Een meertalige interface naar TEX

J. Hagen

PRAGMA, Onderwijskundig Bureau voor Advies- en Ontwikkelwerk,
Postbus 125, 8000 AC Zwolle

Abstract

Macropakketten voor TEX zijn vrijwel altijd Engelstalig. In dit artikel wordt een meertalige interface beschreven, zoals die is toegepast in PPCHTEX, een pakket dat kan worden gebruikt om chemische structuurformules te zetten.

1 Inleiding

Kenmerkend voor de meeste macropakketten is dat ze Engelstalig zijn. Engels heeft bij het schrijven van macro's en programma's als grote voordeel dat de woorden korter zijn dan in het Nederlands. Bovendien bevordert een Engelstalige interface het gebruik van een pakket in het public domain.

Bij het ontwikkelen van macropakketten met een brede inzetbaarheid en flexibiliteit, ontkomt men niet aan een grote hoeveelheid macro's. Zeker een leek zal juist door deze hoeveelheid worden afgeschrikt in het gebruik ervan. Als een gebruiker al eenmaal zover is gekomen dat hij weet wat hij nodig heeft, dan wordt hij vervolgens geconfronteerd met de vraag of iets kan en welke macro's moeten worden aangeroepen. Mogelijk komt hier voor een beginnend gebruiker het taalprobleem nog eens bij.

2 Een interface

De auteur van dit artikel heeft de afgelopen jaren een macropakket ontwikkeld dat zo langzamerhand aardig compleet en veelzijdig mag worden genoemd. Omdat dit pakket, dat als werktitel CONTEXT heeft, in eerste instantie is ontwikkeld voor gebruik in de eigen organisatie, is de interactie Nederlandstalig. De macro's — soms ligt het meer voor de hand om over commando's te spreken — hebben daarbij verschillende, maar herkenbare vormen.

Zo vinden instellingen altijd plaats door middel van commando's met de vorm:

```
\stelietsin[variabele=waarde,variabele=waarde,...]
```

of

```
\stelietsin[optie,optie,...]
```

In beide gevallen staan de instellingen tussen `[ ]`. Er kunnen meerdere instellingen tegelijk, gescheiden door een comma, worden opgegeven. Enkele voorbeelden van instellingen zijn:

```
\stelwitruimtein[groot]
\stelopsommingin[opelkaar,kolommen]
\stelzetspiegelin[rugwit=4cm,kopwit=2.5cm]
```

Daarnaast kennen we definities. Deze hebben als vorm:

```
\definieeriets[naam]
```

of, als instellingen voor de hand liggen:

```
\definieeriets[naam][variabele=waarde,...]
```

Vrijwel altijd kunnen de instellingen ook in een later stadium plaatsvinden met:

```
\stelietsin[naam][variabele=waarde,...]
```

Een voorbeeld van zo'n, aan een 'naam' gekoppelde instelling is:

```
\stelkopin[hoofdstuk][letter=vet]
```

Hoewel hiermee de belangrijkste varianten van de interface gegeven zijn, gebied de eerlijkheid te zeggen dat er op deze regels uitzonderingen zijn. Bijvoorbeeld:

```
\doornummeren[vraag][plaats=inmarge]
\gebruikexternfiguur[logo][file001a]
                        [breedte=4em,kader=aan]
\definieerkop[rubriek][paragraaf]
\stelkopin[rubriek][letter=schuin]
```

Na het eerste commando is (onder andere) het commando `\vraag` beschikbaar waarmee vragen kunnen worden genummerd, waarbij het nummer in de marge wordt geplaatst. Met het tweede commando wordt een buiten TEX aangemaakt figuur gedefinieerd, waarbij `breedte` betrekking heeft op de breedte in de tekst (de figuur wordt automatisch geschaald). Om de figuur wordt een kader geplaatst. Na het derde commando is het commando `\rubriek` beschikbaar, dat zijn eigenschappen erft van het commando `\paragraaf`. Met het laatste commando wordt een van de eigenschappen van de zojuist gedefinieerde kop veranderd.

Veel typografische wensen zijn alleen te realiseren als gebruik wordt gemaakt van `\start`–`\stop`–constructies:

```
\startiets
.........................
\stopiets
```

Vaak kunnen ook hier instellingen worden meegegeven:

```
\startsmaller[2*links,rechts]
.........................
\stopsmaller
```

of

```
\startopsomming[n,ruim,opelkaar]
\som .......................
\som .......................
\stopopsomming
```

Daarnaast is er een \begin–\eind–constructie, die wordt gebruikt voor het markeren van tekstblokken. Gemarkeerde tekstblokken kunnen worden gezet, verborgen, gezet maar verborgen, verplaatst en|/|of elders worden opgeroepen.

```
\beginvaniets
.......................
\eindvaniets
```

Dergelijke blokken dient men eerst te definiëren:

```
\definieerblok[antwoord]
\stelblokin[antwoord][korps=klein]
\verbergblokken[antwoord]
.......................
\hoofdstuk{.......}
.......................
\beginvanantwoord
.......................
\eindvanantwoord
.......................
```

Commando's als de bovenstaande maken het mogelijk antwoorden in de ruwe tekst op te nemen en op de gewenste plaats op te roepen. Zo kunnen de antwoorden aan het eind van het hoofdstuk worden opgeroepen met:

```
\selecteerblokken[antwoord][criterium=hoofdstuk]
```

Er zijn nog vele andere vormen waarin commando's kunnen worden (en ook zijn) gegoten. Neem bijvoorbeeld:

```
\plaatsfiguur
  [links]
  [fig:logo]
  {Dit is een voorbeeld van logo.}
  {\naam{logo}}
```

Dit commando plaatst een figuur links naast de tekst, waarbij de tekst rond de figuur loopt. Deze figuur heeft als referentie `fig:logo`. Het derde argument is de titel. Als hier `geen` wordt gegeven, blijft de titel achterwege. Het laatste argument is de figuur zelf, in dit geval het eerder gedefinieerde logo.

In het laatstgenoemde commando zijn argumenten tussen [ ] optioneel. Dit is het geval bij meer commando's. Zo kan overal waar dat relevant is een referentie tussen [ ] worden meegegeven. Bovendien is een ruime layout van de commando's toegestaan.

3 Constanten en variabelen

Het zal inmiddels duidelijk zijn dat veelvuldig gebruik wordt gemaakt van keywords, zoals `links` of `letter`. Naarmate CONTEXT in omvang toenam, nam het beschikbare geheugen af. Dit was enerzijds een gevolg van het beslag dat referenties op dit geheugen leggen, anderzijds lag er een duidelijke relatie met het gebruik van keywords. Er dient immers veelvuldig te worden getest welke instelling actueel is. Plaatsen we de figuur bijvoorbeeld `links`, `rechts`, `hier`, `boven`, `onder`, op de pagina `naast` de huidige of op een aparte `pagina`. Bepaalde keywords kwamen dan ook tientallen malen voor in de source.

Een oplossing voor dit probleem is gevonden in het gebruik van constanten. Enig experimenteren leerde namelijk al snel dat het gebruik van constanten niet alleen een aanzienlijke besparing in geheugen opleverde, maar ook snelheidswinst. In de source van CONTEXT komen we dan ook erg veel constanten tegen. Deze zijn in de source te herkennen aan de vorm:

```
\!!links \!!rechts \!!hier \!!boven \!!onder
                                \!!naast \!!pagina
```

Hierbij is bijvoorbeeld de constante `\!!links` gedefinieerd als:

```
\def\!!links {links}
```

Naast constanten kennen we ook variabelen. Een variabele wordt gevormd uit een constante en een label. Het mechanisme dat de instellingen afhandelt, genereert de variabele. Zo is het commando:

```
\stelzetspiegelin[hoogte=20cm]
```

verantwoordelijk voor de declaratie:

```
\def\@@zshoogte{20cm}
```

Ieder setup–commando heeft een karakteristiek label, in dit geval `\??zs`. Het label `\??zs` expandeert tot `@@zs` en vormt samen met de constante `\!!hoogte` de variabele `\@@zshoogte` met, in dit geval, de waarde `20cm`.

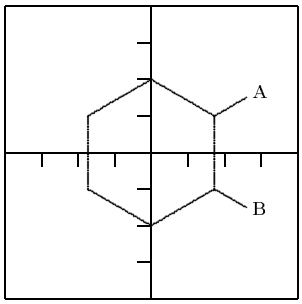
Gebruik van dit mechanisme heeft in de loop der tijd een aantal positieve neveneffecten gehad. Zo resulteert het gebruik van constanten in een consistente interface en een goed leesbare source. Overigens is het mechanisme voor de gebruiker verborgen.

4 Tweetalige macro's

In aanvulling op CONTEXT is een macropakket ontwikkeld voor het zetten van chemische formules: PPCHTEX. Omdat dit pakket nauwelijks terugvalt op specifieke CONTEXT–commando's is het ook met andere pakketten te combineren. Een van de 'problemen' die zich daarbij aandient is dat, wil aansluiting bij andere macropakketten mogelijk zijn, de interface bij voorkeur Engelstalig moet zijn, terwijl hij in ons geval juist Nederlandstalig is.

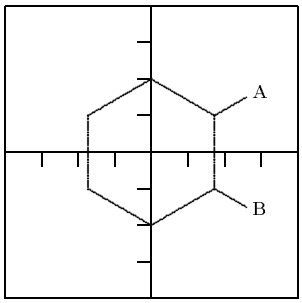
Omdat eenzelfde interface wordt gebruikt als binnen CONTEXT, wordt ook binnen PPCHTEX gebruik gemaakt van constanten en variabelen. Geconfronteerd met de noodzakelijke aansluiting op bijvoorbeeld LATEX, is daarom onderzocht in hoeverre een tweetalige implementatie mogelijk was, zonder dat daartoe ingrijpende wijzigingen in de source nodig zijn. Gebleken is dat inderdaad vrij eenvoudig een meertalige implementatie mogelijk is.

Afhankelijk van de voorliefde van de gebruiker, kunnen we daarom met PPCHTEX een structuurformule zetten met Nederlandse commando's:




```
\startchemie[assenstelsel=aan,kader=aan]
  \chemie[SIX,B,R12,RZ12][A,B]
\stopchemie
```

of, mits daartoe de juiste voorbereidingen zijn getroffen, met Engelstalige commando's:




```
\startchemical[axis=on,border=on]
  \chemical[SIX,B,R12,RZ12][A,B]
\stopchemical
```

We zien dat zowel de commando's als de instellingen in het Engels worden opgegeven.

5 Een oplossing

Een van de problemen die zich voordoet bij omschakeling van de brontaal naar een nieuwe taal, volgt uit het gegeven dat de macro's geschreven zijn in de brontaal en dat aansluitend daarop reeds instellingen in de brontaal hebben plaatsgevonden.

Een nadere analyse van de eerder gegeven commando's ten behoeve van instellingen leert dat links van de = altijd constanten voorkomen.

```
\stelchemiein  [assenstelsel=aan,  kader=aan]
\setupchemical [        axis=on,  border=on]
```

Beide aanroepen moeten resulteren in de onderstaande variabelen:

```
\@@chemieassenstelsel
\@@chemiekader
```

Bij het samenstellen van deze variabelen is het label `\??chemie` gebruikt. Omdat links van de = altijd constanten staan, is conversie eenvoudig te realiseren: `axis` moet `assenstelsel` worden en `border` moet worden omgezet in `kader`. Dit kan met behulp van de hulp–constanten:

```
\def\!!axis   {assenstelsel}
\def\!!border {kader}
```

Lastiger is het als we rechts van de = kijken. Hier staat namelijk niet altijd een constante. Beide onderstaande instellingen zijn namelijk toegestaan.

```
\stelchemiein  [schaal=1000]
\stelchemiein  [schaal=klein]
```

of in het Engels:

```
\setupchemical [scale=1000]
\setupchemical [scale=small]
```

Er kan rechts zelfs een `\macro` staan of een `\teller`. Vertalen is hier dan ook ongewenst en bovendien technisch vrijwel onmogelijk. Gelukkig is het ook niet nodig.

Kijkend naar deze instelling zal duidelijk zijn dat bij het afhandelen van de schaalinstelling gekeken moet worden of het een standaard instelling betreft of niet. Er kan hier namelijk een van de opties `klein`, `middel` en `groot` of een getal worden opgegeven. In PPCHTEX wordt de variabele `\@@chemieschaal` daartoe vergeleken met de constanten `\!!klein`, `\!!middel` en `\!!groot`. In een Engelstalige interface moeten deze constanten dan ook een andere betekenis krijgen:

```
\def\!!klein  {small}
\def\!!middel {medium}
\def\!!groot  {big}
```

Immers, `\@@chemieschaal` is in het geval van een Engelstalige interface `small`, `medium`, `big` of een getal.

We moeten dus dat wat de gebruiker links van de = opgeeft wel, en dat wat hij rechts opgeeft niet vertalen. Met andere woorden:

```
\stelchemiein  [assenstelsel=aan,    kader=aan]
\setupchemical [        axis=on,    border=on]
```

moet opleveren:

```
\def\@@chemieassenstelsel{aan}
                    \def\@@chemiekader{aan}
\def\@@chemieassenstelsel{on}
                     \def\@@chemiekader{on}
```

6 Een implementatie

In de LATEX–stylefile bij PPCHTEX vinden we onder andere de volgende commando's:

```
\defineconstant  {assenstelsel}
\defineconstant  {kader}
\defineconstant  {schaal}
```

Hiermee wordt een deel van de constanten gedefinieerd die worden gebruikt in PPCHTEX. Deze definitie is niet nodig bij gebruik van PPCHTEX met CONTEXT, omdat daar deze constanten reeds zijn gedefinieerd.

Een Engelstalige interface in `m-chemie.sty` gedefinieerd met behulp van (onder andere) de onderstaande commando's.

```
\defineinterface  [english]

\redefinecommand  {stelchemiein}  {setupchemical}

\redefineconstant {assenstelsel}  {axis}
\redefineconstant {kader}         {border}
\redefineconstant {schaal}        {scale}
```

Na deze definities zijn beide interfaces beschikbaar. Meer interfaces zijn mogelijk, maar op dit moment in PPCHTEX nog niet gedefinieerd. Een wisseling van interface vindt plaats met:

```
\setinterface[english]
```

Terugkeer naar de oorspronkelijke (Nederlandstalige) interface vindt plaats met het commando `\resetinterface`. Als de omschakeling binnen een groep ({ }) heeft plaatsgevonden, dan is dit laatste commando niet nodig.

Het commando `\redefinecommand` voegt synoniemen toe aan een lijst. Voor iedere extra taal wordt zo'n lijst bijgehouden:

```
{...,axis=>assenstelsel,border=>kader,
                          scale=>schaal,...}
```

Als we in de brontaal (Nederlands) werken dan geldt:

```
\def\!!kader {kader}
```

Als we echter in een andere taal werken, bijvoorbeeld Engels, dan geldt:

```
\def\!!kader  {border}
\def\!!border {kader}
```

Het commando `\setinterface` draagt zorg voor het herdefiniëren van de constanten en maakt daarbij gebruik van de lijst. Hoewel een implementatie zonder lijst mogelijk is, heeft het werken met een lijst als voordeel dat naar hartelust kan worden gewisseld van taal. Bovendien maakt het werken met lijsten implementatie in meer dan twee talen mogelijk.

Hieronder zijn voor de liefhebbers enkele macro's opgenomen die achter de schermen het werk doen. Voor meer tekst en uitleg van deze (en aangeroepen) macro's verwijzen we naar de documentatie van de CONTEXT–module `cont-00a.tex`.

We beginnen met het declareren van een switch waarmee we bijhouden of we in een andere taal werken.

```
\newif\ifsomeinterface
```

Het commando `\defineinterface` activeert de lijst waarin de nog te definiëren synoniemen worden opgeslagen.

```
\def\defineinterface[#1]%
  {\def\currentdefineinterface%
     {!!interface!!#1}}
```

Met `\redefinecommand` laten we het nieuwe commando het in de brontaal geschreven commando aanroepen.

```
\def\redefinecommand#1#2%
  {\doifnot{#1}{#2}%
     {\setvalue{#2}{\getvalue{#1}}}}
```

Bij `\redefinecommand` wordt de nieuwe naam van de constante aan de lijst met synoniemen toegevoegd.

```
\def\redefineconstant#1#2%
  {\doifundefined{!!#2}%
     {\doifundefined{\currentdefineinterface}%
         {\setevalue{\currentdefineinterface}%
                    {\!!dummy=>\!!dummy}}%
      \setevalue{\currentdefineinterface}%
         {\getvalue{%
              \currentdefineinterface},#2=>#1}}}
```

De onderstaande hulpmacro doorloopt de lijst en roept voor ieder paar synoniemen de macro `\dodoswapinterface` aan, die de nodige toekenningen doet.

```
\def\dodosetinterface%
  {\def\doswapinterface##1%
     {\doifnot{##1}{}{\dodoswapinterface##1\\}}%
   \processcommacommand[\currentinterface]%
                               \doswapinterface}
```

De volgende twee commando's zijn verantwoordelijk voor het wisselen van interface. Hier wordt de hulpmacro `\dodoswapinterface` gedefinieerd.

```
\def\resetinterface%
  {\ifsomeinterface
     \def\dodoswapinterface##1=>##2\\%
       {\setvalue{!!##1}{##1}%
        \setvalue{!!##2}{##2}}%
     \dodosetinterface
     \someinterfacefalse
   \fi}

\def\setinterface[#1]%
  {\resetinterface
   \def\currentinterface%
     {!!interface!!#1}%
   \def\dodoswapinterface##1=>##2\\%
     {\setvalue{!!##1}{##2}%
      \setvalue{!!##2}{##1}}%
   \dodosetinterface
   \someinterfacetrue}
```

De macro `\getparameters` speelt de sleutelrol bij het definiëren van en toekennen aan variabelen:

```
\def\dosetvalue#1#2#3%
  {\doifdefinedelse{!!#2}%
     {\setvalue{#1\getvalue{!!#2}}{#3}}%
     {\setvalue{#1#2}{#3}}}

\def\doassign[#1][#2=#3]%
  {\dosetvalue{#1}{#2}{#3}}%

\def\getparameters[#1]#2[#3]%
  {\def\!!dogetparameter##1%
     {\doassign[#1][##1]}%
   \processcommalist[#3]\!!dogetparameter}
```

Het onderstaande voorbeeld toont hoe deze macro kan worden ingezet (`\??chemie` staat voor `@@chemie`).

```
\def\stelchemiein[#1]%
  {\getparameters[\??chemie][#1]}
```

In het onderstaande fragment van de PPCHTEX–macro `\startchemie` zien we op welke wijze de waarden van instellingen worden geïnspecteerd:

```
\def\startchemie[#1]%
  {.....................
   \doif{\@@chemiebreedte}{\!!passend}
     {.............}
   .....................
   \doifinset{\!!aan}{%
      \@@chemiekader,\@@chemieassenstelsel}
     {.............}
   ....................}
```

Omdat we bij de vergelijking gebruik maken van constanten, die zich aanpassen aan de interactie–taal, hoeven we in `m-chemie.tex` deze macro's dus niet aan te passen.

7 Een complicatie

Tijdens het zoeken naar de meest geschikte oplossing voor een meertalige interface, zijn verschillende varianten van de bovenstaande macro's de revue gepasseerd. De bovenstaande oplossing legt relatief weinig beslag op het geheugen. Bovendien neemt de executietijd nauwelijks toe.

Een van de grootste problemen bij het vinden van een oplossing lag in het feit dat er ook 'variabele' variabelen voorkomen. Zo is binnen CONTEXT het volgende commando beschikbaar (in werkelijkheid is deze macro iets anders gedefinieerd):

```
\def\stelkopin[#1][#2]%
  {\getparameters[\??kop#1][#2]} % \??kop = @@kop
```

Dit betekent dat na de aanroep

```
\stelkopin[hoofdstuk][letter=vet]
```

de variabele `\@@kophoofdstukletter` de waarde `vet` heeft.

Het kan voorkomen dat we in de source gebruik willen maken van de algemene vorm, bijvoorbeeld omdat we een hulp–macro hebben die op alle soorten koppen werkt. In dat geval wordt in de source een wat andere vorm gebruikt voor de variabele: `\@@kop#1\!!letter` met `#1=letter`.

```
\def\dosomething#1%
  {.....................
   \doif{\getvalue{\@@kop#1\!!letter}}{\!!vet}
     {.............}
   ....................}
```

Eerder zagen we echter dat, bij gebruik van een andere interactie–taal, constanten als `\!!letter` een Engelstalige betekenis hebben gekregen. Na implementatie van het tweetalig mechanisme bleek dat daarom aanroepen als de bovenstaande moeten worden vervangen door:

```
\doif{\dogetvalue{\@@kop#1}{\!!letter}}{\!!vet}
```

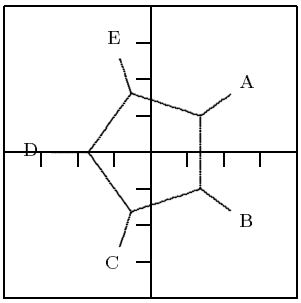
waarbij `\dogetvalue` er voor zorgt dat met de Nederlandse betekenis van `\!!letter` wordt gewerkt.

```
\def\dogetvalue#1#2%
  {\getvalue{#1\getvalue{!!#2}}}
```

Hoewel het bovenstaande verhaal op het eerste gezicht misschien wat ingewikkeld lijkt, valt het in de praktijk gelukkig mee. De source van PPCHTEX hoefde namelijk op slechts op drie plaatsen te worden aangepast.

8 Tot slot

Het geschetste mechanisme werkt naar behoren. Als daar behoefte aan is en als de tijd het toestaat, zal CONTEXT te zijner tijd in meer talen beschikbaar komen in het public domain. De tweetalige implementatie van PPCHTEX wordt in dat kader gezien als een experiment, evenals het onderstaande voorbeeld in hsilgne.




```
\setinterface[hsilgne]

\lacimehctrats[sixa=no,redrob=no]
\lacimehc[FIVE,B,R,CRZ][A,B,C,D,E,F]
\lacimehcpots
```

Een logisch vervolg op het meertalig maken van de interactie is het genereren van meertalige meldingen. Dit staat dan ook op de agneda.