RECHENZENTRUM TH MÜNCHEN
ARBEITSGRUPPE BETRIEBSSYSTEME

INTERNSCHRIFT Nr. 19

THEMA:

Zur Rechnerkernvergabe in einem Mehrprozessorsystem

VERFASSER:

Jürgens

DATUM:

28.08.1969

FORM DER ABFASSUNG

- ENTWURF
- X AUSARBEITUNG
- ENDFORM

SACHLICHE VERBINDLICHKEIT

- ALLGEMEINE INFORMATION DISKUSSIONSGRUNDLAGE
- X ERARBEITETER VORSCHLAG
- VERBINDLICHE MITTEILUNG
- VERALTET

ÄNDERUNGSZUSTAND

BEZUG AUF BISHERIGE INTERNSCHRIFTEN

Vorkenntnisse aus: Internschrift 16

Erweiterung von:

Ersatz für:

BEZUG AUF KÜNFTIGE INTERNSCHRIFTEN

Vorkenntnisse zu:

Erweiterung in:

Ersetzt durch:

ANDERWEITIGE LITERATUR

Zur Rechnerkernverga e in einem Mehrprozessorsystem

Inhalt

1. Einleitung

Wir setzen eine Rechenanlage voraus, die mehrere gleichartige Rechnerkerne (RK-e) hat und die als Multitasking-Maschine betrieben wird. - Multitasking involviert Scheduling: die Betriebsmittel der Anlage müssen unter mehrere Aufträge verteilt werden, für diese Verteilung müssen Strategien und Algorithmen bestehen.

1.1 Insbesondere müssen die Rechnerkerne der Anlage verteilt werden. Im Folgenden verstehen wir unter Scheduling: Planung und Vollzug der Rechnerkernverteilung.

Da es kaum möglich scheint, aus einer bestimmten Strategie in jeder konkreten Einzelsituation direkt und schnell die entsprechende Entscheidung abzuleiten, setzen wir eine mehrstufige Organisation des Programmkomplexes "Scheduling" voraus:

(1.1.i) Gewisse Programme werden aus der Gesamtsituation und der bei der Konstruktion des Systems gewählten Strategie taktische Richtlinien bestimmen.

(1.1.ii) Eine andere Gruppe von Programmen wird aus den taktischen Richtlinien Eingangsdaten für die konkrete Entscheidung ableiten.

(1.1.iii) Schließlich wird es einen Programmkomplex geben, der in einer konkreten Situation aufgrund einfacher Parameter schnell eine Entscheidung trifft und die Zuordnung eines bestimmten Rechnerkernes an ein bestimmtes Programm vollzieht. ("local scheduling" oder auch "Rechnerkernvergabe")

Während die unter (1.1.i) und (1.1.ii) angeführten "übergeordneten Scheduler" stark von der Größe und Kompliziertheit eines Systems abhängen und im Extremfall ganz fehlen können, muß eine Rechnerkernvergabe (RKV) stets existieren. (Denn wenn z.B. ein Programm durch eine Hardware-Unterbrechung unterbrochen wurde, und wenn die Unterbrechung abgehandelt ist, muß entschieden werden, was weiter mit dem betreffenden RK geschehen soll.)

./. 3

Die vorliegende Arbeit beschäftigt sich nur mit der Rechnerkernvergabe im Sinne von (1.1.iii) ("local scheduling").

1.2 Für die "übergeordneten Scheduler" aus (1.1.i) und (1.1.ii) ist i.a. nur die Tatsache relevant, daß man Multitasking betreibt. Ob für dieses Multitasking mehrere reale RK-e zur Verfügung stehen oder nur ein RK, der im Zeitmultiplex betrieben wird, macht hier keinen Unterschied.

Nicht so auf der Ebene der RKV. Hier ergeben sich einschneidende Konsequenzen aus der Anzahl der realen RK-e. Wir setzen einerseits voraus, daß die Anzahl der RK-e größer als 1 ist, und wir beschränken uns andererseits auf den Fall, daß es nur "wenige" RK-e gibt (etwa 2 bis 4).

1.3 Die Objekte der RKV werden vor allem Benutzeraufträge sein. Dabei ist jedoch nicht vorausgesetzt, daß ein Benutzerauftrag sich direkt um einen RK bewirbt; vielmehr lassen wir den Fall zu, wo ein oder mehrere Benutzeraufträge zu einer Gruppe zusammengefaßt werden, die durch einen "Bearbeiter" gegenüber dem BS repräsentiert wird. Außerdem wird zugelassen, daß auch gewisse Systemteile als Objekte der RKV auftreten (und dabei ggf. durch "Bearbeiter" repräsentiert werden), genauso, als ob sie Benutzeraufträge wären.

Andererseits wird es in jedem System Moduln geben, die unabhängig von der RKV einen RK erhalten können; dies kann geschehen:

(1.3.i) durch Hardware-Unterbrechung (kurz: Unterbrechung[1])

(1.3.ii) durch direkten Ansprung von einem anderen Programm aus, das nicht zur RKV gehört.

---

1) In dieser Arbeit wird das Wort "Unterbrechung" nur im Sinne von Hardware-Unterbrechung gebraucht.

./. 4

Wir setzen explizit voraus, daß alle Programm-Moduln, die nicht nach (1.3.i) oder (1.3.ii) einen RK erhalten, durch ein und dasselbe Programm mit RK-en versorgt werden. Dieses Programm bezeichnen wir als die RKV.

Die Objekte der RKV werden Akteure genannt. Es wird angenommen, daß ein Akteur jeweils höchstens einen RK haben kann. (Das ist keine Annahme über Parallelzweige in Programmen.)

2. Einige Begriffe

2.1 Ein Akteur ist stets in genau einem der folgenden drei Zustände:

(2.1.i) rechnend (r): der Akteur hat einen RK[2)]

(2.1.ii) rechenwillig (rw): der Akteur ist in der Lage zu rechnen und bewirbt sich um einen RK[3)]

(2.1.iii) blockiert (b): der Akteur ist momentan nicht in der Lage, (in sinnvoller Weise) zu rechnen, - er soll bei der Vergabe von RK-en nicht berücksichtigt werden.

Es sind folgende Übergänge möglich:
rw → r → b → rw , r → rw , rw → b .

---

2) Im allgemeinen stellt ein BS gewisse Programmoduln bereit, die von Akteuren aus angesprungen werden können und die für die Akteure ähnliches leisten wie eigene Unterprogramme. Läuft ein solches Programm auf dem RK k, nachdem es vom Akteur a angesprungen wurde, so sagen wir ebenfalls: der Akteur a ist rechnend (auf dem RK k).

3) Für diese Arbeit machen wir die Voraussetzung, daß alle RK-e des Systems bezüglich der RKV gleichwertig sind. (Insbesondere hätte es also keinen Sinn zu sagen, daß ein Akteur sich um einen ganz bestimmten RK bewirbt.) - Diese Voraussetzung erscheint z.B. dann problematisch, wenn die Unterbrechungen nicht gleichmäßig auf alle RK-e verteilt werden. Wir werden auf diesen Punkt in einer späteren Internschrift zurückkommen.

./. 5

2.2 Ein Paar (a, k), bestehend aus einem Akteur a und einem RK k, ist stets in genau einem der folgenden drei Zustände:

(2.2.i) liiert: a ist rechnend auf dem RK k

(2.2.ii) bekannt: der RK k beabsichtigt, a demnächst zu rechnen oder: der letzte Akteur, der von k gerechnet wurde, war a, und k hat noch nicht entschieden, welchen Akteur er demnächst zu rechnen beabsichtigt.

(2.2.iii) fremd: k und a stehen in keiner Verbindung zueinander.

Es können folgende Übergänge vorkommen:

fremd → bekannt
bekannt → liiert
liiert → bekannt
bekannt → fremd

2.2.1 Statt "Das Paar (a, k) ist liiert (bekannt, fremd)" werden wir auch sagen: "Der Akteur a ist mit dem RK k liiert (dem RK k bekannt, fremd)" oder: "Der RK k ist mit dem Akteur a liiert (dem Akteur a bekannt, fremd)". Mit der Formulierung: "Der RK k ist liiert" wird ausgesagt, daß es einen Akteur gibt, der auf dem RK k rechnet.

2.2.2 Zu jedem Zeitpunkt ist einem RK genau ein Akteur nicht fremd, während ein Akteur mehreren RK-en "nicht fremd" sein kann. Das Paar (a, k) ist bekannt genau dann, wenn für alle Akteure b mit b≠a gilt "(b, k) fremd" und wenn (a, k) nicht liiert.

2.3 Wir fassen alle Programmstücke des Systems, die nicht zu Akteuren gehören, zusammen unter dem Begriff "Bereich der Nicht-Akteure". Entsprechend reden wir vom "Akteur-Bereich". Dann können wir sagen: Ein RK pendelt dauernd zwischen dem Bereich der Nicht-Akteure und dem Akteurbereich hin und her. Er ist im Akteurbereich genau dann, wenn er liiert ist. Er verläßt den Bereich der Nicht-Akteure stets über die RKV.

./. 6

2.4 Es gibt gewisse Unterbrechungen (z.B. Alarme aufgrund fehlerhafter Rechenwerks-Operationen), auf die nur dadurch sinnvoll reagiert werden kann, daß dem unterbrochenen Programm direkt der RK zurückgegeben wird, auf dem es unterbrochen wurde (i.a. über einen speziellen Eingang).

Um dies zu gewährleisten, modifzieren wir die obigen Definitionen, indem wir sagen: der Akteur a ist rechnend auf dem RK k (liiert mit dem RK k) auch dann, wenn der RK gerade mit der Anfangsbehandlung einer Unterbrechung des o.a. Typs beschäftigt ist. Wir fordern dann, daß der RK nach dieser Anfangsbehandlung automatisch (d.h. ohne die RKV zu durchlaufen) zu dem unterbrochenen Akteur zurückkehrt.

3. Akteurliste und Akteurlistendienst

3.1 Die Akteurliste

Die Gesamtheit der Ablageplätze für die Eingangsinformation der RKV bezeichnen wir als Akteurliste (AKL). O.B.d.A setzen wir folgendes voraus:

3.1.1 Die AKL enthält für jeden Akteur ein Element; dort ist verzeichnet:
eine unveränderliche Kennzeichnung für den Akteur;
eine Aussage darüber, ob der Akteur r, rw oder b ist.

3.1.2 Die AKL enthält für jeden RK ein Element. Dort ist verzeichnet,
ob auf dem RK ein Akteur rechnet, und
wenn ja, welcher Akteur auf dem RK rechnet,
wenn nein, welcher Akteur dem RK bekannt ist.

3.1.3 Die AKL enthält, neben dem in 3.1.1 zitierten, für jeden Akteur ein weiteres Element. Falls der Akteur nicht gerade rechnet, ist dort all die Information abgelegt, die die RKV benötigt, um dem Akteur einen RK zuzuweisen. Diese Information wird immer gewisse Registerstände und Steuerbits umfassen.

./. 7

3.1.4 Wie gesagt (vgl. Anmerkung 3) in 2.1), setzen wir hier alle RK-e als gleichartig und gleichberechtigt voraus. - Dagegen wird für die Akteure eine lineare Ordnung vorausgesetzt, d.h. jeder Akteur hat eine Nummer, die wir als seine Prioritätsnummer bezeichnen (und die in der AKL eingetragen ist). Die Prioritätsnummer gibt die Wichtigkeit an, die ein Akteur bei der RK-Zuteilung haben soll, und zwar ist die Wichtigkeit desto größer, je kleiner die Nummer ist (hohe Priorität $\hat{=}$ niedrige Prioritätsnummer).

Dies bedeutet keineswegs, daß wir eine "lineare" Scheduling-Strategie voraussetzen. Nach welchen Gesichtspunkten die übergeordneten Scheduler vorgehen, bleibt offen; hier wird lediglich angenommen, daß sie für die lokale Entscheidung eine lineare Liste der Akteure liefern. Insbesondere wird die Prioritätsnummer eines Akteurs nicht als konstant vorausgesetzt (vgl. 3.2.1).

3.1.5 Die AKL enthält i.a. weitere Information, sowohl über die Akteure als auch über den Zustand des gesamten Systems. Für unsere Betrachtungen genügen jedoch zunächst die o.a. Voraussetzungen.

3.2 Der Akteurlistendienst

Wir bezeichnen die Menge aller Programmteile, die auf die AKL zugreifen, als den Akteurlistendienst (AKD). Von einem Programm, in dessen Verlauf auf die AKL zugegriffen wird, sagen wir demnach: es ruft den AKD auf. Solche Programme sind:

3.2.1 Die der RKV übergeordneten Scheduler. Sie können aufgrund der Gesamtsituation des Systems mittels ihrer Strategie

neue Aufträge zu Akteuren machen
Akteure aus der AKL streichen
die Prioritäten der Akteure neu festsetzen.

./. 8

Sobald also die übergeordneten Scheduler auf die AKL zugreifen, wird die Liste der Akteure durch eine grundsätzlich neue ersetzt.

3.2.2 Programme, die arbeiten, nachdem ein RK einen Akteur verloren hat (= aufgehört hat, ihn zu rechnen) und bevor er die RKV anspringt. Zum Beispiel kann ein blockierter Akteur aufgrund einer Unterbrechung rechenwillig werden. Dies muß dann in der AKL verzeichnet werden.

3.2.3 Die RKV. Sie ruft den AKD auf, um zu lesen, läßt aber auch Eintragungen über die von ihr verursachten Änderungen machen.

3.2.4 Grundsätzlich sollten auch Akteure die Möglichkeit haben, den AKD aufzurufen. Es scheint jedoch keine wesentliche Einschränkung der Allgemeinheit zu sein, wenn man voraussetzt, daß ein Akteur nur dann den AKD aufruft, wenn er dabei seinen RK abgibt. Wir machen diese Voraussetzung und können damit sagen: Der AKD arbeitet nur dann, wenn der RK (auf dem er arbeitet) im Bereich der Nicht-Akteure ist.

3.2.5 I.a. fällt ein Ereignis zeitlich nicht mit der entsprechenden Eintragung in der AKL zusammen, d.h. die AKL ist i.a. nicht aktuell. Es ist sogar denkbar, daß man absichtlich die Eintragung einer Zustandsänderung verzögert (vgl. 7.2). Wir unterscheiden daher Aussagen wie "a ist blockiert" und "a ist als blockiert verzeichnet" oder "a wird als blockiert geführt".

3.2.6 Es besteht die Möglichkeit, daß der AKD auf einem RK aufgerufen wird, während er auf einem anderen RK bereits läuft. Eine derartige Simultanarbeit auf der AKL ist i.a. unzulässig, und es muß daher ein Koordinierungs-Mechanismus geschaffen werden. Wir beschreiben hier nur eine recht einfache Methode:
Es wird eine Boolesche Variable geschaffen, die wie ein Semaphor nach Dijkstra benutzt wird, und die die gesamte AKL für alle Arten von Zugriffen sperrt (mit Ausnahme der Variablen, die überhaupt nicht geschützt zu werden brauchen, - falls

./. 9

es solche gibt. Die in 3.1.1 bis 3.1.4 genannten Einträge müssen auf jeden Fall geschützt werden). Diese Variable heißt AKL-Semaphor.

Wie bei Dijkstra müssen die P- und V-Operationen auf diesen Semaphor Elementarschritte sein im Sinne von Ununterbrechbarkeit durch Hardware-Unterbrechungen.

Da auf diesen Semaphor von mehreren RK-en aus zugegriffen wird, ergibt sich darüberhinaus noch folgende Forderung: Eine P- oder V-Operation, die auf einem RK abläuft, darf durch andere RK-e nicht gestört werden. (Dies impliziert eine Forderung an die Hardware.)

4. Konsistenz

Grob gesprochen hat die RKV die Aufgabe, den jeweils wichtigsten rechenwilligen Akteuren die verfügbaren RK-e zuzuweisen. Ist diese Aufgabe erfüllt, so sprechen wir von Konsistenz.

4.1 Bei genauerer Betrachtung ergeben sich verschiedene Konsistenzbegriffe:

4.1.1 Wir sagen "Das System ist (zum Zeitpunkt t_0) konsistent vom Typ A", wenn die l RK-e, die zu diesem Zeitpunkt im Akteurbereich sind, die l wichtigsten unter den nicht blockierten Akteuren rechnen.

4.1.2 Wir sprechen von Konsistenz vom Typ B, wenn die liierten RK-e die wichtigsten der Akteure rechnen, die als nicht blockiert verzeichnet sind.

4.1.3 Wir bezeichnen das System als konsistent vom Typ C, wenn die Akteure, die als liiert (= rechnend) verzeichnet sind, unter den als nicht blockiert verzeichneten die wichtigsten sind.

./. 10

4.1.4 Wir nennen das System konsistent vom Typ D, wenn die Akteure, die den n RK-en des Systems (n = Anzahl der RK-e des Systems) nicht fremd sind, die n wichtigsten nicht blockierten Akteure sind.

4.1.5 Hierzu zwei Anmerkungen: Wir gehen davon aus, daß ein Akteur einem RK nicht fremd ist, wenn und nur wenn er als solcher eingetragen ist.

Man beachte, daß ein Akteur sehr wohl einem RK bekannt und gleichzeitig blockiert sein kann.

4.1.6 Wir sagen: "Das System ist konsistent vom Typ E", wenn die Akteure, die den n RK-en des Systems nicht fremd sind, die wichtigsten als nicht blockiert verzeichneten Akteure sind.

4.1.7 Wir nennen das System konsistent vom Typ F, wenn es konsistent vom Typ E ist, und wenn höchstens ein RK nicht als liiert verzeichnet ist.

4.1.8 Wir sprechen von Inkonsistenz eines bestimmten Typs genau dann, wenn Konsistenz dieses Typs nicht vorliegt.
Wenn anhand der AKL geprüft wird, ob Konsistenz oder Inkonsistenz eines bestimmten Typs vorliegt, so sprechen wir von einer Konsistenzprüfung dieses Typs. Es sind nur Konsistenzprüfungen vom Typ C, E oder F möglich.

4.2 Es ist offensichtlich, daß es das Ziel der RKV sein müßte, Konsistenz vom Typ A herzustellen. Realisieren läßt sich jedoch höchstens Konsistenz vom Typ B, so daß sich die Forderung gibt, am Anfang einer jeden RKV eine möglichst aktuelle AKL zu haben. Setzt man voraus, daß die Zeit, während derer ein Akteur einem RK ununterbrochen bekannt ist, jeweils nur kurz ist, so kann man statt Typ A auch Typ D bzw. statt Typ B auch Typ E fordern.

./. 11

Zusammenfassend können wir sagen:

(4.2.i) Der AKD bzw. die ihn aufrufenden Programme müssen so arbeiten, daß zu Beginn einer jeden RKV die AKL so aktuell wie möglich ist.

(4.2.ii) Die RKV hat die Aufgabe, Konsistenz vom Typ B herzustellen.

4.3 Damit ist zunächst klar, daß RKV sich grundsätzlich nicht auf einen einzelnen RK bezieht, sondern auf das ganze System. Auch wenn einmal Konsistenz (vom Typ A) erreicht ist, genügt es zur Aufrechterhaltung der Konsistenz i.a. nicht, daß ein RK, der seinen Akteur verloren hat (= aufgehört hat, ihn zu rechnen), beim Verlassen des Bereichs der Nicht-Akteure durch die RKV nur für sich selbst einen neuen Akteur bestimmt und zuteilt. Hierfür gibt es folgende Gründe:

4.3.1 Der Verlust des Akteurs kann bereits zu einer Inkonsistenz (vom Typ A) führen; er führt immer dann zur Inkonsistenz, wenn der betroffene Akteur vom Zustand r in den Zustand rw übergeht und wenn dabei mindestens ein weniger wichtiger Akteur im Zustand r bleibt.
Hat das Ereignis, das den Verlust des Akteurs bewirkte, außerdem Zustandsänderungen des Typs b→rw zur Folge, so läßt sich über die Anzahl der RK-e, die umbesetzt werden müssen, generell nichts mehr aussagen.

4.3.2 Der Verlust eines Akteurs kann mit einer Zustandsänderung für mehrere Akteure einhergehen (z.B. gleichzeitiges "Wecken" mehrerer Akteure).

4.3.3 Zwischen dem Verlust eines Akteurs und der darauffolgenden RKV kann eine Unterbrechung geschehen, die ihrerseits Zustandsänderungen bei Akteuren zur Folge hat.

./. 12

4.3.4 Nachdem ein übergeordneter Scheduler die AKL verändert hat, muß prinzipiell die Besetzung aller RK-e neu vorgenommen werden.

4.4 Wir haben bisher stets vorausgesetzt, daß die Prioritätsnummern der Akteure Steuerungsparameter sind, deren sich die übergeordneten Scheduler bewußt und gezielt bedienen. Fragwürdig wird diese Voraussetzung dann, wenn gewisse Gruppen von Akteuren untereinander oder gar alle Akteure mehr oder weniger gleichartig sind.
Im Extremfall könnte dann die Aufgabe der RKV lauten:

(4.4.i) auf möglichst einfache Weise dafür zu sorgen, daß ein RK, der seinen Akteur verloren hat, so schnell wie möglich in den Akteurbereich zurückkehrt.

In diesem Fall hätten die Prioritätsnummern nur noch den Sinn, die Entscheidung, welcher von den gleichberechtigten Akteuren als nächster zu rechnen sei, zu vereinfachen. Konsistenz würde dann keine Rolle spielen.

4.4.1 Daß alle Akteure gleichartig sind, scheint für ein größeres Rechensystem eine unrealistische Annahme zu sein. Sie setzt voraus, daß Akteure nur Benutzeraufträge vertreten, und daß alle diese Benutzeraufträge gleichartig und gleichwichtig sind. - Dagegen scheint es realistisch anzunehmen, daß es Gruppen von Aufträgen gibt, die gleichartig und gleichberechtigt sind, z.B. gewisse Konsol-Aufträge in einem time-sharing-System. Wir vernachlässigen hier diesen Aspekt, zumal er nicht notwendig die Ebene der RKV betrifft.

5. Zwei Lösungswege

In 4. hat sich gezeigt, daß jede RKV das Gesamt-System berücksichtigen und ggf. zur Umbesetzung mehrerer RK-e führen muß. Dies kann auf zwei Weisen erreicht werden:

./. 13

5.1 In Einzelschritten (Einzelschrittverfahren):

Der RK, der die RKV anspringt, "bedient sich zunächst an der AKL", d.h. er sucht den wichtigsten rechenwilligen Akteur für sich heraus und reserviert ihn sich. Anschließend prüft er, ob nach Ansprung dieses Akteurs Konsistenz vorliegen würde und benachrichtigt ggf. einen geeigneten anderen RK darüber, daß dies nicht der Fall ist. Danach springt er den ausgesuchten Akteur an und rechnet ihn, während der benachrichtigte RK seinerseits die oben beschriebene RKV durchläuft.

5.2 In einem Gesamtschritt (Gesamtschrittverfahren):

Der RK, der die RKV anspringt, bestimmt aufgrund der Eintragungen in der AKL, welche RK-e mit welchen Akteuren neu zu besetzen sind, damit Konsistenz vorliege, und damit er selbst einen Akteur erhalte. Er bereitet für jeden der betroffenen RK-e eine entsprechende Nachricht vor (auch für sich selbst) und schickt alle Nachrichten gleichzeitig ab. Daraufhin springen alle benachrichtigten RK-e den ihnen jeweils mitgeteilten Akteur an.

5.3 Beide Vorschläge machen davon Gebrauch, daß die RK-e des Systems eine Möglichkeit besitzen, einander aktiv zu benachrichtigen. Wir setzen eine entsprechende Eigenschaft der Hardware voraus.

6. Der Inkonsistenzgrad

6.1 In Zusammenhang mit der RKV in Einzelschritten (5.1) erscheint es wünschenswert, ein Maß für den Grad der Inkonsistenz zu haben. Wir bezeichnen als Inkonsistenzgrad des Systems die Anzahl der Einzelschritte, die notwendig wären, damit das System konsistent vom Typ B würde. (Vorausgesetzt, daß diese Einzelschritte die Zeit 0 benötigen würden).

./. 14

6.2 Für den Inkonsistenzgrad zur Zeit t schreiben wir

$$I(t)$$

Es gilt für alle t: $0 \leq I(t) \leq$ Anzahl der RK-e des Systems.
Es gilt (auch bei zwei RK-en) nicht:
I(t) = Max (Anzahl der RK-e des Systems,
Anzahl der rw Akteure, die wichtiger sind als der unwichtigste r Akteur)
Ebensowenig gilt: I(t) = Max (Anzahl der RK-e des Systems,
Anzahl der Zwischenräume zwischen zwei benachbarten r Akteuren, in denen ein rw Akteur liegt).

6.3 Diese Bemerkungen zeigen, daß die Fortschreibung des Inkonsistenzgrades einen nicht-trivialen Algorithmus erfordern würde (auf dessen Darstellung hier verzichtet wird). Praktische Bedeutung für die RKV in Einzelschritten hätte der Inkonsistenzgrad, wenn man

(6.3.i) ihn nach jeder AKL-Änderung fortschreiben würde oder

(6.3.ii) ihn zu Beginn einer jeden RKV berechnen und dann herunterzählen würde.

Bedenkt man, daß der AKD z.B. während der Anfangsbehandlung einer Unterbrechung aufgerufen werden könnte, so erscheint (6.3.i) als zu aufwendig. (6.3.ii) dagegen hat nur dann Sinn, wenn zwischen zwei RKV-Einzelschritten der AKD die RKV nicht verändert, so daß man einen einmal berechneten Inkonsistenzgrad schrittweise auf Null herunterzählen kann. Das vorauszusetzen scheint jedoch nicht sinnvoll.

6.4 Wir kommen zu dem Schluß, daß der Inkonsistenzgrad für die Praxis keinen Wert hat.

./. 15

7. Reaktionsgeschwindigkeit von AKD und RKV

7.1 Zustandsänderungen von Akteuren, zusammen mit der Prioritäts-Ordnung der Akteure, streben danach, ohne Zeitverzögerung berücksichtigt zu werden. In einem realen System gibt es Zeitverzögerungen aus drei Gründen:

(7.1.i) Es vergeht eine gewisse Zeit, bis der AKD die Änderung in der AKL eingetragen hat.

(7.1.ii) Bevor die RKV angesprungen werden kann, werden i.a. noch gewisse andere Tätigkeiten im Bereich der Nicht-Akteure ausgeführt. (Die Nicht-Akteure sind wichtiger als alle Akteure.)

(7.1.iii) Die RKV benötigt eine endliche Zeit, bis sie den nächsten zu rechnenden Akteur findet und anspringen kann.

7.2 Da die Eintragungen in der AKL erst ausgewertet werden, wenn die nächste RKV oder die nächste Konsistenzprüfung stattfindet, ist die Reaktionsgeschwindigkeit des AKD belanglos, solange nur bei jeder RKV und bei jeder Konsistenzprüfung eine möglichst aktuelle AKL zur Verfügung steht. Hier sollen nur zwei extreme Möglichkeiten angedeutet werden, den AKD zu organisieren:

(7.2.i) Sobald es einem Nicht-Akteur möglich ist, den AKD aufzurufen, tut er dies. Dabei muß i.a. der AKL-Semaphor betätigt werden, und falls dieser gesperrt ist, muß der RK vor ihm in eine Abfrageschleife gehen.

(7.2.ii) Sobald es einem Nicht-Akteur möglich ist, Informationen für den AKD bereitzustellen, schreibt er diese in einen "Abhol-Bereich". Der AKD selbst wird der RKV starr vorgeschaltet und holt vor jeder RKV alle Information aus derartigen Abhol-Bereichen, um sie in der AKL einzutragen. Hierbei entstehen ver-

./. 16

schiedene Organisationsprobleme, - u.a. muß gewährleistet sein, daß mehrere bereitgestellte Informationen einander nicht widersprechen, bzw. daß unter sich widersprechenden Informationen eine als gültig ausgezeichnet ist.

Hier wird nur vorausgesetzt, daß die AKL aktuell ist, sooft ein RK in die RKV oder in eine Konsistenzprüfung eintritt.

7.3 Wir formulieren noch einmal explizit die Voraussetzung: "Jeder Nicht-Akteur ist wichtiger als jeder Akteur."

Unter dieser Voraussetzung liegt die Reaktionsgeschwindigkeit der RKV fest, solange das System nur einen RK hat. In einem Mehrprozessorsystem ergibt sich dagegen folgende Möglichkeit:

Hat ein RK, z.B. RK 1, eine Zustandsänderung erkannt, die eine Inkonsistenz bewirkt, so braucht er mit dem Ansprung der RKV nicht zu warten, bis er selbst den Bereich der Nicht-Akteure verläßt. Vielmehr kann er (normalerweise) sofort einen anderen RK benachrichtigen mit der Aufforderung, Konsistenz herzustellen. Dieser, z.B. RK 2, durchläuft dann die RKV, während RK 1 noch im Bereich der Nicht-Akteure ist.

7.4.1 Wir nennen die RKV <u>träge</u>, wenn von dieser Möglichkeit nicht Gebrauch gemacht wird, d.h. wenn die Benachrichtigung eines RK-es durch einen anderen RK, verbunden mit der Aufforderung, die RKV zu durchlaufen, höchstens dann geschieht, wenn der Absender der Nachricht seinerseits gerade die RKV durchläuft.

7.4.2 Wir nennen die RKV <u>reaktionsschnell</u>, wenn sie nicht träge ist, d.h., wenn auch von anderen Nicht-Akteuren als von der RKV aus ein RK einen anderen RK zur RKV auffordern kann. - Dies bedeutet nicht unbedingt, daß ein RK im Bereich der Nicht-Akteure sobald wie möglich die Konsistenz des Systems prüft, um dann ggf. einen anderen RK zu benachrichtigen.

./. 17

7.4.3 Es sei noch bemerkt, daß die Sprechweise: die RKV ist "träge" - "reaktionsschnell" eigentlich nicht korrekt ist insofern, als diese Eigenschaften jeweils den gesamten Bereich der Nicht-Akteure zukommen.

7.5 Je länger ein RK im Mittel im Bereich der Nicht-Akteure verweilt, und je größer das Gefälle in der Wichtigkeit der Akteure ist, desto mehr wird man dazu neigen, eine reaktionsschnelle RKV zu wählen.

Andererseits ist eine reaktionsschnelle RKV mit erhöhtem Organisationsaufwand verbunden, vor allem dadurch, daß zusätzliche Konsistenzprüfungen vorgenommen werden müssen.

8. Zur RKV in einem Gesamtschritt

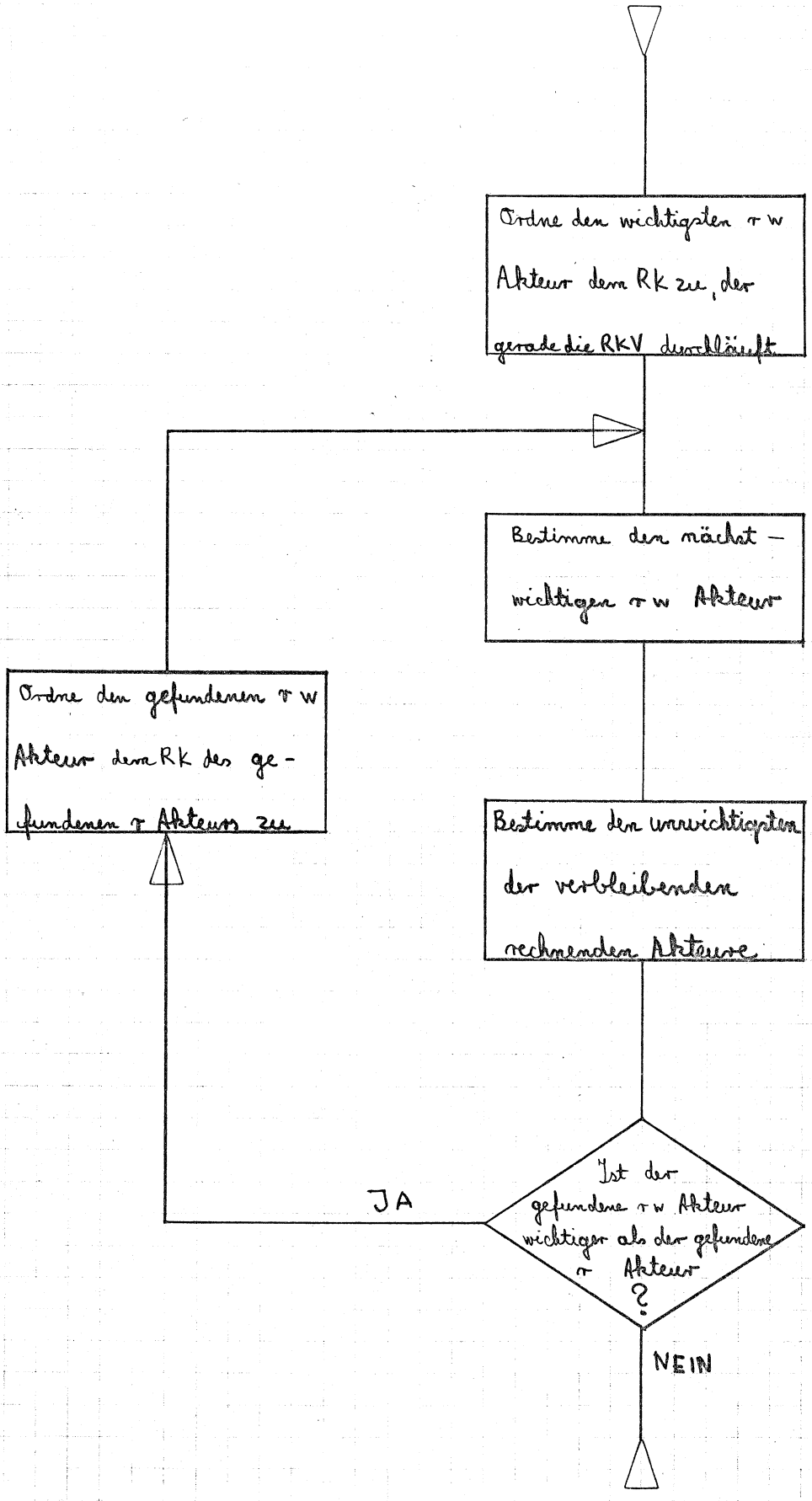
Das Gesamtschrittverfahren zerfällt in zwei Abschnitte:

(8.i) es wird festgesetzt, welche Akteure auf welchen RK-en rechnen sollen;

(8.ii) für jeden RK, dessen neuer Akteur vom alten verschieden ist, wird der Wechsel des Akteurs vorbereitet und angestoßen.

8.1 Da meistens nur ein RK neu besetzt werden muß, und da relativ selten alle RK-e wirklich neue Akteure erhalten müssen, wird man bei der Implementierung von (8.i) einen Algorithmus des folgenden Typs wählen:

./. 18


Ordne den wichtigsten r w Akteur dem RK zu, der gerade die RKV durchläuft
Bestimme den nächst – wichtigen r w Akteur
Bestimme den unwichtigsten der verbleibenden rechnenden Akteure
Ist der gefundene r w Akteur wichtiger als der gefundene r Akteur ?
JA
NEIN
Ordne den gefundenen r w Akteur dem RK des gefundenen r Akteurs zu

./. 19

Man wird also auch bei dem Gesamtschrittverfahren die Berechnung der Neuverteilung in Einzelschritten vornehmen.

8.2 Ändert ein Akteur während der RKV seinen Zustand, so wird dadurch die AKL falsch und es ist dann i.a. nicht sinnvoll, das Gesamtschrittverfahren mit der falschen AKL zu Ende zu führen.

Andererseits ist es auch nicht zweckmäßig, die RKV abzubrechen, die Änderung in die AKL einzutragen und dann von neuem mit einer RKV nach dem Gesamtschrittverfahren zu beginnen, denn wenn z.B. gerade der wichtigste rw Akteur einem RK zugeordnet wurde, so besteht eine hohe Wahrscheinlichkeit, daß nach dem Eintrag der Änderung diesem Akteur wieder ein RK zugeordnet wird.

Wünschenswert wäre, daß die RKV jeweils nach einem Berechnungsschritt die AKL freigibt, daß der AKD sie fortschreibt, und daß die RKV anschließend mit der Berechnung fortfährt, wobei die geänderte AKL berücksichtigt wird und wobei nur dann die vorhergehenden Berechnungsschritte rückgängig gemacht werden, wenn anders keine Konsistenz erreicht werden kann.

Eine derartige Lösung dürfte verschiedene Organisationsprobleme aufwerfen.

8.3 Falls nur der RK neu mit e. Akteur zu besetzen ist, der gerade die RKV durchläuft, besteht kein Unterschied zwischen dem hier skizzierten Gesamtschrittverfahren und der RKV in Einzelschritten. Falls mehrere RK-e betroffen sind, kommt eine Eigenschaft des Gesamtschrittverfahrens zum Tragen:

Alle betroffenen RK-e werden erst am Ende der RKV benachrichtigt, und zwar gleichzeitig.

./. 20

Hieraus ergeben sich verschiedene Nachteile:

8.4 Wenn für den wichtigsten rw Akteur schon erkannt ist, daß er einen RK erhalten soll, muß er dadurch das Ende der Gesamt-RKV abwarten, bis er diesen RK bekommt.

8.5 Je nachdem, welchen der in 8.2 angedeuteten Wege man einschlägt, besteht eine mehr oder weniger große Gefahr, daß am Ende der RKV RK-e benachrichtigt werden, die ihrerseits im Bereich der Nicht-Akteure sind.

Oft wird ein solcher RK den AKD aufrufen, so daß er den ihm zugewiesenen Akteur gar nicht erst übernimmt, sondern statt dessen die RKV durchläuft.

Oder aber, wenn er den AKD nicht (mehr) aufzurufen braucht, wird er mit der Übernahme des zugewiesenen Akteurs warten, bis er seine Tätigkeiten im Bereich der Nicht-Akteure abgewickelt hat; so lange muß dann die AKL oder zumindest ein Teil der AKL gesperrt werden.

8.6 Wenn alle betroffenen RK-e gleichzeitig aufgefordert werden, ihre Akteure zu wechseln, ergeben sich Koordinationsprobleme auf der AKL.

8.7 Die angeführten Probleme und Nachteile treten bei der RKV in Einzelschritten nicht auf.

Wir werden uns im Folgenden nur noch mit dieser beschäftigen.

9. Zur RKV in Einzelschritten

9.1 Nach unseren Überlegungen in 2.3, 5.1 und 7. lassen sich zwei Fälle unterscheiden:

9.1.1 Ein RK springt die RKV "von sich aus" an, weil er den Bereich der Nicht-Akteure verlassen will.

./. 21

9.1.2 Ein RK springt die RKV an, weil er von einem anderen RK dazu aufgefordert wurde.

Einer solchen Aufforderung geht i.a. eine Konsistenzprüfung durch den auffordernden RK voraus, bei der bereits festgestellt wird, welcher Akteur den aufgeforderten RK zu erhalten hat.

Wir setzen voraus, daß dem aufgeforderten RK zusammen mit der Aufforderung, die RKV zu durchlaufen, auch dieser Akteur mitgeteilt wird. Die RKV besteht dann im Wesentlichen nur darin, daß der RK den mitgeteilten Akteur übernimmt, - wir sprechen von RKV nach Aufforderung. Der Fall 9.1.1 wird als freiwillige RKV bezeichnet.

9.2 Grundsätzlich ist jeder RK, der sich im Bereich der Nicht-Akteure befindet, für die Konsistenz des Systems verantwortlich; das soll heißen: bevor er den Bereich der Nicht-Akteure verläßt, muß er prüfen, ob das System konsistent ist (genauer: konsistent sein wird, nachdem er den Bereich der Nicht-Akteure verlassen hat) und ggf. einen anderen RK zur RKV auffordern, d.h. ihm die Verantwortung für die Konsistenz übertragen.

9.2.1 Solange Reaktionsschnelligkeit keine Rolle spielt, genügt es, wenn jeweils ein RK die Verantwortung für die Konsistenz hat.

Daraus folgt für die träge RKV:

(9.2.1.i) Stellt ein RK am Ende einer RKV fest, daß ein anderer RK im Bereich der Nicht-Akteure ist, so braucht er sich um die Konsistenz des Systems nicht zu kümmern.

9.2.2 Für die reaktionsschnelle RKV ist dies nicht uneingeschränkt richtig.

Hier wird man stattdessen sagen:

(9.2.2.i) Wenn ein RK auf die AKL zugreift, hat er die Verantwortung für die Konsistenz des Systems.

./. 22

10. Verhalten der RKV gegenüber Unterbrechungen

10.1 Ein großer Teil der Hardware-Unterbrechungen wird Anderungen der AKL zur Folge haben. Daher wäre es grundsätzlich richtig, eine RKV, während derer eine Unterbrechung geschieht, abzubrechen, zunächst die Unterbrechung abzuhandeln, d.h. insbesondere, die AKL auf den neuesten Stand zu bringen und alsdann die RKV von vorne zu durchlaufen.

Wir bezeichnen die beschriebene Technik als Rücksetztechnik. Sie setzt voraus, daß die AKL in jedem Unterbrechungspunkt der RKV konsistent ist, - wir sagen, die RKV muß dann rücksetzbar programmiert sein.

10.2 Im Falle eines Mehrprozessorsystems scheint die Rücksetztechnik nicht günstig: ist für einen Akteur im ersten RKV-Schritt erkannt worden, daß er einen RK erhalten soll, so wird sich daran durch die Zustandsänderungen infolge einer Unterbrechung nicht unbedingt etwas ändern (der Akteur wird wahrscheinlich weiterhin zu den n wichtigsten nichtblockierten Akteuren gehören, wenn n die Anzahl der RK-e ist). Normalerweise wird es genügen, wenn die Änderungen erst bei dem nächsten RKV-Schritt berücksichtigt werden. - Es erscheint daher wünschenswert, AKL-Änderungen (insbesondere aufgrund von Unterbrechungen auf den anderen RK-en) und Hardware-Unterbrechungen (auf dem RK, der die RKV durchläuft), so lange zu sperren, bis ein RKV-Schritt beendet ist und dann, am Ende eines Schrittes, AKL-Änderungen sowie Unterbrechungen zuzulassen.

10.3 Normalerweise wird jedoch die Hardware des Systems verbieten, daß Unterbrechungen während der ganzen RKV gesperrt werden; dann bietet sich folgender Ersatz:

(10.3.i) die RKV wird eingriffsinvariant programmiert. (Das bedeutet wiederum, daß gewisse Teile der RKV unter Unterbrechungssperre laufen müssen.)

./. 23

(10.3.ii) Unterbrechungen während der RKV werden "vorbehandelt und gekellert", anschließend wird die RKV (= der RKV-Schritt) fortgesetzt und nach Ende der RKV werden die Unterbrechungen soweit weiterbehandelt, daß Aktualität der AKL garantiert werden kann.

10.4 Sollte das "vorbehandeln und kellern" einer Unterbrechung nicht relativ kurz sein im Verhältnis zur Dauer einer RKV, so wird darüber hinaus vorgeschlagen:

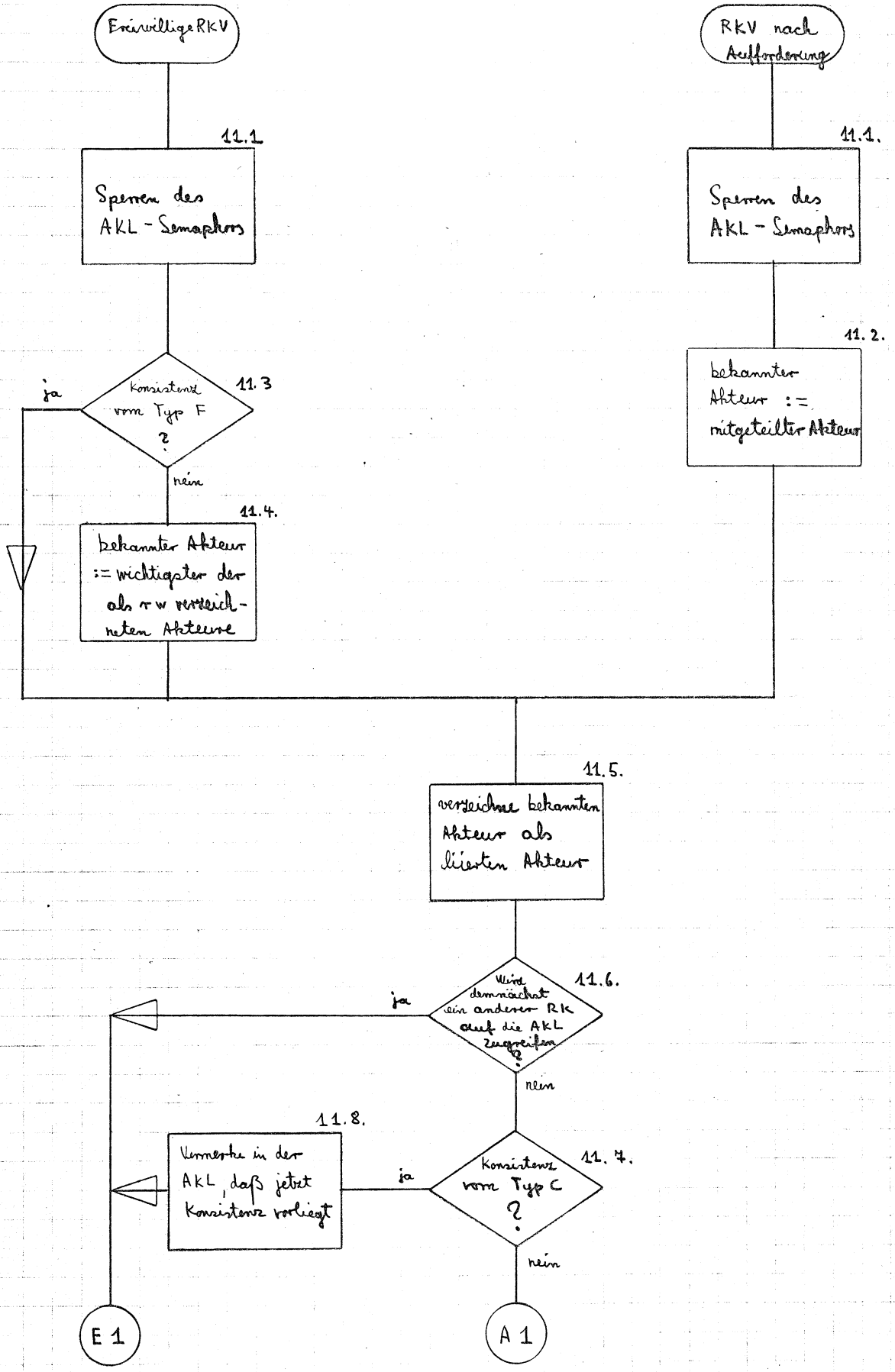
(10.4.i) Die RKV wird rücksetzbar programmiert.

(10.4.ii) Beim Beginn der "Vorbehandlung" wird ein Vermerk gesetzt: "RKV wurde unterbrochen, während sie auf RK k lief", und der AKL-Semaphor wird entsperrt.

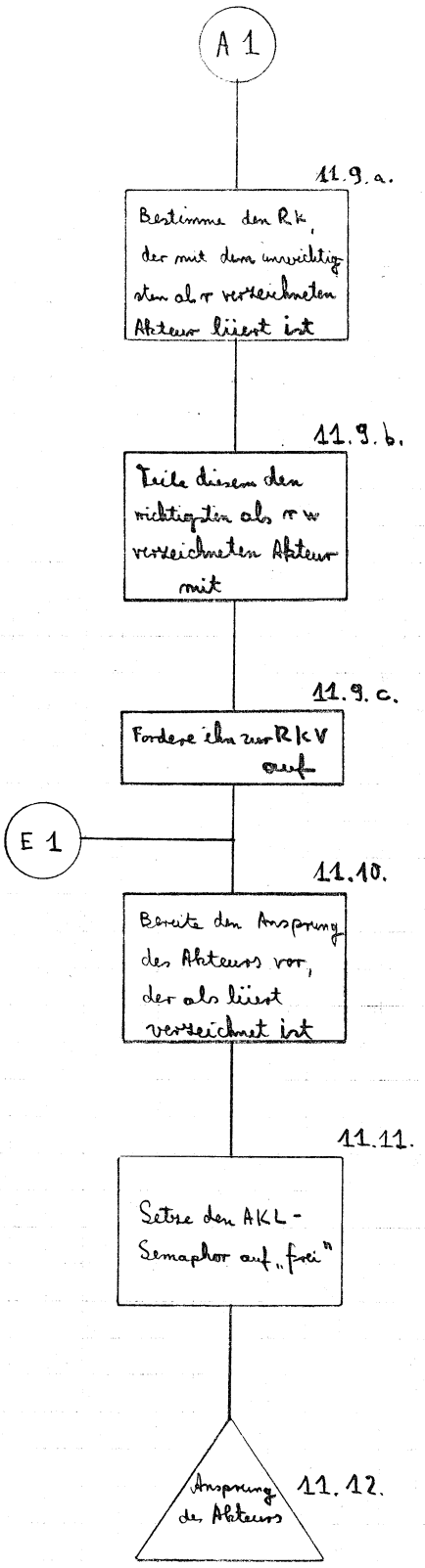
Wenn nun gerade ein anderer RK auf die AKL zugreifen möchte, braucht er nicht in Warteschleife zu gehen. Sobald er auf die AKL zugreift, löscht er den o.a. Vermerk und die RKV wird nicht fortgesetzt (Rücksetztechnik). Ist die Vorbehandlung beendet, der AKL-Semaphor frei und der o.a. Vermerk noch gesetzt, so wird die RKV fortgesetzt.

./. 24

11. Flußdiagramm der RKV (reaktionsschneller Typ)



./. 25


A 1
11.9.a.
Bestimme den R.k., der mit dem unrichtig oder verzeichneten Akteur liiert ist
11.9.b.
Teile diesem den richtigen als o.w. verzeichneten Akteur mit
11.9.c.
Fordere ihn zur R|K V auf
E 1
11.10.
Bereite den Ausgang des Akteurs vor, der als liiert verzeichnet ist
11.11.
Setze den AKL-Semaphor auf „frei"
Ausgang des Akteurs
11.12.

./. 26

Erläuterungen zum Flußdiagramm

11.1 Vergleiche hierzu 3.2.6! - Wird der Semaphor als gesperrt vorgefunden, so geht der RK in eine Abfrageschleife. Falls für die P-Operation die Unterbrechungssperre gesetzt wird, ist diese in der Abfrageschleife wieder zu lösen. (Unterbrechungssperre = die stärkste (zulässige) Sperre gegen Hardware-Unterbrechungen).

11.2 Der Akteur, der dem RK mit der Aufforderung zur RKV mitgeteilt wurde, wird zu dem Akteur, der dem RK "bekannt" ist (im Sinne von 2.2).

11.3 Dem RK ist ein Akteur bekannt. Ansprung dieses Akteurs führt sicher dann zu Konsistenz vom Typ C, wenn vorher Konsistenz vom Typ F vorliegt. Zur technischen Realisierung dieser Abfrage vgl. 11.8!

11.4 -

11.5 In der AKL wird jetzt bereits eingetragen, daß der bekannte Akteur "liiert" (im Sinne von 2.2) ist mit dem RK, der die RKV durchläuft, d.h. daß dieser RK den Akteur rechnet. (Das ist vorläufig noch falsch!)

11.6 Die Bedeutung von "demnächst" ist abhängig vom Grad der Reaktionsschnelligkeit, den man anstrebt. Die Abfrage endet jedenfalls immer dann mit "ja", wenn bereits ein anderer RK vor dem AKL-Semaphor in einer Abfrageschleife ist.

Endet die Abfrage mit "ja", so ist sichergestellt, daß (mindestens) ein anderer RK die Verantwortung für die Konsistenz des Systems hat, und der abfragende RK braucht sich um die Konsistenz nicht zu kümmern. Technisch kann diese Abfrage auf verschiedene Weisen realisiert werden, z.B. indem man für jeden RK eine Variable schafft, die angibt, ob dieser RK demnächst auf die AKL zugreifen will.

./. 27

11.7 Nachdem nun der Akteur, den der RK zu rechnen beabsichtigt, bereits als rechnend verzeichnet ist, prüft der RK, ob jetzt Konsistenz vom Typ C vorliegt.

11.8 Es ist zweckmäßig, das positive Ergebnis einer Konsistenzprüfung in der AKL zu vermerken. Es wird ein Bit gesetzt mit der Bedeutung: "Jetzt liegt sicher Konsistenz vom Typ E vor". Übergänge der Form bekannt → liiert und liiert → "bekannt und rw" ändern nichts an der Richtigkeit dieser Aussage und haben keinen Einfluß auf den Vermerk.

Werden jedoch andersartige Zustandsänderungen in die AKL eingetragen, so wird der Vermerk gelöscht, - unabhängig davon, ob weiterhin Konsistenz vom Typ E vorliegt oder nicht. Dieser Vermerk kann die Abfrage 10.3 wesentlich erleichtern.

11.9 Liegt keine Konsistenz vor, so muß ein anderer RK benachrichtigt und zur RKV aufgefordert werden. Bei der Konsistenzprüfung 11.7 wird die Prioritätsnummer des wichtigsten als rw verzeichneten Akteurs mit der des unwichtigsten rechnenden Akteurs verglichen.

Jetzt wird der RK ermittelt, auf dem letzterer rechnet, und diesem RK wird der schon bekannte wichtigste rw Akteur mitgeteilt.

11.10 Hier wird der "Unterbrechungszustand" des anzuspringenden Akteurs geladen, d.h. insbesondere, es werden Register und Steuerbits geladen.

11.11 Grundsätzlich gehört auch der "Unterbrechungszustand" eines Akteurs zu der Information, die durch den AKL-Semaphor zu schützen ist. Daher darf der AKL-Semaphor erst wieder auf "frei" geschaltet werden, wenn auf diese Information nicht mehr zugegriffen zu werden braucht. (Normalerweise ist dies unmöglich, so daß man, je nach der speziellen Situation, zu Hilfskonstruktionen greifen muß, die entsprechendes leisten).

11.12 -

./. 28

12. Die träge RKV

Während sich die reaktionsschnelle RKV, wie sie oben beschrieben wurde, an der Konsistenz vom Typ C orientiert, richtet sich die träge RKV an der Konsistenz vom Typ E aus. Dem liegt die Annahme zugrunde, daß ein RK sich immer nur relativ kurz im Bereich der Nicht-Akteure aufhält. Entsprechend gibt es einige Unterschiede zwischen träger und reaktionsschneller RKV:

11.3 ist hier zu ersetzen durch: "Konsistenz vom Typ E?"
11.6 ist hier zu ersetzen durch: "Ist ein anderer RK im Bereich der Nicht-Akteure?"
11.7 ist hier zu ersetzen durch: "Konsistenz vom Typ E?"

13. Zusammenfassung

Es werden Probleme des lokalen Rechnerkern-Scheduling in einem Mehrprozessor-System diskutiert.

Das Programm "Rechnerkernvergabe" (RKV) wählt aus einer vorgegebenen Menge von sich bewerbenden Programm-Moduln (Akteuren) diejenigen aus, denen die Rechnerkerne des Systems zuzuteilen sind und vollzieht diese Zuteilung. Dabei wird für die Akteure eine- lokal fixe - lineare Ordnung nach Prioritäten vorausgesetzt.

Aus dieser Prioritäts-Ordnung ergibt sich der Begriff "Konsistenz". Er wird diskutiert, und als Aufgabe der RKV wird erkannt, Konsistenz in einem bestimmten Sinne herzustellen und zu erhalten.

Es werden zwei Lösungsverfahren für diese Aufgabe angegeben, von denen eines als zweckmäßiger und leichter zu handhaben erkannt wird. Dieses Verfahren (Einzelschrittverfahren) wird in zwei Spielarten (träge - reaktionsschnell) dargestellt.

./. 29

Die Arbeit geht nicht auf Probleme der Datenstruktur für die benutzten Variablen ein. - Über das Gesamtsystem wird vorausgesetzt, daß mehrere Rechnerkerne existieren, die sowohl hardwaremäßig gleichartig sind als auch vom Betriebssystem gleichartig eingesetzt werden. Im übrigen versucht die Arbeit, mit einem Minimum an Voraussetzungen über die Hardware wie auch über die umgebende Software auszukommen.

14. Index

./. 30