 patterns, 256 nodes in count trie, triec_max = 256
2 good and 14 bad patterns added (more to come)
finding 4 good and 0 bad hyphens, efficiency = 2.00
pattern trie has 256 nodes, trie_max = 256, 6 outputs
processing dictionary with pat_len = 2, pat_dot = 1

8 good, 0 bad, 98 missed
7.55 %, 0.00 %, 92.45 %
64 patterns, 320 nodes in count trie, triec_max = 431
25 good and 22 bad patterns added (more to come)
finding 36 good and 0 bad hyphens, efficiency = 1.44
pattern trie has 303 nodes, trie_max = 324, 7 outputs
processing dictionary with pat_len = 2, pat_dot = 0

44 good, 0 bad, 62 missed
41.51 %, 0.00 %, 58.49 %
73 patterns, 329 nodes in count trie, triec_max = 453
10 good and 57 bad patterns added (more to come)
finding 33 good and 0 bad hyphens, efficiency = 3.30
pattern trie has 330 nodes, trie_max = 445, 10 outputs
processing dictionary with pat_len = 2, pat_dot = 2

77 good, 0 bad, 29 missed
72.64 %, 0.00 %, 27.36 %
66 patterns, 322 nodes in count trie, triec_max = 410
6 good and 52 bad patterns added (more to come)
finding 9 good and 0 bad hyphens, efficiency = 1.50
pattern trie has 345 nodes, trie_max = 445, 20 outputs
processing dictionary with pat_len = 3, pat_dot = 1

86 good, 0 bad, 20 missed
81.13 %, 0.00 %, 18.87 %
11 patterns, 277 nodes in count trie, triec_max = 278
5 good and 2 bad patterns added (more to come)
finding 9 good and 0 bad hyphens, efficiency = 1.80
pattern trie has 352 nodes, trie_max = 445, 20 outputs
processing dictionary with pat_len = 3, pat_dot = 2

95 good, 0 bad, 11 missed
89.62 %, 0.00 %, 10.38 %
8 patterns, 271 nodes in count trie, triec_max = 271
1 good and 4 bad patterns added (more to come)
finding 4 good and 0 bad hyphens, efficiency = 4.00
pattern trie has 358 nodes, trie_max = 445, 20 outputs
processing dictionary with pat_len = 3, pat_dot = 0

99 good, 0 bad, 7 missed
93.40 %, 0.00 %, 6.60 %
17 patterns, 279 nodes in count trie, triec_max = 309
2 good and 12 bad patterns added (more to come)
finding 3 good and 0 bad hyphens, efficiency = 1.50
pattern trie has 373 nodes, trie_max = 464, 20 outputs
processing dictionary with pat_len = 3, pat_dot = 3

102 good, 0 bad, 4 missed
96.23 %, 0.00 %, 3.77 %
16 patterns, 288 nodes in count trie, triec_max = 289
1 good and 13 bad patterns added (more to come)
finding 1 good and 0 bad hyphens, efficiency = 1.00
pattern trie has 387 nodes, trie_max = 464, 23 outputs
72 nodes and 15 outputs deleted
total of 54 patterns at hyph_level 1

pat_start, pat_finish: 1 3
good weight, bad weight, threshold: 1 5 1
processing dictionary with pat_len = 1, pat_dot = 0

103 good, 0 bad, 3 missed
97.17 %, 0.00 %, 2.83 %
0 patterns, 256 nodes in count trie, triec_max = 256
0 good and 27 bad patterns added
finding 0 good and 0 bad hyphens
pattern trie has 315 nodes, trie_max = 464, 11 outputs
processing dictionary with pat_len = 1, pat_dot = 1

103 good, 0 bad, 3 missed
97.17 %, 0.00 %, 2.83 %
0 patterns, 256 nodes in count trie, triec_max = 256
0 good and 27 bad patterns added
finding 0 good and 0 bad hyphens
pattern trie has 315 nodes, trie_max = 464, 14 outputs
0 nodes and 6 outputs deleted
total of 0 patterns at hyph_level 2
hyphenate word list? y
writing pattmp.2
```

```
103 good, 0 bad, 3 missed
97.17 %, 0.00 %, 2.83 %
```

We zien hier dat in deze run 11 passes worden gedaan (aangegeven door '`processing dictionary`'), waarvan 9 op niveau 1 en 2 op niveau 2. Hierbij is `pat_len` de lengte van de patronen in deze pass, `pat_dot` geeft de positie van het afbreekpunt in het patroon aan. Aan het eind van niveau 1 zijn er 54 patronen gevonden, met 103 goede (97%), 0 slechte en 3 gemiste afbreekpunten. Behoorlijk goed! De behandeling van niveau 2 (2 passes) levert echter niets nieuws op, dus die hadden we net zo goed weg kunnen laten. Dit komt in dit simpele voorbeeld natuurlijk omdat niveau 2 (zoals alle even niveaus) bedoeld is om slechte afbreekpunten te verwijderen, en niveau 1 had geen slechte afbreekpunten. Het ergste wat er had kunnen gebeuren was dat dit niveau ten onrechte een goed afbreekpunt verwijderd zou hebben.

In de gegenereerde woordenlijst kunnen we kijken wat de gemiste afbreekpunten zijn. En we zien dat bijvoorbeeld `wa*ter-staats*in*ge*ni*eur`, dus het afbreekpunt tussen `water` en `staats` is gemist. We zouden dan kunnen proberen met een run op niveau 3 de gemiste afbreekpunten er nog bij te krijgen.

De CELEX-patronen

Bij het Departement Informatica op de Universiteit Utrecht, waar ik een flink deel van mijn leven heb doorgemaakt, werd TEX snel populair toen wij daar onze eigen computers kregen. In eerste instantie vooral voor de wetenschappelijke rapporten, maar ook voor collegedictaten, studiegidsen e.d. die vaak in het Nederlands waren. Dit gaf problemen omdat er geen Nederlandse afbreekpatronen waren. Henk Penning (systeembeheerder) en ik (verantwoordelijk voor de TEX-installatie) zijn toen gaan kijken of we Nederlandse afbreekpatronen konden genereren.

In het TEXbook wordt vermeld dat in het proefschrift van Frank Liang beschreven wordt hoe je de patronen maakt ("how to compute tables by which TEX will be able to reconstruct most of the hyphens …")[13, p. 449]. Ook zullen we waarschijnlijk ergens op een TEX mailing list gelezen hebben dat je hiervoor het programma `patgen` moesten gebruiken. Echter, de enige handleiding van `patgen` was op dat moment te vinden in Liangs proefschrift.

Bedenk wel, dit was in de tijd dat wetenschappelijke artikelen en tijdschriften nog per post bezorgd werden, in echte papieren enveloppen. Amerikaanse proefschriften werden standaard op microfilm of microfiche gezet door University Microfilms (tegenwoordig ProQuest LLC). De Amerikaanse Library of Congress zet nog steeds proefschriften op microfilm/microfiche en heeft er meer dan een miljoen in huis.

Om een proefschrift te krijgen kon je dit bij University Microfilms bestellen en die maakten er dan een afdruk van. Een soort *printing on demand avant la lettre*. Wij hebben het proefschrift van Liang toen besteld en het bestudeerd.

Voor de input van `patgen` heb je een woordenlijst met afbreekpunten nodig zoals hiervoor aangegeven. Ik denk dat Henk Penning erachter gekomen was dat CELEX in Nijmegen zo'n woordenlijst had[12], maar deze was niet gratis. Wij hebben toen toestemming gekregen om deze lijst te gebruiken voor het genereren van de patronen, maar nergens anders voor. De gegenereerde patronen hebben ook het auteursrecht op CELEX staan. Daarom heb ik deze de CELEX-patronen genoemd. Ik weet niet meer of de auteursrechtvermelding een voorwaarde van CELEX was voor het gebruik, of dat de patronen als een bewerking van de woordenlijst beschouwd worden en daarom onder het oorspronkelijke auteursrecht vallen. Tegenwoordig zou deze kwestie waarschijnlijk onder het databankenrecht vallen, maar dat bestond indertijd nog niet als apart recht. Maar ik ben geen jurist, dus neem mijn woorden met een korreltje zout.

CELEX is een uitgebreide database over Nederlandse (en later ook Duitse en Engelse) woorden, ontwikkeld door het *Centre for Lexical Information*, een samenwerkingsverband tussen de Universiteit van Nijmegen, het Instituut voor de Nederlandse Lexicologie (tegenwoordig het Instituut voor de Nederlandse Taal - INT), het Max Planck Instituut voor Psycholinguïstiek in Nijmegen en het Instituut voor Perceptie Onderzoek in Eindhoven[14]. De lijst met afbrekingen is een onderdeel daarvan.

Henk en ik hebben een aantal dagen samen achter een terminal gezeten om met de parameters voor `patgen` te experimenteren, en hieruit zijn uiteindelijk de CELEX-patronen ontstaan. In die tijd hadden we nog TEX82, dus met 7-bitkarakters, en bovendien kon `patgen` toen alleen met de letters '`a`' t/m '`z`' werken, dus we konden alleen patronen zonder diakritische tekens genereren. Deze patronen hebben een aantal jaren dienst gedaan in de Nederlandse TEX-installaties.

De parameters voor `patgen` die we gebruikt hebben zijn helaas niet bewaard gebleven.

Een interessant bijverschijnsel bij het genereren van de patronen was, dat we bij het inspecteren van foute en gemiste afbreekpunten ontdekten dat sommige veroorzaakt werden door fouten in de woordenlijst in plaats van fouten in de gegenereerde patronen. De patronen leggen in feite de regelmatigheden in de afbrekingen weer, en fouten verstoren die regelmaat. We konden deze fouten corrigeren en hebben de correcties weer aan CELEX teruggegeven.

Bij het samenstellen van een TEX-formaat voor de PC (indertijd nog met MS-DOS) bleek dat ze te groot waren voor het geheugen dat een PC had.

Uit het verslag van de NTG-bijeenkomst 11 mei 1989[28, p. 6]:

“

Werkgroep 13 NL Nederlandstalige TEX
1. Afbreekpatronen

De CELEX afbreekpatronen zijn op dit moment beschikbaar. De patronenfile is zo danig groot dan ze niet voor de PC geschikt is en zelfs bij andere systemen een aanpassing van de `tree_size` (naar 11.000) noodzakelijk maken. TEX dient i.h.a. opnieuw aangemaakt te worden. De file is Public Domain (de kop van de file vermeldt wel enige restricties).

”

Daarom is er een beperkte versie gemaakt, waarbij alle patronen die een ‘`5`’ bevatten verwijderd zijn. Dit heeft tot gevolg dat er afbreekpunten gemist worden, maar er ontstaan hierdoor geen extra ongewenste afbreekpunten. Dit is het hierboven genoemde bestand `nehyph1.tex`. Tegenwoordig zijn dergelijke geheugenbeperkingen niet meer relevant.

De patronen van Piet Tutelaers

Rond 1993 heeft Piet Tutelaers nieuwe patronen voor het Nederlands gemaakt[26]. Hij geeft drie redenen waarom de CELEX-patronen niet voldeden:

1. Ze werken niet goed voor woorden met diakritische tekens (8-bitkarakters).
2. Ze breken niet af in de eerste en laatste twee letters van een woord.
3. Ze voldoen niet aan de nieuwere afbreekregels uit de ‘Herziene Woordenlijst Nederlandse Taal’ (Groene Boekje[15] 1990).

De situatie was ook anders vergeleken met de tijd dat de CELEX-patronen gemaakt waren:

- Donald Knuth had intussen TEX 3.0 gelanceerd, waar het mogelijk was om 8-bitkarakters te gebruiken.
- Peter Breitenlohner had een nieuwe versie van `patgen` gemaakt die met 8-bitkarakters kon werken.
- Er waren 8-bitversies van de TEX fonts beschikbaar, toendertijd de DC fonts, de voorlopers van de EC fonts. Voor het gebruiken van 8-bitpatronen moet er ook een 8-bitfont beschikbaar zijn. Het afbreekalgoritme werkt met de codering van de letters in het geselecteerde font. Bij het gebruik van een 7-bitfonts (bijvoorbeeld de originele Computer Modern fonts) worden letters met diakritische tekens omgezet in de gewone letter met een accentcommando (bijvoorbeeld á wordt omgezet in `\'{a}`), en dit past niet in het afbreekalgoritme. Dus om woorden met diakritische tekens correct af te breken moet het juiste 8-bitfont actief zijn (d.w.z. een font met dezelfde codering voor deze letters als de patronen). Bij het maken van de CELEX-patronen waren dan ook alle woorden met diakritische tekens weggelaten.

Piet Tutelaers is uitgegaan van een elektronische versie van het Groene Boekje 1954 (GB54), en heeft deze met behulp van de CELEX-patronen afgebroken. Het resultaat heeft hij handmatig vergeleken, gecorrigeerd en aangevuld door middel van het Groene Boekje 1990 (GB90). De resulterende woordenlijst met afbrekingen is gebruikt als invoer van `patgen`. Ik heb geen discussie kunnen vinden of deze procedure auteursrechtelijk (i.v.m. het gebruik van een elektronische kopie van GB54) acceptabel is. Piet Tutelaers denkt van wel.[16]

level	lengte	parameters		
1	1–4	1	2	20
2	1–4	2	1	4
3	1–5	1	1	1
4	1–6	3	2	1
5	1–8	1	1000	1

Tabel 2: Parameters voor `patgen` voor de Tutelaers patronen.

level	patronen	goed(%)	fout(%)	miss(%)
1	832	97.48	11.13	2.52
2	1196	92.29	0.55	7.71
3	3672	99.86	3.87	0.14
4	2396	87.87	0.01	2.13
5	2167	100.00	0.01	0.00

Tabel 3: Resultaten van `patgen` voor de Tutelaers patronen.

In tabel 2 vinden we de parameters voor `patgen` die Tutelaers gebruikt heeft en in tabel 3 de resultaten in procenten (bron: [26]). Hieruit blijkt dat deze patronen heel goed werken.

In 1995/1996 is een nieuwe spelling van kracht geworden. De patronen zijn toen opnieuw gegenereerd, maar de details hiervan zijn niet gepubliceerd. Piet Tutelaers wist ook niet meer hoe het precies gegaan is.

Wel vond ik enige discussie over de legaliteit op de TEX-NL mailing list, hieronder gekopieerd[17]. Hieruit

heb ik geconcludeerd dat de huidige patronen op het Elektronische Groene Boekje EGB96[18] gebaseerd zijn. Die woordenlijst kan niet verspreid worden, zoals in het citaat van Piet Tutelaers hieronder staat. Dit lijkt mij juist. Maar ook hier denkt Piet dat het toegestaan is om deze legaal aangeschafte woordenlijst te gebruiken om patronen te genereren, en deze te verspreiden. In de afgelopen 29 jaar heeft nog niemand daartegen geprotesteerd of geprocedeerd.

“

```
From: Frans Goddijn <***>
Sender: TEX-NL <TEX-NL@nic.SURFnet.nl>
To: Multiple recipients of list TEX-NL
Subject: Groeneboekje ’54
Date: Fri, 5 Jul 1996 09:44:00 +0100
```

said Piet Tutelaers to All:

> Via de DOS versie van het elektronische Groene boekje is een collega erin geslaagd de woorden uit het EGB96 (via WP macros) met hun afbreekplaatsen te ontfutselen.

Zoals je dit schrijft (“ontfutselen”), wekt het de suggestie dat het tegen de regels in is gedaan. Ik bedoel: wie iets slims doet, hoeft niet per se iets ondeugends te hebben gedaan.
Je collega heeft de afbreekpatronen die VOOR IEDEREEN die WP gebruikt en die die floppen heeft gekocht zijn te gebruiken op legale manier verkregen. Hij heeft immers, mag ik aannemen, WP gekocht en de SDU floppen gekocht. Daarmee heeft hij de gereedschappen en de grondstoffen ingekocht. De patronen die hij eruit heeft gehaald zijn dan weer zijn eigen werk. Hij heeft die patronen weliswaar niet zelf verzonnen, maar dat heeft SDU ook niet: dat werk is gedaan door de instantie die de nieuwe regels heeft bedacht, de overheid dus. Die patronen zijn op zichzelf niemands eigendom.
Net zoals de wetgeving volgens welke je als fietser voor een rood stoplicht, door mag rijden als je rechtsaf slaat. Ik kan dat opschrijven en verspreiden, ik kan me eraan houden zonder dat ik het auteursrecht van de uitgever van het wetboek of van een slimme wetboek-CDrom schaad. Ook al heb ik de regel overgenomen van een slim en interactief CD-wetboekje.

> (Tutelaers)
> Ik ben bezig om hieruit nieuwe afbreekpatronen te destilleren. Omdat het EGB96 auteursrechtelijk beschermd is, kan ik het ‘gekraakte’ woor-den-bestand helaas niet verspreiden.

Er is volgens mij NIETS “gekraakt”. Waarom wordt intelligentie gecriminaliseerd? (lees een smiley hier, het is niet de bedoeling iemand boos te maken). Hier geldt hetzelfde als hierboven. De afbreekpatronen zijn er. De SDU heeft een utility in de verkoop voor WP users die zelf niet in staat zijn die patronen in te bouwen of die van gemak houden. De TeX wereld heeft mensen die slim zijn en die niet uitsluitend op gemak zijn gesteld. Of nu iemand die afbreekpatronen van scratch gaat uittikken en inwerken vanuit een gedrukte tekst met regels, of ze aan de hand van het papieren Groene Boekje gaat afleiden, of ze aan de hand van een eerlijk gekocht WP bestand gaat afleiden, dat maakt niet uit.

Lees dit dus als een hartstochtelijke oproep om WEL en volkomen zonder schuldgevoel die patronen beschikbaar te stellen...

Met hartelijke groet!

Frans Goddijn

”

Frans heeft ook nog een artikel geschreven over de WORDS-L-groep (niet van de NTG) waar dit ook zijdelings ter sprake komt[9].

In 2005 is er nog een spellingwijziging geweest. Alhoewel dit slechts kleine veranderingen waren, heeft dit veel stof doen opwaaien, zie bijvoorbeeld [21]. Het daarin genoemde woord `ca-tas-tro-fe` is vreemd afgebroken. Als je de afbreekregels van 2005 toepast, dan kom je op `ca-ta-stro-fe`. De regel is

> Op de grens van lettergrepen wordt zo afgebroken dat zoveel mogelijk medeklinkerletters naar de volgende regel gaan, maar het rechterelement moet beginnen met een lettercombinatie die aan het begin van een woord kan voorkomen.

Aangezien ‘`str`’ aan het begin van een woord kan voorkomen zou de afbreking ‘`-stro`’ correct moeten zijn. Dat is ook wat onze afbreekpatronen geven (behalve Vanroose, die het middelste afbreekpunt mist). Jager and Neyndorff beargumenteren dat `ca-tas-tro-fe` de afbreking volgens de oude spellingsregeling is[11], en dat het misschien wel een vergissing is. Volgens die spelling moest er bij drie medeklinkers tussen de eerste en de tweede afgebroken worden, behalve als er sprake is van een samenstelling of afleiding. En er was ook nog een regel dat er tussen ‘`s`’ en ‘`t`’ afgebroken wordt.

Van Dale Online[19] heeft ook de afbreking `ca-tas-tro-fe`, maar het Van Dale Handwoordenboek[31] van 1996 en de Prisma Spellinggids[20], die op de spelling van 2005 gebaseerd is en door de Taalunie gecertificeerd, hebben beide `ca-ta-stro-fe`. Indien gewenst kan natuurlijk `ca-tas-tro-fe` aan de uitzonderingslijst worden toegevoegd.

En bekend verschil van de spelling van 2005 met de vorige is het woord `paardenbloem`, vroeger `paardebloem`. Beide worden door de diverse patronen correct afgebroken.

Het lijkt er dus op dat deze patronen nog goed dienst kunnen doen. Maar het zou ook wel goed zijn om te controleren of ze op een geactualiseerde woordenlijst het nog steeds zo goed doen.

Codering

De Tutelaers patronen bevatten letters met diakritische tekens. We hebben hiervoor al gemeld dat het nodig is om een font te gebruiken dat deze letters ondersteunt als je woorden met diakritische tekens wilt kunnen afbreken. Bijvoorbeeld de EC fonts, die de T1-fontcodering (ook wel EC genoemd) hebben. Dit betekent echter dat de afbreekpatronen ook deze codering moeten gebruiken, anders werkt het niet. Bij het maken van de patronen met behulp van `patgen` zou dan de gebruikte woordenlijst ook in deze codering moeten zijn. Waarschijnlijk bestaan er geen teksteditors die de T1-codering ondersteunen, maar wat betreft de Nederlandse letters met diakritische tekens (of algemener letters die niet ASCII zijn, namelijk '`Àáâäåçèéêëíîïñóôöúûü`') valt de T1-codering samen met de meest gebruikte coderingen voor het Nederlands: ISO-8859-1 (ook Latin-1 genoemd) en ISO-8859-15 (Latin-9). De meeste teksteditors ondersteunen dit wel. Ook Unicode gebruikt voor deze letters dezelfde nummering als T1, maar als je de Unicode zoals meestal gebruikelijk is, als UTF-8-codering in een tekstbestand zet, dan zijn de bytewaarden anders (2 bytes per karakter).

Dit is niet voor alle talen zo. Bijvoorbeeld de Duitse ß heeft in ISO-8859-1, ISO-8859-15 en Unicode een andere codering dan in T1.

Wanneer we de patronen in een TEX-formaat opnemen, dan moet hiervoor `initex` gebruikt worden. Dit is een versie van plain TEX, dat in tegenstelling tot LaTEX geen `inputenc` package heeft. Daarom moeten de patronen met de juiste codering aangeboden worden. Voor sommige andere talen zoals Grieks, of de talen met een Cyrillisch schrift is dit weer een andere codering. Het betekent ook dat de afbreking van Nederlandse woorden met niet-ASCII-letters niet goed werkt als een font met een andere codering gebruikt wordt, zoals T4 (diverse Afrikaanse talen; sommige letters ontbreken) of T5 (Vietnamees; sommige letters staan op een andere plaats)[20].

In de tijd dat de Nederlandse patronen gemaakt werden, waren er alleen TEX-implementaties met 8-bits interne codering en fonts. Later kwamen LuaTEX en XeTEX die intern Unicode, en als invoer standaard UTF-8 gebruiken. Hiervoor moesten de patronen naar UTF-8 omgezet worden. Voor het Nederlands is dit een triviale operatie, zoals we gezien hebben. Voor andere talen is het iets meer werk, maar toch altijd een simpele operatie.

Tegenwoordig zijn de meeste afbreekpatronen in UTF-8 opgeslagen en wordt er bij het verwerken ervan door `initex` een conversieslag gemaakt naar de gewenste codering als het op een 8-bits TEX plaatsvindt. Deze conversie wordt tijdens het inlezen gedaan door een TEX-script, zodat de patronen in de juiste codering staan op het moment dat TEX ze verwerkt. Voor elke fontcodering is er zo'n script. Deze TEX-scripts zijn door een Rubyscript gegenereerd is. Dit proces wordt beschreven in [16].

In het bestand `language.dat` in je TEX-installatie staan alle talen vermeld die in het TEX-format ingebed zijn. Bij elke taal staat het TEX-bestand dat hiervoor in `initex` ingelezen moet worden. Je kunt dit bestand zelf aanpassen om talen weg te laten of toe te voegen. Voor het Nederlands staat er bijvoorbeeld:

```
dutch loadhyph-nl.tex
```

En `loadhyph-nl.tex` test eerst wat voor soort TEX-engine actief is, en leest dan het juiste bestand in.

Als het TEX-engine intern Unicode gebruikt (dus LuaTEX en XeTEX) dan wordt direct `hyph-nl.tex` ingelezen; dit bestand is in Unicode zoals hierboven aangegeven.

Als het een Japanse TEX-engine (pTEX) is, die niet met Unicode overweg kan, dan moet die het afbreekbestand direct in de T1/EC-codering inlezen (`hyph-nl.ec.tex`). Deze bestanden worden met een Rubyscript vanuit de Unicode bestanden (in ons geval dus `hyph-nl.tex`) gegenereerd. Op deze manier is er slechts één bronbestand per taal nodig, wat vanuit een software-engineering standpunt het beste is.

En dan de 8-bits TEX die wel met Unicode om kan gaan, maar niet als ingebouwde codering, zoals PDFTEX. Dan wordt eerst `conv-utf8-ec.tex` ingelezen, gevolgd door `hyph-nl.tex`. Het bestand `conv-utf8-ec.tex` definieert voor de niet-ASCII letters macro's die de juiste byte in de gewenste codering opleveren zodat TEX op het moment van inlezen van `hyph-nl.tex` geen UTF-8, maar T1-codering ziet. Voor andere fontcoderingen zijn er soortgelijke conversiebestanden `conv-utf8-*.tex`. Zoals al eerder vermeld zijn deze bestanden door een (ander) Rubyscript gegenereerd.

ConTEXt gebruikt een afbreekroutine die in Lua geschreven is, maar wel met dezelfde afbreekpatronen.

Alle bekende patronen (zowel traditionele als in UTF-8) zijn verzameld in een Githubrepository[21]. Daarin bevinden zich ook bijbehorende tools en de documentatie.

De relevante onderdelen worden gekopieerd naar CTAN (package `hyph-utf8`) wanneer ze stabiel zijn, en van daaruit worden ze in TEX-formats verwerkt. Het Githubrepository is in feite het repository van het `hyph-utf8` package. Dit package wordt ook gebruikt in

diverse niet-TEX software projecten, en dit leidt soms tot problemen met de licenties waaronder de patronen gepubliceerd worden. Dit wordt beschreven in [17].

Er is ook nog een aparte website die bij dit package hoort: https://www.hyphenation.org. Er zijn wel diverse links op deze site die niet meer up-to-date zijn.

De Nederlandse (Tutelaers) patronen in UTF-8 zijn binnen dit CTAN package te vinden in https://mirrors.ctan.org/language/hyph-utf8/tex/patterns/tex/hyph-nl.tex; dit is een TEX-bestand, inclusief de commando's \patterns en \hyphenation. Er zijn ook pure tekstbestanden die alleen de patronen, resp. de uitzonderingslijst bevatten: https://mirrors.ctan.org/language/hyph-utf8/tex/patterns/txt/hyph-nl.pat.txt en https://mirrors.ctan.org/language/hyph-utf8/tex/patterns/txt/hyph-nl.hyp.txt. En in een texlive installatie staan ze op tex/generic/hyph-utf8/patterns/tex/hyph-nl.tex en de tekstuele bestanden in tex/generic/hyph-utf8/patterns/txt/.

Tenslotte zijn de Tutelaers patronen in Latin-1-codering ook nog te vinden op https://github.com/hyphenation/tex-hyphen/tree/master/old/hyphen/nehyph96.tex.

Friese afbreekpatronen

Hoewel dit artikel alleen over Nederlandse afbreekpatronen gaat, wil ik toch nog het Fries noemen als verwante taal. Friese afbreekpatronen (Fryske wurdôfbrekkingspatroanen) zijn door Esger Renkema gemaakt en zijn te vinden in zijn repository https://gitlab.com/renkema/frysk. Er zit ook een woordenlijst met afbreekpunten bij. Voor zover ik kon vinden zijn deze niet in een officiële TEX-distributie terecht gekomen.

OpenTaal

Het OpenTaal project[22] heeft als doel om vrij beschikbare Nederlandse woordenlijsten en verwante software (bijvoorbeeld spellingsinformatie voor Hunspell) te maken. Het is in 2005 begonnen en sinds 2009 is OpenTaal een stichting. De woordenlijsten e.d. zijn beschikbaar onder een opensourcelicentie, die erop neer komt dat ze vrij te gebruiken zijn op voorwaarde van bronvermelding. Voor de precieze voorwaarden zie de licentie[23].

Er zijn ruim 400,000 woorden verzameld uit openbare bronnen. Er zijn ook sublijsten met bijvoorbeeld basiswoorden (worden zonder verbuigingen, vervoegingen of afleidingen). Deze lijsten zijn te downloaden van het Githubrepository van OpenTaal[24].

Deze woordenlijsten worden gegenereerd uit een database met woordinformatie, waaronder ook afbreekpunten, maar deze is niet publiek beschikbaar. De informatie is wel op individuele woordbasis beschikbaar via https://woordinfo.org. Er wordt momenteel onderzocht of deze database omgezet kan worden naar een uitgebreider formaat (dus met meer informatie), gebaseerd op de Duitse woordendatabase van DANTE[25]. Sander van Geloven heeft een voorstel (RFC) voor een dergelijk formaat gemaakt[?]. De tekst hiervan is op het Githubrepository van OpenTaal beschikbaar[26].

Op de 39ste NTG-bijeenkomst (8 juni 2007)[8]: "Sebastian Fuchs meldt de vergadering dat de nieuwe woordenlijst door de Taalunie is gecertificeerd. Dit betekent dat wij nu over een woordenlijst volgens de laatste spelling beschikken. Dit is dus een „groene" lijst. – Met deze lijst kunnen nu de afbreekpatronen aangemaakt worden."

Vele jaren later (in 2021) schreef Sander van Geloven – onder het pseudoniem Pander – op TEX-NL:

"

```
From: Pander <***>
To: "TEX-NL : ..."<tex-nl@ntg.nl>
Subject: Afbreekpatronen voor het Nederlands
Date: Wed, 4 Aug 2021 15:53:52 +0200
```

Hoi allemaal,

Voor OpenTaal heb ik samen met Simon afgelopen jaar een nieuwe woordenlijst en spellingcontrole gemaakt. Deze zijn te vinden in:

- woordenlijst:
https://github.com/OpenTaal/opentaal-wordlist
- spellingcontrole:
https://github.com/OpenTaal/opentaal-hunspell
- laatste installeerbare pakketten:
https://github.com/OpenTaal/opentaal-beta

Nu is het tijd om ook de afbreekpatronen een update te gaan geven.

[*Enkele niet-relevante delen weggelaten*]

Alvast bedankt,

Pander

"

Deze afbreekpatronen zijn nog niet gemaakt, maar dit zal in de nabije toekomst gebeuren[27].

Hoe nu verder?

Ten eerste de vraag of de afbreekpatronen aangepast moeten worden. Ik heb al eerder opgemerkt dat de huidige patronen ook na de spellingwijziging van 2005 waarschijnlijk best goed werken. Maar er zijn sinds 1996 natuurlijk ook nieuwe woorden bijgekomen. Het zou volgens mij goed zijn om de patronen te testen op een nieuwere woordenlijst, bijvoorbeeld die van OpenTaal.

Ik heb zelf hiermee wat geëxperimenteerd door het effect van de diverse in dit artikel genoemde patroonverzamelingen hierop los te laten.

Indien blijkt dat de bestaande patronen op de nieuwere woordenlijsten toch minder effectief zijn, dan is het aan te bevelen om nieuwe afbreekpatronen te genereren. Er zijn ook klachten dat op Nederlandstalige websites de afbrekingen niet goed werken[28]. Dit is ook weer een reden om nieuwe afbreekpatronen te genereren. Zoals hierboven opgemerkt, is dit het plan van OpenTaal.

Dan is er nog de mogelijkheid om het afbreekalgoritme aan te passen zoals het in LuaTEX gedaan is.

Joseph Wright schreef in november 2024[29] (met betrekking tot LaTEX tagging support): "For new documents, moving to LuaTeX is the recommended approach ... The time to move to LuaTeX for new LaTeX documents is here." Misschien geldt dit ook met betrekking tot het afbreekmechanisme, waarbij voor het Nederlands dan woorden als `cafeetje` (zoals eerder in dit artikel genoemd) correct afgebroken worden als de patronen worden aangepast aan het LuaTEX afbreekalgoritme. Maar veel mensen zullen nog PDFTEX en in mindere mate XeTEX gebruiken, en dat geeft dan een discrepantie.

Mijns inziens zou het aan te bevelen zijn om ook in PDFTEX en XeTEX (en vergelijkbare TEX implementaties) ditzelfde algoritme te implementeren. Wat het oorspronkelijke TEX van Knuth betreft is dit uitgesloten. Maar bij de andere zou het mogelijk moeten zijn, maar dit hangt af van degenen die deze implementaties beheren. Deze wijziging heeft geen gevolgen zolang de oorspronkelijke patronen gebruikt worden, dus achterwaartse compatibiliteit is in dit opzicht gewaarborgd.

Dankwoorden

Ik wil hierbij Gerard Kuiken, Piet Tutelaers, Frans Goddijn, Taco Hoekwater en Sander van Geloven bedanken voor hun inbreng.

Referenties

[1] Transforms and non-standard hyphenation with `luatex`. Babel Guides. `https://latex3.github.io/babel/guides/non-standard-hyphenation-with-luatex.html`.

[2] Anoniem. Werkgroep 13: 'neerlandica'; vergadering 9 november 1989. *MAPS*, 4:55–56, 1990. `https://www.ntg.nl/maps/04/19.pdf`.

[3] David Antoš. *Generation of Patterns with the OPatGen Program – User Guide*. Faculty of Informatics, Masaryk University Brno, Slovakia, 2001. Ook op `https://web.archive.org/web/20060622140352/http://www.fi.muni.cz/~xantos/patlib/thesis/userguide-p.pdf`. `https://giellalt.uit.no/proof/hyph/OOo/userguide-p.pdf`.

[4] David Antoš. Patlib, pattern manipulating library. Master's thesis, Faculty of Informatics, Masaryk University Brno, Slovakia, 2001. `https://web.archive.org/web/20060620002725/http://www.fi.muni.cz/~xantos/patlib/thesis/thesis-p.pdf`.

[5] David Antoš and Petr Sojka. Pattern generation revisited. *MAPS*, 2:7–17, 2001. Proceedings of the 16th European TEX Conference, Kerkrade, The Netherlands. `https://www.ntg.nl/maps/26/03.pdf`.

[6] Javier Bezos and Johannes L. Braams. *Babel user guide*, 2025. Locale documentatie met `texdoc babel`. `http://mirrors.ctan.org/macros/latex/required/babel/base/babel.pdf`.

[7] LuaTEX development team. *LuaTEX Reference Manual*, 2025. ch 5, pp. 73–89. `http://mirrors.ctan.org/systems/doc/luatex/luatex.pdf`.

[8] Willi Egger. 39ste NTG bijeenkomst, 06 2007. `https://www.ntg.nl/bijeen/39/verslag.pdf`.

[9] Frans Goddijn et al. De spelling van het lot. *MAPS*, 17:11–16, 1996. `https://www.ntg.nl/maps/17/08.pdf`.

[10] Yannis Haralambous. *A Revisited Small Tutorial on Patgen, 28 Years After*, 2021. In texlive: doc/support/patgen2-tutorial/patgen2-tutorial.pdf. `https://tug.ctan.org/info/patgen2-tutorial/patgen2-tutorial.pdf`.

[11] Onno Jager and Ruud Neyndorff. Wat er in 2005 'zomaar' in de spelling is veranderd. Blog post. `http://www.jagerneyndorff.nl/nl/wat-er-zomaar-is-veranderd`.

[12] H. Kerkman, R. Piepenbrock, R. Harald Baayen, and H. van Rijn. The Celex lexical database. CDROM, 1995. Exemplaar bij de Koninklijke Bibliotheek. `https://search.worldcat.org/title/celex-lexical-database/oclc/053446235`.

[13] Donald E. Knuth. *The TEXbook*. Addison Wesley, 1984.

[14] Donald E. Knuth. *TEX: The Program*, volume B of *Computers & Typesetting*. Addison Wesley, 1986.

[15] Frank M. Liang. *Word Hy-phen-a-tion by Comput-er*. PhD thesis, Stanford University, 1983. Report No. STAN-CS-83-977. `https://www.tug.org/docs/liang/liang-thesis.pdf`.

[16] Mojca Miklavec and Arthur Reutenauer. Putting the Cork back in the bottle — Improving Unicode support in TEX. *TUGboat*, 29(3):454–457, 2008. `https://www.tug.org/TUGboat/tb29-3/tb93miklavec.pdf`.

[17] Mojca Miklavec and Arthur Reutenauer. Hy-

phenation in TEX and elsewhere, past and future. *TUGboat*, 37(2), 2016. https://tug.org/TUGboat/tb37-2/tb116miklavec.pdf.

[18] NTG. NTG FILESERVER faciliteiten. *MAPS*, 7:15–17, 1989. https://www.ntg.nl/maps/03/07.pdf.

[19] László Németh. Automatic non-standard hyphenation in OpenOffice.org. *TUGboat*, 27(2):750–755, 2006. Proceedings of EuroTEX 2006. https://tug.org/tugboat/tb27-1/tb86nemeth.pdf.

[20] Prisma. *Spellinggids*. Het Spectrum, 2005.

[21] Maarten Sneep. Wachten op een ca-tas-tro-fe. *MAPS*, 33:2–3, 2005. https://www.ntg.nl/maps/33/02.pdf.

[22] Petr Sojka. Notes on compound word hyphenation in TEX. *TUGboat*, 16(3):290–296, 9 1995. Proceedings of the 1995 Annual TUG Meeting. https://www.tug.org/TUGboat/tb16-3/tb48soj2.pdf.

[23] Petr Sojka and Ondřej Sojka. The unreasonable effectiveness of pattern generation. *TUGboat*, 40(2):187–193, 2019. TUG 2019 Proceedings. https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf.

[24] Petr Sojka and Ondřej Sojka. New Czechoslovak hyphenation patterns, word lists, and workflow. *TUGboat*, 42(2):152–158, 2021. TUG 2021 Proceedings. https://www.tug.org/TUGboat/tb42-2/tb131sojka-czech.pdf.

[25] Petr Sojka and Pavel Ševeček. Hyphenation in TEX — Quo Vadis? *TUGboat*, 16(3), 1995. Proceedings of the 1995 Annual TUG Meeting. https://tug.org/tugboat/tb16-3/tb48soj1.pdf.

[26] Piet Tutelaers. Herziene afbreekpatronen voor het Nederlands. *MAPS*, 11:187–190, 1993. https://www.ntg.nl/maps/11/35.pdf.

[27] Gerard van Nes. Verslag NTG bijeenkomst 24 november 1988. *MAPS*, 2:2–11, 1988. https://www.ntg.nl/maps/02/01.pdf.

[28] Gerard van Nes. Verslag ntg bijeenkomst 11 mei 1989. *MAPS*, 3:1–8, 1989. https://www.ntg.nl/maps/03/01.pdf.

[29] Gerard van Nes. NTG DOS-diskette Distributie Service. *MAPS*, 4:29–30, 1990. https://www.ntg.nl/maps/04/11.pdf.

[30] Piet van Oostrum. TEX stuff at cs.ruu.nl. *MAPS*, 5:21–26, Nov 1990. https://www.ntg.nl/maps/05/08.pdf.

[31] Prof. Dr. P.G.J. van Sterkenburg and drs. M.E. Verburg. *van Dale Handwoordenboek van hedendaags Nederlands*. Van Dale Lexicografie, 1996.

Notes

1. https://tug.org/mail-archives/texhax/
2. https://ctan.org/pkg/texhax
3. https://ctan.org/tex-archive/obsolete/language/hyphenation/nl
4. https://en.wikipedia.org/wiki/IETF_language_tag
5. Persoonlijke communicatie
6. https://woordenlijst.org/
7. https://en.wikipedia.org/wiki/Trie
8. Zie bijv. de Pythonimplementatie van Ned Batchelder op https://nedbatchelder.com/code/modules/hyphenate.html, die ik (aangepast) gebruikt heb voor de voorbeelden in dit artikel
9. https://github.com/hunspell/hyphen
10. Zie de discussie op de tex-hyphen mailing list:https://tug.org/pipermail/tex-hyphen/2021-October/001901.html en https://tug.org/pipermail/tex-hyphen/2021-October/001894.html
11. Standaard staat \righthyphenmin op 3, daarom worden maatje en tomaatje niet als ...aat-je afgebroken
12. Persoonlijke communicatie
13. https://mirrors.ctan.org/language/hyph-utf8/tex/patterns/tex/hyph-nl.tex of in je lokale TEX-installatie onder zoiets als .../hyph-utf8/patterns/tex/hyph-nl.tex
14. https://taalmaterialen.ivdnt.org/download/celex-2-nl/
15. Het Groene Boekje is de officiële en normgevende woordenlijst met de belangrijkste Nederlandse woorden
16. Persoonlijke communicatie
17. Enigszins geredigeerd
18. Het Groene Boekje van deze spelling is in 1995 gepubliceerd; de elektronische versie in 1996
19. https://www.vandale.nl/pages/gratis-woordenboek
20. Zie texdoc fontenc
21. https://github.com/hyphenation/tex-hyphen
22. https://www.opentaal.org/
23. https://www.opentaal.org/licentie
24. https://github.com/OpenTaal/opentaal-wordlist
25. https://wiki.dante.de/trennmuster/projekt-wortdatenbank
26. https://github.com/OpenTaal/hyphenation-definitions
27. https://github.com/OpenTaal/opentaal-hyphenation
28. Persoonlijke communicate van Sander van Geloven
29. https://www.texdev.net/2024/11/05/engine-news-from-the-latex-project

Pieter van Oostrum
Leidsche Rijn
Utrecht
pieter@vanoostrum.org