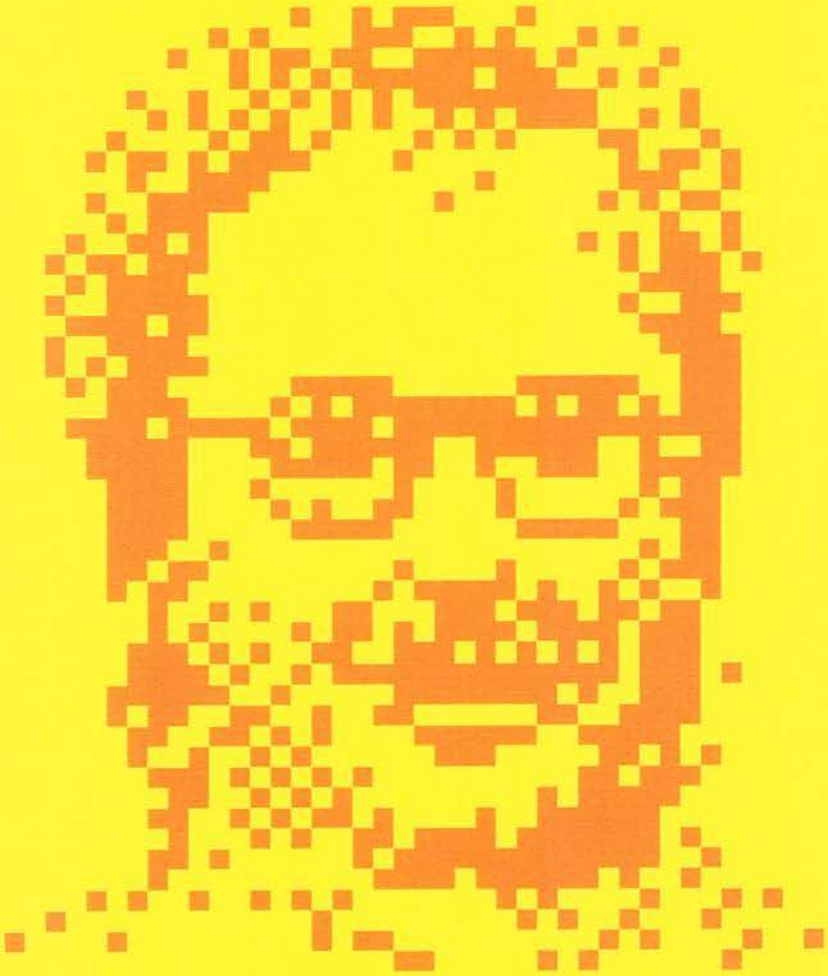
Liber Amicorum

Lambert Meertens CWI



1966 - 1991

Lambert Meertens

CWI

Liber Amicorum

1966-1991

Een uitgave van de Stichting Mathematisch Centrum ter gelegenheid van het 25-jarig dienstverband van Lambert Meertens met het Mathematisch Centrum,
Amsterdam, januari 1991

samengesteld door:	Johan Jeuring
ontwerp omslag:	R.T. Baanders en Steven Pemberton
druk:	CWI, Amsterdam

Voorwoord

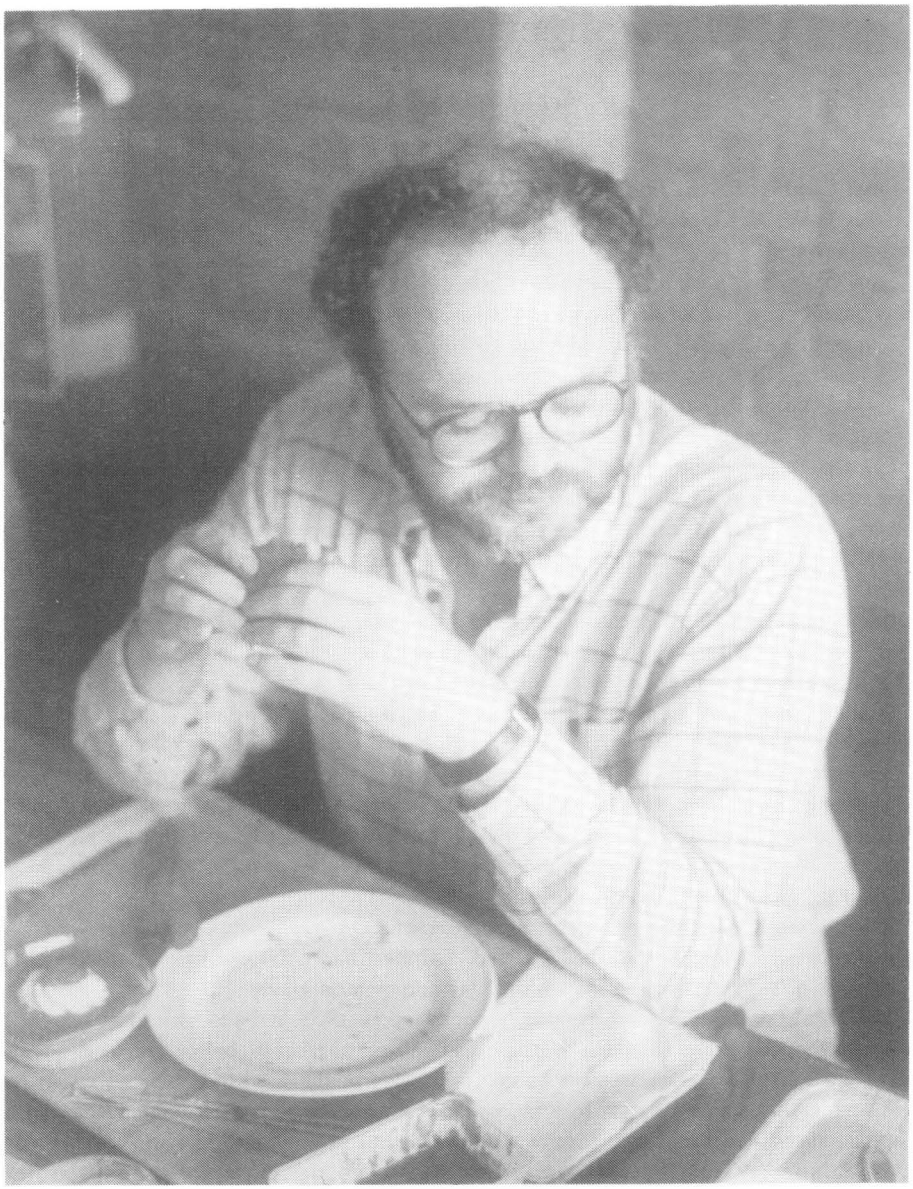
Deze bundel is samengesteld ter gelegenheid van het 25-jarige dienstverband van Lambert Meertens met het Centrum voor Wiskunde en Informatica. In deze 25 jaar heeft Lambert zich wat het onderzoek betreft voornamelijk bezig gehouden met het ontwikkelen van programmeertalen en programmeermethodologieën. Zo was hij een van de auteurs van het rapport 'Revised Report on the Algorithmic Language ALGOL68', en stond hij aan de basis van de programmeertaal ABC. Samen met Richard Bird heeft hij in de tachtiger jaren gewerkt aan de ontwikkeling van een programmeermethodologie, die nu bekend staat als het Bird-Meertens formalisme of Squiggol. Een rode draad in Lamberts werk is de taal; dit wordt weerspiegeld in de inhoud van de artikelen geschreven voor deze bundel.

U treft een bonte verscheidenheid van artikelen, foto's en andere zaken aan in deze bundel. Ter gelegenheid van het 25-jarig jubileum van Lambert Meertens is een feestdag georganiseerd op 25 januari 1991. Op deze dag hebben een drietal wetenschappers en vrienden van Lambert een voordracht gehouden. Aan hen is tevens gevraagd een schriftelijke bijdrage te leveren aan deze bundel. Verder is door middel van een rondschrijven aan collega's, vrienden en bekenden van de jubilaris een uitnodiging uitgegaan om een bijdrage te leveren voor dit Liber Amicorum. In dit rondschrijven is expliciet gevraagd naar informele en ludieke bijdragen, maar de uiteindelijke keuze van de aard van de bijdragen is vrij gelaten.

Ik heb veel plezier gehad in het samenstellen van deze bundel, en ik hoop dat de lezer, met name Lambert, er net zoveel plezier aan beleeft. Rest mij nog dank te zeggen aan alle mensen die meegeholpen hebben aan de totstandkoming van deze bundel. Allereerst natuurlijk bedank ik alle auteurs, die gezorgd hebben voor een mooie collectie grappige en boeiende verhalen. Verder wil ik ook T. Baanders, die de voorplaat heeft verzorgd en Steven Pemberton, die voor de voorplaat zowel ideeën als een figuur heeft aangedragen, bedanken. Tenslotte bedank ik de organisatoren van de feestdag ter gelegenheid van het 25-jarig jubileum van Lambert Meertens. De feestcommissie bestond uit Marja Hegt, Jack Jansen, Steven Pemberton en ondergetekende.

Johan Jeuring

Inhoud



My recollections of Lambert

Krzysztof R. Apt

I met Lambert for the first time in 1975, soon after I joined the Mathematical Centre (in short MC). MC was located then then at Boerhaavestraat 49, just behind the now defunct Amstel brewery.

Paul Vitanyi, with whom I shared an office, quickly informed me that "Lambert is a smart fellow" and I soon had an occasion to convince myself how true this statement was. During my 3 year stay at the MC I learned a lot from him.

Lambert's office was located in a run down barrack adjacent to the back of the building of the MC. Inside it there was a mess, as big as today. Nevertheless Lambert claimed that he could find everything within seconds.

My realization that one can improve one's writing skill originated with our first joint paper when Lambert gently threw my first draft into the paper bin. His counter-proposal was infinitely better drafted, and it made me aware that elegance goes together with simplicity. The third author unfortunately did not have the opportunity to go through this illuminating experience.

Lambert had a habit of recording mistakes which occurred in published reports or papers of his. He would set aside one corrected copy and mark it with "COR", perhaps in anticipation of his future boss's name. I adopted this habit from him.

In those days Lambert was very active in PSP, the Pacifist Socialist Party. Once I met him by chance at the Erasmus University in Rotterdam during the annual congress of the Dutch Computer Science Society. I suggested that we return to Amsterdam together by train. At the platform it turned out that Lambert had a 2nd class ticket whereas I had a 1st class ticket. He did not approve of my elitist attitude and told me that his party is for abolishing all remnants of class differences including the one in trains. To my remark why not then to abolish 2nd class he chose not to reply.

Therefore you can imagine my astonishment when in August this year while we took a train together to Utrecht, the rôles reversed: Lambert had a 1st class ticket whereas I had a 2nd class one. "1st class is less crowded" he commented briefly.

In 1978 I left the MC and eventually ended up in 1981 in Paris. In 1986 I spent six weeks at the MC, which in the meantime had moved southeast, below sea level and was rechristened CWI. For some reason I used to go then to the CWI in a jacket (but without a tie), perhaps because I forgot to bring with me a sweater from Paris. Lambert could not contain himself and eventually commented wryly with obvious disapproval "What kind of jacket is this?". "From Poland" I answered meekly but this did not change his attitude. In 1987 I returned to Amsterdam and joined the CWI again. Some time later Lambert switched his one-day-a-week professor appointment from Delft to Utrecht. He said that Delft left a lot to be desired. "All professors go there in jackets". Therefore you can imagine my astonishment when last year I spotted Lambert at the CWI walking dressed in a too-tight smart jacket. I commented wryly "What kind of jacket is this?". It turned out that it was the day the CWI was evaluated by the external committee.

In 1986 I spent my vacation in Indonesia. I learned that on Java people are so polite that they never answer "No" but only *"belum"* ("Not yet"). When I told this to Lambert his response was: "Has your father already died?" and I felt a bit stupid.

Lambert's political opinions do not leave much space for private cars (in this he shares the views of Albanian leaders). Therefore I felt very honored in 1988 when I was the selected one who transported his Mac by car to his apartment.

In Lambert's apartment, much neater than his office – perhaps because it is much less often inhabited – juggling balls were on a prominent display. Lambert told me then that juggling was a difficult art to master. Indeed, he had already entered his second year of practising and still could not handle three balls.

Therefore you can imagine my astonishment when some time later I mastered this art within three days after joining Carol's course at the CWI, moreover as the last one from my group.

Nowadays Lambert is more into collecting the copy cards left in the xerox machine located next to his office. For some reason he does not mind the noise - his office door is always wide open. For some reason I do mind noise so my office door is usually closed. "Not good" admonishes Lambert "Why are all the doors of the AP Department always closed?" he despairs.

Throughout all these years Lambert has patiently corrected my spoken Dutch and written English (this piece excluded). Some two years ago he disapproved my pronunciation of the to me unpronouncable Dutch word *"dus"*. *"Deuce"* corrected Lambert and all of a sudden I recalled that he had already corrected me on this very word in the late seventies. I shared with him this astonishing recollection. "Hopeless case" he must have thought.

May I disagree Lambert. For most of my reports I have now a COR-copy, I have not worn a jacket for years now, my writing skills have improved, I now travel by train only in 2nd class and can juggle with three balls. The fact that our opinions on most of these points still differ is admittedly my fault - I should have sticked to my initial positions.

Lambert, I hope to continue our (ex)changes of opinions during the next 25 years.

All the best,
your friend Krzysztof

Amsterdam, December 1990

Meertens' Number

Richard Bird
Programming Research Group, Oxford University

Lambert Meertens has been my good friend and collaborator for over a decade. To honour the occasion of his celebration of twenty-five years at the CWI, I would like to give him something. Usually I bring a bottle of Glenlivet as a gift when I visit Lambert; this time I bring just a number. There is a good chance that the distinguishing property of this number has not been thought of before, so Meertens' number is unlikely to turn out to belong to someone else.

The property I have in mind came to me one night as I got ready for bed. My wife and I have a digital alarm clock in our bedroom and the four digits glow with a baleful green light, as if to challenge one to some kind of computational battle. As a result I have recently fallen into the habit of spending a few moments trying to determine their prime factorisation. I call this period my prime time. Fortunately, the clock has a twelve hour display, so 12.59 is the largest number I can encounter. Going to bed early, or late, has its advantages.

It occurred to me one evening that it would make the job easier if there was a number whose prime decomposition was manifest in its digits. More precisely, I wondered whether there was a natural number n with decimal representation $d_1 d_2 \ldots d_n$ and prime decomposition $2^{d_1} 3^{d_2} \ldots p_n^{d_n}$. The existence of such a jewel was not at all obvious. The number would have to be even, and a little mental calculation shows that it cannot have one, two, or three digits. For example, if a three digit number exists, then it has to end with a zero and so be both divisible and not divisible by five. I am told by a mental gymnast that similar reasoning shows that no four digit number exists either. On the other hand, as n increases, the prime decomposition corresponding to the decimal representation of n jumps about quite a lot, so there is no reason why they should never coincide.

By the way, you may think that I have posed the problem in a rather unsatisfactory way, at least from an artistic point of view. Surely, it would be more appropriate to associate the most significant digits with the largest primes, not the smallest. Fortunately, there is a good answer to that. The (smallest) number n with digits $d_1 d_2 \ldots d_n$ and such that $n = 2^{d_n} \ldots p_n^{d_1}$ is 12. Lambert would not forgive me if I named such a common or garden number as 12 after him. In any case, we already have a Baker's dozen and a Dirty dozen.

The problem is not specific to decimals of course. In base 2, each of the numbers 2, 6 and 10 is a Meertens' number. For example, 6 is 110 in binary and $6 = 2^1 3^1 5^0$. In base 3, the number 10 is a Meertens' number. In fact, for bases b, where $2 \leqslant b < 10$, the first Meertens' numbers are:

base	Meertens number	decimal equivalent
2	$(10)_2$	2
3	$(101)_3$	10
4	$(3020)_4$	200
5	$(11)_5$	6
6	$(130)_6$	54
7	$(202)_7$	100
8	$(330)_8$	100
9	$(781000)_9$	4199040

The above table is encouraging, but finding the Meertens' number for mere bits and bytes is not enough. Mere tens, on the other hand, would be quite appropriate.

There is no alternative but to compose a computer program to try and settle the question. Naturally, we shall develop this program according to Squiggolistic principles.

Expressed in Squiggol, x is a Meertens' number if $x = \exp x$, where

$$\text{(0)} \qquad \exp x \;=\; \times/\,(\text{primes}\ \curlyvee_{\text{up}}\ \text{digits}\ x)$$

and $x \text{ up } y = x^y$.

Squiggolists worth their salt will immediately know that the way to specify the function digits is in terms of the inverse image of a homomorphism on *Snoc* - lists:

$$\text{(1)} \qquad \text{digits} \;=\; \sqcap_{\#}/ \cdot \text{Inv}(\oplus\!\!\not\!\!\rightarrow_0)$$

where $n \oplus d = 10n + d$. (These days, with so many young squiggolists around, the notation $\oplus\!\!\not\!\!\rightarrow_0$ is regarded as old-hat. I believe these monkeys prefer to use something called "banana" brackets.) After some fairly hefty work applying GATFISH (Greedy Algorithm Theorem For Inverses of Snoc Homomorphisms — the full proof of this theorem has been eagerly awaited for over ten years), expert squiggolists will be able to come up with the following recursive program for digits:

$$\text{(2)} \qquad \text{digits}\, x \;=\; \begin{cases} [\,], & \text{if } x = 0 \\ \text{digits}\,(x \,\mathbf{div}\, 10) +\!\!+ [x \,\mathbf{mod}\, 10], & \text{otherwise} \end{cases}$$

Novitiates will have to take this deep result on trust.

We can characterise the Meertens' numbers as those which satisfy the predicate

$$(=) \cdot (\text{id} \parallel \exp)$$

(The symbol $\parallel$ is due to Lambert, who is well-known for inventing notation in bars.) Hence the Meertens' numbers in increasing order are given by:

$$\text{(3)} \qquad ms \;=\; ((=) \cdot (\text{id} \parallel \exp)) \triangleleft [1 \ldots]$$

The above expression is executable in at least one programming language I know, but is likely to prove time consuming.

To try and speed up the computation, we shall first break the search into pieces, confining each search to an interval $[10^{N-1} \ldots 10^N - 1]$ consisting of numbers of decimal length N.

The interval $[l \ldots r]$ can be computed in many ways. In particular, we have

$$\text{(4)} \qquad [l \ldots r] = + \,/\!\!\!\!\!\!\rightarrow_l [1]^{r-l}$$

where $[1]^n$ means the list consisting of 1 repeated n times.

Thus we have

$$
\begin{aligned}
& ms \\
= \quad & \{\text{equation (3)}\} \\
& ((=) \cdot (\mathrm{id} \parallel \exp)) \triangleleft [1 \ldots] \\
= \quad & \{\text{law: } (f \cdot g)\triangleleft = g^{-1}* \cdot f\triangleleft \cdot g*\} \\
& ((\mathrm{id} \parallel \exp)^{-1}* \cdot (=)\triangleleft \cdot (\mathrm{id} \parallel \exp)*)[1 \ldots] \\
= \quad & \{\text{law: } (\mathrm{id} \parallel \exp)^{-1} = \pi_1\} \\
& (\pi_1* \cdot (=)\triangleleft \cdot (\mathrm{id} \parallel \exp)*)[1 \ldots] \\
= \quad & \{\text{equation (4) — could have been used earlier}\} \\
& (\pi_1* \cdot (=)\triangleleft \cdot (\mathrm{id} \parallel \exp)* \cdot +\!\!\!\!/_1)[1]^\infty
\end{aligned}
$$

At this point we apply finite differencing, a technique explained in that seminal work 'A Page from a Book in Algorithmics' by L.G.L.T. Meertens. (This work was never published but was circulated privately to members of WG2.1. A discerning reviewer claimed it was 'refreshingly brief'.) In symbols, we have

$$f* \cdot \oplus\!\!\!\!/_e = \otimes\!\!\!\!/_{f\,e}$$

provided $f\,(a \oplus b) = f\,a \otimes b$.

Our aim is to apply this rule, and to that end we consider the term

$$(\mathrm{id} \parallel \exp)\,(x+1) = (x+1, \exp\,(x+1))$$

Using (0) we are lead to consider the term $\mathrm{digits}\,(x+1)$. (By the way, the phrase 'we are lead to consider' is often used by squiggolists to suggest that the big rabbit sitting at their elbow does not in fact exist. Not here of course.) Straightforward calculation (and this phrase always means that the author cannot see how to do it) gives for $10^{N-1} \leqslant x < 10^N - 1$ that:

$$\mathrm{digits}(x+1) = \mathrm{digits}\,x \uparrow (n-1) +\!\!+ [d+1] +\!\!+ [0]^{N-n}$$

where $d = \mathrm{last}\,(\mathrm{digits}\,x \uparrow n)$ and

$$n = \sqcap/\,(\mathrm{all}\,(=9) \cdot (\mathrm{digits}\,x \downarrow)) \triangleleft [0 \ldots N]$$

In the above equations $(\uparrow n)$ denotes the function that takes the first n elements of a list, and $(\downarrow n)$ is the function that drops the first n elements. There are, I think, a number of research groups currently thrashing out the complete theory of the $\uparrow\downarrow$ calculus.

Now we have

$$\begin{array}{ll} & \exp(x+1) \\ = & \quad \{(0) \text{ and above calculation}\} \\ & \times/\,(\text{primes} \curlyvee_{\text{up}} (\text{digits}\, x \uparrow (n-1) +\!\!+ [d+1] +\!\!+ [0]^{N-n})) \\ = & \quad \{\text{arithmetic}\} \\ & \times/\,(\text{primes} \curlyvee_{\text{up}} (\text{digits}\, x \uparrow (n-1)) \times e \end{array}$$

where

$$e = (\text{pr}\,(n-1))^{d+1}$$

and $\text{pr}\, n$ is the nth prime. (I think that $\text{pr}\, 0 = 2$, but will have to check with EWD.)

The above calculation leads us to consider (see, no rabbits!)

$$\text{exps}\, x = (\times/\, \cdot (\text{primes} \curlyvee_{\text{up}})) * \text{inits}^{+}\, (\text{digits}\, x)$$

where $\text{inits}^{+}\, ds$ returns the non empty initial segments of the sequence ds. In particular, the last element of $\text{exps}\, x$ is just $\exp x$. Then we have

$$\text{exps}\,(x+1) = \text{exps}\, x \uparrow (n-1) +\!\!+ [e]^{N+1-n}$$

where

$$e = \text{pr}\,(n-1) \times \text{last}\,(\text{exps}\, x \uparrow n)$$

Putting this all together, we have

$$(\text{id} \parallel \text{exps} \parallel \text{digits})\,(x+1) = (\text{id} \parallel \text{exps} \parallel \text{digits})\, x \otimes 1$$

where

$$(x, es, ds) \otimes 1 = (x+1, es \uparrow (n-1) +\!\!+ [e]^{N+1-n}, ds \uparrow (n-1) +\!\!+ [d+1] +\!\!+ [0]^{N-n})$$

and where d, e, and n are as defined above.

The Meertens' numbers of N digits are thus computed by

$$ms\ N = \pi_1 * \mathrm{eq} \triangleleft \otimes \#\!\!\!\rightarrow_a 1^b$$

where $a = (10^{N-1}, [2]^N, [1] +\!\!\!+ [0]^{N-1})$, and b is something like $9 \times 10^{N-1} - 1$, and $\mathrm{eq}\,(x, es, ds) = (x = \mathrm{last}\ es)$.

Although the result can be coded quite easily in an imperative programming language, the result is still inefficient. What we have omitted to take into account is the possibility of pruning the search whenever a value of e is encountered that is too big, that is, when $e \geqslant 10^N$. But enough is enough.

Omitting details, this is the program I finally derived: (in addition to maintaining an extra array xs, both ds and es have an extra initial value, and I've taken $pr[1] = 2$):

```
ds[0], es[0], xs[0] := 0, 1, 0;
ds[1], es[1], xs[1] := 1, 2, 1;
n := 1;
do n ≠ N →
   n := n + 1
   ds[n], es[n], xs[n] := 0, es[n − 1], 10 × xs[n − 1]
od;
do ds[0] ≠ 1 →
   if  xs[N] = es[N] → output xs[N]
   []  xs[N] ≠ es[N] → skip
   fi ;
   n := N;
   do ds[n] = 9 ∨ es[n] × pr[n] ⩾ 10^N →
      n := n − 1
   od;
   ds[n], es[n], xs[n] := ds[n] + 1, es[n] × pr[n], xs[n] + 1;
   do n ≠ N →
      n := n + 1;
      ds[n], es[n], xs[n] := 0, es[n − 1], 10 × xs[n − 1]
   od
od
```

Fortunately, when executed the program was safely delivered of a rather

pretty eight digit number. There are no nine or ten digit Meertens' numbers, but I haven't looked further.

So, Lambert, here finally is the number which I give to you as a token of our friendship:

$$\boxed{81312000}$$

Handle it carefully and keep it cool, for it decomposes rather easily.

Assotymologie: een wetenschap?

Eddy D.G. Boeve
CWI, Kruislaan 413, 1098 SJ Amsterdam

Het vertrek oogde als een omgeving waar wetenschap welig tiert: overal witte pilaren van opgestapelde publicaties, tijdschriften en hier en daar een NRC, opengeslagen bij het cryptogram. Ook de wetenschapper zelf ontbrak niet: aandachtig turend naar een enigszins vreemd uitgevallen kleuren-tv, een toetsenbord onder z'n ene hand, een smeulend iel peukje in de ander. Ik was hier om Prof. L. Meertens te interviewen in verband met zijn onlangs verschenen boek "Assotymologie: een nieuwe taalwetenschap", een nieuw boek in de taalkundige serie *'t Kofschip*.

– *Professor Meertens, nadat ik uw boek gelezen had, moest ik toegeven dat ik diep onder de indruk was van uw theorie, maar toch bleef ik ook nog met wat vragen zitten. Kunt u bijvoorbeeld uitleggen wat u bedoelde toen u uw nieuwe theorie een "witte vlek in de kaart van de hedendaagse taalwetenschap" noemde?*

"Eh, dat lijkt mij voor de hand liggend! Het woord *assotymologie* is, zoals u ongetwijfeld in mijn werk heeft gelezen, een samentrekking van de woorden *associeren* – het in verband brengen met – en *etymologie*, woordafleidkunde. Ik poneer dan ook een theorie waarin de oorsprong van woorden wordt opgespoord door het gebruik van associatieve elementen. Teveel laat men zich misleiden in de huidige etymologie door de eenvoud: te snel grijpt men naar een min of meer gelijkluidend Grieks, Latijns of Indo-germaans woord en daar stopt vervolgens het denkproces!"

– *U gebruikte in uw boek als voorbeeld het woord "vakantie", wat u herleidt, als ik het me goed herinner, via de begrippen "mieren" en "vakken". Toch kan ik me nog goed herinneren dat ik tijdens m'n Latijnse lessen verklaard kreeg dat vakantie afkomstig was van het woord "vacans", het tegenwoordig deelwoord van "vacare", leeg zijn, vrij zijn. Waarom bent u niet tevreden met een zo'n voor de hand liggende verklaring?*

"Dat is nou net de kern van m'n theorie: als men nu wat meer oog voor de verschillende talen, dialecten en hun samenhang had, gecombineerd met een basis gezond associatief gevoel, dan was men zeker niet gestopt bij de verklaring: komt uit het Latijn. Nu heb ik een redelijke kennis wat betreft de moderne talen, maar ook m'n kennis van vergeten talen en dialecten – zoals Creoolse beneden-rivierse dialecten, Syrillische schuiltalen, Tameense verzetskreten en Neolitaanse koortonen – geven mij de nodige dwarsverbanden om tot mijn assotymologische afleidingen te komen. Om terug te komen op "vakantie": nu komt dat woord eigenlijk niet voor in mijn vocabulaire, maar hopelijk kunt u begrijpen dat ik een woord als dit associeer met vlaktes en eenzaamheid: het ontbreken van de voor mijn onderzoek noodzakelijke bergen papier, stapels ongelezen publicaties en hopen zorgvuldig gemeden ambtelijke rompslomp. Nu, dat vind je exact terug in het woord "vakantie". Je moet dit zien als een samentrekking van "vak" en "ant", waarbij het laatste woord letterlijk is

overgenomen uit het engels en dus voor "mier" staat. Kunt u zich het beeld voor de ogen halen van een mier, nietig klein, zich hulpeloos voortbewegend over een schier eindeloze vlakte van een leeg vak, een schap? Precies zoals ik me zou voelen als ik mijn werkruimte zou moeten verlaten voor een vrij nutteloze bezigheid als vakantie vieren!"

– *Tja, zo had ik dat niet begrepen uit uw boek, maar ik moet toegeven: interessant idee. Trouwens, er waren wel meer stukken die voor een leek als ik wat moeilijk te volgen waren. Bijvoorbeeld de hoofdstukken vol wetenschappelijke symbolen, die u geloof ik "skwikkels" noemde. Kunt u uitleggen wat u voor ogen had toen u deze – naar ik vermeen vrij nieuwe tak van de wiskunde – betrok bij uw taaltheorie?*

"U moet niet vergeten dat taal niet de hoofdwetenschap is die ik beoefen, maar dat dat eigenlijk theoretische wiskunde is. Maar, zoals ik duidelijk maakte op pagina 134 van mijn boek: nooit en te nimmer mag men wiskunde – een exacte, formele wetenschappelijk taal – los zien van de natuurlijke gesproken en geschreven taal. Ik toon aan op pagina 657 dat veel, zo niet de meeste taalconstructies voldoen aan nauwkeurig en eenduidig te omschrijven algoritmen en zo ook eenvoudig zijn uit te schrijven in een door mij en Richard Bird ontwikkelt formalisme: squiggol. Nooit had iemand gedacht dat deze theorie – ook wel het *Bird-Meertens Formalisme* genoemd – simpel te mappen zou zijn op een gereduceerde eindige lijst van taalhomomorfismes. Zodoende ben ik in staat elke assotymologische afleiding van een willekeurig woord in een, voor zo ver ik het nu kan overzien, willekeurige taal, in alle gevallen in minder dan drie squiggols weer te geven. Misschien wel leuk om in het interview op te nemen dat ik gisternacht net het drieëntwintigduizendste squiggol-symbool heb ontwikkeld: een roos-vormig krabbeltje, dat ik ga gebruiken voor een nieuwe datastructuur in mijn taalnotatie: rozenbomen."

– *Ik hoop dat u het mij vergeeft als ik zeg dat die wiskunde mij ver boven mijn pet gaat, maar ik zou nog wel graag willen terug terugkomen op de afleiding van het woord "jubileum", die u bij de aan de lezer overgelaten assotymologische oefeningen aan het eind van hoofdstuk 56 noemt. Ik heb afgelopen weekend geprobeerd om tot een oplossing te komen in de aard van uw nieuwe theorie, maar ik kwam er niet uit. Telkens bleef het latijnse woord "iubileus" door mijn hoofd spoken en ik kwam niet verder met de door u gepropageerde associatie-theorie. Kunt u mij misschien wat op weg helpen?*

"Grappig dat u nou juist dit voorbeeld neemt. Het is een typisch voorbeeld van een afleiding die men geheel zonder enige buitenlandse taal kan oplossen. U geeft ook goed weer hoe moeilijk het is om een eenmaal aangeleerde ethymologiesche redenering te vervangen door een meer associatieve benadering. Maar toch is het in dit geval simpel: waar denkt u aan als u het woord *jubileum* hoort? Ikzelf moet meteen denken aan *feest*, *muziek* en *drank*, voor een persoon gearrangeerd die bijvoorbeeld een bepaalde tijd in dienst is geweest van een bedrijf of instelling. Het feest wordt gerepresenteerd door een zekere *jubel*stemming, uit de spreektaal gemakkelijk overgenomen in de schrijftaal als *jubil*. Het laatste gedeelte, *leum*, werd vroeger zeker niet uitgesproken als *le-um*, maar gewoon op dezelfde manier als *leut*, feestvreugde, en is hiermee dan ook verwant. U ziet, simpel doch elegant, en zonder een woordje Latijn of Tacratische berg-dialecten!"

– *Maar eh ..., is het niet wat te simpel om het woord "jubel" zomaar als basiswoord te introduceren? Kan het niet zijn dat dat toch weer uit het Latijn afkomstig is?*

"(Zucht) Ik zie dat het niet gemakkelijk is om mensen uit hun etymologische dwangbuis te trekken: het woord *jubel* is een simpele samentrekking van *ju* en *bel*. Nu, die jubilaris, die wordt weergegeven door *u* of *jij*, in dit geval samengetrokken tot *ju*, en de feestmuziek door de bel. Simpel, ja? Je kunt deze assotymologische transformatie in twee squiggols

weergeven: $\mp\uplus$, waarbij $\mp$ staat voor de feestende en dansende jubilaris, op weg naar zijn drankje: $\uplus$."

– *Goh, ik moet zeggen dat het met uw theorie in de meeste gevallen wel mogelijk is een verklaring weet te vinden! Wat is trouwens de reactie van andere deskundigen – op uw eigen vakgebied, of op taalgebied – op uw theorie? Ik las in een artikel over u in de Volkskrant dat taalwetenschappers twijfelden aan uw kennis van het Neolitisch, Tameens en het Tacratisch, terwijl wiskundigen in dezelfde krant ook enige kritische voetnoten plaatsten bij de toepasbaarheid van "skwikkel" op een zo levend iets als moderne talen. Wat denkt u van deze toch niet zo positieve kritieken?*

"Ha, ik had die recentie nog niet gelezen, ik lees namelijk een kwaliteitskrant, maar ik had zoiets wel verwacht. Wat die taalkundigen betreft: zij durven gewoon niet toe te geven dat ze zelfs nog nooit van die uitgestorven talen gehoord hebben! Hetzelfde geldt voor de wiskundigen: behalve Richard en een promovendus die ik begeleid, is er nog nooit een mathematicus verder dan hoofdstuk 2 – voor de goede orde: de opstap tot de eigenlijke theorie – van ons *Bird-Meertens Formalisme* boek gekomen. Ik vraag u dan ook: waar praten zij over?"

– *Nou, mij heeft u in ieder geval volledig overtuigd. Als u het mij permitteert, zou ik nog een keer mogen proberen om een assotymologische afleiding te vinden?*

"Natuurlijk!"

– *Eh, in de inleiding van uw lijvige werk vermeldt u dat assotymologie net zo goed toepasbaar is op eigennamen. Als ik bijvoorbeeld uw eigen naam, Lambert Meertens, neem. U roept bij mij het beeld op van een ambitieus wetenschapper, wat meteen een basis is voor het opsplitsen van uw achternaam: "Meer" en "tens", meer tienen dus. Nu nog "Lam" en "bert". "Bert" is een veelgebruikte Nederlandse naam, daarover geen woorden vuil gemaakt. Verder denk ik dat u niet vies bent een goed glas wijn of bier: ook "lam" mag nu duidelijk zijn!*

"Bravo, mevrouw! Ik hoor dat u de essentie van mijn theorie heeft begrepen en ik denk dat u veel plezier zult beleven aan het nieuwe boek wat ik op het moment aan het schrijven ben: "Werkwoorden: een vorm van structurele inductie op maximale initiële elementen".

– *Veel succes met uw nieuwe werk en tegen de tijd dat u gaat publiceren kom ik wel weer eens langs voor een interview ...*

Ik liet de wetenschapper alleen achter in zijn overvolle kamer, hij draaide alweer een nieuw flinterdun shaggie en was al weer helemaal geobsedeerd door zijn scherm en toetsenbord. Ik? Ik verliet duizelend het semi-moderne gebouw waar de professor werkte, mijn hoofd vol nieuwe ideeën.

Bois de Boulogne National Laboratory
Bois de Boulogne, IL, 92366578341
1990 December 21

To Whom It May Concern (if it concerns anyone):

It is with pleasure that I submit this letter of recommendation for Lambert G. L. T. P. D. Q. Meertens for the position of Deputy Chief High Muckety-Muck at the Center for Wishfulthinking Intheattic (CWI). In my estimation, Lambert will fill this position more than adequately.

Lambert has demonstrated talents for dealing with people, which would stand him in good stead should this be an administrative position. For example, in the IFIP WG2.1, even though he is not an officer, he generally manages to maneuver things so that he is the first presenter at a meeting. Moreover, he manages to present his material in a manner that causes the members to agree that he has a point of view.[1]

Of course, Lambert's scientific reputation is spotless.[2] At an early age, he began his career in wishfulthinking intheattic. He built a working computer when he was in high school.[3] After this early success, he moved onto wishfulthinking in the area of fictional programming. One day, he was sitting under an apple tree, and a little Bird dropped a Feather on him. This accident gave Lambert the brilliant idea that there was a Partschal analogy between fictional programming and chemistry. Taking a Paige from Fred Brooks, he used this insight to incorporate the operations of filtration and reduction (to whose discussion I alluded above) into his fictional notation.[4]

It should be evident that Lambert does not suffer from the "not-invented-here" syndrome.[5] Ideas adopted (and adapted) from fields as disparate as architecture and the aforementioned catalog theory Pepper his work. >From a Dutchman in the former, he took the philosophy that "less is more", which has lead him to nvnt n xtrmly cncis notat'n; from the latter he took the idea that arguments should be

---

[1]This success is all the more remarkable for two reasons. First, the members rarely agree on anything. Second, Lambert's presentations are always about completely incomprehensible topics, which usually seem to be unrelated to the purpose of the group (insofar as it is known). As I remember, his presentations have included such topics as business (a discussion of catalog theory) and chemistry (a discussion of operations such as filter, reduce, etc.)

[2]There are a few stains on it, but these are tobacco stains and should not be counted against him.

[3]It is perhaps unfortunate that at this time his parents had no garage. Had he done these Jobs in a garage, instead of intheattic, he might have made millions by founding the nExt aPPLE company.

[4]Unfortunately, Wile these operations are often used, another important chemical operation, Boyling, has not yet been incorporated.

[5]I should mention, however, that as far as cigarettes are concerned, I have never known him to adopt any of the well-made commercial varieties; he always rolls his (in my view, somewhat scruffy) own.

suppressed, which has made him very unpopular with his colleagues.

Lambert has outstanding abilities in languages. He reads Dutch, English, French, Latin, Algol, and Squigol (in two dialects). He is something of a Latin scholar. I well remember the time that a number of us were trying to determine the longest Latin word. Well aware of his Latin scholarship, I asked Lambert's opinion. Lambert answered, "The longest Latin word is *circumspectabamini*." Others greeted this unexpected candidate with derision; "*Circumspectabamini*, what does that mean?" they asked, thinking he had made it up. "Having been looked around at," Lambert replied.[6]

Finally, Lambert is well known as an editor. He edited the long-delayed proceedings, **IFIP WG2.1 Working Conference on Programme Specifications and Transformations**. In addition, he edits the **PQR Newsletter**, which enjoys wide circulation because he sends it to his colleagues free of charge.[7]

In addition to all of these sterling qualities, Lambert has an excellent sense of humor, which makes him easy to work with. Some of his jokes are of near award-quality[8], and Lambert can always be counted on to be entertaining.

All in all, Lambert is an excellent fellow, and bids fair to continue in his present position if not to assume a new one. I recommend him highly!

Sincerely,

James M. Boyle
Computer Scientistque

---

[6]I really must point out, however, that this longest word can be made longer by the addition of the suffix *-que*. It occurs in the original form of the famous statement attributed to Caesar: *Veni, vidi, vici.* Caesar's actual utterance, "*Veni, circumspectabaminique, vici,*" though less alliterative, is historically more accurate, meaning, "I came, and, having been looked around at, I conquered."

[7]Once in a while, however, the editing here is not as tight as one might like. A recent copy of the Newsletter contained an ad offering "life-sized posters of the planets"!

[8]However, the Pulitzer committee is still attempting to understand his most famous humorous (apparently) utterance: the question, "What is the difference between a duck?", and its answer, "One leg is both the same".

Klippe og knippen

til Lambert

Det danske verbum *at klippe* svarer i betydning til det hollandske *knippen*. Desuden ligner ordene hinanden. Kunne de være beslægtet? For at svare dette vil jeg undersøge for det første om disse ord har tilfredsstillende etymologier, og for det andet om der foreligger andre eksempler på en afvekslen af kl- og kn-.

1. At klippe. Nielsen [Dansk etymologisk ordbog] siger om verbet *klippe*: 'af usikker oprindelse', og henviser til det gammel norske *klippa, klyppa*. Wessén [Våra Ord] siger om verbet *klippa*: 'uteslutande nord. ord, trol. ljudhärmande'. Selvfølgelig er det engelse ækvivalent *to clip*, og Partridge [Origins] siger 'prob echoic'. Altså, ordet er velkendt i de skandinaviske sprog, og lånt til engelsk, men ellers ukendt.

Der er et andet ord *klippe*, navneordet, som svarer til det hollandske *klip*, engelsk *cliff*. (Muligvis beslægtet med *kleven, cleave, glue* og med *klimmen, climb, klifa*.)

2. Knippen. Hvad det hollandske *knippen* angår, det ser ud til at komme fra *knijpen*, ligesom *nippen* stammer fra *nijpen*, og n- og kn- afveksler hinanden. (Andre eksempler er *(k)nagen, (k)nikken, (k)noest, (k)nop*, o.s.v. Denne afvekslen skyldes vist forekomsten af rødder hn-. Mærkelig nok afleder de fleste etymologiske ordbøger disse ord fra helt forskellige rødder: de siger fx *nikken, nijgen* < germ. *hniqwan, og *knikken* < germ. *kneik- o.s.v.) Vi har *knijf*, engelsk *knife*, dansk *kniv*. Roden forekommer også udenfor det germanske.

3. kn- versus kl-. Det er ikke lykkedes mig at finde mange overbevisende eksempler på en afvekslen mellem kn- og kl-. Man har på dansk *knebel* for hollandsk *klepel*. Man har på hollandsk *knuppel*, tidligere også *kluppel*, engelsk *club*, tidligere *clubba*, gammel norsk *klumba*. I begge tilfælde er kl- oprindelig og kn- skyldes sandsynligvis dissimilation. Hvis *klippe* og *knippen* er beslægtede, så er kn- vel oprindelig, og kl- er umotiveret. Måske er *klippe* opstået som kontamination med gammel norsk *klypa* (sml. gammel norsk *kljufa*, hollandsk *klieven*, engelsk *cleave*).

Andries Brouwer - 901220:0050

On the Name of a Programming Language

Robert B. K. Dewar

On occasions such as this, one is apparently expected to provide some kind of humourous piece. Ignoring for the moment the well-known fact that the author of this paper does not have a sense of humour, it seemed in any case important to take this opportunity to correct a highly unfortunate situation which has recently come to the author's intention.

As those of you who are familiar with Lambert Meertens style of work know well, he often assumes extensive knowledge on the part of his audience and unfortunately this assumption is often incorrect. The current situation appears to be a particularly egregious case of this phenomenon, and I am pleased to have the chance to correct it.

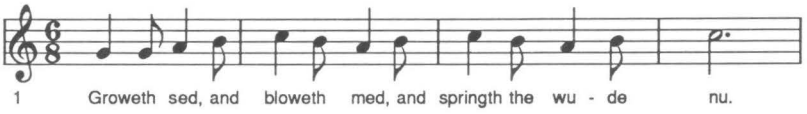
Lambert is of course, like most people who dabble in computer science for light entertainment, first and foremost a musician. Thus it is all the more surprising that so few people have noticed the obvious connection between the field of western music and the name Lambert chose for his recent programming language, ABC.

These three letters represent of course one of the most important underlying themes in Western tonal music, at least since the time of the discovery of equal temperament, since it was not until this discovery that the perfect symmetry of frequency in this sequence of notes A-B-C is achieved, namely that the frequency ratio for the full tone interval A-B is exactly the square of the ratio for the half step B-C.

A full treatise on the importance of the A-B-C theme throughout the recent history of music would take far more space than is reasonably available, especially in these days of concern for the number of trees being chopped down to print technical material, even in cases like this, where the importance of the material is unquestioned. Still the author will attempt to outline in exceedingly brief form, the introduction, development and frequent reprises of this theme.

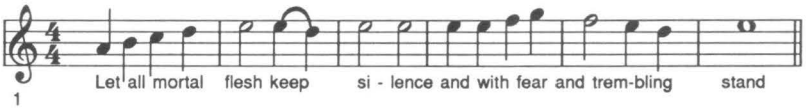
Before the discovery of equal temperament, when attempts to execute musical interpretations on instruments such as Rackets, Krumhorns and Sackbuts yielded at best dubious results, the falling sequence C-B-A was much more in favor. An important milestone in the development of western music occurred with the introduction of the round *The Cuckoo Song* in the mid thirteenth century. This piece is often recognized as the first introduction of polyphony, but much more important is its suggestion for the adoption of the A-B-C in its non-inverted form. The opening phrase is conventional enough, and renders the phrase in its, at that time, familiar, C-B-A form. In this and other musical examples, we will transpose to A minor so that the rising theme is immediately familiar, and we also take some liberties with the original time signatures, again for the purposes of clarity of analysis.

The opening notes are thus the exact inversion of the theme under consideration, but, at the very end, subtly masked by what to the contemporary ear must have sounded like a veritable cacophony of uncoordinated solo performances, we hear the rising phrase for the first time in the last three notes:

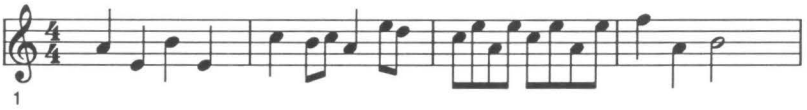


The invention of the round form provided another masterful touch, in that it permitted the author to repeat the suggestion over and over again until the audience had fully absorbed the subtle unconscious message.

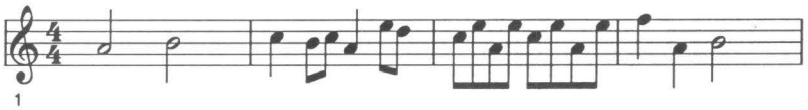
It was several hundred years before either polyphony or the theme in question resurfaced, but the implantation in the collective unconscious eventually bore fruit One of the earliest uses of the theme in its full form appears to be in the 17th century traditional French melody *Picardy* which survives today as hymn number 197 in the 1940 edition of the Episcopal Hymnal:



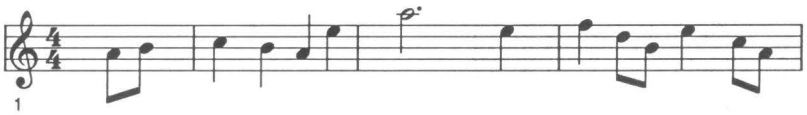
In more formal music, one of the first appearances is in one of Bach's earlier organ fugues, although it is not immediately apparent at first glance:



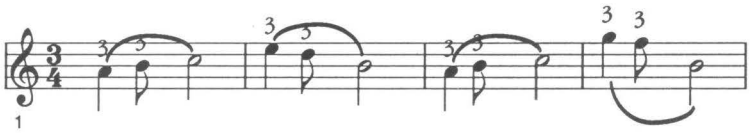
Here the theme is slightly disguised, and the informed musicologist of today may wonder whether Bach intended the theme to appear in its unadorned form:



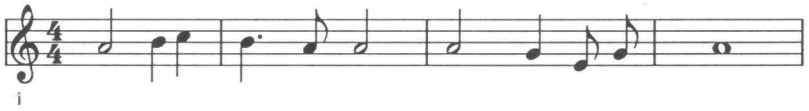
Perhaps Bach did not conciously intend the theme to appear directly at this time, but by the later period of his works, we find the theme, appearing without any disguise in triumphant and undisguised form in the harpsichord concerto:



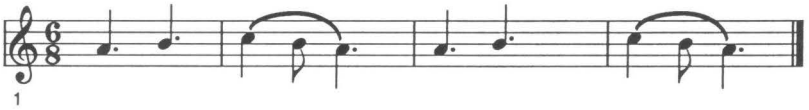
From the point of this introduction on, the theme was used to introduce a wide variety of works. Among many examples we find the the third movement of the Brahms *Third Symphony*:



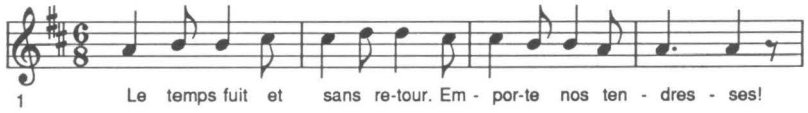
In Dvorak's *New World Symphony* we find the theme announced by the brass section with a tone of outright defiance:



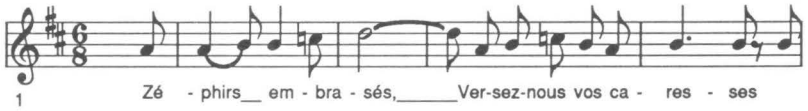
As a final example, Mahler pays tribute not only to the A-B-C theme, but to the round form which introduced it six centuries earlier, in the slow movement of the *First Symphony*:



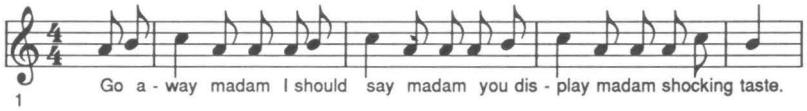
By the end of the nineteenth century, the theme had become so firmly established in the unconcious expectations of audiences that clever composers could create unbearable tension from its absence. Consider for example, Giulietta's phrase as she joins Nicklas in the Barcarolle of Offenbach's *Les Contes D'Hoffmann*:



The appearance of the C sharp, creating the surprising juxtaposition of two full tones instead of the expected tone/half-tone pattern cries out for resolution. What a relief when nearly a page later we finally hear her sing the expected C natural:

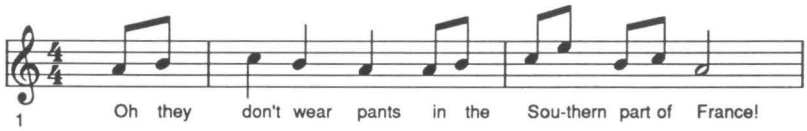


Little wonder then that many serious composers of the time started to feel imprisoned by this theme. Can there be any doubt that Sir Arthur Sullivan, in choosing the following theme for the Lord Chancellor's plea to the Fairy Queen in *Iolanthe*:



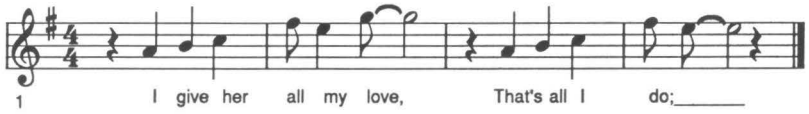
is addressing the sentiments to the by-that-time almost unbearable ubiquity of the A-B-C theme as being itself in shocking taste?

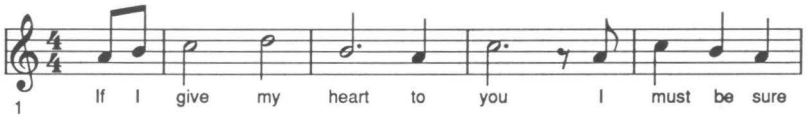
As is so often the case, important themes wander from the world of serious music to more popular arenas. Surely one of the most notorious early popular uses of the A-B-C theme occurs in the once familiar song:



The authorship of this song is unclear, although an interesting claim is one made by a U.S. congressman who later helped to found the U.N. What is clear is that this song was used to accompany Little Egypt for her "performances" at the Columbian Exposition in Chicago.

In more recent times, we would all recognize familiar themes from two of the best loved Beatle's songs:





As the theme has become popularized, serious composers have managed largely to escape its influence by the introduction of atonal and serial music. Though sometimes not fully understood, it is of course clear that the introduction of these forms was primarily motivated by the need to purge music of the odious rising 2-1 frequency ratio. This culminated in Cage's *Seven Minutes of Silence*, whose significance is of course not simply that there is no music for seven minutes, but no appearance whatsoever, either in direct, inverted, transposed or otherwise disguised form, of the A-B-C theme.

This is of course but a bare sketch of the importance of the A-B-C theme in the development of western music, and I must apologize to those, including of course Lambert, to whom these facts are completely familiar. Given the expectation that the new programming language ABC will have as profound an effect on the development of civilization over the next five hundred years as music has had in the previous five hundred years, it is almost inevitable that Lambert found himself forced to choose this name. Nevertheless, it is clear that there are a significant number of people who were not aware of this obvious fact, and instead thought that the name was derived from some naive analogy with simple teaching methods. I hope that this article will prevent this sad confusion in the future.

Over kraaiepoten

The lion is known by his claw
J. Bernoulli

'Biep.'

Het is 11:55. De luncheon daemon verschijnt op het scherm en meldt dat het tijd is om ons los te weken uit ons kot. Tijd om een kopje soep en een broodje kroket te gaan nuttigen. Samen met de andere AA-ers begeven we ons naar de CWI kantine. Tijdens het eten worden behalve kroketten ook andere onderwerpen aangesneden.

'De 68000 denkt.. met een ultra high-speed MMU waardoor.. de ROM geRAMd.. en toen ging de object-oriented knowledge engine stuk... maar gelukkig hadden de migrerende nonblocking amoebes... dus het hele zaakje weer gereboot op de Turing machine met Kolmogorov online,' braakt men uit aan de andere kant van de tafel.

'Eh, eh,' brengt Steven in, 'You are me one guilder guilty,' verwijzend naar de gulden regel.

'Goh, vannacht heb ik me toch keigoeie band gezien, man,' keelt Jack.

Ook Paul V. schuift aan voor zijn ontbijt.

Al snel wordt een servetje rondgestuurd waarop je moest aangeven of en wat voor koffie je wilde. Door middel van het fameuze koffieturfalgoritme kon je dan legio voorkeuren kenbaar maken.

Aan deze kant wordt een interessante programmeeropgave uitgelegd. Ik luister aandachtig. Eens kijken of ik deze opgave snel kan oplossen. Lambert luistert ook met een half oor. Om uit te vinden of ik alle gegevens goed begrepen heb —en om tijd te winnen—, herhaal ik de vraagstelling nog een keer en verzink in diep gepeins...

'Oh, ' zegt Lambert tussen twee happen smeerworst door.

'Ja, da's simpel,' knikt hij nogmaals, alles even snel narekenend en de smeerworst doorslikkend.

Wat? Ik moest de vraagstelling nog verwerken en hij had de oplossing al gevonden. Zomaar. Zonder een vinger te gebruiken. Uit zijn blote hoofd. Hoe deed ie dat? Beschikte hij over hogere machten?

Zoals zo vaak tijdens de lunch legde Lambert daarna geduldig zijn oplossing uit. Servetje na servetje werd volgeklad met zwarte streepjes en rondjes. En aan het einde van het middagmaal lag de tafel bezaaid met kladjes, allemaal vol met afdrukken van kraaiepootjes.

Pas later drong het tot me door over welke krachten Lambert beschikte. Hij had een machtig wapen ontwikkelt: de kraaiepootcalculus. Daarmee deed hij wonderbaarlijke dingen. Hij kon oplossingen voor problemen geven in slechts een paar regels. Een kant van een servetje was vaak voldoende. En als je de regels goed kende dan deed je het uit het blote hoofd. Een paar keer knipperen met je ogen en een keer slikken was voldoende. Die kraaiepootcalculus is inmiddels een populaire calculus geworden. Het 'Bird-Meertens' formalisme heet het nu deftig.

Toen ik weer aan het bovenstaande gebeuren terugdacht, moest ik ook denken aan Bernoulli. (Ik weet niet meer welke Bernoulli het was. Ik geloof dat het Jacob I was, maar

dat weet ik niet meer zeker.) In de tijd dat Newton en Leibniz nog leefden bestond er een grote controverse over wie de grootste wiskundige was: Newton of Leibniz? Beiden hadden ze een calculus ontwikkelt waarmee ze met het grootste gemak vele wiskundige problemen konden oplossen. Om uit te vinden wie nu werkelijk het grootste genie was, had Bernoulli twee moeilijke problemen ontworpen en hun publiekelijk uitgedaagd. Newton had al snel de oplossingen gevonden. Hij stuurde ze op naar Bernoulli, echter niet onder zijn naam, maar anoniem. Toen Bernoulli de oplossingen ontving uitte hij bovenstaande spreuk.

Lamberts kraaiepootcalculus is natuurlijk moeilijk te vergelijken met Newtons leeuwepootcalculus. Maar ja, het zijn ook maar kraaiepootjes.

Vlak voor mijn promotie is Lambert een keer bij ons komen eten. Na enkele flessen wijn eindigde de avond met een fles jenever. Omdat Lambert in mijn promotiecommissie zat, legde ik hem mijn stellingen voor. Ik dacht dat het een goed moment was om ze even op Lambert uit te testen. Stelling voor stelling werd aangevallen. De verdediging kreeg het zwaar te verduren. Uiteindelijk was het de drank die het laatste verzet deed bezwijken. Terwijl Lambert rustig de N-de borrel achterover sloeg, voor $N \to \infty$, moest ik opgegeven. Hoe zou dat gaan op de promotie?

Toen de dag aanbrak gedroeg Lambert zich voorbeeldig. Hij begon met een zeer lastige vraag. Voordat hij nog een vraag kon stellen, kwam de pedel weer binnen.

'De tijd is verstreken.'

(Dr.) Jo Ebergen
Waterloo, Dec. 1990

$$P \overset{\circ}{\circledast} Q = \downarrow / f * Q$$

$$f b = \uparrow / (\circledast b) \dot{*} P$$

$$(x \circledast y) \downarrow (y \circledast z) \sqsubseteq x \circledast z$$

Exploiting Associativity

Maarten M Fokkinga

It is well-known that for the standard implementation of cons-lists, as in Miranda and Lisp, the reduction to canonical form of $x{+\!\!+}y$ ("x cat y") takes size x steps, given that x and y are in canonical form. This is because the cat operation for cons-list is defined by induction on the structure of its left argument as a repeated cons. So, although $(x{+\!\!+}y){+\!\!+}z = x{+\!\!+}(y{+\!\!+}z)$, reduction of the left-hand side takes $\mathsf{size}\,x + \mathsf{size}\,(x{+\!\!+}y) = 2 \times \mathsf{size}\,x + \mathsf{size}\,y$ steps, but only $\mathsf{size}\,x + \mathsf{size}\,y$ steps for the right-hand side. It is therefore more efficient to compute the value of an expression $x = x_1{+\!\!+}x_2{+\!\!+}\ldots{+\!\!+}x_n$ (with some parenthezation) by evaluating $x' = x_1{+\!\!+}(x_2{+\!\!+}(\ldots{+\!\!+}x_n))$ instead (with the parentheses grouped to the right). Less than four years ago I gave a formal treatment of this phenomenom in a seven page note ("Elimination of Left-nesting: an example of the style of functional programming"). Nowadays, in the current status of Constructive Algorithmics and Squiggol Notation, developed by Lambert Meertens and others, it is hardly more than a simple exam question; see Theorem (2.3) below. As a show of what I have learned from Lambert —and maybe of what I still have to learn— I will discuss a generalisation of the elimination of left-nesting in full.

Preliminaries

Notation Function application is denoted by a low dot, $f.a.b = (f.a).b$. Function composition is denoted $\odot$, but when both arguments are present we also write $f \cdot g$ where the dot $\cdot$ has *lowest* binding strength and will *not* be used as an argument of another operation or function. For functions f, g and arbitrary $a : A$ we define

$$\begin{array}{lcl} f \Updownarrow g.\ (x,y) & = & (f.x, g.y)\,, \\ f \triangle g.\ x & = & (f.x, g.x)\,, \\ a^{\mathsf{U}}.\ x & = & a\,. \end{array}$$

For $\oplus : A \Updownarrow B \to C$ and $f : A \to (B \to C)$ we define

$$\begin{array}{lcll} a \oplus b & = & \oplus.(a,b)\,, & \text{i.e., conventional infix notation} \\ \hat{\oplus}.a.b & = & \oplus.(a,b)\,, & \text{i.e., } \hat{\oplus} : A \to (B \to C) \ \text{ (currying)} \\ \check{f}.(a,b) & = & f.a.b\,, & \text{i.e., } \check{f} : A \Updownarrow B \to C \ \text{ (uncurrying)} \\ a\oplus & = & \oplus \cdot a^{\circ} \triangle \mathsf{id}\,, & \text{so that } a\oplus.\,b = a\oplus b \ \text{ (left section)} \\ \oplus b & = & \oplus \cdot \mathsf{id} \triangle b^{\mathsf{U}}\,, & \text{so that } \oplus b.\,a = a\oplus b \ \text{ (right section)} \end{array}$$

The properties of this notation will be referred to as "calculus". Using the notation we may express associativity of $\oplus$, and 'being the identity of $\oplus$', in a variable free form:

$$\begin{array}{lcl} \mathit{assoc}(\oplus) & \equiv & \hat{\oplus} \cdot \oplus \ = \ \odot \cdot \hat{\oplus} \Updownarrow \hat{\oplus}\,, \\ \mathit{unit}(e, \oplus) & \equiv & \oplus e \ = \ \oplus \cdot \mathsf{id} \triangle e^{\mathsf{U}} \ = \ \mathsf{id} \ = \ \oplus \cdot e^{\mathsf{U}} \triangle \mathsf{id} \ = \ e\oplus\,. \end{array}$$

The reader may check $assoc(\oplus)$ by applying the functions to $((x,y),z)$.

Structures Let A be a type. The data type of *structures over* A is given by
type: $A*$
operations: $\tau : A \to A*$, $\mathbin{+\!\!+} : A* \times A* \to A*$, $\boxdot : A*$
laws: $unit(\boxdot, \mathbin{+\!\!+}), \ldots$.
(τ yields a "singleton" structure, $\mathbin{+\!\!+}$ "cat" two structures, and $\boxdot$ is the "empty" structure.) With only the law $unit(\boxdot, \mathbin{+\!\!+})$ the structures are called *trees*. If in addition $assoc(\mathbin{+\!\!+})$ is postulated, then the structures are called *cat-lists*. We shall use postfix symbol $_{\blacktriangle}$ and $_{\triangleright}$ to distinguish between trees and lists; if no subscript is used we mean generically either of them.

For $f : A \to B$ and $\oplus : B \times B \to B$ the function $(\!|f, \oplus|\!) : A* \to B$, called *catamorphism*, is defined to be the unique function satisfying

$$\begin{aligned}
(\!|f,\oplus|\!) \cdot \tau &= f \\
(\!|f,\oplus|\!) \cdot \mathbin{+\!\!+} &= \oplus \cdot (\!|f,\oplus|\!) \times (\!|f,\oplus|\!) \\
(\!|f,\oplus|\!) . \boxdot &= \text{identity of } \oplus .
\end{aligned}$$

(We assume that any binary operation has an identity.) Thus the effect of $(\!|f,\oplus|\!)$ might be described as the systematic replacement $\tau, \mathbin{+\!\!+}, \boxdot \ := \ f, \oplus, \text{unit}(\oplus)$. It is easy to see that $assoc(\mathbin{+\!\!+})$ implies associativity of $\oplus$ on the range of $(\!|f,\oplus|\!)$, so that $(\!|f,\oplus|\!)_{\triangleright}$ does not make sense when $\oplus$ is not associative on its range. Specific functions are

$$\begin{array}{lllll}
\oplus/ & = & (\!|\text{id}, \oplus|\!) : A* \to A & \text{for } \oplus : A \times A \to A & \text{reduce} \\
\overleftarrow{\oplus} & = & (\!|\breve{\oplus}, \odot|\!)^{\smile} : A* \times B \to B & \text{for } \oplus : A \times B \to B & \text{right-reduce.}
\end{array}$$

For example, for $x = (\tau.a \mathbin{+\!\!+} \tau.b) \mathbin{+\!\!+} \tau.c$ we have $\oplus/.x = (a \oplus b) \oplus c$ with the same parenthezation as in x, and $x\overleftarrow{\oplus} = (a\oplus \cdot b\oplus) \cdot c\oplus$ so that $x\,\overleftarrow{\oplus}\,d = a \oplus (b \oplus (c \oplus d))$ with the parentheses grouped to the right! Part of the observation of the introduction amounts to the assertion that for any tree x we have $\mathbin{+\!\!+}_{\triangleright}/.x = x\,\overleftarrow{\mathbin{+\!\!+}_{\triangleright}}\,\boxdot_{\triangleright}$. It is this claim, as well as a generalisation, that we shall prove below. In the proof we use two important properties of catamorphisms:

$$\begin{array}{lllll}
f \cdot (\!|g, \oplus|\!) & = & (\!|f \cdot g, \ \circledast|\!) & \text{if} \quad f \cdot \oplus = \circledast \cdot f \times f & \text{Promotion} \\
f \cdot \overleftarrow{\oplus} & = & \overleftarrow{\circledast} \cdot \text{id} \times f & \text{if} \quad f \cdot \oplus = \circledast \cdot \text{id} \times f & \text{r-reduce Prom.}
\end{array}$$

The reader may prove these properties by induction on the structure of the argument; (actually they are a simple consequence of the definition of the notion of data type — which we have not given here).

Proving the claims

(1) Lemma *For associative* $\oplus$ *we have* $\breve{\oplus} \cdot (\!|f, \oplus|\!) = (\!|\breve{\oplus} \cdot f, \ \odot|\!)$.

The proof is trivial indeed:

$$\overset{\circ}{\oplus} \cdot (\!| f, \oplus \!|) = (\!| \overset{\circ}{\oplus} \cdot f, \ \circledcirc \!|)$$

$\Leftarrow$ Promotion

$$\overset{\circ}{\oplus} \cdot \oplus = \circledcirc \cdot \overset{\circ}{\oplus} \Updownarrow \overset{\circ}{\oplus}$$

$\equiv$ $assoc(\oplus)$

true .

(2) Theorem *Suppose* $assoc(\oplus)$ *and* $unit(e, \oplus)$. *Then*

1. $(\!| f, \oplus \!| = \overline{(\overset{\circ}{\oplus} \cdot f)\, e}$
2. $\oplus /^{\sqsupset} = \overset{\circ}{\oplus}\, e$
3. $\mathbb{H}_{\triangleright} /^{\sqsupset} = \overline{\mathbb{H}_{\triangleright}\, \boxed{\circ}_{\triangleright}}$

To prove Part 1 we argue

$$(\!| f, \oplus \!| = \overline{(\overset{\circ}{\oplus} \cdot f) e}$$

$\equiv$ lhs: $unit(e, \oplus)$; rhs: unfold and calculus

$$\oplus \cdot \mathrm{id} \vartriangle e^{\mathbb{H}} \cdot (\!| f, \oplus \!| = (\!| \overset{\circ}{\oplus} \cdot f, \ \circledcirc \!|^{\smile} \cdot \mathrm{id} \vartriangle e^{\mathbb{H}}$$

$\Leftarrow$ Leibniz (and calculus)

$$\overset{\circ}{\oplus} \cdot (\!| f, \oplus \!| = (\!| \overset{\circ}{\oplus} \cdot f, \ \circledcirc \!|$$

$\equiv$ preceding lemma, $assoc(\oplus)$

true .

Part 2 follows from Part 1 by substituting $f := \mathrm{id}$. Part 3 is an instantiation of Part 2.

* * *

In general, when $assoc(\oplus)$ holds and $\oplus /^{\sqsupset}$ is to be computed, one might wish to restructure the parenthezation of the argument by a specific transformation ε and then evaluate $\oplus /^{\sqsupset} \cdot \varepsilon$. To this end define the *listifying* function

$$lfy \quad = \quad (\!| \tau_{\triangleright}, \mathbb{H}_{\triangleright} \!| \quad : A* \to A*_{\triangleright} .$$

Then $lfy.x$ is the list of tip-values of x in "left to right" order, irrespective of the parenthezation within x ; and $lfy.x = x$ for any list x .

(3) Theorem *Suppose* $lfy \cdot \varepsilon = lfy$ *and* $assoc(\oplus)$. *Then* $(\!| f, \oplus \!| \cdot \varepsilon = (\!| f, \oplus \!|$.

To prove this we first observe that $(\!| f, \oplus \!|_{\triangleright}$ makes sense since $\oplus$ is associative, and that

$$(\!| f, \oplus \!|_{\triangleright} \cdot lfy = (\!| f, \oplus \!|$$

$\Leftarrow$ unfold lfy, Promotion

$$(\!| f, \oplus \!|_{\triangleright} \cdot \tau_{\triangleright} = f \quad \text{and} \quad (\!| f, \oplus \!|_{\triangleright} \cdot \mathbb{H}_{\triangleright} = \oplus \cdot (\!| f, \oplus \!|_{\triangleright} \Updownarrow (\!| f, \oplus \!|_{\triangleright}$$

$\equiv$ definition catamorphism

true .

Now we calculate

$$
\begin{array}{ll}
 & \mathfrak{C} f, \oplus \mathfrak{D} \cdot \varepsilon \\
= & \quad \text{above observation (right to left)} \\
 & \mathfrak{C} f, \oplus \mathfrak{D}_{\triangleright} \cdot \mathit{lfy} \cdot \varepsilon \\
= & \quad \text{premiss} \\
 & \mathfrak{C} f, \oplus \mathfrak{D}_{\triangleright} \cdot \mathit{lfy} \\
= & \quad \text{above observation again} \\
 & \mathfrak{C} f, \oplus \mathfrak{D}
\end{array}
$$

as desired.

The particular parenthezation enforced by a right-reduce is given by

$$\varepsilon_r \quad = \quad \overline{(\mathfrak{C}_{\blacktriangle} \cdot \tau_{\blacktriangle})} \cdot \mathrm{id} \triangle \boxtimes^{\mathfrak{g}}_{\blacktriangle} \quad = \quad \overline{(\mathfrak{C}_{\blacktriangle} \cdot \tau_{\blacktriangle})} \boxtimes_{\blacktriangle} \quad : \quad A* \to A*_{\blacktriangle} \; .$$

This claim is formalized and proved in Part 2 of the following theorem. Part 3 shows that ε_r satisfies the condition of Theorem (3), thus enabling an alternative proof of Theorem (2.2).

(4) Theorem

1. $\mathfrak{C} f, \oplus \mathfrak{D} \cdot \varepsilon_r = \overline{(\hat{\oplus} \cdot f)} e$ if $\mathit{unit}(e, \oplus)$ (not necessarily $\mathit{assoc}(\oplus)$!).
2. $\oplus /^{\triangleright} \cdot \varepsilon_r \quad = \quad \overline{\oplus} e$ if $\mathit{unit}(e, \oplus)$.
3. ε_r satisfies $\mathit{lfy} \cdot \varepsilon_r \quad = \quad \mathit{lfy}$.

To prove the first part we argue

$$
\begin{array}{ll}
 & \mathfrak{C} f, \oplus \mathfrak{D} \cdot \varepsilon_r \quad = \quad \overline{(\hat{\oplus} \cdot f)} e \\
\equiv & \quad \text{calculus} \\
 & \mathfrak{C} f, \oplus \mathfrak{D} \cdot \varepsilon_r \quad = \quad \overline{(\hat{\oplus} \cdot f)} \cdot \mathrm{id} \Updownarrow \mathrm{id} \cdot \mathrm{id} \triangle e^{\mathfrak{g}} \\
\equiv & \quad \text{equation for catamorphism on } \boxtimes \text{ using } \mathit{unit}(e, \oplus) \\
 & \mathfrak{C} f, \oplus \mathfrak{D} \cdot \varepsilon_r \quad = \quad \overline{(\hat{\oplus} \cdot f)} \cdot \mathrm{id} \Updownarrow \mathfrak{C} f, \oplus \mathfrak{D} \cdot \mathrm{id} \triangle \boxtimes^{\mathfrak{g}} \\
\Leftarrow & \quad \text{unfold } \varepsilon_r \text{, and Promotion for right-reduce} \\
 & \mathfrak{C} f, \oplus \mathfrak{D} \cdot \mathfrak{C} \cdot \tau \quad = \quad \hat{\oplus} \cdot f \cdot \mathrm{id} \Updownarrow \mathfrak{C} f, \oplus \mathfrak{D} \\
\equiv & \quad \text{equations for catamorphism on } \mathfrak{C} \text{ and } \tau \\
 & \quad \text{true} .
\end{array}
$$

Instantiating Part 1 with $f := \mathrm{id}$ gives Part 2. For Part 3 we calculate

$$
\begin{array}{ll}
 & \mathit{lfy} \cdot \varepsilon_r \\
= & \quad \text{Part 1 with } f, \oplus, e := \tau_{\triangleright}, \mathfrak{C}_{\triangleright}, \boxtimes_{\triangleright} \\
 & \overline{(\mathfrak{C}_{\triangleright} \cdot \tau_{\triangleright})} \boxtimes \\
= & \quad \text{Theorem (2.1) noting } \mathit{assoc}(\mathfrak{C}_{\triangleright}) \text{ and } \mathit{unit}(\boxtimes, \mathfrak{C}) \\
 & \mathit{lfy}
\end{array}
$$

as desired. ∎

stichting MATHEMATISCH CENTRUM 2e Boerhaavestraat 49 Amster

enveloppe voor rekenopdrachten

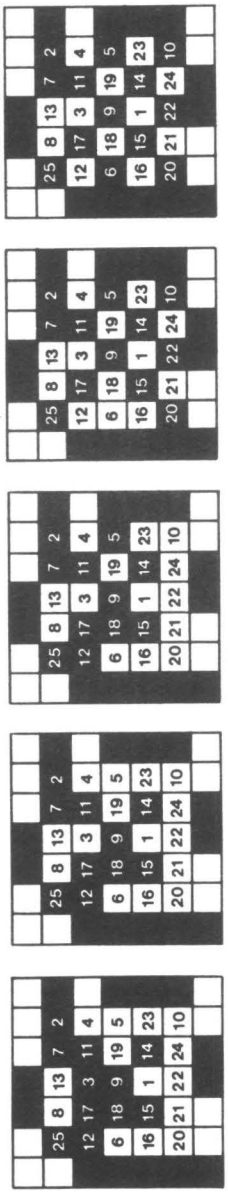
Opdrachtnummer: R 1229

Signalement:

Vakgebied:

Onderwerp: hogere orde approximaties van Nederlands

Prof. Turlings op landb
Europese landbou dig by
(Van onse korrespondent
Waninge, dinsdag – Op d
Waninge wordt gehou, het
aan die landbouhoeskool
pessimisties uit oor d
West-Europa. Op gron van
sonder die Amerikaanse
van die oorproduksie. D
in die landbou met pkt
persen per jaar vermeerd
Sprekende oor die konse
seide prof. Turlings, wa
'n wysiging van die sam
dit nie geskiedt, mag w
in konflik. kom met des
Die hooglerar was van
van 'n l nende afset. D
landbougion in Nederlan
boubeleid sal men oog m
moelikhede. Dit sou wel
marginale bedrywen braa
Spreker bepleitte dit g




```
AVPH    ( 36) VERB + NONM' + 1 + 2 /
        ( 54) VERB + WONM' + 1 + 2 ∧ TOPH /
        ( 71) VERB + NOWM' + 1 + 2 ∧ MVPH /
        ( 80) VERB + WOWM' + 1 + 2 ∧ TOPH ∧ MVPH /
        (100) VERB + TOBE + 1 + 2 ∧ NDVG + 2
MVPH    (100) to ∧ TOPH
NDVG    ( 50) ADIE /
        (100) NPUV + 1
NOUN    ( 40) ADIE ∧ NOUN + 1 + 2 /
        (100) SUBS + 1 + 2
NPPL    ( 40) NOUN + PLUR + 1 /
        (100) the ∧ NOUN + PLUR + 1 /
```

Voor Lambert

mei

op een stoel
naast de kerseboom in bloei
formuleerde ik
specificeerde ik
programmeerde ik
een vogel
een heldere lucht
een oranje vlinder

op een stoel
naast de kerseboom in bloei
bracht ik leven
op het scherm
en hoe
mijn vogel zong
mijn vlinder vloog
de kersebloesems geurden

op een stoel
naast de kerseboom in bloei
zag ik plotseling
de dood
mijn vogel kraakte
mijn vlinder explodeerde
de kersebloesems geurden
toen het GEB de stroom afsloot

Jan Heering

Elegie

In deze kille, duist're gang
beid ik, al zo vrees'lijk lang
kreunend onder een immens gewicht
tot mijn lijden wordt verlicht.

Deemoedig sta ik hier en wacht
bedolven onder mijn loodzware vracht
kon iemand toch mijn klagen horen
lang leven is mij niet beschoren.

Collegae aan mijn beide zijden
weten niet van mijn zwaar lijden
zij staan daar maar, zo voor de vaak
genietend, zonder zware taak.

Onbekommerd staan zij op hun post
en worden altoos snel verlost
de eer, hun toegebracht, is welgepast
maar ik, o arme, bezwijk schier onder mijne last.

Lieve Lambert, hoor naar mijn geschrei
ik ben nu toch werk'lijk de wanhoop nabij
kijk naar mij om, wil aandacht geven
zodat ook je postbakje verlicht kan leven.

Marja Hegt
(met hulp van Loes)

Het Jubileum
—een terugblik als vooruitblik—

Johan Jeuring

Het jubileum zit er op. Het viel ons mee, we dachten dat de hindernis zwaarder zou zijn. Er waren momenten waarop wij Lambert duidelijk ongeveinsde tranen van ontroering weg zagen pinken.

's Ochtends om negen uur leek Lambert danig ontstemd toen hem verteld werd dat de geplande vergadering van het A-team niet doorging. Zoals genoegzaam bekend, is vergaderen Lamberts voornaamste bezigheid. Hoewel hij dit zelf altijd tegenspreekt, lijkt hij vergaderen ook een aangename bezigheid te vinden. Als roken is toegestaan en er koffie aanwezig is, kun je hem zien zitten genieten van deze lichamelijke 'geneugten', het af en toe plaatsen van ter zake doende opmerkingen en het vervaardigen van bijzonder fraaie kunstwerken op toevallig in de buurt ronslingerende blaadjes papier. Wij vertelden hem dat we een dag ter ere van zijn 25-jarig jubileum georganiseerd hadden. Daar reageerde hij op met 'Wat een verrassing, dat had ik niet verwacht!', hetgeen ons deed twijfelen aan zijn toch niet geringe, dachten wij, intelligentie en zijn deduktievermogen. Minstens vijf keer kwam hij binnen lopen terwijl wij deze dag aan het voorbereiden waren en twee keer is hem gevraagd naar antieke ponskaarten en andere curiosa die nergens anders dan voor goedbedoelde jubileumtoespraken gebruikt kunnen worden. Voor het 40-jarig jubileum verdient het aanbeveling de bovenstaande imbeciele zin te vervangen door het ook door Bomans aangeraden: 'Dat hadden jullie niet moeten doen'. Deze zin moet bewogen worden uitgesproken, in een toon van zacht verwijt. Ook goed is: 'Dat was nu wérkelijk niet nodig geweest,' maar het eerste is beter, omdat het laatste reeds algemeen wordt ingezien.

Voor het eerst in haar bestaan was geheel AA reeds om negen uur op het CWI aanwezig om gezamenlijk met ontbijt de dag te beginnen. Het ontbijt werd begeleid door een strijkkwartet, dat lustig het door Lamberts computerprogramma gecomponeerde muziekstuk 'Strijkkwartet no. 1 in C' streek. De aanwezigheid van het musicerende strijkkwartet verhinderde de normale gespreksvoering en gaf alle medewerkers de gelegenheid te wennen aan de het-was-toch-een-mooie-tijd en je-hebt-je-kranig-geweerd sfeer die zo goed bij een jubileum past. Het inhuren van het strijkkwartet bleek een hele verstandige zet te zijn, en wij kunnen alle organisatoren van jubilea aanraden de, uiteraard onnatuurlijke, sfeer op een dergelijke soepele en aangename wijze op te bouwen.

Tot de lunch stond er verder niets op het programma. Vrolijk keuvelend, op een toon van wederzijdse waardering en zelfs genegenheid, als een grote familie, liep AA terug naar de derde etage. Schrijver dezes merkte op dat bij sommigen, met name bij de jubilaris zelf, het goede humeur gedeeltelijk veroorzaakt leek te worden door het vooruitzicht toch nog twee uurtjes te mogen werken. Aan dezen zou hij de volgende, ernstige, woorden toe willen

voegen.

Het vergeefse streven van Prediker

Ik, Prediker, was koning over Israël te Jeruzalem; en ik zette mijn hart er op na te vorsen en onderzoek te doen naar de wijsheid in alles, wat onder den hemel geschiedt. Dat is een kwade bezigheid, die God aan de mensenkinderen gegeven heeft om zich daarmee te kwellen. Ik nam in ogenschouw alle daden, die onder de zon verricht worden, en zie: alles is ijdelheid en najagen van wind! Het kromme kan niet recht zijn en het ontbrekende kan niet geteld worden. Ik zeide bij mijzelf: Zie, ik ben groter en rijker in wijsheid geworden dan allen die vóór mij over Jeruzalem geregeerd hebben, en mijn hart heeft in overvloed wijsheid en kennis opgedaan; zo heb ik er mijn hart op gezet om wijsheid en kennis, verdwaasdheid en onverstand te leren kennen. Ik heb ingezien, dat ook dit is najagen van wind. Want in veel wijsheid ligt veel verdriet, en als iemand kennis vermeerdert, vermeerdert hij smart. *Prediker 1:12-18*

De lunch, samen met de directeuren, de sprekers van het middagprogramma en de feestcommissie, was eenvoudig en gezellig. Na afloop van de lunch sprak de directeur Lambert toe. Deze horde, misschien wel de moeilijkste van het jubileum, werd zowel door de direkteur als Lambert zonder veel moeite genomen. De direkteur stond voor de moeilijke taak een aantal overduidelijke leugens overtuigend op te sommen en Lambert had het zo mogelijk nog moeilijker met het bepalen van zijn gezichtsuitdrukking bij het aanhoren van deze ijdelheid. Een glimlach zou hebben betekend dat hij de spreker niet geloofde. Daar dit inderdaad het geval was, diende deze gesteldheid verborgen te blijven. Ernst daarentegen had te kennen gegeven dat hij alles voor zoete koek slikte, en dit was nog erger geweest, want een kind wist dat het wartaal was. Dus hield Lambert de blik omlaag gericht, strak op de vloer. Bij bepaalde passages, waar de spreker in zijn vuur teveel lof toezwaaide, schudde hij twijfelend het hoofd.

De sprekers van het middagprogramma, niet gehandicapt door de noodzaak een serieuze toespraak te houden, amuseerden het bij tientallen toegestroomde publiek in hoge mate. De eerste spreker, Richard Bird uit Oxford, betrad het podium met onder zijn arm een klein vrieskistje. Hierin bleek, tot ons aller verbazing, een wel heel bijzonder geschenk voor Lambert te zitten: een getal dat bij blootstelling aan kamertemperatuur onmiddelijk zou ontbinden. Wij waren enigszins teleurgesteld dat wij dit getal niet van dichtbij in ogenschouw mochten nemen. Kees Koster uit Nijmegen, de tweede spreker, haalde herinneringen op uit de tijd dat hij en Lambert nog programmeerden met behulp van draden in plaats van een toetsenbord. Het pluggen van concrete draden is vervangen door het leggen van abstracte pijlen, aldus Koster, die hiermee een interessante lijn in Lamberts werk blootlegde. De laatste spreker, Michel Sintzoff uit Leuven, gaf een verhandeling over geesten. Naast heilige en boze geesten onderscheidde hij formele geesten en klassificeerde Lambert hieronder. De wel zeer lijvige aanwezigheid van Lambert ontkrachtte zijn stelling enigermate, maar dit bezwaar wimpelde hij met aanstekelijke, uitbundige lachsalvos resoluut weg.

Op deze drie voordrachten volgde de uitreiking van het liber amicorum aan Lambert Meertens door Johan Jeuring. In een korte, bondige en vlammende toespraak wist Jeuring vele punten op evenzovele i's te zetten. Leugens werden ontkracht en nieuwe waarheden aan het licht gebracht. Zo sprak hij onder andere over Meertens' levensfilosofie, de zogenaamde consistentiologie. Deze filosofie behelst het vatten in regels van alle aspecten van de dagelijkse werkelijkheid. Een eenvoudig en gechargeerd voorbeeld is de volgende regel: als Peter van Buren een film lovend bespreekt in de Volkskrant, hoedt u dan voor deze film en mijdt liefst zelfs de bioscoop waarin zij draait. Echter, indien van Buren met afschuw schrijft over een film, tracht deze dan zo spoedig mogelijk te zien, daar u anders met grote waarschijnlijkheid een film met hoge amusementswaarde zou missen. Zoals reeds gezegd, dit is slechts een voorbeeld; een uitvoerige behandeling van de consistentiologie kunt u in de memoires van Lambert ('How rules governed my life', te verschijnen herfst 1991) terug vinden. Lambert was verguld met het liber amicorum.

Tijdens het copieuze diner haalden de sprekers en de jubilaris herinneringen op uit ver vervlogen tijden, tijden, waarin solderen nog in de bagage van een programmeur zat en waarin chips nog schaars waren. Zoals altijd blijkt bij het ophalen van herinneringen, scheen het solderen een aangename bezigheid te zijn geweest waar zij met dierbare herinneringen op terugzagen. Na deze intieme ontboezemingen vond uw verslaggever het niet kies verder te luisteren. Het diner eindigde zoals het begon, in een rustige, familiaire sfeer.

Ter afsluiting van de feestdag was er een feest georganiseerd in het Joseph Lam Jazzcafé. Het feest werd geopend met het zingen van het Squiggol-lied op de wijs van 'Vader Jakob'. De tekst van dit lied luidt:

left reduce map
horner's rule
twiddle right reduce spot
zip spot spot

Na dit modern en sfeervol lied stond de avond, wat muziek betreft, in het teken van de zestiger jaren. Een luidruchtige gelegenheidsband van het CWI, bestaande uit voornamelijk blazers en gitaristen, deed verwoede pogingen het zaaltje op te blazen, zowel met behulp van de geproduceerde enorme hoeveelheid decibellen als met hun enthousiasme. Dat hun pogingen mislukten was slechts te danken aan de 300 jaar ervaring van de spanten, binten en steunberen in het dragen van feesten en partijen. En het feest, het feest duurde voort ...

Two Boys and a Computer

C.H.A. Koster
Vakgroep Informatica
Kath. Universiteit Nijmegen

December 21, 1990

For Lambert Meertens,
Il miglior fabbro ...

Would this hour never end? Father Adriaanse's voice was droning on and on as he tried to get Class 5 Gym of the Ignatiuscollege interested in the question: What is the Task of Laity in Church.

One boy was not paying attention. Surreptitiously his hand had drifted under the table to grasp the angular form of his latest acquisition: a 3-digit decimal counting relay, purchased at the flea market for 50 cents.

A curious device, it had a window through which three digits could be seen, and a tumbler operated by an electromagnetic coil. Each time the tumbler was pressed, the number shown advanced by one, returning at last to zero to resume its cycle. The boy was now operating it by hand instead of by electricity, silently pressing the tumbler, observing the absolute precision of the camwheels, the unswerving regularity of the numerical cycle.

His neighbour in the bank, while feigning attention to the Theology lesson, had moved closer and closer, and now stretched out his left hand, to grasp the wonder with his own hands. Softly clicked the mechanism, while two boys felt the excitement of a two- or even three-place carry, or tried to make the counter advance only half a step.

"'Ere, what's that?" Blacktogaed Father Adriaanse stood towering over them, looking down the wart on his nose, extending an imperious hand to grasp the artifact and bring it closer to his eyes, turning it this way and that, studying it, judging it not sinful but detracting, and finally dropping it into one grimy pocket of his soutane.

"I'll keep that for you!" And turning around he loudly resumed: "Let me put the question differently: What is the Task of Laity in Church?"

Two boys, after exchanging one more glance, lapsed into their own thoughts, hiding behind the back of the boys in front of them, waiting for the hour to end.

Singlemindedly, on behalf of the Jesuit Order, Father Adriaanse was doing all he could to restore orthodoxy to a class become unruly through the radical teachings and Socratean questioning of his predecessor, Father van Kilsdonk.

It must have been around March 1960. Lambert Meertens had drifted into my attention after the classes of the Gymnasium were reshuffled in the fifth year, to separate those with a mathematical leaning from those who preferred the humaniora. We had a heavy load of courses, including Latin and Greek and four modern languages, as well as Mathematics, Physics, Chemistry, Biology and lesser subjects such as History, Geography and — Theology.

Lambert was the youngest of our class, having skipped two years in elementary school. He was small and highly introvert, with dark, curly hair and thick and large spectacles. He was quite bad at sports and only a little better than others at Languages, but he soon established himself as the best mathematician in class, against a stiff competition: Henk Pijls, Cor van de Bosch and I, we soon learned to value his sharpness of mind. At this stage, the class was no longer dominated by bullies, so we could quite simply find pleasure in Learning.

Two boys went to the Efficiency Fair that spring, to see the computers. I don't remember whose idea it was: Lambert's, because of his mathematical interest or mine, because of my interest in electronics, but I am sure the counting relay had to do with it.

We sauntered past the gaudy exhibits of Efficiency, which seemed to have to do with storing vast amounts of colourful paper forms. We learned to recognize the man with Manager Appeal, the greyhaired allknowing expert with the hornrimmed glasses which he moves cyclically through three positions: on his nose, tapping against his teeth and stretched like a rapier in the direction of his listeners — in fact, Professor Frielink. We watched endless streams of punched cards moving through complicated readers, punchers, sorters and hoppers.

But most of all we were attracted by the novelty of that years show: The possibility to put questions to a remote computer. The machine itself stood in far away Leidschendam and was connected by teletype. It purported to be able to factorize large numbers. The visitors to the fair were doing their best to invent large numbers, and marveled at the speed of computation.

Lambert suddenly said to me:"I know what we should try", and when our turn came he typed in: 0. The teletype started to rattle

```
2 x 2 x 2 x 2 x 2 x 2 x 2 x ...
```

A boyish grin came over Lambert's face: This was going to continue forever! After two-and-a-half lines the teletype hesitated, then laboriously completed the sequence

```
... 2 x 2 x x 0
```

Some human being had interrupted the stupid behaviour of the program.

This was our first encounter with Willem Louis van der Poel, who thereupon invited us to Leidschendam, to the Computer laboratory of the PTT, where we saw the Stantec ZEBRA and learned to program it in Simple Code. But that is another story.

It was during our visit to the Fair that we decided to build our own computer.

The two boys met again in the attic of the office of Lambert's father, who was himself a highly successful efficiency expert. The stuffy cabinet was crammed with all the paraphernalia of efficiency: Folders, ledgers, hangers in Folio and A4 format, planning boards covered with multicoloured plastic strips, staplers and punches, weirdly shaped paperclips, cards in racks and stacks, with multicoloured riders suitable for systematizing knowledge. It was a highly stimulating environment for our brainstorming.

Real computers were based on the binary numbering system. We drew tables of addition and multiplication for the binary digits and pondered. In order to make a computer, we needed a bistable device, like the BIC CLIC ballpoint. Obviously, a mechanical device was much too clumsy, could we find an electrical analogon? At that point I thought of a 12-stable device available in shady dumpshops: a 12-step switching relay from a telephone exchange. By connecting together alternate banks, we could turn it into a bistable device, with an extra make/break relay thrown in for free, suitable for dealing with the carry.

While Lambert sat down with a stub of pencil, in order to design a binary adder, I set out to screen all dumpshops in downtown Amsterdam. By incredible good luck I managed to find ten 12-step relays, so ten bits our computer was to have. When I came back, Lambert had invented the ones-complement adder with end-around-carry. Within twenty four hours I had constructed a rapid prototype: two bits, with two switches as input and two small lamps as output. We proudly showed it to our Physics teacher, but his reaction was less than encouraging. He much preferred us to do Physics, instead.

In retrospect I do not blame him for being unable to extrapolate from our rickety contraption to the potential electronic brain that we saw in it.

It was Lambert who made the first contact with the Mathematical Center, in the summer of 1960, through Chris Harmse, a greyhaired sternfaced mathematician and programmer, who took great interest in us and in our work, and introduced us to the ARMAC.

The ARMAC was an electronic computer, built at the Mathematical Center in the late fifties. It took a room by itself, with an imposing facade of pleasantly glowing electronic tubes, a Creed fast paper tape reader (at something like 30 characters per second), a Creed fast paper tape punch at

15 cps and an electric typewriter typing some 12 characters per second. It had a rotating drum for 4000 words or instructions of 35 bits each, and even a newfangled device for storing 32 words of fast memory: a rough mat of ferromagnetic rings with interwoven wires.

The ARMAC was the province of Joop van Loenen, the wizard maintenance engineer, who kept the machine in such good order that it sometimes ran for an hour without breaking down. Van Loenen knew the whole machine by heart, he was reputed to be capable of stopping the machine in mid cycle, replacing some tube and restarting it within the same cycle.

It was Chris Harmse who presented us with the full user documentation of the ARMAC, such as it was: one typewritten page listing all 32 instructions. Our eyes grew bigger in our heads: now this was a real computer!

Immediately we started to work on a program called Test Interplanetary Person. The program put the user through a gruelling sequence of quizzes and speed tests, ending with simulated rocket landing, all in order to decide whether the user came from some other planet or was a mere human being. It produced some spectacular sound effects, making use of the fact that the sign-bit of the A-register was wired to a loudspeaker.

After demonstrating it to Professor van Wijngaarden, we were allowed one quarter of an hour per week on the flagship of the MC: the brand new Electrologica X1.

In the Fall of 1960 we completed the design and construction of our computer and gave it a name: NEBAC 1101. The acronym was ambiguous, standing either for Nederland's Eerste Binaire Amateur Computer (the Netherland's first binary amateur computer) or for Nep Electronische Binaire Amateur Computer (the fake electronic binary amateur computer). 1101, the binary number 13, looked quite pretty to us.

One of the few documents from this period which we conserved shows an artist's impression of the way it was supposed to look after its completion. In reality it was a table full of intriguing contraptions, connected by bundles of wire.

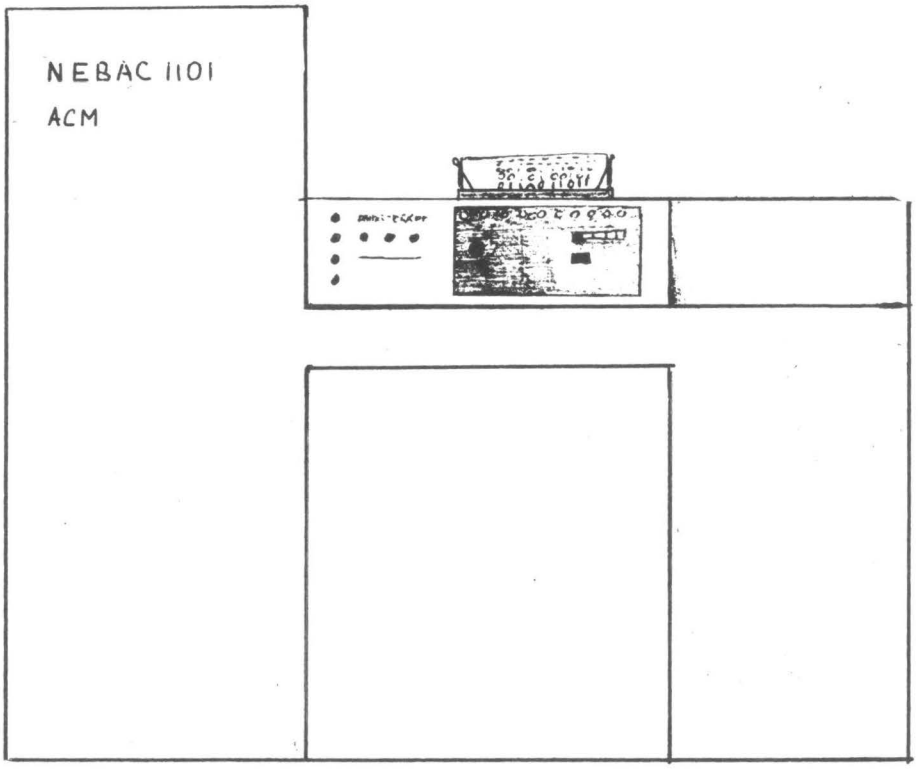
The NEBAC 1101 consisted of a number of devices: a ten-bit accumulator, a control unit, a number of memory cells, input keys, output lights and a power pack. The accumulator had three instructions:

- (half-add) adding one bit in some position
- (half-subtract) subtracting one bit
- (invert) inverting the sign of the accumulator by inverting all bits.

Especially this last instruction was spectacularly noisy.

The control unit had space for 35 instructions, each 15 bits wide, and was programmed by inserting small banana plugs into a board. The control

COMPUTER NEBAC



would cycle through its 35 instructions, the completion of one instruction causing the switch-over to the next. Each instruction consisted of up to 5 command contacts (providing an impulse to the selected device) and a 10 bit connection to one of the memory cells or input keys (either copying information into the accumulator, which had to be cleared beforehand, or copying information from the accumulator to a memory cell, which then had to be cleared beforehand). We had also designed a subroutine mechanism, the control passing temporarily to a simpler version of the control unit, which served to compose an addition out of half-additions and a multiplication out of (shifted) additions. These latter devices were however never built.

There were two hard problems: the construction of the clock pulse generator and the power supply. How should you feed dozens of hungry relays with 12 Volt and generate a solid 2Hz. clock pulse, especially if you are penniless? For both of these problems I found solutions at the Mathematical Center.

The Mathematical Center was housed in one half of a decrepit old school building, within easy smelling distance of the Amstel Brewery. Its three-and-a-half floors were teeming with life, mathematicians doing both pure and impure mathematics, technicians designing and building the new X1 computer, as well as a diminishing number of computors — shy girls and ladies who could work wonders with calculators and spreadsheets. The presence of the latter was a Godsend to the mathematicians, who would otherwise never even have met a woman in their lives. The whole beehive was kept together by the firm hand of Mrs. Beppie Reckman.

The MC was a happy place, full of strange types. There were for instance the Dijkstra Twins, E.W. Dijkstra and J. Zonneveld, who were at that time implementing the world's first ALGOL 60 compiler. Stern, thoughtful Dijkstra and voluble Zonneveld, it was a marvel that two such disparate people could work together.

I arrived one day at the center, to find Dijkstra hanging out of a second floor window, reeling in an enormous paper tape from a heap on the sidewalk, where Zonneveld was busy disentangling the many tape loops, swearing loudly and inventively. That was my first sight of a compiler.

Lambert and I of course attended the first Algol 60 course, given by Dijkstra, and soon were reading and correcting all the ALGOL 60 programs that we could find in the waste basket.

One of the technicians took time off to explain, on the back of an envelope, the working of the Thyratron, and after buying one in a second hand shop I constructed a dependable 2Hz. oscillator. Joop van Loenen presented us with an enormous Selenium Rectifier. left over from the ARRA computer, with the help of which I could finally build a reliable power supply, stabilized by means of a 12 Volt car battery.

It was a pleasure to set the NEBAC 1101 counting through its binary numbers for hours and even nights. While I was diligently building the control unit, Lambert was writing the first programs for it. But somewhere along the line, after more than a thousand soldering joints, my interest in completing the computer started to flag.

While programming the ARMAC and the X1 I discovered that I could achieve more with paper, pencil and a rubber than by soldering connections, and resolutely hung my soldering iron in the willows.

Abstract computers turned out to be much more interesting than concrete ones.

Rather unexpectedly in the Spring of 1961 the two boys were summoned to the Director of the Mathematical Center, and asked to bring the blueprints of the NEBAC 1101. At the appointed hour we took the lift up to the third floor of the MC, where Professor van Wijngaarden held office like the Olympian Jove. We found in a little room a committee of three: Aad van Wijngaarden was seconded by Edsger W. Dijkstra on the one side and C. Scholten, one of the designers of the ARMAC and X1, on his other side. Dijkstra was unusually quiet, guardedly considering our every word, whereas Scholten obviously was of the opinion that two schoolboys could not build a computer without his counseling. He was questioning us closely, while van Wijngaarden was gleefully trying to poke holes in our story. In the end we seemed to have passed their examination: we obviously knew what we were doing.

It was then that Edsger Dijkstra turned around to a pigeonhole cabinet containing reprints of MC publications, took out two copies of a two-page pamphlet, scribbled something on them with his fountain pen and gave us one each, without saying a word. I looked at my copy. It was an article by a certain E.W. Dijkstra, concerned with the determination of primality of large numbers by means of sequences and modulo arithmetic. On it he had written (in rather archaic Dutch): "Van den schrijver". My heart jumped up in my breast, and beaming we left the room: We had been accepted as Disciples by Real Science.

From then on, the world was open to us. We passed our final school examinations, and decided about our futures. Rather than becoming an engineer in Delft, as I had intended for many years, I chose to stay in Amsterdam with Lambert and study Mathematics and Physics. We had a glorious time at the Municipal University of Amsterdam, doing precisely as we liked. We followed lectures by E. W. Beth, F. Staal, A. Heyting, A. van Wijngaarden, E.W. Dijkstra, F.E.J. Kruseman Aretz, A. Perlis, A.D. de Groot, N. Frijda. At one occasion we even heard and saw L.E.J. Brouwer.

Meanwhile, at the Mathematical Center and shortly joined by Leo Geurts, the Third Man in our troika, we occupied ourselves with extracurricular subjects like Computer Linguistics (Affix Grammars, analysis and translation of Natural Languages), Computer generated Speech (our aim was for the X1 to pronounce "Yess, Prroffessorrr" as a surprise for Aad van Wijngaarden; it only got as far as saying "stoep" and "Informatie"), Computer Music (Lambert winning a prize at the 1968 IFIP Congress with his computer-composed string quartet; the X1 computer singing loudly to itself, while the population of the MC was attending some solemn jubilation meeting), Artificial Intelligence (from Pavlov simulation and Lambert's General Problem Solver to a quite serious involvement in LSD-based research into Schizophrenia), Computer Graphics (one of the first applications of the finite elements method in the Netherlands), Computer Art (the Computer Art Society Holland, the opera Reconstruction, Lambert and Leo's silk prints) and whatever else caught our attention.

Lambert Meertens and I had the great luck to be introduced to Computer Science during its heroic period, when every problem led to a novel solution and nothing appeared impossible.

We were present at the cradle of Informatics. We have rubbed shoulders with the heroes of the first hour and saw the present generation grow up.

The world was young, the sixties of this century were just starting, and we have enjoyed them to the full.

$Κυβισμος\ Τετραγωνων$*

$Πλατων$
$Τμημα\ Φιλοσοφιας$
$Ακαδημια\ Αθηνων$
$Αθηνα$

$Πυθαγορας$
$Τμημα\ Αριθμων\ και\ Μουσικης$
$Πανεπιστημιον\ Συρακουσων$
$Σικελια$

$Πλατων$: It is not very often I get to see you my friend. But you always seem to report important discoveries to me. So what is new this time?

$Πυθαγορας$: Well, you know all these crazy problems you've been studying in your Academy on trisecting an angle and squaring the circle.

$Πλατων$: What is wrong with that?

$Πυθαγορας$: Nothing in particular, but my flight to $Αθηνα$ was very long. I had to stop in $Αλεξανδρεια$ and on the way there I thought of an interesting related problem.

$Πλατων$: Why don't you share it with me!

$Πυθαγορας$: Can you cube a square by adding k units to it?

$Πλατων$: Sure! For $k = 0$, 1 is both a square and a cube.

$Πυθαγορας$: Come on now $Πλατωνα$ I mean something less trivial than that! At least assume k is not 0!

$Πλατων$: Well, I have to think then. Mmmm... I don't seem to find any solutions for $k = 1$.

$Πυθαγορας$: Precisely! You can't cube a square by adding a unit to it! Here is the proof. I wrote it on this $Παπυρο$. You can study it with your students.

$Πλατων$: Very interesting. How about adding two units to a square!

$Πυθαγορας$: Good. Now you are asking a good question.

$Πλατων$: Oooops... I think I found a solution. It is 25. It is a square and if you add two units it becomes 27 which is a cube.

$Πυθαγορας$: Great $Πλατωνα$! I know you are very smart. Actually on the $Παπυρο$ I gave you I wrote on the margin that 25 is a "magic" number since it is also the only square which becomes a cube when you add two units to it. I have a simple proof of it but didn't fit on the margin.

$Πλατων$: Well then what't happening for k greater than two?

$Πυθαγορας$: Frankly, I don't know. I visited $Απολλωνειο$ while in $Αλεξανδρεια$ and asked hime the question.

$Πλατων$: I think this guy is crazy. I won't trust what he says.

$Πυθαγορας$: Yes I know. He told me that the problem is related to conic sections and elliptic curves and that I have no hope of solving it. I don't believe him though.

---

*This is one of $Πλατων$'s lost dialogues and was accidentally discovered by Evangelos Kranakis, while drinking a belgian beer, in the $παπυρος$ wrapping of a herring he bought at the Kuiperstraat markt, in Amsterdam. It is reprinted with the kind permission of the $Ακαδημια\ Αθηνων$, exactly as found, with only a single editorial note at the end for the benefit of CWIers.

$\Pi\lambda\alpha\tau\omega\nu$: I don't either.

$\Pi\upsilon\theta\alpha\gamma o\rho\alpha\varsigma$: I also went to $\Delta\epsilon\lambda\phi o\iota$ to ask $\Pi\upsilon\theta\epsilon\iota\alpha$.

$\Pi\lambda\alpha\tau\omega\nu$: What did she say?

$\Pi\upsilon\theta\alpha\gamma o\rho\alpha\varsigma$: Hard to make sense of what she said. You know she takes too many drugs! She said something like: "this problem can only be solved by the barbarians from the north."[1]

[1]editor's note: $\Pi\upsilon\theta\epsilon\iota\alpha$ was right. It was Fermat, a Gaul, who first suggested that the number 25 is is the only square which becomes a cube by adding 2; and another Gaul, Pepin, who gave the first rigorous proof. You see, $\Pi\upsilon\theta\alpha\gamma o\rho\alpha\varsigma$ was wrong to claim he had a simple proof that didn't fit on the margin of his $\Pi\alpha\pi\upsilon\rho o$. In fact it was Saxons (Mordell), Vikings (Thue), Huns (Brauer), and Russians (Uspensky) who subsequently studied the problem very thoroughly. But all that is by now history. After all the authors seem to establish the magic properties of the number 25 beyond any doubt.

De dialektiek van een ABC verslaafde

Timo Krijnen

> Daarmee zijn we bij de klassieke vorm der predikatenantinomie aangekomen: de inconsequentie die consequent is, is geen consequente inconsequentie (want de inconsequentie zelve wordt door het beginsel der inconsequentie uitgesloten), daarentegen is de inconsequente inconsequentie juist een consequente inconsequentie. Wij doen derhalve in zoverre het voorstel om het principe der consequentie in zijn volle waarde te laten, als wij het principe der inconsequentie inconsequent in praktijk brengen.
> Zover dan de lof der inconsequentie. De rest laat zich niet onder woorden brengen. De rest moet men doen.
>
> Leszek Kolakowski, Lof der inconsequentie.

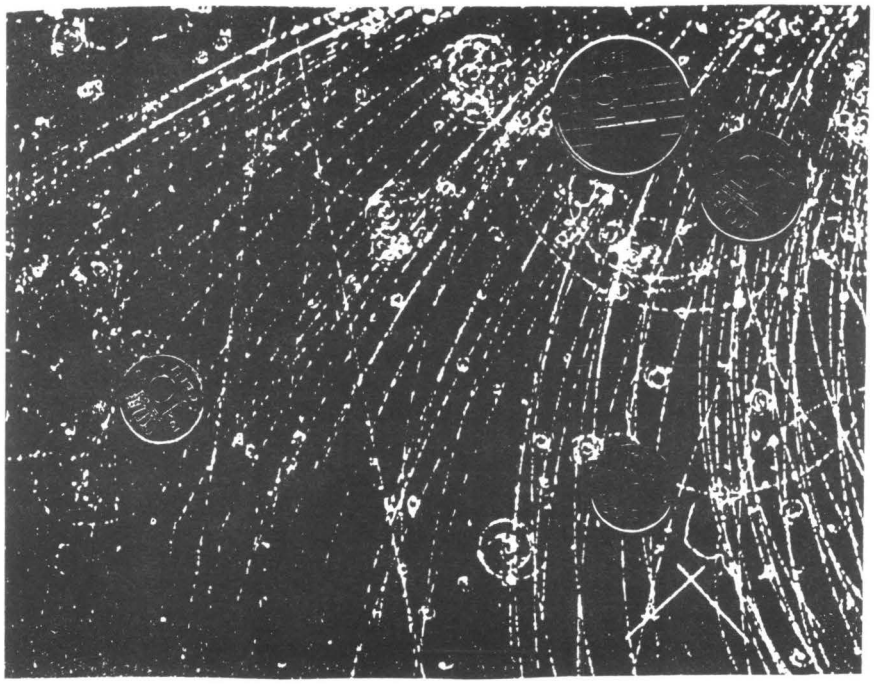
Mij was altijd geleerd dat de wiskunde slechts een hulpwetenschap is. Daarom koos ik waarschijnlijk ook een "echt vak" als scheikunde toen ik aan de universiteit ging studeren. Zolang ik er als hulpmiddel mee om leerde gaan, haalde ik ook netjes tienen voor zaken als algebra, analyse en statistiek. Tegen de tijd dat ik op het Mathematisch Centrum kwam werken, was mijn visie op de wiskunde echter al veranderd door mijn kennismaking met de dialektiek in de diskussies in de studentenbeweging en tijdens mijn bijvak Natuurfilosofie.

Voor de dialektiek is de formele logica de logica van het ogenblik, van de bevestiging van het objekt, dat geisoleerd en in zijn isolement wordt vastgehouden. In de werkelijkheid veranderen de dingen echter voortdurend in hun tegendeel. Om het worden en de aktiviteit uit te drukken hebben we de dialektiese logica nodig. Daarbij vernietigt de dialektiek die statische bevestigingen niet, maar overstijgt ze. Ofwel ook voor de dialektiek is de wiskunde een hulpmiddel bij het greep krijgen op de werkelijkheid. De dialektiek kan echter niet als een abstacte vorm, een taal, ontwikkeld worden; dat was de fout van Hegel. Haar kategorieën moeten worden ontwikkeld uit de worsteling met de feiten, met het gehele werkelijke en historiese materiaal, zoals Marx voor een tak van wetenschap, de politieke ekonomie, in Das Kapital heeft laten zien.

Toch heb ik de afgelopen tien jaar van mijn werkzame leven goeddeels besteed aan het meehelpen ontwikkelen en prakties bruikbaar maken van een formele taal. Lange tijd heb ik daarbij niet goed begrepen wat van mij als programmeur werd verwacht. Bovendien legde het slaafse onderhoudswerk aan het ABC systeem zoveel beslag op onze tijd dat we aan de uitvoering van al die mooie nieuwe plannen voor schermbespeling, schoonmaakalgoritmiek, en grafische tekenarij niet meer toekwamen.

Het gaat mij dan ook aan mijn hart, dat nu ik denk mijn taak als wetenschappelijk programeur beter te begrijpen, en het onderhoudswerk op zijn einde loopt zodat we weer bij het ontwikkelen van de nieuwe gezichtspunten (Views) kunnen worden betrokken, de valbijl boven ons hoofd is strak gespannen.

Mijn langste ABC programma staat bekend onder de naam "koekalender". Misschien is het een troost te weten dat er minstens twee boeren wel bij varen.



The Existence of the Dubbletje: A New Fiscal Particle

Bon Larfield

NIKHEF, Amsterdam
5 December 1990

Key Words: fiscal particles, grand monetary unification, loose change, the ecu, 'Big Bank' theory.

Abstract

The current system of fiscal particles in the Netherlands is the result of many years of research and experiment, and fiscal particles such as the guilder, the stuiver and the quartje are well known and well documented. However, within the past few years there seems to be a growing body of experimental evidence that points to the existence of a new fiscal particle. The size of this fiscal particle is several orders of magnitude smaller than that of the more well known particles, and it is because of this that it has gone undetected for so long. This paper reviews fiscal particle observations in the Netherlands, draws together some of the new evidence and attempts to define the nature of the new fiscal particle. It then goes on to examine how detection of the new particle supports evidence from the UK of a cyclic, non-steady state model of the fiscal universe.

1 The Netherlands Fiscal Model

For many years now the fundamental Dutch fiscal particle has been the Stuiver [Kei77], [Sla72], [Sla75]. The value of the stuiver has thus been adopted as the standard unit of fiscal value. Fiscal values of other particles are measured against that of the Stuiver. For example the quartje is 5 stuivers, the guilder is 20 stuivers etc. For a long time fiscal experimenters have pointed to the apparent gap between the stuiver and the quartje and have theorised about the possible existence of a fiscal particle with a value somewhere between that of the stuiver and the quartje. Within the last few years experiments have been conducted that throw more light on this area.

2 The Larfield-Egberts Group

In the Netherlands, in 1982, Larfield and Egberts received a large allocation of SPIN money to construct a prototype beverage vending machine; the so called coffee machine. The idea being that it could be used to investigate the nature of fiscal particles by transforming incoming particles into beverages and emitted fiscal particles [Lar83].

The incoming fiscal particles could be controlled and the value of the out going beverages was well known, thus the only unknown factor was the emitted fiscal particles (the so called 'change group'). By applying the laws of conservation of monetary value [Kei78] it would be possible to make predictions about the value of the change group and these predictions could then be compared to experimental observations, any discrepancies in the two would thus be an indication of the existence of undetected fiscal particles.

The first results were inconclusive but there was enough interest to enable a second coffee machine to be built. This was completed in 1985.

3 The Larfield-Egberts Coffee Machine Experiment

In August 1986 the first experiments were carried out with the new coffee machine. They again involved the firing of elemental fiscal particles into it and the subsequent analysis of the fiscal value of the emitted beverages and the change group.

One of the first, simple experiments in this series is described by the following equation expressed in the Larfield-Egberts notation. The first line gives the equation; the initial fiscal particles, the emitted beverage and the change group. In the second line this equation is translated into stuiver terms and, in the event of a discrepancy the third line gives the discrepancy in stuiver terms:-

```
1 guilder → 1 tea + 3 quartjes
20 stuivers → 5 stuivers + 15 stuivers
(no discrepancy)
```

In several of these experiments fiscal value discrepancies were observed between the particles fired into the machine and the value of the emitted beverage and change group.

This was the first indication of an, as yet, unobserved fiscal particle. Some of these events are notated below:-

```
1 guilder → 1 coffee + 2 quartjes
20 stuivers → 6 stuivers + 10 stuivers
(4 stuivers discrepancy)


1 guilder → 2 coffees + 1 quartjes + 1 stuiver
(20 stuivers → 12 stuivers + 5 stuivers + 1 stuiver)
(2 stuivers discrepancy)


2 quartjes → 1 coffee
(10 stuivers → 6 stuivers)
(4 stuivers discrepancy)
```

From conducting large numbers of these experiments it was observed that the discrepancies in monetary value in the equations was always bi-stuiveral, i.e. it was always an even number of stuivers.

Eventually funding was received to build a coffee machine that not only converted the incoming fiscal particles into beverages and change groups but was also able to give a digital read out of the fiscal particles fired into it [Lar87].

By isolating the change group from some of the above experiments and firing them into the new coffee machine it was possible to make direct measurements of the change groups fiscal value. The results showed that some of these change groups did indeed have a fiscal value greater than that expected and that this difference was quantized in even numbers of stuivers.

As a result of these experiments we proposed the existence of a fiscal particle with a fiscal value equivalent to double that of the stuiver, yet a lot smaller in size. Initially the particle was referred to as the 'bi-stuiver' or the 'Larfieldje', but it is now usually referred to as the 'dubbletje' since it is equivalent to double the fiscal value of the base particle; the stuiver. Using the concept of the dubbletje it is possible to balance the above equations, consider the first of them:-

```
1 guilder → 1 coffee + 2 quartjes + 2 dubbletjes
20 stuivers → 6 stuivers + 10 stuivers + 4 stuivers
(no discrepancy)
```

4 Background Radiation

The dramatic results of the above experiments led us to construct a portable dubbletje detector which was then used to take readings under a variety of different conditions.

The results of this were unexpected; dubbletjes were detected wherever the apparatus was used. It appeared that the universe was littered with dubbletjes. Everywhere we set up the dubbletje detector we got positive readings, on the pavements, on shop floors, in the post office. Further experiments of this type are needed, and indeed are already being conducted, but from those initially carried out it appears that there is an even background fiscal radiation, in the form of stray dubbletjes, throughout the entire universe.

This background fiscal radiation is one of the observations predicted by the 'Big Bank' model of the universe [Eng78]. The 'Big Bank' model states that the entire fiscal universe began millions of years ago from one huge Bank resulting in all the fiscal particles that are now in existence. The flux of dubbletjes that we have detected could be the remains of this Big Bank. If this is so, then the universe is quite obviously not in a steady-state, and the next important question is; is the universe expanding continually after this Big Bank, or is it a cyclic process; a series of Big Banks followed by contractions or recessions? There is a possible answer to this from the UK.

5 Fiscal Observations in the UK

The British fiscal experiments community has been in a state of flux over the past few years with new particles being discovered and old particles disappearing within short spaces of time [Law89], [Wob83]. One of the most recent observations is the repeat of Roberts famous shilling measuring experiment [Rob69]. This has been repeated by a number of different groups up and down the country with the results that the spatial dimensions of the shilling (the 5p piece) appear to be getting smaller [Plum88]. If this is taken to indicate that the fiscal universe is indeed in a state of contraction or recession then, taken in conjunction with the background fiscal radiation, this points to a cyclic model of the fiscal universe.

6 Future Research

A conclusive proof of this, and many other fiscal questions, could be solved by advances in the search for a grand monetary unification theory. Research is still being carried out [Lir90], and several European groups have pooled their results and proposed the existence of a basic fiscal particle underlying all the different fiscal particles; the so called hard ecu particle. To date however there is no experimental evidence of the existence of this particle. To solve this a European consortium has been formed called PEC (Pan European Coffeemachine). They propose the construction of a huge coffee machine to be sited underground in Geneva. The PEC machine is envisaged as being completed sometime in 1995 and it may then be possible to conduct more thorough experiments with the dubbletje and even make headway with grand monetary unification and isolate the elusive ecu fiscal particle.

7 References

[Eng78] 'Evidence for the Big Bank.'
B. O'England.

[Kei77] 'The Isolation of the Stuiver.'
J. Keingeld.

[Kei78] 'Violations of The Conservation of Monetary Value.'
J. Keingeld.

[Lar83] 'The Coffee Machine in Monetary Value Experiments.'
B. Larfield, D. Egberts.

[Lar87] 'Building a Coffee Machine with a Digital Readout.'
B. Larfield, D. Egberts, M. Havelaar.

[Law89] 'Investigating Fiscal Inertia: The Thatcher Effect.'
N. Lawson, M. Hesseltine.

[Lir90] 'Beyond the Stuiver; The Cent and the Ecu.'
L. Lirpa.

[Plum88] 'Decreases in the Size of the Shilling.'
V. Plummy, B. Manager, A. Till.

[Rob69] 'Measuring the Shilling.'
A. Robert.

[Sla72] 'Flaws in the Guilder Model.'
J. van de Slagtand.

[Sla75] 'A Stuiver Based Model of the Universe'
J. van de Slagtand.

[Wob83] 'Heads or Tails; The indeterminacy of fiscal particles with spin.'
T.A. Wobbler.

It is with great pleasure that I salute Lambert Meertens on his twenty fifth year at the CWI. I have known him for only a small fraction of that time, yet I have come to regard him as a warm and generous person and a gifted scientist. His "Algorithmics: towards programming as a mathematical activity" is, to me, a constant delight and stimulus, and has been seminal to a large body of exciting research. Research which I consider myself fortunate to have been able to contribute to, for through the work I have come to know the man, and have benefited from his influence and example. In appreciation of his concern for fluid and concise notation, and for his amusement, I append the following (completely uncorroborated) story.

Once, while Lambert, Richard Bird and other eminent algorithmicists were making a ferry crossing to a certain island in the Dutch Waddenzee, they had the misfortune to fall prey to a rather violent storm. The little ferryboat (and its passengers) was tossed up and down on huge waves that drew it ever further out to sea. Now, the Waddenzee is famous the world over for the many sandbanks that lurk in its waters, and walking along these sandbanks at low tide is a recreation that delights hundreds of tourists every summer. Even at high tide, it is most diverting to lean on the gunwale of the ferryboat as it threads its way through the sandbanks and watch the seagulls wading through the shallow water, so shallow, indeed, that it seems as though the birds are walking on the very surface of the water! However, as the little ferryboat (and its passengers) was being remorselessly pitched hither and thither on the towering waves, these sandbanks posed a great danger to the lives of the disconcerted passengers. Seeing that his fellow travellers were alarmed, and that they desperately needed something to distract their thoughts from the dire straits they were in, Lambert decided to do something constructive — some algorithmics. As though by magic, an overhead projector was found, and Lambert began a magisterial account of twiddle-fiddling and widget-fidgeting. He surpassed even himself, forging new notations by the barrel-load, yet as his notations became ever more malleable and fluid, so his audience became ever greener. His slides, full of long, graceful wavers and sinuous twiddles:

$$(\underbrace{\tilde{\oplus}} \approx \underbrace{\tilde{\otimes}})\oint_{\approx\approx} \Re^{\alpha}_{\beta_{\substack{\ell\rightsquigarrow\aleph_{\beta_i}\\ \tau y p^{\infty}\frown_{\alpha_{\zeta_0\vartheta}}}}} \overset{\approx}{\rightsquigarrow} \mathfrak{M}^{\approx}_{\mho}$$

only served to recall to his listeners the violent listing and lurching of the ferryboat, and doubled the effect on their queasy stomachs. By the end of his talk, the company were in a sorry state: slumped over chairs or lying in heaps on the floor, hands clamped grimly over their mouths, some hanging perilously out of portholes, all of them emitting piteous groans and moans. All, that is, except Richard Bird, who had been so captivated by the grace and elegance of his friend's discourse, that he had suffered no ill effects of the sea storm. Still enraptured, he turned around and took in the wretched condition of his fellow travellers. "O sea sick omnia," he gasped.

Lambert, *στην υγεία σου.*

Grant Malcolm
Oxford 1990

Stichting Mathematisch Centrum

Centrum voor Wiskunde en Informatica

Kruislaan 413 1098 SJ Amsterdam
Telefoon (020) 592 9333 *Telex 12571*

Uw referentie
Datum
Onze referentie LM/mh
Datum June 18, 1986

Mr. Ian Currie,
Royal Signals and Radar Establishment,
N113, St. Andrews Rd.,
Malvern WR14 3PS,
England.

Telefoon (020) 592 4141

Dear Ian,

I have made a terrible mistake. On the form I sent you for the September meeting, I indicated that I "am prepared to share a room".

In fact, I am not. I am an incurable insomniac and roam around in my room at night, except if I sleep, but then I snore. Moreover, I smoke day and night. I am sure you do not want to hear about my other obnoxious habits, but you will understand I could not possibly suffer to inflict this upon others.

So please mark me as "not prepared".

Best wishes,

Lambert Meertens.

Postbus 4079 1009 AB Amsterdam

Ja!

Carroll Morgan

It's 11:45, time for the Game. By dribs and drabs the players gather, flowing then in lines down the staircases. Eddies form at the bottom, near the long dark tunnel, before the small groups merge and grow denser as they are funnelled through it.

Forced finally into a single line, they talk nervously — some voices suddenly booming, others scarcely heard — and shuffle forward to choose their equipment: today's lunch. Each item is carefully considered, for there must be no delay at the till. That hidden butter-pat or unidentifiable croquette could jeopardise the last chance to choose an advantageous position when the players sprinkle down onto the long table in the far corner.

For position is everything. The two long lines, as they sit, form the channel through which subsequent moves are made. A concentration of strength, while ensuring a technical victory, diminishes the result because of the inevitable attrition of weaker players' leaving early: they are too far from the sources of power. But any significant polarisation, on the other hand, runs the risk of breaking the play into smaller fragments.

In the best games, a subtle web is formed linking an even dispersion of the strongest; the others are caught, safe, fixed in between. And the moves begin: Soup? Now a disadvantage if it's too hot. How much bread? Without its proper accompaniments one commits an indiscretion that cannot lead to success. And the fruit... is it clear how it's peeled? If not, deep subtleties may be required.

As the play builds toward its climax, the various foods arranged, prepared, and consumed at careful rates, it becomes more and more obvious who the critical players will be. Not those who have finished: they have already lost. Nor those still with three pieces of fruit and six slices of bread: they run the risk of a double loss, being left abandoned with a full plate.

It is those who, well-poised, slowly assemble their bread sculptures, or nibble their yoghurts, neither pulling ahead nor dropping behind. Those also who may yet employ desperate tactics, deciding suddenly to wrap up their peeled fruit in film or sandwiches in a napkin 'for later' so that, then joining the elite group of finalists, they have disrupted the careful rhythms established by their opponents.

And at the very end, an almost electric tension has built up, gathering all players again — with one of them at the focus. It is he that draws in the invisible strands, he who snags the corner of every eye, he who, spreading his hands apart, cries 'Ja!' as he softly snaps both palms down on the table. All rise, and he has won... for today.

The Lambert Walk

Sape J. Mullender

CWI, Amsterdam

I've never seen Lambert run. It just doesn't appear to occur to him to hurry. Once or twice, on my way to CWI, I would catch up with him riding his bicycle through the Oosterparkstraat on his way to CWI. I tried once to keep him company, but I can't ride quite as slowly as him without losing my balance, so now I just wave at him in passing.

I think Lambert doesn't run, because he's lazy. But he's not ordinarily lazy; he's lazy in a very particular way. Lambert doesn't like to do anything that doesn't tax his brain. Running doesn't, so he doesn't.

The effect of Lambert's laziness is quite extraordinary. He'll rack his brain for hours trying to come up with ways of avoiding five minutes' work. Sometimes he succeeds, but sometimes he fails, and then we'll encounter him, after a whole night's effort, on his way home just when other people go to work.

By engaging in what he calls 'Algorithmics', Lambert has found the ultimate form of his favourite occupation: Investing thousands of hours of strenuous thinking in order to avoid having to write boring programs for the rest of his life. Algorithmics, in a sense, is having to program everything at most once, and hopefully not having to program anything at all. You just write down a specification and, by means of a series of mind-boggling twists and turns of the system, out will come a runnable program.

Algorithmics is grist to Lambert's mill. It takes the boringness out of programming and turns it into a task of such gargantuan intellectual complexity that only somebody as lazy as Lambert could possibly attempt it.

So here is Lambert's Principle: Avoid any activity that does not require major intellectual skills. Instead, invest as much time as is necessary to find a challenging intellectual alternative. If this takes so much longer, so much the better.

Being somewhat lazy myself, I have tried to learn from Lambert's ways. Unfortunately, my laziness is of almost the opposite nature. I don't like working my brain too hard, but I don't mind (within certain limits, of course) running, or even cycling. I'll happily spend five more minutes if I can avoid any heavy-duty thinking by doing so. This, of course, made it even harder to learn from Lambert, and it may well be the reason it took me almost seven years to learn something useful from him.

What I learned from Lambert is doing the Lambert Walk. Bear with me while I explain Sape's Principle of making systems run very fast:

There is only one way in which a system can run fast it must execute fewer instructions for the same amount of work. This is, of course, easier said than done. There is a

relationship between the complexity of a program and the number of instructions in it. Complex programs usually need more.

But the number of times each instruction in a program is executed depends very much on where the instruction is in the program. Instructions in an inner loop get executed very often, whereas instructions in the initialization code of a program are typically executed only once. To a certain extent, one can move instructions around in the program and thus reduce the number of instructions executed. And this is precisely what we do.

In fault-tolerant distributed programs there is a large body of code involved with recovery from various irregular situations. Faults that must be rectified include events such as lost data packets in a network, magnetic memories suffering from amnesia, and processors crashing.

As an example of fault-tolerant programming consider a program that makes transmission over a network reliable a network protocol. Such a protocol consists of code that sends packets and returns so-called acknowledgement packets to let the sender know that the data has arrived safely. If a packet is lost, such an acknowledgement will not arrive in time and the sender retransmits the packet. The sender has to start a timer whenever it sends a packet and when the timer goes off before an acknowledgement has arrived, it must retransmit the packet.

The normal case, however, is that the acknowledgement *does* arrive before the timer goes off. The protocol will therefore spend a lot of time starting and stopping timers, but only very rarely act on a timer
modern networks are very reliable; typically only one packet out of a million gets lost.

The alternative and fast way of programming these protocols is not to start and stop timers in the main body of code, but to have some code run periodically (perhaps once every few seconds, a period in which hundreds of packets are sent) which checks to see if the protocol is still making progress (for instance by looking at the current packet number). If it isn't, it will instigate a retransmission.

The Lambert Walk is intended to find pieces of code that are executed very very often and to move them elsewhere where they still do their useful function, but are not executed so often. The code that gets executed when things work normally is called the *critical path*. Doing the Lambert Walk is walking through the critical path, instruction by instruction, examining each one to see if it really must be on the critical path. If not, it is moved elsewhere. Sometimes the code must be reorganized to do this.

In doing such a Lambert Walk, one spends a big intellectual effort, executing instructions in one's mind with the intent of finding the unneccesary ones. This can and does lead to dramatic improvements in the execution speed of distributed systems' code. Unfortunately, in my case, doing the Lambert Walk only makes my machines lazy they can get by executing fewer instructions.

The High Cost Of Reading

Robert Paige [1]

Computer Science Department
New York University/ Courant Institute
New York, NY 10012

ABSTRACT

This paper is in two disjointed fragments - hedged bets on an anniversary gift for one of the Principals of Programming and a consummate wit. The first, on scientific practices, introduces the high cost of reading according to the school of hard knocks. The second, on scientific research, analyzes this cost and proposes a remedy. As a notational device, the term 'you' will be used throughout this paper to denote Lambert Meertens.

1. Introduction And Excuses

During the Congressional hearings just prior to Senate approval of Henry Kissinger's appointment to Secretary of State, he was asked what made him prepared for public office. His answer was, "surviving decades of departmental infighting at Harvard." Likewise, I am certain that twenty-five years at the Mathematical Centrum has fully prepared you for this strange, yet appropriate, *liber amicorum.* If not, there are twenty-five more years to reconsider before the golden anniversary issue.

Before launching into Section 2 on scientific practices, my lawyers told me to make the disclaimer that I was only following the editor's orders. But, as it turned out, my original task of actually creating a humorous chapter proved too formidable, and was abandoned. There were more titles than there were ideas, and the dreary pages that could be hatched from these sad beginnings threatened to be too dull to honor the occasion. Not wanting to be found in my office like the ambitious, but neurotic, would-be novelist in Camus's "The Plague" with seven hundred odd titles and an occasional first sentence, I chose another route to get me off the hook.

It should be a relief to learn that the following earlier proposals for an article have been set aside in a research scrapbook, perhaps to emerge much later after Dijkstra is gone in my own "Personal Perspective on Computing" (cf. [6]).

(i) *Endomorphic Type Reduction in the Lambert Calculus*: This article would be a highly technical prescription for weight reduction. It was tabled after I asked the editor Johan Jeuring whether he thought a paper with this title would be in bad taste, and whether he could offer some advise for someone just entering the field. His

1 No part of this article was supported by DARPA.

enthusiasm for the potential humor in vacuous mathematical endomorphisms convinced me that Jeuring was too civilized to imagine the unspeakable intentions of my true proposal.

(ii) *Software, Hardware, and Lingerie: 1965 to 1990*: This would be a hyperbolic attack (outdoing even Dijkstra) on much that is held sacred in computer science. Solicitations for CASE tools appearing in the trade journals would be compared to comic book ads for Charles Atlas protein tablets. The notion of interoperability would be likened to methods for mating a Chihuahua with a Saint Bernard. Abstract Data types would be parodied as data structure rendering mechanisms, amounting to little more than a crude form of data compression. The notion of software modularity, which DARPA's propaganda mill is attempting to compare with architectural modularity, would conjure up hell holes like Lefrak City and La Guardia airport, where the idea runs amok. (Notre Dame is an example of ad hoc architecture.) My daughter understands modularity in a game that allows a face to be constructed with a pig's nose and the antlers of a moose. Much invective would be unleashed against the major programming paradigms. In short, this paper would be thoroughly boring.

(iii) *By the Seat of One's Pants, or The Through Thread*: Here would be a followup to the weaving application found in [12]. The idea was to concoct several seemingly practical weaving problems, to formalize them as Squiggol specifications, and to derive implementations by transformation. Of course, this involved a good deal of effort, and I needed to refer to the classical combinatorial text by Grunbaum and Shepherd [7] to make it work. But some one faculty member, typical I'm told at The University of Wisconsin, has kept the book out of the library for a year, and can't be forced to return it. So there's a legitimate excuse to duck out of this one. Anyway, the topic is grimly serious.

(iv) *Psychological Anatomy of a Professional Relationship - Dijkstra vs. Schwartz*: The title speaks for itself. The conflict is natural. One man daily tidies his hair by laying each one flat with tweezers. The other uses a blow drier. This article could have become libelous, and had to be set aside -- for now.

Three major reasons led me to abandon these efforts to manufacture humor. First is the realization that computer scientists and their social situations are far funnier than the subject of computer science. Second is that it's often funnier to recount real shared events, filtered by time and eccentricity, than to roll one's own humor. The real clincher, however, is that Michel Sintzoff's laugh in free fall, when used as a benchmark, wins handily over any feeble attempt by me to mold laughable material into an article. I may look like Woody Allen, but that's as far as the comparison goes.

2. Irreverent And Irrelevant Recollections.

As an academic enterprise, personal recollections must avoid all temptation for sentiment in favor of crocodiile tears (e.g., "Too bad Dijkstra can't be horse-back riding with us"); likewise, sordid exposé must be replaced by back-handed compliments (e.g., "Robert Dewar is our leading COBOL scholar."). In this section we share a few whimsical recollections at the expense of our little group in IFIPS WG2.1.

Why should we pay these respects to WG2.1? It is because our organization is a relief from academic life and all of its profundity without sacrificing any of its pretensions. There is the unending search for universal truth combined with the attitude that there are many ways to skin a cat. The result is anarchy. An impasse is reached, exemplified by a failure to resolve so simple an issue as the title and goals of the group. This ironic mixture of uncomprising hope and wavering activity endears WG2.1 to us all.

Now let's extract from our time capsule a few journal entries describing highlights from past workshop meetings. Aside from a useful chronicle of our activities, these descriptions serve to rebut the dangerous anti-academic diatribe of Charles Sykes [16].

W. Virginia, Aug. 1980 (outside Pittsburgh): This was my first experience as an observer. There were lots of flies. It soon became clear that oddballs and reactionaries would be tolerated, even encouraged. When Steve Schumann probed for crackpots in asking us to vote on what properties a good problem specification language should support, I was exposed in proposing performance analysis. But they didn't drop my name from the invited list. During the extraneous speech ceremony at the banquet, you sought various explanations for why Dijkstra wasn't invited. This stimulated me to catch up on my Dijkstra reading and make up for the mysterious blackout on his works during my graduate student days at Courant. These readings (e.g. [6]) made me somewhat envious of those people Dijkstra considered important enough to decorate with insults. Jack Schwartz wears at least three of these epaulets. Although it was a relief to know that you were also not attacked, I still reserve hope that Dijkstra's censors will let through some choice barbs targetting both of us in some future EWD report.

Nijmegen, May 1981: It is said that in academia a little bit of paranoia is healthy. Once again WG2.1 proved a safe haven. Everyone on the tour bus to the Roman ruins flew into a convulsive uproar when my anxiety over setting foot in Germany for the first time caused me to forget my passport. It was a great relief not to be detained at the border, and the trip turned out to be very enjoyable. There was also the banquet at a Greek restaurant with all of us drinking plenty of Retsina, which didn't detract from the great skill with which Michel Sintzoff meticulously picked his left ear with a shish kebob skewer. You were at your best during the banquet with glasses in hand.

Munich Workshop, Sep. 1983: This workshop was well thought out. F. L. Bauer's concern for minute detail was exemplified by the amusing spectacle of student assistants tripping over wires as they raced from speaker to speaker with a single microphone to record every word for the book. The motley mixture of industrial and academic people was enlightening, and tested our perceptions of reality. There was a lot of skepticism. Many people approved (Peter Lucas clapped loudly) when I assailed the slow progress in transformational programming, and provided a few painless escape routes to other areas. Horst Remus from IBM Development Lab at Santa Teresia reported that System R development, which he directed, was set back by years, because the research prototype patched together at IBM San Jose had to be used as the starting point. Hmm! I can't recall you being here.

Harvard Workshop, Jul. 1986: Bill Scherlis ran the workshop until the banquet. At the dinner table in a crowded room, Bob Balzer started his coup d'état. The following exchange took place. Balzer: "What's that, Bill?" Bill: "Oh, that's

tomorrow's schedule." Balzer (heartily shredding schedule): "Now it's garbage!". Later Balzer enjoyed carbon dating John Darlington's seminal JACM paper with Burstall [1]. I think you missed this one, too.

Montreal, Mar. 1987: I risk sentimentality in recalling the rustic accomodations, including indoor outhouses -- a contrast to the highly technical discussions in the meeting rooms. In an inspiring keynote speech recommending that from now on everyone should speak one language, Squiggol, you galvanized the entire group into embracing a Babel of dialects and foreign notations. During the speech, my Chinese coworker Jiazhen Cai, who is usually even more inert than I am, came to life and muttered: "I've heard these things before in the camps." Your strategy worked, and anarchy prevailed.

There may be a bit of irony in recalling a probably apocryphal statistic that is apropos. It's been said that once, during a doctor's strike in Israel, the death rate dropped. Extrapolating from this, one could imagine the crime rate decreasing during a police slowdown, or literacy going up when teachers walk off the job. Likewise, members of WG2.1 might ask whether the local rejection of a uniform notational discipline right now is the best global strategy for ultimately attaining the perfect language. In other words, will an investigation of how transformations manifest themselves in different languages and language paradigms lead to a more generic form of transformational programming that depends mainly on a computational model, and that captures the essence of algorithmic design principles, out of which an ideal notation can be extracted?

It was also at Montreal when a wonderful extended misunderstanding between Richard Bird and me about list and pointer machines led to some of my best work.

Rome, Mar. 1988: Here we viewed the catecombs, where the persecuted religious idealists of the Roman Empire were buried. Like a Fellini backdrop, the subterranean caverns echoed "Silencio" in response to Robert Dewar's operatic basso. With shadows dancing dramatically on his face, he gestured with a finger over his mouth to hush my 11 month old daughter. She hushed. Later, in the spirit of Bunuel, Michel Sintzoff asked for a telephone so that Helmut Partsch could place a long distance call from the tomb. The mood shifted again when we surfaced for an Almadovarean feast bracketted by your hilarious opening banquet speech, and Alberto's captivating after-dinner performance.

Chamrousse, Jan. 1989: This meeting can be remembered for great skiing and wonderful food. Here was a serious meeting, needing no elaboration.

Nordwidjerhout, Apr. 1989: Nothing delights our members more than a healthy disrespect for arcane argot. I will always recall Jim Boyle's lapse, when he referred to Balzer's notion of 'jittering' as 'dithering'. I will always be grateful to you, Lambert, for putting me up in your apartment after the meeting, when all the hotels in Amsterdam had been booked up for the annual marathon race. This saved me from an unwanted tour of the Netherlands. You were a gracious host, and I was glad to learn that beer doesn't cause hangovers.

Lost Valley Ranch, Aug. 1989: It's fitting that the ecumenical spirit should come from Rome to this idyllic spot in the Rockies. Alberto Petterossi implicitly elaborated on the theme introduced earlier in Montreal that an **algorithmic**

transformation transcends disparate notations and even paradigms. He illustrated these ideas with generic loop fusion applied in two different contexts - logic and functional programming. At this meeting we learned how to take a horse to water.

Unfortunately, I missed the next two meetings of WG2.1, but intend to attend the next one to begin gathering data for your golden anniversary volume.

Having just resurrected all these fond memories of the past, it is interesting to consider what might be in store for the future of WG2.1 under the continued guidance of our intellectual leadership, especially in the context of the utopian visions that you have occasionally described. Gazing into an imaginary crystal ball, I see a future meeting of WG2.1 in Shanghai. Buz Uzgalis has convinced one of his oriental patrons to cover our organization's non-profit costs as a profitable tax dodge.

At the banquet you announce a new era in education and research. Over the years, you have wisely used the massive course buyout from STOP funding to spin off educational theories, and also to save yourself from the classroom until the time when these theories would be vindicated. That time has come. In a parallel development, the progress in your transformational research has culminated in the ultimate utopian dream - an executable Squiggol specification of a universal cloner. We are liberated from the class room, from the office, from science itself. In this dramatic moment the final transformation is applied. We hold our breath and pause for thought. The Chinese chef serves paws for food. With much food forethought, we deliberate to resolve the only essential issue that remains -- where to hold the next banquet!

If you got this far, you already know about the high cost of reading, so why go on? Well, the next section is not about the toll taken by bad jokes, nor is it about the price of bifocals. It concerns the computational expense of reading input in a very high level programming language, an expense that I believe is important, ignored, and improvable. The main objective is to augment a data structure selection transformation with a faster way to precondition raw input into initial data structures.

3. The Cost Of Reading

Although reading input is an essential operation, its costs are almost never considered in algorithmic analysis, even when the presentation of input data greatly effects the computation. In determining software performance, input/output costs are usually ignored or separated from other costs. However, in a very high level languages such as SETL [15], where the read operation, must transform high level datatype denotations that are input into elaborate data structures, an efficient implementation is essential.

This section investigates the cost of reading and transforming input in a SETL-like language. New algorithms are presented for turning an input string I into a space-efficient data structure D without duplicate set elements and without redundancy. The worst case time bound is linear in the number of symbols in I. Extensions to these algorithms are expected to support a linear time transformation from D into complex but efficient non-hashed data structures for a new data structure selection transformation with better performance guarantees than is possible in the current SETL optimizer [13].

Currently, both the default and optimized SETL implementations for sets use hash tables that are constructed during the read operation [5]. Hashing to eliminate duplicates, detect redundancy, and construct SETL default set implementations can be done in linear

expected time [4]. The cost to implement SETL optimized data structures has not been reported.

3.1. The Reading Problem

Input data for algorithms and programs are typically in two forms. External form is a string in a language of constant dataype denotations over an alphabet, say ASCII, with size much smaller than machine memory. Internal form is a representation in machine memory for these datatypes. All data that eventually gets input is initially in external form and must be converted to internal form for assignment to program variables.

There are two general forms of read operations - coded read and binary read. Coded read performs three tasks: (1) verifying external data, (2) converting it to internal form, and (3) assigning internal data to variables. Binary read just performs task (3); i.e., it assigns preconverted internal data stored on some intermediate storage device to variables in machine memory. Algorithm texts invariably stick to binary reading of sometimes elaborately preconditioned data (e.g. augmented adjacency lists).

Throughout this paper we assume a sequential pointer RAM model of computation [8, 17], which accesses data using pionters but lacks direct access into arrays, and cannot perform address arithmetic. In practice Task (1) boils down to a parsing problem with running time linear in the number of input symbols. Task (3) is just a copy operation, which takes linear time in the number of words. Although Task (2) is known to take linear time in the number of input symbols for many datatypes, set denotations pose a problem, because duplicate elements in the input string must be detected and removed.

The SETL implementation solves Task (2) for sets by converting each set element to internal form, which is then hashed to remove duplicates and to construct the hash table form of the set on-the-fly. For an input string of m symbols over an alphabet Σ of k symbols, we will show how to solve Task (2) without hashing in worst case time $O(m)$. If the input string is stored in n words of machine memory, then $n \log n \geq m \geq n \log n / \log k$. Thus, our performance is about the same as sorting when k is small. If we assume that verification and conversion of raw string data must examine this data one symbol at a time, then our approach is within a constant factor of optimal performance.

Consider an external data language for a single item of data to be converted to internal form. Let Σ be a k-symbol alphabet and **0** be a special non-alphabet symbol to be used as a sentinel. Consider datatype denotations for alphabet symbols, sets, multisets, and tuples. Let *item** denote a string of zero or more items in the input language. Let sets be of the form {*item**}, multisets of the form <*item**>, and tuples of the form [*item**].

Task (1) of the read operation verifies that the input string belongs to the language just described. If so, it also constructs an ordered tree T that reflects the nested structure of the data item. Tree T contains a node for each occurrence of an alphabet symbol and each pair of brackets for either sets, multisets, or tuples. The children of each node representing bracketted data, say $\{i_1\ i_2 \cdots i_j\}$, are the nodes representing $i_1, i_2, \cdots, i_j$, **0** respectively, where **0** is the special sentinel. Each node n is a record implemented with four fields: *succ.n* is a pointer to the right sibling of n, *child.n* is a pointer to the leftmost child, *parent.n* is a pointer to the parent, and *type.n* is the *type* represented by the node

(either *alphabet*, *set*, *multiset*, or *tuple*). All the alphabet symbols appearing in the input and the sentinel are stored in memory as an initial universe U of data items; each node in T representing an alphabet symbol or sentinel stores a pointer in its *child* field to its corresponding alphabet symbol in U. Assume that Task (1) is solved by a combination of hardware and software.

We are concerned with Task (2), which converts tree T and initial universe U produced by Task (1) into an internal form consistent with the datatype semantics. Task (2) consists of two subtasks: (i) to prune duplicate elements from each set; and (2) to compress tree T into a dag D in which every two nodes representing equal data share the same children. Note that Subtask (2) makes the set of all *child* pointers in dag D represent all distinct data items appearing in the input.

Equality of data items is determined as follows. The alphabet symbols, sets, multisets, and tuples form four disjoint sets. A tuple is an ordered list of elements, where the empty tuple is denoted by []. Two tuples are equal if their elements are component-wise equal. A multiset is an unordered list of elements with possible repetition of values; the empty multiset is denoted by <>. Two multisets are equal if the two tuples, formed from sorting these multisets relative to an arbitrary linear ordering of their elements, are equal. A set is an unordered collection of elements without duplication. Set equality is the same as multiset equality after each set is purged of duplicates. It follows that data represented by nodes at different heights of the tree T produced by Task (1) cannot be equal.

The algorithm described in the next section makes use of the following notation. If x is a variable that stores a pointer to a variable y, then $x\uparrow$ represents the value stored at y. If x is a record with field f, then the term $f.x$ retrieves the value of field f in record x. Each cell in a one-way linked list is a record with special field *succ* storing a pointer to the next successive cell in the list or *nil* if it is the last cell. Each cell in a doubly-linked list contains field *succ* and also *prev*, which is a pointer to the predecessor cell or *nil* if it is the first cell. Unless otherwise stated, sets, multisets, and tuples are assumed to be implemented as linked lists.

3.2. Solution By Multiset Discrimination

Our solution to Task (2) is a tree-to-dag transformation based on methods described in [2] for constructing algorithms without hashing. The most general method is multiset discrimination, which is a way of finding duplicate elements in a multiset whose elements are of a particular datatype. We consider three main algorithms tailored to three different element types - tuples, multisets, and sets.

Multiset discrimination of tuples can be solved in much the same way that multiset discrimination of strings was solved in [2] (based on the earlier array-based method found in [9]), by repeated application of multiset discrimination of pointers . Multiset discrimination of pointers is easy. Let Q be a set of records x, where $child.x$ is a pointer to an element in a universe U. The goal is to compute the set $DomB = \{child.x : x \in Q\}$, and for each element y in U compute the set $B.y$ of pointers to all elements x of Q in which $child.x$ points to y. The solution involves a single linear time scan through the elements of Q.

First consider multiset discrimination of tuples. Let R be a set of tuple-valued nodes in tree T whose children collectively satisfy the following property: if x and y are two

child nodes (not necessarily siblings) and $child.x \neq child.y$, then the values represented by x and y cannot be equal. In this case we say that the tree below R is *dagified*. The algorithm is sketched just below.

(i) Let Q be the set of records $\{(child.x)\uparrow : x \in R\}$. Let P be a workset initialized to $\{Q\}$.

(ii) While P is nonempty, remove a set Q from P and perform multiset discrimination on Q to obtain $DomB$ and sets $B.x$ for $x \in U$. For each pointer $p \in DomB$, if $p\uparrow$ represents the sentinel symbol, then for each pointer $y \in B.p\uparrow$, node $y\uparrow$ belongs to a copy of the same tuple. Choose one of these copies, say the first one in the bucket, and reassign the child pointer of the parent of each node $y\uparrow$ to point to this copy. Otherwise, if $p\uparrow$ does not represent the sentinel symbol, and if $B.p\uparrow$ contains more than one element, then add the set $\{(succ.x\uparrow)\uparrow : x \in B.p\uparrow\}$ as a new element of P. Both $DomB$ and sets $B.p\uparrow$ for $p \in DomB$ need to be reset after each execution of multiset pointer discrimination.

Let m be the total number of all children of nodes in R. Let m' be the total number of pointers in the shortest prefixes needed to distinguish each of these tuples. The multiset tuple discrimination algorithm just above only scans the child pointers of these shortest prefixes. Let k be the total number of distinct pointers among the m' child pointers scanned. Then, following the analysis in [2,9], the preceding algorithm runs in worst case time $O(m')$ and worst case auxiliary space $O(k+|R|)$.

Recall that equality of multisets is defined in terms of equality of tuples formed by sorting these multisets relative to an arbitrary total ordering of their elements. By this definition we can solve multiset discrimination of multisets by reduction to multiset tuple discrimination. Let R be a set of multiset-valued nodes in T in which the tree below R is dagified. The following procedure solves this discrimination problem:

(i) Perform multiset pointer discrimination on the collection of all children of nodes in R. The arbitrary order in which the elements of $DomB$ are stored creates an arbitrary linear ordering on the children of R. We can then use this ordering to rearrange all of the multisets by the following procedure, called *weak–sort*.

(ii) (weak-sort) Assign *nil* to the child pointers of each node in R. Next, for each pointer x in $DomB$ and each pointer y in $B.x\uparrow$, perform assignments $succ.y\uparrow := child.(parent.y\uparrow)$ and $child.(parent.y\uparrow) := y$.

(iii) Perform multiset discrimination of tuples on the ordered multisets of R.

The performance of the preceding algorithm is the same as for multiset discrimination of tuples, except that parameter m' must be replaced by m, because weak-sort requires that every child of every node in R must be scanned.

Multiset discrimination of sets follows the same logic as multiset discrimination of multisets with one exception. Weak-sort must be modified so that duplicate set elements are eliminated. This involves introducing a test to perform the two assignments only when $child.(child.(parent.y\uparrow))\uparrow \neq child.y\uparrow$. The performance is the same as before.

We can now describe the general algorithm for solving Task (ii); i.e., converting the tree T produced by Task (1) into compressed form D. Let d be the depth of T.

(i) For each height $h = 1,2,...,d$ perform Step (ii).

(ii) Partition all nodes in T at height h into set-valued nodes R_{set}, tuple-valued nodes R_{tuple}, and multiset-valued nodes $R_{multiset}$. Then perform the datatype-specific form

of multiset discrimination on each of these sets.

The correctness depends on an inductive argument with the following basis: the leaves of T are dagified to begin with. If T has n nodes, then the preceding algorithm produces D in $O(n)$ worst case time and space.

3.3. Improvements and Applications

The multiset discrimination algorithms just described can be made more efficient by combining the passes for weak-sort and multiset tuple discrimination in the procedures for multiset discrimination of sets and multisets. All these algorithms extend, practically without modification, to handle several input data items together. It is also straightforward to introduce external datatype denotations for fixed numbers, float numbers, and strings, which can be converted to internal form on-the-fly within the bottom up processing of the general Task (2) algorithm. Sets of pairs can be converted to SETL-like maps (essentially adjacency lists) in linear time by performing multiset pointer discrimination on the first components of pairs, followed by similar discrimination on the second components for each image set. This explains how inputs can be preconditioned efficiently for many of the standard text book algorithms.

The SETL data elaboration language [5, 15] is a full-blown language for algebraically specifying data structures that are constructed and assigned with set theoretic input by the **read** operation. We are currently designing an improved data structure language (which can specify data structures supporting operations asymptotically faster in the worst case than those in optimized SETL) suggested in [3, 10], which does not require hashing, and is based on formal complexity considerations stated in [11].

Based on the tree-to-dag transformation of the last section, we can input the same unpreconditioned data as in SETL, transform it, and construct our initial data structures in linear time with respect to the length of the input string. This overcomes a serious shortcoming in our work cited above, where the user was required to completely reorganize the raw data by hand so that the **read** operation could construct the initial data structures without hashing.

Risking oversimplification, we can still convey the essential idea without introducing a morass of detail. Recall that the set U_{final} of all *child* pointers in every node of D represents the set of distinct data values extracted from the input. Our problem is to reorganize these values and store them in data structures that facilitate efficient computation.

If $Q \subseteq U_{final}$, then the set *induced* by Q, denoted by $Q_{induced}$, is the set of all *child* pointers of the nodes $x\uparrow$ and the siblings of $x\uparrow$, for each $x \in Q$. Our data structure specification is a set of triples $[B, W, S]$, such that $B, W, S \subseteq U_{final}$, $W_{induced}, S_{induced} \subseteq B_{induced}$, and $|S|$ is a constant independent of the size of D. Each triple $[B, W, S]$ specifies a relation with key represented by the child pointers in the nodes belonging to $B_{induced}$ and attributes represented by the set S. (The condition restricting the number of attributes to a constant conserves space.)

More specifically, for each pointer $x \in B_{induced}$ there is a one-way linked list l_x whose first record y stores x in its value field. For each pointer $x \in S$ the relation stores a triply linked list as follows. Let id_x identify the attribute associated with x. Copy the list of siblings in D that begins with $x\uparrow$, but remove child and parent fields, and add a field storing id_x and a *pred* pointer. Add each cell in this list to l_z, if it was copied from a D

node with child pointer z. For each pointer $x \in W$ copy the list of siblings in D that begins with $x\uparrow$, but replace the child pointer p by a pointer to l_p, remove the parent field, and add a *pred* field.

The set $B_{induced}$ corresponds to Schwartz's notion of a ***base*** [14]. The columns of the relation formed from S corresond to our notion of a ***strongly based*** representation, and the lists formed from W correspond to a ***weakly based*** representation in [11].

The set $B_{induced}$ can be computed in linear time in the number of siblings in the lists pointed to by B using multiset pointer discrimination. Computing the remaining portion of the data structure takes linear time in the number of siblings in the lists pointed to by S and W. In rough terms, the linear time result for computing the full data structure follows from several restrictions, most important of which is that the bases in different triples must be disjoint. More efficient refinements of these data structures, as indicated in [11] can be constructed with no more than a constant factor overhead.

A general principle of program improvement is to push computation back as far as possible towards the beginning of a program. The earliest point at which this is possible is often a **read** statement. To push computation back beyond that is to place it into the hands of the hapless user, which is what the standard algorithm texts seem to suggest in their ubiquitous requirements for preconditioned input.

The preceding section has presented some evidence that data entry and conversion time are useful points in a program for pipelining potentially expensive computations. It would be interesting to consider the extent to which this can be done without introducing more than a constant factor overhead over ordinary reading. We have at least suggested that lexicographic sorting along the lines of [9] might be usefully combined with reading data without asymptotic penalty.

4. Conclusion

Having considered the high cost of reading, and having offered some improvement, I am forced by the high cost of writing to end the paper here. The high cost of writing is another interesting topic, which can wait until later, perhaps even another twenty-five years after gaining more experience at it.

References

1. Burstall, R. and Darlington, J., ''A Transformation System for Developing Recursive Programs,'' *JACM*, vol. 24, no. 1, pp. 44-67, Jan., 1977.
2. Cai, J. and Paige, R., ''Look Ma, No Hashing, And No Arrays Neither",'' in *ACM POPL*, Jan, 1991.
3. Cai, J., Facon, P., Henglein, F., Paige, R., and Schonberg, E., *Type Transformation and Data Structure Choice*, 1990. unpublished manuscript
4. Carter, J. and Wegman, M., ''New Hash Functions and Their Use in Authentication and Set Equality,'' *JCSS*, vol. 22, no. 3, pp. 265 - 279, 1981.
5. Dewar, R., Grand, A., Liu S. C., Schwartz, J. T., and Schonberg, E., ''Program by Refinement, as Exemplified by the SETL Representation Sublanguage,'' *TOPLAS*, vol. 1, no. 1, pp. 27-49, July, 1979.

6. Dijkstra, E. W., *A Personal Perspective*, Springer-Verlag, 1982.

7. Grunbaum, B. and Shepherd, G., *Tilings and Patterns*, Freeman, 1987.

8. Knuth, D. E., *The Art of Computer Programming, Vol 1: Fundamental Algorithms*, Addison-Wesley, 1973.

9. Paige, R and Tarjan, R., ''Three Efficient Algorithms Based on Partition Refinement,'' *SIAM Journal on Computing*, vol. 16, no. 6, Dec., 1987.

10. Paige, R. and Henglein, F., ''Mechanical Translation of Set Theoretic Problem Specifications Into Efficient RAM Code - A Case Study,'' *Journal of Symbolic Computation*, vol. 4, no. 2, pp. 207-232, Aug., 1987.

11. Paige, R., ''Real-time Simulation of a Set Machine on a RAM,'' in *ICCI '89*, ed. W. Koczkodaj, Computing and Information, Vol II, pp. 69-73, 1989.

12. Paige, R., Tarjan, R., and Bonic, R., ''A Linear Time Solution to the Single Function Coarsest Partition Problem,'' *TCS*, vol. 40, no. 1, pp. 67-84, Sep, 1985.

13. Schonberg, E., Schwartz, J. T., and Sharir, M., ''An Automatic Technique for Selection of Data Representations in in SETL Programs,'' *ACM TOPLAS*, vol. 3, no. 2, pp. 126-143, Apr, 1981.

14. Schwartz, J.T., ''Automatic Data Structure Choice in a Language of Very High Level,'' *CACM*, vol. 18, no. 12, pp. 722-728, Dec, 1975.

15. Schwartz, J., Dewar, R., Dubinsky, D., and Schonberg, E., *Programming with Sets: An introduction to SETL*, Springer-Verlag, 1986.

16. Sykes, C., *ProfScam*, Regnery Gateway, 1988.

17. Tarjan, R., ''A Class Of Algorithms Which Require Nonlinear Time To Maintain Disjoint Sets,'' *J. Comput. Sys. Sci.*, vol. 18, pp. 110-127, 1979.

An Englishman's Difficulties with the Dutch

Steven Pemberton

Want ziet gij den splinter in het oog van uw broeder maar den balk in uw eigen oog bemerkt gij niet.
Mattheus VII, 3

''Wat is die 'regen' waar de anderen het over hebben, ma?''

It was raining. It was wet. I was in Amsterdam. Amsterdam, The Netherlands. A voice behind me, recognising me, remarked: *Engels weer, Steven.*

Now, what was I supposed to make of such a remark. The equivalent of the English comment ''Nice weather for ducks''? That it was raining and that would be a good cure for homesickness for me? Did he mean ''Lucky we don't get this sort of weather in the Netherlands!'', or did he mean ''I wish you lot would keep your weather at home, where it belongs''. I don't know, though I had also noticed a tendency on the Dutch TV weather forecasts to say that when rain was coming, that it was coming from England. Could this be some sort of racial stereotyping in progress? On British TV they tend to blame cold weather on the Russians, in the same sort of way, though it's surprising that they never blame the rain on the Irish, but on the Atlantic instead, even though it usually has to pass over Ireland on its way.

Well, I looked up some statistics for rain. London: 610 mm per year; roughly the same as Rome, Moscow, Peking, Cape Town, Delhi. Amsterdam: 1010 mm per year; roughly the same as New York, Washington, Buenos Aires. So where do the English get their reputation from in the Dutch mind? Especially since if you know what *Nederpop* and *Nederwiet* are, you'll also understand where the word *Neerslag* comes from.

Another racial stereotype that the Englishman abroad has to cope with is the English reputation for cookery. Now I realise that the Dutch cuisine is highly esteemed in the rest of the world, and that the English have therefore a lot of catching up to do, but really, most Dutch people haven't even tasted Yorkshire Pudding (let alone even know what it is!), so it's a bit unfair of them to judge us the way they do.

But the subject I'd really like to treat here is Dutch spelling. When I complain to Dutch people about Dutch spelling, their immediate and unanimous response is ''But what about English spelling!''. This is what is referred to in knowledgeable circles as a *non-sequitur*, or in other words, ''What on earth does that have to do with it?''*

*The English use the same tack when accused of being a Nation-Not-Very-Good-at-Foreign-Languages: ''Yes, but the Americans are even worse!'' By the way, this topic usually comes up when Dutch people discover that I speak Dutch: ''But you're English! How did you manage to do that?''

You see, the thing is, English spelling nowhere claims to be consistent. I've looked everywhere, in dictionaries, encyclopaedias, English language teaching books, all over the place, and I've never found even the suspicion of a reference to a claim that English spelling is in any way consistent. The Dutch, on the other hand regularly have a spring-cleaning session in their spelling to further confuse foreigners in their attempts to learn Dutch, failing in the process to make the spelling consistent, and having the effect of losing continuity. At least the English can claim to still be able to read Shakespeare.

This spring-cleaning is done by a committee of wise persons who as a result of their deliberations produce 'The Little Green Book' full of their pronouncements.

Before we treat the problematic aspects of Dutch spelling in detail, let us first treat some theoretical aspects, especially for the case where you are striving for consistency.

Firstly, one has to recognise that the written and spoken language are not identical, and are actually two variants of the language. The written language is a formalisation of the spoken, merging many spoken variants, and usually lagging behind the changes in the spoken language.

There are several requirements of spelling, among which is that it should be usable by all members of a language group.

Choosing spelling rules is fundamentally choosing the mapping between sounds and symbols on the written page. As a simplification in this process, certain sounds are considered identical, and mapped to a single formal representative; these are then referred to as *homophones*. Take, for instance, the casual pronunciation by an educated English speaker of the phrase 'hot water bottle'. All three sounds represented in spelling with a *t* here are actually different, only the one in 'water' being the sound normally associated with the letter. In fact, the influence of education and spelling is so great, that most speakers of a language are not even aware of homophones*.

There are several possible mappings of sounds to spelling. Firstly there is the one-to-many mapping such as used in the Dutch spellings *ei* and *ij*. These are obviously easy to read (as long as the mapping is one-to-many, and not many-to-many), and difficult to write, since you need training to know which spelling to use for each separate word. For this reason, these spellings will be further referred to here as *Read-only Spellings* or ROS for short.

Similarly, you have many-to-one mappings. The homophones come into this category, as do for instance the Dutch *b* (*bed*, *heb*) and *d* (*dek*, *bed*), and 'context-dependent' spellings, such as caused by the Dutch 'open syllable' rule (treated below). These we will call *Write-only Spellings*, or WOS.

Since one's passive knowledge of a subject is always greater than one's active knowledge, it is obvious that ROS spellings are to be preferred to WOS.

In any case, the ideal spelling choice, for both reader and writer, is the one-to-one mapping (RAS), where each sound has a unique spelling, and each spelling a unique pronunciation.

---

*Homophones are also the source, for instance, of a English speaker's problems with the Dutch pronunciations of the letters w and v. Both sounds occur in English, but they are homophones for English v, and so to the English ear, both sound like a v.

The final, obviously undesirable, possibility is the many-to-many mapping, or OS*. As an example, English has 14 ways to represent the sound *-sh-*: shoe, sugar, issue, mansion, mission, nation, suspicion, ocean, nauseous, conscious, chaperone, schist, fuchsia, pshaw. Of these 14 spellings, most have other pronunciations in other positions. Most Dutch people know the *-ough-* example too (usually by heart), and have no qualms about boring English people with it at parties and other social events.

Don't go and think that Dutch is immune from OS either. For instance, consider the Dutch *-ti-* spelling, like *station*, *politie* and *portie*.

Let us make a remark here. Although RAS is preferable to ROS, the requirement for the spelling system to be usable by the whole language group sometimes requires ROS. A good example is the Dutch *g/ch* difference, where certain parts of the language area still distinguish the sounds.

Another remark worth making is that there need be no formal requirement that a sound be represented by a single letter, as long as the combination remains non-WOS. The English *-sh-*, *-th-* and *-ch-* spellings are examples (though *sh* is non-WOS in dishonest), as well as the Dutch *-tj-*, *-sj-*, and *-ch-* spellings.

With these initial formal comments now behind us, now let us examine certain aspects of Dutch spelling.

The Open Syllable

The first real problem a foreigner learning Dutch spelling is confronted with is the rule of the 'open syllable'. This rule states that if one of the four long vowels written with double letters (aa, ee, oo, uu; treated below) ends a syllable, it should be written only with a single letter. So you get *geven* instead of *geeven*, and *lopen* instead of *loopen*. The definition of an open syllable is already troublesome (caused by, amongst others, multi-letter sounds like *-ch-*), but the real problem is that the rule is not consistently applied throughout the language, but is context-dependent, and inconsistent. For instance, alongside *mede*, you have *tweede*, a case where the rule hasn't been applied, similarly with *waarom*, and if you take *degelijk*, change the *d* into a *t*, you have *tegelijk*, where you just have to know that the pronunciation is different (these could be classified as read-only and write-only inconsistencies).

Furthermore, *e* is an exception when the open syllable is the last syllable in the word. In such an open syllable, *a*, *o* and *u* are still long, but for some reason *e* is not. Compare *sla*, *zo* and *nu* with *ze*, which if the rules were consistent would be pronounced *zee*.

When I was learning Dutch, I asked my Dutch teacher why the open syllable rule was used. "To save ink" was his reply†. Now, while I appreciate that the Dutch are a thrifty race, I'd like to point out that I am aware of no studies that demonstrate that the rule actually does save ink. You have first to balance out all the doubling of consonants

*OS in French texts on spelling is also variously referred to O, AU, AUX, EAU, and EAUX spelling.

†The Little Green Book says that it is *bij wijze van vereenvoudiging* ("as a simplification"). This is of course just one of the many little jokes that they added to perk up an otherwise rather dull work.

that has to take place to overcome the effects of the rule in other places. For instance, suppose the rule didn't exist, so that spellings were context-free. Then *boter* would have to be spelt *booter*, but then you could spell *botter* as *boter*, saving ink in that case. So you see, without a full investigation, we can hardly say that the rule is beneficial, especially considering the problems it causes. Consider the spelling *-eren*. Put a *waard* before it, and you have *waarderen*, but put a *goed* before it, and you have *goederen*. Consider the subtle difference between *grote* and *grootte*. Consider the word *belevenissen*. Of the three, formal, open syllables, only one affects the pronunciation of the vowel. When I first saw this word, I could only guess it was pronounced *beeleveenissen*, using a word like *degelijk* as model.

Proposal. Scrap the open-syllable rule. It is complicated, introduces all sorts of special cases, and is difficult to learn. There are no indications of any advantages. Advantages of the change are that the spelling rules are much simpler; the spelling of vowel sounds is always context-free; there is no need for double consonants any more.

Examples of the change:

goederen → goederen
waarderen → waardeeren
tegelijk → tegelijk
degelijk → deegelijk
belevenissen → beleevenisen
regelen → reegelen
regeren → regeeren
(de) gevel → geevel
(het) gevel → gevel

The Voiced Consonants

The four voiced consonants *b*, *d*, *v*, and *z*, are never voiced in Dutch if they fall at the end of the word, being instead pronounced *p*, *t*, *f* and *s*: they never say *bed* but *bet*; they don't say *rib* but *rip*, and so on. The two choices possible here were: write the letters as they are pronounced (context free), or write the letters as they are, and add the 'not-voiced-at-the-end-of-a-word' rule to the spelling rules (context dependent). I suspect that the spelling committee had supporters from both camps as members, because they did both. The context-freeers got *v* and *z*, and the context-dependers got *b* and *d*. The foreigners got confused.

Proposal. You could drop the context-dependent part. No one has problems with *graaf/graven*, *lees/lezen*, and enough Dutch people already write *t* where they should write *d*, that dropping this rule would go rather smoothly. I suspect though that the educated part of the Netherlands would rather feel that the uneducated had won a battle if this change were effected, so dropping the context-free part might have to be considered. Then you would have to get used to spellings like

een glaz bier,
ik leez in bed,
ik beloov het.

The Long Vowels

Each of the vowels *a*, *e*, *o*, and *u* have 'long' counterparts (*vrij* in Dutch), all written by doubling the vowel: *aa*, *ee*, *oo*, *uu* (except in open syllables as pointed out already). The odd one out is *i*. It also has a long version, but it is written *ie* (causing complications with the open syllable rule, too).

In fact, originally, the spelling of a long i *was* ii, although you also have to know that at that time, there was no separate letter *j*, and the *j* shape was just a stylistic variant of the letter *i*. Only later were different sounds assigned to the two shapes*. So *ii* was written *ij*. Unfortunately, the pronunciation of the long *i* changed into the diphthong it now is, so that *ei* and *ij* gained the same pronunciation, and a new spelling for long *i* had to be found, i.e. *ie*. The old pronunciation of *ij* can been seen from words like *Parijs*, which used to be pronounced *Paries*, i.e. with the same vowel sound as the French use, with *Berlijn* similarly, and, exceptionally, *bijzonder*. Many Dutch people in my acquaintance also say *Bie ons* for *Bij ons*.

Proposal. For consistency, drop the *ie* spelling, and reintroduce the *ii* spelling for long *i* (though spelt *ii*, not *ij*).

Examples:

fiets → fiits
die → dii
vriend → vriind
driewieler → driiwiiler
liefdesbrieven → liivdesbriiven.

The IJ disaster

While we are on the subject of the long *i*, we may as well also treat the long *ei* completely. I need hardly go into the chaos that *ij* causes in Dutch, firstly because of the ROS *ei/ij*, many people don't know whether to write projectleider or projectlijder; secondly, what do you do if the character isn't available on a typewriter or similar? Write it as a *y*, or as two letters *i* and *j*? Thirdly, where to put it in the sorting order. Because of problem 2, along with the Y's (the solution used in phone books), or because of its history, with the i's (the solution used in dictionaries). And finally, you have the problem of the simple word *bijou*. A word taken from French, with 5 letters, *b-i-j-o-u*, not *b-ij-o-u*. The problem with this word is that every Dutch person knows that there's something special about it, and half of them can't remember what it is that's special, and through hyper-correction use a *y* and spell it as *byou*. You see the word spelt everywhere like that. Even stores that project themselves as up-market (and therefore as 'educated'), such as the *Byenkorf*, spell their jewellery department *Byoux*.

Proposal. Get rid of *ij* completely; its sound is already available with *ei*, and the chaos, confusion and inefficiency caused by the letter is too great to worry about the problem of the uneducated having won yet another battle. I realise that this will be difficult, since in some respects the letter *ij* is *the* symbol of Dutch identity: the one letter

*The same thing happened with *u* and *v*, too, which is why a *w* is called a *double-u* in English, and a *double-v* in French.

that no other language has in its alphabet, the one letter that causes the Dutch language to appear in international standards standardising character sets for computers. To appease the possible lack of National Pride that might be caused by abolishing the letter, I would propose then keeping it for two cases: the stretch of water by Amsterdam called the IJ (since it would cause the Dutch a double loss of pride to have to go sailing on the Ei), and (in order to retain a need for the lower-case letter) for the Bijenkorf and its jewellery department.

The -isch Stupidity

How the committee ever got away with this one is really beyond me. I suspect that it's either intended as an enormous joke on the Dutch population (which they still haven't got), or some sort of appeasement to hardline members of the committee. After changing words like *bosch* and *visch* into the phonetic *bos* and *vis*, they go and leave the ending *-isch*, with both an incorrect vowel sound and an incorrect consonant cluster, untouched. Madness.

Proposal. What do I need to say? Correction of both faults immediately.

Example:

historisch → histooriis

A Larger Example

In order to try and demonstrate the effects of these proposals, I include a piece of Dutch text with (I hope) all changes made. For copyright reasons, I use a story by a colleague. An English translation by myself appears in *The CWI Quarterly,* December 1984. Of course there are other possible changes to spelling that I could have proposed, such as the representation of the schwa sound (currently spelt in Dutch with *e* (*ge-*), *ij* (*-lijk*), *ee* (*een*), *'* (*'t, z'n*), *en* (*-en*), *i* (*-ig*), and so on, and so on) but I have left those out of the current discussion and transcription.

Woonen in Amsterdam

S. van der Ton

Omdat ik al eerder naar Neederland geweest was, wist ik dat ze mei 3 vraagen bei de grenz zouden stelen: Hoe lang bleivt Uu?, Hebt Uu al een ıetourkaartje?, en Hoeveel geld hebt Uu bei Uu? Omdat ik verwachte dat 2 van mein antwoorden onbevreedigend zouden zein, hiild ik vast in mein hand de uitnoodiging van het MC, terweil ik mein kofer, zoo groot als een klein huiz, meetrok.

Een vriindin was met mei meegekoomen en zei ging eerst. "How long are you staying?" "A week". "Have you a return ticket?" "Yes". "How much money have you got with you?" "250 guilders". Goede antwoorden alemaal, en dus kon ze verder gaan. Toen keek hei naar mei. "Are you together?" "Yes" "OK, you can go". En zoo makeleik was het! Ik vraag meizelv af wat hei dacht dat ik in mein kofer had, voor 1 week!

Amsterdam. Stad van grachten, fiitsen, trams, bars dii laater dan 11 uur 's nachts oopen bleiven, en een hipy op elke hoek dii mafe Bob Dylan liidjes staat te zingen.

Het was mein bedoeling om geen buitenlandse Engelsman te zein dii aleen met andere Engelsen omgaat, en wiins huiz een plek Engelsachtigheid is. Nee, ik wilde een echte Amsterdamer zein.

Naatuurleik, de eerste stap om een Amsterdamer te worden is een fiits koopen op het Waaterlooplein. Dat deed ik, en een heel mooie fiits met drii versnelingen was het. Een week later ontdekte ik de tweede stap om een Amsterdamer te worden: jouw fiits wordt gestoolen.

In feite heb ik een weinig bekend feit ontdekt: fiitsers moeten een fiitsbelasting betaalen, en dii wordt geheeven door jouw fiits mee te neemen en te verkoopen. Het huidige taariiv is 1 fiits per jaar: vraag aan wii dan ook hoe lang ze in Amsterdam woonen, en hoeveel fiitsen zei verlooren heben — het is alteid hetzelvde getal. De enige uitzondering dii ik ooit gevonden heb is Arthur Veen. Dii had in 4 jaar geen fiits verlooren. Maar binen een jaar werden 4 fiitsen van hem gestoolen, dus moest iimand meegeluisterd heben.

Mensen vraagen aan mei of ik geen last om rechts te fiitsen heb. Maar het antwoord is nee. Ik geloov dat het mensdom in tweeën gedeeld is: degeenen dii een landkaart leezen door de kaart te draaien in de richting van de reiz, en degeenen dii hun hersens draaien in de richting van de reiz. Van de tweede soort zeinde, hoevde ik aleen mein hersens om te draaien, en had ik geen last van de verandering van links naar rechts. Doch nu wel met landkaarten leezen...

Evenwel, ik verwachte dat fiitsen makeleiker zou zein in dit plateland. Niimand waarschuuwde mei voor de verplichte pasaagiir dii je achterop je fiits moet meeneemen in Amsterdam; ik vind het erg, hoor. En verder zal ik nooit wenen aan een rood licht dat 'gaa' beteekent voor fiitsers. Vreeseleik is dat.

Nog een stap om Amsterdamer te worden is een rel te ziin. Ik denk dat de poolitsii zich dit realiiseerde en, als vriindeleik welkom, een rel orgaaniiseerde in mein eigen buurt. Ik woonde toen in een rustige straat met boomen en voogels, geen gewoone relstraat dus, en op een aavond, toen ik van het werk naar huiz fiitste, vervolgde de poolitsii enkele relschopers in mein straat als een soort "surprise party", en begon mei neer te slaan.

Net dii week had ik mein eerste drii Neederlandse zinen geleerd, en dit was mein groote kans dii zinen te oefenen. Dus, toen ik het weteleike miiniimum aantal klapen gekreegen had, kon ik van de grond opstaan, en zegen: "Ik ben Engels, ik woon in deeze straat". Dit bleek hem te oovertuigen, en hei schoov mei onvriindeleik de straat uit, met hulp van wat van zein vriinden, voor de pret.

Dii aavond dus kreeg ik de kans om het splinterniiuwe Acadeemiis Ziikenhuiz te proobeeren, terweil mein arm gefootoograafeerd werd.

Neederlands leeren is, naatuurleik genoeg, een belangreike stap om Amsterdamer te worden. Wat ik net ontdekt heb is iits dat Uu zeeker al weet, naameleik het georgaaniiseerde plan om Engelstaalige mensen te verhinderen om Neederlands te leeren. Uu weet het zeeker al, maar voor de Engelse leezers, zal ik het eeven uitlegen.

De hoofdtaktiik is deemooraaliiseeren: "Neederlands is een ver-schrik-ke-lei-ke taal, vind je niit?" zegen ze als begin. Subtiiler is: "Wat spreek je goed Neederlands" als je staamelt: "Mag ik mischiin een andere kop van kofii alstuubliift heben" of "ik vraag Uuw pardon?". En ten derde, als je in een winkel bent, en één enkele fout maakt, of een

woord niit verstaat, gaan ze verder in het Engels, en wat je dan ook zegt, ze gaan niit meer terug naar het Neederlands.

Maar waarom dit plan, vraag je? Naatuurleik, als Engelsen Neederlands spreeken, met wii kunen Neederlanders dan hun Engels oefenen?

Woonen en werken in Amsterdam heeft een verbaazend gevolg gehad: als ik naar een internaationaale conferentsii gaa, zegen mensen teegen mei "What good English you speak!". Ik weet nog niit zeeker of ik "oh I lived there awhile" of "oh, my mother's English, you know" moet zegen. Maar het verschrikeleikste gebeurde op vaacantsii in Griikenland. Ik sprak met een Engelsman, en zei dat ik op weg terug naar Amsterdam was. "Oh!" zei hei, "I *thought* you weren't English!"

Albertus Pettorossi (+) Lamberto Meertens salutem dicit plurimam.

Magnopere tibi gratulor, amice carissime necnon clarissime collega, istos viginti quinque annos apud Centrum Mathematicae et Informaticae feliciter peractos.

Scientiam vero constanter ac strenue auxisti, sed quaeso ante celebrationem tui quinquagesimi anni apud CWI fac ut diploma consequaris ad currum automobilem vehendum!

Latino sermoni quo utor ignosce quaeso, carissime: mihi enim patrius sermo hic est, sed tibi quoque intellectu difficile non erit, cum et laponicum sermonem ipse intellegis!

Gratias agens tibi propter amicitiam tuam, peramanter dico: fac ut valeas!

Datum Romae, Kal. Jan. anno MCMXCI p. Ch. n.

(+) collega in coetu laboris 2.1 Foederationis internationalis pro elaboratione informationis.

Annals of Computing History
Extra Aflevering: Radikale Automatisering

Guido van Rossum

Mijn eerste ontmoeting met Lambert was bij de automatiseringsclub van de PSP. Deze partij zat nog in de lift en vanwege het groeiende ledental was een computer aangeschaft. Tot op dat moment — en nog lang daarna, want ook dit automatiseringsprojekt duurde langer dan gepland — werd de administratie gevoerd met zuiver mechanische middelen, die de indruk wekten in de vijftiger jaren te zijn aangeschaft. Voor ieder lid bestond een aluminium plaatje, waar met een electrisch gedreven huiskamerversie van de stoomhamer letters geponsd konden worden. Voor het verzenden van het partijblad (afwisselend "Bevrijding" en "Radikaal" geheten, per decreet van het partijcongres) werden van de plaatjes afdrukken gemaakt op wikkels. Misschien heeft de PTT wel de doorslag gegeven bij de beslissing tot aanschaf van een computer door de op administratief gebied beslist behoudend denkende radikalen. Wie weet was de invoering van de postcode, en met name de tariefsverhoging voor drukwerk dat zonder deze aanduiding werd aangeboden, een goede gelegenheid om met de tijd mee te gaan.

Het automatiseringsprojekt werd direct radikaal aangepakt. Er was een belangrijke randvoorwaarde aan het projekt gesteld: de adresgegevens van de partij mochten het partijburo niet verlaten. De koude oorlog was immers nog in volle gang (dit was rond 1980!) en de BVD lag overal op de loer. Wanneer het partijblad door de PTT niet prompt op tijd bezorgd werd, beweerden kwade tongen dat dit kwam doordat de BVD de oplage had onderschept om eens op zijn gemak te bekijken wie er zoal op de ledenlijst stonden. Wie weet hadden die tongen gelijk. In elk geval mocht er geen sprake van zijn dat de automatisering van de PSP ook de BVD werk zou besparen: computer-serviceburo's en dergelijke oplossingen (zoals gehobby bij SARA!) waren uit den boze.

Vandaar dat de PSP niet alleen een eigen computer aanschafte, maar ook de ontwikkeling van de benodigde programma's zelf ter hand nam: de partij had immers leden genoeg met informatica-kennis, en als goede partijleden droegen die de zaak zo'n goed hart toe dat ze avonden en weekenden wilden opofferen. Op een van de oproepen dienaangaande in "Radikaal" (of was het "Bevrijding"?) reageerde ik, informatica-student te Amsterdam, tesamen met misschien een dozijn anderen.

De situatie was als volgt. Er was een kleine mini-computer aangeschaft van een onbekend merk (de PC was nog niet uitgevonden) met twee terminals, een harde schijf, een cassettedrive, een operating system (ook van onbekend merk) en een COBOL-compiler. Met de conversie van het ledenbestand was al een begin gemaakt door de aanstaande systeembeheerder, hoofd van de partijadministratie, en een "computer literate" lid van het partijbestuur (niet Lambert). Dit gebeurde met de standaard editor van het operating system, waarmee je een record van het bestand op het scherm kon halen, wijzingen en terugschrijven. Er was een soort invulsysteem bij waarmee de velden van het record — Naam, Adres e.d. — konden

worden aangegeven, en zelfs, als mijn geheugen me niet bedriegt, kon je ermee zoeken, maar echt handig was het allemaal niet.

Een van de taken van de software-groep was daarom een net mutatieprogramma te schrijven — in COBOL uiteraard. De andere opdracht was een programma te maken waarmee het ledenbestand of een selectie eruit kon worden afgedrukt, hetzij om als overzicht te dienen, hetzij op etiketten, gesorteerd en gebundeld op postcode. Om mislukking niet bij voorbaat te verzekeren werd er eerst uitgebreid vergaderd over het ontwerp van beide programma's. Snel bleek dat veel leden van de groep van goede wil waren en alles wisten van computers en programmeren, maar nauwelijks verstand hadden van ontwerpen, laat staan van het moeizame proces waarin automatiseerder en geautomatiseerde (in dit geval de partijadministrateurs) het eens worden over wat de computer eigenlijk moet gaan doen.

Toch werden we het eens. In mijn herinnering lijkt het of we ondertussen een half jaar verder waren, maar misschien overdrijf ik wat. De taken werden verdeeld. Er waren eigenlijk maar twee grote programmeerklussen te onderscheiden, maar dat kwam wel goed uit, gezien het geringe aantal vrijwilligers dat werkelijk genoeg tijd en kennis van zaken had.

Ik nam de verantwoordelijke taak op me om het mutatieprogramma te schrijven, volgens de democratisch tot stand gekomen specificatie.

Hiervoor moest ik eerst COBOL leren.

Weliswaar was ik hier op college al eens mee in aanraking gekomen, maar ik kon niet zeggen dat ik het echt in de vingers had. Als vingeroefening besloot ik een echte editor te schrijven — ook uit noodzaak, want de standaard editor kon slechts één regel van het programma tegelijk op het scherm toveren. Ten gunste van de leverancier van computer en compiler moet ik zeggen dat de geleverde handboeken de werking van de compiler en de regels van de taal duidelijk uitlegden. Ook waren er wat uitbreidingen aan COBOL gemaakt die de constructie van de door mij beoogde editor mogelijk maakten — terwijl dit toch niet het soort programma is waarbij je direct aan COBOL denkt. Hier stond tegenover dat de compiler bestond uit zevenentwintig passes die een voor een in het minuscule werkgeheugen werden geladen, zodat zelfs de eenvoudigste compilatie een eeuwigheid duurde. En uiteraard kon je vrijwel niets anders meer doen op de computer zolang de compilatie liep.

Was de traagheid van de compiler voor mijn vingeroefening nog acceptabel, het werd een serieus probleem tegen de tijd dat het mutatieprogramma zijn voltooiing naderde. Een compilatie van het volledige programma kostte makkelijk een hele avond, zodat ik mijn code wel twee keer nakeek voor ik liet compileren! Ondertussen was de computer op andere tijden ook voor de andere programmeurs nodig (die trouwens dankbaar van mijn editor gebruik maakten). Ook waren de partijadministrateurs al druk bezig met de bestandsinvoer. Af en toe maakten deze al uitdraaien die ergens voor gebruikt werden en op tijd af moesten, zodat het verkrijgen van computertijd voor mij niet altijd even makkelijk was. Gelukkig had ik veel vrije avonden en weekenden, al werkten de administrateurs ook vaak genoeg over — misschien nog wel meer zelfs.

Er waren andere tegenslagen: de hardware raakte onklaar, het operating system gooide het bestand door elkaar waardoor leden verdwenen of juist verdubbelden, de programmeur van het afdrukprogramma had zo'n anarchistische stijl van programmeren dat iedereen voor de goede afloop vreesde. (Volgens Lambert viel die stijl trouwens samen met 's mans politieke overtuiging — wie weet is hierin een algemene wetmatigheid te vinden?) Ook werd er in de regelmatige voortgangsvergaderingen nog fiks aan de wensenlijst voor de programma's gesleuteld. Sommige wensen bleken bij nader inzien onvervulbaar, andere kwamen op nadat de eerste tests gedraaid waren. Niets bijzonder dus, eigenlijk!

Lambert bracht mij hierbij eens tot wanhoop door met een voorstel te komen voor prachtige free-format invoer van queries, zo van "vind alle leden die Jansen of Janssen heten, in Amsterdam wonen en lid zijn van een milieuwerkgroep." Later, toen ik bij het CWI (toen nog Mathematisch Centrum) werkte, herkende ik een echo van dit voorstel in de grammatica voor Booleaanse uitdrukkingen in Lamberts taal B (nu ABC). Het was een prachtig voorstel, echter totaal onuitvoerbaar met de door COBOL aangereikte middelen. Ook had ik de indruk dat de administrateurs, die uiteindelijk de gebruikers van het mechanisme zouden zijn, het niet echt snapten. Er werd dan ook uiteindelijk besloten tot een veel eenvoudiger oplossing, minder elegant maar praktischer onder gegeven omstandigheden — iets wat ik wel vaker heb zien gebeuren met Lamberts voorstellen...

Het mutatieprogramma naderde zijn voltooiing, en mijn medewerking aan het projekt zijn einde. Ik was een beetje uitgekeken op het projekt en gelukkig was de groep na weer eens een oproep in "Bevrijding" (of was het "Radikaal"?) uitgebreid met nieuw bloed, waaronder een talentvolle jongen aan wie ik de voltooiing van mijn geesteskind met vertrouwen kon overlaten. (Ik geloof niet dat er ook maar een vrouw of meisje bij het hele projekt betrokken was.)

Ikzelf maakte mij op voor de voltooiing van mijn studie. Mijn kontakten met Lambert begonnen echter nu pas echt. Na afloop van een van de vergaderingen raakten we aan de praat over wat ons verder zoal bezighield (voornamelijk computers!) en op Lamberts voorstel togen we naar de kroeg op de hoek. Hier verhaalde hij in gloedvolle bewoordingen over zijn werk op het Mathematisch Centrum: hij was een programmeertaal genaamd *B* aan het uitvinden, een taal die nu eens echt makkelijk in het gebruik was. Ik was een en al oor (en vermoedelijk wierp ik ook het nodige tegen, want programmeertalen hadden toen al mijn warme belangstelling).

Om kort te gaan, wat later kreeg ik van Lambert de hint dat er op het MC een vacature was waar ik maar eens op moest gaan solliciteren. Ik was immers op een haar na afgestudeerd, en op zoek naar een baan. Ik schreef een brief en werd uitgenodigd voor een sollicitatiegesprek. Een of twee avonden van tevoren werd ik door Lambert opgebeld: hijzelf zou bij het gesprek aanwezig zijn, maar het ware beter als ik niet liet merken dat we elkaar al uitgebreid ontmoet hadden. Ik kreeg tijdens het gesprek het klamme zweet in de handen toen de afdelingschef vroeg door welke MC-medewerker ik getipt was over de vacature (dit had ik in al mijn naiviteit op mijn sollicitatieformulier ingevuld). Verder was alles in orde, ik werd aangenomen, en de rest is geschiedenis — dat schrijf ik later nog wel eens op.

Squiggoling Cryptograms
—a secret revealed—

Arno Siebes
CWI, Amsterdam
arno@cwi.nl

Abstract

It is common knowledge that the scientist-in-the-street would not get very far as a novelist. In general, their papers are utterly boring, the plot is thin, their style is decayed and their sentences are carved in stone.

The plot often consists of one main result; a novel often has many plots intertwined to entertain the reader. The only way they can conceive to reach their shallow plot is by a tortuous list of definitions, lemmas and lesser theorems; about as entertaining as a Sunday night in Spakenburg. Finally, they still haven't learned that a dead sentence like: "Let x be 5, then x^2 equals 25" could be phrased more excitingly like:"x is the coolest kid in the block, like, man, his jacket says 5; his brother is square, the dude's favourite number is 25"; the best results are reached if such sentences are read in a rap rythme.

Most scientists are aware that doctors prescribe their papers instead of valium. Therefore they have thought of a number of tricks that might help the reader to stay awake. The most hilarious is clearly the invention of the abstract. In such an abstract, the scientist already states his main theorem. Consider a novelist, giving away the plot on the first page...

Abstracts are a time-honoured tradition, this author did not want to break with. However, he feels that an abstract should be a short piece of nonsense, preferably stating nothing. Imagine me giving away my plot, with such a subtitle. No way Jose.

1 Introduction

Surely, anyone who has met Lambert for over 5 seconds must have noticed that he knows an incredible amount of facts on completely(?) unrelated things. In fact, I use Lambert as my personal encyclopedea; this is much easier, as you do not even have to know spell.

There are many theories explaining how this curious fact came to be. The most entertaining explanation is given by a family doctor who wishes to stay anonymous; his explanation went as follows [5]:

> ...it is well-known that some infants put anything in sight in their mouth. Years ago, a young couple, came to me with their son L.G.L.T., who had stuck a complete encyclopedia in his mouth. When I tried to get the books out, I slipped, kicked the kid in his stomach, and as a result of the shock he swallowed the complete set. As it was genuine leather bound, no harm could possibly be done, therefore, I let him digest the encyclopedia. And boy, did he digest ...

Although it is well known that the best way to a man's heart is through his stomach, *cf.* [7], I do not belief that the same holds for the brains.

Anyway, back to our subject. As I was saying, Lambert is interested in diverse areas like: languages (both syntax and semantics), solving cryptograms, programming, modern physics et cetera et cetera. The aim of this paper is to show that these areas are not as unrelated as one might be tempted to think[1].

The only subject that will be left out is modern physics. Anybody who has ever seen Lambert typing knows why. With a cup of coffee in one hand and a cigarette in the other, typing is a one-finger exercise. Clearly, Newtonian mechanics is a good enough approximation.

So, in the remainder of this paper, we will briefly discuss etymology, grammar, cryptograms, programming and then the secret is revealed. We conclude this paper with the conclusions.

As should be clear from this short description, this paper is the result of multi-disciplinary research. Usually, this puts a strain on the reader as he is supposed to know his way in the various fields used. However, in this particular case, the author does not know what he is talking about. Not hindered by any knowledge in any of these fields he happily types away. Hence, the reader does not have to have any knowledge whatsoever; in fact, knowledge may be a serious obstacle. Note that the author's lack of insight in no way invalidates his conclusions. Contrary, the most devastating results are often reached by crackpots, *cf.* any newsgroup you like.

2 Etymology

Nobody, well except nerds and gits, speaks the same language as his parents; or, for that matter, listens to the same music and wears the same clothes. Imagine the 2 Live Crew, dressed in tuxedos, singing Christmas Carols.

[1] To those of you who think "hey, he is giving away something of the plot now", I can only say, this is not called giving away, but building up the tension.

However, if you nick a really old book, say from the sixties, from the library, you will discover that you can actually read it. Although the vocabularies have changed, for example, they said "like wow man" instead of the cool "like man" we use.

If you jump over more than one generation gap, languages change even more. The science of generation gap jumping is colloquially known as Etymology. More precisely, Etymology traces back the origin of words. For example, the English word *Queen* can be traced back to the Old English word *cwen* which simply means wife. Actually there are people who feel that the appearance of a word may change, but its meaning does not, as a leopard cannot change its spots. Clearly, such feelings are not very common among royalists.

Etymology is one of Lambert's favourite playgrounds. In fact, he once indicated a way to translate the (in)famous Dutch word *gezellig* into English based on the etymology of gezellig. It appears that this word stems from the word *gezel*. A gezel was a craftsman who was also pupil of a master of the guild inorder to become a master himself. Such pupils (gezellen) lived in a dormitory, the atmosphere in those dormitories was rather pleasant; hence the word *gezellig*. So, to translate gezellig, simply use *pupily*.

However, back to our subject. Above we have seen that etymology can be used to map words in a contemporary language onto (sets of) words of a much older language.

This suggests that all of contemporary English could be translated to say, Old English. However, this is not true as words like *television* are actually derived from classical greek. Still, the suggestion holds ground, only we have to be more careful.

Like modern English, Old English evolved from an (even) older language, viz., Indo-European. As a matter of fact, classical greek (and thus modern greek) also evolved from Indo-European. Hence, perhaps all English words can be mapped on Indo-European words?

Alas, during the colonial days words slipped into the English language that were not of Indo-European descent. So, if we want a complete map to an older language, we would need some sort of a *first language*. So, the question becomes: "is it reasonable to assume that all languages evolved from the same first language?" To answer this question, we turn to biology.

Basically, biologists can be classified into two camps. One silly way to accomplish this is to differentiate those who see two kinds of biologists from those who don't. A more useful differentiation for our purposes is to distinguish the creationists from the Darwinists.

Creationists are a happy lot, if only because they believe that God is a great practical joker. You see, all the fossils that are found were put there by God on purpose. Moreover, he already knew that the happy finder would rush home and write a very learned paper about his fossil and all the consequences it has for evolution theory. But this fossil is just as fake as the Piltdown Man.

Darwinists are a much more troubled lot. Not only do they have to change their theory with every new fossil, but the creationists tell them they are wrong and will find out after they died. Darwinists do not have the same kind of ammunition to shoot at the creationists.

Remarkably, there are some very basic facts on which the two classes (almost) agree. The creationists know that all human beings descend from Adam; the bible says so loud and clear. Based on genetic evidence, the Darwinists know that all people descend from the same mother!

So, all biologists know for a fact that we descend from a small group, which we could call the first people. Hence, this fact is true.

One of the descriptions of a human being is an animal that talks. To talk, one needs a language. So, the first people had a language, which we will call the first language.

By induction, we have now proven the following lemma:

LEMMA 1: All words in any contemporary language can be mapped onto (sets of) words from the first language.

An immediate but important consequence of this lemma is:

COROLLARY 1: All Dutch words can be mapped onto the first language

Further on in this paper, we will need a property of the first language that can only be phrased using Category Theory. This might baffle quite a lot of people, but it certainly does not baffle me. If I wanted to brag, I would refer you to my prize-winning essay on the etymology of the word *cocktail*, which is completely categorical. However, I'm modest, so I won't [2].

As not all of my readers will be dyed in the wool categorists, I am not going to wade through all of the gory details. Rather, I appeal to your common sense by stating the obvious: mapping words to words actually describes a category, called $Word$, with vocabularies as objects and arrows derived from the word-mappings. Then, Lemma 1 immediately gives us that:

COROLLARY 2: The first vocabulary is the initial object in $Word$

For nit-pickers, yes we should use $\mathcal{P}(vocabularies)$ et cetera. However, it is the spirit that counts; rigidity is for amateurs who are afraid to put their foot in their mouth.

[2] However, out of fear that the reader would forget how elegant my derivation in that paper actually was, I will give a brief abstract. In category theory the prefix *co* always means: dualise that what follows. Well, cktail follows. The only sensible way to pronounce this is cocktail! Hence, cocktail is a self-dual concept. As the domain of discourse is the set of beverages, both alcoholic and non-alcoholic (which are obviously dual classes), it has to be a mixture of these two classes. Presto, is this elegant or what?

3 Grammars

Sets of words are not all there is to a language. This observation goes back to the famous Neanderthaler linguist Herr Professor Doctor vom Grammatik. He was the first man who discovered that not every random sequence of words he might choose to utter made sense to the man in the street. To distinguish the meaningful from the senseless sequences he defined the notion *sentence* as follows[4]:

DEFINITION 1: A sentence is a sequence of words such that one has a reasonable chance that the average blockhead grasps what your intention is.

Note that Herr Professor Doctor vom Grammatik did not distinguish between the different languages in his initial definition. However, after a life of travel, he realised the importance of that distinction, and he changed his definition accordingly.

After his father started kicking up the daisies, Herr Professor Doctor vom Grammatik jr., realised that it might be beneficial if one could distinguish sentences from non-sentences (as he dubbed them). Therefore, he started to write down all sentences from his language. Tragically, he died before his seminal work was finished.

However, as they formed a industrious clan, Herr Professor Doctor vom Grammatik III followed in his father's footsteps. In fact, centuries went by, and still the work was not finished. Then, one day Frau Professor Doctor vom Formellen–Grammatik had a brilliant idea: "wouldn't it be better if one gave a finite specification of all sentences rather than a complete collection?" After many years of hard labour, she made one up.

Pretty soon, such specifications were made up for all languages available. To honour her, and in fact all the brilliant linguists from that family, such a specification is nowadays called a Grammar.

Perhaps the nicest property of a grammar, is that it allows the linguist to accuse the layman of using wrong sentences. However, due to the ratio of layman versus linguists, the linguists sometimes have to give in. This phenomenon gave rise to the so-called exceptions. In spite of being forced upon them by the layman, exceptions turned out to be a major weapon for the linguists: exceptions make a grammar so complicated that almost no-one has any chance to learn it.

The troublesome side-effect of this advantage is that if the grammar is too complicated to learn, the number of linguists will drop. Therefore, similar to words, grammars evolve. A fine illustration of this evolution is given by Dutch. Not to long ago, Dutch used to have declinations, nowadays only rudiments are left.

As an aside, note that German still has declinations. Some people use this fact as evidence that the Dutch are more advanced than the Germans. However, Germans are known to use this fact as a proof for the hypothesis that the Germans

are smarter than the Dutch. Clearly, this last viewpoint is nonsense as one cannot prove a hypothesis; because than it would be a theorem rather than a hypothesis. Moreover, we are better in soccer than the Germans[9] and therefore we are smarter.[3]

In the previous section, we proved that all words in any contemporary language can be mapped to (sets of) words in the first language. Now, we have seen that grammars also evolve, so a natural question is to ask whether all grammars descend from some sort of first grammar. A major difference with the previous section is, however, that grammars were made up. Still, we will see that our conjecture is again true.

This time, we have to turn to linguists. Similar to biologists, linguists can be partitioned into two populations, viz., the creationists and the Chomskians.

Again, for creationists the world is relatively simple. Everyone spoke the same language until the great Babylonian disaster. As the Babylonians certainly came after the Neanderthalers, there is a first grammar.[4]

For the Chomskians, life is a bit more difficult, but not too difficult for us. Briefly, the Chomskians know that the notion of grammar is encoded in your genes. All natural grammars are learned by adapting this built-in grammar.

As an aside, let me tell you that Lambert told me once of a Gedanken-experiment based on this fact. He is convinced that he can define an artificial language, whose grammar is weird enough so that no child has a chance in Hell to learn to speak this language properly. Clearly, the only way to check this claim, is to design such a language, and let a baby grow up in an environment in which all people only speak this language (of course, they are assisted by a computer). Such an experiment might seem to be inhumane. But hey, who cares, there are enough of those little bastards around.[5]

Anyhow, as both groups of linguists adhere to the same principle, we have the following universal truth:

LEMMA 2: All grammars descend from a common first grammar.

Again, we have a particularly interesting consequence:

COROLLARY 3: Dutch grammar descends from the first grammar.

It is worthwhile to investigate this first grammar a bit deeper. We have seen that grammars evolve by getting simpler and simpler. Hence, if we construct the category $Grammar$ of all grammars, we have that the first grammar is the most difficult among all. Hence, we have:

[3] Those of you who don't agree with this proof (mostly Germans I suspect), are clearly not very smart. So, your disagreement gives an alternative proof of my claim.

[4] To those of you who would like to ask why I didn't use this argument in the previous section, I would like to say that any great novelist (like your humble author) knows that one cannot use the same (sub)plot twice in the same story.

[5] If you think this is cruel, you have probably never attended an amoeba fight show [6].

LEMMA 3: The first grammar is the final object in the category $Grammar$.

For those of you who wonder what this first grammar looks like, we will briefly describe it. Compare German to Dutch, it is easy to see that there are many German sentences which are ungrammatical if they are considered as a Dutch sentence. So, the more complicated a grammar is, the more sequences are sentences. Hence, in the first grammar, all sequences were sentences.

Some dilletants try to minimise this result by saying: "big deal those primitives could only say *grunt*, so sentences looked like *grunt, pause, grunt grunt* ..". Of course, they can only make noise, their ancestors (at least those that are also mine) were smarter.

Others like to object that a grammar which allows all sequences is certainly easier than one that forbids some. My rebuttal is: "sure,;,?;""you you example blockhead this this case not shows is example you the".

Anyhow, if we juxtapose the vocabularies of the previous sections with the grammars of this section, we get all languages. Categorically, we get the category $Lang$, with the distinct object $FirstLanguage$, which is actually, the first language ever spoken. Combining the results of this section with those of the previous section, we get the following seminal result:

THEOREM 1: $FirstLanguage$ is both initial and terminal in $Lang$.

It is a well-known fact from category theory that categories in which initial objects are also terminal, all objects are isomorphic. So, the devastating consequence of our theorem is:

COROLLARY 4: All languages are equivalent to Dutch.

This means that all languages can be translated exactly into Dutch and vice versa; including the first language. This result will prove to be of utmost importance later in this paper. We conclude this section by observing modestly that this corollary is as far as the author knows the first scientific rebuttal to the Shapir-Worff Hypothesis. Ain't I brilliant.

4 BMF

Imagine a piece of paper, and an army of ants soaked in ink. If the ants choose to wander over the paper, the paper ends up resembling a page in a book. At least to an illiterate it does. Similarly, a page from a mathematics book is completely unintelligible to the layman. Actually, this is rather pleasant, as it stimulates funding.

In computer science, the situation is completely different, every idiot with a home-computer thinks he[6] is God's gift to programming. The upshot is that

[6] Yes, it are almost invariably man

the world is overflowing with would-be programmers, who release bug-pestered programs, miss deadlines, triple budgets and so on and so on.

There are two ways to correct this situation. One would be to devise a language in which even a monkey could write bug-free programs. The other is to turn programming in a mathematics-like science. Lambert tries both options. On the one hand he developed ABC [3], on the other he invented *Squiggol*; more aptly named BMF, from Bird Meertens Formalism.

ABC is considered a success by a lot of people as it can be used to teach school children to program. However, on a test with monkeys it failed misserably. The little devils kept typing pages of Shakespeare instead of programs [8]. Hence, we will limit our attention to BMF.

Like any part of mathematics, BMF starts with the definition of a collection of curious symbols (hence the name squiggol), such as:

$$\begin{array}{ccc} \times_A & \uparrow_H & \uparrow_{\#\cdot\pi_1}/ \\ S(\downarrow_B) & \mathcal{S} & \not\vdash_G \\ \lhd & \downarrow_f/ & S(\downarrow_f)C \\ \circledless & \circledgtr & \uparrow_J/ \end{array}$$

The immediate advantage of these symbols should be clear: their meaning is obscure. Moreover, the collection is not fixed, new symbols are constantly invented and old symbols are given new meanings. Thus outsiders are kept outside.

As usual in mathematics, squiggolists like to write as little as possible, therefore, brackets are omitted. However, this could give a well-educated mathematician some chance to understand a paper. To solve this problem, the squiggolists are separated into two groups. The first group (headed by Richard Bird) believes that operator symbols associate to the left, while the other group (headed by Lambert) adheres to association to the right.

This suggests a rule of thumb: "association is to the side the author prefers to drive on". But as in any religion, there is no logic involved[7]. Only those who know the author intimately have a clue to which church he adheres.

Anyhow, besides symbols and rules for the formation of correct sentences, BMF comes with a set of *laws*, which allows you to derive new sentences from old sentences. Perhaps the most renowned law is the so-called *promotion law*. The importance of this law follows from the observation that many squiggolists are OIO's who would like to get their PhD.

The results of the BMFers are incredible. For many years programmers around the world have tried to find palindromes. In fact, this problem has been called more important than $\mathcal{P} \neq \mathcal{NP}$. In BMF this is trivial, as is proven by the following simple algorithm:

$$\ll \cdot \not\uparrow_{\odot} i \cdot (\not\prec \omega)$$

[7] It is even less logical than the choice on which side to drive

where

$$i = ((\gg \epsilon) \propto [\,], \epsilon)$$

and

$$(\phi, \psi) \odot \alpha = ((\phi \diamond \alpha) \uparrow_{\#\cdot\bowtie} ((\gg (\psi \otimes \alpha)) \propto [\,])), \psi \otimes \alpha)\ .$$

with

$$\begin{array}{ll}
(\zeta, (\phi, \psi)) \otimes \alpha = & \\
(\zeta, (\mathsf{lead}\ \phi, [\mathsf{last}\ \phi] +\!\!+ \psi +\!\!+ [\alpha])) & \text{if } \phi \neq [\,] \wedge \alpha = \mathsf{last}\ \phi \\
(\zeta_1, (\phi +\!\!+ [\mathsf{head}\ \psi] +\!\!+ \upsilon, \mu)) \oplus \alpha & \text{if } (\phi = [\,] \vee \alpha \neq \mathsf{last}\ \phi) \wedge \psi \neq [\,] \\
(\zeta +\!\!+ [(\#\ \phi, 0)], (\phi, [\alpha])) & \text{otherwise}\ ,
\end{array}$$

where

$$\begin{array}{l}
(\upsilon, \mu) = ((\#\ \psi - \tau - 1) \rightharpoonup (\mathsf{tail}\ \psi), (\#\ \psi - \tau - 1) \leftharpoonup (\mathsf{tail}\ \psi)) \\
\tau = \gg(\mathsf{cut}'\ (\phi, \psi))\ /\!\gg ((\leq_{\#}\ \phi) \cdot \ll) \triangleleft \zeta \\
\zeta_1 = \zeta +\!\!+ [(\#\ \phi, \#\ \psi)] +\!\!+ (\mathsf{rev} *(\mathsf{mir}'\ (\#\ \phi, \#\ \psi))\ (>_{\mathsf{ctr}'}\ (\phi, \mu)) \triangleleft \zeta) \\
(\mathsf{cut}'\ (\upsilon, \mu))\ (\phi, \psi) \ = \ \begin{array}{ll} (\phi, \psi) & \text{if } \upsilon < \phi \\ (\upsilon, \psi - 2 \times (\upsilon - \phi)) & \text{otherwise} \end{array} \\
(\mathsf{mir}'\ (\upsilon, \mu))\ (\phi, \psi) = (\phi + 2 \times (\mathsf{ctr}'\ (\upsilon, \mu) - \mathsf{ctr}'\ (\phi, \psi)), \psi) \\
\mathsf{ctr}'\ (\phi, \psi) = \phi + \frac{1}{2}\psi
\end{array}$$

Correctness is of course trivial. More important, however, is the observation that almost no-one is able to read or write such beautiful algorithms.

So, it is like mathematics, but is it also scientific? There are two well-known rules to decide this question:

1. Johan's rule: There should be an abundance of greek symbols.

2. The Polish Rule: Indices should be nested at least two levels deep.

Obviously, BMF fullfills the first rule. However, it fails the second: indices are only one level deep. However, a recent result by Netty van Gasteren[1], applied reversedly, gives arbitrary deep indices. Therefore, we have the following result:

LEMMA 3: BMF is a mathematics-like science.

The success of BMF can easily explained if one cbserves that it is category theory in disguise, as we have:

- Data types are initial algebras, i.e. (least) fixed points of recursive endofunctors.

- Specifications are natural transformations.

- Derivation rules are categorical equations, such as provided by adjunctions, Grothendiek constructions, fibrations and the like.

Now, category theory is best characterised as general abstract nonsense, hence, the enormous incomprehensibility, and thus the success of BMF.

5 Crosswords and Cryptograms

Crossword-puzzles are made for people who indulge in obscure facts. Example questions are:

1. The 47th Hsuiang-dynasty emperor (47 letters).
2. The medieval name of the smallest river in Chile (12.5 letters).
3. The favourite prey of Dodos (0 letters)
4. A tree often found in forests (5 letters).

However, after you have solved around 2334 of those puzzles, you have solved them all. Hence, the invention of the Cryptogram. Basically, the collection of answers to the questions of a cryptogram is the same collection of boring facts. There is, however, a major difference between ordinary crosswords and cryptograms. The major difficulty in a cryptogram is to find out what the question is.

For example, in a crossword, a question could be: "river in Italy (2 letters)". The answer is obviously the river Po. The same question would be given in a cryptogram as: "Dutch toilet which is flushed with Italian water (2 letters)". The clue is that "po" is a Dutch kind of toilet.

Similarly, the question: "The Italians think he is still wet behind the ears (four letters)" would refer to yours truly.

A more difficult question is: "His brilliant ideas are invariably on small areas (6 letters)". As above, the question should be split into two parts, viz., "his" and "brilliant ideas are invariably on small areas". The first part indicates that the answer is the name of a male person; which person is encoded in the second part.

In Dutch one says:"There goes me a little light on", when one has a good idea. A little light is then roughly one lumen. Similarly, a small area is 1 cm^2. Hence, brilliant ideas on small areas are $1\,lumen$ on $1\,cm^2$, i.e., $1\,lumen/cm^2$. Curiously, the scientific name of this unit is the *lambert*. As this is also a male persons name, the answer is lambert.

We have already seen that Lambert loves crazy facts. As he solved 2335 crossword puzzles at the age five, he loves cryptograms. More specifically, he loves the cryptograms in the Saturday edition of the NRC-Handelsblad.

You might ask, why this preference? Well, some questions are considered to be more genuine cryptogrammatical than others. Take for example our second question from above, both parts of the question hint at the solution arno. However, it is not true that I am still wet behind the ears.

The third question is much better, as Lambert's best ideas are always on small areas, such as BMF. So, the answer to your question is simply that the

NRC-cryptograms are better. Moreover, they are safe for the environment and clean your dirtiest clothes at only 40 degrees.

The good questions have a very regular structure, which induces their solution. In fact, the puzzle-linguist Prof. dr. H Verkuijl wrote a book on cryptogrammars [2]. However, now we are close to revealing our secret, hence, it is time to start a new section.

6 The secret revealed

In the previous section, we already hinted at the final piece of the puzzle: good cryptograms can be described by grammars, cryptogrammars. If you have read [2], you know that this grammar is built around diagrams: *diagrams in the language category !* It is a straightforward exercise to prove that these diagrams define natural transformations. The key observation in the proof is that the diagram polymorphically describes the structure of the question. So, Reynolds abstraction theorem[10] gives the required naturality.

Solving a cryptogram amounts, naturely, to the gradual transformation of such diagrams to their solution, using obvious equations. Hence, we have:

- Dutch is initial in the language category.
- Good cryptograms are natural transformations.
- Good cryptograms are solved using equations.

In other words:

REVELATION 1: *Solving the NRC-Handelsblad Saturday edition cryptogram is an exercise in Squiggol!*

But there is more. If you would dig in Lambert's *bin* directory, you would find a little program called *kruis* (cross). If you would try this program out, you would find that it helps you to solve crosswords and cryptograms. Specify the number of letters, and the letters you already have, and *kruis* will give you all Dutch words that fit the pattern.

Clearly, this program is not very intelligent, as it does not consider the question. But this program dates from the seventies, that is pre-BMF. It is well-known that Lambert is not very satisfied with *kruis*. His longstanding complaint used to be that *kruis* generates too many possibilities.

However, ever since the start of the STOP project, the flow of complaints dried out. Is this a coincidence? No, actually, we have:

REVELATION 2: *The motivation for the development of abstracto (as it was called) and the subsequent start of the STOP project were the frustrations caused by* kruis.

So, we have:

PRINCIPAL THEOREM OF SQUIGGOL: *The aim of BMF is to solve the NRC-Handelsblad Saturday edition cryptogram.*

7 Conclusions

What conclusions should we draw from our principle theorem? On the negative side we have:

1. Lambert uses governmental money to pursue private interest.
2. He lures able young scientists into the field of BMF, by making them believe that it advances the art of programming.
3. If his quest succeeds, cryptograms will be trivial and thus he robs many people from their favourite pass-time.

However, a scientist always solves those problems that are the most interesting to him. That is, every scientist simply does what pleases him the most. None gives a damn about society. Moreover, given the recent funding cuts, society does not give a damn about its scientists.

If those young scientists were not doing research, they would smash bus-stops, rob old ladies, or even become soccer-fans. Hence, he is doing society a favour by keeping them of the streets.

Finally, solving cryptograms is a waste of time. Doing research is far more valuable to society. So, if he succeeds, chances are that cryptogrammists turn to science. Personally, I would not be surprised if one of those people will find the cure for Aids.

As the negative points turned out to be positive after all, I hesitate to explain the positive points. Rather, I wish Lambert luck on his quest.

References

[1] A.J.M. van Gasteren. *On the shape of mathematical arguments.* PhD thesis, Technical University Eindhoven, 1988.

[2] Dr. H.J. Verschuyl. *Cryptogrammatica, het geheim van het cryptogram.* Kosmos, 1988.

[3] Leo Geurts, Lambert Meertens, and Steven Pemberton. *ABC Programmer's Handbook.* Prentice Hall, 1990.

[4] Herr Professor Doctor vom Grammatik. On sentences. *Proceedings of the Neanderthalic Academy of Sciences*, 2:12.5, -4500.

[5] I told you he wished to stay. *anonymous*. dummy.

[6] L. Meertens. Algorithmics—towards programming as a mathematical activity. In J.W. de Bakker, M. Hazewinkel, and J.K. Lenstra, editors, *Proceedings of the CWI Symposium on Mathematics and Computer Science*, volume 1 of *CWI Monographs*, pages 289–334. North–Holland, 1986.

[7] L.G.L.T. Meestens. *The perfect housewife*. Sexist Books Inc., 1990.

[8] Milton. On monkeys typing shakespeare. *Proceedings of the American Academy of Urban Legends*, 2565:14–151515, 1876.

[9] Pick any European or World championship you like. *we*. were better.

[10] John C. Reynolds. Types, abstraction and polymorphism. In *Information Processing 83*, pages 513–523, 1983.

Esprit et formalisation

Michel Sintzoff

Université de Louvain

La vie scientifique, et peut-être aussi non scientifique, de Lambert peut se voir comme une incarnation fascinante de la tension permanente entre la lettre et l'esprit: comme l'on sait, *"La lettre tue, mais l'esprit vivifie"* et *"L'esprit est fort, mais la chair est faible."* Une version moderne de cette dernière maxime, obtenue par traductions automatiques successives à travers l'anglais et le russe, est bien connue: *"La vodka est bonne, mais la viande est avariée."* Ci-dessous, je vais présenter une vue personnelle, donc partielle et partiale, de la manière dont Lambert a vécu cette interaction entre lettre et esprit, entre concret et abstrait, entre implémentation et spécification.

Mes premières rencontres avec Lambert remontent à l'époque turbulente des brouillons successifs du rapport définissant Algol68, dans les années 1967-1968.

La conception globale du langage était assurée par le groupe 2.1 de l'IFIP, sous une impulsion importante d'Aad van Wijngaarden; il s'agissait de combiner les propriétés d'expressivité, d'orthogonalité, de généralité, d'efficacité et de typage strict. Ces bonnes intentions, en quelque sorte l'"esprit" du langage, devaient être incarnées dans des "lettres" ou une "chair" qui seraient la définition rigoureuse et la description formelle d'Algol68. Préparer cette chair était la tâche de la "cuisine" installée à l'étage supérieur du bâtiment du Mathematisch Centrum. Celui-ci était directement voisin d'une célèbre brasserie amstellodamoise, ce qui entourait le travail d'un esprit-de-bière trop tentant. Les W-grammaires de van Wijngaarden suscitent ainsi chez certains une odeur de bonne bière; Aad aurait sans doute préféré qu'elles rappellent plutôt du vin fin. Les cuisiniers de la description étaient Aad van Wijngaarden, bien sûr, et Barry Mailloux, John Peck et Kees Koster; Aad et Barry, hélas décédés depuis, auraient évoqué cette époque bien mieux que moi. Lambert servait d'aide-cuistot; son travail était circonscrit non seulement par la conception globale du langage contrôlée par le groupe 2.1, mais aussi par les choix techniques de description décidés par les cuisiniers en titre. Pourtant, déjà dans ce contexte bien délimité, Lambert faisait preuve de sa créativité et de sa liberté d'esprit: par des remarques et des suggestions apparemment innocentes et locales utilisées comme épices syntaxiques ou sémantiques, il sut améliorer avec bonheur le goût et la teneur de plats délicats. Son activité pouvait ainsi se comparer à la préparation attentive des *uitsmijter* du midi, des *rijstafel* du soir, et du fameux buffet chez Aad pour fêter l'édition finale du rapport de définition.

Lambert et moi, chercheurs innocents et débutants à l'époque, avons assisté avec surprise et tristesse aux conflits scientifiques et politiques qui ont déchiré le groupe 2.1; pour rappel, le langage Pascal et les travaux du groupe 2.3 fournissent de bons exemples des voies alternatives préférées par plusieurs chercheurs de valeur. Les recherches ultérieures de Lambert sur B et Squiggol montrent cependant que le choix entre ces voies ne doit pas nécessairement séparer les personnes: Lambert ne semble pas avoir mené de

guerre sanglante contre lui-même.

C'est dans la phase de révision d'Algol68 que j'ai travaillé de manière plus intensive avec Lambert et que j'ai donc pu apprécier encore mieux ses qualités intellectuelles et psychologiques.

Le travail de révision était soumis à une contrainte précise et décidée par le groupe 2.1, à savoir que les révisions au langage proprement dit devaient rester absolument minimales. Les cuisiniers avaient pour tâche de mettre en oeuvre cet esprit restreint de la révision dans une révision large de la lettre: autrement dit, notre travail consistait à améliorer la description le mieux possible par une nouvelle cuisine, tout en incluant les modifications minimes au langage approuvées par le groupe 2.1. Les changements apportés à la syntaxe ont consisté principalement en la construction de W-grammaires formalisant complètement les conditions de contexte et la vérification des types. La sémantique a été reconstruite en vue de remplacer la technique par manipulations de textes, telle que la règle de copie, en une approche par fonctions d'états dirigées par les structures syntaxiques; plus tard, Plotkin formalisera cette approche dans la sémantique opérationnelle structurée. Quatre nouveaux cuisiniers vinrent renforcer le quatuor initial; il s'agissait de Lambert, Charles Lindsey, R. Fisker (jeune collègue du précédent), et moi. Le travail fut mené surtout par les nouveaux cuisiniers, avec une participation active des deux anciens Barry Mailloux et John Peck.

Les contributions de Lambert furent centrales, aussi bien dans la syntaxe que dans la sémantique. Sa longue familiarité avec les grammaires paramétrées, dont les W-grammaires de van Wijngaarden constituent une des classes, lui permit de mettre au point la brillante méthode de formalisation de prédicats par des ensembles structurés de règles de production; il s'agissait en quelque sorte d'utiliser les W-grammaires comme grammaires logiques, tout comme Prolog peut servir de véhicule de programmation logique; le rapport entre W-grammaires et Prolog a d'ailleurs été rappelé par Alain Colmerauer. Cette expression des prédicats rendit possible une formalisation complète de la vérification de types; pour mémoire, il s'agit non seulement de types simples au sens du lambda-calcul classique, mais aussi de types produit et somme, et de types récursifs. La description obtenue constitue encore à ce jour la seule définition standard de langage de programmation où la vérification des types soit formalisée strictement et complètement, sinon clairement. Notons que les types polymorphiques, la synthèse automatique de types et les types dépendants inspirés de la logique constructive constituent une évolution naturelle des types d'Algol68; Per Martin-Loef l'a noté.

Lambert aida ainsi à formaliser les règles d'équivalence structurelle des types récursifs; Nobuo Yoneda, Georg Tseytin et Don Knuth avaient identifié auparavant les problèmes posés par cette équivalence. Dans les premières moutures des W-règles associées, des incomplétudes et des incorrections furent hélas détectées, souvent par le regard inquisiteur de Lambert. Le besoin est donc vite apparu de prouver la correction de la formalisation en W-règles par rapport à la définition mathématique de l'équivalence. De nouveau, la rigueur et la clarté Lambertoises permirent d'élaguer les broussailles de raisonnements. La preuve résultante, rédigée en anglais, fait partie de "commentaires" dans le rapport de définition. Nous aurions pu évidemment formaliser cette preuve, par exemple en une autre W-grammaire, puis prouver la correction de celle-ci, et ainsi de suite jusqu'au point fixe. Grâce au bon sens de Lambert, nous nous sommes heureusement calmés. Remarquons à ce propos que c'est assez récemment que les "programmeurs

logiques" ont eux aussi constaté le besoin de prouver la correction de leurs programmes logiques.

Le volet sémantique de la définition bénéficia de ce que j'appellerais volontiers *"le rasoir mental de Lambert"*, par lequel la sémantique fut ciselée comme un diamant. Lambert s'attaqua avec force à une simplication, une purification et une structuration maximales de la sémantique. Ce travail concerna d'une part la théorie axiomatique des ensembles, des relations et des actions sémantiques fondamentaux, et d'autre part la définition et la composition des fonctions sémantiques. Le défi consistait ici à structurer au mieux une description sémantique d'un langage qui ne pouvait être modifié que le moins possible et dont la richesse expressive n'était pas négligeable, et même excessive pour certains. Lambert projetait le phare de son intelligence partout, n'évitant aucun coin. Ses collègues cuisiniers ont du ainsi passer par les fourches caudines d'un esprit implacable, qui heureusement proposait en même temps des solutions élégantes aux conflits identifiés.

Ceci me conduit fort naturellement à évoquer avec grand plaisir l'agrément personnel du travail avec Lambert. Il a toujours été un exemple vivant de la nécessité et la possibilité de distinguer entre désaccords techniques et conflits personnels. Il pouvait parfaitement mettre un doigt cruel sur une faute scientifique grave, tout en gardant un sourire désarmant exprimant une sorte de quasi-complicité dans l'erreur. C'est donc avec confiance et plaisir que je me suis toujours addressé à Lambert pour lui demander avis et conseil, entre autres à propos de problèmes neufs encore mal dégrossis; cette combinaison de compétence technique et de sympathie personnelle n'est que trop rare.

La phase Algol68 de Lambert peut ainsi se caractériser comme une lutte entre l'esprit et la lettre, c'est-à-dire entre l'entendement et la formalisation. Il s'agit bien de l'essence du travail de tout informaticien: nous n'avons que trop tendance à oublier que les programmes sont les objets les plus formels au monde, puisqu'ils sont traités par des automates matériels, alors que les problèmes qu'ils résolvent concernent de nombreuses sphères de l'activité humaine. Il est donc crucial d'une part de bien construire le système d'expression, qu'il s'agisse d'un système formel ou d'un langage formel, et d'autre part de concevoir une formalisation claire et effective de la modélisation abstraite et de la résolution constructive de problèmes. Lambert ne peut renier ses contributions précieuses ni à des systèmes formels tels que les W-grammaires, le langage B ou les notations de Bird-Meertens, ni à des méthodes de formalisation telles que les prédicats syntaxiques, les preuves de programmes ou les transformations algébriques d'algorithmes.

Il est remarquable que cette variété apparente d'activités scientifiques (la lettre) soit sous-tendue par des invariants conceptuels évidents (l'esprit), par exemple la correction, la clarté et l'économie de formalisations. Ces invariants correspondent à des difficultés incontournables que d'autres chercheurs découvrent tôt ou tard. Certains ajouteront qu'un problème ouvert reste le mariage harmonieux entre puissance d'expression et accessibilité intellectuelle: qu'est-il plus facile de lire, des W-grammaires, du B ou du Squiggol? Et qu'est-il plus facile de manipuler ou de prouver formellement? D'autres diront que les recherches de Lambert manifestent aussi une convergence globale: les preuves informelles de correction de grammaires logiques concrètes sont devenues des lois algébriques de construction d'expressions fonctionnelles abstraites. Quelle sera alors le point fixe ou objet limite de cette convergence? Peut-être une belle structure

chaotique?

Pour terminer, je tiens à insister sur un autre aspect séduisant de l'approche Meertensienne en informatique, à savoir une grande liberté d'esprit. Lambert a toujours insisté sur le fait que la "lettre" des théories, des formalismes et des techniques, doit constamment rester subordonnée à l'"esprit" que donnent les idées et les intuitions. Il est donc non seulement légitime mais aussi nécessaire de construire des systèmes formels de calcul ou de raisonnement d'abord par des *Gedanken Experimente,* avant de déclencher l'artillerie logico-mathématique indispensable; les expériences formelles elles-mêmes suscitent de nouvelles intuitions qui à leur tour sont formalisables. C'est en ce sens que Lambert est sans doute un véritable informaticien, passant et repassant de l'esprit à la formalisation, à tâtons et sans trêve, comme les habitants d'une variante récursive de la caverne de Platon.

En un mot, je dirai donc de Lambert: *"La bière est mousseuse, et les vivres sont variés."*

Op de Titelloze Mens

"Nee," zei Tinnef. "Ik doe het niet meer. Ik heb geen zin meer in die stukjes waarin we altijd de lollige poes moeten uithangen en zo nodig gevat moeten zijn."

"Ik ook niet," zei Daantje.

"Toe nou," zei ik. "Het is voor het liber amicorum voor Lambert."

"Wie is Lambert?" vroeg Daantje.

"Nee," zei Tinnef. "We staan al in talloze liber amicora. Ik pas."

"Libri amicorum," zei ik. "Lambert houdt niet van taalfouten. Dan gaan zijn tenen krullen."

"Hij is wel bezorgd om zijn gezondheid," zei Tinnef. "Bij het minste of geringste gaan zijn tenen krullen en hij eet alleen zelfgebakken oliebollen, want van die andere krijgt hij rheuma, zegt hij. Maar ondertussen wel uit de vuilnisvaten eten om acht uur 's ochtends."

"Hij at niet," wierp ik tegen. "Hij kwam net terug van het MC en zag een mooi presenteerblaadje tussen het vuilnis. Daar heeft de afdeling nog altijd veel profijt van."

"Wie is Lambert nou?" zeurde Daantje.

"Lambert is die jongen met een vergiet op zijn hoofd."

"Wat voor kleur vergiet?" wilde ze weten.

"Hij is net een kameleon, hij past zich aan aan de kleur van het feest. Volgens mij heeft hij thuis een immense collectie vergieten."

"Waarom geef je hem dat gedicht niet Op de Titelloze Mens," stelde Tinnef voor.

"Dat heb ik hem jaren geleden al gegeven. En als hij iets één keer gelezen heeft, kent hij het uit zijn hoofd. Wie kent altijd alle woorden van alle Sinterklaasliedjes, als we in Italiaanse restaurants gaan zingen? En de volledige tekst van Ja dan gaan we naar de speeltuin? En waarschijnlijk ook nog van achter naar voor, voor het geval dat het een satanische boodschap bevat."

"Kent hij de woorden van Stop al je zorgen in je plunjezak?" vroeg Daantje. "Die zoek ik al een tijd." Ze begon te zingen: "Stop al je zorgen in je plunjezak en ploem ploem ploem. Het maakt me wild dat ik niet op dat ploem kan komen."

"Bah!" zei Tinnef vol minachting. "Ik herinner me hoe hij vroeger na Praathuis de MC-ers afmarcheerde onder het zingen van socialistische strijdliederen. Kinderliedjes, bah! Maar de oorspronkelijke tekst van Bloem kende hij toch mooi niet."

"Verdomd," zei ik. "Dan kan ik alleen maar hopen dat hij die ondertussen nog niet heeft opgezocht."

"Dat is dan afgehandeld," besliste ze. "Maar dit was de allerlaatste keer." Ze stond op en liep langzaam de kamer uit, terwijl ze somber begon te brommen: "Broeders verheft u ter vrijheid..."

"Broeders de ploem ploem ploem," viel Daantje in en marcheerde mee.

De Dapperstraat

Natuur is voor tevredenen of legen.
En dan: wat is natuur nog in dit land?
Een stukje bos, ter grootte van een krant,
Een heuvel met wat villaatjes ertegen.

Geef mij de grauwe, stedelijke wegen,
De in kaden vastgeklonken waterkant,
De wolken, nooit zo schoon dan als ze, omrand
Door zolderramen, langs de lucht bewegen.

Alles is veel voor wie niet veel verwacht.
Het leven houdt zijn wonderen verborgen
Tot het ze, opeens, toont in hun hoge staat.

Dit heb ik bij mijzelve overdacht,
Verregend, op een miezerige morgen,
Domweg gelukkig, in de Dapperstraat.

J.C. Bloem

Diep ongelukkig om mijn dwaze staat

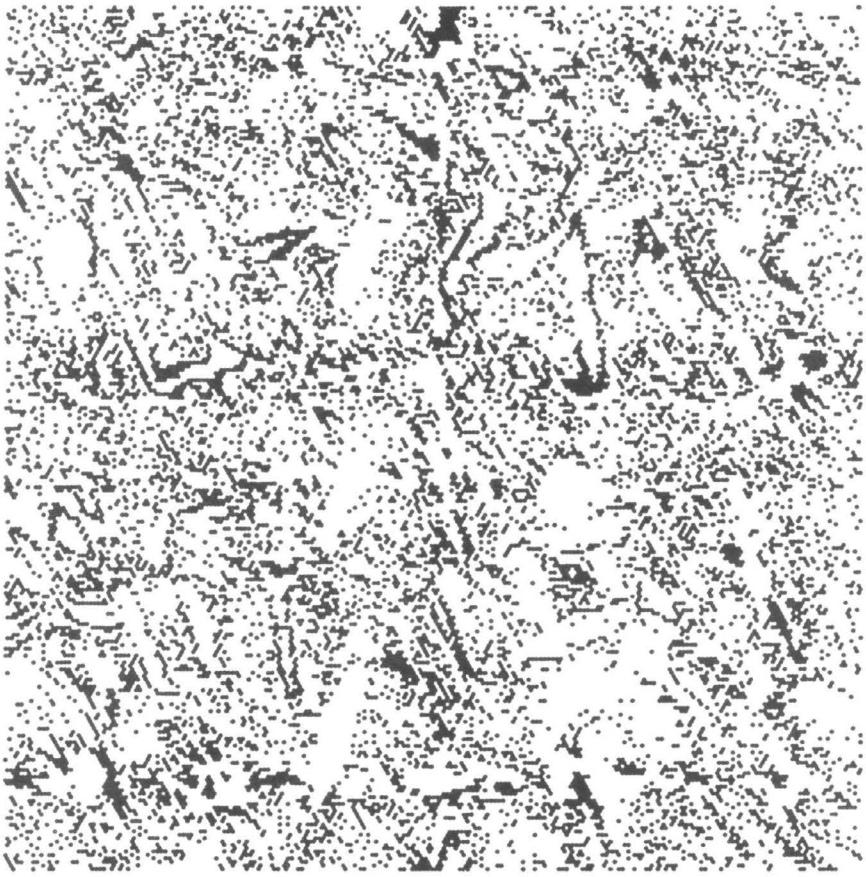
Drs. is voor de middelmaat en legen.
Want wat is doctorandus in dit land?
Een directeur ter hoogte van een krant,
Een socioloog om hersenen verlegen.

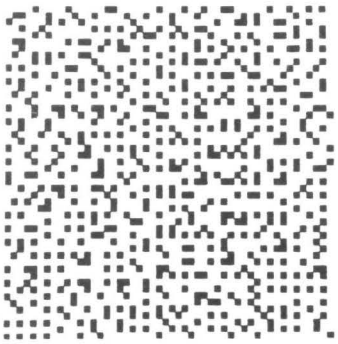
Geef mij die flitsend geniale zegen,
De man die met verbeelding en verstand,
Maar volmaakt titelloos en onomrand
Door d-r-s gaat over 's werelds wegen.

Het leven wordt door academici verkracht.
Voor doctorandi blijven wonderen verborgen
Al liggen ze voor 't oprapen op straat.

Dit heb ik, bij mijzelve in de nacht,
Geschreven, in plaats van mij te worgen
Met de bul die mij verleent mijn graad.

Joke Sterringa





Inmiddels wat meer
beheersing van structuur.
1968 - 1998
Jij ook?
Hartelijke groet
Peter Struycken.

De Boomhiërarchie

Doaitse Swierstra

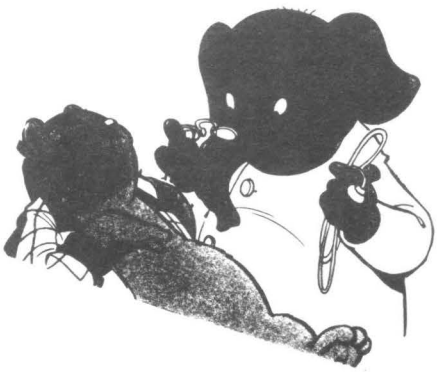
De hoge boom leek het zo fijn
een echte rozeboom te zijn
dus moest hij eerst leren
zijn blaadjes te laten associëren
want "cut en paste" dat doet zo'n pijn.

De lange lijst zat flink te kniezen
tussen *cons* en *snoc* kon hij niet kiezen
dus vormde hij zich maar om tot zak
om zo met 't allergrootst gemak
zijn kop en staart voorgoed te verliezen.

De zak voelde zich wel wat bezwaard
te veel elementen had hij vergaard
de dubbele, die hadden pech
die gooide hij maar heel ver weg
en slechts de enkeling, die werd gespaard.

De verzameling is pas echt content
het meest met wetten is hij verwend
en al komt hij dan ook achteraan
we laten hém gewoon weer gaan
want verder is niets *idem potent*.

De Uitvinder



Ten huize van de heer Lambertus G. Bommel heerst een beklemmende stilte. De heer Bommel is in diep gepeins verzonken en butler Joost ligt overspannen in bed. De dokter heeft hem zojuist een kalmeringsspuitje gegeven. Dit behoeft enige uitleg. De heer Bommel is uitvinder, niet van beroep natuurlijk, want hij is een heer van stand. Hij is zogezegd een heer-uitvinder, een uitvinder van adelijke afkomst die zich heeft toegelegd op het onderzoeken van het onbekende om de mensheid te dienen. Wat doet een uitvinder zoal? Uitvinden natuurlijk! Zo is de heer Bommel bekend als de uitvinder van de interpolerende typemachine, van de grootste-kwadraten-methode en van de Bommel-viool. De Bommel-viool is niet meer weg te denken uit de moderne jazzmuziek en ook de betekenis van de grootste-kwadraten-methode voor de Bedrijfseconomie behoeft geen nadere uitleg. En iedereen weet dat de interpolerende typemachine gebruikt wordt om televisieprogramma's te ondertitelen, misschien niet iedereen dat deze machine ook veel gebruikt wordt door mensen met leesblindheid.

Onze uitvinder is dus van alle markten thuis, maar de laatste 25 jaar heeft hij zich voornamelijk bezig gehouden met programmeerbaar rekentuig en alles wat daar mee van doen heeft. In de Rekenwetenschap (ook wel Informatica genoemd) heeft hij dusdanig baanbrekend werk verricht dat hij wel de Peetvader van de Rekenwetenschap wordt genoemd. Als van geen ander is zijn naam verbonden met bijna alle gebieden binnen de Rekenwetenschap. Samen met een andere heer van stand heeft hij een specificatietaal voor grammatica's ontworpen, het wel bekende Bommel-Naur formalisme. In de gegevensbankhoek heeft een door hem ontwikkelde opslagstructuur, de B-boom, tot een ware revolutie geleid. Ook is hij de eerste geweest die het resolutieprincipe geïntroduceerd heeft in een programmeertaal; de taal BROLOG is daar het resultaat van.
Waarom is iemand uitvinder? Omdat hij het leuk vindtom het onbekende te onderzoeken! Een uitvinder begint als onderzoeker. Hij doet een of ander experiment en kan dan zo gefascineerd raken dat hij er niet meer mee kan stoppen. Als hij pech heeft belandt hij op dood spoor, als hij geluk heeft ontdekt hij iets nuttig-nieuws. Op het moment dat een onderzoeker iets nuttig-nieuws ontdekt is hij uitvinder geworden. Het ik-heb-het-gevonden-gevoel dat zich dan van een onderzoeker meester maakt is onbeschrijflijk. Voor heel even bevindt de onderzoeker zich in uiterste verrukking over zijn zojuist ter wereld gekomen geesteskind.

De heer Bommel houdt zich echter niet alleen bezig met experimenteren, maar ook met lezen, want hij is graag op de hoogte. Elke dag verslindt hij een aanzienlijk aantal boeken, ter lering en ter vermaak. Is hij als uitvinder een levende legende, zijn kennis is spreekwoordelijk, getuige het vaderlands gezegde 'zo wijs als Lambertus'. Ook heeft hij enkele boeken geproduceerd, waarvan 'De Dikke van Bommel', de meest frequent geraadpleegde woordenlijst der Nederlandse taal, ongetwijfeld het bekendste is. Ik moet mij even verontschuldigen dat ik zover ben afgedwaald. Ik had beloofd om U enige inlichtingen te verschaffen omtrent de toestand van butler Joost. Zoals reeds gezegd is de heer Lambertus G. Bommel iemand die van experimenteren houdt. Dit experimenteren beperkt zich niet tot de Rekenwetenschap. Ook andere terreinen zijn een gewild object van Bommels experimenteerdrang. Een van deze terreinen is de kookkunst. Alhoewel de heer Bommel personeel in dienst heeft om voor de inwendige mens te zorgen, schrikt hij er niet voor terug om regelmatig zelf te experimenteren op culinair gebied. Het aantal slachtoffers van deze experimenten is beperkt gebleven tot de heer Bommel zelf. Van enkele andere experimenten valt dit helaas niet te zeggen. Zo is meester Mol al zijn haren kwijt geraakt na een mislukt experiment met een bliksemafleider en is butler Joost een hele tijd doof geweest als gevolg van een ontplofte radio.

Ook nu is Joost er weer niet helemaal ongeschonden van afgekomen. Hij heeft weliswaar niet aan fysiek geweld bloot gestaan, maar het laatste experiment van Bommel heeft de zenuwen van onze butler danig op de proef gesteld. Ik zal U vertellen waarom. Al enige tijd was Bommel in het diepste geheim bezig met de voorbereidingen van een experiment dat hij als het ultieme experiment zag. Dit ultieme experiment was gebaseerd op een idee van de wiskundige Turing, de zogenaamde Turing-test. Daartoe had Bommel een robot gebouwd die als twee druppels water op hem geleek. Hij had de robot zo geprogrammeerd dat deze schijnbaar dezelfde kennis en vaardigheden bezat als hijzelf. Als Joost niet in staat zou zijn om de robot van de echte Bommel te onderscheiden, dan beschouwde hij zijn ultieme experiment, dat een robot een heer van stand kon nabootsen, als geslaagd. Zulks geschiedde. Op zekere morgen activeerde Bommel de robot en verstopte zichzelf achter het gordijn. Toen Joost het ochtendblad gebracht had en enkele woorden met de Bommel-robot gewisseld had merkte hij op dat de heer Bommel een schorre stem had. Derhalve zou hij een glaasje warme melk met honing brengen. Het drankje wordt snel opgediend en al even snel opgedronken. Plotseling begint de robot heftig te schudden, Joost snelt toe en onder zijn ogen stort de Bommel-robot in elkaar. Joost probeert vanalles om Bommel weer tot bewustzijn te brengen, maar dat lukt niet. In paniek rent hij het huis uit, in volle vaart naar het huis van de dokter.

In huize Bommelstein heeft heer Bommel intussen weer de plaats ingenomen van de robot. Nu de eerste vreugde over het geslaagde experiment is weggeëbd, begint heer Bommel zich zorgen te maken over butler Joost. Als hij hoort dat er een auto aankomt, spoedt hij zich naar de deur. Uit de auto stappen de dokter, die hij sinds het ongeluk met de radio niet meer gezien heeft, en natuurlijk zijn trouwe butler. Maar voor Joost is het allemaal wat veel. Wanneer hij ontdekt dat heer L.G.B. springlevend in de deuropening staat, draait hij door en springt in de fontein. Onze arme butler krijgt een kalmeringsspuitje en wordt onder de wol gestopt. En heer Bommel krijgt van de dokter het advies om zijn experimenten te beperken tot pen en papier. Maar voor een heer van stand is dat geen enkel probleem.

Lambert, The Erdös Graph and the Beast

Paul M.B. Vitányi

C.W.I. and U.v.A., Amsterdam

Let him that hath understanding count the number of the beast: for it is the number of a man; and his number is six hundred three score and six.
St. John, Book of Revelations.

The number of the beast has been interpreted to stand for Pope Leo X (by Michael Stifel (1486 - 1567)). Consider LEO DECIMVS X, add L,D,C,I,V,X, discard the other letters because they do not denote a Roman numeral and 'M' because it denotes 'Mystery'. More sweepingly, 666 has been interpreted to symbolize the Pope of Rome in general (according to Napier); Martin Luther (according to Napier's contemporary the Jesuit Father Bongus S.J.); Kaiser Wilhelm (during World War 1); Adolf Hitler (during World War 2); and earlier the Roman Emperor Nero (because 666 is claimed to spell 'Nero' when expressed in the letter symbols of the Aramaic language, in which the Book of Revelations was originally written; cf. [3].)

However, it is obvious that the foretold beast, pursuing the mother church in the Apocalyps, is none other than LAMBERT MEERTENS. For bear with me. The position numbers of the constituent letters in the latin alphabet are: 12, 1, 13, 2, 5, 18, 20, 13, 5, 5, 18, 20, 5, 14, 19. Adding them yields 170. The name LAMBERT has seven letters, and LAMBERT MEERTENS consists of two words, so we add another 9 to obtain 179. We must multiply this by 4, since this crypto-catholic has four christian names. In any case, by now we have obtained 716. >From this we must subtract the value of the *only* roman numeral in LAMBERT MEERTENS, which does not stand for 'mystery', to obtain the stigma: 666.

The Erdös graph has been the subject of some papers [1,2] and certainly of many conversations, usually enlivened by the intake of appropriate liquids. In [2] it is conjectured that as time goes by the scrutinized graph will contain arbitrary large cliques. The inspiration for an example in [4], or more to the point, the incentive to write that paper at all, was the emergence of the Beast from the Graph. The purpose of this note is to examine this matter in some more detail, viz., to show how the assumption of arbitrary large cliques in the Erdös graph evokes the number of the Beast, that is 666, from the graph. The paper continues the age old tradition of finding contrived interpretations for the Number of the Beast.

Assume that we consider undirected graphs without multiple edges and self loops. The Erdös graph has the mathematicians (nowadays perhaps mathematicians and computer scientists) for its vertices. In its basic form, two vertices are joined by an edge if the corresponding mathematicians have coauthored a paper. A

clique in a graph is a subgraph in which each pair of vertices is connected by an edge. A clique, in the Erdös graph, with more than two vertices must contain at least one edge joining equal sexed vertices. Let (p_1,p_2) denote a *partition* of the p vertices of a clique K_p in p_1 *male* and p_2 *female* mathematicians. Let $\alpha(p_1,p_2)$ denote the *quotient* between the number of edges joining vertices of equal sex and the number of edges joining vertices of opposite sex, in the partition (p_1,p_2) of K_p. Let $M(K_p)$ be the *infemum* of $\alpha(p_1,p_2)$, for all partitions of vertices of K_p. The following equation is easily proven.

$$M(K_p) = \frac{p^2-2p+\text{remainder}(p/2)}{p^2-\text{remainder}(p/2)} \quad . \qquad *$$

It follows therefore that, for all $p>1$, we have $M(K_p)<1$. Moreover, that $M(K_p)$ is monotone increasing and $\lim_{p\to\infty} M(K_p) = 1$.

If we replace the arbitrary large clique assumption by the more pleasant one of arbitrary large cliques and all possible male-female ratios, for each $p>1$ there eventually will appear a clique in the graph in which the male-female ratio realizes $M(K_p)$. According to the equation *, for each positive constant ϵ there is a p_ϵ such that $1-\epsilon < M(K_p) < 1$. Since $M(K_p) \leqslant \alpha(p_1,p_2) \leqslant \infty$ we can, for each P, always find a partition (p_1,p_2) of p such that $\alpha(p_1,p_2)$ is closest to 1. (However, for most arguments, $\alpha(.) >> 1$.)

It is well known that, although the male/female ratio amongst mathematicians is vastly greater than 1, in the general human populace this ratio differs for various age groups, ethnic groups, and at different times for the same group, but is usually somewhere in between 100/100 and 112/100.

We consider cliques with $\alpha(p_1,p_2) = 1$. Calculation shows that this is the case for $p_1-p_2 = \sqrt{p_1+p_2}$. Setting the ratio male/female to the realistic circa 4% more males than females, that is, 108.062/100, we obtain $p_1-p_2 = 25.803$ and $p = 666$. This necessitates dividing one vertex into a partly male and partly female mathematician. We would like to avoid this*. Hence we allow $\alpha(p_1,p_2)$ to depart slightly from 1. It follows that

$$\alpha(p_1,p_2) = \frac{\binom{p_1}{2}+\binom{p_2}{2}}{\binom{p_1+p_2}{2}-\binom{p_1}{2}-\binom{p_2}{2}} \quad ,$$

which yields

$$p_1^2-2\alpha(p_1,p_2)p_1p_2+p_2^2-p_1-p_2 = 0.$$

Using $p_1 = 1.08125p_2$ we have that

$$\alpha(p_1,p_2) = 1.00305-0.96242/p_2 \quad .$$

Since $p_1/p_2 = 160/173$ and 173 is prime, p_2 must be a multiple of 160 to avoid mathematicians with sex chromosome deficiencies. So $p_2=160$ yields

* The case is not too far-fetched, and, anyway, the clique only needs one such case.

$\alpha(p_1,p_2)<0.997$; $p_2=320$ yields $\alpha(p_1,p_2)=1.0000425$ and $p_2=480$ yields $\alpha(p_1,p_2)>1.001$. Since $\alpha(.)$ is strictly increasing with p_2, the nearest we can get $\alpha(p_1,p_2)$ to 1 is by taking $p_2=320$ and $p_1=346$, again yielding the ominous clique K_{666}.

REFERENCES

1 Coffman, C., And what is your Erdös number? *American Mathematical Monthly* **79** (1969) 791.

2 Erdös, P., On the fundamental problem of mathematics, *American Mathematical Monthly* **79** *(1972) 149 - 150.*

3 Eves, H., *An Introduction to the History of Mathematics.* Holt, Rinehart and Winston, New York, 1967.

4 Vitányi, P.M.B., How well can a graph be *n*-colored? *Discrete Mathematics,* **34** (1981), 69 - 80.

GROPE THEORY FOUNDATIONS FOR SQUIGOL
Computer Science Press Release

David S. Wile, Research Professor at USC's Information Sciences Institute in Southern California, sends the following report:

Coinventor of 'Squigol,' Professor Lambert Meertens, recently announced completion of a breakthrough effort: the theoretical foundations for Squigol have been expressed in Grope Theory. Grope Theory is the little-known father of 'Chaos Theory,' recent darling of the popular scientific press. Grope theory describes how other theories arise; this involuted meta-theory thereby intercepts all other theories' claims to fundamentalism.

The basic operators of Grope Theory include: *represent*, *examine*, *decide*, *reject*, *rework*, *rethink*, and *drink*. As usual, we represent the theory after applying operator O to theory T as OT. Subtleties of the theory are represented via axioms such as:

$$\forall T \;\; T \neq \textbf{rethink} \;\; T$$

thereby differentiating 'rethinking' from 'examining.' (The latter is permitted to retain the original theory.) Unusual is the fact that all of the operators of Grope Theory are nondeterministic – some say, "especially drink." Hence the strongest claim that can be made for any operator, O, is embodied in the axiom:

$$\forall T \;\; ((OT) = OT) \vee (OT) \neq OT$$

Astoundingly, Grope Theorists are divided on whether the axiom should read:

$$\forall T \;\; ((OT) = OT) \wedge (OT) \neq OT$$

The following formulae are true in all *known* models of Grope Theory, but no one has been able to establish their validity:

$$\forall T \;\; \exists T_1, \ldots, T_n, n > 1 | (\textbf{reject} \;\; T_n) \wedge (T_i = \textbf{rethink} \;\; T_{i-1}) \wedge T_1 = T \quad (i \in (1, n))$$

(i.e., any theory wanes upon sufficient reflection.)

$$\forall T \;\; (\textbf{reject} \;\; T) \Rightarrow \exists T_1, \ldots, T_n, n > 1 | (T_n = T) \wedge T_{i+1} = \textbf{drink} \;\; T_i \quad (i \in [1, n))$$

(i.e. all rejected theories could have arisen during drinking bouts.)

Grope Theory's explanation of Squigol's evolution was presented to IFIPS WG2.1 on 'Algorithmic Languages and Calculi' in early January, 1991. Most interesting to this reporter was the intense use of the *drink* operator in the early years of its development (with the consequent

frequent uses of *reject*, *rework*, and *rethink*) *versus* the relative stability of the theory in the later years (reflected in the long sequences of *examines*). Because of the many discontinuities and reinventions encountered during the Squigol experience, Professor Meertens is considering incorporating inter-grope operators into Grope Theory itself, including *spawn* – to allow parallel lines of development –, *branch* – to explicate new, apparently unrelated theories, and, sadly, *goto* – to explain his occasional reinvention of theories he could have "gone to" in the first place.

When questioned whether such a general theory as Grope Theory could provide any leverage in actual applications of the theory, for example through adaptation of potent meta-theorems, Professor Meertens answered, "Categorically! Well, actually *only* categorically..." This not-so-veiled reference to 'category theory' – an advanced algebraic child of Grope Theory – irked its German (formerly West German) proponents to the point where one suggested "grope theory no 'natural transformations' have and not useful be could." Professor Meertens was quick to reply that in Grope Theory "all transformations are natural, just some transformations are more natural than others." Other national reactions varied from Californians' inability to "relate" to the theory to the explicit statement by the French that it should be "reinvented."

Professor Meertens is not only famous for abstract thinking; his invention and practice of 'Abject, Disoriented Programming' in the early '70s has influenced his many students and colleagues alike. When queried on what implications Grope Theory's application to Squigol might have to practical programming practices, the scholar began to mutter what seemed to be the alphabet – in several languages, no less – concluding our brief interview.

Seriously, Lambert, congratulations on the last 25 years with CWI. I consider it an honor to be your friend and to have been able to participate in this celebration of your accomplishments.

Free style Specwrestling : Demonic composition and choice.

Jaap van der Woude

Notwithstanding the celestial order days may differ in length, especially when you share an apartment with Lambert. Some 44 hours without sleep, balancing between caffeine poisoning, tar accumulation and "spirituality", in short: (I'm sorry, trowers!) typical preparation for a lecture.
The boxcalculus he was to talk about turned out to be very delay sensitive, but he managed to dream up an occupation for me too : a demonic composition!

The morbidity of the problem dawned slowly, but I had no choice ... in a paroxysm of mathodism, demonic choice was added to the problem

Let me state the demonic composition question:

Given two machines R and S (say: relations or spec(ifi-cation)s). Define the demonic composition R@S as the usual composition, provided that it is only defined in states x such that R is defined on all the S-results Sx.
<u>Task 0</u> Show that @ is associative.

For the demonic choice : define R□S as the usual union, provided that it is only defined in states in which both R and S are defined.
<u>Task 1</u> Show that @ distributes over □ in both directions.

In this setting the problem may look familiar to those well educated in (wp)-semantics. Is there any point in redoing it? Indeed, is there any point at all? At that time (and still) I took a pointless approach. How about it then?
For the sequel some familiarity with the specalculus is needed, see the sweatshop syllabus for Ameland 1990.

The pointwise definitions of @ and $\square$ can be formulated as

$$x\ R@S\ y \equiv x\ R\circ S\ y \wedge (\underline{A}z : z S y : (\underline{E}w :: w R z))$$

and

$$x\ R\square S\ y \equiv x\ R \sqcup S\ y \wedge (\underline{E}w, z :: w R y \wedge z S y)$$

In case you don't like nightmares don't try to formulate the tasks pointwise. The pointless reformulations:

(0) $\quad R@S = R\circ S \sqcap \neg(\neg(\Pi\circ R)\circ S)$

(1) $\quad R\square S = (R \sqcap \Pi\circ S) \sqcup (S \sqcap \Pi\circ R)$

If you really want to interpret (0) and (1), think of "Π" as the full relation and read "$\sqcap \Pi\circ P$" as the domain restriction to the domain of P.
The negationful conjunct in (0) may be shaped up by the division operator "/", defined by

$$P/Q \sqsupseteq X \equiv P \sqsupseteq X\circ Q$$

Since "/" will be used extensively, it is useful to state some properties and prove two of them that are less familiar: (priority : $(\sqcap, \sqcup)$; / ; $\circ$; $(\neg, \cup)$)

(2) $\quad P/Q = \neg(\neg P\circ Q^{\smile})$

(3) $\quad (P/Q)/R = P/R\circ Q$

(4) $\quad (P \sqcap Q)/R = P/R \sqcap Q/R$

(5) $\quad P/(Q \sqcup R) = P/Q \sqcap P/R$

(6) $\quad \Pi\circ(\Pi\circ P/Q) = \Pi\circ P/Q$

(7) $\quad (P \sqcap Q)\circ R \sqcap P/R^{\smile} = Q\circ R \sqcap P/R^{\smile}$

(8) $\quad \Pi\circ P \sqcap Q/(\Pi\circ P \sqcap R)^{\smile} = \Pi\circ P \sqcap Q/R^{\smile}$

[Interpret "$\smile$" as relation inversion]

By (2) the reformulation (0) can be expressed as

(9) $\quad R@S = R \circ S \sqcap \pi \circ R / S^{\smile}$

For the proofs of (7) and (8) the following three specclassics are needed

(10) $\quad (P \sqcup \neg Q) \sqcap Q = P \sqcap Q$ "complement rule"

(11) $\quad P \circ (Q \sqcup R) \circ S = P \circ Q \circ S \sqcup P \circ R \circ S$ "disjunctivity of "$\circ$" "

(12) $\quad P \circ (Q \sqcap \pi \circ R) = P \circ Q \sqcap \pi \circ R$

<u>Proof</u> for (7) and (8)

$$
\begin{aligned}
& (P \sqcap Q) \circ R \sqcap P / R^{\smile} \\
= & \{(10);(2)\} \\
& ((P \sqcap Q) \circ R \sqcup \neg P \circ R) \sqcap P / R^{\smile} \\
= & \{(11);(10-\text{dual});(11)\} \\
& (Q \circ R \sqcup \neg P \circ R) \sqcap P / R^{\smile} \\
= & \{(10);(2)\} \\
& Q \circ R \sqcap P / R^{\smile}
\end{aligned}
$$

["complement rule for / "

$$
\begin{aligned}
& \pi \circ P \sqcap Q / (\pi \circ P \sqcap R)^{\smile} \\
= & \{(2)\} \\
& \pi \circ P \sqcap \neg (\neg Q \circ (\pi \circ P \sqcap R)) \\
= & \{(12);\text{De Morgan}\} \\
& \pi \circ P \sqcap (\neg (\pi \circ P) \sqcup \neg (\neg Q \circ R)) \\
= & \{(10);(2)\} \\
& \pi \circ P \sqcap Q / R^{\smile}
\end{aligned}
$$

["domain restriction distributes over / "

$\square$

These are the preliminaries, they belong to the standard specalculus, but (7) and (8) were not developed.

In order to fullfill the tasks, it is sufficient to extend the standard specalculus with a few observations. From (1) and (9) it is clear that "@" and "$\square$" are just domain-restrictions of "$\circ$" and "$\sqcup$". So

(13) $R @ S = R \circ S \sqcap \pi_\circ R @ S$

(14) $R \,\square\, S = (R \sqcup S) \sqcap \pi_\circ (R \,\square\, S)$

Expressions for $\pi_\circ R @ S$ and $\pi_\circ (R \,\square\, S)$ will be calculated below. They, in turn, can be used for almost trivial proofs for (13, 14).

(15) <u>Lemma</u> $\pi_\circ R @ S = \pi_\circ S \sqcap \pi_\circ R / S^\vee$

(16) $\pi_\circ (R \,\square\, S) = \pi_\circ R \sqcap \pi_\circ S$

<u>Proof</u>

$$
\begin{aligned}
& \pi_\circ (R @ S) \\
= & \{ (9) \} \\
& \pi_\circ (R \circ S \sqcap \pi_\circ R / S^\vee) \\
= & \{ (6) ; (12) ; (6) \} \\
& \pi_\circ R \circ S \sqcap \pi_\circ R / S^\vee \\
= & \{ \pi_\circ R = \pi_\circ R \sqcap \pi \ ; (7) \} \\
& \pi_\circ S \sqcap \pi_\circ R / S^\vee
\end{aligned}
$$

$$
\begin{aligned}
& \pi_\circ (R \,\square\, S) \\
= & \{ (1) ; (11) \} \\
& \pi_\circ (R \sqcap \pi_\circ S) \sqcup \pi_\circ (S \sqcap \pi_\circ R) \\
= & \{ (12) \ 2x \ ; \ \text{calc} \} \\
& \pi_\circ R \sqcap \pi_\circ S
\end{aligned}
$$

$\square$

<u>Proof</u> <u>for</u> <u>Tait</u>$_\circ$:

$$
\begin{aligned}
& (R @ S) @ T \\
= & \{ (9) ; (13) \} \\
& (\pi_\circ R @ S \sqcap R \circ S) \circ T \sqcap \pi_\circ R @ S / T^\vee \\
= & \{ (7) \} \\
& R \circ S \circ T \sqcap \pi_\circ R @ S / T^\vee \\
= & \{ \text{below (deferred proof obligation)} \} \\
& R \circ S \circ T \sqcap \pi_\circ S / T^\vee \sqcap \pi_\circ R / (S @ T)^\vee \\
= & \{ (6), (12), (6) \} \\
& R \circ (S \circ T \sqcap \pi_\circ S / T^\vee) \sqcap \pi_\circ R / (S @ T)^\vee \\
= & \{ (9) \ 2x \} \\
& R @ (S @ T)
\end{aligned}
$$

deferred proof obligation :

$$
\begin{aligned}
& \Pi \circ (R @ S)/T^{\vee} \\
= & \{(15)\} \\
& (\Pi \circ S \sqcap \Pi \circ R/S^{\vee})/T^{\vee} \\
= & \{(4);(3);\text{"}\cup\text{"}\} \\
& \Pi \circ S/T^{\vee} \sqcap \Pi \circ R/(S \circ T)^{\vee} \\
= & \{(8);(6)\} \\
& \Pi \circ S/T^{\vee} \sqcap \Pi \circ R/(\Pi \circ S/T^{\vee} \sqcap S \circ T)^{\vee} \\
= & \{(9)\} \\
& \Pi \circ S/T^{\vee} \sqcap \Pi \circ R/(S @ T)^{\vee}
\end{aligned}
$$

$\square$

Since $P/I = P$ and $\Pi/P = \Pi$ it follows easily that

$$R @ I = R = I @ R.$$

So, the specs with "@" as operation form a monoid.
For "$\square$" things are different. Clearly, "$\square$" is associative, but "$\square$" doesn't have a unit. It can be shown that with $\leqslant$:

$$P \leqslant Q \equiv P \square Q = Q$$

the specs form a CPO with bottom Π, but it's not a complete lattice as is the case with $\sqsubseteq$.

<u>Proof for Task 1</u>

$$
\begin{aligned}
& (R \square S) @ T \\
= & \{(9);(14)\} \\
& (\Pi \circ (R \square S) \sqcap (R \sqcup S)) \circ T \sqcap \Pi \circ (R \square S)/T^{\vee} \\
= & \{(7)\} \\
& (R \sqcup S) \circ T \sqcap \Pi \circ (R \square S)/T^{\vee} \\
= & \{(11);(16);(4)\} \\
& (R \circ T \sqcup S \circ T) \sqcap \Pi \circ R/T^{\vee} \sqcap \Pi \circ S/T^{\vee} \\
= & \{\text{calc};(9)\ 2x\} \\
& (R @ T \sqcap \Pi \circ S/T^{\vee}) \sqcup (S @ T \sqcap \Pi \circ R/T^{\vee}) \\
= & \{P @ T \sqsubseteq \Pi \circ T;(15)\ 2x\} \\
& (R @ T \sqcap \Pi \circ S @ T) \sqcup (S @ T \sqcap \Pi \circ R @ T) \\
= & \{(1)\} \\
& R @ T \square S @ T
\end{aligned}
$$

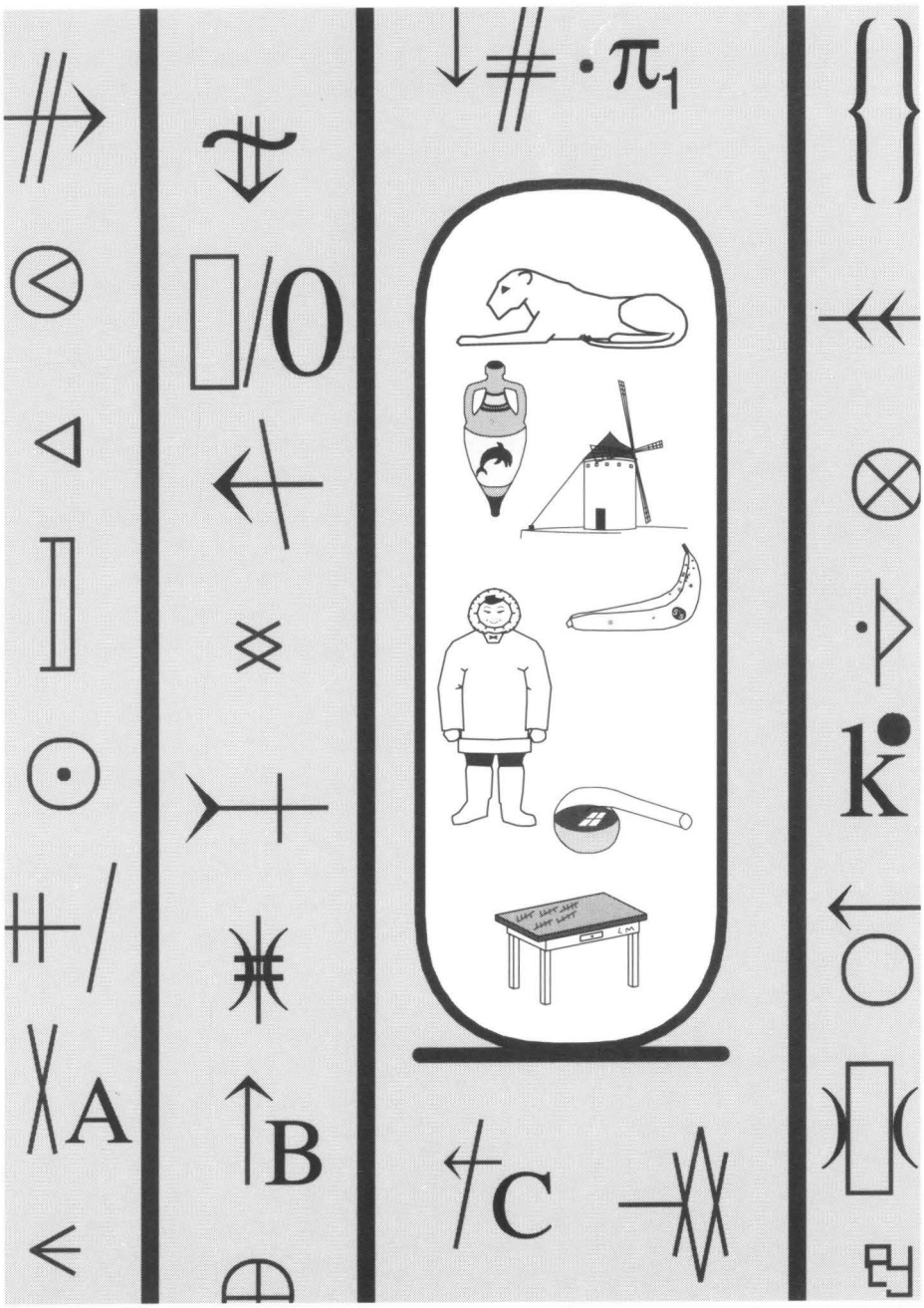
and

$$
\begin{aligned}
& R@(S \,\square\, T) \\
= \; & \{ (9) \,;\, (14) \} \\
& R \circ ((S \sqcup T) \sqcap \pi \circ (S \,\square\, T)) \;\; \sqcap \;\; \pi \circ R / (S \,\square\, T)^{\smile} \\
= \; & \{ (12) \,;\, (11) \} \\
& (R \circ S \;\sqcup\; R \circ T) \sqcap \pi \circ (S \,\square\, T) \;\; \sqcap \;\; \pi \circ R / (S \,\square\, T)^{\smile} \\
= \; & \{ (14) \,;\, (8) \} \\
& (R \circ S \;\sqcup\; R \circ T) \;\; \sqcap \;\; \pi \circ (S \,\square\, T) \;\; \sqcap \;\; \pi \circ R / (S \sqcup T)^{\smile} \\
= \; & \{ (16) \,;\, (5) \} \\
& (R \circ S \;\sqcup\; R \circ T) \;\; \sqcap \;\; \pi \circ S \;\; \sqcap \;\; \pi \circ R / S^{\smile} \;\; \sqcap \;\; \pi \circ T \;\; \sqcap \;\; \pi \circ R / T^{\smile} \\
= \; & \{ (15) \; 2x \;\; ; \; \text{calc} \} \\
& (R \circ S \;\; \sqcap \;\; \pi \circ R@S \;\; \sqcap \;\; \pi \circ R@T) \;\sqcup\; (R \circ T \sqcap \pi \circ R@T \;\; \sqcap \;\; \pi \circ R@S) \\
= \; & \{ (13) \; 2x \;\; ; \; (1) \} \\
& R@S \;\square\; R@T
\end{aligned}
$$

$\square$

Frustrating is still the fact that the truth of statements in the tasks is easily seen with a pointwise interpretation but a rigorous proof on the spec level is unappealing. However, the bottomline is

Even if there's no point in life, things behave as they should.



Stelling: *Het is niet onmogelijk een PostScript-programma zodanig op te zetten dat de uitvoer op Lambert is toegesneden.* **Bewijs:** *volg bovenstaand lijnenspel.* (**PS:** *Je komt er wel uit hè, Lambert? Groeten, Bert*)