THE UNIVERSITY OF NORTH CAROLINA
University of North Carolina in CHAPEL HILL · *North Carolina State College of Agriculture and Engineering in* RALEIGH · *The Woman's College in* GREENSBORO

JOHN W. CARR, III, *Director*
COMPUTATION CENTER

CHAPEL HILL, NORTH CAROLINA

29. Sept. 1961.

Kaere Jensen.

Mange tak for dit meget velkomne brev fra den 28. sept. Det var dejligt at høre saa positivt nyt om oversaetteren. Kun er min appetit blevet forøget: Har I tal for oversaettetid? Har I prøvet andet end testprogrammer? Vilstrup vil i alle tilfaelde kunne levere en del, hvis det behøves.

Angaaende oversaetteteknik: Backus-notationen er en beskrivelse af de tilladte synteser. En oversaetter har brug for an analysemetode. Irons kommer igennem ved brutal kraft (i virkeligheden er hans metode jo slet ikke elegant!) idet han systematisk prøver alle synteseregler. Men det vil jo sige at hans metode slet ikke gør brug af de mere specielle egenskaber ved syntaksen, saadanne egenskaber, som kan afledes ved et systematisk studium. For eksempel udnytter han slet ikke at visse symboler er parenteser, mens andre ikke er det. Denne forskel mellem de forskellige symboler tilsløres i virkeligheden i Backus-notationen, men kan dog uden videre udledes af den. Ved at ignorere disse specielle egenskaber ved ALGOL syntaksen opnaar han den generalitet som tillader at man aendrer syntaksen vilkaarligt, men paa bekostning af et større besvaer ved oversaettelsen af hvert eneste program.

Den modsatte fremgangsmaade er Dijkstras (se Automatic Programming Information no. 7, May 1961, findes blandt andre API ting i en af mine haengemapper, bed evt. Kirsten Andersen om at rode den frem, det er meget vigtigt at du studerer den nøje, Dijkstras artikel er klar som kildevand). Jeg pusler for øjeblikket med at aflede de specielleegenskaber for delimiters fra syntaksen. Jvff. Dijkstras CI, state of comma interpretation. F. eks. kan man spørge: Hvor mange forskellige slags semikolon findes der? Svar: 4. For hver enkelt slags, praecis hvilke symboler kan følge umiddelbart efter? Osv. Jeg er derfor ikke enig i at Irons metode og de metoder som baseres paa en oversaetterstak er identiske.

Det var herligt at høre om Jensens fremragende foredrag ved NORDDAM. Det er derhen vi skal! Du skal udelukkende beskaeftige dig med en ny maskine (eller maskiner) og bør helt glemme GIER. Eller hvordan med at gennemføre direktørens plan: at sy en helt ny logik paa en af de foreliggende GIER'er. Og lave den saa god at alle forbrugere straks bestiller en ny syning og aldrig bruger den gamle! Den skal includere automatiske tromleoverførsler som paa ATLAS. Og naturligvis aritmetik i stak, og dynamisk typebehandling, og maaske virkelige boolean arrays, direkte adresserbare, og naturligvis saedvanlige arrays indbygget, og hvad med dynamisk behandling af data beskrevet som i COBOL Data Division? Og saa naturligvis alle fornuftige operationer, inclusive relationer og logiske. Og hvad med lidt parallel operation mellem konsecutive ordrer. Jeg har tit taenkt at den vaesentligste indvending mod en aritmetik som arbejder helt i stakken kunne fjernes hvis den tid der gaar med at udføre de lidt mere indviklede operationer (flydende aritmetik) kunne bruges til samtidig at udføre de adskillige lageraccesser, som er nødvendige til at

THE UNIVERSITY OF NORTH CAROLINA
University of North Carolina in CHAPEL HILL · *North Carolina State College of Agriculture and Engineering in* RALEIGH · *The Woman's College in* GREENSBORO

JOHN W. CARR, III, *Director*
COMPUTATION CENTER

CHAPEL HILL, NORTH CAROLINA

opsøge den naeste operand. Hvis det altsaa overhovedet er umagen vaerd, det ville utvivlsomt kraeve en del logik og administration. Men for at vende tilbage til det vaesentligste: du maa ikke spilde din tid med GIER i dens nuvaerende skikkelse. Vi staar overfor at skulle designe en helt ny maskine og har ikke et sekund at spilde. Vi vil tids nok blive tvunget til at traeffe beslutninger som vi ikke har taenkt igennem.

Er der nogen som har sat sig ind i hvad der vil kraeves af et kørende COBOL program? Har du sat dig ind i COBOLs data division? Har du overvejet hvad der ville kraeves af en maskine som skulle udføre COBOLs operationer, med type og format leveret i det kørende program og med meget lidt oversaettelse? Jeg sender en COBOL 61 rapport som jeg har faaet til overs.

Hvor mange fejl er der fundet i det syede lager siden jeg rejste?

Med mange hilsener til Mondrup og alle de andre (undtagen Bech, som faar et eget brev).

Peter Naur

PS. Jeg kommer i tanke om at jeg glemte at maerke Bechs brev med AIR MAIL. Jeg vedlaegger derfor en kopi, som du bedes give videre til ham.
PPS. Jeg har laest korrektur paa vor artikel om Storage Allocation. Damen hos ACM var meget ulykkelig over ikke at høre fra dig.

Mr. Peter Naur
c/o Computation Center
University of North Carolina
Chapel Hill
North Carolina U.S.A.

Dit brev 14.1o.61.

Kære Naur.

Hermed følger diverse smukke papirer: Algolprogram til trykning af kontrol-identifikatorliste, output fra dette arbejdende på sig selv samt arbejdende på "Dorte" (et af Vilstrups programmer). Jeg leverer ingen forklaringer til disse udskrifter, hvis de ikke kan læses uden må de gøres bedre.

Efter de sidste rettelser, der er lavet for 3-4 dage siden, har vist sig at være i orden føler vi os nu ovenpå - det er så dejligt at kunne ringe til folk og sige at der er en parentes for meget i program det og det på side .. osv. Vi glæder os dog til , at Mondrup får fejludskrifterne til det kørende program færdig, ethvert lidt større program vi har forsøgt at køre, er på et eller andet tidspunkt gået i stå, og det kan tidt være svært at hitte ud af hvor man er henne.

Trods dine bemærkninger om Irons metode og matrix-metoden kan jeg stadig ikke se den store principielle forskel (du har førøvrigt ret i at der ikke er noget elegant ved Irons) såvidt jeg kan se udnytter vi heller ikke det at noget er parenteser til andet end afslutningen af en syntaktisk enhed - hvad Irons vel også må gøre. Oversætterstakken indeholder hos os præcis de samme oplysninger (selvfølgelig på anden form) som den implicerede stak der ligger i at "diagram" er en rekursiv procedure. Den måde vi har komprimeret ~~sta~~ matricen på svarer i realiteten til at vi skal gennemsøge, om ikke alle syntaktiske konstruktioner, så dog et betragteligt udvalg. Der er ingen tvivl om, at man i den praktiske realisation af en oversætter skal nærme sig Dijstra, idet jeg dog vil anse det for meget væsentligt, om man kan systematisere de skiftespor jeg har indtrykket af står spredt over programmet og gør det for "skræddersyet" til let at ændre.

Jeg har førøvrigt tilladt mig at rode lidt i dine skabe og skuffer for at se om du havde noget materiale direkte fra Dijstra - artiklen i A.P.I. har jeg men det ville ligne dig meget dårligt hvis du ikke har skaffet noget mere direkte fra Dijstra. Har ikke fundet noget.

Håber skrivemaskinen fremmer læseligheden af mine udgydelser - jeg fandt osse ud af at det var nødvendigt at have kopi - jeg kunne ikke huske hvor meget jeg havde skrevet de andre gange - men fra nu af kan jeg altså.

Det ville være interessant at vide, hvad der er blevet af korrekturen til COM. Jeg postede dem højst personlig (luftpost) mellem 8 og 8½ time efter modtagelsen ???

19.10.61 (det kom altså ikke afsted):

Jeg har funderet en hel del over hurtigere metoder til f.eks. identifikator-opslag - der er vist ingen tvivl om, at disse skal foretages i et specielt gennemløb - når man da ikke har et meget stort lager. Ved de fleste af de anvendelige metoder opnår man desuden at hele identifikatorens længde bliver identificerende., faktisk uden ekstra omkostninger.

Bech har så travlt med de forskellige udvidelsesplaner at han ikke har tid til at skrive - hvad du vel heller ikke havde ventet, men han beder mig hilse.

Analex-printeren er nu så småt brugbar - offline, med en Facit strimmellæser - Mondrup brugte den i dag til en KP2o udskrift, og han var dybt imponeret af hastigheden - hvis vi nu bare ikke får alt for meget bøvl med at den går i stykker skulle vi hermed være over den værste udskrivningsflaskehals.- samt parate til den endelige og uafvendelige papirdød.

GIER: 1. eksemplar ser endelig ud til at køre - de har naturligvis måttet lave adskillige ændringer i elektronikken,

Jeg har endnu ikke prøvet at trykke på knapperne, men har skrevet et trykprogram som skulle være klar til indkøring nu. Olga virker, men det er kedsommeligt at skrive i absolutte adresser. Det kneb gevaldig for mig at komme igang med Gier-koden, men nu hjælper det - der er bare alt for store muligheder før en halvordsjæger x som mig, man kan nørkle i en uendelighed - eller måske i flere. Jeg har gjort det i rigt mål ved dette første program - forhåbentlig er jeg derved kommet ind i koden.

Iøvrigt er det hverken Bechs eller min mening at jeg skal fortsætte med den almindelig Gier-kodning. Efter trykprogrammet skal jeg tilrettelægge et ordreindlæseprogram med symbolsk-addressering. Jeg vil skrive det i ALGOL og lade andre om resten, og så selv tage fat på ALGOLoversætter , og jeg tror at oversætteren skal skrives uafhængig af maskinen selv om den muligvis skal realiseres på GIER.

Nå, nu får du ikke mere - jeg skal nå i byen inden lukketid.

Hilsen

28. Oktober 1961.

Kære Jensen, Mange tak for dit sidste meget fyldige brev. Det har givet mig mangfoldige frydefulde stunder. Først og fremmest er det dejligt at se at i det mindste du rent praktisk giver til kende at du mener at Algol kan bruges til noget. Og dit program til udskrift af lister er i sandhed smukt. Jeg har blot et par temmelig uvæsentlige bemærkninger: I den del af programmet der "ved fup og svindel trykker en identifikator" er der en syntaktisk fejl. If kan aldrig følge umiddelbart efter +. Du må derfor indsætte en parentes: Kar := kar + (if ... Det er en fejl i matricen at denne fejl kan passere. I de smukke udskrifter er der en detaille som undrer mig: Den adresse som opgives ved identifikatoren for en ferrit-procedure er 4 mindre end den som senere opgives under den tilhørende blokbeskrivelse. Ligeledes er der en overlapning af en ordre mellem på hinanden følgende procedurer, så vidt jeg kan se. I udskriften for blokoversigtprogrammet er det f.ex. ikke klart om 130 hører til "start" eller "første".

Hvad angår oversættermetoder er det jo en smagssag hvad man anser for en principelt forskel - lad os hellere komme til sagen: Jeg er enig med dig i at det er ønskeligt at man kan finde et system i de skiftespor som dirigerer færdselen, men jeg anser Irons' metode for at være for kostbar. Hverken Dijkstra eller Bauer og Samelson angiver nogen systematik til at nå frem til en konkret oversætter, det meste er vist foregået "på lurenkik". Jeg har i de sidste uger prøvet at afbøde herpå, og jeg tror at være nået til ganske rimelige resultater. Mit øjeblikkelige billede er omtrent følgende:

1. gennemløb: Identifikatorer samles i liste med eet individ pr distinkt identifikator, dvs. uden hensyn til blokstruktur. Af hensyn til vilkårligt lange identifikatorer består identifikatorlisten af en primær og en sekundær. Den primære har eet ord pr identifikator. Den sekundære har nul, eet eller flere pr indentif. efter behov. De sekundære ord som hører sammen hægtes sammen på LISP-liste-vis. Det primære ord indeholder: 1 bit til at angive om der følger sekundært ord, 3 første og 2 sidste characte rer og totalt antal karakterer modulo 32 eller hvad der nu bliver plads til.

Konstanter samles i liste, på sædvanlig vis.

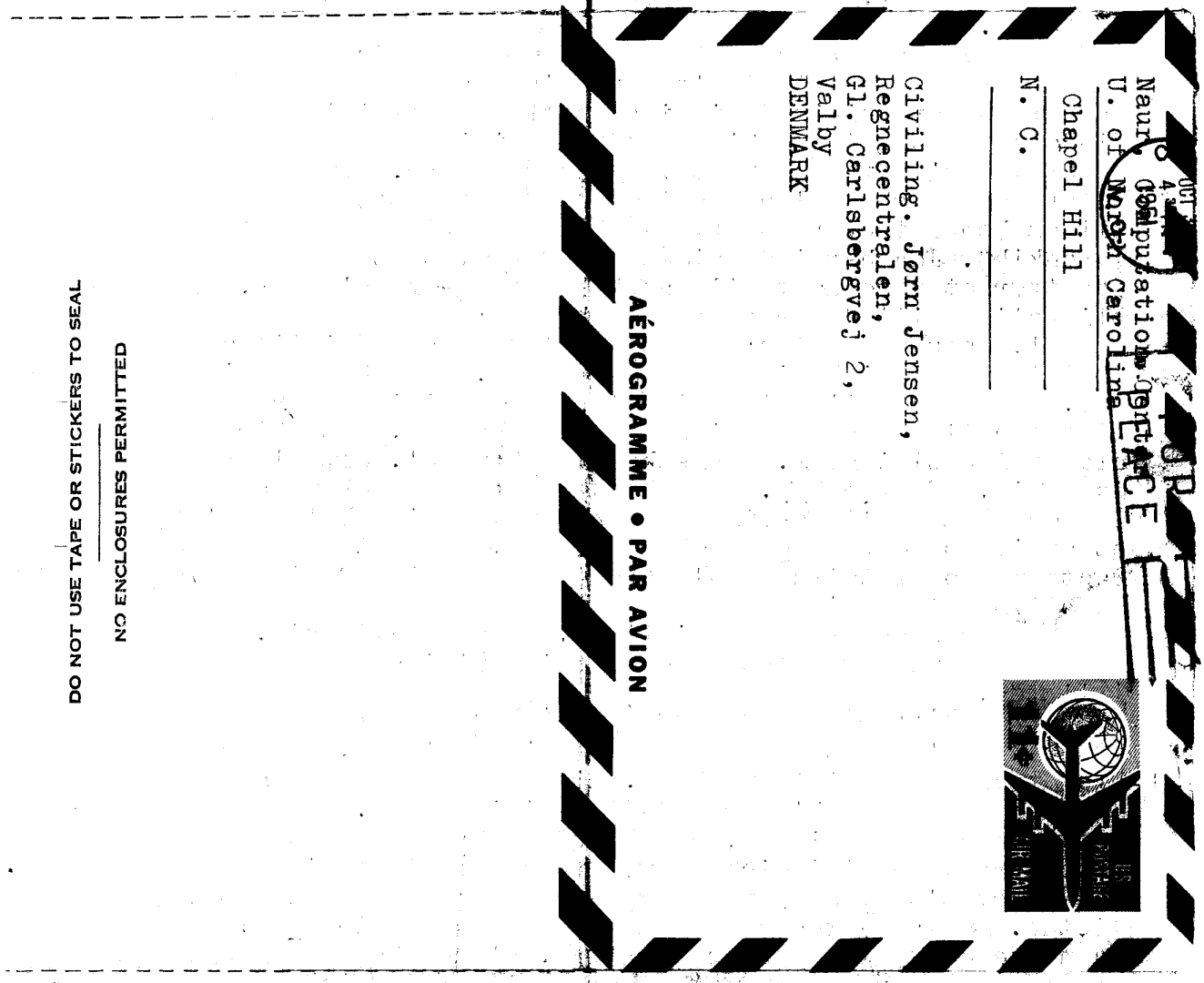
I output erstattes enhver identifikator med dens nummer i listen.

Deklarationer samles (dog ikke de bestanddele som består af udtryk eller statements) samles i een liste, som både har tak-blokstruktur og listestruktur. Listestrukturen bruges til at sammenhægte ens deklarerede identifikatorer. Desuden indsættes diverse oplysninger, såsom symbolsk adresse for labels og procedures, antal indices for arrays, specificationer for formelle. Ved hvert blok-end fjernes deklarationerne for denne blok fra listen. De kan enten sendes ud som output, efter end; i så fald må næste gennemløb forløbe baglæns. Ellers kan de sendes til tromle. - Denne metode forudsætter intetsomhelst om deklarationernes rækkefølge. Det er også let at sortere dem ved end.

Operatorer kopieres uden analyse til output. Blandt operatorer medregner jeg aritmetiske, logiske og relationelle. Alle andre delimiters som ikke allerede er væk (såsom . 10 og strengparenteser) henregner jeg til klasserne parenteser og pseudoparenteser. Under dette gennemløb kontrolleres disse symboler, hvorvidt deres forekomst er syntaktisk mulig. Denne kontrol vil gøre brug, dels af en lille stak som holder alle endnu ikke afsluttede parenteser, dels af en tilhørende situations-

parameter med 10-15 mulige værdier, som fortæller noget om hvilken konstruktion man er i gang med, og endelig et boolean array som med indgang situationsparameter og ny delimiter fortæller om den ny delimiter er mulig. Det meste heraf tjener kun kontrolformål. Hovedformålet er at komplettere alle pseudoparenteser med interne komplette parentespar. Dette vil så vidt jeg kan se simplificere de følgende gennemløb umådelig.

2. gennemløb: Analyse og kontrol af udtryk. Under dette gennemløb vil blok-begin og -end bevirke at en identifikatorliste bliver ført a jour. Denne liste vil igen have eet ord pr distinkt identifikator. Hver ny information, som indsættes ved blok-begin, bevirker at den hidtil gældende overføres til en stak (på tromle). Under analysen af udtryk udføres en fuldstændig typekontrol ved hjælp af en typemæssig simulation af udtrykkets endelige udregning. Tillige vil på den måde udtrykkets endelige type kunne sammenlignes med hvad der kan ventes ud fra omgi-



vende parenteser og pseudoparenteser. For de ambitiøse vil der på dette stadium være fulde muligheder for at optimalisere i specielle tilfælde.

3. gennemløb. Jeg forestiller mig at optput fra 2. gennemløb kun er adresseret for variables vedkommende, mens alle interne referencer enten er symbolske (labels og procedures) eller implicitte (for-statements, then, else, med flere). Dette output vil derfore være meget kompakt. Det sidste gennemløb vil være at sammenligne med en loading-process. Dette gennemløb vil sikkert kunne slås sammen med 2det hvis dette er en fordel.

Jeg håber inden jeg kommer hjem at have udviklet ALGOL algortimer for alle væsentlige dele af dette system - med forbehold overfor ændringer. Fra Dijkstra har jeg ikke andet end en oversigt over hans system. - Det er rædselsfuldt at høre at du laver tryk-program og ordreindlæsning til GIER. Programmerne i det syede lager kan da enhver kopiere, og hvorfor bruger man ikke Wenesers symbolsk system?

Hilsen fra Naur

Mr. Peter Naur
Computation Center
University of North Carolina
Chapel Hill
North Carolina
U.S.A.

brev 28.10.61 tjah 3.nov.61

Kære Naur.

Du har ret - lad os holde op med det pjat og snakke realiteter.

Vi bøjer os dybt i støvet for eksperten, matricen er gal for betingede udtryk, fejlen ligger i læsningen af den grå - den vil ikke blive rettet i øjeblikket.

De blokgrænser, der udskrives inkluderer af en eller anden grund ikke procedure hovedet (antagelig fordi det er blokken og ikke proceduren, der angives ellers bliver der vrøvl ved tromleprocedures) slutgrænsen er simpelt hen første efter blokken (også i overensstemmelse med de faktiske parametre til kald af tromle-blok, hvor der vist forøvrigt er en ubetydelig uoverensstemmelse mellem den gule og de faktiske forhold. Sekvensen for tromleoverførsel er ligeglad om den får sidste hao (ulige), der skal overføres eller første (lige), der ikke skal, og oversætteren giver altså det sidste (tilfælde).

Til dit billede af oversætteren har jeg følgende bemærkninger:

I ferritlagre af vores størrelse tror jeg det vil være nødvendigt med flere gennemløb end du har planlagt, hvis man da ønsker en hurtigere oversættelse end vi præsterer iøjeblikket,
Desuden vil jeg tro, det kan være fordelagtigt at prøve om man kan lave en oversætter der så langt som muligt oversætter uden hensyn til deklarationer, såvidt jeg kan se gør man derved selve oversættelsen mere maskinuafhængig, navnlig ved fremtidige udvidelser af deklarationernes art (uha, det var vist en uartig tanke).

Mit billede ser derfor sådan ud:

1. gennemløb (indlæsning): Som DASK dog uden afkortning af identifikatorer og uden dannelse af label-og procedurelisen.

2. gennemløb. Oversættelse af identifikatorer, dannelse af liste over labels, procedures og switches (jeg kan endnu ikke overskue om denne liste er nødvendig stadig). Denne oversættelse bør gøres så hurtig som mulig. En fornuftig udformning af vedlagte algoritme skulle i de fleste tilfælde skære antal sammenligninger ned til et minimum (algoritmen svarer til at man har en selvstændig liste for identifikatorer der f.eks. begynder med samme bogstav og ender på de samme 3 bits eller noget tilsvarende).
Muligvis akn dette gennemløb inkludere en deklarationsliste som din.
3. gennemløb: Syntaktisk analyse, der gennemføres uden at vide noget om deklarationer dette svarer til at vi i DASK ville have et specielt gennemløb, hvor vi blot slår op i matricen, finder hvilke programprogrammer vi skal udføre, samt udfører tilstandsfunktionerne, men ikke genererer program.

4.gennemløb: Identifikatorer erstattes med deres deklarationer og symbolske adresser.

5.gennemløb. Oversættelse til maskinkode, eventuel optimalisering mv må ske her Output er stadig kun symbolsk addresseret.

6,gennemløb: som dit 3.

Ja, det var jo lidt flere end du har skitseret, men jeg tror, det kan betale sig at undgå uafladelige tromletransporter - båndtransporter er billigere, og jeg tror at amerikanerne dog i nogen grad har ret i deres bestræbelser forat skille oversættelsen op i mere distinkte syntaktiske og maskinkodeprægede faser.
Selvfølgelig kan ALGOL bruges til noget. Selv Mondrup synes vist så, og der er ganske afgjort flere og flere der udtrykker sig ålgolsk, i erkendelse af at tvetydigheder derved reduceres - det er også en stor stimulans at det faktisk er mulig at oversætte maskinelt. Rygter Kirsten Andersen udspreder (dit brev til Chr.A.) dementeres på det kraftigste. (Undskyld min uafladelige forveksling af - og)
Jeg kan forøvrigt betro dig en lille pudseløjerlighed: Mondrup finder det ganske naturligt, rimeligt og let at skrive understregede ord fuldt ud - for øjeblikket er det vist kun Toke, der afkorter.

Ja, det er selvfølgelig trist at jeg beskæftiger mig med mere inferiøre sager - det er dog efterhånden meget sjovt, og lidt nytte kommer der da også ud af det. Jeg har f.eks. lige fået nedfældet, i smukt ALGOL, tryksekvensen - vist endda i så smukt Algol at det skulle kunne køre på DASK, den er ikke renskrevet, så du får den først næste gang - det kniber med fexowritere her, og jeg vil gerne have den på strimmel (dette skrives på en IBM maskine af dem der høter til GIER, jeg kan i denne forbindelse ikke dy mig for at udtale min glæde over den måde strimmellæser, perforator og skrivemaskine er tilkoblet GIER: Afhængig af et internt regist , der selvfølgelig kan sættes kodningsmæssigt, kan in- og output efter frit valg tages fra strimmellæser eller skrivemaskine henholdsvis sendes til perforator og/eller skrivemaskine, dette simplificerer kommunikationenc ganske væsentligt vedrørende rettelsesmuligheder under indlæsning, kontroludskrifter do programmer osv).

Dette er hvad jeg har på hjerte for øjeblikket - åh nej to ting til:

Sekvenslageret har ikke udvist nogen fejl siden de afrejse (jeg må for resten have undersøgt om Kirsten Andersen har sendt dig The Program of The Wired Store on DASK).

Vi fandt (dvs Mondrup) en fejl i oversætteren i går: indiceret integer i betingelse gik galt, er såvidt vides rettet.

Skal antagelig hilse fra Bech, Chr.A. m.fl. - ialt fald fra mig selv

THE UNIVERSITY OF NORTH CAROLINA
University of North Carolina in CHAPEL HILL · *North Carolina State College of Agriculture and Engineering in* RALEIGH · *The Woman's College in* GREENSBORO

JOHN W. CARR, III, *Director*
COMPUTATION CENTER

CHAPEL HILL, NORTH CAROLINA
21. Nov. 1961

Kaere Jensen,
Mange tak for dit brev af den 3. Nov. som har interesseret mig i allerhøjeste grad. Jeg er slet ikke ~~xxxxx~~ uenig med dig i at man maaske skal bruge flere end 4 gennemløb. Princippet bør vaere at programmet for hvert gennemløb, inclusive tabeller, skal kunne ligge i ferritlageret, kun programmet bør passere fra og til tromle eller baand. Opdelingen i gennemløb vil derfor vaere i høj grad maskinafhaengig.

Derimod er jeg knap saa overbevist om rigtigheden af din anden tanke om oversaettelse uden hensyn til deklarationer. Jeg er ikke engang sikker paa hvad du mener hermed. Hvis du mener at det kørende program bør klare det meste af typebehandling osv. er jeg ikke uenig. Men man kan dog ikke ignorere declarationerne under oversaettelsen, tvaertimod bør de behandles tidligt da udtryk ikke kan gøres faerdige uden dem.

Jeg er kommet meget laengere med andet gennemløb: identifikatoroversaettelse, opsamling af deklarationer og syntaktisk kontrol for alt undtagen simple udtryk. Din identifikatoralgortime er vist ikke korrekt saadan som den staar, men jeg forstaar nok princippet med at "randomisere" og opdele listen i omtrent lige lange smaalister. Der er naeppe tvivl om at ved meget store programmer er dette den rigtige metode. Jeg har selv puslet med en anden fremgangsmaade som i højere grad gaar ud fra en maskine med ord af saedvanlig størrelse (35-50 bits). Min tanke er at gøre den inderste løkke saa hurtig som mulig, samtidig med at chancen for falske koincidenser er minimal. Saa vidt jeg husker sendte jeg dig 8 sider herom. Fortsaettelsen af historien er imidlertid meget mere interessant (efter min egen ringe mening). Den er vedlagt. Deklarationsbehandlingen er ikke helt tosset, tror jeg. Fremfor alt undgaar den forskelsbehandlingen af labels og procedureidentifiers, og samtidig falder alle krav om en bestemt orden for deklarationerne bort. Langt det interessanteste er imidlertid mine resultater vedrørende scanningen i 2. gennemløb. Princippet er at prøve at gøre maximal brug af alle forhaandenvaerende oplysninger hele tiden. Altsaa: du er midt i programmet, i faerd med 2. gennemløb. Jeg spørger saa: hvad kan du sige om hvilke symboler der nu kunne taenkes at komme? F.ex. ville begin eller step vaere tilladt? Men mere end det: naar du nu faktisk laeser det naeste symbol hvilken af sine saedvanligvis forskellige betydninger har det da? Den opgave at give reglerne herfor kan synes paa forhaand uoverkommelig. Det forunderlige er at hele historien i det vaesentlige vil fylde ca. 60 helord i DASK. Der er faktisk kun omkring 33 mulige tilstande i Algol og hver tilstand kan beskrives med ca. 50 bits. En af grundene til denne økonomi er naturligvis at simple udtryk slet ikke tages med, da de skal vente til 3die gennemløb. Noget mere information indgaar ogsaa i reglerne for parentesstrukturen, men store sager er det nu ikke. Du kan selv granske vedlagte.

se side 25

En af de interessanteste ting ved denne metode er at den slet ikke som man skulle tro kraever megen kyndighed at anvende. En af studenterne her er i faerd med at rekonstruere alle successors (side 18 - 24) paa grundlag af en liste over de ca. 80 delimiterprogrammer med deres mening. Hun har kunnet gøre det paa faa dage og har endda fundet en fejl hos mig og siger i øvrigt at det gaar meget let.
Den samme metode kunne ogsaa anvendes under 1ste gennemløb til analyse af understregede ord, osv. Men nu ikke mere i dag, jeg skal gennemgaa vedlagte ved min nu ugentlige foreslaesningseftermiddag.

Naur