MTA Számitástechnikai és Automatizálási Kutató Intézete, Közlemények 18/1977.

PARALLEL FOLYAMATOK GRÁF MODELLJEI

Szlankó János
MTA Központi Fizikai Kutató Intézet

1. Bevezetés

A programozás alapvető problémája a programok korrektségének a bizonyitása. Ez különösen nehéz feladatot jelent akkor, ha egyes programrészek párhuzamos feldolgozása is megengedett, sőt szükséges mint például operációs rendszerekben, real-time rendszerekben. Szükséges egy preciz és lehetőleg szemléletes elméleti eszköz, amelynek segitségével az ilyen rendszerek konstuálása megkönnyithető, specifikáció szerinti müködésük ellenőrizhető. A korábbiakban a konstruálás valamilyen ad hoc módon történt, a helyes müködést pedig teszt feladatok elvégzésével valószinüsitették. A 60-as évek végén jelentek meg a Dijkstra féle P és V operációk. Ezek alkalmazásával már lehetett kérdéseket feltenni és megoldani a rendszer korrektségével kapcsolatban, de a bizonyitások kissé nehézkesek. Az utóbbi években kezdik alkalmazni a Petri hálókat parallel folyamatok modellezésére. A Petri hálók segitségével egy parallel feldolgozható programrendszernek olyan tulajdonságait (kommunikációs, szinkronizációs) állapithatjuk meg, amelyek függetlenek a részfolyamatok végrehajtási idejétől, tervezésnél pedig felhasználhatók arra, hogy a megvalósitandó programrendszernek bizonyos tulajdonságai függetlenek legyenek a részfolyamatok végrehajtási idejétől tehát a kontrol szerkezet biztositsa, tekintet nélkül a processzor ütemezési stratégiájára). Az alábbiakban a Petri hálók elméletéből ismertetünk néhány hasznos fogalmat, eljárást és alkalmazást.

2. Petri hálók

Egy *Petri háló* egy négyes $N = (P,T,per,post)$, ahol P és T véges diszjunkt halmazok, *pre* és *post* pedig bináris relációk:

$$pre \in PXT$$
$$post \in PXT$$

P és T minden elemének szerepelni kell legalább egy reláció elemben. P elemeit *helyek*nek, T elemeit *tranziens*eknek nevezzük. $p_1 \epsilon P$ input helye a $t \epsilon T$-nek, ha $(p_1,t) \epsilon\ pre$, $p_2, \epsilon P$ output helye $t \epsilon T$-nek, ha $(p_2,t) \epsilon\ post$.

Egy Petri hálót szemléletesebben megadhatunk egy irányitott páros gráffal. A helyeket körökkel, a tranzienseket vonással jelöljük. Ha $(p_1\ t) \epsilon\ pre$, akkor p_1-ből vezet egy él t-be, ha $(p_1,t) \epsilon\ post$, akkor t-ből vezet egy él p_1-be.

Ugyanezt a Petri hálót számitási célokra mátrox formában is megadhatjuk, ha a háló *tisz-*

ta : nincs olyan hely, amely egy tranziensnek az input és output helye is. Egy N Petri hálóhoz tartozó C incidencia mátrixot a következőképpen definiáljuk: a sorokat a P halmaz elemeivel, az oszlopokat a T halmaz elemeivel indexeljük és

$$C(p,t) = \begin{cases} -1, & \text{ha } (p,t) \in pre \\ +1, & \text{ha } (p,t) \in post \\ 0 & \text{egyébként} \end{cases}$$

Az 1. ábra hálóját a következő mátrixszal adhatjuk meg:

$$\begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \end{array}$$

Egy $M : P \to I$ függvényt az N háló pontozásának nevezzük. I a nemnegativ egészek halmaza. Ha $p \in P$, akkor $M(p)$ a p helyen lévő pontok száma. Egy helyet pontozottnak nevezünk, ha $M(p) \geqslant 1$. Az M függvényt egy vektorral adhatjuk meg, ahol a vektor elemeit a helyekkel indexelhetjük. A (N,M) párt *pontozott Petri hálónak* nevezzük.

Egy $t \in T$ tranziens *aktivált* egy adott M pontozás alatt, ha $M \geqslant (B(-,t))$ ahol

$$B(p,t) = \begin{cases} 1, & \text{ha } p \text{ input helye } t\text{-nek} \\ 0 & \text{egyébként} \end{cases}$$

A fenti egyenlőtlenség azt jelenti, hogy a hálóban legalább t input helyei pontozottak. Egy aktivált $t \in T$ tranziens *átbillenése* az M pontozást egy M_1 pontozásba transzformálja ugy hogy $M_1 = M + C(-,t)$. Ez az jelenti, hogy a t input helyein a pontok száma eggyel csökken és minden output helyén eggyel megnő a pontok száma.

A háló *kiértékelése* során az aktiv tranziensek közül (ha van egyáltalán) egy tetszőleges kiválasztásra kerül, átbillen. Ezzel előáll egy új pontozás, amelyen folytatódhat a kiértékelés.

3. A Petri hálókkal kapcsolatos legfontosabb kérdések

Ha egy M kezdeti pontozású háló minden p helyére igaz, hogy a háló müködése során p helyen lévő pontok száma egy k korlátnál nem lehet nagyobb, akkor a háló *k-biztonságosnak* mondjuk. Ez azt jelenti, hogy az M-ből elérhető pontozások $[M]$ halmaza véges. (M_i pontozásból M_j *elérhető*, ha a kiértékelés során van a billenések olyan S sorozata, amely az M_i pontozást M_j-ben viszi át). A biztonság kérdésének eldönthetősége nagyon fontos a további vizsgálatok szempontjából. Szerencsére a kérdés eldönthető [5]. A Petri hálók részosztályai-

ra vonatkozó biztonsággal foglalkozik [8], [9], [10]. A továbbiakban csak biztonságos hálókkal foglalkozunk. A gyakorlatban előforduló hálók biztonságosak, vagy viszonylag egyszerü módositással funkciójuk megzavarása nélkül biztonságossá tehetők.

Egy (N,M) hálóhoz hozzárendelhető egy *tranziens-gráf*, amelyet fontos kérdések megválaszolásához használhatunk fel. A tranziens gráf irányitott gráf, amely a következőképpen konstruálandó.

A szögpontokat az $[M]$, az éleket T elemeivel fogjuk indexelni. Legyen a konstruáláshoz a kezdőpont M. Az M esetben aktivált összes tranziensre nézzük meg, hogy milyen pontozásba viszi át M-t, és a különböző pontozásokhoz rendeljünk különböző szögpontokat, mindegyikbe élt irányitva M-ből, az éleket a megfelelő tranzienssel indexelve. Válasszunk ki egy másik M_1 szögpontot (pontozást) és ismételjük meg az eljárást. Folytassuk ezt addig, amig minden – a közben keletkező szögpontokra is – el nem végeztünk. A konstruálás biztonságos gráfoknál nyilván mindig véges lépés után végetér. Példaként nézzük a 2. ábrát.

A gráf alapján az elérhetőség is eldönthető: M_i pontozásból M_j elérhető, ha van olyan irányitott út, amely M_i-ből M_j-be vezet.

Egy M_i *holt pontozás,* ha nincs aktivált tranziens. Ez a gráfban azt jelenti, hogy M_i-ből nem vezet ki él. Egy M_i pontozás *életképes,* ha nem mehet át holt pontozásba.

Egy M_i pontozás *reprodukálható* ha van a billenéseknek olyan sorozata, amely önmagába viszi át. Egy $[M]$ reprodukálható, ha minden eleme reprodukálható. A tranziens gráfban ez azt jelenti, hogy M_i egy irányitott körön helyezkedik el, illetve a gráf erősen összefüggő.

Az $M_j = M_i + CX$ egyenletrendszer nemnegativ egész megoldásaiban egy $t \epsilon T$ tranziensre $X(t)$ azt jelenti, hogy t-nek hányszor kellene billeni ahhoz, hogy M_i átmenjen M_j-be. Az hogy egy X megoldásvektor realizálható-e a tranziesgráf alapján eldönthető. Ha C' jelenti C transzponáltját, a $C'.\ z = 0$ egyenlet nemnegativ egész megoldásait *invariánsoknak* nevezzük, ugyanis

$$M_j'z = (M_i + CX)' \cdot z = M_i'z + (CX)' \cdot z =$$
$$= M_i'z + X'C'z = M_i'z \quad \text{ha} \quad C'z = 0$$

Ha a z vektor minden eleme a $\{0,1\}$ halmazból kerül ki, akkor $M'z$ egyenlő azon pontok összegével, amelyek azon a helyeken találhatók amelyekhez 1 tartozik a z vektorban. Innen származik az invariáns elnevezés: ez az összeg a háló müködése során nem változik.

4. A hálók alkalmazása

a.) Ha a helyeknek feltételeket, a tranzienseknek eseményeket (az események pillanatszerüek) feleltetünk meg, akkor az olyan folyamatok, amelyek csak induláskor és befejezéskor lépnek kapcsolatba a környezetünkkel méghozzá úgy, hogy az indulás feltételektől függ, befejezéskor pedig feltételeket állitanak be feltétel-esemény szempontjából a 3.a. szerint modellezhetők.

Petri hálókkal csak, olyan folyamatokat modellezhetünk, ahol a feltétel-esemény szerkezet időben nem változik.

b.) Mivel a program feldolgozása is folyamat, ezen a téren is alkalmazhatjuk a Petri hálókat. A program alapján létrehozhatunk egy hálót: feleltessünk meg a program változóit számára történő értékadásnak a 3.a. ábrát (egy be- és egy kimeneti feltétellel), a feltételes elágazásnak a 3.b. ábrát, és ha program feltétel változókon való müveletként tartalmazza a P és V operációkat [1], [2], akkor azoknak a 3.c. ábrát. Az igy kapott háló nem tartalmaz információt a müveletek, predikátumok interpretációjáról, tehát a program olyan tulajdonságai vizsgálhatók, amelyek bármilyen interpretációt mellett igazak. (Az interpretációra vonatkozó ismeretek is bevihetők a hálóba, de ennek korlátai vannak: Petri hálókkal nem valósitható meg minden kiszámitható függvény [5].) A 4. ábrán látható egy program, amely a feldolgozandó adatokat egy bufferban kapja és egy másik bufferbe helyezi az eredményt. Mellette látható egy háló, amelyet a fenti szabályok szerint képeztünk. A hálón a vizsgált kérdéstől függően egyszerüsitések végezhetők: ha például a tranziensgráfot csak a holtpont keresésre akarjuk felhasználni, a háló több eleme összevonható.

c.) Ha már adott a folyamat hálója és a kezdeti pontozás a tranziens gráfon vizsgálhatók a holtpont, kölcsönös kizárás stb. A kölcsönös kizárást az invariánsok segitségével kényelmesebb lehet, mert igy nem kell az esetleg negyméretü gráfot létrehozni. (A lineáris algebra még számos, a Petri hálók alosztályait érintő kérdéshez felhasználható [9]).

Nézzük az 5. ábra hálóját. A háló egyik invariánsa $z' = (1\ 0\ 1\ 1\ 0)$ vektor. Ha a kezdeti pontozás az ábrán kijelölt vagyis $M' = (1\ 1\ 0\ 0\ 1)$, akkor $M'z = 1$. Ez azt jelenti, hogy P_1, P_3, P_4 helyek közül mindig csak egy tartalmaz 1 pontot.

5. A Petri hálók módositásai

a.) A Petri hálók tranziensei *(AND, AND)* tipusuak: minden bemeneti helyről elvesz egy pontot, és minden kimeneti helyre tesz egy pontot. [6]-ban bevezetésre kerülnek *(EOR, AND)*, *(AND, EOR)*, *(EOR,EOR)* jellegü tranziensek. Ha a tranziens input oldala "kizáró vagy"' jellegü, akkor csak egy input helyről vesz el pontot, ha az output oldal *EOR*, akkor csak egy helyre tesz pontot. Az *EOR* jellegü tranziensek használata esetenként kényelmesebb például a döntés *EOR* kimenetü tranzienssel modellezhető. A tranziens gráf változatlanul használható.

b.) Ha egy real time program korrektségével kapcsolatos kérdésekkel probáljuk eldönteni a Petri hálók segitségével, akkor ábrázolnunk kell hálókkal az operációs rendszernek szóló utasitások hatását. Az ütemezési (inditás, inditás periodusokon), kommunikációs (*SEND, RECEIVE*) utasitások hatása még viszonylag egyszerüen leirható. A 6. ábrán látható egy példa. Sajnos a jelenlegi operációs rendszerekben csak ritkán vannak megvalósitva a P és V operációk, és az megnehezeti a vizsgálatokat. A feltétel változók helyett csak flagekat használnak. Például ilyen a *PDP/11 RSX 11/D* operációs rendszere. A flageken a *CLEAR, SET, WAIT* müveletek vannak definiálva. A *CLEAR* és *SET* utasitások mindig végrehajthatók, a *WAIT* utasitás hatására a task feldolgozása megszakad, ha nincs minden flag beállitva. Ezen utasitásoknak a sze-

mantikája a folyamtok az egymáshoz szinkronizálás szempontjából Petri hálókkal leirható (8. ábrán a *CLEAR* és *SET* hatása) de csak bonyolultan, igy elvész a szemléletesség és a gyorsaság. A másik megoldás, hogy a módositott Petri hálóba bevezetjük a flageket is és olyan tipusu tranzienseket is megengedünk mint amilyenek a 7. ábrán láthatók.. Igy ugyan definiciós szempontból a háló bonyolulttá vált és nem lehet könnyen találni hasznos tételeket, de a tranziens gráf módszer továbbra is használható. Ezen pedig a holtpont, szinkronizáció, kölcsönös kizárás vizsgálható. Az eljárások programozhatók, amelyek a vizsgálandó program szövegét inputként kapják. Az értékadásokhoz, predikátumokhoz, szinkronizációs, kommunikációs, ütemezési utasitásokhoz rendelt hálókból a szöveg alapján összerakható a program hálója.

I r o d a l o m

[1] E.W. Dijkstra, Cooperating Sequential Processes. In programming Languages, F. Genuys Ed., Academi Press, New York, 43-112. (1968)

[2] E.W. Dijkstra, Hierarhical Ordering of Sequential Processes, Acta Informatica 1,2 115-115-138 (1971).

[3] C.A. Petri, Concepts of Net Theory, Proceedings of the Symposium on Mth. Foundations of Computer Science, High Tatras, 137-146. (1973)

[4] K. Lautenbach, M.A. Schmid, Use of Petri Nets for Proving Correctness of Concurent Process Systems, Proc. of the IFIP Congress, II. 187-191 (1974)

[5] R.M. Keller, Vector Replacement Systems: A Formalims for Modelling Asyncronous Systems, TR 117, Dept. of EEcs, Prineton Unversity, January, (1974)

[6] J.L. Baer, Modelling for Parallel Computation: A Case Study, Proc. of the Computer Conference on Parallel Processing, Sagamore, (1973)

[7] F.Commoner et all, Marked Directed Graphs, Journal of Computer and System Sciences, 5, 511-523 (1971).

[8] M.H.T. Hack, Analysis of Production Scemata by Petri Nets. MAC TR-94, Massachusetts, 1972.

[9] K. Lautenbach, Exakte Bedingungen der Lebendigkeit für eine Klasse van Netren, Dissertation, Universitat Bonn, (1973)

[10] K.P. Gostelow, Computation Moduls and Petri Nets. TR # 61 Dep. of Information and Computer Sciences, University of California, Irvine, (1965)

[11] K.P. Gostelow, Proper Termination of Flow of Control in Program Involving Concurent Processes, SIGPLAN Notices, 7. 11. nov. (1972).

[12] K.P. Gostelow, A model of Processes Based on Petri Nets, TR # 69, Dep. of Information and Computer Science, University of California, Irvine, (1975)

[13] J.L. Paterson, Modelling of Parallel Systems, Stanford, Ph. D. dissertation, December (1973) (NTIS # PB 231 926)

[14] T. Agarwala, Complete model for Representing the Coordination of Asynchronous Processes, Hopkins Computer Research Report # 32 Computer Science Program, John Hopkins University, Baltimore, Maryland, July (1974).

[15] J.L. Baer, A survey of some Theoretical Aspests of Multiprocessing, Computing Surveys 5, No. 1, March (1973)

[16] R.M. Karp and R.E. Miller, Parallel Program Schemata, J. Computer and System Sciences 3 147-195 (May 1969)

Summary

Graph Modells of Parallel Processes

János Szlankó

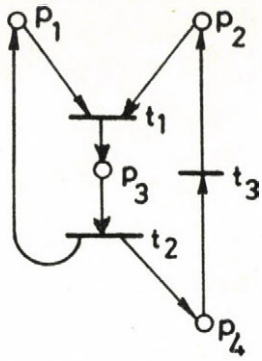
The theory of parallel programming is of great practical importance in the field of operating systems and real time programs. Recently some new graph modells have been introduced. In this area Petri net looks the most promasing formal construction. It has stimulated the researches for further investigations and applications to some practical problems have been demo demonstrated. The article defines the basic concepts of Petri nets and shows the way for mechanisable proofs of parallel program properties.

Р Е З Ю М Е

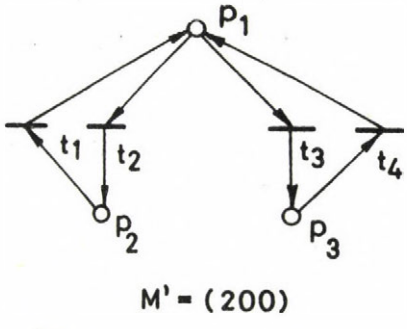
МОДЕЛИ ГРАФОВ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОРОВ

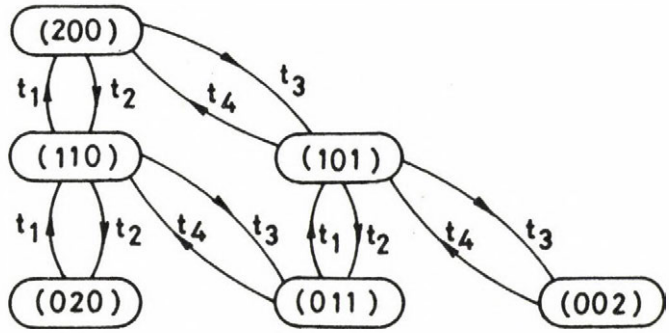
Я. Сланко

Важным практическим вопросом является анализ проведения программ, обрабатываемых параллельно. За последние годы в этой области были сделаны попытки с разными моделями графов. Самой хорошей из них является сеть Петри, которая способствовала возникновению более обобщенных моделей и возникло несколько областей применения. Данная статья рассматривает основы теории сетей Петри и автоматизацию доказательств корректировки.



1. ábra



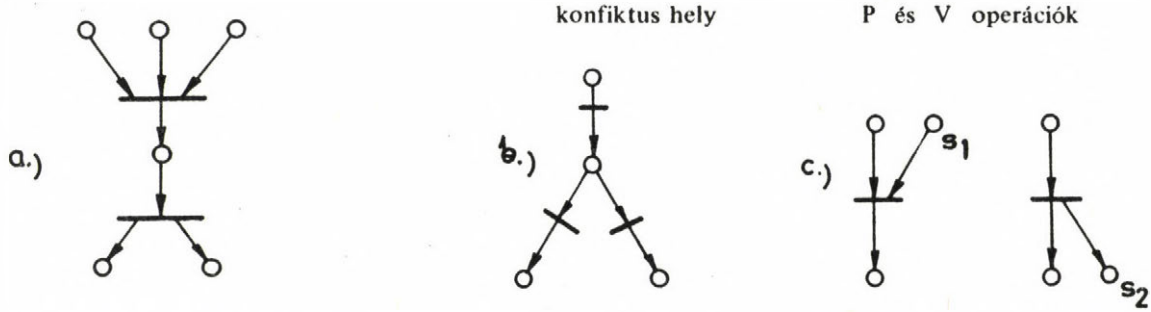


2. ábra

a folyamat indulásához
szükséges feltételek

konfiktus hely

P és V operációk

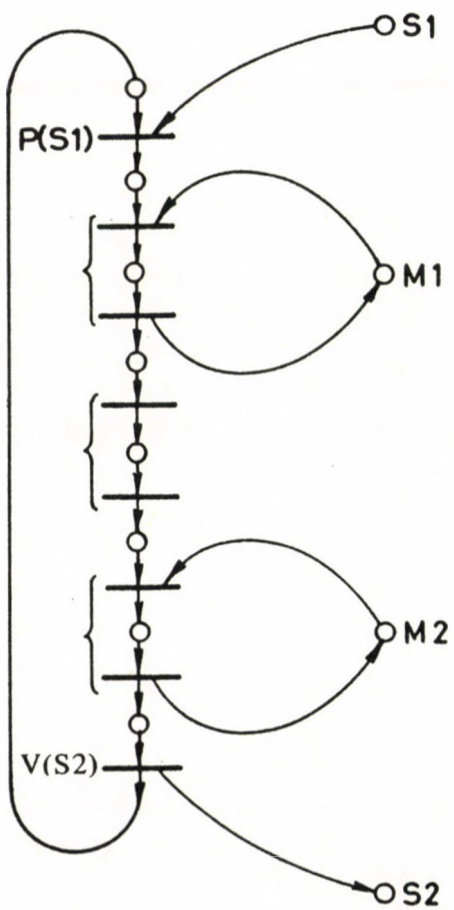


a folyamat befejeződésekor
előálló feltételek

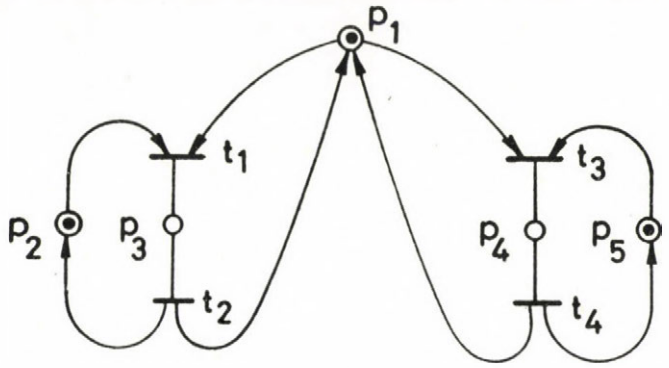
3. ábra

```
C: P(S1);
   P(M1);
   input bufferból olvasás;
   V(M1);
   feldolgozás:
   P(M2);
   output bufferba irás;
   V(M2);
   V(S2);
   goto C;
```

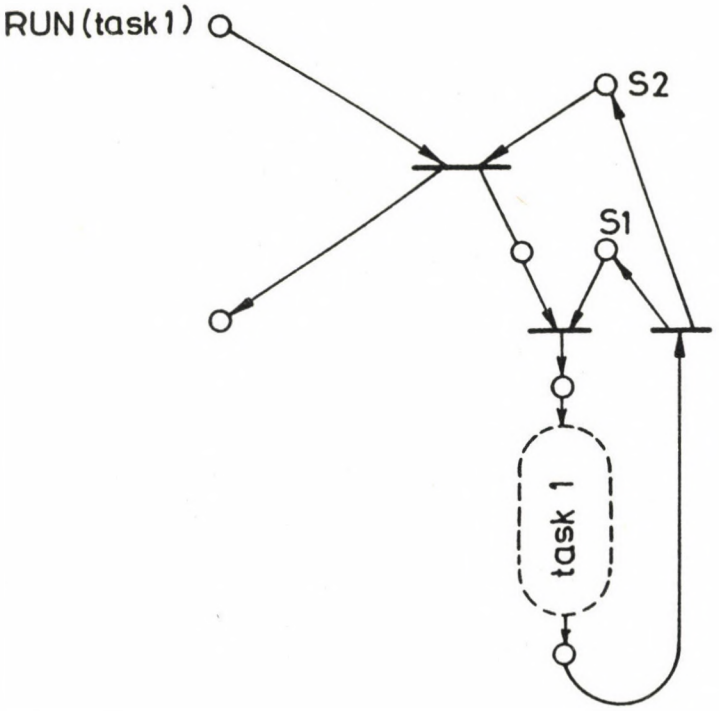
```
P(M1);
input bufferból olvasás;
V(M1);

feldolgozás;

P(M2);
output bufferba irás;
V(M2);
```



4. ábra



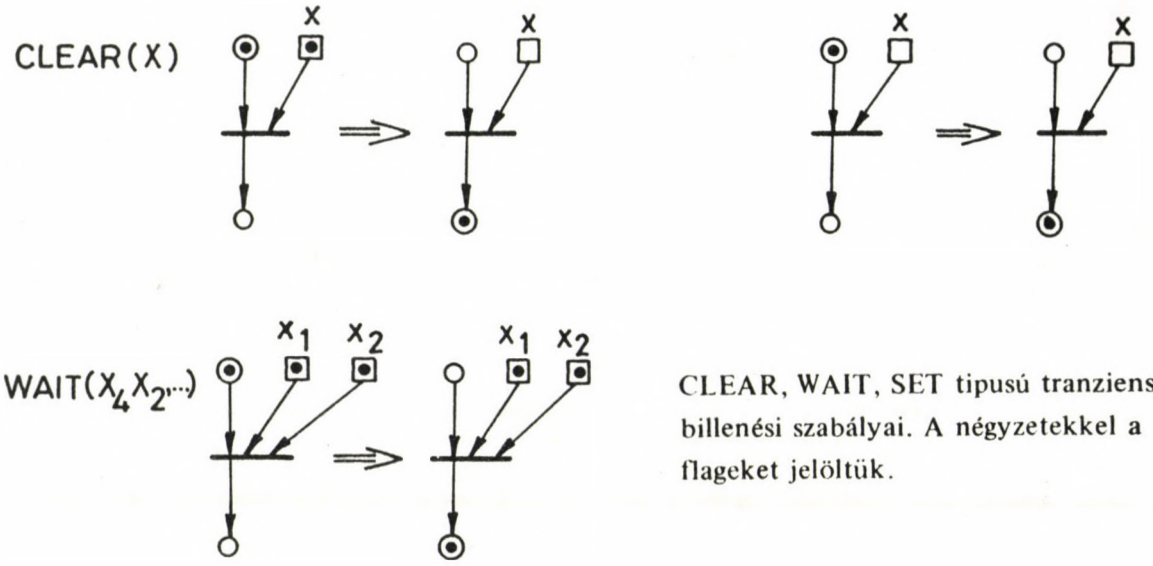
5. ábra



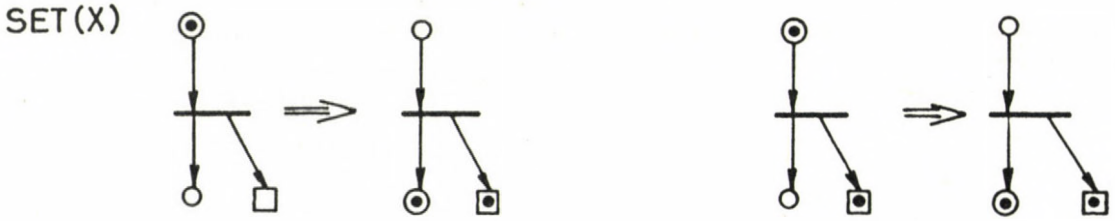
kezdetben: $M(S_1) = 1$

$M(S_2)$ = várólista max hossza

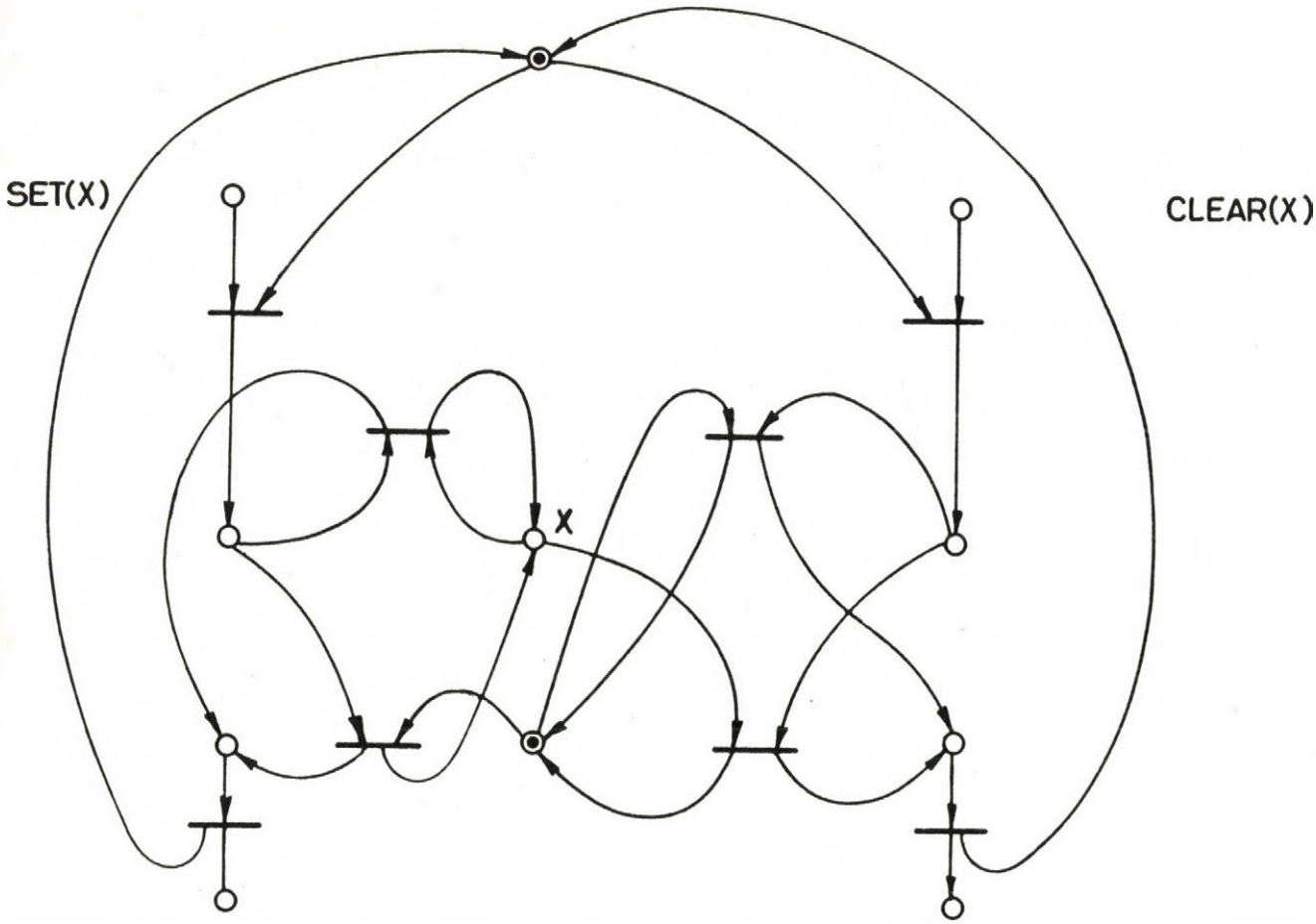
6. ábra



CLEAR, WAIT, SET tipusú tranziensek billenési szabályai. A négyzetekkel a flageket jelöltük.



7. ábra



8. ábra