Logistică Industrială și Comercială

Curs pentru master

Sisteme Informatice Aplicate în Producţie şi Servicii

Dr.ing. Loredana STANCIU, PhD

Cuprins

1 Modelare cu rețele Petri (1)

1.1 Introducere

Creșterea în complexitate a sistemelor industriale moderne, precum producția, controlul procesului, sisteme de comunicații etc., a indus apariția a numeroase probleme privind dezvoltarea acestora. În faza de planificare apare confruntarea cu capabilitățile crescute ale acestor sisteme, datorită combinațiilor unice de hardware și software care operează sub un număr mare de constrângeri ce rezultă din resursele limitate ale sistemului. În condițiile naturii complexe și intensive a capitalului sistemelor moderne industriale, designul și operarea acestora necesită modelare și analiză pentru selectarea alternativei optime de design și a politicii de operare. Este binecunoscut faptul că fluxul în procesul de modelare poate contribui substanțial la timpul și costul de dezvoltare. Chiar și eficiența operațională poate fi afectată. Din acest motiv, o atenție specială trebuie acordată corectitudinii modelor care sunt folosite la toate nivelurile de planificare.

Ca unelte grafice și matematice, rețelele Petri asigură un mediu uniform pentru modelare, analiză formală și design al sistemelor cu evenimente discrete. Unul dintre principalele avantaje al folosirii rețelelor Petri îl constituie faptul că același model este folosit atât pentru analiza proprietăților comportamentale și evaluarea performanțelor, cât și pentru construcția sistematică a simulatoarelor și controlerelor cu evenimente discrete. Rețelele Petri au fost numite după Carl A. Petri, care a creat în 1962 o unealtă matematică sub formă de rețea pentru studiul comunicării cu automatele. Dezvoltarea lor ulterioară a fost ușurată de faptul că rețelele Petri pot fi folosite pentru modelarea unor proprietăți precum sincronizarea proceselor, evenimente asincrone, operații concurente, rezolvarea conflictelor sau partajarea resurselor. Aceste proprietăți caracterizează sistemele cu evenimente discrete care includ sistemele automate industriale, sistemele de comunicare și sistemele bazate pe calculator. Toate acestea transformă rețelele Petri într-o unealtă promițătoare și o tehnologie pentru aplicații în automatizări industriale.

Ca unealtă grafică, rețelele Petri asigură un puternic mediu de comunicare între utilizator (de regulă, inginer) și client. Cerințele complexe din caietele de sarcini pot fi reprezentate grafic folosind rețele Petri în locul unor descrieri textuale ambigue sau al unor notații matematice dificil de înțeles de către client. Acest aspect, combinat cu existența unor unelte computerizate care permit simularea grafică interactivă a rețelelor Petri, asigură inginerilor de dezvoltare o unealtă puternică ce să îi asiste în procesul de dezvoltare al sistemelor complexe.

Ca unealtă matematică, un model de rețea Petri poate fi descris de un set de ecuații lineare algebrice sau de alte modele matematice care să reflecte comportamentul sistemului. Acest lucru permite o verificare formală a proprietăților asociate comportamentului sistemului vizat (relații de precedență între evenimente, operații concurente, sincronizările necesare, eliminarea situațiilor de blocare *(deadlock)*, activitățile repetitive și excluderile mutuale ale resurselor partajate, pentru a aminti câteva dintre ele).

Validarea modelului prin simulare poate doar produce un set limitat de stări ale sistemului modelat, și astfel poate arăta doar prezența (nu și absența) erorilor din model și specificațiile sale de bază. Abilitatea rețelelor Petri de a verifica formal modelul este importantă în mod special pentru sistemele în timp real critice din punct de vedere al securității, precum sistemele de control al traficului aerian, sistemele de control al traficului feroviar, sistemele de control al reactoarelor nucleare etc. Rețelele Petri au fost folosite pentru modelarea sistemelor de timp real tolerante la defectare și critice din punct de vedere al securității, pentru detectarea erorilor și pentru monitorizarea proceselor.

Un domeniu de succes de aplicare al rețelelor Petri îl constituie modelarea și analiza protocoalelor de comunicare (încă de la începutul anilor ’70). În ultimii ani au fost propuse câteva abordări care permit construirea modelelor de rețele Petri pentru protocoale din specificațiile scrise într-un limbaj relativ nespecializat.

Rețele Petri, însă, au fost folosite în mod extensiv pentru modelarea și analiza sistemelor de producție. În acest domeniu, rețeaua Petri reprezenta linii de producție cu buffer-e, sisteme automate de producție, sisteme flexibile de producție, linii automate de asamblare, sisteme cu partajarea resurselor și, recent, sisteme de producție de tip *just-in-time*.

Un alt domeniu de succes îl constituie aplicarea rețelelor Petri în modelarea controlerelor secvențiale. Controlerele logice programabile (PLC) sunt folosite în mod uzual pentru controlul secvențial al sistemelor automate. Ele sunt proiectate folosind diagrame logice scară *(ladder logic diagrams)*, care sunt cunoscute ca fiind dificile de depanat și modificat. Controlerele secvențiale bazate pe rețele Petri, pe de altă parte, sunt ușor de proiectat, implementat și întreținut. La începutul anilor ’80, Hitachi Ltd. a dezvoltat un controler secvențial bazat pe rețele Petri care a fost folosit cu succes în aplicații reale pentru controlul sistemului de asamblare a pieselor și în sistemul automat de încărcare/descărcare din depozit. Utilizatorii rețelelor Petri, conform statisticilor, au redus substanțial timpul de dezvoltare, în comparație cu metodele tradiționale.

Rețele Petri au fost folosite extensiv și în dezvoltări software. Utilizarea în acest domeniu s-a concentrat pe modelarea și analiza sistemelor software, iar cea mai complexă dezvoltare a implicat folosirea rețelelor Petri colorate. S-a demonstrat că acest tip de rețele Petri este un limbaj folositor pentru proiectarea, specificarea, simularea, validarea și implementarea sistemelor software complexe.

Ca unealtă matematică, rețelele Petri permit evaluarea performanțelor sistemelor modelate. Performanțele deterministice și stocastice pot fi măsurate și evaluate folosind o gamă largă de modele de rețele Petri care încorporează în definiția lor funcții de timp deterministice și/sau probabilistice. Evaluarea performanțelor poate fi realizată fie prin tehnici analitice, bazate pe rezolvarea proceselor (semi)Markov de bază, sau prin simularea cu evenimente discrete. Folosirea modelelor care încorporează funcții de timp cu distribuție probabilistică permit obținerea ratelor de producție pentru modelele sistemelor de fabricație, capacitatea de producție, întârzieri, capacitatea pentru comunicare și modelele sistemelor cu microprocesor, utilizarea resurselor critice și măsuri de fiabilizare ale acestora. În ultimii ani, această clasă de rețele Petri a fost folosită extensiv pentru modelarea și studiul performanțelor analitice ale

sistemelor multiprocesor, ale magistralelor sistemelor multiprocesor, ale canalelor de comunicare DSP, ale arhitecturilor paralele de calculatoare, precum și ale algoritmilor paraleli și distribuiți.

Un alt domeniu de aplicare îl constituie rețelele de comunicare. S-a lucrat pe rețele locale cu fibră optică *(Fiber Optics Local Area Networks)* precum Expressnet, Fastnet, D-Net, U-Net, Token Ring. Protocoalele de tip *fieldbuss*, precum FIP și ISA-SP50 au atras foarte multă atenție în ultimii ani, acest lucru fiind oarecum normal, ele fiind rețele importante pentru sistemele industriale complexe. De asemenea, s-a semnalat un interes în creștere pentru modelarea și evaluarea rețelelor de mare viteză *(High Speed Networks)*, cruciale pentru dezvoltarea cu succes a sistemelor multimedia.

Rețelele Petri cu extindere de timp, combinate cu tehnici euristice de căutare, au fost folosite pentru modelarea și studiul problemelor de dispecerizare din sistemele de fabricație și din sistemele cu roboți. De asemenea, acest tip de rețele Petri au fost folosite și pentru modelarea și analiza proceselor chimice continue.

1.2 Descrierea rețelelor Petri

O rețea Petri poate fi identificată cu un tip particular de grafuri orientate bipartite populate cu trei tipuri de obiecte. Aceste obiecte sunt locuri, tranziții și arce orientate care conectează locuri cu tranziții sau tranziții cu locuri. Din punct de vedere grafic, locurile sunt reprezentate prin cercuri iar tranzițiile prin bare sau dreptunghiuri. Un loc este intrare pentru o tranziție dacă există un arc orientat de la acel loc la tranziție. Un loc este ieșire pentru o tranziție dacă există un arc orientat de la tranziție la loc. În forma sa cea mai simplă, o rețea Petri poate fi reprezentată printr-o tranziție împreună cu locurile sale de intrare și de ieșire. Această rețea elementară poate fi folosită pentru reprezentarea unor aspecte diverse ale sistemelor modelate. Spre exemplu, locurile de intrare (ieșire) pot reprezenta precondiții (postcondiții), iar tranzițiile — evenimente. Locurile de intrare pot semnifica disponibilitatea resurselor, tranziția — utilizarea lor, iar locurile de ieșire — eliberarea resurselor. Un exemplu de rețea Petri este prezentată în Fig. 1.1. Această rețea este formată din cinci locuri, reprezentate prin cercuri, patru tranziții, reprezentate prin bare și arce orientate ce conectează locurile cu tranzițiile și tranzițiile cu locurile. În rețea, locul p_1 este intrare pentru tranziția t_1, iar locurile p_2, și p_3 sunt ieșiri pentru tranziția t_1.

Pentru a studia comportamentul dinamic al sistemului modelat, adică stările acestuia și modificările lor, fiecare loc poate deține niciunul sau un număr pozitiv de jetoane, reprezentate grafic prin mici cercuri solide, așa ca în Fig. 1.1. Prezența sau absența unui jeton dintr-un loc poate indica faptul că o condiție asociată cu acel loc este adevărată sau falsă. Pentru un loc care reprezintă disponibilitatea unor resurse, numărul de jetoane în loc indică numărul resurselor disponibile. La un anumit moment dat de timp, distribuția jetoanelor în locuri, denumită marcajul rețelei Petri, definește starea curentă a sistemului modelat. Marcajul unei rețele Petri cu m locuri este reprezentat de un vector M cu dimensiunea (m x 1), ale cărui elemente, notate $M(p)$, sunt numere întregi pozitive reprezentând numărul de jetoane în locurile corespunzătoare. O rețea Petri ce conține jetoane se numește rețea marcată. Spre exemplu, în modelul de rețea Petri din Fig. 1.1, $M = (1,0,0,0,0)^T$.

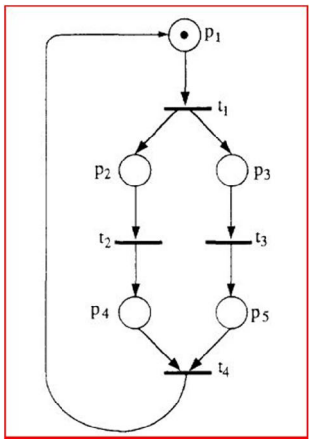

Fig. 1.1 Exemplu de reprezentare grafică a unei rețele Petri

Formal, o rețea Petri poate fi definită astfel:

RP = *(P, T, I,* O, M_0); unde

1) $P = \{p_1, p_2, \dots p_m\}$ este un set finit de locuri,
2) $T = \{ t_1, t_2, \dots t_m \}$ este un set finit de tranziții, $P \cup T \neq \emptyset$, și $P \cap T \neq 0$,
3) $I: (P \times T) \rightarrow N$ este o funcție de intrare care definește arcele orientate de la locuri la tranziții, unde *N* este un set de întregi pozitivi,
4) $O: (P \times T) \rightarrow N$ este o funcție de ieșire care definește arcele orientate de la tranziții la locuri și
5) $M_0: P \rightarrow N$ este marcajul inițial.

Dacă *I(p, t) = k (O(p, t) = k)*, atunci există *k* arce orientate (paralele) conectând locul *p* cu tranziția *t* (tranziția *t* cu locul *p).* Dacă *I(p, t)* = 0 *(O(p, t)* = 0), atunci nu există niciun arc orientat care să conecteze *p* cu *t* (*t* cu *p).* Frecvent, în reprezentarea grafică, arcele paralele care conectează un loc (tranziție) cu o tranziție (loc) sunt reprezentate printr-un singur arc etichetat cu multiplicitatea sa, sau ponderea *k.* Această reprezentare compactă a arcelor multiple este reprezentată în Fig. 1.2.

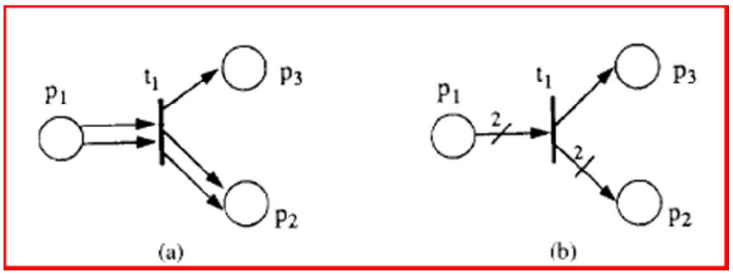

Fig. 1.2 (a) Arce multiple. (b) Reprezentare compactă a arcelor multiple.

Modificând distribuția jetoanelor în locuri, lucru care poate reflecta apariția unor evenimente sau execuția unor operații, se poate studia comportamentul dinamic al sistemului modelat. Următoarele reguli sunt folosite pentru controlul fluxului de jetoane:

Regula de activare. O tranziție *t* se spune că este activată dacă fiecare loc de intrare *p* al lui *t* conține cel puțin numărul de jetoane egal cu ponderea arcelor orientate ce conectează *p* cu *t.*

Regula de execuție:

(a) O tranziție activată *t* poate sau nu să fie executată dependent de interpretarea adițională asociată și

(b) Execuția unei tranziții activate *t* înlătură din fiecare loc de intrare *p* un număr de jetoane egal cu ponderea arcului orientat care conectează *p* cu *t*. De asemenea, depozitează în fiecare loc de ieșire *p* un număr de jetoane egal cu ponderea arcului direcțional care conectează *t* cu *p*.

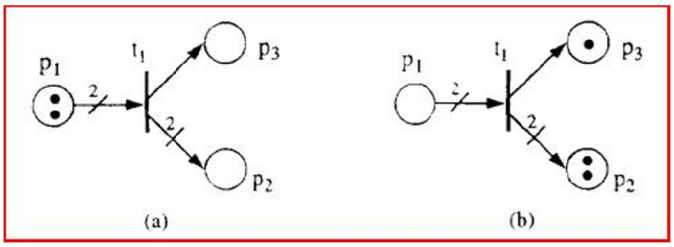

Fig. 1.3 (a) Tranziția t_1 activată. (b) Tranziția activată t_1 este executată

Regulile de activare și de execuție sunt ilustrate în Fig. 1.3. În Fig. 1.3 (a), tranziția t_1 este activată deoarece locul de intrare p_1 al tranziției t_1 conține două jetoane și $I(p_1, t_1) = 2$. Execuția tranziției activate t_1 înlătură din locul de intrare p_1 două jetoane, deoarece $I(p_1, t_1) = 2$, și depozitează un jeton în locul de ieșire p_3, $O(p_3, t_1) = 1$ și două jetoane în locul de ieșire p_2, $O(p_2, t_1) = 2$. Toate acestea sunt reliefate în Fig. *1.3* (b).

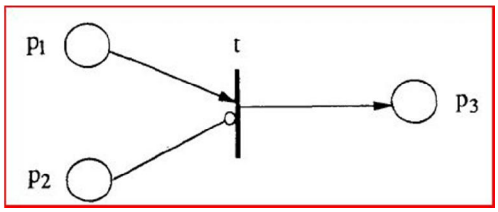

Fig. 1.4 Rețea Petri cu un arc inhibitor

Puterea de modelare a rețelelor Petri poate fi crescută prin adăugarea abilității de testabilitate zero, adică abilitatea de a testa dacă un loc nu are jetoane. Acest lucru este realizat prin introducerea unui arc inhibitor, care conectează un loc de intrare cu o tranziție și este reprezentat grafic cu un arc terminat cu un mic cerc (Fig. 1.4). Prezența unui arc inhibitor între un loc de intrare și o tranziție va schimba condițiile de activare a tranziției. În acest caz, tranziția este activată dacă fiecare loc de intrare, conectat la tranziție printr-un arc normal (terminat cu săgeată), conține cel puțin numărul de jetoane egal cu ponderea arcului și nici un jeton nu este prezent în fiecare loc de intrare conectat la tranziție printr-un arc inhibitor. Regula de execuție a tranziției este aceiași, doar că nu modifică marcajul locurilor conectate prin arc inhibitor.

Se spune că o rețea Petri este pură sau fără bucle dacă nu există un loc care să fie și intrare și ieșire a aceleiași tranziții. O rețea Petri care conține bucle poate fi convertită la o rețea Petri pură, ca în Fig. 1.5.

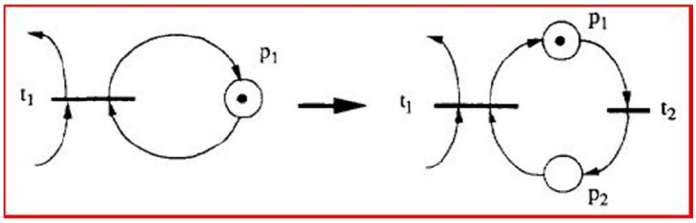

Fig. 1.5 Înlăturarea buclelor

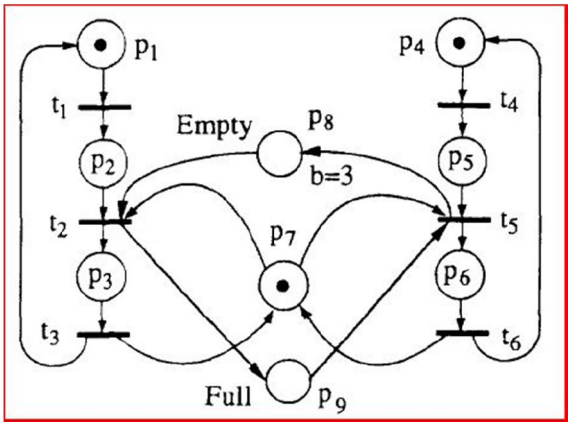

Fig. 1.6 Model de rețea Petri pentru un sistem multirobot

Locuri (cu jetoane)	Interpretare
p_1 (p_4)	Robotul R_1 (R_2) realizează operații în afara spațiului comun
P_2 (p_5)	Robotul R_1 (R_2) așteaptă accesul în spațiului comun
p_3 (p_4)	Robotul R_1 (R_2) realizează operații în spațiului comun
p_7	Excludere mutuală
p_5 (p_9)	Buffer plin (gol)
Tranziții	**Interpretare**
t_1 (t_4)	Robotul R_1 (R_2) cere acces în spațiului comun
t_2 (t_5)	Robotul R_1 (R_2) intră în spațiului comun
t_3 (t_4)	Robotul R_1 (R_2) părăsește spațiului comun

Tabel 1 Interpretarea locurilor și tranzițiilor pentru modelul de rețea Petri al sistemului de asamblare multirobot

Pentru a ilustra modul în care rețelele Petri pot fi folosite pentru a modela proprietăți precum activități concurente, sincronizare, excludere mutuală etc., se consideră un exemplu simplu al unui sistem cu roboți. Sistemul este reprezentat de rețeaua Petri din Fig. 1.6, cu detalii în Tabelul 1. În acest model, două brațe robotizate realizează operații de preluare și plasare, accesând un spațiu comun la preluarea sau la transferul pieselor. Pentru a evita coliziunile, se presupune că doar un robot poate accesa spațiul de lucru la un anumit moment de timp. În plus, se consideră că spațiul de lucru conține un buffer cu spațiu

limitat pentru produse. Exemplul poate reprezenta operarea a două brațe robotizare într-un sistem cu două mașini unelte, unde un braț transferă semiprodusele de la prima mașină în buffer, iar celalalt braț transferă semiprodusele de la buffer la a doua mașină.

În acest model, locurile p_1, p_2, p_3 și tranzițiile t_1, t_2, t_3 modelează activitățile brațului robotizat *R1*. Locurile p_4, p_5, p_6 și tranzițiile t_4, t_5, t_6 modelează activitățile brațului robotizat *R2.* Tranzițiile t_1 și t_4 reprezintă activități concurente ale lui *R1* și *R2.* Fiecare dintre aceste tranziții poate fi trasă înainte sau după sau în paralele cu cealaltă. Accesul la spațiul comun necesită sincronizarea activităților brațelor pentru a evita coliziunea. Doar un braț robotizat poate accesa spațiul comun de lucru la un anumit moment de timp. Această sincronizare este realizată prin mecanismul de excludere mutuală implementat de subrețeaua formată din locurile p_7, p_3, p_6 și tranzițiile t_2, t_3, t_5, t_6. Execuția tranziției t_2 dezactivează t_5, presupunând că t_5 este activată, și viceversa. Astfel, doar un braț robotizat poate accesa spațiul comun la un moment dat. În plus, se consideră capacitatea bufferului ca fiind b. În acest fel, spre exemplu, dacă p_8 este gol, atunci t_2 nu poate fi activată. Această împiedică brațul *R1* să încerce transferul unei piese către buffer când acesta este plin. De asemenea, *R2* nu poate accesa bufferul dacă nu sunt piese în el, adică locul p_9 este gol.

1.3 Proprietăți ale rețelelor Petri

Ca instrument matematic, rețelele Petri posedă un număr de proprietăți. Aceste proprietăți, atunci când sunt interpretate în contextul sistemului modelat, permit designer-ului de sistem să identifice prezența sau absența proprietăților funcționale specifice domeniului de aplicare al sistemului proiectat. Astfel, pot fi distinse două tipuri de proprietăți: comportamentale și structurale. Proprietățile comportamentale sunt acelea care depind de starea inițială, sau marcajul, unei rețele Petri și de topologia acesteia. Vor fi discutate în cele ce urmează proprietăți precum accesibilitatea, limitabilitatea, conservativitatea, nivelul de activare *(liveness)*, reversibilitatea și starea de pornire *(home state)*.

1.3.1 Accesibilitate

O problemă importantă în designul sistemelor distribuite constă în capacitatea sistemului de a atinge o anumită stare sau de a prezenta un comportament funcțional particular. În general, întrebarea la care se caută răspuns este dacă sistemul modelat cu rețele Petri are toate proprietățile dorite, așa cum sunt ele specificate în cerințe, și nicio proprietate nedorită.

Pentru a afla dacă sistemul modelat poate atinge o anumită stare ca rezultat al comportamentului funcțional cerut, este necesară găsirea unei astfel de secvențe de execuții a tranzițiilor care va avea ca efect transformarea marcajului M_0 în M_i, unde M_i reprezintă starea specifică și secvența de execuții reprezintă comportamentul funcțional cerut. Trebuie subliniat faptul că sistemele reale pot atinge o anumită stare prin mai multe comportamente funcționale permise. Într-o rețea Petri, acest lucru este reflectat de existența unor secvențe specifice de execuții de tranziții, reprezentând comportamentul funcțional cerut ,care vor transforma un marcaj M_0 în marcajul M_i cerut. Dacă într-un model de rețea Petri există secvențe adiționale de execuție a tranzițiilor care să transforme un marcaj M_0 în marcajul M_i poate indi-

ca faptul că acel model de rețea Petri nu reflectă cu exactitate structura și dinamica sistemului descris. De asemenea, acest fapt poate indica și prezența unor aspecte neanticipate privind comportamentul funcțional al sistemului real, ceea ce înseamnă că rețeaua Petri reflectă cu precizie specificațiile sistemului descris.

Un marcaj M_i este accesibil pornind de la marcajul M_0 dacă există e secvență de execuție a tranzițiilor care transformă marcajul M_0 în M_i. Un marcaj M_1 este imediat accesibil după marcajul M_0 dacă o execuție a tranzițiilor activate din M_0 determină obținerea marcajului M_1. Spre exemplu, în modelul sistemului cu multirobot din Fig. 1.6, starea în care brațul robotic *R1* realizează sarcini în spațiul comun, în timp ce brațul robotic *R2* așteaptă în afara spațiului, este reprezentată de vectorul $M_i = (0, 0, 1, 0, 1, 0, 0, 2, 1)^T$. M_i este accesibil din marcajul inițial M_0, unde $M_0 = (1, 0, 0, 1, 0, 0, 1, 3, 0)^T$, prin următoarea secvență de execuție a tranzițiilor: $t_1\ t_2\ t_4$. Marcajul $M_i = (0, 1, 0, 0, 1, 0, 1, 3, 0)^T$, care reprezintă starea sistemului în care brațul robotic *R1* așteaptă accesul în spațiul comun și brațul robotic *R2* realizează sarcini în afara spațiului comun, este accesibil imediat din marcajul inițial M_0 prin execuția tranziției t_1. În M_0, atât tranziția t_1, cât și tranziția t_4, sunt activate. Setul tuturor marcajelor accesibile din M_0 este denumit setul de accesibilitate și este notat $R(M_0)$. Setul tuturor secvențelor posibile de execuție din M_0 este notat cu $L(M_0)$. În acest fel, problema identificării existenței unei stări specifice M_i în care sistemul poate să ajungă poate fi redefinită ca problema găsirii lui $M_i \in R(M_0)$.

1.3.2 Limitabilitate și siguranță

De regulă, locurile sunt folosite pentru reprezentarea unei zone de păstrare a datelor în comunicare sau sistemele computerizate, a unor produse sau zone de păstrare a uneltelor în sistemele de producție etc. Este foarte important de stabilit dacă strategiile de control stabilite asigură evitarea stării de supraîncărcare a acestor zone. Zonele de păstrare a datelor pot păstra, fără a le corupe, doar un număr restricționat de părți de date. În sistemele de fabricație, încercarea de a înmagazina mai multe unelte în zona destinată acestui lucru poate duce la defectarea echipamentului. Proprietatea unei rețele Petri care permite identificarea în cadrul sistemului modelat a situației de supraîncărcare se numește limitabilitate. O rețea Petri este *k*-limitată dacă numărul de jetoane în orice loc *p*, unde $p \in P$, este totdeauna mai mic sau egal cu *k* (*k* este un număr întreg pozitiv), pentru orice marcaj *M* accesibil din marcajul inițial M_0, $M \in R(M_0)$. O rețea Petri este sigură dacă este 1-limitată (Fig. 1.7). Într-o astfel de rețea, niciun loc nu poate conține mai mult de un jeton. O rețea Petri este nelimitată dacă există cel puțin un loc care să conțină un număr oricât de mare de jetoane (Fig. 1.8, locul p_4).

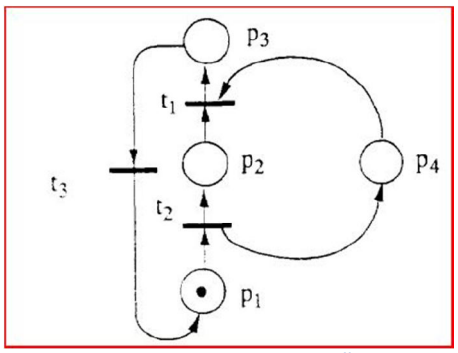

Fig. 1.7 Rețea Petri sigură

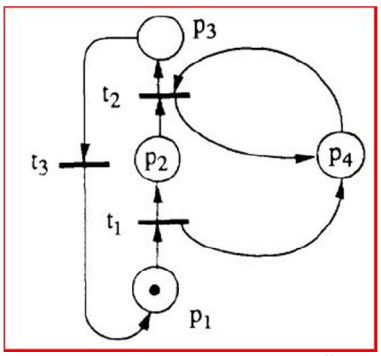

Fig. 1.8 Rețea Petri nelimitată

1.3.3 Conservativitate

În sistemele reale, numărul resurselor utilizate este, în mod normal, limitat prin constrângeri financiare sau de alt gen. Dacă jetoanele sunt utilizate pentru reprezentarea resurselor, al căror număr într-un sistem este fix, atunci numărul jetoanelor din rețeaua Petri a respectivului sistem ar trebui să rămână neschimbat indiferent de marcajul curent al sistemului. Această directivă descinde din faptul că resursele nu pot fi nici create, nici distruse, cu excepția cazului când ar trebui să se întâmple așa. Spre exemplu, o unealtă distrusă poate fi înlăturată din celula de fabricare, iar numărul uneltelor disponibile se va reduce.

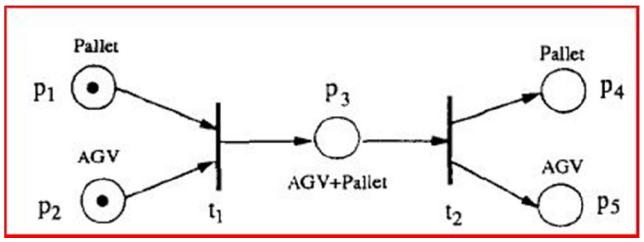

Fig. 1.9 Rețea Petri conservativă în raport cu w = [1,1,2,1,1]

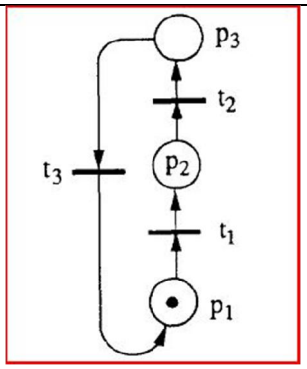

Fig. 1.10 Rețea Petri strict conservativă

O rețea Petri este conservativă dacă numărul de jetoane este conservat. Din punct de vedere structural, acest lucru este posibil doar dacă numărul de arce de intrare în fiecare tranziție este egal cu numărul de arce de ieșire. În sistemele reale, însă, resursele sunt, în mod frecvent, combinate astfel încât anumite sarcini să fie executate, apoi separate după finalizarea sarcinilor. Spre exemplu, într-un sistem de fabricație flexibilă un vehicul ghidat automat colectează paleții cu produse de la o celulă de prelucrare și îi transportă la o stație de descărcare unde paleții sunt preluați (scenariu ilustrat în Fig. 1.9). Tranziția t_1 modelează încărcarea unui palet într-un vehicul. Tranziția t_2 reprezintă livrarea paletului la stația de descărcare și înlăturarea acestuia de pe vehicul. Deși numărul de jetoane se schimbă de la două la unu atunci când este executată t_1 și înapoi la două când este executată t_2, numărul resurselor din sistem nu se modifică. Pentru a preîntâmpina această problemă, locurilor li se pot asocia ponderi pentru ca suma ponderată a jetoanelor din rețea să rămână constantă.

Se spune că o rețea Petri este conservativă dacă există un vector *w*, $w = [w_1, w_2, ..., w_m]$, unde *m* este numărul de locuri și $w(p) > 0$ pentru fiecare $p \in P$, astfel încât suma ponderată a jetoanelor rămâne neschimbată pentru fiecare marcaj *M* care poate fi accesat din marcajul inițial M_o. O rețea Petri este strict conservativă dacă toate intrările vectorului *w* sunt unitare. Rețeaua Petri din Fig. 1.9 este conservativă în raport cu vectorul *w* = [1,1,2,1,1] deoarece suma ponderată a jetoanelor în fiecare marcaj este doi. Un exemplu de rețea Petri care nu este conservativă este prezentat în Fig. 1.8 deoarece locul p_4 poate deține un număr nelimitat de jetoane. Dacă o rețea Petri este conservativă în raport cu un vector unitar, atunci rețeaua este strict conservativă (Fig. 1.10).

1.3.4 Nivelul de activare

Acest concept este strâns corelat cu situația de blocare *(deadlock)*, care a fost studiată în mod extensiv în contextul sistemelor de operare. S-a arătat că această situație poate să apară în patru condiții:

1) Excluderea mutuală: o resursă este fie disponibilă, fie alocată unui proces care are acces exclusiv asupra ei.
2) Deține și așteaptă: unui proces i se permite să dețină o resursă (sau mai multe) și să acceseze încă o resursă (sau mai multe).
3) Fără preempțiune: o resursă (sau mai multe) alocată unui proces nu poate fi eliberată decât de către procesul în sine.
4) Așteptare circulară: două sau mai multe procese sunt aranjate într-un lanț în care fiecare proces așteaptă după resursele deținute de procesul poziționat înaintea lui în lanț.

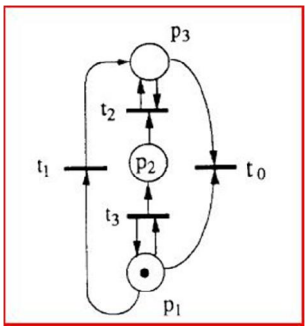

Fig. 1.11 Rețea Petri cu diferite niveluri de activare a tranzițiilor

Spre exemplu, într-un sistem flexibil de fabricație poate interveni o situație de blocare atunci când un buffer de intrare/ieșire al unei unelte de prelucrare deține un palet cu produse prelucrate și alt palet cu produse care trebuie prelucrate a fost trimis la buffer. Dacă buffer-ul poate păstra doar un palet la un moment de timp și vehiculul ghidat automat, care transportă paleții, are loc doar pentru un palet, atunci tocmai a intervenit o situație de blocare. Paletul cu piese prelucrate nu poate fi mutat din buffer în mașină, iar paletul cu piese neprelucrate nu poate fi mutat din mașină în buffer. În acest exemplu, toate cele patru condiții de mai sus au fost îndeplinite, dacă spațiile pentru paleți din buffer și de pe mașină sunt privite ca resurse. Dacă în software-ul de control nu există prevăzută o rutină pentru detectarea și ieșirea din starea de blocare, o astfel de stare, deși apărută într-un subsistem, se poate propaga și poate afecta o mare parte a sistemului.

O rețea Petri care modelează un sistem fără stări de blocare este o rețea activă. Aceasta înseamnă că pentru toate marcajele M, care pot fi accesate din marcajul inițial M_o, este posibilă execuția oricărei tranziții din rețea prin progresul obținut de parcurgerea a câteva secvențe de execuții. Rețeaua Petri din Fig. 1.10 este activă. Cu toate acestea, scenariul de mai sus poate fi prea strict pentru a reprezenta anumite sisteme reale care prezintă comportament fără blocaje. Spre exemplu, inițializarea unui sistem poate fi modelată de o tranziție (sau mai multe) care se execută de un număr finit de ori. După inițializare, sistemul poate avea un comportament lipsit de blocaje, deși rețeaua Petri reprezentând acest sistem

nu mai este activă, conform cu definiția anterioară. Din acest motiv, există mai multe niveluri de activare pentru o tranziție *t* și marcajul M_o. Astfel, o tranziție *t* într-o rețea Petri poate fi:

- L0-activă (sau moartă) dacă nu există nicio secvență de execuție din $L(M_o)$ în care *t* să fie executată,
- L1-activă (potențial executabilă) dacă *t* poate fi executată cel puțin o dată în anumite secvențe de execuție din $L(M_o)$,
- L2-activă dacă *t* poate fi executată de cel puțin *k* ori în anumite secvențe de execuție din $L(M_o)$ pentru orice *k* întreg și pozitiv,
- L3-activă dacă *t* poate fi executată de un număr infinit de ori în anumite secvențe de execuție din $L(M_o)$, şi
- L4-activă (sau vie) dacă *t* este L1-activă (potențial executabilă) în orice marcaj din $R(M_o)$.

Urmând această clasificare, o reţea Petri se spune că este *Li*-activă, pentru marcajul M_o, dacă orice tranziție din rețea este Li-activă. În Fig. 1.11 sunt prezentate diverse niveluri de activare. Astfel, tranzițiile t_0, t_1, t_2 și t_3 sunt L0, L1, L2, și, respectiv, L3-active.

1.3.5 Reversibilitate şi starea de pornire

O problemă importantă în sistemele de operare reale, precum sistemele de fabricație, sistemele de control al proceselor etc., constă în abilitatea acestor sisteme de a-și reveni dintr-a situație de eroare. Aceste sisteme trebuie să se întoarcă din starea care a eșuat la stări corecte anterioare. Această cerință este strâns legată de proprietățile unei rețele Petri denumite reversibilitate și stare de pornire. Pentru marcajul inițial M_o, o rețea Petri este reversibilă dacă pentru orice marcaj *M* din $R(M_o)$, M_o este accesibil din *M*. Starea de pornire este o proprietate mai puțin restrictivă, și mult mai practică din acest motiv, decât proprietatea de reversibilitate a unei rețele Petri. O stare *M* a unei rețele Petri este stare de pornire dacă pentru orice marcaj *M* din $R(M_o)$, M_i este accesibilă din *M*. Rețeaua Petri din Fig. 1.7 este reversibilă, iar rețeaua din Fig. 1.8 este nereversibilă.

1.4 Metode de analiză

În paragraful anterior au fost definite câteva proprietăți ale rețelelor Petri care sunt folositoare pentru analiza sistemelor modelate. Un aspect important care trebuie avut în vedere în timpul analizei constă în verificarea existenței unei corespondențe funcționale de unu-la-unu între modelul rețelei Petri și specificațiile originale, de regulă, exprimate într-un mod informal. Conceperea unor modele de rețele Petri din specificații informale nu este o sarcină ușoară, ea necesitând o mare experiență în modelare, precum și cunoașterea tehnicilor de asistență în construirea modelului. Ca urmare, un model poate diferi foarte mult de specificațiile inițiale, lucru general valabil când este vorba despre rețele Petri mari ale unor sisteme complexe. Existența unei corespondențe funcționale de unu-la-unu între specificațiile inițiale și reprezentarea în rețea Petri a sistemului permite proiectarea rezultatelor analizei, obținute pe model, asupra descrierii inițiale. Acest lucru permite obținerea unui feedback pentru clienți pe baza căruia, în multe situații, clienții își clarifică propria percepție despre sistem.

Un alt aspect important care trebuie urmărit în timpul analizei este respectarea în totalitate a specificațiilor. De cele mai multe ori, aceste specificații definesc comportamentul funcțional extern al sistemului, exprimat uzual prin relaționările de tip intrare/ieșire. Intrările sunt generate de mediul înconjurător sistemului, iar ieșirile reprezintă răspunsurile sistemului la aceste intrări. Dacă anumite intrări, generate de către mediu asupra sistemului, nu sunt incluse în specificații, atunci sistemul nu va fi capabil să răspundă la aceste intrări în mod corespunzător atunci când ele vor apărea pe parcursul operării normale. Necesitatea ca specificațiile să fie complete este cu atât mai importantă cu cât sistemul este mai critic, când specificații incomplete pot duce la apariția unor evenimente catastrofice. Spre exemplu, apariția unei stări neanticipate în operarea unui reactor nuclear poate duce la imposibilitatea rezolvării ei de către sistemul de control, fapt deosebit de grav pentru siguranța întregii zone.

Consistența specificațiilor inițiale este o altă problemă care trebuie luată în considerație în timpul analizei. Inconsistențe apar atunci când pentru o combinație permisă de intrări specificațiile permit două sau mai multe combinații permise ale ieșirilor. Acest lucru se datorează, în general, unei percepții vagi, incomplete și deseori incorecte a funcționalității sistemului. În continuare vor fi prezentate două metode fundamentale de analiză. Una se bazează pe arborele de accesibilitate și cealaltă pe reprezentarea matriceală a rețelei. Pe lângă aceste metode mai există și altele, destinate analizei unei rețele Petri, care permit o transformare sistematică a rețelei prin reducerea numărului de locuri și tranziții, dar cu păstrarea unor proprietăți precum limitabilitatea, conservabilitatea, nivelul de activare etc., pe principiul că rețelele mai mici sunt mult mai ușor de utilizat.

1.4.1 Arborele de acoperire

Această metodă se bazează pe enumerarea tuturor marcajelor posibile accesibile din marcajul inițial M_o. Pornind de la marcajul inițial M_o, se poate construi setul de accesibilitate prin execuția tuturor tranzițiilor posibile activate în toate marcajele posibile accesibile din marcajul inițial. În arborele de acoperire, fiecare nod este etichetat cu un marcaj și fiecare arc cu tranzițiile necesare. Nodul rădăcină este etichetat cu marcajul inițial M_o. Setul de accesibilitate devine nelimitat din două motive: existența unor marcaje duplicate sau rețeaua în sine este nelimitată. Pentru a preveni un arbore de acoperire să devină foarte mare, trebuie parcurși doi pași pe parcursul construirii arborelui. Primul pas presupune eliminarea marcajelor duplicate: dacă pe calea dintre un marcaj inițial M_o și un marcaj curent M apare un marcaj M' identic cu marcajul M, atunci marcajul M, fiind duplicat, devine nod terminal. Apariția unui marcaj terminal implică faptul că toate marcajele posibile a fi accesate din M au fost deja adăugate arborelui.

În ceea ce privește rețelele nelimitate, pentru a păstra arborele finit, s-a introdus simbolul ω, care este interpretat drept infinit. Astfel, pentru un întreg n, $\omega + n = \omega$, $\omega - n = \omega$, $n < \omega$. În acest caz, dacă pe calea de la marcajul inițial M_o către un marcaj curent M apare un marcaj M' ale cărui intrări sunt mai mici sau egale cu intrările corespondente din marcajul M, atunci intrările lui M, care sunt strict mai mari decât intrările corespondente din M', trebuie înlocuite cu simbolul ω. În anumite căi, existența marcajelor cu intrările corespondente egale sau mai mari (pe măsură ce se depărtează de nodul rădăcină) indică faptul că secvențele de execuție care transformă M' la M pot fi repetate la infinit. De fiecare dată când această secvență este repetată, numărul de jetoane din locurile etichetate cu simbolul ω va crește.

Arborele de acoperire este construit conform cu următorul algoritm:

1) Marcajul inițial M_o este rădăcina arborelui și se etichetează cu „nou”.
2) Atât timp cât există marcaje „noi”realizează următoarele:
3) Selectează un „nou” marcaj M
   a. Dacă M este identic cu alt marcaj din arbore, se etichetează M ca „vechi” și se trece la un alt marcaj „nou”.
   b. Dacă nicio tranziție nu se activează în M, se etichetează M ca „terminal”.
4) Pentru orice tranziție t activată în marcajul M realizează:
   a. Obține marcajul M' care rezultă din execuția lui t în M.
   b. Dacă pe calea de la rădăcină la M există un marcaj M'' astfel încât $M'(p) \geq M''(p)$ pentru fiecare loc p, și $M' = M''$, atunci înlocuiește $M'(p)$ cu ω pentru fiecare p, atâta vreme cât $M'(p) > M''(p)$.
   c. Introduce M' ca nod, trasează un arc de la M la M' etichetat t și etichetează M' ca „nou”.

Următorul exemplu va ilustra această metodă. Se consideră rețeaua din Fig. 1.12 și arborele ei de acoperire din Fig. 1.13. Dat fiind marcajul inițial, nodul rădăcină este $M_o = (1,0,1,0)^T$. În acest marcaj, tranziția t_3 este activată.

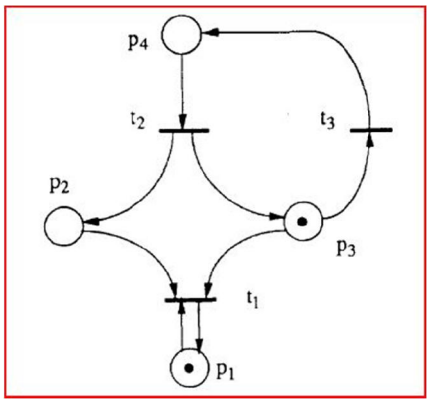

Fig. 1.12 Model de rețea Petri

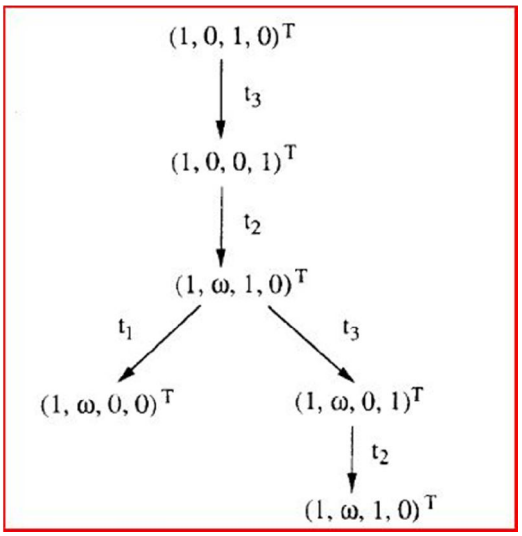

Fig. 1.13 Arborele de acoperire pentru modelul de rețea Petri din Fig. 1.12

Când t_3 este executată, se obține un nou marcaj: $M_1 = (1,0,0,1)^T$. Aceste este un marcaj „nou” în care tranziția t_2 este activată. Execuția lui t_2 din M, va determina obținerea lui $M_2 = (1,1,1,0)^T$. Deoarece $M_2 = (1,1,1,0)^T \geq M_o = (1,0,1,0)^T$, a doua componentă trebuie înlocuită cu simbolul ω. Aceasta reflectă faptul că execuția secvenței t_3t_2 poate fi repetată de un număr arbitrar de ori. În marcajul $M_2 = (1,\omega,1,0)^T$ sunt activate două tranziții: tranziția t_1 și tranziția t_3 . Execuția lui t_1 va determina marcajul $M_3 = (1,\omega,0,0)^T$, care este un nod „terminal”. Execuția tranziției t_3 va determina un marcaj „nou” $M_4 = (1,\omega,0,1)^T$, care activează tranziția t_2. Execuția lui t_2 din M_4 va duce la un nod „vechi”: $M_5 = (1,\omega,1,0)^T$ care este identic cu M_2.

Folosind arborele de acoperire se pot studia câteva proprietăți ale rețelei Petri. Spre exemplu, dacă un nod din arbore conține simbolul ω, atunci rețeaua este nelimitată, dat fiind faptul că simbolul ω poate deveni oricât de mare. Altfel (adică nu apare simbolul ω), rețeaua este legată. Dacă fiecare nod al arborelui conține doar valori de 0 și 1, atunci rețeaua este sigură. O tranziție este moartă dacă nu apare ca o etichetă de arc în arbore. Dacă un marcaj M este accesibil din marcajul M_o, atunci există un nod M' astfel încât $M \leq M'$. Cu toate acestea, deoarece simbolul ω poate fi oricât de mare, anumite probleme, precum acoperirea sau nivelul de activare, nu pot fi rezolvate doar prin studiul arborelui de acoperire. Pentru o rețea Petri limitată, arborele de acoperire conține, ca noduri, toate marcajele posibile accesibile din marcajul inițial M_o. În acest caz, arborele de acoperire se numește arbore de accesibilitate și orice problemă de analiză poate fi rezolvată prin simpla lui inspecție.

1.4.2 Matricea de incidență și ecuația de stare

O metodă alternativă de reprezentare și analiză a rețelelor Petri se bazează pe ecuații matriciale, folosite pentru reprezentarea comportamentului dinamic al rețelelor Petri. Metoda presupune construirea matricei de incidență care definește toate interconexiunile posibile dintre locuri și tranziții. Matricea de incidență a unei rețele Petri pure este o matrice A cu dimensiunile n x m întregi, unde n este numărul de tranziții și m este numărul de locuri. Intrările în matricea de incidență sunt definite astfel:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

unde: a_{ij}^{+} este egal cu numărul de arce care conectează tranziția t_i cu locurile sale de ieșire p_j

$$(a_{ij}^{+} = O(p_j, t_i))$$

a_{ij}^{-} este egal cu numărul de arce care conectează tranziția t_i cu locurile sale de intrare p_j

$$(a_{ij}^{-} = I(p_j, t_i)).$$

Când tranziția t_i se execută, a_{ij}^{+} reprezintă numărul de jetoane depozitate în locurile sale de ieșire p_j, a_{ij}^{-} reprezintă numărul de jetoane înlăturate din locurile sale de intrare p_j, iar a_{ij} reprezintă modificarea numărului de jetoane în locul p_j. Astfel, se spune că tranziția t_i este activată în marcajul M dacă

$$a_{ij}^{-} \leq M(p_j),\ i = 1, 2, \ldots m$$

Pentru rețelele Petri cu bucle, $a_{ij} = 0$ pentru un loc p_j și o tranziție t_i care aparțin unei bucle. Din acest motiv, pentru a avea siguranța că matricea de incidență reflectă structura rețelei Petri, se presupune că rețeaua este pură sau este făcută pură prin introducerea de locuri adiționale (Fig. 1.5). Ecuația de stare pentru o rețea Petri reprezintă modificarea din distribuția jetoanelor pe locuri (marcaje) ca rezultat al executării tranzițiilor. Această ecuație este definită astfel:

$$M_k = M_{k-1} + A^T u_k, k = 1,2, \ldots$$

M_k este un vector coloană cu dimensiunea m x 1 reprezentând un marcaj M_k accesibil imediat din marcajul M_{k-1} după execuția tranziției t_i. Vectorul k de execuție, u_k, este un vector coloană de dimensiune n x 1, care are toate intrările diferite de zero. Valoarea 1 în poziția i reprezintă execuția tranziției t_i în execuția k a secvenței de execuții a rețelei, pornind de la marcajul inițial M_o. Această intrare corespunde cu linia i

a matricei de incidență *A*, care reprezintă o modificare în marcaj ca urmare a execuției tranziției t_i. Ecuația matriceală este folositoare în studiul problemei de accesibilitate.

Două concepte asociate cu matricea de incidență sunt folositoare în studiul proprietăților modelelor cu rețele Petri: invariantul *T* și invariantul *P.*

O soluție întreagă x a ecuației $A^T x = 0$ este denumită invariant *T*. Intrările diferite de zero într-un invariant *T* reprezintă numărul execuțiilor tranzițiilor corespunzătoare care aparțin unei secvențe de execuție ce transformă marcajul M_0 până se ajunge din nou în M_0. Deși un invariant *T* conține tranzițiile cuprinse în secvența de execuții care transformă marcajul M_0 în M_0, precum și numărul de ori în care aceste tranziții apar în secvență, nu specifică ordinea de execuție a tranzițiilor.

O soluție întreagă y a ecuației $Ay = 0$ este denumită invariant *P*. Acesta poate fi explicat, în mod intuitiv, astfel: intrările deferite de zero reprezintă ponderile asociate locurilor corespunzătoare astfel încât suma ponderată a jetoanelor din aceste locuri să fie constantă pentru toate marcajele accesibile din marcajul inițial.

Subsetul de locuri (tranziții) care corespund intrărilor diferite de zero din invariantul *P* (invariantul *T*) este denumit suport pentru invariant și este notat $\|x\|(\|y\|)$.

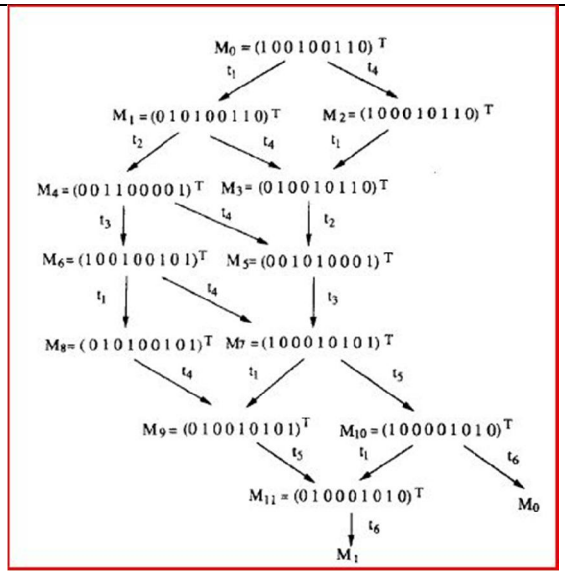

Fig. 1.14 Arborele de acoperire al rețelei Petri din Fig. 1.6

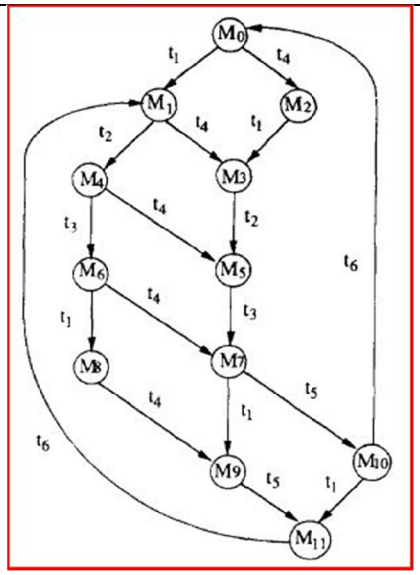

Fig. 1.15 Graful de accesibilitate al rețelei Petri din Fig. 1.6

1.4.3 Un exemplu

În această secțiune se va demonstra modul în care arborele de acoperire și tehnicile bazate pe invarianți pot fi folosite pentru a analiza un model de rețea Petri, pe baza exemplului din Fig. 1.6 privind un sistem multirobot. Presupunând că *b* = 1, arborele de acoperire, în acest caz un arbore de accesibilitate, este prezentat în Fig. 1.14, iar matricea de incidență în Fig. 1.15.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
t_1	−1	1	0	0	0	0	0	0	0
t_2	0	−1	1	0	0	0	−1	−1	1
t_3	1	0	−1	0	0	0	1	0	0
t_4	0	0	0	−1	1	0	0	0	0
t_5	0	0	0	0	−1	1	−1	1	−1
t_6	0	0	0	1	0	−1	1	0	0

Fig. 1.16 Matricea de incidență a rețelei Petri pentru celula de asamblare cu multirobot

Invarianții *P* obținuți pentru această rețea sunt:

$$y_1 = (1\,1\,1\,0\,0\,0\,0\,0\,0)^T$$
$$y_2 = (0\,0\,0\,1\,1\,1\,0\,0\,0)^T$$
$$y_3 = (0\,0\,1\,0\,0\;\;1\,1\,0\,0)^T$$
$$y_4 = (\,0\,0\,0\,0\,0\,0\,0\,1\,1)^T$$

Următorii sunt invarianții suport corespunzători:

$$\|y_1\| = \{p_1, p_2, p_3\}$$
$$\|y_2\| = \{p_4, p_5, p_6\}$$
$$\|y_3\| = \{p_3, p_6, p_7\}$$
$$\|y_4\| = \{p_8, p_9\}$$

Limitabilitate și siguranță: Rețeaua Petri din Fig. 1.6 este limitată. Acest lucru este evident din arborele de accesibilitate: nici un marcaj accesibil din marcajul inițial M_0 nu conține simbolul ω. În plus, deoarece, pentru toate marcajele, nicio intrare nu este mai mare decât unu, rețeaua este sigură. Aceste proprietăți pot fi foarte ușor determinate folosind invarianții *P*. Deoarece fiecare loc din rețea aparține unui invariant suport iar rețeaua pornește de la un marcaj inițial limitat, întreaga rețea este limitată. În plus, deoarece numărul de jetoane din fiecare invariant suport din marcajul inițial este unu, rețeaua este sigură. Din proprietatea de limitabilitate a modelului de rețea Petri pot fi deduse două proprietăți privind operarea sistemului real. Buffer-ul nu poate fi supraîncărcat, deci nu există nicio probabilitate că *R1* va accesa zona buffer-ului când acesta este plin. De asemenea, buffer-ul nu poate fi supradescărcat, deci nu există nicio probabilitate că *R2* va accesa zona buffer-ului când acesta este gol. Folosind arborele de accesibilitate, aceste proprietăți decurg din siguranța rețelei. Intrările în fiecare marcaj, care reprezintă numărul de jetoane în locurile p_8 și p_9 sunt fie zero, fie unu. Folosind invarianți: invariantul suport $\|y_4\|$ conține locurile p_8 și p_9. Din moment ce conținutul de jetoane în $\|y_4\|$ în marcajul inițial este unu,la un moment dat va fi doar un jeton fie în p_8, fie în p_9. Din acest motiv nu va fi nici supraîncărcare, nici subdescărcare a buffer-ului.

Conservativitatea: Rețeaua Petri din Fig. 1.6 este conservativă. Din graficul de accesibilitate, se poate observa că rețeaua este conservativă raportat la vectorul ω = [1,1,2,1,1,2,1,1,1]. Suma ponderată a je-

toanelor rămâne aceeași pentru fiecare marcaj accesibil din marcajul inițial și este egală cu patru. Folosind invarianți: conținutul de jetoane în fiecare invariant suport din marcajul inițial este unu. Invarianții suport $\|y_1\|$, $\|y_2\|$, și $\|y_4\|$ sunt mutual exclusivi. Invarianții suport $\|y_1\|$ și $\|y_3\|$ conțin locul p_3 ca element comun. Invarianții suport $\|y_2\|$ și $\|y_3\|$ conțin locul p_6 ca element comun. Astfel, ponderea locurilor p_3 și p_6 ar trebui să fie doi pentru ca rețeaua să fie conservativă. Implicația acestei proprietăți este că numărul de brațe robotizate care operează în sistemul de asamblare este doi și nu se poate modifica. De asemenea, spațiul din buffer este unu și nu se poate modifica

Nivel de activare: Rețeaua Petri din Fig. 1.6 este vie, toate tranzițiile fiind activate. Fig. 1.15 prezintă un graf de accesibilitate al rețelei Petri din Fig. 1.6. Acesta este un graf direcționat, format dintr-un set de noduri și un set de arce direcționate. Setul de noduri prezintă toate nodurile distincte din arborele de accesibilitate, iar setul de arce direcționate, unde fiecare arc este etichetat cu câte o tranziție, reprezintă toate tranzițiile posibile între nodurile din arborele de accesibilitate.

La simpla inspecție, rețeaua este *L4*-activă, deoarece pentru orice marcaj accesibil din marcajul M_0, este posibilă execuția oricărei tranziții prin parcurgerea unei anumite secvențe de execuție. Invarianții pot fi folosiți pentru a demonstra „manual" că rețeaua este vie. Cu toate acestea, pentru o rețea de această dimensiune, acest lucru ar fi laborios. Deoarece rețeaua este vie, sistemul nu poate ajunge într-o stare în care nicio operație să nu fie posibilă.

Reversibilitatea: Rețeaua Petri din Fig. 1.6 este reversibilă. Tot prin inspecție, se poate constata, pe baza grafului de accesibilitate, că M_0 este accesibil din orice marcaj $M \in R(Mo)$.

1.5 Rețele Petri: concluzii

Dezvoltarea rețelelor Petri a fost motivată de nevoia de a modela sistemele industriale. Rețelele Petri ordinare nu sunt totdeauna suficiente pentru reprezentarea și analiza sistemelor industriale complexe, fapt care a dus la dezvoltarea unor noi clase de rețele. În rețele ordinare, jetoanele nu au identitate, fapt care ridică probleme în modelarea unor sisteme precum cele de comunicare sau de fabricație, ce necesită resurse fizice sau mesaje cu identitate (dacă sunt reprezentate prin jetoane). Fără această identitate este imposibil de urmărit cursul diferitelor resurse sau mesaje în sistem. O soluție potențială constă în construcția unui model într-o așa manieră încât fluxul fiecărui mesaj să fie asociat cu o subrețea dedicată. Cum resursele și mesajele partajează, în multe cazuri, același sistem, toate aceste subrețele sunt identice, doar că această metodă crește complexitatea grafică a modelului. Pentru a rezolva aceste probleme au fost propuse rețele Petri care permit jetoanelor să aibă identitate distinctă. Aceste rețele, denumite rețele Petri de nivel ridicat, includ rețele cu tranziția predicatelor, rețele colorate și rețele cu jetoane individuale.

În rețelele Petri de nivel ridicat, un jeton poate fi un obiect compus care transportă date. Aceste date pot avea complexitate arbitrară, implicând numere întregi, numere reale, șiruri de caractere, înregistrări, liste etc. Toate tipurile de rețele Petri au aceeași putere descriptivă, dar rețelele de nivel ridicat asigură facilități de structurare mult mai bune. Rețelele colorate și cele cu tranziția predicatelor sunt aproape

identice în ceea ce privește descrierea și simularea. Cu toate acestea, există diferențe considerabile în ceea ce privește analiza formală. Rețelele Petri colorate sunt utilizate în diverse domenii, incluzând protocoale de comunicare, sisteme de producție etc. O dezvoltare importantă în domeniul rețelelor Petri de nivel ridicat constă în introducerea modelelor orientate pe obiecte. În această clasă de rețele, jetoanele sunt considerate instanțe sau tuplete ale instanțelor claselor de obiecte definite ca liste de atribute. Acest tip de rețea a fost utilizat pentru a modela și analiza sisteme FMS *(Flight Management System)*, sarcini de asamblare și sisteme de asamblare.

Nevoia de specificații calitative ale controlului industrial, precum și nevoia pentru reprezentări ale informațiilor aproximative și incerte au dus la dezvoltare de rețele Petri fuzzy. Definirile acestor rețele au fost influențate, de regulă, de domeniile diverse de aplicare. Rețele Petri fuzzy au fost folosite pentru reprezentarea cunoștințelor și a raționamentului, precum și pentru modelarea sistemului de monitorizare și control al FMS.

Rețele Petri ordinare nu sunt suficient de puternice pentru reprezentarea și studiul unor proprietăți importante ale sistemelor concurente, precum eventualitatea (anumite tranziții trebuie să fie executate, în cele din urmă) și corectitudine *(fairness)* (dacă o tranziție devine executabilă de un număr infinit de ori, atunci trebuie să fie executată de un număr infinit de ori). Pentru a rezolva aceste probleme au fost create rețelele Petri temporale, în care sunt reprezentate constrângerile de timp prin operatori precum *next*, *hencefort*, *eventually*, *until* etc. Abilitatea rețelelor Petri temporale de a exprima eventualitatea au făcut aceste modele potrivite pentru reprezentarea și studiul comportamentului funcțional extern al sistemelor. Aceste funcționalități sunt exprimate prin relaționări de intrare/ieșire, spre exemplu dacă a fost stabilit un anumit șablon de intrare, atunci, în cele din urmă, va fi generat un anumit șablon de ieșire.

Deși încercările de a combina rețelele Petri cu alte tehnici, precum rețelele neuronale, logica fuzzy etc. par să fie în vogă, acest tip de rețele sunt, totuși, restricționate doar în cercetare și mediul academic. Acest lucru rezultă din lipsa unor unelte software ieftine și disponibile pe scară largă pentru dezvoltarea sistemelor industriale. Acest tip de unelte vor fi necesare pentru a asigura facilități de lucru cu probleme din domenii specifice pentru un nivel de pregătire relativ scăzut care să nu necesite cunoștințe privind rețelele Petri și metodele lor de analiză. Transformarea modelelor realizate cu rețele Petri în cod executabil va fi, de asemenea, esențială, permițând prototipizarea rapidă a sistemelor dezvoltate direct în mediul operațional

Un alt motiv pentru care rețelele Petri sunt folosite mai mult în mediu academic și în instituțiile de cercetare constă în dificultatea construcției modelelor. Aceasta necesită o mare experiență, mai ales pentru sistemele complexe și foarte mari. Nu există metodologie disponibilă, pentru a automatiza în vreun fel procedeul de construcție, ci modelele sunt concepute într-o manieră adhoc. În ultima perioadă s-au făcut încercări de a sistematiza acest aspect, clasificare, folosind termeni ai ingineriei software, în metode de jos în sus *(bottom-up)*, de sus în jos *(top-down)* și hibride. Refolosirea rețelelor Petri este, de asemenea, restricționată, mai ales de felul în care se concep modelele, pe baza unei documentații insuficiente. Este clar că, folosirea pe scară largă a rețelelor Petri, mai ales în industrie, va trebui să fie susținută de metode și unelte suport care să permită o construcție automată sau semiautomată a modelelor pe

baza specificațiilor de dezvoltare. Soluțiile care au încercat să pună în practică acest deziderat folosesc pentru construcția automată specificații exprimate prin reguli de producție, diagrame de fluxuri, logică temporală, limbaje semi-formale pentru anumite domenii de aplicare etc.

2 Bibliografie

1. *Petri Nets and Industrial Applications: A Tutorial.* **Zurawski, R., Zhou, M.C.** 6, s.l. : IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, 1994, Vol. 41.