Poglavje 3

Petrijeve mreže

Temelje *Petrijevih mrež*[1] je postavil Carl Adam Petri leta 1962, njihove osnove pa so opisane v viru [10]. Petrijeve mreže predstavljajo univerzalno matematično orodje za modeliranje in simulacijo dinamičnih in distribuiranih sistemov. Na osnovi postavljenega modela in izvedene simulacije lahko pridemo do *analize dinamike modela sistema*, ki nam pove, ali je slednja ustrezna ali ne. Dinamičen sistem skozi čas spreminja svoja stanja in tako pod pojmom dinamike v sistemu smatramo *časovno zaporedje stanj sistema*. Analiza dinamike modela nam pomaga pri odločitvah, ali je opazovani dinamičen sistem potrebno popraviti, dopolniti ali preoblikovati, s čimer dosežemo pravilno delovanje sistema.

V računalništvu se Petrijeve mreže uporabljajo za modeliranje na področjih analize pravilnosti delovanja komunikacijskih protokolov, analize zmogljivosti in zanesljivosti računalniških sistemov, analize pravilnosti delovanja algoritmov, analize sinhronizacije procesov itd., za modeliranje pa se uporabljajo tudi izven polja računalništva. Tako se Petrijeve mreže uporabljajo tudi za modeliranje bioloških omrežij [11], modeliranje prometnih omrežij, modeliranje interoperabilnih[2] sistemov itd.

3.1 Gradniki Petrijevih mrež

Struktura[3] Petrijeve mreže je sestavljena iz sledečih petih osnovnih tipov gradnikov:

- *akcij* (angl. *transitions*);
- *pogojev* (angl. *places*);
- *usmerjenih povezav* med akcijami in pogoji;

[1] Iskalnik Google nam na ključno besedo *Petri nets* oktobra 2019 vrne 31.000.000 zadetkov.

[2] Interoperabilnost - sposobnost opazovanega sistema, da sodeluje z okoljem in drugimi sistemi brez uporabnikovega poseganja (Vir: http://www.termania.net/).

[3] Struktura - kar je določeno z razporeditvijo elementov in razmerji med elementi, ki sestavljajo kako snov, predmet ali objekt (Vir: SSKJ).

- *usmerjenih povezav* med pogoji in akcijami;
- *žetonov*, s katerimi so številčno definirane kratnosti izpolnjenosti posameznih pogojev;

Dinamiko v kakršnemkoli dinamičnemu sistemu si lahko na abstraktnem nivoju interpretiramo kot zaporedje izvedenih *akcij*. Za izvedbo posamezne akcije morajo biti izpolnjeni neki zanjo specifični, vnaprej podani *pogoji*. Proženje (začetek izvajanja) posamezne akcije je tako pogojeno z vnaprej podano množico izpolnjenih pogojev. Uspešna izvedba akcije lahko povzroči neveljavnost izpolnjenosti pogojev potrebnih za njeno proženje, istočasno pa lahko izvedba akcije povzroči izpolnjenost nekih novih pogojev, ki ne sodijo v množico pogojev potrebnih za njeno proženje. Relacije med akcijami in pogoji v Petrijevih mrežah ponazorimo z *usmerjenimi povezavami*. Prva skupina usmerjenih povezav vodi od pogojev v akcije in definira potrebne izpolnjene pogoje za proženje akcij, druga skupina usmerjenih povezav pa vodi od akcij do pogojev in definira posledice, ki jih izvedbe akcij povzročijo v obliki novih izpolnjenosti izbranih pogojev.

Posamezen pogoj v Petrijevi mreži je lahko izpolnjen ali pa neizpolnjen. V primeru izpolnjenosti pogoja nas običajno zanima tudi *kratnost njegove izpolnjenosti*, ki nam poda odgovor na vprašanje, „kolikokrat je pogoj izpolnjen". Za označevanje kratnosti pogojev v Petrijevih mrežah uporabljamo *žetone*. Glede na povedano je tako nek opazovani pogoj lahko neizpolnjen (število žetonov v njem je 0), izpolnjen enkrat (število žetonov v njem je 1), ..., izpolnjen n-krat (število žetonov v njem je n) itd.

3.2 Definicije Petrijevih mrež

V pričujočem razdelku si bomo ogledali definiciji Petrijeve mreže in grafa Petrijeve mreže, v nadaljevanju pa osnovne koncepte formalizacije zapisovanja Petrijevih mrež ter pravila za proženje akcij in formacijo novega stanja prostora izpolnjenosti pogojev.

3.2.1 Formalna definicija Petrijeve mreže

Na začetku zapišimo formalno definicijo Petrijeve mreže (angl. *Petri net*, *place - transition net*) povzeto po viru [10].

Definicija 7 *Petrijeva mreža je četvorček $C = (P, T, I, O)$, pri čemer P predstavlja končno množico pogojev, T končno množico akcij, I vhodno in O izhodno preslikavo. Množici P in T sta si tuji ($P \cap T = \emptyset$).*

Vhodna preslikava I in izhodna preslikava O definirata relacije med akcijami in pogoji ali izvore in ponore usmerjenih povezav med akcijami in pogoji ter pogoji in akcijami. Vhodna preslikava I za opazovano akcijo t_j tako določa množico pogojev $I(t_j)$, iz katerih vodijo usmerjene povezave proti akciji t_j. Izhodna preslikava O za opazovano akcijo t_j določa množico pogojev $O(t_j)$, v katere vodijo usmerjene povezave iz akcije t_j. Tako usmerjene povezave vodijo le iz akcij v pogoje in obratno, niso pa dovoljene usmerjene povezave iz pogoja v pogoj ali iz akcije v akcijo.

Običajno preslikavi I in O zapišemo v obliki *prehajalnih matrik* reda $m \times n$, pri čemer m predstavlja število akcij ($m = |T|$) in n število pogojev ($n = |P|$), ali v obliki *multimnožic*[4]. Matrični zapis preslikav I in O lahko ponazorimo z izrazom

$$I = [i_{jk}]\,,\ O = [o_{jk}]\,,\ j = 1, ..., m,\ k = 1, ..., n,\ i_{jk}, o_{jk} \in \mathbb{N}_0. \tag{3.1}$$

Pri tem posamezni element i_{jk} matrike I ponazarja število usmerjenih povezav iz pogoja k v akcijo j, posamezni element o_{jk} matrike O pa število usmerjenih povezav iz akcije j v pogoj k.

V izrazih od (3.2) do (3.5) je predstavljen primer formalnega zapisa Petrijeve mreže s tremi akcijami, štirimi pogoji in desetimi usmerjenimi povezavami najprej v matrični notaciji, nato pa še v notaciji na osnovi multimnožic. Pri tem izraza (3.2) in (3.3) predstavljata matrični način zapisa, izrazi (3.2), (3.4) in (3.5) pa zapis na osnovi multimnožic.

$$C = (P, T, I, O), P = \{p_1, p_2, p_3, p_4\}\,, T = \{t_1, t_2, t_3\}\,, \tag{3.2}$$

$$I = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},\ O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

$$I(t_1) = \{p_1, p_2, p_3\}\,, I(t_2) = \{p_4\}\,, I(t_3) = \{p_3\}\,, \tag{3.4}$$

$$O(t_1) = \{p_1\}\,, O(t_2) = \{p_2, p_2, p_3\}\,, O(t_3) = \{p_4\}\,. \tag{3.5}$$

V nadaljevanju pričujočega dela bomo uporabljali oba načina zapisa.

3.2.2 Definicija grafa Petrijeve mreže

Pri praktičnem delu poleg formalnega matematičnega zapisa zaradi preglednosti za ponazarjanje Petrijevih mrež uporabljamo tudi njihovo grafično predstavitev. Slednjo imenujemo za *graf Petrijeve mreže*. Njegova definicija, povzeta po viru [10], je navedena v nadaljevanju.

Definicija 8 *Graf Petrijeve mreže je dvodelni usmerjeni graf G=(V,A), kjer V ($V = P \cup T$) predstavlja množico vozlišč in A multimnožico povezav med njimi ($a_i = (v_j, v_k)$). Velja naslednja relacija:*

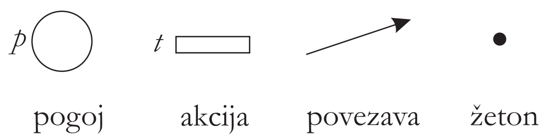
$$\forall i : a_i = (v_j, v_k),\ (v_j \in T)\&(v_k \in P) \vee (v_j \in P)\&(v_k \in T). \tag{3.6}$$

[4]Multimnožica (angl. *multiset*) je množica, v kateri je dovoljena večkratna pojavitev vsakega od elementov (Vir: Wikipedia).

Vozlišča dvodelnega grafa[5] nam predstavljajo vsi pogoji iz množice P in vse akcije iz množice T. Tako imamo v grafu opravka z $n+m$ vozlišči ($m = |T|$, $n = |P|$, $|V| = n+m$). Multimnožica A je sestavljena iz usmerjenih povezav, ki smo jih v matričnem zapisu zapisali v matriki I in O. Moč multimnožice usmerjenih povezav $|A|$ je izražena z vsoto vseh usmerjenih povezav upoštevajoč obe matriki preslikav po izrazu

$$|A| = \sum_{j=1}^{m}(\sum_{k=1}^{n} i_{jk}) + \sum_{j=1}^{m}(\sum_{k=1}^{n} o_{jk}). \tag{3.7}$$

Za tvorbo grafov Petrijevih mrež uporabljamo grafične primitive, ki so prikazani na sliki 3.1.



Slika 3.1: Grafični primitivi za tvorbo grafa Petrijeve mreže.

Predhodno smo v izrazih (3.2), (3.3) na osnovi matričnega zapisa formalno zapisali primer Petrijeve mreže. Njen graf je prikazan na sliki 3.2. Pomemben detajl s slike je označevanje akcij in pogojev z indeksi, ki nam omogoča povezovanje vozlišč po smiselnemu vrstnemu redu, ki izhaja iz indeksiranih elementov množic P in T in matrik I in O.

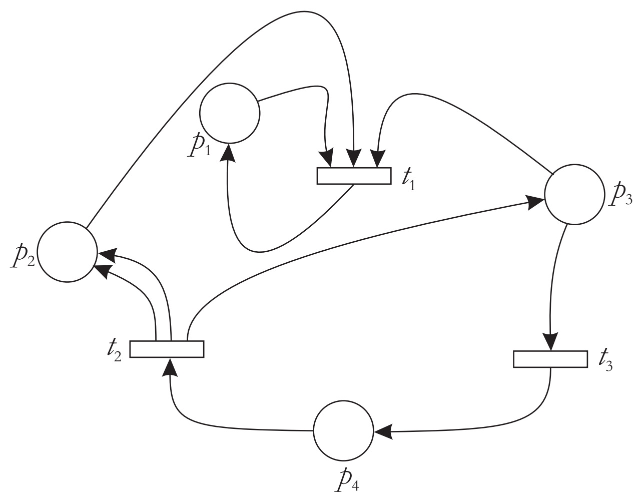
3.2.3 Označitve pogojev v Petrijevih mrežah

Kot smo povedali že v uvodu, z *označitvami pogojev* z žetoni v Petrijevih mrežah določamo ali odčitavamo kratnost izpolnjenosti pogojev. Formalno *označitev Petrijeve mreže* na k-tem časovnem koraku zapišemo z vektorjem $o(k)$

$$o(k) = (o_1(k), o_2(k), ..., o_n(k)),\ \forall i \in \{1, ..., n\}:\ o_i(k) \in \mathbb{N}_0, \tag{3.8}$$

pri čemer n predstavlja število pogojev ($n = |P|$), $o_i(k)$ pa število žetonov v i-tem pogoju ali kratnost izpolnjenosti i-tega pogoja na k-tem diskretnem časovnem koraku izvajanja modela Petrijeve mreže. Ob izvajanju dinamike na osnovi modela se skozi čas sprožajo različne akcije in praviloma se s tem spreminjajo tudi izpolnjenosti posameznih pogojev. Iz tega razloga je elementom vektorja $o_i(k)$ kot tudi celotnemu vektorju označitve $o(k)$ dodan diskretni časovni atribut k. Oglejmo si še formalno definicijo označitve podano v nadaljevanju.

[5]Dvodelni ali bipartitni graf je v teoriji grafov graf, ki se mu lahko vozlišča razdeli na dve med seboj tuji si množici, pri čemer ima vsaka povezava izvor v vozlišču iz ene od obeh množic, ponor pa v vozlišču iz preostale od obeh množic. Tako dvodelni graf vsebuje samo povezave, ki povezujejo vozlišča iz obeh med seboj tujih si množic, ne vsebuje pa nobene povezave, ki bi povezovala dve vozlišči iz iste množice [7].



Slika 3.2: Primer grafa Petrijeve mreže podane z izrazoma (3.2) in (3.3).

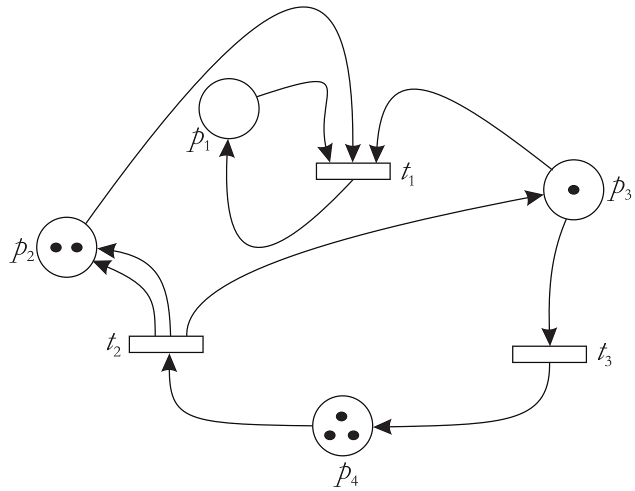
Definicija 9 *Označitev Petrijeve mreže C=(P,T,I,O) je funkcija, ki množico stanj P preslika v vektor nenegativnih celih števil po izrazu*

$$o : P \to \mathbb{N}_0. \tag{3.9}$$

Z označitvijo vseh pogojev pridemo do *označene Petrijeve mreže*, ki jo zapišemo kot petorček $M = (P, T, I, O, o(k))$. Upoštevajoč izraz (3.8), ki ne omejuje števila žetonov v posameznem pogoju, za vsako Petrijevo mrežo obstaja neskončno možno označitev, če ima Petrijeva mreža vsaj en pogoj.

Za primer Petrijeve mreže formalno zapisane v izrazih (3.2) in (3.3) predpostavimo *začetno označitev* $o(k_0) = (0, 2, 1, 3)$. Pod pojmom začetne označitve imamo v mislih označitev, ki je veljavna pred izvajanjem dinamike v Petrijevi mreži. V tem primeru bi prišli do grafične ponazoritve mreže, ki jo prikazuje slika 3.3. V vsakem od pogojev na sliki predstavimo kratnost izpolnjenosti pogoja z ustreznim številom žetonov.

V primeru, da je število žetonov v posameznem pogoju izredno veliko, v takšen pogoj namesto izrisa posameznih žetonov zapišemo število, ki predstavlja kratnost izpolnjenosti pogoja. Označitev mreže $o(k)$ pomensko enačimo s *stanjem*, v katerem se nahaja Petrijeva mreža na k-tem diskretnem časovnem koraku.



Slika 3.3: Primer označenega grafa Petrijeve mreže iz izrazov (3.2) in (3.3) ob podani začetni označitvi $o(k_0) = (0, 2, 1, 3)$.

3.2.4 Proženje izvajanja akcij v Petrijevih mrežah

Na tem mestu zapišimo najprej nekaj predpostavk, povezanih z izvajanjem akcij in pomenom časa v Petrijevih mrežah, ki se jih bomo držali v nadaljevanju pričujočega poglavja. Le te so sledeče:

- *diskretnost časa*: simulirana dinamika opazovanega modela na osnovi Petrijeve mreže se izvaja v diskretnih časovnih korakih;

- *istočasnost izvajanja le ene akcije*: na posameznem diskretnem časovnem koraku se lahko izvede le ena akcija; paralelno izvajanje več akcij hkrati tako ni možno;

- *hipnost izvedbe akcije*: čas trajanja izvedbe posamezne akcije je hipen ali infinitezimalen;

Upoštevajoč navedene predpostavke je dinamika v Petrijevi mreži enolično določena s *časovnim zaporedjem izvedenih akcij*, slednje pa lahko zapišemo tudi s *časovnim zaporedjem stanj* oziroma *časovnim zaporedjem označitev*.

Proženje izvajanja posamezne akcije na k-tem diskretnem koraku je pogojeno z označitvijo Petrijeve mreže $o(k)$. Posamezna akcija se lahko sproži, če je na k-tem diskretnem koraku *omogočena*. Akcija t_i je omogočena, če se po vsaki usmerjeni povezavi, ki vstopa v to akcijo, lahko prenese žeton iz pogoja, ki predstavlja izvor usmerjene povezave. Način preverjanja omogočenosti akcije t_i za izvajanje formalno zapišemo z izrazom

$$o(k) \geq e(t_i) * I, \ i = 1, ..., m, \tag{3.10}$$

pri čemer je $e(t_i)$ enotski vrstični vektor dolžine m ($m = |T|$), enica v njem pa se nahaja na i-ti poziciji, ki jo definira indeks i opazovane akcije t_i. I predstavlja vhodno preslikavo podano z matriko reda $m \times n$. V primeru, da velja relacija iz izraza (3.10) za vse istoležne pare levega in desnega vektorja (oba vektorja sta dimenzije $1 \times n$), je opazovana akcija t_j omogočena za proženje. Če je na k-tem diskretnem časovnem koraku ta akcija edina omogočena, se njeno izvajanje sproži, v primeru, pa da je v Petrijevi mreži na k-tem diskretnem koraku več omogočenih akcij, se izmed njih *nedeterministično* izbere za proženje le ena.

Zgled 13 *Za označeno Petrijevo mrežo s slike 3.3 določi akcije, ki so ob podani označitvi* $(0, 2, 1, 3)$ *omogočene za proženje.*

Rešitev na osnovi vizuelnega pregleda grafa Petrijeve mreže: *Iz slike je razvidno, da se preko vseh usmerjenih povezav, ki vstopajo v posamezno akcijo od pogojev (izvornih vozlišč) lahko žetoni prenesejo le v akciji t_2 (en žeton iz p_4) in t_3 (en žeton iz p_3). Tako sta akciji t_2 in t_3 na danem diskretnem časovnem koraku omogočeni za proženje. Akcija t_1 ni omogočena, saj ima kar tri vstopajoče usmerjene povezave, pri čemer se žeton vanjo lahko prenese le iz pogojev p_2 in p_3, iz pogoja p_1 pa ne. Ker s tem ni možno prenesti v akcijo t_1 žetonov iz vseh pogojev, iz katerih vodijo usmerjene povezave v opazovano akcijo, slednja ni omogočena za proženje.*

Rešitev na osnovi uporabe formalnega izraza (3.10): *Glede na veljavnost izrazov*

$$o(k) = (0, 2, 1, 3),\ e(t_1) = (1, 0, 0),\ e(t_2) = (0, 1, 0),\ e(t_3) = (0, 0, 1), \tag{3.11}$$

$$I = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.12}$$

lahko po vrsti zapišemo izraze za preverjanje relacije „$\geq$“

$$t_1 :\ o(0, 2, 1, 3) \geq (1, 1, 1, 0), \tag{3.13}$$

$$t_2 :\ o(0, 2, 1, 3) \geq (0, 0, 0, 1), \tag{3.14}$$

$$t_3 :\ o(0, 2, 1, 3) \geq (0, 0, 1, 0). \tag{3.15}$$

Upoštevajoč nastavljene izraze lahko ugotovimo, da je relaciji „$\geq$“ glede na parne primerjave istoležnih komponent zadoščeno v izrazih (3.14) in (3.15), kar pomeni, da sta akciji t_2 in t_3 omogočeni, izraz (3.13) pa relacije „$\geq$“ ne izpolnjuje, kar pomeni, da akcija t_1 ni omogočena.

Sprožitev in v nadaljevanju izvedba akcije t_i torej odvzame žetone iz pogojev, iz katerih vodijo usmerjene povezave proti opazovani akciji. Po hipnem infinitezimalnem trajanju akcije t_i pride do prenosa žetonov v pogoje, do katerih vodijo usmerjene povezave iz opazovane akcije. Po vsaki usmerjeni povezavi, ki vodi od akcije t_i do nekega pogoja p_j, se tako prenese en žeton, ki se odloži v tem ponornem pogoju p_j. Prehajanje žetonov od izvedene akcije t_i proti ponornim pogojem spremeni označitev pogojev Petrijeve mreže na naslednjem diskretnem časovnem koraku $k+1$, kar formalno zapišemo z izrazom

$$o(k+1) = o(k) + e(t_i)(O - I),\ i = 1, ..., m. \tag{3.16}$$

Pri tem O predstavlja matriko izhodne preslikave, ostale gradnike izraza (3.16) pa že poznamo. Na tem mestu poudarimo dve pomembni značilnosti prenosa žetonov preko prožene akcije t_j v ponorne pogoje. Le ti sta sledeči:

- čas prehajanja žetonov od pogojev, s katerimi je pogojena omogočenost akcije t_j, do pogojev, v katere ista akcija odlaga žetone, je hipen ali infinitezimalen;
- ni nujno, da je število žetonov, ki vstopajo v proženo akcijo enako številu žetonov, ki iz nje izstopajo;

Zgled 14 *Za označeno Petrijevo mrežo predstavljeno na sliki 3.3 določi označitev na naslednjem diskretnem časovnem koraku za primer, ko sprožimo akcijo t_2 (i) in za primer, ko sprožimo akcijo t_3 (ii).*

Rešitev na osnovi vizuelnega pregleda grafa Petrijeve mreže: *Kot izhodišče za rešitvi nam služi trenutna označitev Petrijeve mreže, ki je podana z izrazom*

$$o(k) = (0, 2, 1, 3). \tag{3.17}$$

(i) Akcija t_2 na vhodnem delu odvzame en žeton iz pogoja p_4, na izhodnem delu pa odpošlje dva žetona proti pogoju p_2 in en žeton proti pogoju p_3. Nova označitev bi tako bila $o(k+1) = (0, 4, 2, 2)$.

(ii) Akcija t_3 na vhodnem delu odvzame en žeton iz pogoja p_3, na izhodnem delu pa odpošlje en žeton proti pogoju p_4. Nova označitev bi tako bila $o(k+1) = (0, 2, 0, 4)$.

Rešitev na osnovi uporabe formalnega izraza (3.16): *Glede na veljavnost izrazov*

$$o(k) = (0, 2, 1, 3),\ e(t_1) = (1, 0, 0),\ e(t_2) = (0, 1, 0),\ e(t_3) = (0, 0, 1), \tag{3.18}$$

$$O - I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \tag{3.19}$$

izračunamo nove označitve po naslednjih izrazih:

(i) proženje akcije t_2:

$$o(k+1) = (0, 2, 1, 3) + (0, 1, 0) * \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} = \tag{3.20}$$

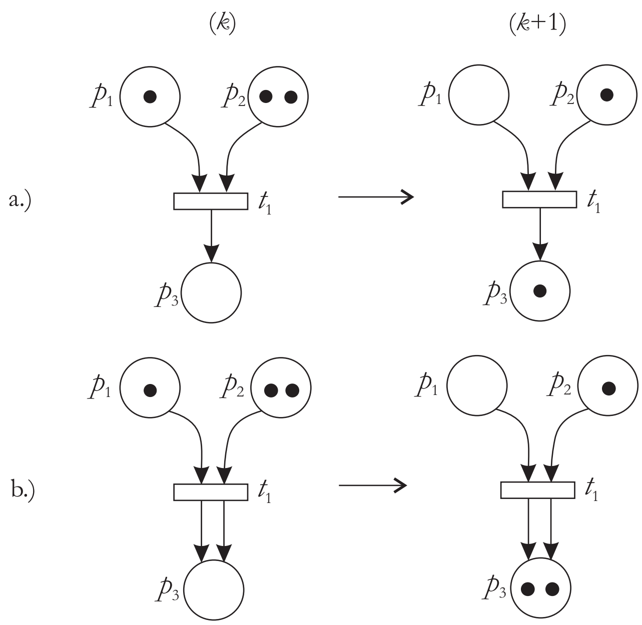
$$= (0, 2, 1, 3) + (0, 2, 1, -1) = (0, 4, 2, 2). \tag{3.21}$$

(ii) proženje akcije t_3:

$$o(k+1) = (0,2,1,3) + (0,0,1) * \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} = \tag{3.22}$$

$$= (0,2,1,3) + (0,0,-1,1) = (0,2,0,4). \tag{3.23}$$

Predhodno smo že omenili, da ni nujno, da se število vstopajočih žetonov v opazovano proženo akcijo ujema s številom žetonov, ki v nadaljevanju opazovano izvedeno akcijo zapustijo. Na sliki 3.4 imamo podana dva zgleda grafov označenih Petrijevih mrež. V primeru grafa na zgornjem delu slike (a) lahko ugotovimo, da pri proženju akcije t_1 v akcijo vstopata dva žetona (prvi iz pogoja p_1 in drugi iz pogoja p_2), akcija pa proti pogoju p_3 odpošlje le en žeton. V primeru na spodnjem delu slike (b) je število žetonov, ki vstopajo v akcijo enako, kot je število izstopajočih žetonov. V tem primeru vstopa v akcijo t_1 en žeton iz pogoja p_1 in en žeton iz pogoja p_2, akcija pa proti pogoju p_3 odpošlje dva žetona (po vsaki usmerjeni povezavi po en žeton).



Slika 3.4: Primera pretoka žetonov v Petrijevih mrežah. V zgornji Petrijevi mreži se celotno število žetonov v mreži ob proženju akcije t_1 spremeni, v spodnji pa se število žetonov ob proženju iste akcije ohrani.

Iz podanega primera pridemo do sklepa, da je sprememba celotnega števila žetonov v Petrijevi mreži ob izvedbi posamezne akcije (spremembi stanja Petrijeve mreže) odvisna od razmerja med številom vstopajočih povezav v to akcijo in številom izstopajočih povezav iz te akcije. V primeru, da je to razmerje enako 1, se celotno število žetonov v Petrijevi mreži ohrani, v nasprotnem primeru pa spremeni.

3.2.5 Drevo označitev v Petrijevi mreži

Razlago pojma *drevesa označitev* v Petrijevi mreži bomo predstavili na konkretnem zgledu Petrijeve mreže. Predpostavimo, da je podana Petrijeva mreža z matričnim zapisom po izrazu

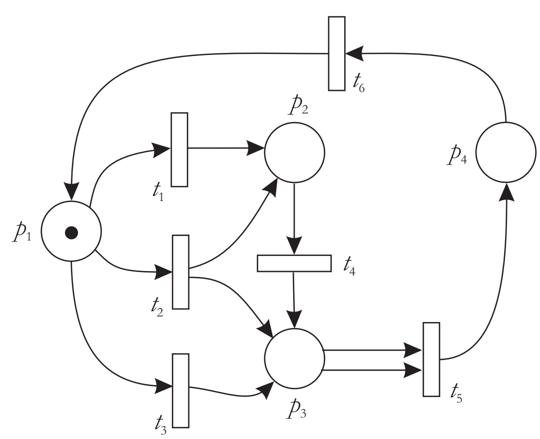
$$P = \{p_1, p_2, p_3, p_4\}\,,\ T = \{t_1, t_2, t_3, t_4, t_5, t_6\}\,,$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ O = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.24}$$

in začetno označitvijo po izrazu

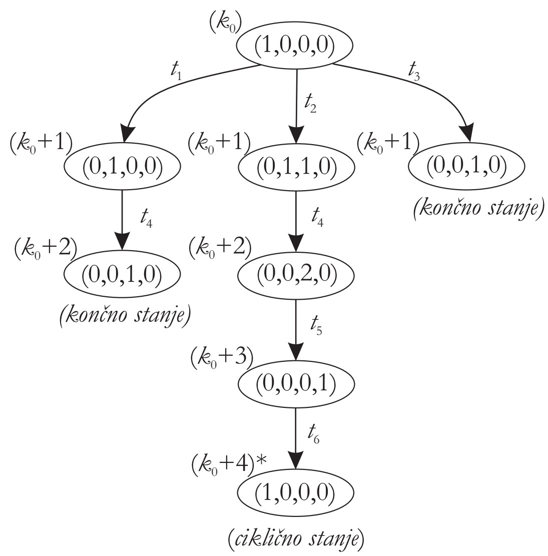
$$o(k_0) = (1, 0, 0, 0). \tag{3.25}$$

Na sliki 3.5 je predstavljen graf podane Petrijeve mreže z začetno označitvijo žetonov. Glede na podano začetno označitev so na osnovi vizuelnega pregleda



Slika 3.5: Označen graf Petrijeve mreže podane z matričnim zapisom po izrazu (3.24).

grafa na k_0-tem koraku omogočene akcije t_1, t_2 in t_3. Do enake ugotovitve bi prišli tudi s formalnim preverjanjem omogočenosti akcij po izrazu (3.10). Ob predhodno navedeni predpostavki, da se istočasno ne more sprožiti več akcij, imamo iz začetne označitve (začetnega stanja Petrijeve mreže) tri možne tokove dinamike, ki jih ponazorimo z *drevesom označitev*, predstavljenim na sliki 3.6. Pri tem lahko označitve v drevesu določamo na osnovi vizuelnega pregleda grafa ali pa na osnovi računskega izraza (3.16).



Slika 3.6: Drevo označitev za primer Petrijeve mreže s slike 3.5. Ovalna vozlišča predstavljajo stanja ali označitve Petrijeve mreže, vsako od stanj pa je označeno z indeksom diskretnega časovnega koraka $(k_0, k_0+1, ..., k_0+4)$.

Drevo označitev s slike 3.6 ponazarja tri možne tokove dinamike. Vsakega od njih lahko zapišemo s *časovnim zaporedjem stanj* Petrijeve mreže ali s *časovnim zaporedjem označitev*, ter s *časovnim zaporedjem izvedenih akcij.* Omenjeni tokovi dinamike so sledeči:

- tok z začetnim proženjem akcije t_1: po dveh izvedenih akcijah dosežemo **končno stanje** (končno označitev) $(0,0,1,0)$, v katerem obtiči model; s tem se dinamika v Petrijevi mreži ustavi;
  - časovno zaporedje izvedenih akcij: $(t_1,\ t_4)$;
  - časovno zaporedje stanj: $(1,0,0,0) \mapsto (0,1,0,0) \mapsto (0,0,1,0)$;
- tok z začetnim proženjem akcije t_2: po štirih izvedenih akcijah se vrnemo v začetno stanje $(1,0,0,0)$; omenjeno stanje imenujemo za **ciklično stanje**, enako pa bi lahko imenovali tudi vsa njemu predhodna stanja vse do njegove prejšnje pojavitve;
  - časovno zaporedje izvedenih akcij: $(t_2,\ t_4,\ t_5,\ t_6)$;

  - časovno zaporedje stanj: $(1,0,0,0) \mapsto (0,1,1,0) \mapsto (0,0,2,0) \mapsto (0,0,0,1) \mapsto (1,0,0,0) \mapsto$ ...;

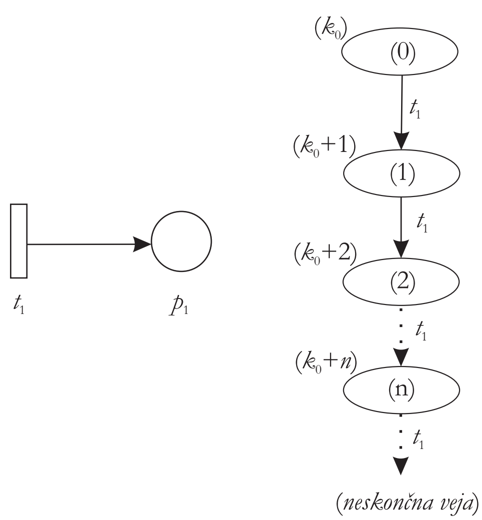
- tok z začetnim proženjem akcije t_3: po eni izvedeni akciji dosežemo **končno stanje** (končno označitev) $(0,0,1,0)$, v katerem obtiči model; s tem se dinamika v Petrijevi mreži ustavi;
  - časovno zaporedje izvedenih akcij: (t_3);
  - časovno zaporedje stanj: $(1,0,0,0) \mapsto (0,0,1,0)$;

Ob analizi dinamike v Petrijevi mreži moramo biti pozorni tudi na akcije brez vhodnih povezav, ki bi vodile od nekih pogojev do opazovane akcije. Ob predpostavki, da je t_j takšna akcija, zanjo v notaciji multimnožic velja izraz

$$I(t_j) = \emptyset. \tag{3.26}$$

Tovrstne akcije so vedno omogočene in se lahko neprestano prožijo, kar posledično vodi v drevesa označitev z **neskončnimi vejami**.

Na levem delu slike 3.7 je predstavljen primer Petrijeve mreže z akcijo t_1 brez vhodnih povezav, na desnem delu pa neskončna veja v njenem drevesu označitev.



Slika 3.7: Primer grafa Petrijeve mreže, kjer je akcija t_1 brez vhodnih povezav in se neprestano proži.

Ključni dejavniki, na katere moramo biti pozorni pri tvorbi in analizi dreves označitev, so tako sledeči:

- *končna stanja* ali *listi drevesa*: končna stanja v drevesu označitev predstavljajo stanja, v katerih se dinamika v modeliranem sistemu ustavi in opcijsko lahko predstavljajo nepravilno delovanje sistema; praviloma se iz teh stanj modelirani dinamični sistem brez posredovanja iz zunanjega sveta (npr. uporabnika sistema) ne more izvleči;

- *ciklična stanja*: ciklična stanja so v modelih sistemov zaželjena, saj po opravljeni dejavnosti vidni skozi zaporedje akcij ponazarjajo zmožnost vračanja sistema v neko začetno stanje pripravljenosti;
- *neskončne veje v drevesu*: neskončne veje v drevesu označitev modeliranega sistema so nezaželene, saj v realnih sistemih (npr. v računalnikih) ne moremo zagotoviti poljubne količine resursov (npr. pomnilnika), s katerimi bi lahko realizirali poljubno veliko število možnih različnih sistemskih stanj;

Glede na povedano si v drevesu označitev modeliranega sistema na osnovi Petrijevih mrež želimo čimveč cikličnih stanj, temeljito moramo preveriti smiselnost obstoja končnih stanj, pojavitve neskončnih vej v drevesu pa so enostavno nezaželjene. Ena od metrik za oceno modeliranega sistema je tudi število njegovih različnih stanj ali različnih označitev, do katerih pridemo skozi proces simulacije dinamike. Obvladljivejši so sistemi, pri katerih je število različnih označitev na nivoju modeliranja manjše.

3.2.6 Dosegljivost stanj v Petrijevi mreži

Označitev o' je *neposredno dosegljiva* iz označitve o, ko obstaja takšna akcija t_j, ki nas z izvajanjem pripelje iz označitve o v označitev o'. Povedano drugače je o' neposredno dosegljiva iz o takrat, ko velja izraz

$$o' = \delta(o, t_j), \tag{3.27}$$

pri čemer δ predstavlja funkcijo, ki staro označitev o (natančneje $o(k)$) na osnovi izvedene akcije t_j preslika v novo označitev o' (natančneje $o(k+1)$).

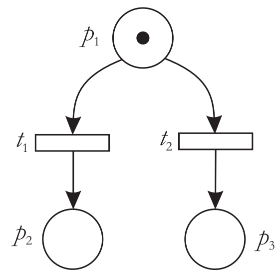
Množica *dosegljivih stanj* $R(C, o)$ je množica vseh označitev Petrijeve mreže C, ki so dosegljive iz označitve o. Označitev o' pripada množici $R(C, o)$, če obstaja zaporedje akcij, ki nas pripelje iz označitve o v označitev o'. Omenjeno zaporedje mora vsebovati najmanj eno akcijo. V tem primeru velja pravilo

$$if\ (o' \in R(C, o))\ and\ (\exists t_j \in T : o'' = \delta(o', t_j))\ then\ o'' \in R(C, o), \tag{3.28}$$

ki ponazarja, da so vse neposredno dosegljive označitve o'' iz označitve o' tudi v množici dosegljivih stanj $R(C, o)$. Zaradi preglednosti zapisa smo časovne indekse označitev v tem razdelku namerno izpustili.

3.2.7 Nedeterminizem in konkurenčnost

Eden od temeljnih hipotetičnih konstruktov, ki ga Petrijeve mreže omogočajo, je *nedeterminizem* izbire akcije za proženje. Omenjeno značilnost razložimo s primerom Petrijeve mreže, podane z grafom predstavljenim na sliki 3.8. Glede



Slika 3.8: Graf Petrijeve mreže nedeterministične izbire poti žetona in konkurenčnosti akcij t_1 in t_2.

na označitev Petrijeve mreže sta za proženje omogočeni akciji t_1 in t_2, pri čemer se izbira akcije, ki bo sprožena izvede nedeterministično ali naključno. Slednje pomeni, da bo naključno izbrana akcija od obeh omogočenih prevzela žeton in bo izvedena, druga akcija pa v nadaljevanju zaradi odsotnosti žetona v pogoju p_1 ne bo izvedena.

Pojem *konkurenčnosti* v primeru s slike 3.8 ponazarja konkuriranje ali tekmovanje akcij t_1 in t_2 za prevzem žetona iz pogoja p_1. Akcija, ki bo za svoje proženje prejela žeton, bo izbrana nedeterministično.

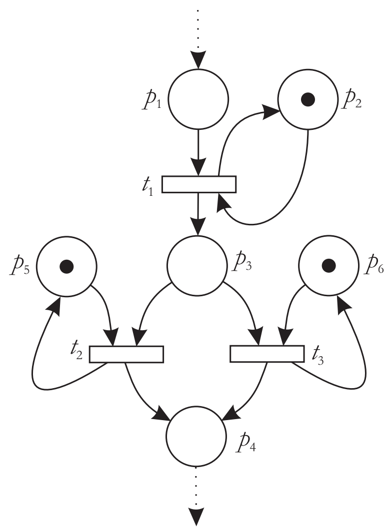
3.3 Splošni zgledi modeliranja s Petrijevimi mrežami

V pričujočem razdelku si bomo ogledali nekaj konkretnih primerov modeliranja različnih dinamičnih sistemov s Petrijevimi mrežami, s čimer se bomo končno usmerili tudi na pomene akcij in pogojev, o katerih do sedaj nismo govorili. Večina primerov je povzeta po viru [10], vsi dinamični sistemi pa so predstavljeni z grafi Petrijevih mrež.

3.3.1 Računalniški strežniški sistem

Predpostavimo, da imamo opravka z računalniškim strežniškim sistemom, ki ga sestavljajo računalniške enote R_1, R_2 in R_3. Vsaka zahteva, ki vstopi v strežniški sistem, mora biti najprej obdelana na enoti R_1, v nadaljevanju pa na enoti R_2 ali na enoti R_3. Na sliki 3.9 je predstavljen graf Petrijeve mreže, ki ponazarja opisani sistem, pri čemer prekinjene povezave označujejo vhodno in izhodno točko strežniškega sistema. Pomeni posameznih pogojev in akcij na sliki 3.9 so sledeči:

- p_1: v sistem je vstopila zahteva in čaka na obdelavo v enoti R_1;
- p_2: enota R_1 je prosta;



Slika 3.9: Graf Petrijeve mreže za ponazoritev računalniškega strežniškega sistema.

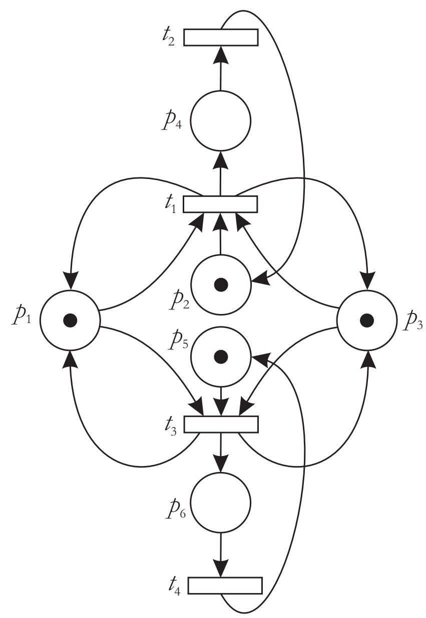
- p_3: zahteva čaka na obdelavo v enoti R_2 ali v enoti R_3;
- p_4: zahteva je dokončno obdelana;
- p_5: enota R_2 je prosta;
- p_6: enota R_3 je prosta;
- t_1: zahteva se obdeluje na R_1;
- t_2: zahteva se obdeluje na R_2;
- t_3: zahteva se obdeluje na R_3;

Komentar modela s slike 3.9 bi bil sledeč:

- predstavljeni model vsebuje nedeterministično izbiro enote R_2 (proženje akcije t_2) ali R_3 (proženje akcije t_3), do katere pride v pogoju p_3; slednja je nedeterministična le pod pogojem, da sta obe enoti prosti; v primeru, da je prosta le ena, zahtevo prevzame prosta enota, v primeru pa da sta zasedeni obe, zahteva čaka na prvo prosto enoto;
- izpolnjeni pogoji p_2, p_5 in p_6 predstavljajo proste računalniške enote; imenujemo jih za *resurse*, do katerih mora priti zahteva, da bi bila ustrezno postrežena; pomembno je, da po zaseganju resursa (prevzemu žetona iz pogoja, ki označuje prostost resursa) in izvedbi akcije zahteva resurs sprosti (vrne žeton v pogoj, ki označuje prostost resursa); v ta namen imamo na sliki povezave (t_1, p_2), (t_2, p_5) in (t_3, p_6);

3.3.2 Problem dveh kitajskih mislecev

Dva kitajska misleca sedita za okroglo mizo. Vsak od njiju se izmenično bodisi prehranjuje, bodisi razmišlja. Za prehranjevanje potrebuje posamezni mislec dve paličici, pri čemer je na začetku dinamičnega procesa ena paličica na levi in druga paličica na desni med sedočima mislecema. Z vidika posameznega misleca je tako prisotnih dovolj paličic za prehranjevanje, z vidika obeh pa število razpoložljivih paličic omogoča istočasno prehranjevanje le enemu od obeh mislecev. Na sliki 3.10 je predstavljen graf Petrijeve mreže, ki ponazarja opisani sistem. Pomeni posameznih pogojev in akcij na sliki 3.10 so sledeči:



Slika 3.10: Graf Petrijeve mreže za ponazoritev problema dveh kitajskih meditatorjev.

- p_1: prva paličica je prosta;
- p_2: prvi mislec je pripravljen za prehranjevanje;
- p_3: druga paličica je prosta;
- p_4: prvi mislec je pripravljen za razmišljanje;
- p_5: drugi mislec je pripravljen za prehranjevanje;
- p_6: drugi mislec je pripravljen za razmišljanje;
- t_1: prvi mislec se prehranjuje;

- t_2: prvi mislec razmišlja;
- t_3: drugi mislec se prehranjuje;
- t_4: drugi mislec razmišlja;

Komentar modela s slike 3.10 bi bil sledeč:

- model sistema vsebuje dva resursa (paličici), ki ju ponazarjata pogoja p_1 in p_3; pomembno je, da po izvedenem prehranjevanju (izvedbi akcij t_1 ali t_3) pride do vračila paličic v njuna izhodiščna pogoja p_1 in p_3;
- izredno pomembna je začetna označitev, ki z žetoni v pogojih p_1 in p_3 inicializira začetne resurse (paličici) in z žetoni v pogojih p_2 in p_5 začetni stanji, v katerih se nahajata misleca;
- zajetje obeh paličic je izvedeno nedeterministično, saj na osnovi označitve ne moremo določiti, kateri od obeh mislecev bo pridobil obe paličici; pri dodelitvi paličic lahko pride do *smrtnega objema* (angl. *deadlock*) ali končnega stanja drevesa označitev Petrijeve mreže, če eden od mislecev pridobi eno paličico, drugi pa drugo; v tem primeru bosta oba misleca čakala še na pridobitev druge paličice, pri čemer je nobeden od njiju ne bo sprostil, kar vodi v stanje sistema (označitev), ki je končno;

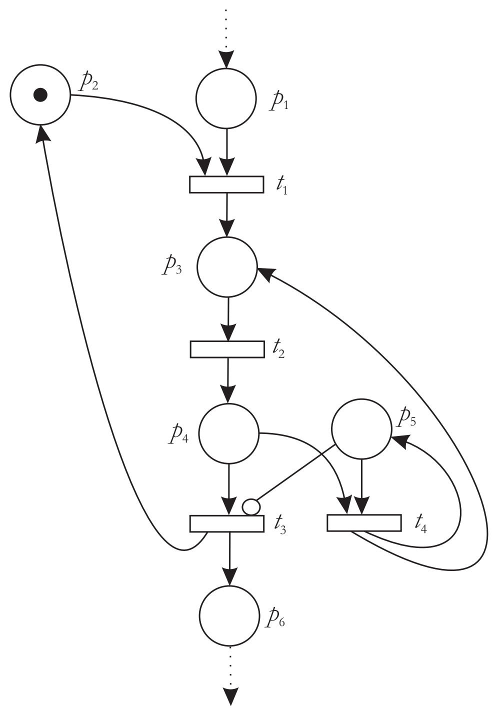
3.3.3 DO WHILE programski stavek s čuvajem

DO WHILE programski stavek definira število izvajanj zaporedja programskih ukazov S, pri čemer se zaporedje prvič izvede brezpogojno, vse njegove nadaljnje izvedbe pa so pogojene z veljavnostjo pogoja P. Formalno programski stavek zapišemo z izrazom

$$do\ \{S\}\ while(P). \tag{3.29}$$

Na sliki 3.11 je predstavljen graf Petrijeve mreže, ki ponazarja opisani stavek. Pomeni posameznih pogojev in akcij na sliki 3.11 so sledeči:

- p_1: v sistem je vstopila zahteva za izvajanje programskega stavka;
- p_2: pogoj predstavlja funkcijo *čuvaja*; le ob njegovi prisotnosti je možno izvršiti programski stavek;
- p_3: zahteva čaka na izvedbo S stavka;
- p_4: izvedba S stavka je opravljena;
- p_5: pogoj P;
- p_6: DO WHILE stavek je končan;
- t_1: ob prisotnosti čuvaja se izvede vstop v programski stavek;
- t_2: izvede se S stavek,



Slika 3.11: Graf Petrijeve mreže za ponazoritev DO WHILE programskega stavka.

- t_3: ker pogoj P ni izpolnjen, se izvajanje programskega stavka zaključi in čuvaj se vrne v svoj izhodiščni pogoj;
- t_4: ker je pogoj P izpolnjen, se vrnemo na ponovno brezpogojno izvajanje stavka S;

Komentar modela s slike 3.11 bi bil sledeč:

- pogoj p_2 predstavlja funkcijo *čuvaja*, ki spremlja zahtevo za izvajanjem programskega stavka skozi notranjost grafa Petrijeve mreže; v nadaljevanju njegova odsotnost v izhodiščnem pogoju p_2 onemogoča vstopanje novih zahtev v programski stavek, če je le ta že zaseden; s tem onemogočimo nekonsistentnost njegovega izvajanja (npr. vrednosti spremenljivk, število ponovitev zaporedja ukazov itd.); ko se posamezno izvajanje programskega stavka preko akcije t_3 dokonča, se čuvaj vrne v izhodiščni pogoj p_2 in je pripravljen za spremljanje naslednje zahteve za izvajanje programskega stavka; v splošnem nam torej čuvaj onemogoča vstopanje večjega števila zahtev (žetonov) v podsegment grafa Petrijeve mreže;
- iz pogoja p_5 v akcijo t_3 vstopa nam do sedaj nepoznana vrsta povezave, ki se ne zaključuje s puščico, temveč s krožcem; gre za *inhibirno* povezavo, ki

ima nasproten pomen, kot ga ima običajna povezava; po inhibirni povezavi se v akcijo pripelje žeton (in s tem soaktivira akcijo), če v izvornem pogoju inhibirne povezave ni žetona, ali povedano drugače, če izvorni pogoj inhibirne povezave ni izpolnjen; v našem primeru se izvajanje programskega stavka zaključuje preko akcije t_3, pri čemer se to zgodi takrat, ko pogoj P (p_5) ni več izpolnjen;

- po vsakem zaključku akcije t_2 (izvedba zaporedja programskih ukazov) preverjamo pogoj P (p_5); pri tem je iz slike 3.11 razvidno, da z uspešnim preverjanjem (izvedbo akcije t_4) pogoj p_5 spremenimo (mu odvzamemo žeton), kar pomeni, da moramo po preverjanju omenjeni žeton vrniti v opazovani pogoj p_5;
- pomembno je začetno nahajanje žetona v pogoju p_2, s čimer modelu dodelimo stražarja; brez njega izvajanje programskega stavka ne bi bilo mogoče;

Ob uporabi inhibirnih povezav se moramo zavedati, da v tem primeru nimamo več na voljo formalnega zapisa Petrijeve mreže (problem zapisa preslikav I in O) in ne veljata več izraza (3.10) in (3.16). Uporaba inhibirnih povezav ne sodi v domeno klasičnih Petrijevih mrež, temveč je ena od njihovih možnih razširitev.

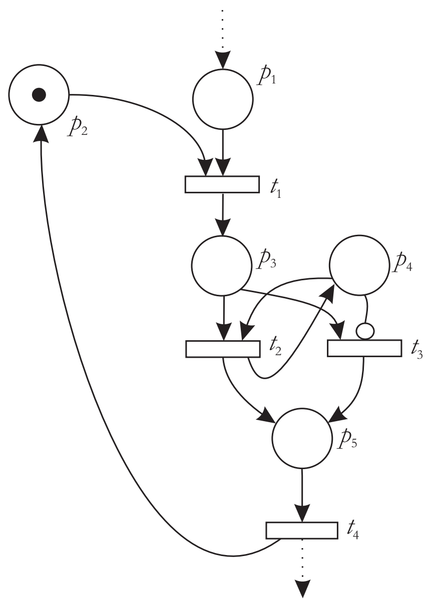
3.3.4 IF programski stavek s čuvajem

IF programski stavek ob izpolnjenosti pogoja P vodi v izvajanje zaporedja programskih ukazov S_1, ob njegovi neizpolnjenosti pa v izvajanje zaporedja programskih ukazov S_2. Formalno programski stavek zapišemo z izrazom

$$if\ (P)\ then\ \{S_1\}\ else\ \{S_2\}\,. \tag{3.30}$$

Na sliki 3.12 je prikazan graf Petrijeve mreže, ki ponazarja opisani stavek, v katerem pogoj p_2 zopet predstavlja funkcijo čuvaja. Pomeni posameznih pogojev in akcij na sliki 3.12 so sledeči:

- p_1: v sistem je vstopila zahteva za izvajanje programskega stavka;
- p_2: pogoj predstavlja funkcijo čuvaja programskega stavka;
- p_3: zahteva je pred preverjanjem veljavnosti pogoja P, ki določi katero zaporedje ukazov izvesti;
- p_4: pogoj P;
- p_5: eno od obeh možnih zaporedij programskih ukazov je izvedeno;
- t_1: ob prisotnosti čuvaja se izvede vstop v programski stavek;
- t_2: izvede se zaporedje programskih ukazov S_1;
- t_3: izvede se zaporedje programskih ukazov S_2;



Slika 3.12: Graf Petrijeve mreže za ponazoritev IF programskega stavka.

- t_4: programski stavek se z vračilom čuvaja v izhodiščni pogoj dokončno izvede;

Komentar modela s slike 3.12 bi bil sledeč:

- za vejanje programskega toka ali izbiro zaporedja programskih ukazov izvajamo preverjanje pogoja P (pogoj p_4), v katerem tako kot v primeru DO WHILE stavka uporabimo inhibirno povezavo;

- preverjanje pogoja P preko povezave (p_4,t_2) spreminja veljavnost pogoj P (p_4), zato akcija t_2 v pogoj p_4 po izvedbi vrne en žeton;

3.3.5 FOR programski stavek s čuvajem

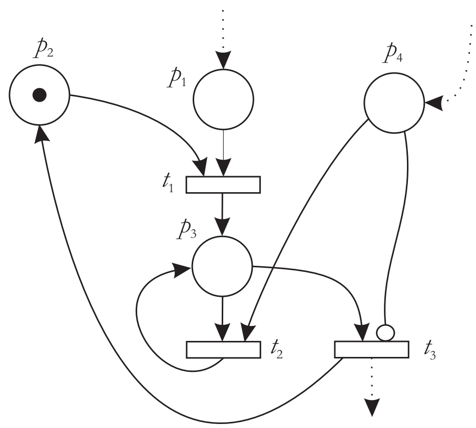
FOR programski stavek definira n-kratno izvajanje zaporedja programskih ukazov S. Formalno programski stavek zapišemo z izrazom

$$for\ (i = 1\ to\ n)\ do\ \{S\}\,. \tag{3.31}$$

Na sliki 3.13 je prikazan graf Petrijeve mreže, ki ponazarja opisani stavek. Pomeni posameznih pogojev in akcij na sliki 3.13 so sledeči:

- p_1: v sistem je vstopila zahteva za izvajanje programskega stavka;

- p_2: pogoj predstavlja funkcijo čuvaja programskega stavka;



Slika 3.13: Graf Petrijeve mreže za ponazoritev FOR programskega stavka.

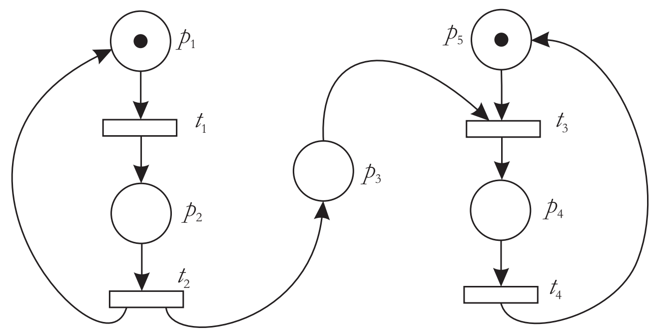
- p_3: zahteva je pred odločitvijo ali stavek S glede na vrednost števca n izvesti ali ne;
- p_4: predstavlja števec n, ki mora biti pred začetkom izvajanja programskega stavka inicializiran z n žetoni;
- t_1: ob prisotnosti čuvaja se izvede vstop v programski stavek;
- t_2: izvede se stavek S;
- t_3: programski stavek je dokončno izveden v celoti;

Komentar modela s slike 3.13 bi bil sledeč:

- črtkana povezava, ki vstopa v p_4 brez izvora ponazarja potrebo, da se pred samo izvedbo FOR stavka inicializira število žetonov (število n) v pogoju p_4, črtkana povezava, ki vstopa v p_1 pa inicializacijo programskega stavka;
- pogoj p_4 predstavlja števec, ki ga z vsako izvedbo zaporedja programskih ukazov S dekrementiramo za 1; ko le ta pade na 0, izvedba zaporedja programskih ukazov S ni več mogoča in zaradi inhibirne povezave (p_4,t_3) se izvede akcija t_3, ki vrne čuvaja na izhodiščno mesto in s tem je programski stavek končan;

3.3.6 Proizvodno porabniški sistem

Proizvodno porabniški sistem (angl. *producer consumer system*) je sestavljen iz dveh procesov in sicer proizvodnega in porabniškega. Ključni problem omenjenega sistema je vzpostavitev sinhronizacije med obema procesoma. Na sliki 3.14 je predstavljen graf Petrijeve mreže, ki ponazarja izhodišče za omenjeno sinhronizacijo. Levi del slike predstavlja proizvodni proces, desni del slike pa porabniški proces. S prikazanim modelom ponazorimo predajo proizvedenih enot porabniškemu procesu. Pomeni posameznih pogojev in akcij s slike 3.14 so



Slika 3.14: Graf Petrijeve mreže za ponazoritev proizvodno porabniškega sistema. Levi del slike predstavlja proizvodni, desni del slike pa porabniški proces.

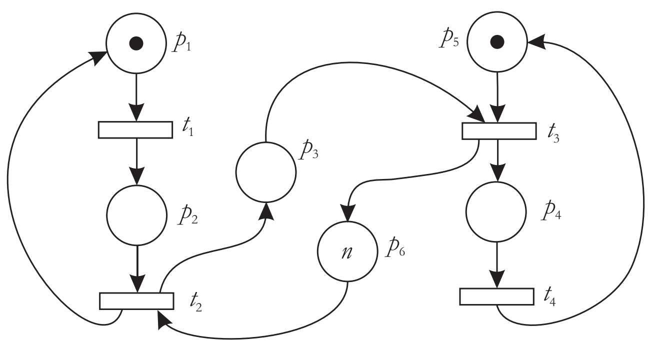
sledeči:

- p_1: proizvodni proces je pripravljen na proizvodnjo ene enote;
- p_2: ena enota je proizvedena;
- p_3: pogoj predstavlja funkcijo vmesnika (angl. *buffer*), kjer proizvedene enote čakajo na prevzem s strani porabniškega procesa;
- p_4: ena enota je prevzeta iz vmesnika;
- p_5: porabniški proces je pripravljen na porabo ene enote;
- t_1: proizvede se ena enota;
- t_2: proizvodni proces preda enoto v vmesnik (p_3) in inicializira se ponovna pripravljenost na proizvodnjo (žeton preide v p_1);
- t_3: porabniški proces prevzame enoto iz vmesnika;
- t_4: porabniški proces porabi enoto in inicializira se ponovna pripravljenost na porabo (žeton preide v p_5);

Komentar modela s slike 3.14 bi bil sledeč:

- pomembna je začetna označitev, s katero inicializiramo stanje proizvodnega (žeton v pogoju p_1) in porabniškega procesa (žeton v pogoju p_5); brez vsaj enega žetona v proizvodnem procesu in vsaj enega žetona v porabniškem procesu izmenjava enot med procesoma ne bi bila mogoča, saj bi bila dinamika obeh procesov v celoti onemogočena;

- evidentna slabost modela se izkazuje z dogajanjem v pogoju p_3, v katerem lahko pride do prekomernega kopičenja števila žetonov, ki jih realni vmesnik (skladišče enot) zaradi svoje omejene kapacitete ne bi bil zmožen hraniti; v ta namen na sliki 3.15 predstavimo graf Petrijeve mreže, ki predstavlja izboljšan model opisanega sistema; v njem nastopa nov pogoj p_6, v katerem je na začetku n žetonov, pri čemer število n predstavlja kapaciteto vmesnika; ko se iz p_6 odstrani vse žetone, akcija t_2 ne more več odlagati enot v vmesnik in je potrebno za nadaljevanje procesa proizvodnje počakati fazo porabe, da v p_6 dostavi vsaj en žeton (odvzame iz vmesnika vsaj eno enoto);

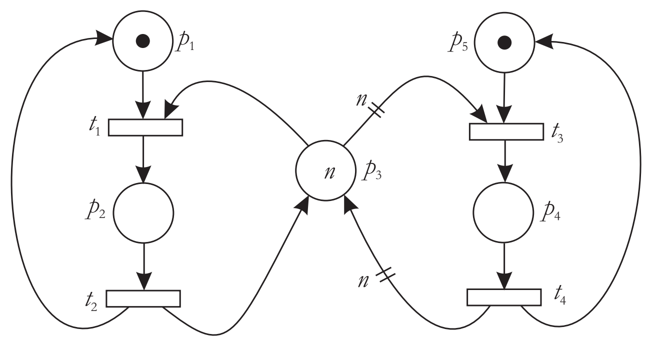


Slika 3.15: Graf Petrijeve mreže za ponazoritev proizvodno porabniškega procesa s končnim vmesnikom.

3.3.7 Problem branja in pisanja

Problem branja in pisanja (angl. *read - write problem*) izhaja iz njunega eventuelnega paralelnega izvajanja v domeni elektronskih podatkovnih baz, do katerih dostopa večje število uporabnikov in s tem posledično do potrebne sinhronizacije obeh procesov. Pri tem zahteve uporabnikov ločujemo na skupini zahtev za branje in zahtev za pisanje. V primeru porajanja zahtev za branje običajno omogočamo n paralelnih dostopov (pravic do branja) do podatkovne baze, v primeru porajanja zahtev za pisanje pa moramo predhodno podatkovno bazo zakleniti in s tem onemogočiti vsa aktivna branja in morebitna ostala pisanja. Na sliki 3.16 je predstavljen graf Petrijeve mreže, ki ponazarja opisani sistem, pri čemer levi del slike predstavlja proces porajanja zahtev za branje, desni del slike pa proces porajanja zahtev za pisanje. Pomeni posameznih pogojev in akcij s slike 3.16 so sledeči:

- p_1: pripravljena je zahteva za branje;
- p_2: omogočen je dostop do enkratnega branja podatkov;
- p_3: vmesnik, v katerem je shranjenih n dostopnih pravic;



Slika 3.16: Graf Petrijeve mreže za ponazoritev problema branja in pisanja, pri čemer levi del slike predstavlja proces porajanja zahtev za branje, desni del slike pa proces porajanja zahtev za pisanje.

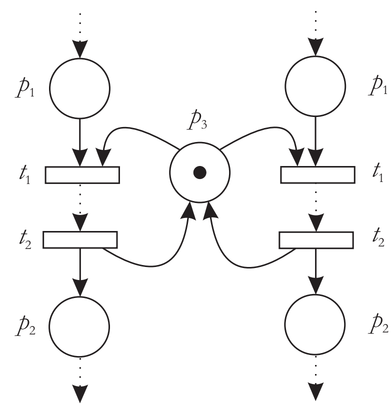
- p_4: omogočen je dostop do enkratnega pisanja podatkov;
- p_5: pripravljena je zahteva za pisanje;
- t_1: izvede se zaseganje začasne pravice za enkratno branje iz p_3;
- t_2: izvede se branje, v nadaljevanju pa se začasna pravica do branja vrne v p_3;
- t_3: izvede se zaseganje začasne n-kratne pravice za pisanje (zaseg n žetonov iz pogoja p_3);
- t_4: izvede se pisanje, v nadaljevanju pa se začasna n-kratna pravica do pisanja vrne v p_3;

Komentar modela s slike 3.16 bi bil sledeč:

- pogoj p_3 predstavlja skladišče bralno - pisalnih pravic; začetno število žetonov v p_3 predstavlja število vseh bralno - pisalnih pravic;
- začetna označitev mora vsebovati vsaj en žeton v enem od pogojev v levem bralnem ciklu, vsaj en žeton v enem od pogojev v desnem pisalnem ciklu in vsaj en žeton v skladišču bralno - pisalnih pravic (pogoju p_3), ki ponazarja njegovo kapaciteto;
- povezavi (p_3, t_3) in (t_4, p_3) sta označeni s številom n; slednje ponazarja število povezav, ki tečejo med navedenima ponoroma in izvoroma; tako proces pisanja venomer zajame vseh n razpoložljivih pravic za dostop do podatkovne baze in jih po izvedbi venomer tudi vseh n istočasno vrača;

3.3.8 Problem medsebojnega izključevanja

Problem medsebojnega izključevanja (angl. *mutual exclusion problem*) izhaja iz obstoja *kritičnih sekcij* (angl. *critical section*) opazovanega sistema, v katerih se istočasno ne sme nahajati več zahtev. Na sliki 3.17 je prikazan graf Petrijeve mreže, ki ponazarja model dveh ločenih procesov ali kritičnih sekcij (dogajanje od vključno akcije t_1 do vključno akcije t_2) z dodeljevanjem pravice do dostopa do sekcije preko pogoja p_3. Pomeni posameznih akcij in pogojev s slike 3.17 so



Slika 3.17: Graf Petrijeve mreže za ponazoritev eliminacije možnosti istočasnega nahajanja več zahtev v kritičnih sekcijah.

sledeči:

- p_1: pred kritično sekcijo se nahaja zahteva;
- p_2: zahteva je zapustila kritično sekcijo;
- p_3: prisotnost enkratne pravice dostopa do kritične sekcije;
- t_1: dodelitev dostopne pravice do kritične sekcije iz pogoja p_3 in njen začetek; črtkana povezava med akcijama t_1 in t_2 ponazarja množico pogojev in akcij, ki se lahko izvedejo med akcijama t_1 in t_2;
- t_2: konec kritične sekcije in vračanje dostopne pravice v pogoj p_3;

Komentar modela s slike 3.17 bi bil sledeč:

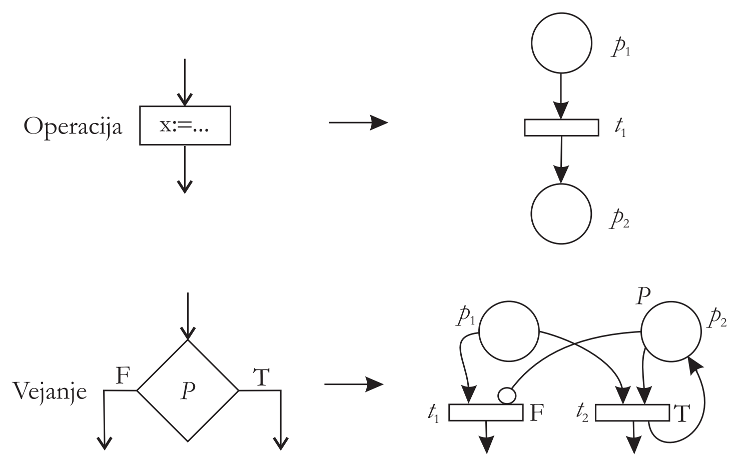
- začetna označitev mora vsebovati natanko en žeton v p_3;
- pogoj p_3 nam izvaja funkcijo *stražarja* sekcije ali funkcijo *semaforja*[6], ki dovoljuje vstop v sekcijo;

Kritične sekcije so običajno vezane na dostop do podatkov in njihovo spreminjanje ter s tem posredno vezane na problematiko branja in pisanja podatkov.

[6]Pojma *stražar* in *semafor* povzemamo po terminologiji operacijskih sistemov in sistemske programske opreme.

3.3.9 Modeliranje diagrama poteka

S Petrijevimi mrežami lahko modeliramo tudi diagrame poteka (angl. *flowchart*), ki predstavljajo eno od osnovnih metod za enolično ponazarjanje delovanja algoritmov. Osnovni konstrukti diagrama poteka so *operacije* (prirejanja) in *vejanja*. V prvih prihaja do neposrednega in posrednega prirejanja spremenljivk algoritma, v drugih pa vejimo programski tok, glede na veljavnost nekega pogoja, pri čemer je ena od poti pogojena z veljavnostjo pogoja (angl. *true* - T), druga pa z njegovo neveljavnostjo (angl. *false* - F). Na sliki 3.18 so prikazane preslikave konstruktov diagrama poteka v konstrukte Petrijevih mrež. Komentar modelov s slike 3.18 bi bil sledeč:



Slika 3.18: Preslikava gradnikov diagramov poteka (levi del slike) na osnovne konstrukte Petrijevih mrež (desni del slike).

- pretvorba operacije: operacija prirejanja se v konstruktu Petrijeve mreže izvede v akciji t_1; pogoj p_1 predstavlja zahtevo za izvajanjem prirejanja;
- pretvorba vejanja: po pretvorbi pogoj p_2 predstavlja preverjani pogoj P za izvedbo vejanja programskega toka; akcija t_1 je prvi korak na poti v primeru neveljavnosti preverjanega pogoja (F), akcija t_2 pa prvi korak na poti v primeru njegove veljavnosti (T); pogoj p_1 predstavlja zahtevo pred izvedbo vejanja;

3.3.10 Modeliranje delovanja končnega avtomata

Zapis kakršnegakoli končnega avtomata (angl. *finite state machine*) temelji na osnovi obstoja treh nepraznih množic X, Q in Y. Termin „*končnosti*“ avtomata določa končnost vseh treh navedenih množic. Množica X predstavlja nabor različnih možnih vhodnih črk avtomata, množica Q nabor različnih možnih notranjih stanj avtomata in množica Y nabor različnih možnih izhodnih črk avtomata. Formalno tako končni avtomat po viru [12] zapišemo z izrazom

$$A = (X, Q, Y, \delta, \lambda). \tag{3.32}$$

δ predstavlja funkcijo za tvorbo novega stanja avtomata na naslednjem diskretnem časovnem koraku, kar zapišemo z izrazoma

$$\delta : X \times Q \to Q, \tag{3.33}$$

$$D^1 q = \delta(x, q),\ x \in X,\ q \in Q, \tag{3.34}$$

pri čemer zapis $D^1 q$ predstavlja novo stanje avtomata, λ pa predstavlja funkcijo za tvorbo izhodne črke avtomata. Poznamo dve vrsti končnih avtomatov in sicer Mealyjeve in Mooreove končne avtomate. V primeru Mealyjevega končnega avtomata funkcijo λ ponazorimo z izrazoma

$$\lambda : X \times Q \to Y, \tag{3.35}$$

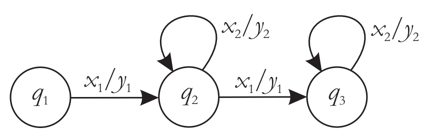
$$y = \lambda(x, q),\ x \in X,\ q \in Q,\ y \in Y, \tag{3.36}$$

v primeru Moorovega avtomata pa z izrazoma

$$\lambda : Q \to Y, \tag{3.37}$$

$$y = \lambda(q),\ q \in Q,\ y \in Y. \tag{3.38}$$

Predpostavimo, da imamo opravka z enostavnim Mealyjevim končnim avtomatom, predstavljenim z njegovim diagramom prehajanja stanj na sliki 3.19. Predstavljeni avtomat formalno zapišemo z izrazom



Slika 3.19: Grafična ponazoritev diagrama prehajanja stanj vzorčnega končnega avtomata.

$$X = \{x_1, x_2\},\ Q = \{q_1, q_2, q_3\}, Y = \{y_1, y_2\} \tag{3.39}$$

in s prehajalno tabelo 3.1.

Pravila za formiranje modela končnega Mealyjevega avtomata z ekvivalentno Petrijevo mrežo lahko strnemo v naslednje alinee:

- črke vhodne abecede se preslikajo v pogoje; za naš primer tako velja sklep $X = \{x_1, x_2\} \Rightarrow P_1 = \{p_{x1}, p_{x2}\}$;
- stanja avtomata se preslikajo v pogoje; za naš primer tako velja sklep $Q = \{q_1, q_2, q_3\} \Rightarrow P_2 = \{p_{q1}, p_{q2}, p_{q3}\}$;

X/Q	q_1	q_2	q_3
x_1	q_2/y_1	q_3/y_1	-
x_2	-	q_2/y_2	q_3/y_2

Tabela 3.1: Tabela prehajanj med stanji končnega avtomata, prikazanega na sliki 3.19. Prva črka na posamezni povezavi predstavlja vhodno, druga pa izhodno črko.

- prehajanja med stanji avtomata, ki istočasno vršijo formacijo izhodnih črk, se preslikajo v akcije; za naš primer tako formiramo množico $T_1 = \{t_1, t_2, t_3, t_4\}$, pri čemer so posamezne akcije bodoče Petrijeve mreže povezane s funkcijama avtomata δ in λ, kar simbolično zapišemo z izrazoma

$$t_1 = (\delta(x_1, q_1), \lambda(x_1, q_1)),\ t_2 = (\delta(x_1, q_2), \lambda(x_1, q_2)), \tag{3.40}$$

$$t_3 = (\delta(x_2, q_2), \lambda(x_2, q_2)),\ t_4 = (\delta(x_2, q_3), \lambda(x_2, q_3)); \tag{3.41}$$

- ker se formacija črke izvede že v akciji, ki izraža posamezno prehajanje v avtomatu, moramo za vsako črko izhodne abecede pripraviti le pogoj, v katerega se bodo odlagali žetoni, kar bo ponazarjalo tvorbo izhodne črke; za naš primer tako velja sklep $Y = \{y_1, y_2\} \Rightarrow P_3 = \{p_{y1}, p_{y2}\}$;

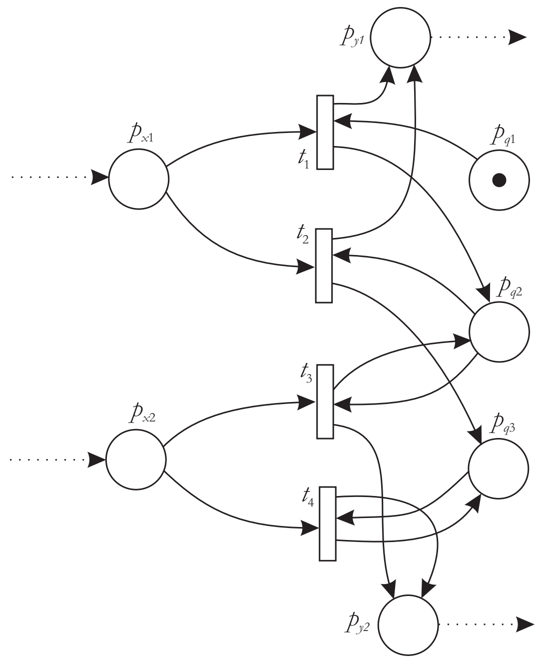
- formira se dokončna množica pogojev P ($P = P_1 \cup P_2 \cup P_3$); za naš primer tako velja izraz

$$P = \{p_{x1}, p_{x2}, p_{q1}, p_{q2}, p_{q3}, p_{y1}, p_{y2}\}\,,\ T = \{t_1, t_2, t_3, t_4\}\,; \tag{3.42}$$

Na sliki 3.20 je predstavljen model grafa Petrijeve mreže, ki ponazarja delovanje končnega avtomata s slike 3.19. Formalizacijo tvorbe matrik I in O Petrijeve mreže prepuščamo bralcu, pogojena pa je s potrebnimi pogoji za spreminjanje stanja avtomata in novimi stanji, ki jih avtomat dosega. Z vidika izvajanja mreže je pomembno, da v enega od pogojev v mreži, ki ponazarjajo stanja avtomata (p_{q1}, p_{q2}, p_{q3}), vstavimo na začetku natanko en žeton, s čimer inicializiramo začetno stanje modeliranega avtomata. Vstopajoči črtkani povezavi v p_{x1} in p_{x2} simbolizirata prihajanje vhodnih črk iz zunanjega sveta, izstopajoči črtkani povezavi iz p_{y1} in p_{y2} pa simbolizirata odhajanje izhodnih črk proti zunanjemu svetu.

3.3.11 Petrijeve mreže kot generatorji jezikov

V predhodnih razdelkih smo bili pozorni predvsem na pogoje, označitve in zaporedja označitev. V pričujoče razdelku bomo pozornost preusmerili na zaporedje proženih akcij, ki okarakterizirajo modelirani sistem. V kontekstu *jezikov Petrijevih mrež* posamezna sprožena akcija generira nek *znak* iz abecede, zaporedje sproženih akcij *besedo* tvorjeno na osnovi abecede, vsa možna zaporedja proženih



Slika 3.20: Pretvorba končnega avtomata s slike 3.19 v graf Petrijeve mreže.

akcij pa *jezik* opazovane Petrijeve mreže. Ob tem smo že predhodno predpostavili, da se dve akciji ne moreta sprožiti istočasno. V nadaljevanju zapišimo formalno definicijo jezika Petrijeve mreže.

Definicija 10 *L je jezik Petrijeve mreže, če obstajajo Petrijeva mreža C=(P,T,I,O), preslikava $\rho : T \to \Sigma$, začetna označitev $o(k_0)$ in množica končnih označitev F, tako da velja izraz*

$$L = \{\rho(\beta) \in \Sigma^* : \beta \in T^* \textit{ and } \delta(o(k_0), \beta) \in F\}. \tag{3.43}$$

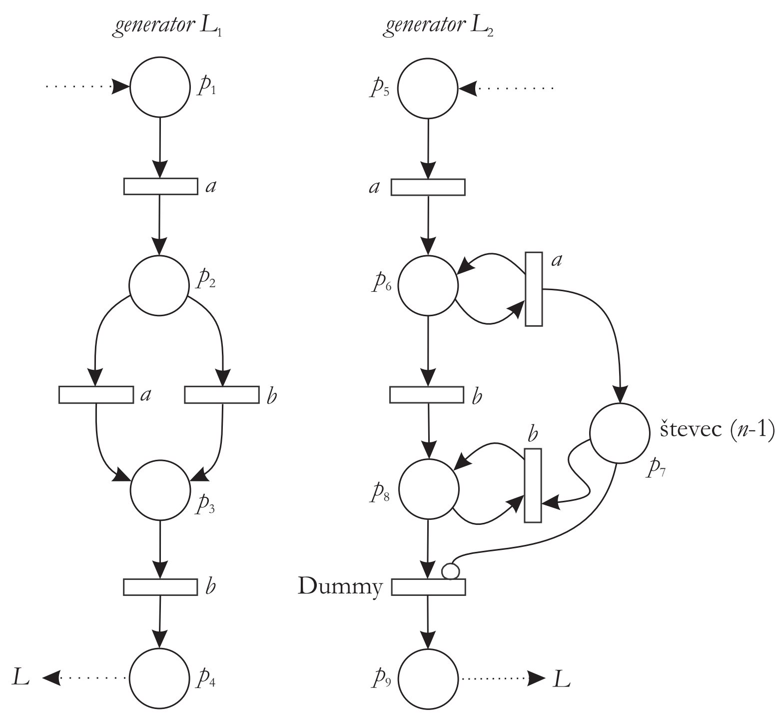
Pri tem Σ predstavlja množico vseh črk nekega jezika, β poljubno zaporedje izvedenih akcij, Σ^* množico vseh možnih končnih tvorb besed glede na množico črk jezika, T^* množico vseh možnih zaporedij akcij, F množico vseh končnih označitev Petrijeve mreže, δ pa preslikovalno funkcijo, ki na osnovi začetne označitve $o(k_0)$ in zaporedja akcij β formira neko označitev. $\rho(\beta)$ tako predstavlja besedo tvorjeno na osnovi zaporedja akcij (zaporedja črk), pri čemer po tvorbi te besede v Petrijevi mreži ni omogočenih več nobenih akcij. Lingvistično bi tako izraz (3.43) lahko zapisali na sledeč način: „*Preslikava zaporedja akcij v domeno zaporedja črk (torej besedo) predstavlja besedo jezika opazovane Petrijeve mreže,* *(i) če je zaporedje akcij v opazovani Petrijevi mreži možno izvesti in če (ii) je označitev, ki jo dosežemo z izvedbo zaporedja akcij končna označitev opazovane Petrijeve mreže*". Velja, da sta dve Petrijevi mreži *ekvivalentni* natanko takrat, ko imata enaka jezika.

Predpostavimo, da imamo opravka z jezikom L, ki ga formalno zapišemo kot unijo dveh jezikov z izrazom

$$L = L_1 \cup L_2,\ L_1 = a(a \vee b)b,\ L_2 = a^n b^n,\ n \geq 1, \tag{3.44}$$

pri tem pa si zastavimo nalogo, da poiščemo graf Petrijeve mreže, ki modelira *generator besed* tovrstnega jezika. Rešitev bomo iskali parcialno in sicer se bomo najprej polotili iskanja generatorjev za jezika L_1 in L_2. Pri tem bomo predpostavljali, da posamezno črko besede jezika generira posamezna akcija v Petrijevi mreži.

Generator jezika L_1 je prikazan na levem delu slike 3.21. Vanj preko črtkane



Slika 3.21: Grafična ponazoritev generatorjev jezika $L_1 = a(a \vee b)b$ (levo) in $L_2 = a^n b^n$ (desno) s Petrijevimi mrežami.

povezave iz zunanjega sveta v pogoj p_1 vstopajo zahteve za generiranje posamezne besede jezika L_1. Po vstopu zahteve se hipno generira črka a, pri čemer na sliki akcij ne označujemo več z njihovimi indeksi, temveč z generiranimi črkami. Po generiranju črke a zahteva v pogoju p_2 nedeterministično izbira med proženjem akcije a ali b. Ko je ena od obeh izvedena (se generira črka a ali b), zahteva preko pogoja p_3 sproži še izvedbo zadnje akcije, ki generira še zadnjo črko besede b. Zahteva je tako dokončno postrežena in preko pogoja p_4 po črtkani povezavi zapusti generator jezika L_1. Tako pridobljeno zaporedje izvedenih akcij ponazarja tvorjeno besedo jezika. Pomeni posameznih pogojev in akcij na levem delu slike 3.21 so sledeči:

- p_1: v generator je vstopila zahteva za generiranje besede jezika L_1;
- p_2: zahteva nedeterministično izbira med generiranjem črke a ali b;
- p_3: zahteva je pred generiranjem zadnje črke v besedi in sicer je to črka b;
- p_4: beseda jezika L_1 je bila uspešno generirana;
- a: generira se črka a;
- b: generira se črka b;

Generator jezika L_2 je prikazan na desnem delu slike 3.21. Vanj preko črtkane povezave iz zunanjega sveta v pogoj p_5 vstopajo zahteve za generiranje posamezne besede jezika L_2. Po vstopu zahteve se hipno generira črka a. Po generiranju črke a zahteva v pogoju p_6 nedeterministično izbira med proženjem akcije a ali b. Izbiro proženja akcije a lahko ponovi večkrat, saj se žeton po proženju akcije a dosegljive iz pogoja p_6 venomer vrne v ta pogoj. Omenjena akcija a poleg vračanja žetona v p_6 ob vsaki svoji izvedbi en žeton odloži v pogoj p_7. Ob predpostavki, da je ta pogoj na začetku prazen, lahko smatramo, da ta pogoj predstavlja števec ponovitev nedeteriministično izbranega generiranja črke a. Ker se akcija a dosegljiva iz pogoja p_6 lahko izvede poljubnokrat, lahko pa tudi sploh ne, naredimo zaključek, da se nam v opazovanem pogoju p_7 skozi končno število ponovitev akcije a shrani vrednost števca $n-1$ ($n \geq 1$), ki odraža to število ponovitev. Pri tem še enkrat izrecno poudarimo, da je število ponovitev izvajanja akcije a dosegljive iz p_6 določeno nedeterministično. Ko po $n-1$ ponovitvah generiranja črke a zahteva preneha z generiranjem a-jev, preide žeton iz pogoja p_6 preko akcije, ki generira natanko eno črko b, v pogoj p_8. Odtod se ob lahko ob izpolnjenosti pogoja p_7 lahko izvaja le akcija b, ki ob vsaki svoji izvedbi dekrementira vrednost števca $n-1$ (zmanjša število žetonov v p_7 za en žeton). Ko se pogoj p_7 izprazni (iz njega smo počrpali preko izvajanja akcije b vseh $n-1$ žetonov), se akcija b ne more več izvesti, omogočena pa zaradi inhibirne povezave postane akcija `Dummy`. Slednja ne generira črke, temveč zahtevo zgolj prenese v pogoj p_9, kar pomeni, da je zahteva v celoti postrežena. Zahteva preko pogoja p_9 po črtkani povezavi zapusti generator jezika L_2. Tako pridobljeno zaporedje izvedenih akcij ponazarja tvorjeno besedo jezika L_2. Pomeni posameznih pogojev in akcij na desnem delu slike 3.21 so sledeči:

- p_5: v generator jezika je vstopila zahteva za generiranje besede jezika L_2;
- p_6: zahteva nedeterministično izbira med generiranjem črke a ali b;
- p_7: pomnjenje števila $(n-1)$ generiranj črke a;

- p_8: zahteva deterministično izbira akcijo b, vse dokler pogoj p_7 ni prazen;
- p_9: beseda jezika L_2 je bila uspešno generirana,
- `Dummy`: akcija nima zmožnosti generiranja črke; izvede se pod pogojem, da smo iz pogoja p_7 preko izvajanja akcije b počrpali vse žetone;
- a: generira se črko a;
- b: generira se črko b;

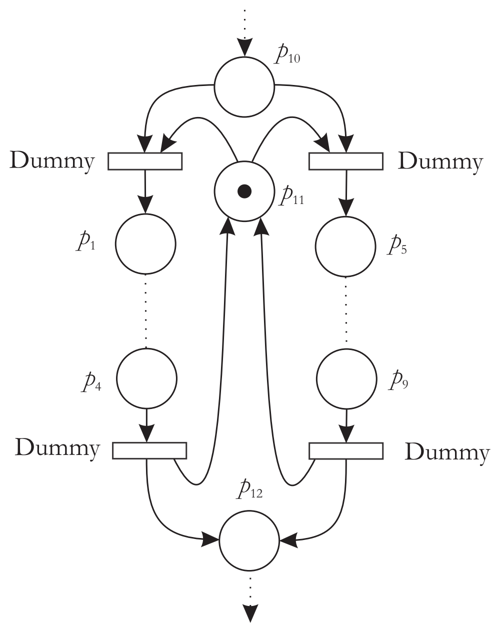
Ob koncu opisa obeh generatorjev navedimo nekaj splošnih ugotovitev:

- za oba generatorja s slike 3.21 velja, da morajo biti vsi pogoji ob začetku simulacije inicializirani na nično število žetonov; v primeru, da temu ni tako, lahko pride do porajanja zaporedja akcij, ki ne bo predstavljalo besede prvega ali drugega jezika;
- generiranje besed jezika L_1 je nedeterministično v izbiri druge črke (a ali b) besede, generiranje besed jezika L_2 pa je nedeterministično v izbiri števila n, ki definira število ponovitev a-jev in s tem posredno število b-jev v besedi; točki nedeterministične izbire se nahajata v pogojih p_2 in p_6;
- mehanizem za izvedbo nedeterministične odločitve je v Petrijevih mrežah inherentno[7] prisoten;

V nadaljevanju iskanja modela generatorja besed jezika L moramo združiti obe predhodno opisani rešitvi s slike 3.21. Njuna združitev je prikazana na sliki 3.22. V generator jezika L preko črtkane povezave iz zunanjega sveta v pogoj p_{10} vstopajo zahteve za generiranje posamezne besede jezika. Po vstopu zahteve se v pogoju p_{10} nedeterministično izbere vrsto generirane besede (besedo jezika L_1 ali L_2), pri čemer je možnost izbire generatorja in vstop vanj pogojen s prisotnostjo žetona v pogoju p_{11}, ki ima funkcijo čuvaja vstopa v posamičen generator. S tem onemogočimo večkratno izvajanje enega ali drugega generatorja, kar bi onemogočilo pravilno tvorbo besed ali ustrezno zaporedje izvajanja akcij. Odtod zahteva potuje po poti, ki se začne bodisi v pogoju p_1 (generira se beseda jezika L_1) ali v pogoju p_5 (generira se beseda jezika L_2), ki smo ju že spoznali. Zahteva se po infinitezimalnem času pojavi v pogoju p_{12}, preko njega pa kot v celoti postrežena (zgenerirala se je beseda jezika L_1 ali L_2) zapusti generator, istočasno pa se čuvaja vrne v izhodiščni pogoj p_{11}, kar omogoči, da v generator v spremstvu čuvaja lahko vstopi nova zahteva. Pomeni posameznih pogojev in akcij na sliki 3.22 so sledeči:

- p_{10}: v generator L je vstopila zahteva za generiranje besede jezika L in pride do nedeterminističnega izbiranja tipa besede (beseda jezika L_1 ali L_2);
- p_{11}: pogoj predstavlja prisotnost čuvaja, ki preprečuje nahajanje več zahtev v notranjosti generatorja L;
- p_{12}: zahteva je bila uspešno dokončno postrežena;

[7] Inherentno: neločljivo, nerazdružno povezano s čim (Vir: SSKJ).



Slika 3.22: Grafična ponazoritev generatorja jezika $L_1 \cup L_2$ s Petrijevo mrežo.

3.4 Analiza Petrijevih mrež

Modeliranje in simulacije realnih sistemov s Petrijevimi mrežami nas vodita do možnosti analize njihove dinamike. Predhodno smo že spoznali metodo analize *drevesa označitev* in definirali pojem *dosegljivosti stanj* v Petrijevih mrežah. V pričujočem razdelku si bomo ogledali še značilnosti *varnosti*, *omejenosti* in *konservativnosti* v Petrijevih mrežah.

3.4.1 Varnost Petrijeve mreže

Ena od možnih značilnosti opazovane Petrijeve mreže je njena *varnost* (angl. *safeness*). Posamezen pogoj v mreži je *varen* natanko takrat, če se skozi simulacijo dinamike v opazovani Petrijevi mreži število žetonov v tem pogoju nikdar ne dvigne nad 1. Petrijeva mreža je kot celota *varna* natanko takrat, ko so v njej varni tudi vsi pogoji. Jezikovni opis pojma varnosti zapišemo s formalno definicijo v nadaljevanju, pri čemer se v definiciji zaradi preglednosti izognemo indeksu časovnega koraka označitve k, ki je implicitno vsebovan v sklicu na dosegljivost označitve $R(C, o)$. Slednje pomeni, da ima označitev o' večji časovni indeks kot začetna označitev o. o'_i (enako velja za o_i) pri tem predstavlja sklic na i-to komponento vektorja označitve o' (o) po vzoru izraza (3.8).

Definicija 11 *Posamezni pogoj* $p_i \in P$ *je v Petrijevi mreže* C=(P,T,I,O) *z začetno označitvijo* o *varen natanko takrat, ko za vse označitve* $o' \in R(C, o)$ *velja* $o'_i \leq 1$. *Velja tudi* $o_i \leq 1$. *Petrijeva mreža* C *je varna natanko takrat, ko so varni vsi njeni pogoji.*

Analiza varnosti opazovane Petrijeve mreže se uporablja predvsem na področjih modeliranja logičnih enačb, logičnih preklopnih struktur in podobnih sistemov, ki lahko zasedajo le dve možni stanji. Na ta način posamezni pogoj v Petrijevi mreži obravnavamo kot dvovrednostni logični pogoj. Tovrstna analiza je zanimiva tudi za analizo dinamike v kritičnih sekcijah, v katerih si ne želimo več paralelnih zahtev, ki v obliki žetonov potujejo skoznje, ter tudi iz vidika pomnjenja, saj nam zniža število možnih stanj sistema in s tem poceni realizacijo modela v obliki strojne realizacije.

V primeru, da opazovani pogoj p_i v Petrijevi mreži ni varen in nima večkratnih vhodnih ter večkratnih izhodnih povezav, lahko tudi ta pogoj spremenimo v varen pogoj. To dosežemo tako, da vpeljemo nov pogoj p'_i. Postopek vpeljave novega pogoja zapišemo s praviloma

$$if\ (p_i \in I(t_j))\ and\ (p_i \notin O(t_j))\ then\ add\ p'_i\ to\ O(t_j), \tag{3.45}$$

$$if\ (p_i \in O(t_j))\ and\ (p_i \notin I(t_j))\ then\ add\ p'_i\ to\ I(t_j), \tag{3.46}$$

njun pomen pa je sledeč:

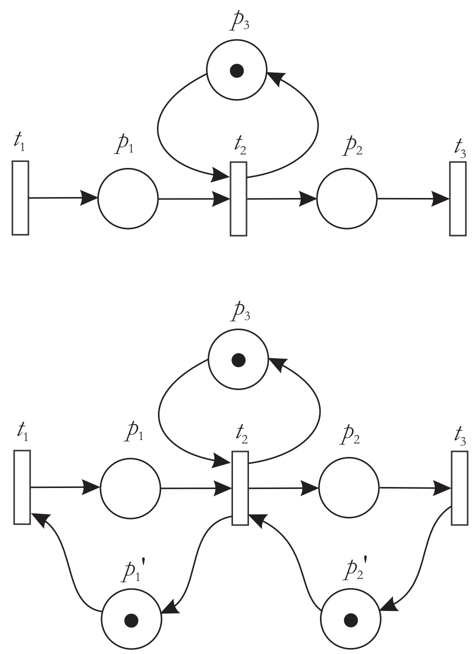
- uporaba izraza (3.45): če je pogoj p_i, ki ni varen, vhodni pogoj za akcijo t_j, ni pa njen izhodni pogoj, vpeljemo nov pogoj p'_i, ki predstavlja izhodni pogoj za akcijo t_j;

- uporaba izraza (3.46): če je pogoj p_i, ki ni varen, izhodni pogoj za akcijo t_j, ni pa njen vhodni pogoj, vpeljemo nov pogoj p'_i, ki predstavlja vhodni pogoj za akcijo t_j;

Pogoj p'_i je *komplementaren* pogoju p_i, pomensko pa predstavlja izjavo, da je prvotno opazovani pogoj p_i neizpolnjen (v p_i ni žetonov). Po vpeljavi novega pogoja moramo poskrbeti tudi za spremembo začetne označitve, ki mora vsebovati žeton bodisi v p_i, bodisi v p'_i. Na sliki 3.23 je prikazan primer pretvorbe ne-varnih pogojev p_1 in p_2 v varna pogoja.

3.4.2 Omejenost Petrijeve mreže

Omejenost (angl. *boundedness*) je splošnejša oblika značilnosti varnosti. Petrijeva mreža je k-omejena natanko takrat, ko je k največje število žetonov, do katerega pride v enem od pogojev v njenem drevesu označitev. Iz značilnosti k-omejenosti Petrijeve mreže tako lahko sklepamo, da skozi dinamiko Petrijeve mreže število žetonov v nobenem pogoju ne bo preseglo števila k, bo pa vsaj



Slika 3.23: Primer pretvorbe ne-varnih pogojev p_1 in p_2 (zgoraj) v varna pogoja (spodaj) z dodajanjem dveh novih komplementarnih pogojev p_1' in p_2' .

v enem od pogojev doseženo. Jezikovni opis pojma k-omejenosti zapišemo s formalno definicijo v nadaljevanju.

Definicija 12 *Posamezni pogoj p_i ($p_i \in P$) Petrijeve mreže C=(P,T,I,O) z začetno označitvijo o je k-omejen, če za vse $o' \in R(C, o)$ velja $o_i' \leq k$. Petrijeva mreža C je k-omejena, če je vrednost k maksimalna gledano preko vseh omejenostih pogojev.*

Lastnost k-omejenosti ($k < \infty$) ali maksimalnega možnega števila porajanih žetonov v posameznem pogoju Petrijeve mreže je pogoj za končno in efektivno realizacijo modeliranega sistema. Še več, največkrat si želimo, da je število k čim manjše, s čimer dosežemo manjše število različnih možnih stanj (označitev) modeliranega sistema.

3.4.3 Konservativnost Petrijeve mreže

Pri uporabi Petrijevih mrež za namene modeliranja nam pogoji mnogokrat predstavljajo neke *resurse*, ki morajo biti razpoložljivi za izvedbo akcij. Resurse delimo na skupini *porabljivih* (takšnih, ki jih s časom porabimo) in *trajnih* resursov. Slednjih skozi dinamiko mreže ne moremo porabiti in ostaja njihovo število skozi čas konstantno. S tega zornega kota nas v Petrijevi mreži ali v kakšnem od njenih segmentov mnogokrat zanima obstoj lastnosti *konservativnosti* (angl. *conservative*). Opazovana Petrijeva mreža je konservativna natanko takrat, ko število žetonov v pogojih Petrijeve mreže skozi čas (skozi zaporedje izvajanja akcij) ostaja enako. Slednje velja, ko je vsota žetonov v vsaki označitvi iz drevesa označitev konstantna. Pojem *striktne konservativnosti* obrazložimo s formalno definicijo v nadaljevanju.

Definicija 13 *Petrijeva mreža C=(P,T,I,O) z začetno označitvijo o je striktno konservativna, če za vse $o' \in R(C,o)$ velja izraz*

$$\sum_{i=1}^{|P|} o'_i = \sum_{i=1}^{|P|} o_i. \tag{3.47}$$

Izraz (3.47) posredno vključuje tudi veljavnost izraza

$$\forall j : |I(t_j)| = |O(t_j)|\,,\ j = 1, ..., m,\ m = |T|\,, \tag{3.48}$$

ki za striktno konservativnost zahteva enakost števila vstopajočih in izstopajočih povezav za vsako akcijo v Petrijevi mreži. Že sam obstoj pogoja s pomenom števca bi tako onemogočil lastnost striktne konservativnosti.

Lastnost striktne konservativnosti glasi po definiciji na celotno opazovano Petrijevo mrežo, v praksi pa nas običajno zanima nespremenljivost števila žetonov skozi časovno dinamiko zgolj v določeni podmnožici pogojev opazovane Petrijeve mreže. V takšnih primerih s primerno formalno obravnavo eliminiramo pogoje, katerih število žetonov ni v obsegu opazovanja in v njej ohranimo samo tiste pogoje, v katerih želimo skozi čas imeti konstantno vsoto števila žetonov. Preverjano lastnost poimenujemo za *konservativnost*, obrazložimo pa jo s formalno definicijo v nadaljevanju.

Definicija 14 *Petrijeva mreža C=(P,T,I,O) z začetno označitvijo o je konservativna glede na utežni vektor w predstavljen v izrazu*

$$w = (w_1, w_2, ..., w_n),\ n = |P|\,,\ w_i \in \{0, 1\}, \tag{3.49}$$

če za vse $o' \in R(C,o)$ velja izraz

$$\sum_{i=1}^{|P|} o'_i * w_i = \sum_{i=1}^{|P|} o_i * w_i. \tag{3.50}$$

Petrijeva mreža je tako striktno konservativna le ob upoštevanju utežnega vektorja $w = (1, 1, ..., 1)$, poljubna mreža pa venomer konservativna z utežnim vektorjem $w = (0, 0, ..., 0)$. Zaradi slednjega pogoj za konservativnost zaostrimo in sicer zahtevamo, da je vektor w neničeln ($\exists i : w_i = 1$).

3.5 Zgleda modeliranja s področja računalniških omrežij

V pričujočem razdelku si bomo ogledali še dva specifičnejša zgleda modeliranja s Petrijevimi mrežami, ki sodita na področje računalniških omrežij.

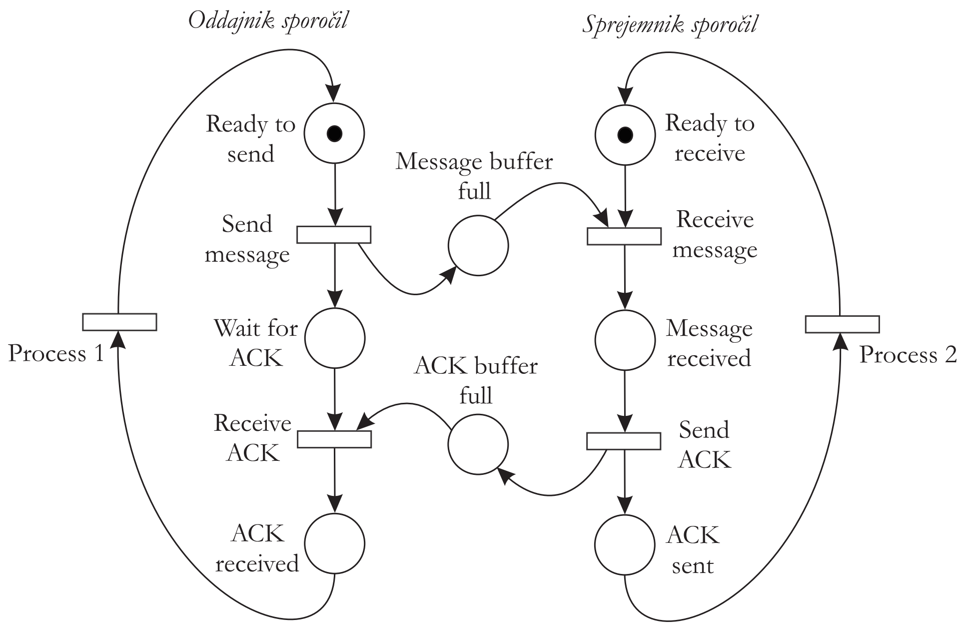
3.5.1 Poenostavljen model protokola med oddajnikom in sprejemnikom

Predpostavimo, da imamo opravka z enostavnim protokolom za izmenjavo sporočil med *oddajnikom* in *sprejemnikom*. Prvi posamezno sporočila zgenerira, odpošlje, počaka na potrditev prejema oddanega sporočila in gre nato v generiranje novega sporočila, drugi pa posamezno sporočilo sprejme, potrdi njegov prejem in gre nato v sprejemanje novega sporočila. Model tovrstnega sistema ponazorjen s Petrijevo mrežo vključno z začetno označitvijo povzet po viru [13] je predstavljen na sliki 3.24. Levi del modela predstavlja oddajnik, desni del modela sprejemnik, osrednji del modela (pogoja `Message buffer full` in `ACK buffer full`) pa vmesnika za sporočila in njihove potrditve. Začetna označitev nam pove, da sta tako oddajnik (žeton v pogoju `Ready to send`), kot tudi sprejemnik (žeton v pogoju `Ready to receive`) na začetku inicializirana na svoji stanji pripravljenosti, oba vmesnika pa sta prazna.

Glede na začetno označitev lahko graf Petrijeve mreže s slike 3.24 komentiramo s sledečimi alineami:

- *oddajnik*: oddajnik iz stanja pripravljenosti (žeton v pogoju `Ready to send`) preko akcije `Send message` odpošlje predhodno pripravljeno sporočilo (odlaganje žetona v pogoj `Message buffer full`) in preide v stanje čakanja na potrditev (odlaganje žetona v pogoj `Wait for ACK`); vse do prejema potrditve (porajanja žetona v pogoju `ACK buffer full`) oddajnik ne bo izvedel nobene akcije; ob prejemu potrditve bo oddajnik preko pogoja `ACK received` izvedel akcijo `Process 1`, v kateri bo zgeneriral novo sporočilo za oddajo in prešel v stanje pripravljenosti za oddajo novega sporočila (odlaganje žetona v pogoj `Ready to send`); glede na začetno označitev pridemo do ugotovitve, da je oddajniški del modela varen (v vsakem od njegovih pogojev je lahko največ en žeton) in konservativen (število žetonov v oddajniškem delu je ves čas enako 1);

- *sprejemnik*: sprejemnik iz stanja pripravljenosti (žeton v pogoju `Ready to receive`) preide preko akcije `Receive message` v pogoj `Message received` samo pod pogojem, da iz vmesnika (prisotnost žetona v pogoju `Message`



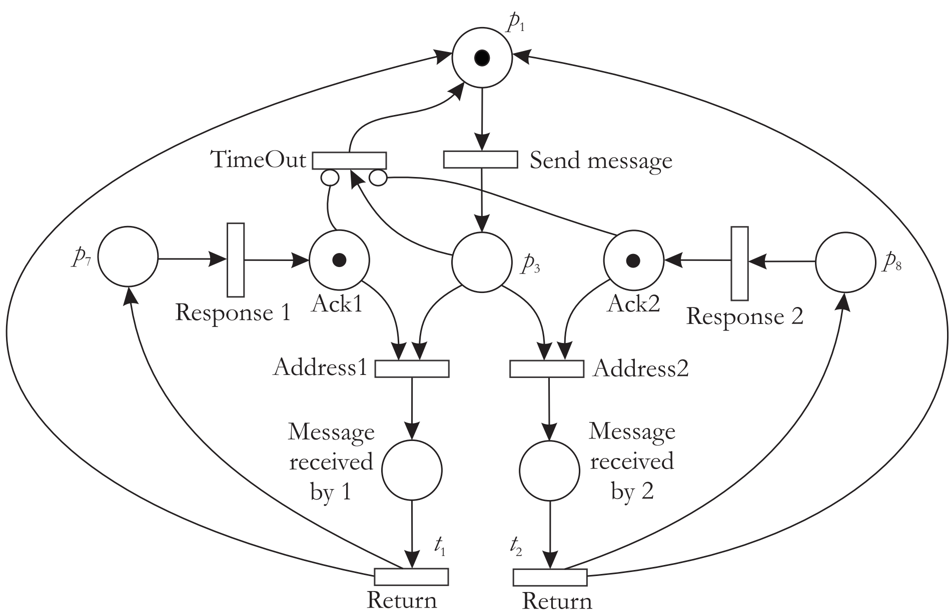
Slika 3.24: Graf Petrijeve mreže poenostavljenega modela protokola izmenjave sporočil s potrjevanjem med oddajnikom in sprejemnikom povzet po viru [13].

`buffer full`) lahko prevzame sporočilo poslano s strani oddajnika; akcija `Receive message` tako predstavlja prevzem sporočila iz vmesnika in njegovo sprejetje; po dospetju žetona v pogoj `Message received` se izvede akcija `Send ACK`, ki odpošlje potrditveno sporočilo proti oddajniku (odlaganje žetona v pogoj `ACK buffer full`) in istočasno preko pogoja `ACK sent` in akcije `Process 2` (npr. shrambe sporočila) sprejemnik preide v stanje pripravljenosti za sprejem novega sporočila (žeton v pogoju `Ready to receive`); tudi v primeru modela sprejemnega dela glede na njegovo začetno označitev ugotovimo, da je model varen in konservativen (število žetonov v sprejemniškem delu je ves čas enako 1);

- glede na začetno označitev sta pogoja `Message buffer full` in `ACK buffer full` varna, saj se v vsakem od njiju lahko nahaja največ en žeton;

3.5.2 Model nedeterministične izbire naslovnika sporočila

Predpostavimo, da imamo opravka s sistemom za pošiljanje sporočil. Posamezno odpošiljanje se izvede na osnovi nedeterministične izbire enega od dveh različnih možnih naslovnikov. Predpogoj za izbiro posameznega naslovnika je prejetje potrditve njemu predhodno poslanega sporočila. V primeru, da se paketa ne da poslati nobenemu od obeh naslovnikov, sistem preide v reinicializacijo odpošiljanja sporočila. Model opisanega sistema ponazorjen s Petrijevo mrežo ter povzet po viru [13] je vključno z začetno označitvijo predstavljen na sliki 3.25.



Slika 3.25: Model nedeterministične izbire naslovnika v procesu odpošiljanja sporočil s potrjevanjem povzet po viru [13].

Glede na začetno označitev lahko graf Petrijeve mreže s slike 3.25 komentiramo s sledečimi alineami:

- preko pogoja p_1 in akcije `Send message` pride do inicializacije odpošiljanja sporočila (odlaganja žetona v pogoj p_3);
- v pogoju p_3 pride do nedeterministične izbire naslovnika (izvedbe akcije `Address1` ali `Address2`); če v pogojih `Ack1` ali `Ack2` ni žetona (predhodno poslano sporočilo s strani naslovnika ni bilo potrjeno), do izbire naslovnika ne pride, temveč preidemo preko akcije `TimeOut` v reinicializacijo odpošiljanja istega sporočila (odlaganja žetona v pogoj p_1);
- zaporedje `Address1` - `Message received by 1` - `Return` - p_7 - `Response1` predstavlja potovanje potrditve s strani prvega naslovnika do pogoja `Ack1`, zaporedje `Address2` - `Message received by 2` - `Return` - p_8 - `Response2` pa predstavlja potovanje potrditve s strani drugega naslovnika do pogoja `Ack2`;
- opazovana Petrijeva mreža ni striktno konservativna (ob podani začetni označitvi celotno število žetonov v mreži skozi čas simulacije variira med

številoma 2 in 3); istočasno lahko ugotovimo, da je predstavljena Petrijeva mreža varna;

3.6 Razširitve Petrijevih mrež

V predhodnih razdelkih smo si ogledali množico primerov zgledov Petrijevih mrež. Nekateri zgledi so temeljili na klasični definiciji Petrijevih mrež, nekateri pa v to skupino niso sodili. V skupino slednjih sodijo vsi zgledi, v katerih smo uporabljali *inhibirne povezave*. Tovrstni konstrukti niso definirani za skupino klasičnih Petrijevih mrež, tako da zglede, v katerih nastopajo inhibirne povezave smatramo za zglede ponazorjene z *razširjenimi Petrijevimi mrežami*.

Poleg inhibirnih povezav med možne konstrukte razširjenih Petrijevih mrež sodijo sledeči dejavniki:

- *stohastičnost*: v tem primeru dopuščamo verjetnostno pogojenost vejanj ali izbire izvajanja akcij, s čimer pridemo do stohastičnih Petrijevih mrež (angl. *stochastic Petri nets*);

- *mehkost*: v tem primeru dopuščamo mehkost izvajanja akcij ali izpolnjenosti pogojev, s čimer pridemo do *mehkih Petrijevih mrež*; pod pojmom mehkosti izvajanja akcij imamo v mislih neko delno izvajanje akcije, pod pogojem mehkosti izpolnjenosti pogojev pa neko delno ali približno izpolnjenost pogoja;

- *časovnost*: v tem primeru je omogočeno, da posameznim akcijam pripišemo časovnost njihovega trajanja; tovrstne Petrijeve mreže imenujemo za *časovne Petrijeve mreže* (angl. *timed Petri nets*), več o njih pa povemo v naslednjem razdelku pričujočega poglavja;

- *„barvanje“ žetonov*: v klasičnih Petrijevih mrežah informacijska vrednost žetona ponazarja le kratnost izpolnjenosti pogoja; v primeru *„barvanja žetonov“* je možno posamezni žeton deklarirati z množico spremenljivk, implementacijo žetona pa inicializirati s prirejanjem vrednosti; tovrstne Petrijeve mreže imenujemo za *barvne Petrijeve mreže* (angl. *coloured Petri nets*), več o njih pa povemo v naslednjem poglavju;

3.7 Časovne Petrijeve mreže

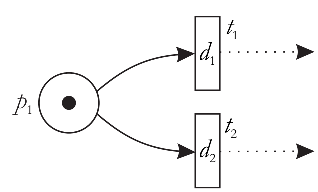
Časovne Petrijeve mreže nam omogočijo vpeljavo *časa trajanja* posameznih akcij. Kot zalogo vrednosti za trajanje posameznih akcij privzamemo množic naravnih števil upoštevajoč tudi ničlo. Definicijo časovnih Petrijevih mrež, povzeto po viru [14], zapišemo v nadaljevanju.

Definicija 15 *Šestorček* $PN = (P, T, I, O, o(k_0), D)$ *imenujemo za časovno Petrijevo mrežo, kjer* P *predstavlja končno množico pogojev,* T *končno množico akcij,* I *vhodno ter* O *izhodno matriko in* $o(k_0)$ *začetno označitev.* D *predstavlja vektor nenegativnih števil vključujoč ničlo, ki posameznim akcijam določajo časovno trajanje, pri čemer velja izraz*

$$D = (d_1, d_2, ..., d_m),\ m = |T|\,,\ \forall i : d_i \in \mathbb{N}_0. \tag{3.51}$$

Trajanje posamezne akcije si interpretiramo na sledeč način. Opazovana akcija t_j s trajanjem d_j časovnih enot se sproži (se začne izvajati), ko so izpolnjeni vsi pogoji, iz katerih vodijo vhodne povezave v opazovano akcijo. Omenjeni pogoji morajo biti izpolnjeni vse do konca trajanja opazovane akcije. Šele po d_j časovnih enotah se žetoni iz vhodnih pogojev hipno prenesejo na izhodne pogoje opazovane akcije. Povedano drugače, žetoni v vhodnih pogojih ob sprožitvi akcije t_j ostanejo na svojih mestih d_j časovnih enot, šele nato pa se prenesejo preko akcije t_j v izhodne pogoje na nam že znani način. V primeru, da se vsaj en potreben žeton v vhodnih pogojih akcije t_j skozi čas trajanja te akcije odvzame s strani neke druge akcije, pride do prekinitve izvajanja akcije t_j. Slednja preide v fazo čakanja na novo izpolnjenost potrebnih pogojev za njeno proženje in s tem posredno na svoje vnovično celotno izvajanje s časom d_j časovnih enot.

Na sliki 3.26 je prikazan enostaven primer grafa Petrijeve mreže z akcijo t_1 s trajanjem d_1 časovnih enot in akcijo t_2 s trajanjem d_2 časovnih enot. Če žeton v času t_0 prispe v pogoj p_1, se bo iz tega pogoja preko akcije t_1 hipno preselil v časovni točki $t_0 + d_1$ le pod pogojem, da je čas trajanja d_1 krajši od časa trajanja akcije d_2. V primeru, da sta časa trajanja enaka ($d_1 = d_2$), bo izvedena ena od obeh akcij, ki bo izbrana nedeterministično, v primeru pa da je čas d_2 krajši od d_1, pa bo uspešno izvedena le akcija t_2.

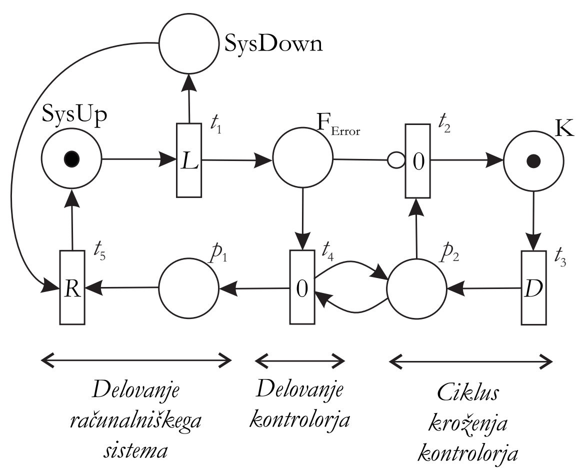


Slika 3.26: Graf Petrijeve mreže z dvema akcijama in njima pripadajočima časovnima trajanjema d_1 in d_2.

V nadaljevanju razdelka si bomo ogledali dva primera zgledov uporabe časovnih Petrijevih mrež in sicer zgled časovno pogojenega servisiranja popravljivega računalniškega sistema in zgled `Stop&Wait` protokola.

3.7.1 Model servisiranja računalniškega sistema

Uporabo časovnosti trajanja akcij v Petrijevih mrežah si bomo najprej ogledali na zgledu modela servisiranja popravljivega računalniškega sistema. Predpostavimo, da je predvideni čas med posameznima sosednjima odpovedima opazovanega sistema (angl. *mean time between fail* - MTBF) definiran s časom L časovnih enot, predvideni čas servisiranja (angl. *mean time to repair* - MTTR) pa definiran s časom R časovnih enot. Delovanje računalniškega sistema s periodičnimi obiski na vsakih D časovnih enot nadzira kontrolor. V primeru, da ob obisku detektira odpoved sistema, sproži postopek servisiranja (npr. naroči servisni poseg). Primer in njegov model sta povzeta po viru [14]. Graf Petrijeve mreže opisanega sistema je predstavljen na sliki 3.27, pri čemer levi del slike predstavlja dinamiko računalniškega sistema (prehajanje med stanji njegovega delovanja in nedelovanja), osrednji del delovanje kontrolorja, desni del slike pa ciklus kroženja kontrolorja.



Slika 3.27: Graf Petrijeve mreže za modeliranje periodične diagnostike popravljivega računalniškega sistema.

Računalniški sistem ponazarjata pogoja `SysUp` (žeton v njem ponazarja delujoče stanje sistema) in `SysDown` (žeton v njem ponazarja nedelujoče stanje sistema) ter akciji t_1 (teče življenjska doba delovanja sistema) in t_5 (poteka servisiranje sistema). Iz povezave navedenih akcij in pogojev je razvidno, da bo sistem po L časovnih enotah prešel v stanje nedelovanja (prenos žetona iz pogoja `SysUp` preko akcije t_1 v pogoj `SysDown`). Pri tem bo pojavitev odpovedi preko akcije t_1 povzročila tudi prenos žetona v pogoj $\mathtt{F}_{\mathtt{Error}}$. Po R časovnih enotah servisiranja bo sistem prešel nazaj v fazo delovanja (prehod žetona iz pogoja `SysDown` preko akcije t_5 v pogoj `SysUp`), pri čemer pa bo za omenjeni prehod od samega začetka servisiranja (izvajanja akcije t_5) potreben tudi žeton v pogoju p_1.

Žeton, ki ponazarja pomen kontrolorja, se na samem začetku nahaja v pogoju `K`. Po D urinih periodah preko akcije t_3 preide v pogoj p_2, odkoder ga pot vodi nazaj v fazo latence (prenos žetona v pogoj `K`), če le ni izpolnjen pogoj $\mathtt{F}_{\mathtt{Error}}$ (spuščena zastavica[8], ki signalizira odpoved sistema). V primeru, da je pogoj $\mathtt{F}_{\mathtt{Error}}$ izpolnjen (zastavica je dvignjena), žeton s pomenom kontrolorja hipno preide iz pogoja p_2 preko akcije t_4 nazaj v pogoj p_2, pri čemer izvedena akcija t_4 spusti zastavico v pogoju $\mathtt{F}_{\mathtt{Error}}$, istočasno pa se en žeton odloži v pogoj p_1, ki omogoči začetek servisiranja sistema. Pri tem imata akciji t_2 in t_4 označen čas njunega trajanja z 0, na osnovi česar sklepamo, da je čas njune izvedbe hipen ali infinitezimalen. Tako žeton s pomenom kontrolorja z izvedbo teh dveh akcij z bivanjem v vhodnih pogojih ne izgubi nobenega časa.

3.7.2 Model `Stop&Wait` protokola

Predpostavimo, da imamo opravka z enosmernim prenosnim kanalom podatkovnih paketov med oddajnikom in sprejemnikom, v katerem na podatkovnem nivoju (2. nivo po OSI modelu, angl. *data link layer*) veljajo sledeče zakonitosti:

- oddajnik *podatkovne pakete* zgolj pošilja, sprejemnik pa podatkovne pakete zgolj sprejema (angl. *simplex protocol*[9]);
- na prenosnem kanalu se paketi lahko izgubljajo ali okvarijo (angl. *noisy channel*);
- za vsak uspešno prejeti podatkovni paket sprejemnik pošlje oddajniku *potrditveni paket*;
- oddajnik po pošiljanju posameznega podatkovnega paketa ne pošlje novega podatkovnega paketa, temveč čaka vse dotlej, dokler ne prejme potrditvenega paketa; na ta način preprečimo eventuelno poplavljanje sprejemnika (angl. *flooding*) z zaporedjem podatkovnih paketov v primeru, da sprejemnik podatkovne pakete sprejema počasneje, kot jih oddajnik oddaja; v primeru, da poteče od oddaje podatkovnega paketa T urinih period in oddajnik ni prejel potrditvenega paketa, podatkovni paket pošlje znova; pri tem mora biti T večji od vsote potovalnih časov podatkovnega in potrditvenega paketa, režijskega časa sprejema paketa ter priprave in pošiljanja potrditvenega paketa; ob predpostavki, da do okvare ali izgube paketa lahko pride le ob prenosu od oddajnika do sprejemnika (kar je dokaj nerealistična predpostavka), do ponovnega pošiljanja podatkovnega paketa tako lahko pride v sledečih dveh primerih:
  - paket ne prispe do sprejemnika in ta ga ne potrdi (paket se je na kanalu izgubil);

[8] Pojem zastavice (angl. *flag*) izhaja iz področja operacijskih sistemov, pri čemer pod zastavico smatramo entiteto, ki se lahko nahaja v enem od dveh možnih stanj. Odtod nahajanje žetona v pogoju $\mathtt{F}_{\mathtt{Error}}$ predstavlja dvignjeno zastavico, odsotnost žetona v njem pa spuščeno zastavico. Pri tem predpostavljamo, da je čas D manjši od časa L, s čimer onemogočimo večkratno porajanje žetonov v pogoju $\mathtt{F}_{\mathtt{Error}}$.

[9] Simplex protokol je zmožen pošiljanja podatkov zgolj v eni smeri (Vir: Wikipedia).

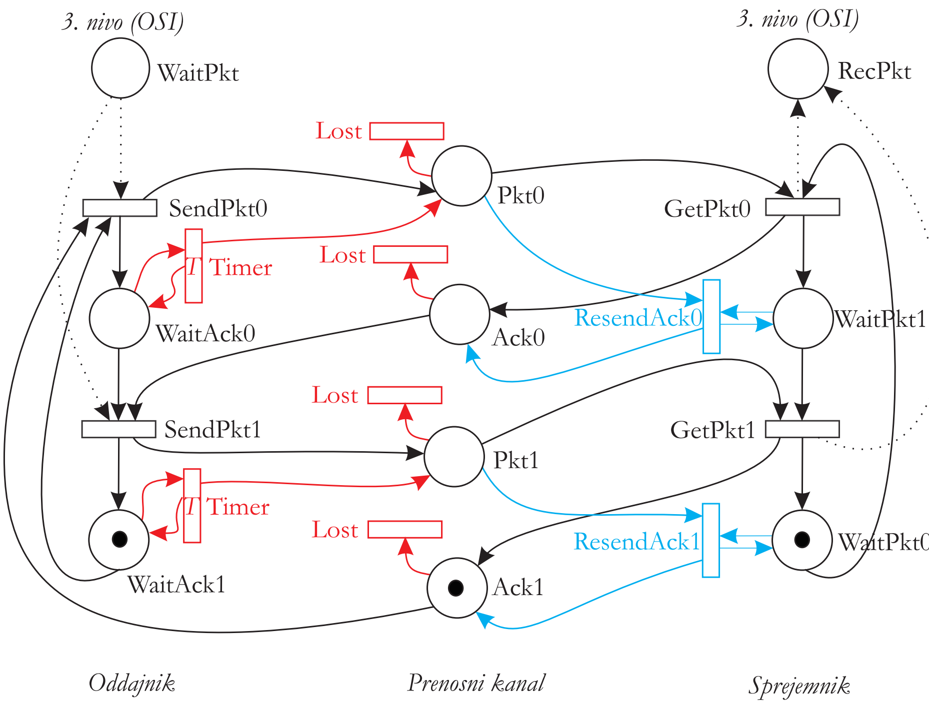
  - paket pride do sprejemnika okvarjen in sprejemnik ga zavrže ter ne izvede potrditve;

- naša predhodna predpostavka, da do okvare ali izgube paketa lahko pride le ob prenosu od oddajnika do sprejemnika je dokaj nerealistična; predpostavimo, da je sprejemnik podatkovni paket prejel, poslal potrditveni paket, le ta pa se je izgubil ali okvaril; v tem primeru bo oddajnik podatkovni paket poslal po T urinih periodah še enkrat in v primeru uspešnega prenosa bo ponovno sprejet s strani sprejemnika; tako v prvem, kot tudi v drugem primeru bo sprejemnik podatkovni paket posredoval na mrežni ali 3. nivo po OSI modelu (angl. *network layer*), s čimer pride do *podvajanja podatkovnih vsebin na mrežnem nivoju*; do slednje anomalije naj po definiciji 2. nivoja ne bi prihajalo, zato moramo dodati logiki opazovanega 2. nivoja še dodatne funkcije, ki bodo to anomalijo onemogočale; iz doslej povedanega je očitno, da bi bilo smiselno na sprejemni strani vpeljati mehanizem za identifikacijo in s tem razlikovanje med *prvič sprejetim podatkovnim paketom* in *ponovno sprejetim podatkovnim paketom*; ena od možnih rešitev je označevanje podatkovnih paketov na strani oddajnika; v namene označevanja vpeljemo zaporedno številko paketa (angl. *sequence number*), ki jo dodeljuje podatkovnemu paketu oddajnik; ker zaporedna številka paketa predstavlja *redundančno vsebino* in s tem obremenjuje procesne in prenosne vire je smiselno, da je le ta čim manjša z vidika števila bitov; ker je pošiljanje naslednjega podatkovnega paketa odvisno le od potrditve predhodno poslanega in uspešno prejetega podatkovnega paketa se izkaže, da je za efektivno označitev dovoljšnja podatkovna vsebina 1 bita (kontrolne vsebine 0 ali 1); tako se naš opazovani oddajnik lahko nahaja v dveh različnih fazah čakanja:
  - stanje 0: oddajnik je poslal paket s kontrolno oznako 0 in čaka na potrditev paketa tipa 0; ko jo prejme gre v oddajo paketa s kontrolno oznako 1;
  - stanje 1: oddajnik je poslal paket s kontrolno oznako 1 in čaka na potrditev paketa tipa 1; ko jo prejme gre v oddajo paketa s kontrolno oznako 0;

Omenjeni protokol imenujemo za `Stop&Wait` protokol neidealnega kanala [15]. Podatkovne pakete je od sprejemnika do oddajnika zmožen prenašati brez izgub, brez podvajanja prejetih podatkovnih paketov in v pravilnem vrstnem redu. Model protokola ponazorjen z grafom Petrijeve mreže je predstavljen na sliki 3.28[10].

Model na sliki 3.28 je zaradi preglednosti narisan v treh fazah. V prvi fazi (črna barva) postavimo model, ki onemogoča poplavljanje sprejemnika s podatkovnimi paketi in vpeljemo potrjevanje podatkovnih paketov. V drugi fazi (rdeča barva) ponazorimo izgubljanje obeh vrst paketov in ponovno pošiljanje

[10]Model protokola je deloma povzet po viru „C. Panayiotou: *Petri-Nets and Other models*“, ki je bil v preteklosti dosegljiv na spletu.



Slika 3.28: Model `Stop&Wait` protokola narisan v treh fazah. V prvi fazi (črna barva) postavimo model, ki onemogoča poplavljanje sprejemnika s podatkovnimi paketi in vpeljemo potrjevanje podatkovnih paketov. V drugi fazi (rdeča barva) ponazorimo izgubljanje obeh vrst paketov in ponovno pošiljanje podatkovnih paketov. V tretji fazi (modra barva) ponazorimo ponovno pošiljanje potrditvenih paketov.

podatkovnih paketov. V tretji fazi (modra barva) ponazorimo ponovno pošiljanje potrditvenih paketov.

Specifike modela protokola ponazorjenega s **črno** barvo so ob upoštevanju začetne označitve sledeče:

- oddajnik v stanju čakanja (žeton v pogoju `WaitAck1`) na potrditev podatkovnega paketa tipa 1 ob prejemu slednjega (žeton v pogoju `Ack1`) odpošlje nov podatkovni paket tipa 0 (izvedba akcije `SendPkt0`); za pošiljanje podatkovnega paketa tipa 0 mora biti istočasno izpolnjen tudi pogoj `WaitPkt`, v katerem 3. nivo pripravi podatkovno vsebino (žeton); ob izvedbi akcije `SendPkt0` se odloži podatkovni paket na prenosni kanal (žeton se odloži v pogoj `Pkt0`), istočasno pa oddajnik preide v stanje čakanja na potrditev podatkovnega paketa tipa 0 (prenos žetona v pogoj `WaitAck0`); ko potrditev prejme preko prenosnega kanala (prihod žetona v pogoj `Ack0`) in je pripravljena nova podatkovna vsebina na 3. nivoju (žeton v pogoju

WaitPkt), se izvede pošiljanje podatkovnega paketa tipa 1 (izvedba akcije SendPkt1); pri tem se paket odloži na prenosni kanal (odlaganje žetona v pogoj Pkt1), oddajnik pa preide v čakanje na potrditev podatkovnega paketa tipa 1 (prenos žetona v pogoj WaitAck1);

- sprejemnik v stanju čakanja (žeton v pogoju WaitPkt0) na prejem podatkovnega paketa tipa 0 s prenosnega kanala ob prejemu slednjega (pojavitev žetona v pogoju Pkt0) odpošlje potrditveni paket tipa 0 in prejeti podatkovni paket posreduje 3. nivoju (izvedba akcije GetPkt0); po izvedbi akcije GetPkt0 sprejemnik preide v stanje čakanja na tip podatkovnega paketa 1 (prenos žetona v pogoj WaitPkt1); ko omenjeni podatkovni paket prispe po prenosnem kanalu (pojavitev žetona v pogoju Pkt1), sprejemnik izvede akcijo GetPkt1, s čimer posreduje prejeti podatkovni paket 3. nivoju, odpošlje potrditveni paket tipa 1 (prenese žeton v pogoj Ack1) in preide v stanje čakanja na nov podatkovni paket tipa 0 (prenos žetona v pogoj WaitPkt0);

Specifike modela protokola ponazorjenega z **rdečo** barvo so ob upoštevanju začetne označitve sledeče:

- oddajnik v stanju čakanja na prejem potrditve podatkovnega paketa tipa 0 (žeton v pogoju WaitAck0) na vsakih T urinih period preko akcije Timer na prenosni kanal znova odloži podatkovni paket tipa 0 (prenos žetona v pogoj Pkt0);
- oddajnik v stanju čakanja na prejem potrditve podatkovnega paketa tipa 1 (žeton v pogoju WaitAck1) na vsakih T urinih period preko akcije Timer na prenosni kanal znova odloži podatkovni paket tipa 1 (prenos žetona v pogoj Pkt1);
- vse štiri vrste paketov (podatkovni paket tipa 0, podatkovni paket tipa 1, potrditveni paket tipa 0, potrditveni paket tipa 1) se lahko na prenosnem mediju izgubijo; v modelu je to ponazorjeno s konkurenčnimi akcijami Lost, ki odvzemajo žetone iz pogojev Pkt0, Ack0, Pkt1 in Ack1; odvzem vseh vrst paketov iz prenosnega kanala se vrši nedeterministično;

Specifiki modela protokola ponazorjenega z **modro** barvo sta ob upoštevanju začetne označitve sledeče:

- sprejemnik v stanju čakanja na podatkovni paket tipa 1 (žeton v pogoju WaitPkt1) ob ponovnem prejemu podatkovnega paketa tipa 0 na prenosni kanal znova odloži potrditveni paket tipa 0 (prenos žetona v pogoj Ack0);
- sprejemnik v stanju čakanja na podatkovni paket tipa 0 (žeton v pogoju WaitPkt0) ob ponovnem prejemu podatkovnega paketa tipa 1 na prenosni kanal znova odloži potrditveni paket tipa 1 (prenos žetona v pogoj Ack1);

Časovnost trajanja akcij je definirana samo za akciji Timer in je nastavljena na T urinih period. V modelu je zaradi preglednosti vključena samo možnost izgubljanja paketov (izvedbe akcij `Lost`), ne pa tudi možnost detekcije okvare paketov. Predpostavimo lahko, da je slednja integrirana v izvedbi akcij `GetPkt0`, `GetPkt1`, `SendPkt1`, `SendPkt0`.

3.8 Povzetek uporabe modeliranja na osnovi Petrijevih mrež

V pričujočen poglavju smo spoznali, da s pomočjo Petrijevih mrež lahko modeliramo različne konstrukte s področja računalništva kot so programski ukazi, diagrami poteka, končni avtomati, generatorji jezikov, nedeterministični algoritmi, računalniški protokoli itd. V splošnem lahko Petrijeve mreže proglasimo za univerzalno okolje modeliranja v domeni dinamičnih procesov in njihovih sinhronizacij.

V preteklih razdelkih v opisu zgledov nismo eksplicitno izpostavili različnih *pomenov žetonov*. V zgledu s slike 3.28 imamo tako v pogojih oddajnika in sprejemnika venomer le en žeton, ki ponazarja njegovo stanje, v pogojih prenosnega kanala pa žetone, ki imajo pomene podatkovnega paketa tipa 0 (nahajanje v pogoju `Pkt0`), podatkovnega paketa tipa 1 (nahajanje v pogoju `Pkt1`), potrditvenega paketa tipa 0 (nahajanje v pogoju `Ack0`) in potrditvenega paketa tipa 1 (nahajanje v pogoju `Ack1`). Običajno imamo v modelu Petrijeve mreže več različnih tipov žetonov, pri čemer se tipi žetonov med seboj ločijo po njihovem pomenu.

Literatura

[1] N. C. Hock, *Queuing Modelling Fundamentals.* John Wiley & Sons, Chichester, Anglija, 1996.

[2] M. Anu, "Introduction to modeling and simulation," in *Proceedings of the 29th conference on Winter simulation* (S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, eds.), pp. 7–13, 1997.

[3] L. Kleinrock and R. Gail, *Queuing systems, problems and solutions.* John Wiley & Sons, New York, ZDA, 1996.

[4] N. Zimic and M. Mraz, *Temelji zmogljivosti računalniških sistemov.* Založba FE in FRI, Ljubljana, Slovenija, 2006.

[5] R. Jamnik, *Verjetnostni račun in statistika.* Društvo matematikov, fizikov in astronomov socialistične republike Slovenije, Zveza organizacij za tehnično kulturo Slovenije, Ljubljana, Slovenija, 1986.

[6] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing and Computer Science Applications.* John Wiley & Sons Inc., New York, ZDA, 2002.

[7] H. Stöcker, *Matematični priročnik z osnovami računalništva.* Tehnična založba Slovenije, Ljubljana, Slovenija, 2006.

[8] J. Virant, *Modeliranje in simuliranje računalniških sistemov.* Didakta, Radovljica, Slovenija, 1991.

[9] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory.* John Wiley & Sons, Hoboken, ZDA, 2018.

[10] J. L. Peterson, *Petri Net Theory and the Modeling of Systems.* Prentice Hall Inc., Englewood Cliffs, ZDA, 1981.

[11] J. Bordon, M. Moškon, N. Zimic, and M. Mraz, "Semi-quantitative Modelling of Gene Regulatory Processes with Unknown Parameter Values Using Fuzzy Logic and Petri Nets," *Fundamenta Informaticae*, vol. 160, no. 1–2, pp. 81–100, 2018.

[12] J. Virant, *Logične osnove odločanja in pomnjenja v računalniških sistemih.* Založba FE in FRI, Ljubljana, Slovenija, 1996.

[13] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of The IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[14] W. G. Schneeweiss, *Petri Nets for Reliability Modeling.* LiLoLe Verlag, 1999.

[15] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks.* Prentice Hall Inc., Boston, ZDA, 2011.