Multiprogrammering en de X8, aflevering 2. (Vervolg van EWD51) X8 Nr. 18.

2.4. Cumulatie van startopdrachten en terugmeldingen.

Als de seinpalen, die de synchronisatie met de externe apparatuur beheersen, maximaal tweewaardig zouden zijn, zouden we de haastsituatie onvoldoende uitgebannen hebben. De essentiele haastsituatie mag dan wel uitgebannen zijn, de "morele" haastsituatie blijft nog over. Een voorbeeld moge dit toelichten.

Stel, dat we een aantal trommeltransporten moeten uitvoeren in overigens willekeurige volgorde. Bij het eerste transport heb je geen weet van de stand van de trommel en zal de machine gemiddeld een halve omwenteling moeten wachten; na voltooiing van dit transport weet je echter drommels goed, hoe de trommel staat en het is niet onverleidelijk om te proberen deze kennis te gebruiken, dwz. de transporten in een zodanige volgorde uit te voeren, dat de wachttijden tussen de transporten zo mogelijk wat kort gemaakt kunnen worden.

Als je nu een organisatie voor trommeltransporten hebt, die van het volgende transport pas notitie kan nemen -door tussenkomst van een dan via programma aangeboden volgende startopdracht- als het vorige transport volslagen is afgehandeld, dan kweekt het streven, die volgende startopdracht dan ook op tijd te geven een "morele" haastsituatie: het is dan wel niet noodzakelijk, maar toch wel hoogstgewenst, dat de "klaarmelding" van het vorige transport een rappe reactie ten gevolge heeft.

In het geval van trommeltransport wordt de autonoom uit te voeren handeling bepaald door:

a) beginadres in kernen
b) beginadres op trommel (plus specificatie welke trommel, als nl. hoogstens 1 trommel tegelijkertijd transporteert)
c) lengte van het transport
d) richting van het transport

De terugmelding bestaat uit twee bits:

e) of het transport succesvol is geweest -dwz. er geen paritsitsfout is opgetreden
f) de terugmelding, dat het transport -gelukt of mislukt, zie e)- voltooid is.

Per startopdracht stelt men nu een aantal consecutieve woorden ter beschikking, waarin in een vaste, nu niet ter zake doende,codering ruimte is voor de gegevens a) t/m e); het gegeven f) wordt door verhoging van een daarvoor gereserveerde seinpaal doorgegeven.

De controle van de trommeltransporten moet dus kennis dragen van deze seinpaal -en toegevoegde flip-flop- plus van de plaats van de codeinformatie a) t/m e).

Ter vermijding van de morele haastsituatie stelt men nu een aantal van dergelijke startruimtes ter beschikking op een aantal consecutieve geheugenplaatsen: als per startopdracht voor de gegevens a t/m e bv. twee woorden nodig zijn en er bv. vier startopdrachten geaccumuleerd kunnen worden, dan kost dit startmagazijn ons dus 8 consecutieve geheugenplaatsen. De trommeltransportorganisatie moet deze startruimtes nu cyclisch afwerken.

Er zijn in dit geval twee extra complicaties, de cyclische afwerking en de foutreactie.

2.4.1. De cyclische afwerking.

De startopdrachten staan nu in een cyclisch magazijn van bekende lengte. Dit startmagazijn wordt onder controle van een ronddraaiende vulwijzer gevuld, wat een geheel programmeerbare aangelegenheid is. Het opnieuw gevuld zijn wordt iedere keer door "V(startseinpaal trommeltransport)" aangegeven.

Het autonome trommeltransport leegt het startmagazijn -werkt de startopdrachten af- onder controle van een cyclisch ronddraaiende "leegwijzer", waarmee de computer niets te maken heeft, behalve dat bij het prille begin deze in phase moet staan: de eerst uitgevoerde startopdracht moet inderdaad de eerstgegevene zijn. Vanaf dat moment draait, rampen voorbehouden, de leegwijzer netjes achter de vulwijzer aan.

2.4.1.1. Een van de oplossingen is om als onderdeel van het proces, dat de machine aanzet, deze "leegwijzer" te clearen; het "tandepoetsprogramma" kan de vulwijzer clearen en we starten de twee processen automatisch in phase. Idealistisch is deze organisatie in orde, van practisch oogpunt is het geen stijl, omdat je in geval van twijfel de boel nooit meer "zo zeker als een huis" weer in het gareel kunt krijgen, anders dan door de machine af en aan te zetten.

2.4.1.2. Aan dit extra bezwaar komt men tegemoet -zij het ten koste van een extra opdracht- als vanuit de computer de leegwijzer gecleard kan worden. Als je deze clear-opdracht in het tandenpoetsprogramma opneemt, dan kun je ook tussentijds de boel in geval van twijfel door de computer onder controle van programma weer onder bedwang krijgen. Hoewel het een extra opdracht in de code van de computer kost, is deze oplossing vooralsnog niet onaantrekkelijk.

2.4.1.3. We kunnen afspreken, dat de leegwijzer op een vaste plaats in het geheugen staat, bv onmiddellijk voorafgaand aan het startmagazijn. Als dit de enige plaats is, waar de vulwijzer onthouden wordt, dan zou dit bij een autonoom transport per woord wel tot erg veel geheugencontacten aanleiding geven. Immers: 1 geheugencontact voor het woord zelf, maar voordat het bijbehorende phisiche adres bekend is, moet een geheugencontact met de bijbehorende -inmiddels eventueel opgehoogde- startinformatie gemaakt zijn, maar om dit mogelijk te maken, is het nodig, dat eerst nog een geheugencontact gemaakt wordt om de heersende waarde van de leegwijzer te pakken te krijgen.

Dit is wel wat al te gek; men kan het aantal geheugencontacten per woordtransport van drie tot twee terugbrengen door te stellen, dat we weliswaar een vaste geheugenplaats voor de leegwijzer hebben ingevoerd, maar tevens bij de trommelorganisatie een paar flip-flips -in ons geval twee- toevoegen, waarin bij elke startopdracht-acceptering (dwz. de P-operatie door de autonome apparatuur) de opgehoogde leegwijzer uit de vaste geheugenplaats copieren. Dit onder het motto, dat tijdens een heel blok-transport de waarde van de leegwijzer constant is.

Opm. Het totaal van 2 geheugencontacten per woordtransport was gebaseerd op de veronderstelling, dat beginadres op de kernen en lengte samen in een woord ondergebracht zouden zijn. Wij komen hier later aan het einde van 2.4 nog weer even op terug.

2.4.2. De foutreactie.

Het is natuurlijk erg prettig om een aantal startopdrachten "in het voor" te kunnen geven, maar laten we ons wel ervan bewust zijn, dat je over het algemeen op de latere transporten slechts prijs stelt, als de voorafgaande transporten succesvol zijn verlopen. Heel duidelijk is dat bij een mislukking van een transport van trommel naar kernen: als hier bij woordtransport iets mis gaat -pariteitsfout- dan kan je het allicht nog eens proberen. Als we nu twee transportopdrachten accumuleren, waarvan

de eerste een stuk en de trommel leest, waarna het volgende iets anders op diezelfde trommelplaatsen wil schrijven, dan is het kennelijk niet de bedoeling, dat dat tweede transport plaatsvindt als het eerste mislukt is.

De vraag is dus, hoe het autonoom transport reageert op een gedetecteerde pariteitsfout, behalve dat in de startgegevens de mislukking wordt geindiceerd (zie punt e, uit 2.4.).

Pogingen om de autonome organisatie voor verdere actie te behoeden door de actieflipflop te clearen, eventueel ook de startseinpaal op nul te zetten, zijn mislukt, omdat de analyse van 2.1. hierdoor in elkaar stortte.

Een gezondere oplossing lijkt het, om de trommelorganisatie met twee toestanden in te voeren, aangegeven door een OK-bit (op een flipflop uitgevoerd). Zodra een pariteitsfout gedetecteerd is, gaat de organisatie in de toestand ~~XXX~~ non OK over; als non OK, dan vindt geen woordtransport plaats, elk transport wordt onmiddellijk onder foutmelding beeindigd. Dit geldt zowel voor het transport, waarin de pariteitsfout is opgetreden, als de volgende transporten, die nog in het startmagazijn staan. Het startmagazijn wordt dus "symbolisch" afgewerkt, daarmee garanderend, dat het hele seinplaenspel ongehinderd doorloopt.

De vraag is, hoe dan de overgang van non OK naar OK bewerkstelligd moet worden. Even hebben we gedacht, dat dit kon op grond van leegheid van het startmagazijn. Het feit, dat het transportmechanisme, om die leegheid te ontdekken, niet voldoende heeft aan de P-operatie als enig contact met de startseinpaal had ons moeten achterdochtig moeten maken. De onbetrouwbaarheid van een automatische overgang naar de OK-toestand realiseert men zich, wanneer een nieuwe startopdracht aan een bijna leeg magazijn wordt toegevoegd. Als de P-operatie vlak voor de V-operatie plaatsvindt, kan de seinpaal in kwestie net even =0 worden, terwijl dat niet het geval is, als de P-operatie juist even na de V-operatie plaats zou vinden: plotseling zou de sequens " 1 → 0 → 1 " een ander effect hebben als de sequens " 1 → 2 → 1 ". En dit was niet de bedoeling: een dergelijke impliciete beeindiging van de non OK toestand maakt een veilige hantering onmogelijk.

Er zijn zo op het oog twee voor de handliggende mogelijkheden, om de machine expliciet de OK-toestand te laten herstellen.

2.4.2.1. Dit is de uitbreiding van de situatie in 2.4.1.2. beschreven: dezelfde opdracht, die de leegwijzer cleart herstelle tevens de OK toestand. Men kan deze twee dingen immers koppelen, omdat de OK-toestand hersteld moet worden op een ogenblik dat het startmagazijn toch leeg is; met vullen kan men dan ook wel weer aan het begin beginnen.

2.4.2.2. De andere mogelijkheid is een uitbreiding van de situatie, zoals deze in 2.4.1.3. beschreven is. Men kan ~~XXX~~ in het startmagazijn behalve transportopdrachten in beide richtingen ook nog "herstel OK-toestand"opdrachten toelaten. In het geval ~~XXX~~ een foutmelding door de computer wordt ontdekt, voegt de computer de opdracht "herstel OK-toestand" aan het magazijn toe en herhaalt de mislukte opdrachten.
De eerste oplossing voegt een opdracht aan de code van de centrale computer toe, de tweede voegt een opdracht toe aan het arsenaal, dat de autonome organisatie uit het startmagazijn accepteert. Behalve uit "uitbreidingsoverwegingen" heb ik voor de laatste oplossing tevens een uitgesproken voorkeur, omdat de centrale computer hier zijn maatregelen al kan treffen zonder dat de eis van leeg startmagazijn op de proppen komt of de extra-specificatie, hoe de autonome transportorganisatie reageert op de nulzettingen als er nog iets loopt. Op deze wijze immers garanderen we dat de autonome organisatie de haar sturende informatie netjes sequentieel krijgt toegediend.

De functionering van het autonoom trommeltransport kunnen we in de veronderstelling, dat de laatste oplossing is gekozen, voor de welwillende lezer alsvolgt documenteren.

Legenda:

de grootheid "cyclische tellerstand" bevindt zich op een vaste plaats in het geheugen;
de diverse gegevens in het startmagazijn worden beschouwd als evenzovele vectoren ter lengte vier:
de eerste vector is boolean array OK herstel [0:3]; als de waarde van een element van deze vector true is, dan is het een herstel OK-toestand opdracht en doen de overeenkomstige waarden uit de andere arrays er niet toe;
voorts hebben we:
integer array beginadres trommel, beginadres kernen, lengte [0:3];
boolean array richting transport, succesmelding [0:3];

De boolean "pariteit in orde" is globaal t.o.v. de ongedeclareerde procedure "volgend woordtransport" evenals de anonyme grootheden, die spporselectie, transportrichting en trommelsynchronisatie regelen. De hieronder gedeclareerde grootheden staan op flip-flops.

```
begin  boolean OK;  integer teller;
start: P(startseinpaal trommeltransport);
       teller:= cyclische tellerstand:= mod(4,cyclische tellerstand + 1);
       if OK herstel[teller] then  OK:=true  else
       begin  if OK then
           begin stel transportrichting in (richting transport[teller]);
                 stel spoorselectie in op (beginadres trommel[teller] ; 1024);
                 wacht op trommelstand (mod(1024,beginadres trommel[teller]));
   volgend woord: if  lengte[teller] ≠ 0 then
                begin volgend woordtransport(beginadres kernen[teller]);
                      beginadres kernen[teller]:= beginadres kernen[teller] + 1;
                      lengte[teller]:= lengte[teller] - 1;
                      OK:= pariteit in orde;
                      if  OK  then  goto volgend woord
                end
           end;
     succesmelding[teller]:= OK
       end;
       V(ingreepseinpaal trommeltransport);
       goto start
end
```

Later hoop ik, bij wijze van voorbeeld de structuur te laten zien van het programma, dat het bovenbeschreven stuk gereedschap bespeelt.

Opm. Bij de startopdracht voor de trommel moeten we van te voren ruimte vinden voor:

lengte	12 bits
beginadres kernen	18 bits
beginadres trommel	19 bits
(trommel)	2 bits
OK-herstel	1 bit

Voor de succesmelding heb je geen nieuwe positie nodig en zou je de OK-herstel-bit kunnen gebruiken. Wij komen zo op een totaal van 52 bits, dat schitterend in twee woorden is onder te brengen. Een complicatie is echter, dat de twee eerste artikelen samen 30 bits vergen. De 3 bits, die je tekort komt, zou je in het tweede woord onder kunnen brengen, bij het accepteren extern op 3 flipflop's overnemen, zodat het het aantal geheugencontacten per woord niet nodeloos oploopt.

3. De coordinator.

3.1. Over de administratie van een bevolking.

In de coordinator moet de administratie plaatsvinden van bv. welke input-en output-apparaten aan welke programma's zijn toegekend, welke programma's zijn opgehouden tengevolge van welke seinpalen etc.etc..

Als het aantal programma's bv. maximaal 10 zou zijn, als de verzameling seinpalen een vaste, beperkte collectie zou zijn, dan zouden we met het in het volgende gestelde probleem niet te maken hebben. Ik wil, vooral in het begin van mijn exploratie's mij niet een zo drastische beperking opleggen en ik zoek dus naar een organisatie, waarin het aantal programma's, het aantal seinpalen dat meedoet, in de tijd willekeurig kan varieren, maar zo, dat al deze entiteiten gedurende hun bestaan hun identiteit niet verliezen.

Een recht-door-zee-methode is om alle entiteiten van een bepaalde klasse naar volgorde van introductie doorlopend te nummeren. Deze methode is practisch onbruikbaar omdat

a) na verloop van tijd deze nummers willekeurig hoog gaan worden
b) op de vraag, hoe men nadere gegevens over entiteit nr. zoveel van een bepaalde klasse ~~XXXXXX~~ kan krijgen, slechts een zoeksysteem antwoord kan geven.

De oplossing, die mij op het ogenblik het meeste aantrekt is geinspireerd op het hotelwezen: zodra je in het hotel komt, krijg je een vrije kamer toegewezen en voor de duur van je bezoek is je kamernummer je identificatie.

Extra faciliteiten zijn, dat als mijn hotel vol is, ik er een verdieping op kan bouwen en, als de bovenste verdieping leeg is, ik deze desnoods weer kan afbreken.

De consecutief genummerde hotelkamers worden in het geheugen op consecutieve plaatsen (of vaste formatie's daarvan) afgebeeld: afbraak van de bovenste etage betekent dus, dat geheugen vrijgemaakt wordt voor andere doeleinden. Het zou dus zin kunnen hebben.

De taktiek is nu, dat elke nieuwe gast de laagstgenummerde vrije kamer krijgt. programmatechnisch is dit eenvoudig: we onthouden de omvang van het hotel, in elke kamer of hij bezet is of niet, en tenslotte het rangnummer van de laagst bezette kamer. Komt er een nieuwe gast bij, dan krijgt hij meteen zijn kamer en zoeken we naar de volgende vrije kamer, verlaat een gast het hotel, dan wordt zijn kamer vrijgegeven en tevens wordt dit kamernummer even vergeleken met het heersende laagste vrije kamernummer, dat zonodig vervangen wordt. Het zoekwerk is hierdoor heel een-

voudig, het is zelfs de vraag of we het abundante gegeven "laagste vrije kamernummer" eigenlijk wel nodig hebben. (Als er veel gezocht zou moeten worden, zou ik de bezetbits in woorden onderbrengen en gebruikmaken van de normeeropdracht om de eerste vrije te vinden; vanwege het "zoveel bits per woord" trekt deze oplossing me minder.)

Ik weet, dat de tactiek "nieuwe gast krijgt laagste vrije kamer" het tijdsgemiddelde van de effectieve hotelomvang niet minimaliseert: als op een ogenblik, dat het hotel heel vol is, zich een blijvertje aanmeldt, dan kan het gastental nog zo frastisch zakken, voordat het blijvertje op de bovenste etage zijn biezen gepakt heeft, kan ik niet gaan afbreken. Ik dacht desalniettemin, dat deze tactiek ten koste van weinig me al een heleboel zou helpen, zeker voldoende, omdat de hotelkamers, die ik voor ogen heb, niet zo vreselijk omvangrijk zullen zijn.

De hotelanalogie gaat verder: menig hotel zal een aantal stamgasten herbergen - de hardware seinpalen, de standaard communicatieprogramma's e.d.; het is duidelijk, dat de stamgasten de laagstgenummerde kamers zullen krijgen, van meet af aan tot in de eeuwigheid.

Ik neem aan, dat het aantal hotels, dat direct door de coordinator beheerd wordt, beperkt, bekend en constant is. Naast de twee genoemde administratieve gegevens zullen we per hotel ook nog de plaats -dwz. het adres van de nulde kamer- moeten onthouden. Ook dit denk ik me op een vaste plaats geborgen.

3.2. De wachtlijsten voor abstracte machines.

Een woord vooraf ter waarschuwing. Ik moet een gebouw optrekken, waarvan mij de contouren beginnen te dagen. Ik heb een idee van het soort fundament, waarop dit gebouw zal rusten: wat het fundament precies zal moeten dragen, zal ik pas weten, wanneer de bovenbouw klaar is. Bij het ontwerp van de bovenbouw moet ik me echter steeds afvragen, of hiervoor nog wel een fundament te bouwen is. Mw. niemand verwachte in dit stadium een waterdicht betoog, waarin nu iets geintroduceerd wordt met een haarscherpe verwijzing naar de noodzaak, die 100 pagina's verder uiteengezet zal worden: die 100ste pagina is nl. nog niet geschreven!

De volgorde, waarin ik de diverse "bedrijfsruimten" van dit gebouw in eerste aanleg zal proberen te beschrijven is min of meer toevallig. Waarschijnlijk kies ik steeds dat onderdeel, waarvoor de minst duistere taakomschrijving en een mogelijke constructie me op dat moment het helderst voor de ogen staan. Dat de leesbaarheid soms moet lijden onder het feit, dat ik niet alles tegelijkertijd zeggen kan en af en toe mijn gedachten verder uitgewerkt zijn, dan ik op dat moment beschrijven kan, zie ik helaas als onvermijdelijk. Tot zover deze apologie.

Als een concrete machine -een transputapparaat- in zijn P-operatie blijft hangen, dat hoeft de coordinator daar geen speciale notitie van te nemen: de concrete machine blijft immers op het vinkentouw en zodra de belemmeringen zijn weggenomen, begint de concrete machine autonoom te lopen.

Met de abstracte machine ligt dat anders: dit zijn programma's, die hun activiteit gestaakt hebben, omdat ze toen niet verder konden. Zodra nu de belemmeringen weg zijn, dan moeten ze uit hun diepe sluimer gewekt worden. Het nagaan van de gevolgen -het bijhouden van de veranderde situatie- van een V-operatie op een of meer seinpalen is een taak, die typisch op de weg van de coordinator ligt.

We beschouwen alle absracte machines, die op een gegeven moment meedoen, dwz. alle abstracte machines, die op een gegeven moment óf hun activiteit expliciteIijk gestaakt hebben -door een P-operatie- of door interruptie ergens in hun activiteit onderbroken zijn. Deze abstracte machines vormen samen de bevolking van HAM, dwz. "Hotel voor Abstracte Machines", en worden gedurende hun bestaan geindentificeerd door het kamernummer van de hen in HAM toegewezen kamer.

Opm. Volgens deze definitie is de coordinator zelf geen abstracte machine en is de coordinator dus niet in HAM opgenomen. Dit geldt, let wel, voor de kern van de coordinator, dwz. dat gedeelte van de coordinator, dat altijd als centrale administrator in de kernen aanwezig zal zijn. (Hoe het daar komt is zo'n liefelijk "prille-begin-probleem", waarmee we ons voorlopig wijselijk niet zullen bezighouden. Het prille-begin-probleem is nl. geisoleerd.) Om het tijdsgemiddelde van de kerngeheugenbezetting door de coordinator niet te groot te maken, zullen we stukken van de coordinator normaliter op de trommel hebben en slechts voorgaats halen, als ze een keer in actie moeten komen. Het kon wel eens wezen, dat we dit het elegantst kunnen organiseren door deze uitlopers van de coordinator te beschouwen als abstracte machines, die dan wel tot de stamgasten van HAM zullen behoren.

Een van de stamgasten van HAM zal zijn de "Dummy Abstracte Machine" c.n. DAM. De tekst luidt:

"LDAM: P(SDAM); goto LDAM"

Een en ander impliceert, dat SDAM een stamgast zal zijn van het seinpalenhotel, waarover later meer.

De Coordinator is nu zo, dat wanneer er geen enkele abstracte machine meer te simuleren valt, omdat ze allemaal door seinpalen geblokkeerd zijn, de operatie

"V(SDAM)"

uitgevoerd wordt; DAM komt daardoor een slag aan bod en schakelt zichzelf door de operatie "P(SDAM)" voorlopig weer uit. Een dergelijke dummy-machine is logisch niet noodzakelijk, het maakt de hele bespiegelingen waarschijnlijk een stukje eenvoudiger. We hoeven dan niet in onze beschrijving er rekening mee te houden, dat de coordinator mogelijk geen abstracte machine vindt om de computer aan te delegeren. In het geval, dat de coordinator bv. altijd uitgevoerd wordt door dove X8 kunnen we in DAM de machine horend maken, wat wel eens prettig zou kunnen zijn.

Opm. Het is niet strict noodzakelijk, dat de operatie V(SDAM) slechts uitgevoerd wordt, als er werkelijk niets meer te doen is. Als het zo zou uitkomen, dat we van DAM een machine met minimale prioriteit maken, dan zou het best kunnen wezen, dat DAM bv. eens per paar seconde eens een slagje maakt. Zoiets zou kunnen voorkomen, als je aan abstracte machines twee prioriteiten zou toekennen, een statische prioriteit en een [illegible] effectieve prioriteit, waarbij de effectieve prioriteit, aanvankelijk van de statische afgeleid, langzaam groeit, als de abstracte machine steeds maar niet aan bod komt. Dit om het gevaar aan permanente verwaarlozing te bezweren. Als het eleganter uitkomt, om DAM hierin rustig mee te laten hobbelen, dan is daar dus helemaal niets tegen.

Over het feit, dat er wel enkele bezwaren aan kleven om de kamers van HAM een vaste omvang te geven, stappen we daar voorlopig even over heen. (Als we, zoals ik het me op dit ogenblik beslist voorstel, de kamers van HAM een vaste omvang zullen geven, dan is de vraag, die we voorlopig onbeantwoord laten of we de kamers zo groot moeten kiezen "dat ze altijd wel voldoende zullen zijn" -een riskante beslissing!- of dat we een organisatie moeten maken, waarin een gast zijn exces aan bagage in volgkamers of een centraal depot moet achterlaten. Wij signaleren het probleem nu slechts.)

De gasten in HAM vallen uiteen in twee gescheiden categorieen: geblokkeerde machines en onderbroken machines. Tijdens non-activiteit van de coordinator is er één machine, die in geen van beide categorieen valt, nl. de abstracte machine, die op dat ogenblik in actie is.

Indien de besturing ten gevolge van een interruptie in de coordinator terecht komt, komt de abstracte machine, die in actie was, in de groep der [illegible] onderbroken machines.

Indien de besturing ten gevolge van een P-operatie naar de coordinator verwezen wordt, dan geschiedt dit onder opgave van één of meer seinpalen. Als een van deze seinpalen non-positief is, dan gaat de machine, die in actie was, over naar de groep van de geblokkeerde, als alle opgegeven seinpalen positief zijn, worden zij alle met 1 verminderd en wordt de abstracte machine, die in actie was, opgenomen in de klasse der onderbroken machines.

De onderbroken machines zijn dus de machines, waaruit de coordinator liezen kan, de geblokkeerde machines zijn die, waarvan de coordinator geen notitie hoeft te nemen, zolang de bijbehorende seinpaal niet positief geworden is. Dit geschiedt door middel van de V-operatie en als een V-operatie uitgevoerd wordt, moet de coordinator dus in het geweer komen.

Als dit een V-operatie op een hardware seinpaal is, uitgevoerd door een autonoom apparaat, dan geschiedt dit met een ingreep als neveneffect, als de V-operatie, uiteraard door programma, op een geprogrammeerde seinpaal uitgevoerd wordt, dan doet dit programma zulks niet met een additieven uitopdracht maar door een (sub)routinesprong naar de coordinator. De zich in de V-operatie onderbrekende machine wordt in HAM in de klasse der onderbroken opgenomen; logisch zijn er immers geen bezwaren om deze machine na voltooiing der V-operatie weer gewoon door te laten werken.

Om de onderverdeling in klassen aan te geven, gaan we kettingen rijgen: elke kamer van HAM bevat ruimte voor een ander kamernummer. De abstracte machines, die onderbroken zijn, zijn in een of andere volgorde gerangschikt. Extern noteren we het nummer van de eerste onderbroken abstracte machine: in elke kamer noteren we het kamernummer van de volgende onderbroken machine en in de kamer van de laatste van de rij noteren we een als zodanig herkenbare marker.(Als blijkt, dat we vaak een nieuweling van het gezelschap aan het einde van de rij moeten toevoegen en de rij een appreciabele lengte kan krijgen, is het raadzaam om extern behalve het nummer van de eerste ook het nummer van de laatste van de ketting te noteren. Als we vaak "achter in de ketting" moeten opereren, kan je de ketting in twee richtingen doorloopbaar maken, door in elke kamer ook het nummer van de voorganger te noteren. Toevoegen en wegnemen wordt daardoor omslachtiger; ik dacht niet, dat deze "dubbele schakeling" nodig was. Het extern bijhouden van het nummer van de laatste is wat anders, want dat heeft nevenvoordelen: het maakt de marker overbodig.) Praemisse van deze techniek is, dat we stiekum toch een maximum omvang van HAM invoeren, nl. via het aantal bits, dat we in elke kamer voor het nummer van de volgkamer ter beschikking stellen; omdat dit aantal echter slechts logarithmisch oploopt kan je zonder ongehoorde verspilling een majorant aanhouden, die altijd safe is.

Voordat we nu de kettingen, waarin de geblokkeerde machines zijn opgenomen, nader gaan bespreken, is het goed even te overwegen, waarom we het idee der geblokkeerde machines eigenlijk hebben ingevoerd. We zouden zonder kunnen. Als een machine de P-operatie uitvoert, maar deze kan wegens non-positieve seinpaal niet beeindigd worden, dan zou je deze deze abstracte machine in de lijst van onderbroken machines kunnen opnemen, mits als "staat van vordering" een verwijzing naar het begin van de de P-operatie wordt opgenomen. Komt deze abstracte machine ooit weer aan het bod, dan zal hij wel opnieuw met frisse moed de P-operatie proberen. Dit heeft twee nadelen. Ten eerste is het niet ondenkbaar, dat de machine dan een aanwijsbaar deel van zijn tijd zoet brengt met het laten mislukken van P-operaties, ten tweede moet de bestrijding van de permanente verwaarlozing dan waterdicht zijn, omdat anders de computer al zijn tijd zou kunnen besteden aan mislukkende P-operaties, terwijl een zinvol proces geen doorgang zou vinden. Het invoeren van de klasse der geblokkeerde machines kunnen we dus zien als middel én om de efficiency op te voeren door dingen, die kennelijk tot mislukken gedoemd zijn, niet eens te proberen, én als middel om straks bij de strategie de handen vrij te hebben.

Laten wij nu de geblokkeerde abstracte machines bekijken, voorlopig onder de vereenvoudigende veronderstelling, dat elke P-operatie in een geblokkeerde machine slechts 1 seinpaal als argument meekrijgt.

Als de actieve machine nu een P-operatie tegenkomt, wordt de besturing naar de coordinator verwezen; als de seinpaal positief is, wordt de P-operatie voltooid en wordt de machine opgenomen in de klasse der onderbroken machine, anders is hij mogelijk een van de vele machines, die wachten moet, totdat de seinpaal in kwestie positief wordt.

Om nu, als de V-operatie wordt uitgevoerd op een seinpaal, te ontdekken, welke abstracte machines daardoor mogelijk naar de klasse der onderbroken machines kunnen worden overgevoerd, moet het niet nodig zijn om alle kamers van HAM af te zoeken om te kijken of er misschien ook een geblokkeerde machine juist op deze seinpaal stond te wachten. Dit zoekproces kunnen we weer met een ketting ondervangen.

Alle seinpalen zijn ondergebracht in het Hotel voor Seinpalen HS en worden door hun kamernummer in dit hotel geidentificeerd. In elke HS-kamer wordt, behalve de waarde van de seinpaal, genoteerd, of er inmiddels abstracte machines op het positief worden van deze seinpaal staan te wachten; zo ja, dan zijn deze abstracte machines in een ketting aan elkaar verbonden en de HS-kamer bevat (in elk geval) het HAM-kamernummer van de eerste van de reeks.

Wordt nu op een bepaalde seinpaal de V-operatie uitgevoerd, dan zijn de consequenties, die daaraan verbonden moeten worden, duidelijk. Als de seinpaal al positief was, dan hoeven we er enkel 1 bij te tellen; was hij aanvankelijk = 0, dan moeten we kijken, of er inmiddels abstracte machines op stonden te wachten, zo ja, dan kan één van deze machines uit deze ketting losgemaakt worden en in de ketting der onderbroken machines opgenomen worden. De strategische kwestie komt hier om de hoek kijken in het geval er meer dan 1 abstracte machine op de seinpaal in kwestie stond te wachten; in dat geval moet immers een keuze gemaakt worden.

Gecompliceerder wordt het, wanneer we bedenken, dat er ook programma's zullen zijn, die via P-operatie's met meer dan 1 seinpaal de controle aan de coordinator overgeven, en 1 of meer van de opgegeven seinpalen is op het moment non-positief. Een mogelijke oplossing is de volgende, in de veronderstelling, dat de HAM-kamer van deze machine groot genoeg is, om de HS-nummers van de bij de P-operatie meegegeven seinpalen te bevatten.

De abstracte machine in kwestie wordt opgenomen in de ketting van de eerste de beste seinpaal, die non-positief bevonden wordt. Wordt deze seinpaal later door een V-operatie positief en wordt dit programma uitgezocht als mogelijke candidaat voor deblokkade, dan wordt eerst het rijtje seinpalen, dat in zijn HAM-kamer staat opgegeven, getest. Alleen, als die alle positief zijn, kan de P-operatie voltooid worden, anders is het resultaat, dat de machine in een andere seinpaalketting opgenomen wordt. De ketting, die hangt aan de seinpaal, waarop de V-operatie was uitgevoerd, is daarmee 1 korter geworden; als hij daarmee niet leeg is geworden, wordt er opnieuw een schakel uit gekozen. Etc. etc. totdat de ketting leeg of de seinpaal weer = 0 is.

Het komt er dus op neer, dat men de candidaat voor deblokkade opnieuw zijn P-operatie uit laat voeren, de P-operatie, waarvan nu inmiddels de seinpalen in zijn HAM-kamer gespecificeerd staan. (Dat 1 van de tests daarbij overbodig is, omdat van minstens 1 seinpaal zeker bekend is, dat hij positief is, laten we voor wat het is; deze test onderdrukken zal meer tijd vergen dan hem even uitvoeren.)

We zien hier dus twee keren, twee situaties, waarin de seinpalen van een P-operatie onderzocht moeten worden. Ik zou ze willen noemen: de "aangeboden P-operatie en de gesuggereerde P-operatie", gesuggereerd nl. door het positief worden van een eerst

blokkerende seinpaal. Omdat het verleidelijk is, hun overeenkomst uit te buiten, is het goed even op het verschil te wijzen: als de coordinator met de aangeboden P-operatie klaar is, kan hij overgaan tot de orde van de dag, als de coordinator klaar is met een gesuggereerde P-operatie, dan moet de coordinator (in elk geval als de P-operatie tot een nieuwe blokkade aanleiding heeft gegeven) terug naar de suggerende seinpaal!

Organisatorisch is er nog een open kwestie, nl. of een HAM-kamer voor een abstracte machine inderdaad dit lijstje HS-nummers kan herbergen. Ook het Hotel voor Seinpalen heeft een maximum omvang, die tot uiting komt in het aantal bits, dat we voor een HS-kamernummer ter beschikking stellen. Aangezien P-operatie's willekeurig veel seinpalen mee kunnen krijgen, kan ik van te voren nooit garanderen, dat een HAM-kamer groot genoeg is. De beste oplossing onderscheidt waarschijnlijk twee soorten P-operaties, die met niet meer dan, zeg, drie seinpalen -dit is de overgrote meerderheid- en de P-operatie met meer seinpalen. Als de eerste nu maar zo vlot mogelijk verwerkt wordt en de tweede soort ook kan, dan hebben we het ene gedaan en het andere niet gelaten.

Een mogelijke oplossing voor de P-operatie met willekeurig veel seinpalen is de volgende: in de HAM-kamer neemt men niet [illegible] alleen het terugkeeradres op voor het geval van succesvol voltooide P-operatie, maar ook het terugkeeradres, dat verwijst naar het begin van de P-operatie. De aangeboden P-operatie werkt de seinpalen af, zonder deze in de HAM-kamer op te slaan; de gesuggereerde P-operatie kan men nu simuleren door de abstracte machine vlak voor zijn P-operatie weer te laten beginnen. De coordinator krijgt dan een P-operatie aangeboden, waarvan nog bekend moet zijn, dat dit een aangeboden P-operatie voorstelt. Ik suggereer deze oplossing met schroom om twee redenen:

a) proberen de P-operatie te voltooien gaat op deze manier wel vrij wat tijd kosten; de overweging, dat P-operatie's met zoveel seinpalen wel niet zoveel zullen voorkomen wordt een beetje teniet gedaan door de overweging, dat juist bij een P-operatie met veel seinpalen de kans op mislukking groot is.
b) dit loopt logisch lekker, als het programma, waarin de P-operatie voorkwam, nog op de kernen aanwezig was; zo niet, dan kon dit spelletje, speciaal vanwege de speciale toestand van de coordinator, die immers onthouden moet, dat het hier eigenlijk om een gesimuleerde gesuggereerde P-operatie gaat, wel eens naar worden.

Een alternatieve oplossing is om een abstracte machine meer HAM-kamers ter beschikking te geven: de eerste keer, dat een abstracte machine met een P-operatie met zo'n lange waslijst aankomtzetten, zal je de extra moeite moeten doen, om een of meer extra HAM-kamers als volgkamers te reserveren. Ik geloof niet, dat het verstandig is om na deze veeleisende P-operatie,die "volgkamers"weer vrij te geven: je kunt er beter van uitgaan, dat deze abstracte machine nu een keer zo veeleisende P-operatie's aanbiedt en dat we dus de reservering in HAM maar permanent zo moeten laten, totdat de abstracte machine volledig wordt afgevoerd.

Wat betreft de strategie voor het overvoeren van geblokkeerd naar onderbroken denk ik, dat je vrij goed zit, wanneer je bij blokkade van een aangeboden P-operatie de abstracte machine aan het einde van de ketting, bij herblokkade van de gesuggereerde P-operatie de abstracte machine aan het begin van de ketting van de blokkerende seinpaal onderbrengt en vervolgens bij de V-operatie de aan de seinpaal hangende abstracte machines in volgorde van begin naar eind probeert te deblokkeren. Dit is de goedkoopste versie van "Wie het eerst komt, die het eerst maalt". Dit sluit permanente verwaarlozing van een machine, die is blijven hangen in een P-operatie met meer dan 1 seinpaal niet uit, nl. als door andere [illegible] abstracte machines om deze seinpalen gevochten wordt. Kunnen we hier zeggen "Het zij zo" en concluderen, dat deze machine dus incompatibel is met de rest?