CH 633 643 A5

(19) SCHWEIZERISCHE EIDGENOSSENSCHAFT
BUNDESAMT FÜR GEISTIGES EIGENTUM

(51) Int. Cl.³: G 06 F 9/46

Erfindungspatent für die Schweiz und Liechtenstein
Schweizerisch-liechtensteinischer Patentschutzvertrag vom 22. Dezember 1978

ETH-BIBLIOTHEK ZÜRICH

(12) PATENTSCHRIFT A5

(11) 633 643

(21) Gesuchsnummer: 10249/78

(22) Anmeldungsdatum: 03.10.1978

(30) Priorität(en): 11.10.1977 US 841163

(24) Patent erteilt: 15.12.1982

(45) Patentschrift veröffentlicht: 15.12.1982

(73) Inhaber:
International Business Machines Corporation, Armonk/NY (US)

(72) Erfinder:
Paul Rudolf Roever, Sunnyvale/CA (US)

(74) Vertreter:
Dipl.-Ing. Wolfgang Möhlen, c/o IBM Corp., Rüschlikon

(54) Verfahren zur blockierungsfreien verzahnten Simultanverarbeitung mehrerer Aufgaben mittels mehrerer Programme in einer Datenverarbeitungsanlage.

(57) Die Wartebeziehungen zwischen N Aufgaben in verzahnter Simultanverarbeitung einer Datenverarbeitungsanlage werden als Vektor mit N+1 Feldern ausgedrückt. Der Vektor gibt an, welche Aufgaben aktiv sind bzw. auf welche anderen Aufgaben sie warten. Der Wert p in Feld r bedeutet, dass Aufgabe r auf Aufgabe p wartet. Ein Wert j der Werte 0,1,2..., N+1 ist dazu bestimmt, eine aktive Aufgabe zu bezeichnen. Der Wert im Feld j hat stets die Grösse j. Eine Blockierung innerhalb der Wartebeziehungen wird durch wiederholte Übersetzung der Vektorfelder in und mit sich selbst in höchstens $[\log_2(N+1)]$ Iterationen festgestellt, wobei $[\log_2(N+1)]$ die kleinste ganze Zahl gleich oder grösser dem Basis-zwei-Logarithmus von (N+1) ist. In jede Registerstelle r, die eine Hinweisadresse p enthält, wird dabei der Inhalt g der Registerstelle p gesetzt. Wenn schliesslich nicht in allen Registerstellen der Hinweisadressenwert j steht, ist dies ein Anzeichen für eine vorliegende Blokkierung.

PATENTANSPRÜCHE

1. Verfahren zur blockierungsfreien verzahnten Simultanverarbeitung mehrerer Aufgaben mittels mehrerer Programme in einer Datenverarbeitungsanlage, wobei jede zu lösende Aufgabe auf höchstens eine andere Aufgabe warten muss, gekennzeichnet durch folgende Schritte:
- Bildung eines Feldes aus N+1 Hinweisadressen in einem Register, welche zusammen mit ihrer Stellung im Register die Wartebeziehungen zwischen N Aufgaben angeben, wobei einer Registerstelle j der Hinweisadresswert j zugeordnet ist;
- Übersetzung des Feldes mit sich selbst derart, dass in jede Registerstelle r, die eine Hinweisadresse p enthält, der Inhalt g der Registerstelle p gesetzt wird, wobei r, p und g grösser oder gleich Null und kleiner oder gleich N sind;
- Prüfung ob das Register eine nicht-j-Hinweisadresse enthält;
- bei Feststellung wenigstens einer nicht-j-Hinweisadresse Wiederholung der Übersetzung und der Prüfung, bis die festgestellten nicht-j-Hinweisadressen von einer Iteration zur nächsten unverändert bleiben, und
- Anzeige dieses Resultates als Vorliegen einer Blockierung.

2. Verfahren nach Anspruch 1, wobei jede Aufgabe einen aktiven Zustand oder einen Wartezustand annehmen kann, gekennzeichnet durch
- anfordern eines bestimmten Verarbeitungsmittels durch eine Aufgabe;
- feststellen, ob das Mittel zur Verfügung steht und wenn dies der Fall ist, Reservieren des Mittels für die anfordernde Aufgabe, andernfalls
- prüfen, ob ein Versetzen der Aufgabe in den Wartezustand eine Blockierung mit anderen Aufgaben erzeugt und, wenn dies der Fall ist,
- rückstellen und neu starten einer der Aufgaben, die den Blokkierungskreis bilden.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass bei der Bildung eines Feldes von N+1 Hinweisadressen in einem Register die Aufgaben-Hinweisadressen in den Registerpositionen derart zugeordnet werden, dass für jede Aufgabe x, die auf eine Aufgabe y wartet, x eine höhere Registerposition einnimmt als y, wobei die Registerposition 0 den Wert 0 zugeordnet hat.

4. Verfahren nach Anspruch 3, dadurch gekennzeichnet, dass es zur Feststellung einer Blockierungsschleife höchstens einmal wiederholt wird.

Die Erfindung bezieht sich auf die Steuerung gleichzeitig ablaufender Datenverarbeitungsaufgaben mittels mehrerer Programme und insbesondere auf die Feststellung blockierter Aufgaben unter den auf die Verarbeitung wartenden Aufgaben.

Der Ausdruck Datenverarbeitungsanlage bezieht sich im nachfolgenden auf eine zentrale Verarbeitungseinheit mit Hauptspeicher, Eingangs- und Ausgangskanälen, Steuereinheiten, Direkt-Zugriffsspeicher und anderen mit ihr verbundenen Einheiten, wie sie beispielsweise im US-Patent 3 400 371 beschrieben ist. Eine Aufgabe soll nachfolgend eine unabhängige Arbeitseinheit bezeichnen, die die Ausführungs- oder Verarbeitungsmittel der Datenverarbeitungsanlage beansprucht. Ein Aufgabensteuerblock ist die Zusammenfassung von Steuerinformation, die sich auf eine Aufgabe bezieht. Sie enthält u.a. die vom Benützer zugeordnete Priorität sowie den Zustand der Aufgabe, d.h. aktiv oder wartend. Im Wartezustand wartet eine Aufgabe auf die Ausführung anderer Aufgaben, worauf sie dann aktiv wird.

Unter Verarbeitungsmittel wird nachfolgend jeder Teil einer Datenverarbeitungsanlage oder eines Systemprogrammes verstanden, der für die Ausführung einer Aufgabe notwendig ist. Dazu gehören der Hauptspeicher, Eingangs-Ausgangs-Vorrichtungen, die zentrale Verarbeitungseinheit, Datensätze und Steuer- oder Verarbeitungsprogramme. Ein Systemprogramm besteht in dieser Hinsicht aus einer Anzahl von Überwachungsroutinen, die auf der Datenverarbeitungsanlage ablaufen, um z.B. sicherzustellen, in welcher Reihenfolge anstehende Aufgaben erledigt werden, dass Datensätze und Programme auch für lange Zeit ordnungsgemäss gespeichert werden und dass sie vor unerwünschtem Zugriff oder unerwünschter Benützung geschützt sind. Zudem haben sie gewisse Haushaltungs- und Überwachungsfunktionen.

Verzahnte Datenverarbeitung (Multiprogramming), die sich auf die gleichzeitige Ausführung mehrerer Programme durch eine Datenverarbeitungsanlage bezieht, ist beispielsweise mittels eines IBM System/360 Operating System möglich, wie es die IBM-Druckschrift GC28-6646, Form Nr. GA22-6822 beschreibt. Diese modernen Betriebssysteme erlauben es, die Mittel der Anlage wesentlich besser auszunützen. Wenn während der Ausführung eines Programmes ein Unterbruch entsteht, beispielsweise um zusätzliche Daten in die Zentraleinheit einzulesen, kann während dieser Zeit an einer anderen Aufgabe weitergearbeitet werden. Die Zentraleinheit kann also ein anderes Programm ausführen oder mittels desselben Programmes eine andere Aufgabe ausführen.

Beim Warten auf nacheinander zu benützende Mittel der Anlage kann eine Aufgabe dann blockiert sein und unendlich lange warten müssen, wenn sie in einer Warteschleife zusammen mit anderen Aufgaben festgefahren ist. Unter diesen Umständen hat jede Aufgabe für sich ein nicht vorberechtigtes Verarbeitungsmittel reserviert, das von einer anderen Aufgabe zur Lösung benötigt wird, z.B. wartet jede Aufgabe in der Schleife darauf, dass eine andere Aufgabe ihre Reservation abgibt. Die Anzeichen der Blockierung sind also gegenseitiger Ausschluss, mangelnde Vorberechtigung und Warten auf Verarbeitungsmittel. Gegenseitiger Ausschluss bedeutet, dass eine Aufgabe ausschliessliche Herrschaft über die nötigen Verarbeitungsmittel beansprucht. Nicht Vorberechtigung bedeutet, dass eine Aufgabe ein Verarbeitungsmittel nicht freigibt, bevor seine Benützung desselben beendigt ist. Warten auf Verarbeitungsmittel schliesslich entsteht, weil jede Aufgabe Verarbeitungsmittel reserviert hat, während sie darauf wartet, dass andere Aufgaben ihre reservierten Verarbeitungsmittel freigeben.

Verschiedene Vorschläge zur Vermeidung von Blockierungen sind schon bekannt, vergleiche Coffmann und Denning «Operating Systems Theory», 1973, Prentice Hall, Seite 46. Die Vorschläge beziehen sich auf Verhütung, Entdeckung und Beseitigung sowie Vermeidung. Für nähere Ausführungen vergleiche A.C. Shaw, «The Logical Design of Operating Systems», 1974, Prentice Hall, Seiten 227–232 sowie 215–224. Shaw stellt fest, dass zur Vermeidung von Blockierungen die Anlage im allgemeinen so eingeschränkt wird, dass zu jeder Zeit nur eine Aufgabe überhaupt Verarbeitungsmittel benützen kann. Verzahnte Verarbeitung kann also zugelassen werden, wenn jede Aufgabe die benötigten Verarbeitungsmittel bei ihrer Entstehung meldet. Aufgaben mit ihnen zugeordneten Mitteln könnten dann nicht blockiert werden, weil sie nicht weitere Mittel beanspruchen können und diese daher schliesslich wieder freigeben. Der Nachteil dieses Verfahrens liegt darin, dass es voraussetzt, dass das Ausmass und die Reihenfolge der benötigten Mittel zum voraus vollständig festgelegt werden kann. Auch werden die Mittel dadurch für unnötig lange Zeiten festgelegt. Eine Aufgabe kann z.B. die Verarbeitungsmittel in der Reihenfolge a, b, c verlangen, wenn tatsächlich die Reihenfolge b, c, a erforderlich ist. Auch ist das Mittel c vielleicht nur im letzten Teil der Arbeitszeit erforderlich. Zur Beseitigung einiger dieser Nachteile vergleiche J.W. Havender, «Avoiding

Deadlock in Multitasking Systems», 1968, IBM Systems Journal, Seite 74-84, wo eine «geordnete» Verarbeitungsregel angegeben wird. Nach Coffmann und auch Shaw ist es wichtig, solche Blockierungen in der Anlage festzustellen. In seiner Schrift «System Deadlock», Computing Surveys, Juni 1971, Seiten 67-78, untersucht Coffmann die Warterelationen zwischen den Aufgaben einschliesslich derer, in welchen Aufgaben auf zwei oder mehr Aufgaben warten. Im allgemeinen Fall ist nach Coffmann die nötige Zeit, um eine Blockierung bei N Aufgaben zu entdecken, proportional N^2. Zudem ist die Entdeckungszeit eine lineare Funktion von N+(R·log R), wobei (R log R) die Zeit ist, die man benötigt, um R Verarbeitungsmittel zu sortieren. Schliesslich erfordert die Coffmannische Methode die Kenntnis der zur Entdeckungszeit vorhandenen Mittel zusätzlich zu Angaben betreffend die Aufgabenidentität, und ihre Warterelationen.

Obermarck beschreibt im IBM Technical Disclosure Bulletin, Band 12, Seiten 2338-2339, die Benützung einer Matrix, in der die Zeilen und Spaltenkoinzidenz zwischen einem Anfrager und einem Mitteleigentümer zusammen mit einem Tabelleneintrag die Blockierung bestimmen. Auch P. Roever benützt eine Matrix, jedoch nur wenn die Aufgaben bereits in einer Schleife A → B → C → D → A warten, vergleiche IBM Technical Disclosure Bulletin, Band 16, Seiten 1243-1244.

Es ist ein Ziel der vorliegenden Erfindung, Blockierungen unter N Aufgaben in einer Datenverarbeitungsanlage für verzahnte Simultanverarbeitung festzustellen. Dabei sollen hauptsächlich Angaben über die Warterelationen unter den Aufgaben benützt werden und weniger Information betreffend die Verarbeitungsmittel. Wo die Blockierungsentdeckung iterative Operationen benötigt, ist es ein Ziel der Erfindung, die Zahl der Iterationen klein zu halten.

Die Ziele können erreicht werden durch ein Verfahren, in welchm die Warterelationen zwischen N Aufgaben so beschränkt sind, dass jede Aufgabe auf höchstens eine weitere Aufgabe wartet. Dazu wird ein Feld von N+1 Hinweisadressen in einem Register erstellt, wobei die Hinweisadressen und ihre Stellung im Register die Wartebeziehungen zwischen den Aufgaben angeben und wobei einer beliebigen Registerstelle j der Hinweisadresswert j zugeordnet ist. Im nächsten Schritt wird das Feld mit sich selbst derart übersetzt, dass in jede Registerstelle r, die eine Hinweisadresse p enthält, der Inhalt g der Registerstelle p gesetzt wird, wobei r, p und g grösser oder gleich Null und kleiner oder gleich N sind.

In den nachfolgenden Schritten werden die Register überprüft, um eine nicht-j-Hinweisadresse festzustellen und, sofern wenigstens eine solche Adresse festgestellt wird, die Schritte des Übersetzens und Feststellens zu wiederholen, bis der Satz von nicht-j-Hinweisadressen von einem Schritt zum nächsten unverändert bleibt. Dieses Resultat zeigt eine Blockierung an.

Die Erfindung soll nachfolgend anhand von Ausführungsbeispielen im einzelnen beschrieben werden. Die folgenden Beispiele basieren auf Instruktionen für IBM-System-360-Maschinen, wie sie im eingangs genannten US-Patent 3 400 371 erwähnt sind.

Beispiel 1

Es sei angenommen, dass die Aufgaben a, b, c, d, e, f, g folgendermassen aufeinander warten:

a → b → c → d → e → f → g j

Im Registerfeld wird das wie folgt dargestellt:

Hinweisadressen	j b c d e f g j	= t(x)
Position	j a b c d e f g	

Position j ist die beliebig gewählte Position mit dem Hinweisadresswert j, der als Adresswert unter anderem angibt, dass die ihm zugeordnete Aufgabe aktiv ist, d.h. Aufgabe g wartet nicht auf eine andere Aufgabe. Die N Aufgaben heissen a, b, c ... f und die (N+1)-ste Aufgabe ist Aufgabe j. Wird nun das Feld mit sich selbst $[\log_2(N+1)]$ mal übersetzt, dann zeigt ein Wert der von j verschieden ist eine Schleife an. Da jedoch bekannt ist, dass eine Schleife nicht vorliegt, sollte nach $[\log_2(N+1)]=\log_2 8=3$ Übersetzunen, der Vektor ausschliesslich Werte enthalten, die gleich j sind.

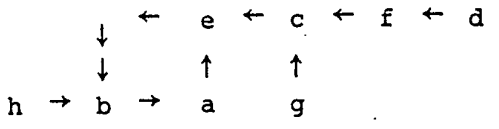
Hinweisadressen	j c d e f g j j	= $t^2(x)$ 1. Übersetzung
Position	j a b c d e f g	
Hinweisadresse	j e f g j j j j	= $t^3(x)$ 2. Übersetzung
Position	j a b c d e f g	
Hinweisadresse	j j j j j j j j	= $t^4(x)$ 3. Übersetzung
Position	j a b c d e f g	

Die Übersetzung eines jeden Feldes $t^i(x)$ nach $t^{i+1}(x)$ wird für jede Registerposition r, die eine Hinweisadresse p enthält, vorgenommen durch Substitution des Inhalts g der Position p als neuer Inhalt der Position r, wobei r, p, g grösser oder gleich Null und kleiner oder gleich N sind.

Wird dieses Prinzip auf die verschiedenen Übersetzungen des Beispiels 1 angewendet, so zeigt sich im Feld t(x), dass die Position «a» die Hinweisadresse «b» enthält. Position b jedoch enthält den Wert «c». Der neue Wert, der im Feld $t^2(x)$ in Position «a» kommt, ist also «c». Dadurch verschwindet die Hinweisadresse «f» von Position «e» in t(x). Der neue Wert «g», der in $t^2(x)$ nach «e» kommt, ist der Wert, der in t(x) in Position «f» stand. Die Substitutionen sind leicht zu verfolgen und ergeben das Resultat j in der 3. Übersetzung.

Beispiel 2

Angenommen die Aufgaben a, b, c ... h warten folgendermassen aufeinander:

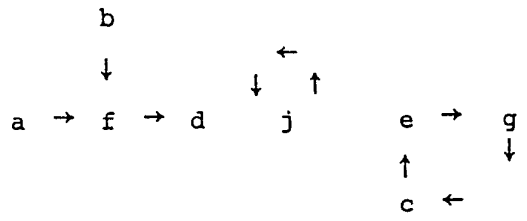


Hinweisadressen	e a e f j b c c b	= t(x)
Position	a b c d j e f g h	
Hinweisadressen	b b b c j b b b b	= $t^2(x)$ 1. Übersetzung
Position	a b c d j e f g h	
Hinweisadressen	b b b b j b b b b	= $t^3(x)$ 2. Übersetzung
Position	a b c d j e f g h	
Hinweisadressen	b b b b j b b b b	= $t^4(x)$ 3. Übersetzung
Position	a b c d j e f g h	
Hinweisadressen	b b b b j b b b b	= $t^5(x)$ 4. Übersetzung
Position	a b c d j e f g h	

In diesem Beispiel ist N+1=9 und $[\log_2(N+1)]=4$. Position j wurde absichtlich mitten in die Reihe der Registerpositionen eingesetzt, um zu zeigen, dass ihre beliebige Stellung das Resultat nicht beeinflusst. Bei der zweiten Übersetzung zeigt das Auftreten eines ungleich-j-Resultates eine Schleife an. Dabei ist interessant, dass alle Positionen ungleich-j-Hinweisadressen enthalten. Weiter ist zu bemerken, dass das nach der zweiten Übersetzung vorliegende Resultat in nachfolgenden Übersetzungen nicht ändert und somit $[\log_2(N+1)]$ ein oberer Grenz-

wert und nicht notwendigerweise die Anzahl der Übersetzungen darstellt.

Beispiel 3



Hinweisadressen	j f f e j g d c	= t(x)
Position	j a b c d e f g	
Hinweisadressen	j d d g j c j g	= t^2(x) 1. Übersetzung
Position	j a b c d e f g	
Hinweisadressen	j j j g j g j g	= t^3(x) 2. Übersetzung
Position	j a b c d e f g	
Hinweisadressen	j j j g j g j g	= t^4(x) 3. Übersetzung
Position	j a b c d e f g	

Dieser Fall zeigt zwei Gruppen von Aufgaben, wovon lediglich die Aufgaben c, e und g in einer Warteschleife stehen. Für N+1=8 wird also $\log_2(N+1)=3$. Ungleich-j-Hinweisadressen sind nach der dritten Übersetzung stabilisiert. Offensichtlich ist Aufgabe g wenigstens ein Glied in der Warteschleife.

In einem Betriebssystem, das die Zuordnung von Verarbeitungsmitteln zu Aufgaben überwacht, ist bekannt, Blockierungen zu vermeiden, durch prüfen, welche Auswirkungen es hat, wenn eine Aufgabe, die ein nicht erhältliches Verarbeitungsmittel verlangt, in den Wartezustand versetzt wird. Entsteht daraus eine Blockierung dann geht die Aufgabe nicht weiter.

Eine Abwandlung der Methode bewirkt, dass eine Schleife mittels einer geringeren Zahl von Übersetzungen entdeckt wird, indem die Hinweisadressen, welche bestimmte Aufgaben angeben, bestimmten Registerpositionen derart zugeordnet werden, dass für jedes Paar von Aufgaben p, q, worin eine Aufgabe p auf eine Aufgabe q wartet, der Aufgabe q eine numerisch niedrigere Position als der Aufgabe p im Register zugeordnet wird. Die Positionszuordnung wird vor der Übersetzung vorgenommen. Im vorliegenden Verfahren wird j immer der Wert 0 zugeordnet. Die Registerposition 0 hat daher stets den Wert 0.

Betrachtet man diese Abwandlung genauer, so zeigt sich, dass für alle Paare von Aufgaben p, q, worin p auf q wartet, folglich p eine niedrigere Position wie q hat. Nun können folgende Schritte unternommen werden, um eine Blockierung festzustellen, wenn eine zuvor aktive Aufgabe A auf eine Aufgabe B wartet. Zunächst wird festgestellt, ob die Aufgabe B eine niedrigere Registerposition als A einnimmt. Ist dies der Fall, dann kann keine Blockierung entstehen, weil die Kette von Hinweisadressen sich nicht schliessen kann. Demzufolge darf die Aufgabe A auf die Aufgabe B warten und ihre Position wird auf die Position der Aufgabe B hinweisen.

Hat jedoch Aufgabe B eine numerisch höhere Position als Aufgabe A, dann wird zwecks Entdeckung einer Blockierung die Hinweisadresszahl von A vorübergehend als Wert in die Position von A eingesetzt. Nun ergibt eine einzige Übersetzung des Registerinhalts mit sich selbst ein stabiles Resultat. Der Grund dafür liegt darin, dass alle Registerpositionen Werte enthalten, die auf niedrigere Positionen hinweisen. Die Übersetzung, die nun von niedrigeren zu höheren Registerpositionen fortschreitet, erfolgt daher mit grösster Wirkung. Als nächstes muss durch Prüfung des Hinweiswertes in der Position der Aufgabe B festgestellt werden, ob eine Blockierung vorliegt (sofern in Position von A feststellbar) oder nicht (sofern 0). Zeigt sich keine Blockierung so darf Aufgabe A auf Aufgabe B warten, nachdem die Aufgabenpositionen neu zugeordnet wurden so, dass wiederum für jedes Paar p, q von Aufgaben, worin A und B enthalten sind und wobei p auf q wartet, q eine niedrigere Position einnimmt als p.

Nachfolgend soll die Funktion der Übersetzungsinstruktion (Translate) erläutert werden. Das Verfahren kann auf einem IBM System/360 Digitalrechner angewendet werden. Ein derartiger Rechner ist im schon genannten US-Patent 3 400 371 beschrieben. Die Translate Instruktion ist dort in Kolonne 90, Zeile 25–52 erwähnt. Sie benützt ein Format, das in Kolonne 88, Zeile 16–56 erwähnt ist. Die zugehörigen Microcode Logic ist in Figuren 5NA1 bis 5NA3 dargestellt.

Die Translate Instruktion als Funktion auf Maschinenebene ist festgelegt in: IBM 360 und 370, Principles of Operation, IBM-Druckschrift GA22-6821-8 und GA22-7000-4, 9. Ausgabe, November 1970 respektive 4. Ausgabe, September 1974. Die Instruktion hat folgenden Aufbau:

Field	TR OP. Code	List Size	Addn of Operand 2		Addn of Operand 1		
Byte Position	0	8	16	20	32	36	47

Die Acht-Bit-Bytes des ersten Operanden werden durch die Adresse des zweiten Operanden als Argumente zum Hinweis auf die Liste bezeichnet. Jedes Acht-Bit-Funktionsbyte aus der Liste ersetzt das entsprechende Argument im ersten Operanden. Die Bytes des ersten Operanden werden eins um das andere zur Übersetzung ausgewählt, von links nach rechts fortschreitend. Jedes Argumentbyte wird zur ursprünglichen Adresse des zweiten Operanden addiert. Die Addition folgt den Regeln der Adressenarithmetic, wobei das Argumentbyte, ein Acht-Bit-Byte, als ganze Zahl ohne Zeichen behandelt wird. Die Summe bildet die Adresse des Funktionsbytes, welche dann das ursprüngliche Argumentbyte ersetzt.

Bevor das Ausführungsbeispiel im einzelnen beschrieben wird, muss die Frage gestellt werden, worin die wesentliche Eigenschaft oder Natur des iterativ angewendeten Übersetzungsschritts liegt, der immer in sich geschlossen sein sollte. Es ist festzustellen, dass eine Übersetzungstabelle, sofern keine ihrer Werte ausserhalb des Bereichs des Arguments liegt, eine endliche Funktion beschreibt, deren Potenzen sämtlich definiert sind. Der Ausdruck Potenzen einer Funktion bezieht sich auf die iterative Anwendung der Funktion, z.B. wenn f(f(f(x))), kurz geschrieben wird als f^3(x). Für solche Funktionen folgt für jeden Wert von p innerhalb des Argumentenbereichs, dass die Sequenz f(p), f^2(p), f^3(p)... periodisch wird für $i \geq N$, wobei N die Länge des Argumentbereiches ist.

Nach m Übersetzungen eines Feldes, das eine endliche Funktion von f darstellt, mit und in sich selbst, wird die Position p enthalten $f^{[2m+e]}(p)$, wobei $e \geq 0$.

Da $f(p) = f^{[2^e+0]}(p)$, ist der Ausdruck korrekt für m=0.

Der Herleitungsschritt von m nach m+1 kann durch ein Beispiel gezeigt werden. Es sei angenommen, dass die Gleichung richtig ist für m = 2, so dass in Position p steht $f^{[4+a]}(p)$ = q und in Position q steht $f^{[4+b]}(q)$ = r. Nun ist ersichtlich, dass die nächste Übersetzung (m = 3) p ergeben wird, mit $f^{[8+c]}(p)$. Wenn nun $p \leq q$, so ergibt sich wenn Position p übersetzt wird $f^{[4+b]}f^{[4+a]}(p) = f^{[8+c]}(p)$. Ist jedoch $q < p$, dann wurde Position q bereits übersetzt und enthält $f^{[8+d]}(q)$ sofern Position p verar-

beitet wurde. p enthält also $f^{[8+d]}f^{[4+a]}(p) = f^{[8+4+e]}(p) = f^{[8+c']}(p)$.

Die Werte, die die Sequenz f(p), $f^2(p)$... annimmt, nachdem sie periodisch geworden ist, werden Endwerte der Position p genannt. Zu dieser Definition kann festgestellt werden, dass nach $\log_2(N)$ Übersetzungen einer Tabelle mit und in sich selbst alle Positionen ihre Endwerte anzeigen. Dies folgt daraus, dass entsprechend der ersten Feststellung $\log_2(N)$ Übersetzungen in einer Position p den Wert $f^{[N+e]}(p)$ erzeugen und dass $f^i(p)$ periodisch ist wenn $i \geqq N$.

Für die Feststellung von Blockierungen können Übersetzungsfelder oder Tabellen so benutzt werden, dass jede Aufgabe p, die auf eine Aufgabe t wartet, ausgedrückt wird als f(p)=t. Eine aktive Aufgabe p kann ausgedrückt werden als f(p)=j, wobei für j=0 dann f(p)=0 wird. Daraus können die Aufgaben auf die direkt gewartet wird und auch die Aufgaben auf die indirekt gewartet wird für eine Aufgabe p festgestellt werden als f(p), $f^2(p)$, $f^3(p)$.

Nach dieser Konvention ist eine Aufgabe dann aktiv oder wartet, möglicherweise indirekt, auf eine aktive Aufgabe dann und nur dann, wenn ihr einziger Endwert j beträgt. Der Wert j kann = 0 gesetzt werden. Das gilt für aktive Aufgaben weil j = f(p) = $f^2(p)$ = $f^3(p)$ = ... Es gilt ebenfalls für Aufgaben p, die auf eine aktive Aufgabe q warten, da für manche Werte i, $f^i(p) = q$ und $j = f^{[i+1]}(p) = f^{[i+2]}(p) = \ldots$

Da gemäss Definition eine Aufgabe dann und nur dann blockiert ist, wenn sie nicht aktiv ist und nicht auf eine aktive Aufgabe wartet, kann gesagt werden, dass eine Aufgabe dann und nur dann blockiert ist, wenn ihr einziger Endwert j beträgt. Wie oben gezeigt kann dies durch $\log_2(N+1)$ Übersetzungen festgestellt werden. Eine Aufgabe ist dann und nur dann nicht blockiert, wenn nach $\log_2(N+1)$ Übersetzungen ihre Position ein j Resultat enthält.

Wenn umgekehrt nach $\log_2(N+1)$ Übersetzungen eine Position p einen Wert q, der verschieden von 0 ist enthält, ist die Aufgabe p blockiert und der Wert q ist einer ihrer Endwerte. Dies bedeutet, dass die Aufgabe p wartet, gegebenenfalls indirekt, auf die Aufgabe q und dass für gewisse i und d gilt $q = f^i(p) = f^{[i+d]}(p)$. Da $f^i(p) = q$, gilt auch $f^d(q) = q$, was bedeutet, dass q auf sich selbst wartet, eventuell indirekt. In anderen Worten, die Aufgabe p ist entweder ein Glied einer Warteschlaufe oder sie wartet auf ein anderes Glied derselben. Die Aufgabe q ist Glied der Warteschlaufe. Obwohl eine Aufgabe mehrere Endwerte haben kann, warten zwei Aufgaben die denselben von null verschiedenen Endwert haben beide, eventuell indirekt, auf dieselbe Aufgabe, da f eine Funktion ist.

Wie schon erwähnt ist es vorteilhaft, das vorliegende Verfahren als Sequenz von Instruktionen in einer Datenverarbeitungsanlage als Teil eines Betriebs- oder Datenbanksystems anzuwenden. Darauf wird nun näher eingegangen.

Ein Datenbank-Überwachungssystem kann zusammen mit einem Betriebssystem benützt werden, das die üblichen Möglichkeiten der Datenverwaltung sowie der Verwaltung von Zusatzspeichern aufweist und die üblichen Programmsprachenkompiler enthält. Ein solches System ist die Konversationelle Überwachung, wie sie in der IBM-Druckschrift «Virtual Machine Facility/370; Command Language Guide», No. GC20-1804, 5. Auflage, May 1975, beschrieben ist.

Ein Datenbanksystem kann zur Transaktionsüberwachung eingerichtet sein. Dazu gehört beispielsweise die Zuteilung von Verarbeitungsmitteln sowie deren Zulassung und Wiederbereitstellung. Auch eine Sperrenüberwachung ist vorhanden. Die Sperrenüberwachung prüft die Verträglichkeit erwünschter Sperren. Die Blockierungsüberwachung bezieht sich lediglich auf die Feststellung blockierter Aufgaben. Beide wirken dahin, dass Aufgaben der Reihe nach behandelt werden durch Steuerung und Freigabe der seriell benützbaren Verarbeitungsmittel pro Aufgabe. Im zuvor genannten CMS-Betriebssystem geschieht diese Steuerung durch die beiden Makroinstruktionen ENQ und DEQ.

Die ENQ-Makroinstruktion veranlasst das Steuerprogramm, die seriell arbeitenden Verarbeitungsmittel der Reihe nach den Aufgaben zur Verfügung zu stellen. Jedes Verarbeitungsmittel ist mit Namen bezeichnet. Das Steuerprogramm bringt den Namen mit einem bestimmten Verarbeitungsmittel in Verbindung. Der Zugriff zum Verarbeitungsmittel ist daher rein logisch und nicht physisch beschränkt. Das bedeutet, dass die Aufgaben ein seriell arbeitendes Mittel ohne die ENQ-Makroinstruktion benützen können und dabei die Zuverlässigkeit des Programms in Frage stellen.

Wenn eines der Verarbeitungsmittel nicht zur Verfügung steht, weil es in einer ausschliessenden ENQ-Anforderung und nicht in einer nachfolgenden DEQ-Anforderung genannt worden war und wenn es sich um eine unbedingte Anforderung handelte, dann wird die aktive Aufgabe in den Wartezustand versetzt, bis alle angeforderten Verarbeitungsmittel zur Verfügung stehen. Sofern die ENQ-Anforderung bedingt ist, kommt die aktive Aufgabe sofort an die Reihe. Wenn die Benützung eines Mittels symbolisch einer Aufgabe zugeordnet ist, bleibt sie bei dieser Aufgabe bis eines der Programme der Aufgabe eine DEQ-Instruktion enthält, die dasselbe Mittel anfordert.

Die ENQ-Makroinstruktion kann also dazu benützt werden, den Zustand eines Verarbeitungsmittels festzustellen. Das heisst festzustellen, ob es unmittelbar zur Verfügung steht oder benützt wird und ob eine Benützungsanforderung für die aktive Aufgabe in einer anderen ENQ-Makroinstruktion bereits vorliegt.

Die DEQ-Makroinstruktion wird dazu benützt, die Benützungsanforderung einer aktiven Aufgabe für ein oder mehrere seriell benützbare Verarbeitungsmittel zu beseitigen. Sie kann auch benützt werden, um festzustellen, ob die Anforderung einer aktiven Aufgabe vorliegt oder bewilligt wurde. Ein dazu vorgesehenes Register kann auf null gesetzt werden, wenn die Anforderung befriedigt wurde. Wenn in der Ausführung im Datenbanksystem eine Aufgabe ein Verarbeitungsmittel anfordert und der Anforderung nicht unmittelbar genügt werden kann, wird überprüft, ob die anfordernde Aufgabe in eine Blokkierung gerät, wenn sie in Wartezustand versetzt wird. Ist dies der Fall, dann muss das Blockieren vermieden werden, indem z.B. die anfordernde Aufgabe zurückgeschaltet und neu gestartet wird.

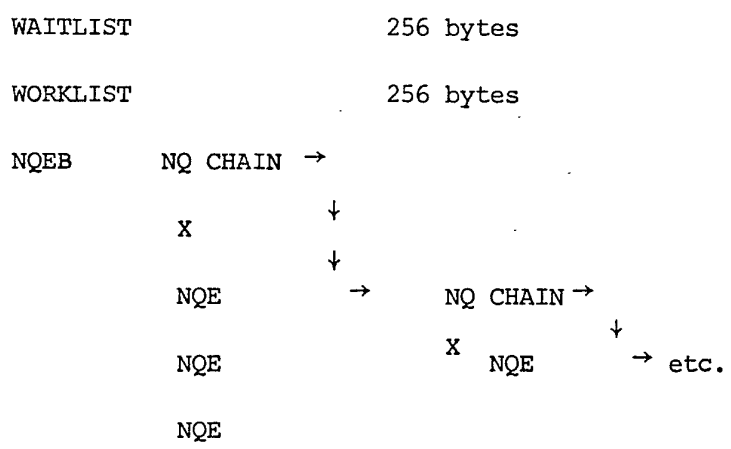
Nun soll auf die Sperren und Freigabeüberwachung eingegangen werden. Die Aufgabenanforderung eines Verarbeitungsmittels wird durch eine ENQ-Makroinstruktion veranlasst und enthält eine acht Zeichen lange Bezeichnung der verlangten Mittels. Die Steuerung geht auf die Sperrenfreigabeüberwachung über. Diese stellt mittels der Aufgabensteuerblocks fest, ob die Bezeichnung des verlangten Mittels in der Reihe der Steuerblocks schon irgendwo erscheint. Ist dies der Fall, dann existieren schon aktive Aufgaben. Nun kann die Aufgabe in den Wartezustand versetzt werden. Bevor jedoch das Datenbanksystem vom CMS-Betriebssystem diese Versetzung verlangt, wird die Blockierungsüberprüfungsmethode angewendet, um festzustellen, ob die Versetzung der Aufgabe in den Wartezustand eine Blockierung bewirkt. Nebenbei sei bemerkt, dass wenn der Name des Verarbeitungsmittels nicht in der Liste erscheint, der anfordernden Aufgabe ein ausschliessliches Recht, d.h. eine Sperre über das Verarbeitungsmittel zuerkannt werden kann.

In der praktischen Ausführung gibt es für jedes Verarbeitungsmittel, das gerade besetzt ist, einen Informationsblock, der Warteschlangenelement (NQE) genannt wird. Jedes NQE enthält: 1. einen Namen mit 8 Zeichen, 2. die Kennzeichnung der benützenden Aufgabe, 3. einen Hinweis auf den Aufgabensteuerblock, der die Ausführung bestimmt.

Eine bestimmte Anzahl von z.B. NQEs werden zu einer

Gruppe vereinigt. Die Gruppen können durch geeignete Hinweise (Pointers) aneinander gehängt werden. Diese Struktur erscheint in den Instruktionen 0052-0058 der Instruktionsfolge in Tabelle 2. Das Vektorfeld der Aufgaben, die auf andere Aufgaben warten, heisst Warteliste (waitlist) und erscheint in Instruktion 0050. Ein Arbeitsfeld (worklist) ist in Instruktion 0051 bezeichnet. Diese Daten sind in Tabelle 1 zusammengestellt:

Tabelle 1



Jede ENQ-Instruktion soll die Sperre über ein Verarbeitungsmittel anfordern und benutzt eine der NQE-Instruktionen. Die Anzahl der NQE-Instruktionen ist natürlich gleich der Anzahl der zur Zeit benützter Sperren.

Der Instruktionscode ist in Tabelle 2 wiedergegeben, deren Unterschrift den Zweck und die Wirkung angibt. Einerseits sind die Instruktionen beliebig numeriert, andererseits ist es klar, dass einzelne Instruktionen anderen vorausgehen müssen. So müssen beispielsweise Datenbestimmungen den Steuer- und Operationsdaten vorausgehen. Die Befehle 50-57 sowie 544-557 und 565-570 sind in der PL/1-Sprache geschrieben. Diese Sprache ist dargestellt in PL/1(F), IBM System/360 Operating System Language Reference Manual, IBM-Druckschrift No. GC 28-8201-4, 5. Auflage 1972. Die Befehle 571-580 sind, wie schon erwähnt, in der IBM-System-360-Assembler-Sprache geschrieben.

Tabelle 2
Feststellbefehle für ENQ, DEQ und Blockierung

Befehls-Nr.	Befehl
0050	DECLARE WAITLIST (256) BINARY(8) BOUNDARY (BYTE);
0051	DECLARE WORKLIST (256) BINARY(8) BOUNDARY (BYTE);
0052	DECLARE 1 NQEB BASED (R1),
0053	DECLARE 2 NQCHAIN POINTER,
0054	DECLARE 2 NQRES BINARY (15),
0055	DECLARE 2 NQE (25),
0056	DECLARE 3 NQNAME CHAR (8),
0057	DECLARE 3 NQOWNER BINARY (8) BOUNDARY (BYTE),
0058	DECLARE 3 NQDISPEL BINARY (8) BOUNDARY (BYTE),

Tabelle 3
ENQ-DEQ Logik

Befehls-Nr.	Befehl
0544	ENQ:
0545	FREENQEB=O;
0546	R1=ADDR(NQEB1);
0547	NQESRCH:
0548	DOI=1 TO 25;
0549	IF NQNAME(I,1)='FF' X & FREENQEB=O THEN DO;
0550	FREENQE=1;
0551	FREENQEB=R1;
0552	END;
0553	ELSE DO;
0554	IF NQNAME(I)=RCBARG1 THEN DO;/*=ARE THE NAMES EQUAL*/
0555	IF RCBRTYPE=23 THEN GO TO DODEQ;
0556	WAITLIST(WUAID+1)=NQOWNER(I);
0557	DEDISPL(CURRDE)='O'B

Tabelle 4
Steuerungs-Übergang auf Blockierungs-Feststellogik

Befehls-Nr.	Befehl
0566	WORKLIST(1)=0;
0567	RS=WUIDS-1;
0568	J=R2,
0569	GENERATE SETS(WORKLIST,J,R2)
0570	REFS(MVC,TR,CLC,WAITLIST,DEADLOCK)

Tabelle 5
Blockierungs-Feststellogik

Befehls-Nr.	Befehl	
0575	EX R2,TR	Execute a «translation», length in R2
0576	SRA J,1	Shift rt. reg. J. by 1
0577	BNZ *,–8	Branch not zero
0578	BCTR R2,0	Decrement R2 by 1
0579	EX R2,CLC	Execute compare logical character
0580	BNE DEADLOCK	Branch not equal (to zero)

Tabelle 6
Logik nach Feststellung

Befehls-Nr.	Befehl
0585	WUADE→DEDISPL(1)='O'B;
0586	R15=ADDR(SWITCH);
0587	GEN CODE(BALR 14.15);
0588	R14=RCBSAVE(4)
0589	RETURN;
0590	END;
0591	END;
0592	END;
0593	IF NQCHAIN=0 THEN DO:
0594	R1=NQCHAIN;
0595	GO TO=NQESRCH;
0596	END;
0597	IF RCBRTYPE=21 1 RCBRTYPE=23 THEN RETURN;
0614	GO TO DOENQ
0637	DEADLOCK:
0638	GENERATE;
0639	GETCB TYPE=RCB
0640	MESSAGE ID=
"	.
"	.
0647	MVC MVC WORKLIST(O), WAITLIST
0648	TR TR WORKLIST(O), WORKLIST
0649	CLC CLC WORKLIST +1 (O), WORKLIST

Die Tabellen 2 bis 6 enthalten Segmente oder Folgen von fortlaufend numerierten Befehlen. Die Befehlsnummern sind relativ, d.h. dass die Nummern in Tabelle 2, die von 0050 bis 0058 laufen, gerade so gut von 1450 bis 1458 gehen könnten. Die Folge in Tabelle 3 sollte derjenigen in Tabelle 2 gleichen und ähnlich die Folgen in Tabellen 4, 5 und 6. Wie bereits bemerkt, enthalten derzeitige Betriebssysteme und Datenbanksysteme Kompiler für höhere Programmsprachen sowie Binder, die dafür sorgen, dass Programmfolgen, die in höheren Sprachen, beispielsweise PL/1 geschrieben sind, ohne weiteres in direkte Maschinenbefehle umgesetzt werden. Sowohl Kompiler als auch Verbinder sind wohl bekannt und haben mit der vorliegenden Erfindung direkt nichts zu tun. Die detaillierten Programmsequenzen sind hier lediglich des besseren Verständnisses halber angegeben. Die Tabellen 2 bis 4 und 6 beziehen sich dabei auf die Sperren-Freigabeüberwachung, wogegen die Folge in Tabelle 5, die im Maschinencode des früher genannten US-Patentes geschrieben ist, sich auf die Blockierungsüberwachung bezieht.

Nachfolgend sollen einige der Befehle dieser Tabellen erläutert werden.

Der Befehl «RCBARG1» bezeichnet ein Arbeitsmittel das reserviert werden soll. Der Name ist völlig willkürlich gewählt, da der Name allein die Reservierung veranlasst. Aufgaben, die dasselbe Verarbeitungsmittel benötigen, benützen dazu alle denselben Namen. Die anfordernde Aufgabe bekommt das Verarbeitungsmittel zugeteilt, wenn dieses frei ist, andere Aufgaben werden nicht mehr zugelassen. Wenn das Mittel nicht frei war, dann hat bereits eine andere Aufgabe ein ENQ veranlasst. Die anfordernde Aufgabe wartet nun auf das Verarbeitungsmittel, bis dessen DEQ erfolgt. Wenn eine wartende Aufgabe mit einer anderen Aufgabe zusammen eine Blockierung

veranlasst, wird die wartende Aufgabe zurückgestellt und wenn möglich neu gestartet.

Wenn der Ausdruck «RCBARG2» nicht gleich null ist, wird, wenn eine Aufgabe auf ein Verarbeitungsmittel warten musste, der Fertigstellungscode zurückgegeben, sofern das ENQ erfüllt ist. Ist «RCBARG2» gleich null, so wird der Fertigstellungscode unterdrückt.

Nun werden die Vereinbarungen für ENQ/DEQ und für Blockierungsfeststellung erläutert. Die Instruktionen 0050 bis 0058 in Tabelle 2 sind Verfahrensvereinbarungen in der PL/1-Sprache. Eine Verfahrenvereinbarung muss eine Bezeichnung haben, damit das gesamte Verfahren mittels dieser Bezeichnung oder des Verfahrensnamens aufgerufen werden kann. Die Instruktion «DECLARE» liefert die Information für den PL/1-Kompiler des Betriebssystems respektive der Datenbank, damit Speicherstellen für die im Verfahren benützten Namen reserviert werden können. Der Befehl «DECLARE» beschreibt die Einzelheiten der zu jeder Variablen gehörenden Daten. Ein Name, der in DECLARE vorkommt, gilt als «vereinbart» und die Worte, mit welchen die Daten beschrieben werden, heissen Attribute. Die Bedeutung der Namen in den Befehlen 0050 bis 0056 ist aus Tabelle 1 ersichtlich. Der Name «NQOWNER» bezieht sich auf die Aufgabe, die ein bestimmtes Verarbeitungsmittel verlangt. Der Ausdruck «NQDISTEL» besteht aus einem Hinweis auf den Aufgabensteuerblock (TCB) der verlangenden Aufgabe.

Der Abschnitt in Tabelle 2 wird entschlüsselt, um festzustellen, welche Möglichkeiten verlangt werden. Steht fest, dass eine Aufgabe ein Verarbeitungsmittel verlangt, so geht die Steuerung auf Befehl 0544 in Tabelle 3 über. Nun wird die ENQ/DEQ-Logik erläutert. Wenn ENQ sich auf die Sperrenüberwachung bezieht, wird es als Bezeichnung behandelt. Das bedeutet, dass die Bezeichnung auch ein Identifizierer ist, der als Vorspann zu einer Anweisung geschrieben ist, damit während der Ausführung die Steuerung über einen Hinweis auf ihre Bezeichnung auf diese Anweisung übertragen werden kann. Die Anweisung 0545 stellt das erste freie NQE in einer Gruppe von NQEs dar. Die Adresse ist rückgestellt auf null. Wird ein freies NQE gefunden, so kommt seine Adresse anstelle der Null, z.B. FREENQEB=ADDRESS. Die Anweisung 0546 macht klar, dass die Adresse des ersten Blocks von NQE angenommen ist.

Die Anweisung 0547 enthält die Bezeichnung NQESRCH für die DO-Gruppe der Anweisungen 0548 bis 0552. Die DO-Gruppe hat die Funktion nachzusuchen, ob der Name des angeforderten Verarbeitungsmittels im ersten Block von 25 NQEs genannt ist. Dementsprechend ist eine DO-Gruppe eine Folge von Anweisungen, deren Anfang durch eine DO-Anweisung und deren Ende durch eine END-Anweisung gebildet wird. Wird eine DO-Gruppe in einer IF-Anweisung als THEN-Wendung benützt, so werden alle Anweisungen der DO-Gruppe ausgeführt, bevor ein Sprung gemacht wird oder es werden alle Anweisungen übersprungen und die ELSE-Wendung wird ausgeführt. Wird eine DO-Gruppe als ELSE-Wendung benützt, so werden alle Anweisungen dieser Gruppe übersprungen, nachdem die THEN-Klausel ausgeführt ist.

Mit Anweisung 0548 beginnt die Suche nach dem ersten Block von 25 NQEs. Die IF-THEN-Anweisung 0549 ist so zu verstehen, dass wenn der Name eines NQEs die spezielle Bezeichnung («FF»X) hat, dies bedeutet, dass das NQE nicht benützt ist und ein freies NQE noch nicht gefunden ist (FREENQUEB=O), dann wird der DO-Teil der Anweisung ausgeführt.

Der Ausdruck FREENQUEB=R1 in 0551, bedeutet, dass die NQE-Adresse aufgehoben wird. Wenn das verlangte Verarbeitungsmittel momentan nicht reserviert ist, dann sollte dieses NQE benützt werden, um die Reservation auszudrücken. Die ELSE-DO-Anweisung 0553 bedeutet, dass wenn das NQE ein reserviertes Arbeitsmittel angibt, etwas anderes getan werden muss. Dieses andere ist in den Anweisungen 0554 bis 0557 zu finden. 0554 bedeutet, dass, wenn der Name im NQE dem verlangten Namen gleich ist und wenn gemäss 0555 ein ENQ verlangt ist, dann muss gemäss 0556 die verlangende Aufgabe warten. Die Aufgabennummer des Verarbeitungsmittel Eigentümer (NQOWNER) ist die Identifikation der Aufgabe, auf die gewartet werden muss. Sie wird in den Vektor der Warteliste WAITLIST eingesetzt, und zwar in der Stelle, die den Anfordernden zugeordnet ist (WUAID+1). Schliesslich wird in der Anweisung 0557 die verlangende Aufgabe veranlasst zu warten. Wenn gewartet werden muss, ist es notwendig zu prüfen, ob eine Blockierung erfolgen würde, wenn die Aufgabe in den Wartezustand versetzt würde.

Nun geht die Steuerung auf die Blockierungsfeststellungslogik über. Die Folge von PL/1-Anweisungen in Tabelle 4 bezieht sich auf diesen Übergang. In Anweisung 0566 wird die erste Stelle der Arbeitsliste WORKLIST 1 auf null gesetzt. In den Anweisungen 0567 und 0568 wird die Zahl der möglichen Aufgabe (WUIDS-1) in ein beliebiges Register R2 und in Register J gesetzt. Dies ist der Binärwert (N) dessen $\log_2$ die Minimalzahl von Übersetzungsbefehlen TR bedeutet die benötigt sind, um eine Schleife festzustellen. Schliesslich bedingen die Anweisungen 0569 und 0570 das Vorhandensein der benötigten Daten und Maschinenbefehle in IBM System 360-370 Maschinensprache für die Durchführung der Blockierungsfeststellung, wie sie in den Anweisungen 0574-0580 in Tabelle 5 angegeben sind.

Was die Blockierungsverhütung oder Behebung anbetrifft, sei daran erinnert, dass, wenn das Resultatfeld der Blockierungsentdeckungslogik lauter Nullen enthält, keine Blockierung vorliegt. Die Stellung von null verschiedener Bytes dagegen dient dazu, Aufgaben zu identifizieren, die in der Blockierungsschleife liegen. Sollen alle Aufgaben im inneren Blockierungskreis festgestellt werden, so muss ein Wert des Resultats ausgewählt und durch den ursprünglichen Aufgabenwartevektor verfolgt werden. Das kann von Vorteil sein, um die günstigste Aufgabe zur Auflösung einer Blockierung auszuwählen. In der Sperren und Auflösungsüberwachung, die hier als beispielhaftes Betriebssystem oder Datenbanksystem gewählt ist, kann eine Aufgabe die Reservation eines Verarbeitungsmittels erlangen, wenn dasselbe frei ist. Andere Aufgaben werden nicht mehr zugelassen. Ist ein Verarbeitungsmittel nicht frei für eine anfordernde Aufgabe, dann steht bereits eine andere Aufgabe an. Die anfordernde Aufgabe wartet, bis keine Aufgabe mehr ansteht. Wenn das Warten eine Blockierung mit einer weiteren Aufgabe auslöst, wird die anfordernde Aufgabe zurückgestellt und wenn möglich neu gestartet. Das kann so ausgedrückt werden, dass, wenn das Warten einer Aufgabe eine Blockierung veranlasst, eben diese Aufgabe ausgewählt wird, um zurückgestellt und neu gestartet zu werden.

Im Zentrum der Ausführung der Blockierungsentdeckung steht der Übersetzungsbefehl. Wie schon erwähnt, ist der Befehl TRANSLATE ein System 360 Maschinenbefehl mit einer spezifizierten Feldlänge, beispielsweise maximal 256 Bytes, die 256 Aufgaben identifizieren.

Da die Anzahl der Aufgaben die auf Blockierung überprüft werden sollen dynamisch variiert, beispielsweise N=15 zur Zeit 1 und N=150 zur Zeit 2, kommt der TRANSLATE-Befehl nicht unmittelbar vor. Er wird vielmehr veranlasst durch den EXECUTE-Befehl, der zu jeder Zeit die richtige Länge angibt. Der EXECUTE-Befehl erlaubt indirekt die Ausführung jedes beliebigen IBM-System-360-Befehls und kann bestimmte Eigenschaften dieses Befehls, beispielsweise die Länge der Operanden, ändern. Der Befehl TRANSLATE (TR) benützt jedes Byte des SOURCE-Feldes, um ein Byte der Zieltabelle zu bezeichnen. Das aus der Zieltabelle ausgewählte Byte ersetzt den Index in der SOURCE. Die Ausführung schreitet von links nach rechts, um jeweils ein Byte weiter, bis alle L-Bytes der SOURCE übersetzt sind. Die TRANSLATE-Tabelle, TARGET, wird adressiert durch zuzählen jedes SOURCE-Bytes zu der

Adresse des zweiten Operanden, d.h. zur ursprünglichen Adresse der Übersetzungstabelle. Diese allgemeine Funktion von TRANSLATE kann wie folgt erläutert werden:

Hinweisadressen	1430315256	TARGET
Position	0123456789	
Hinweisadressen	3649836	SOURCE vor
Position	0123456	Übersetzung
Hinweisadressen	0536205	SOURCE nach
Position	0123456	Übersetzung

Position 0 in der Source enthält den Hinweis 3. Darauf wird Position 3 im Target nachgesehen und ihr Wert 0 als entsprechender Hinweis in Position 0 der Source nach der Übersetzung gesetzt. Das ist ersichtlich für Position 1, die den Hinweis enthält. Der Targetwert 5 von Position 6 wird nach der Übersetzung in Position 1 der Source gesetzt. Die übrigen Werte kommen auf die selbe Art zustande. Im Gegensatz zum allgemeinen Fall wird bei der Blockierungsfeststellung der Befehl speziell verwendet. Dies darum weil Source und Target ein und dasselbe Feld sind. Das Feld, das in diesem speziellen Fall übersetzt wird, ist das Übersetzungsfeld für seine eigene Übersetzung.

Die Blockierungsfeststellungslogik benötigt drei Arten von Maschinenverzweigungsbefehlen. Diese heissen BRANCH NOT ZERO (BNZ) und BRANCH NOT EQUAL (BNE), um die Steuerung zu übertragen. Im Fall der ersten Verzweigung geht die Steuerung zurück auf den EXECUTE-Befehl. Im Fall der zweiten Verzweigung geht die Steuerung über auf eine Reihe von Befehlen zur Bearbeitung einer festgestellten Blockierung. Der Befehl BRANCH ON COUNT (BCTR) wird benützt mit der speziellen Nicht-Verzweigungsmöglichkeit und vermindert lediglich den Inhalt des Registers R2 um 1.

Der Befehl 0574 in Tabelle 5 ist die Ausführung des Übertragungsbefehls, durch welchen der Inhalt der Speicherstellen WAITLIST in den als WORKLIST bezeichneten Speicherstellen übertragen werden. Die Länge der Übertragung ist der Wert von R2. Der Übertragungsbefehl steht in der Anweisung 0647. Die Anweisung 0575 bezieht sich auf die Durchführung des Übersetzungsbefehls, der selbst in der Anweisung 0648 zu finden ist. Die Anzahl der Übersetzung wird durch den Ausdruck $\log_2(N+1)$ gesteuert. Diese Funktion wird durch den SRA-Befehl in der Anweisung 0576 ausgelöst. Das heisst nach der Durchführung jedes Übersetzungsbefehls wird der Inhalt des J-Registers, um eine Position nach rechts verschoben. Der Wert dieses Registers wird durch den BNZ-Befehl in Anweisung 0577 überprüft. Ist der Wert ungleich 0 so ist eine weitere Übersetzung nötig, damit die Verzweigung zurück zur Anweisung 0575 erfolgt. Ist der Wert gleich null, dann ist 0578 die nächste auszuführende Anweisung. Der BRANCH-Befehl BCTR vermindert den Inhalt des Registers R2 aus Gründen der System-360-Befehlslänge. Die letzte Anweisung 0579 vergleicht das Resultat der Übersetzungsbefehle, um festzustellen, ob alle Elemente gleich null sind. Sind sie es nicht, dann veranlasst die Anweisung BNE (Verzweigung wenn unglich null) einen Sprung auf Anweisung 0637 (Tabelle 6), um die Blockierung aufzulösen. Übrigens ist Blockierung eine Bezeichnung und die Anweisungen 0638 bis 0644 machen ersichtlich, dass weitere Anweisungen nötig sind, um die Auflösung auch zu erreichen.

Zeigt das Resultat durchwegs Nullen und liegt somit eine Blockierung nicht vor, so wird die Adresse des Zuteilungsteuerblocks in Anweisung 0585 festgestellt, wie Tabelle 6 zeigt. Anweisung 0586 und 0587 führen die Steuerung auf eine andere Anweisung, damit die anfordernde Aufgabe in den Wartezustand versetzt werden kann.

Interessanterweise weist Anweisung 0592 zurück auf die Anweisung 0555 in Tabelle 3. Dies bedeutet, das kein NQE gefunden wurde und dass die Suche im gegebenen Block von 25 NQEs fortgesetzt werden muss. Die Anweisungen 0593 bis 0596 weisen darauf hin, dass ein NQE-Block durchsucht wurde. Sofern es einen weiteren Block gibt, muss nun zu Anweisung 0547, ebenfalls Tabelle 3, zurückgekehrt werden, um diesen Block zu durchsuchen. Die Anweisung 0597 bedeutet ein DEQ oder GATE. Auf ein DEQ hin wird ein Verarbeitungsmittel wieder zur allgemeinen Verfügung gestellt. Ein Gate verschafft keine Reservation auf ein Verarbeitungsmittel, aber verhindert weitere Verarbeitung, sofern eine Aufgabe eine Reservation verlangt hat. Dies kommt vor, wenn gewisse Aufgaben nicht verlangen, dass Verarbeitungsmittel exklusiv für sie reserviert werden, sondern dass auf sie nicht zugegriffen werden soll, bis eine gewisse andere Aufgabe sie reserviert hat.