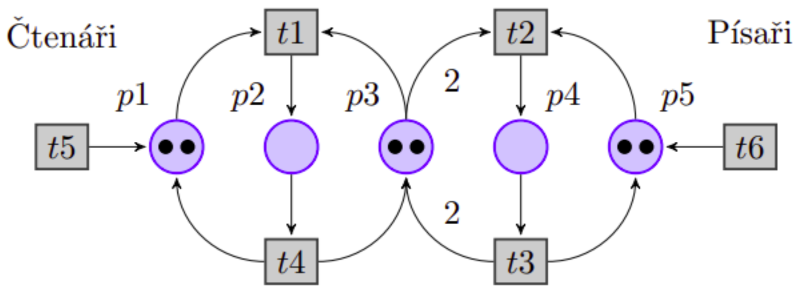
	1	2	3	Σ
xmarci10				

Úloha č. 1

(Termín odovzdania 26.03.2020)

Všetky definície a vety (ktorých dôkazy možno nájsť v citovanom texte) použité v tejto práci sú zo štúdijnej opory predmetu PES [1].



Obr. 1: Čitatelia čakajú v *p1*, čítajú v *p2*. Pisári čakajú v *p5*, zapisujú v *p4*.

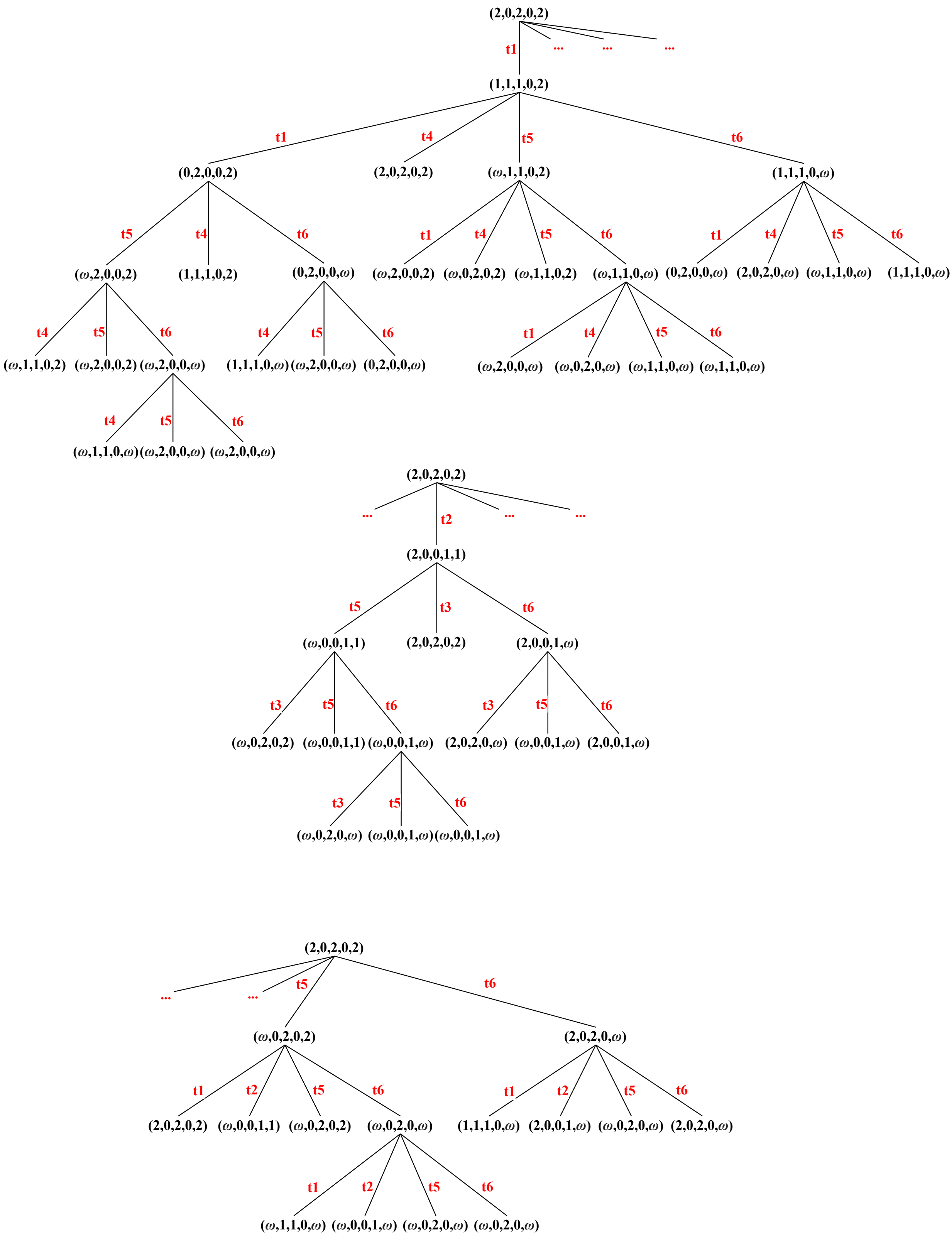
Príklad 1. Uvažujte P/T Petriho sieť z obrázku 1:

1. Zostrojte strom dosiahnuteľných značení. S jeho využitím určte a odôvodnite či:
   (a) je P/T sieť obmedzená,
   (b) je P/T sieť bezpečná,
   (c) je značenie $M_1 = (3, 0, 1, 1, 2)$ pokryteľné, a
   (d) môže byť P/T sieť živá.
2. Uvažujte sieť bez prechodov *t5*, *t6* a s $M_0 = (3, 0, 2, 0, 3)$. Vypočítajte P-invarianty. S ich využitím určte a odôvodnite, či:
   (a) sú vektory $v_1 = (1, 2, 1, 3, 1)$ a $v_2 = (1, 2, 1, 2, 1)$ P-invarianty,
   (b) je P/T sieť striktne konzervatívna, konzervatívna vzhľadom k nejakému váhovému vektoru (ak áno, tak uveďte príklad takého vektoru),
   (c) je značenie $M_2 = (3, 0, 1, 1, 2)$ dosiahnuteľné, a
   (d) interpretujte, čo hovoria P-invarianty o systéme Čitatelia-pisári.
3. Uvažujte sieť bez prechodov *t5,t6* a s $M_0 = (3, 0, 2, 0, 3)$. Vypočítajte T-invarianty.
   (a) Určte a odôvodnite, či sú vektory $v_1 = (30, 20, 20, 30)$ a $v_2 = (2, 3, 2, 3)$ T-invarianty.
   (b) Čo možno z vypočítaných T-invariantov určiť o živosti siete a prečo ?

(*5 bodov*)

Riešenie

1. Skonštruovaný strom dosiahnuteľných značení je možné vidieť na obrázku 2. Na konštrukciu stromu bol použitý algoritmus prezentovaný v štúdijnej opore [1].



Obr. 2: Strom dosiahnuteľných značení Petriho siete zobrazenej na obrázku 1. Pre rozsiahlosť stromu je skonštruovaný strom rozdelený na tri časti, pričom každá z nich začína vždy z počiatočného značenia **(2,0,2,0,2)**. Strom možno taktiež vidieť vo formáte nástroja `Netlab` v prílohe A.

(1a) *Je P/T sieť obmedzená ?*

Podľa [1] je Petriho sieť $N = (P, T, F, W, K, M_0)$ obmedzená ak platí, že:

$$\forall p \in P, \exists k \in \mathbb{N} : \forall M \in [M_0\rangle : M(p) \leq k$$

Zo stromu dosiahnuteľných značení na obrázku 2 môžme teda jednoducho rozhodnúť, že daná sieť **nie je obmedzená**, keďže niektoré značenia obsahujú symbol ω hovoriaci o neobmedzenom (nekonečnom) počte značiek v danom mieste (čo taktiež znamená, že množina dosiahnuteľných značení je nekonečná).

(1b) *Je P/T sieť bezpečná ?*

Podľa **definície 8.1** je Petriho sieť $N = (P, T, F, W, K, M_0)$ bezpečná v prípade, že platí:

$$\forall p \in P, \forall M \in [M_0\rangle : M(p) \leq 1$$

Keďže v predchádzajúcom bode sme dokázali, že naša sieť nie je obmedzená, potom z vyššie uvedenej definície je zrejmé, že **nie je bezpečná**.

(1c) *Je značenie $M_1 = (3, 0, 1, 1, 2)$ pokryteľné ?*

Podľa **definície 8.8** je v Petriho sieti $N = (P, T, F, W, K, M_0)$ značnie M_1 pokryté značením M_2, ak $M_1 \leq M_2$, teda $\forall p \in P : M_1(p) \leq M_2(p)$.

Zo stromu dosiahnuteľných značení na obrázku 2 možno vidieť, že neexistuje taká značenie, ktoré by pokrývalo M_1 a teda M_1 **nie je pokryteľné**.

(1d) *Môže byť P/T sieť živá ?*

Podľa [1] (**Sekcia 8.2.5**) môže byť Petriho sieť živá v prípade ak strom dosiahnuteľných značení neobsahuje koncový vrchol (vrchol bez následníkov).

Keďže v nami skonštruovanom strome na obrázku 2 sú všetky listové uzly uzlami *duplikovanými*, z toho vyplýva, že zadaná sieť **môže byť živá**.

---

2. V Petriho sieti $N = (P, T, F, W, K, M_0)$ P-invariantom nazývame vektor $i : P \to \mathbb{Z}$, ak platí, že $\underline{N}^T.i = 0$ ($\underline{N}^T$ je transponovaná matica Petriho siete N). (**Definícia 9.1**)

Z vyššie uvedenej definície vyplýva, že P-invarianty získame riešením sústavy algebraických rovníc tvaru $\underline{N}^T.x = 0$ ($x \neq 0$).

$$\underline{N} = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ -1 & -2 & 2 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix} \end{array} \qquad \underline{N}^T = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 \\ 0 & 0 & 2 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 \end{pmatrix} \end{array} \qquad (1)$$

Eq. 1: Matica $\underline{N}$ Petriho siete z obrázku 1 a k nej odpovedajúca transponovaná matica $\underline{N}^T$.

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 \\ 0 & 0 & 2 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 \end{pmatrix} \end{array} \cdot \begin{array}{c} x \\ \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} \end{array} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{array}{cccc} b & -a & -c & = 0 \\ d & -2c & -e & = 0 \\ 2c & -d & +e & = 0 \\ a & -b & +c & = 0 \end{array} \tag{2}$$

Eq. 2: Sústava algebraických rovníc, ktorej riešením získame hľadané P-invarianty.

Tabuľka 1: Matica a výsledné minimálne P-invarianty Petriho siete z obrázku 1. pozn.: Na výpočet invariantov bol použitý nástroj `Netlab` a postup zobrazovaný v rovniciach (1) a (2) je iba názorný.

	t_1	t_2	t_3	t_4	$\boldsymbol{i_1}$	$\boldsymbol{i_2}$	$\boldsymbol{i_3}$
p_1	-1	0	0	1	1	0	0
p_2	1	0	0	-1	1	0	1
p_3	-1	-2	2	1	0	0	1
p_4	0	1	-1	0	0	1	2
p_5	0	-1	1	0	0	1	0

(2a) *Sú vektory v_1=(1,2,1,3,1) a v_2=(1,2,1,2,1) P-invarianty ?*

Nech i_1 a i_2 sú P-invarianty siete N a nech $z \in \mathbb{Z}$. Potom $i_1 + i_2$ a $z.i_1$ sú tiež P-invarianty siete N (**Lemma 9.1**).

Podľa vyššie uvedenej vety môžme ukázať, že vektor $\mathbf{v_1}$=(1,2,1,3,1) **je P-invariant** danej Petriho siete, keďže ho môžeme rozložiť na súčet invariantov $i_1 + i_2 + i_3$:

$$(1, 1, 0, 0, 0) + (0, 0, 0, 1, 1) + (0, 1, 1, 2, 0) = \mathbf{(1, 2, 1, 3, 1)}$$

Ďalší spôsob ako overiť, či je vektor v_1=(1,2,1,3,1) P-invariant vyplýva priamo z definície P-invariantu. Konkrétne môžme overiť platnosť vzťahu $\underline{N}^T.v_1 = 0$:

$$\underline{N}^T \cdot v_1 = \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 \\ 0 & 0 & 2 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \\ 1 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{0} \tag{3}$$

Podobným spôsobom budeme postupovať aj pri vektore v_2:

$$\underline{N}^T \cdot v_2 = \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 \\ 0 & 0 & 2 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 1 \\ 0 \end{pmatrix} \neq \mathbf{0} \tag{4}$$

Vidíme, že vzťah z definície neplatí a z toho vyplýva, že vektor $\mathbf{v_2}$=(1,2,1,2,1) **nie je P-invariant** danej Petriho siete.

(2b) *Je P/T sieť striktne konzervatívna, konzervatívna vzhľadom k nejakému váhovému vektoru (ak áno, tak uveďte príklad takého vektoru) ?*

(**Definícia 8.3**) Nech $N = (P, T, F, W, K, M_0)$ je Petriho sieť. N je striktne konzervatívna, ak platí:

$$\forall M \in [M_0\rangle : \sum_{p \in P} M(p) = \sum_{p \in P} M_0(p)$$

.

Z definície 8.3 vyplýva, že ak má byť naša sieť striktne konzervatívna musí byť podľa definície 8.4 (viz ďalej) konzervatívna vzhľadom k váhovému vektoru $(1, 1, 1, 1, 1)$. Tento vektor (alebo jeho k-násobok) by tak musel byť P-invariantom našej siete, čo nie je pravda a z toho vyplýva, že zadaná sieť **nie je striktne konzervatívna**.

(**Definícia 8.4**)Nech $N = (P, T, F, W, K, M_0)$ je Petriho sieť a $v : P \to \mathbb{N}$ vektor. Sieť N je *konzervatívna vzhľadom k váhovému vektoru* v, ak platí

$$\forall M \in [M_0\rangle : \sum_{p \in P} v(p)M(p) = \sum_{p \in P} v(p)M_0(p)$$

V zmysle predchádzajúcej definície, môže byť za váhový vektor považovaný každý P-invariant siete N. A teda každý P-invariant P/T siete vo všeobecnosti vyjadruje váhový vektor, vzhľadom na, ktorý je daná sieť **konzervatívna**. Príkladom môžu byť invarianty i_1, i_2, i_3 a taktiež všetky ich lineárne kombinácie.

(2c) *Je značenie $M_2 = (3, 0, 1, 1, 2)$ dosiahnuteľné ?*

(**Veta 9.1**) Nech N je Petriho sieť s počiatočným značením M_0. Potom pre každý P-invariant i siete N a pre každé dosiahnuteľné značenie $M \in [M_0\rangle$ platí

$$M.i = M_0.i$$

V našom prípade, ale pre značenia $M_0 = (3, 0, 2, 0, 3)$, $M_2 = (3, 0, 1, 1, 2)$ a invariant $i_3 = (0, 1, 1, 2, 0)$ vyššie uvedená veta neplatí:

$$M_0.i_3 = (3, 0, 2, 0, 3).(0, 1, 1, 2, 0) = (0, 0, 2, 0, 0)$$

$$M_2.i_3 = (3, 0, 1, 1, 2).(0, 1, 1, 2, 0) = (0, 0, 1, 2, 0)$$

$M_0.i_3 \neq M_2.i_3 \Rightarrow$ značenie M_2 **nie je dosiahnuteľné**

(2d) *Interpretujte, čo hovoria P-invarianty o systéme Čitatelia-pisári.*

Z jednotlivých invariantov i_1, i_2, i_3 vyplýva pre každé značenie $M \in [M_0\rangle$ nasledujúce:

- Invariant $\boldsymbol{i_1} = (1, 1, 0, 0, 0)$:

$$\sum_{i=1}^{2} M(p_i) = \sum_{i=1}^{2} M_0(p_i) = 3$$

  To znamená, že počet procesov je konštantný,rovný 3 (žiadne procesy sa nestrácajú ani nepribúdajú), a že každý proces je v jednom zo stavov p_1, p_2. Na týchto miestach sú umiestnený čitatelia a každý z nich buď čaká, alebo číta.

- Invariant $\boldsymbol{i_2} = (0,0,0,1,1)$

$$\sum_{i=4}^{5} M(p_i) = \sum_{i=4}^{5} M_0(p_i) = 3$$

Význam je podobný ako pri invariante i_1. Zmena je iba v miestach, keďže v tomto prípade je každý z procesov v jednom zo stavov p_3, p_4. Na týchto miestach sú pre zmenu umiestnený pisári a znova každý z nich buď čaká alebo zapisuje.

- Invariant $\boldsymbol{i_3} = (0,1,1,2,0)$

$$M(p_2) + M(p_3) + 2 \cdot M(p_4) = M_0(p_2) + M_0(p_3) + 2 \cdot M_0(p_4) = 2$$

Teda p_4 obsahuje nanajvýš jednu značku, tj. vždy existuje nanajvýš jeden zapisujúci proces. Ak miesto p_4 obsahuje značku, potom $M(p_2) = M(p_3) = 0$, tj. akonáhle niektorý z procesov zapisuje, žiadny ďalší proces nemôže čítať. Miesto p_2 môže obsahovať maximálne 2 značky, tj. maximálne 2 procesy môžu simultánne čítať z vyrovnávacej pamäte. V takomto prípade nikto nezapisuje (p_4 prázdne) a rovnako nie sú k dispozícii žiadne zdroje (p_3 prázdne).

---

3. Tentokrát sa budeme oproti P-invariantom zaoberať riešením sústavy rovníc tvaru:

$$N.u = 0$$

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array}\begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ -1 & -2 & 2 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix} \end{array} \cdot \begin{array}{c} u \\ \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \end{array} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{array}{rrrrcl} & & -a & +d & = & 0 \\ & & a & -d & = & 0 \\ -a & -2b & +2c & +d & = & 0 \\ & & b & -c & = & 0 \\ & & -b & +c & = & 0 \end{array} \quad (5)$$

Eq. 3: Sústava algebraických rovníc, ktorej riešením získame hľadané T-invarianty.

Tabuľka 2: Výsledné T-invarianty Petriho siete z obrázku 1.

	t_1	t_2	t_3	t_4
u_1	0	1	1	0
u_2	1	0	0	1

(3a) *Určte a odôvodnite, či sú vektory* $v_1 = (30,20,20,30)$ *a* $v_2 = (2,3,2,3)$ *T-invarianty.*

Nech i_1 a i_2 sú T-invarianty siete N a nech $z \in \mathbb{Z}$. Potom $i_1 + i_2$ a $z.i_1$ sú taktiež T-invarianty siete N (**Lemma 9.3**).

Podľa vyššie uvedenej vety môžme ukázať že vektor $\mathbf{v_1} = (30,20,20,30)$ **je T-invariant**, pretože ho môžeme vyjadriť lineárnou kombináciou $20.i_1 + 30.i_2 = (0,20,20,0) + (30,0,0,30) = (30,20,20,30)$.

Rovnako ako pri P-invariantoch môžme použiť alternatívny spôsob vychádzajúci z definície a overovať platnosť vzťahu $N.u = 0$:

$$\underline{N} \cdot v_1 = \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 \\ 0 & 0 & 2 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 30 \\ 20 \\ 20 \\ 30 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{0} \tag{6}$$

Podobne postupujeme pre vektor v_2:

$$\underline{N} \cdot v_2 = \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 \\ 0 & 0 & 2 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 2 \\ 3 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \neq \mathbf{0} \tag{7}$$

Vzťah z definície neplatí a teda $\mathbf{v_2} = (2, 3, 2, 3)$ **nie je T-invariant** danej Petriho siete.

(3b) *Čo možno z vypočítaných T-invariantov určiť o živosti siete a prečo ?*

Tentokrát budeme vychádzať z **definície 9.6**, hovoriacej o pokrytí siete T-invariantami: Petriho sieť N je pokrytá T-invariantami, ak pre každý prechod t siete N existuje nezáporný T-invariant i siete N taký, že $i(t) > 0$.

Taktiež použijeme **vetu 9.9** ktorá tvrdí, že každá živá a obmedzená Petriho sieť je pokrytá T-invariantami.

Z predchádzajúceho tvrdenia vychádza, že ak Petriho sieť nie je pokrytá T-invariantami tak potom nie je živá a obmedzená. To však automaticky neznamená, že ak je pokrytá T-invariantami tak je živá a obmedzená. Z definície pokrytia vidíme, že naša Petriho sieť je pokrytá a tak **nevylučujeme, že by mohla byť živá**.

Príklad 2. Modelujte P/T sieťou Dijkstrov algoritmus pre vzájomné vylúčenie. Pseudokód v Algoritme 1 predpokladá neobmedzene mnoho paralelne spustených procesov, každý s unikátnym identifikátorom, a popisuje algoritmus pre proces s identifikátorom i. Pole booleovských hodnôt *flag* a booleovská premenná p sú zdieľané všetkými procesmi, a sú inicializované na 0. Modelujte verziu systému s práve dvoma procesmi, s indexmi 0 a 1 (unikátne indexy pre neobmedzene mnoho procesov P/T siete modelovať neide).

Algoritmus 1: Proces s indexom i

```
while true do
    flag[i] := true;
    if p ≠ i then
        wait until not flag[p];
        p := i;
    if ∃j : j ≠ i ∧ flag[j] then continue ;
    critical_section;
    flag[i] := false;
```

1. Modelujte systém v nástroji `Netlab`. Použijte modelovacie techniky, kde miesta v sieti odpovedajú programovým riadkom a hodnotám premenných. Snažte sa o čo najväčšiu prehľadnosť modelu, použite textové označenie miest a prechodov. Urobte v `Netlab`-e dostupné analýzy a interpretujte výsledky. Na ich základe odôvodnite odpovede na nasledujúce otázky:
   (a) Garantuje protokol vzájomné vylúčenie (t.j., procesy nemôžu byť súčasne v kritickej sekcii) ?
   (b) Garantuje nemožnosť uviaznutia ?
2. Čo sa stane, ak dovolíme, aby kód procesov s indexmi 0 a 1 vykonávalo zároveň neobmedzene veľa procesov ? Vyskúšajte v `Netlab`-e.

Model musí byť spustiteľný vo verzii nástroja `Netlab`, odkazovanej zo stránky predmetu `https://www.fit.vutbr.cz/study/courses/PES/private/.cs`. Na stránke nájdete aj návod, ako nástroj použiť na vlastnom počítači. Nástroj funguje v počítačových učebniach, viz. stránka predmetu MBA. Odovzdajte ho do informačného systému do termínu odovzdania úlohy.

(*5 bodov*)

Riešenie

1. Vytvorený model v nástroji `Netlab` je zobrazený na obrázku 3. Výslednú sieť je tiež možno nájsť v odovzdanom súbore `xmarci10.net`.

Popis modelu Červené miesta $\{1, 2, \ldots, 6\}$ predstavujú zdieľané premenné (pozn.: každá hodnota zdieľanej premennej je modelovaná ako samostatné miesto, viz popisy miest). Zelené čiary znázorňujú pristupovanie procesu P_0 k zdieľaným zdrojom. Naopak modré čiary značia prístup procesu P_1 k zdieľaným premenným.

Ďalej miesta $7, 8, \ldots, 14$ predstavujú kód procesu P_0 (každé miesto je jeden riadok kódu, viz popisy miest). Zvyšné miesta predstavujú kód procesu P_1.

Pre jednoduchosť sme sa rozhodli pre menšiu úpravu vo výslednom modele oproti algoritmu 1. Konkrétne ide o príkaz `continue` na riadku 6. V sieti sa tento príkaz nevráti na úplný začiatok cyklu, ale len na test podmienky na riadku 3 (nie je nutné znova nastavovať `flag[i]` na hodnotu `true`).

Obmedzenosť V strome dosiahnuteľných značení vytvoreného pomocou nástroja `Netlab` si možno všimnúť, že v žiadnom značení sa nenachádza symbol ω, čo znamená, že naša sieť sĺňa podmienku z **definície 8.2** a tým pádom **je obmedzená**. Túto vlastnosť je možné určiť aj pomocou P-invariantov viz výstup z `Netlab`-u:

```
Boundedness (RG):
The net is bounded.
Sufficient conditions for invariants:
  There exists a positive P-invariant.
  Therefore, the sufficient condition for boundedness is satisfied,
  and the net is bounded.
```

Bezpečnosť Keďže kapacita každého miesta v našej siete je 1, naša sieť splňuje podmienku bezpečnosti z **definície 8.1** a tým pádom **je bezpečná**. Možno taktiež overiť pomocou stromu dosiahnuteľných značení, ktorého listy tvoria iba binárne vektory (vektory pozostávajúce z núl a jednotiek).

Živosť Z výstupu `Netlab`-u vidíme, že postačujúca podmienka (hovoriaca o T-invariantoch) pre živosť je splnená no sieť napriek tomu **nie je živá**.

```
Liveness (RG, condensed):
The net is not live.
Necessary conditions for invariants:
  There exists a positive T-invariant.
  Therefore, the necessary condition for liveness is
  satisfied, and the net may be live.
```

Dôvodom je možnosť vzniknutia dvoch čiastočných uviaznutí (jedno pre každý proces; závisí na poradí počiatočnej sekvencie).

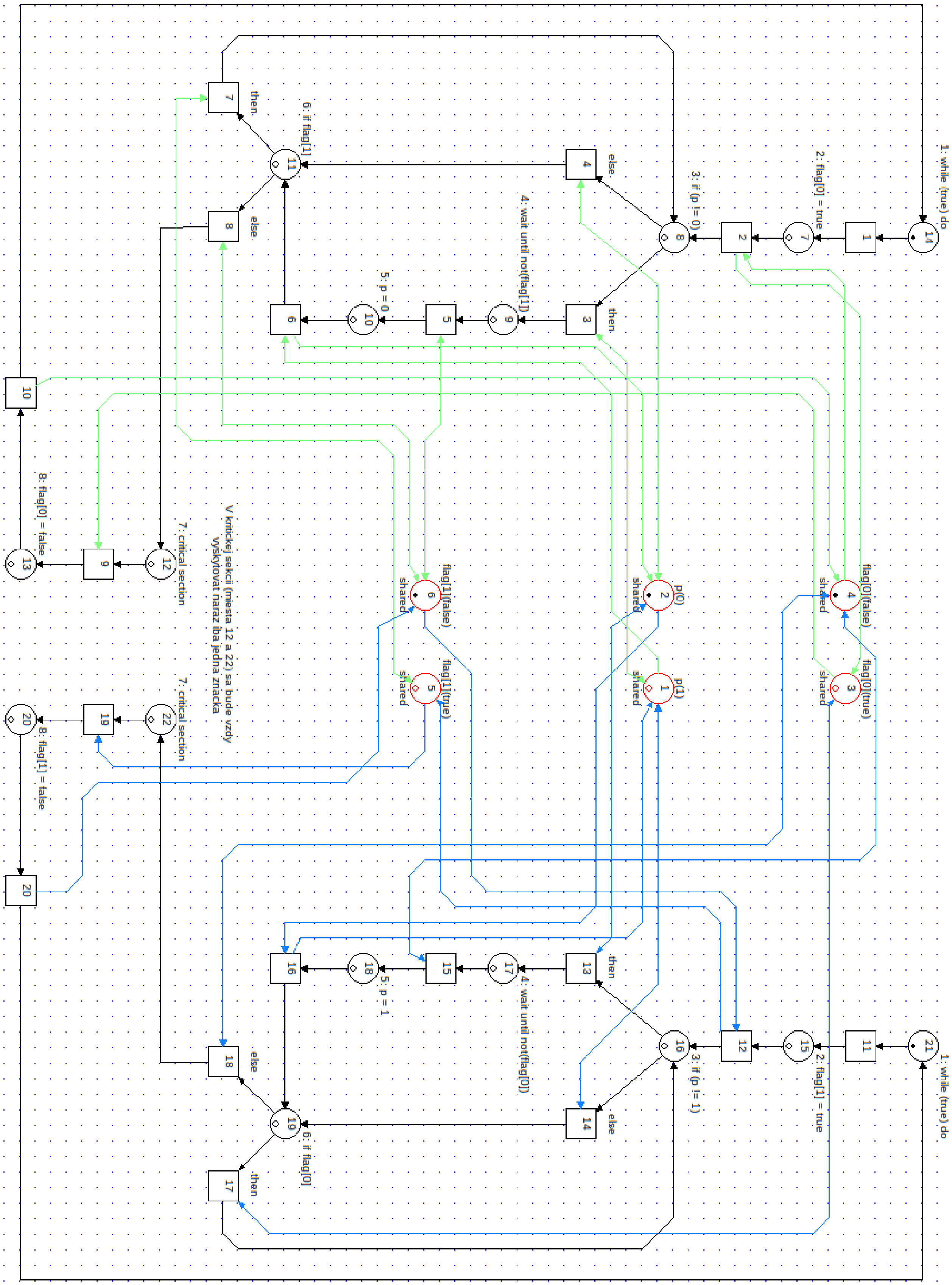
Prvé čiastočné uviaznutie nastane po vykonaní nasledujúcej sekvencii prechodov z počiatočného značenia M_0 :

$$t_1, t_2, t_{11}, t_{12}, t_{13}.$$

Aktiváciou prechodov t_1, t_2 nastaví proces s indexom 0 `flag[0]=true`, následne proces 1 aktiváciou prechodov t_{11}, t_{12}, t_{13} nastaví `flag[1]=true` a keďže nemá rezervovanú kritickú sekciu (`p!=1`) vykoná prechod t_{13}, ktorým začne čakanie na uvoľnenie kritickej sekcie. Následne proces 0 pokračuje vo vykonávaní ale keďže `flag[1]=true` bude donekonečna cykliť medzi miestami 8 a 11. No a následkom toho bude proces 1 donekonečna čakať v mieste 17. Z **definície 8.5** potom vyplýva, že naša sieť **nie je živá**.

Druhé čiastočné uviaznutie je podobného charakteru s tým rozdielom, že k nemu dôjde inou sekvenciou z počiatočného značenia a procesy si vymenia svoje role. Tentokrát bude donekonečna cykliť proces 1 a čakajúcim procesom bude proces 0.

Konzervatívnosť Výsledná sieť na obrázku 3 **je striktne konzervatívna**, keďže sa počet značiek počas behu simulácie nikdy nezmení (vždy sa bude v sieti nachádzať práve 5 značiek). Túto vlastnosť možno overiť buď zo stromu dosiahnuteľných znační alebo pomocou P-invariantov.



Obr. 3: Model P/T siete Dijkstrov algoritmus podľa zadania v príklade 2.

(1a) *Garantuje protokol vzájomné vylúčenie (t.j., procesy nemôžu byť súčasne v kritickej sekcii) ?*

ÁNO. Na overenie môžme opäť použiť strom dosiahnuteľných značení. Stačí skontrolovať, že v žiadnom zo značení sa nenachádza 1 zároveň v miestach 12 a 22, ktoré reprezentujú kritickú sekciu (viz Obr. 3). To znamená, že musí platiť:

$$\forall M \in [M_0\rangle : (M(12) + M(22)) \leq 1$$

Vyššie uvedená podmienka pre našu sieť platí a teda protokol **garantuje vzájomné vylúčenie**.

(1b) *Garantuje nemožnosť uviaznutia ?*

ÁNO. Možno to určiť znova zo stromu dosiahnuteľných značení, ktorý nemá žiadny *koncový* vrchol. Tým pádom **nemôže** nikdy nastať situácia, že žiadny prechod siete nebude uskutočniteľný (čo je vlastne definícia uviaznutia). Na druhej strane môže nastať čiastočné uviaznutie ako už bolo spomenuté pri popise životnosti siete.

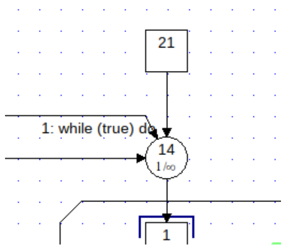
```
Total deadlock (RG):
none.
```

---

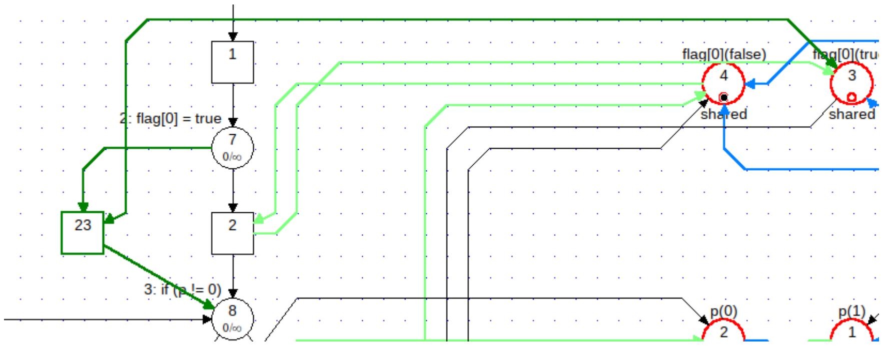
2. *Čo sa stane, ak dovolíme, aby kód procesov s indexmi 0 a 1 vykonávalo zároveň neobmedzene veľa procesov ? Vyskúšajte v* `Netlab`*-e.*

Na to aby daný kód mohlo vykonávať neobmedzene veľa procesov bolo nutné v prvom rade urobiť pár zmien v modelovanej sieti.

- Bolo nutné zabezpečiť neobmedzený prísun procesov do siete. To sme dosiahli pridaním prechodov, ktoré pridávali procesy do miest 14 a 21 v pôvodnej sieti.



- Bolo taktiež nutné upraviť kapacitu miest na nekonečno. Jediné kapacity, ktoré ostali nastavené rovnako ako v pôvodnej sieti boli kapacity miest, ktoré modelujú hodnoty zdieľaných premenných.
- Posledná modifikácia spočíva v úprave siete tam, kde proces zapisuje hodnotu do niektorej zo zdieľaných premenných. V pôvodnej sieti sa pri zápise hodnoty x do premennej vyžaduje značka z miesta reprezentujúceho hodnotu $\neg x$. To by však malo za následok, že po zapísaní hodnoty jedného z procesov by ostatné museli čakať vo fronte (tým pádom by kód nebol vykonávaný viacerými procesmi naraz). Modifikácia tak spočíva v pridaní nového prechodu, ktorý bude vykonateľný v prípade, že premenná už obsahuje požadovanú hodnotu. Úprava je zobrazená na obrázku 4.

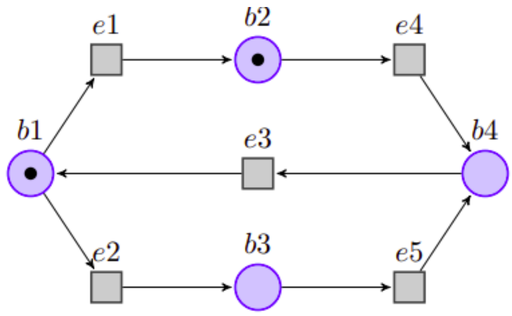


Obr. 4: Úprava zapisovania hodnôt do zdieľaných premenných pri použití viacerých procesov. Modifikácia je zobrazená tmavo zelenou farbou.

Spomínaná modifikácia tak má za následok to, že nová sieť už nebude obmedzená z čoho vyplýva, že nebude ani bezpečná. Taktiež už **nebude zaručené vzájomné vylúčenie procesov** (napr.: prvý proces, ktorý opustí kritickú sekciu ju sprístupní aj v prípade, že sa tam ešte nachádzajú ďalšie procesy). V modifikovanej sieti, ale stále môže dojsť k už spomínanému čiastočnému uviaznutiu.

Príklad 3. Pre C/E systém na obrázku nižšie urobte nasledujúce:

1. Rozhodnite, ktorá z nasledujúcich formulí je platná a zakreslite ju pomocou faktov.
   (a) $(\neg b1 \rightarrow (\neg b4 \rightarrow (\neg b2 \rightarrow \neg b3)))$
   (b) $(b1 \wedge b2) \rightarrow (b2 \vee b4)$
2. Komplementujte systém.
3. Nakreslite prípadový graf.
4. Pre komplementovaný systém nakreslite najkratší proces, kde sa jeden prípad vyskytuje dvakrát (má dva rôzne S rezy, ktoré zobrazuje na rovnaký prípad).
5. Ktorým cestám v prípadovom grafe odpovedá tento proces ?



Obr. 5: C/E Systém.

(4 body)

Riešenie

1. *Rozhodnite, ktorá z nasledujúcich formulí je platná a zakreslite ju pomocou faktov.* Pri riešení tejto úlohy budeme vychádzať z **definície 6.9**, ktorá hovorí že v C/E systéme Σ je formula $a \in A_\Sigma$ *platnou* (angl. *valid*), ak $\forall c \in C_\Sigma : \hat{c}(a) = 1$.

(1a) Platnosť formuly *(a)* overíme vyplnením pravdivostnej tabuľky:

	b_1	b_2	b_3	b_4	$(\neg \mathbf{b1} \rightarrow (\neg \mathbf{b4} \rightarrow (\neg \mathbf{b2} \rightarrow \neg \mathbf{b3})))$
$\{b_1, b_2\}$	1	1	0	0	**1**
$\{b_2, b_3\}$	0	1	1	0	**1**
$\{b_3, b_4\}$	0	0	1	1	**1**
$\{b_1, b_3\}$	1	0	1	0	**1**
$\{b_1, b_4\}$	1	0	0	1	**1**
$\{b_2, b_4\}$	0	1	0	1	**1**

Ako vidíme z tabuľky, formula *(a)* je pre všetky prvky prípadovej triedy pravdivá a teda podľa **definície 6.9 je platná**.

(1b) Ak z formuly *(b)* odstránime implikácia pomocou pravidla $a \rightarrow b \leftrightarrow \neg a \vee b$ dostávame:

$(b1 \wedge b2) \rightarrow (b2 \vee b4) \leftrightarrow \neg(b1 \wedge b2) \vee (b2 \vee b4) \leftrightarrow \neg b1 \vee \neg b2 \vee b2 \vee b4 \leftrightarrow \neg b1 \vee 1 \vee b4 \leftrightarrow 1$

Formula *(b)* je teda vždy pravdivá a teda **platná**.

Pri faktoch sa oprieme o druhú časť **definície 6.9** hovoriacu o tom, že pre fakt t formulu *a(t)* definujeme rovnako ako pre udalosť e formulu *a(e)* (**definícia 6.8**). Napríklad ak: ${}^{\bullet}t = \{b_1, \ldots, b_n\}$ a $t^{\bullet} = \{b'_1, \ldots, b'_m\}$, tak

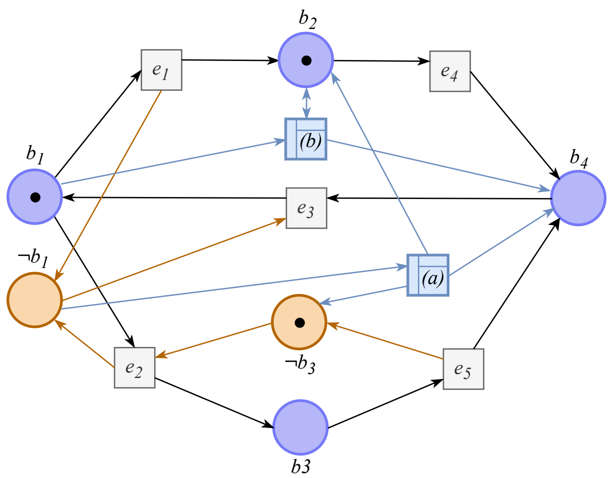
$$a(t) = (b_1 \wedge b_2 \wedge \cdots \wedge b_n) \rightarrow (b'_1 \vee b'_2 \vee \cdots \vee b'_m)$$

Pred samotným zakreslením faktov si tak musíme príslušné formuly previesť do požadovaného tvaru. Formula *(b)* sa už v takomto tvare nachádza a formulu *(a)* upravíme nasledovne:

$$(\neg b1 \rightarrow (\neg b4 \rightarrow (\neg b2 \rightarrow \neg b3))) \leftrightarrow (\neg b1 \rightarrow (\neg b4 \rightarrow (b2 \vee \neg b3))) \leftrightarrow$$

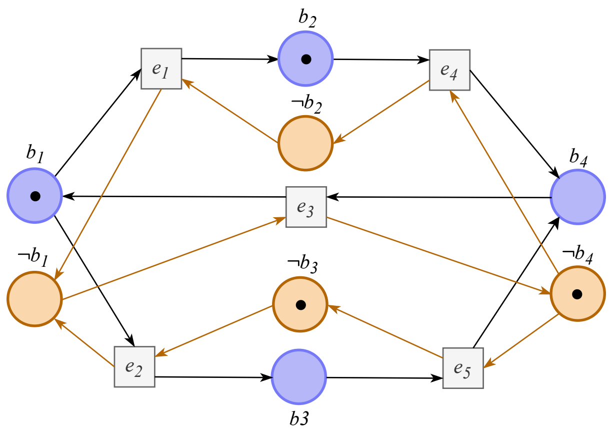
$$\leftrightarrow (\neg \mathbf{b1} \rightarrow (\mathbf{b4} \vee \mathbf{b2} \vee \neg \mathbf{b3}))$$

Z výsledného tvaru formuly *(a)* vidíme, že je ešte potrebné komplementovať miesta b_1 a b_3. To môžme urobiť bez ovplyvnenia fungovania C/E systému, čo potvrdzuje aj **veta 4.6**. Výsledný systém s faktami *(a)* a *(b)* je na obrázku 6.



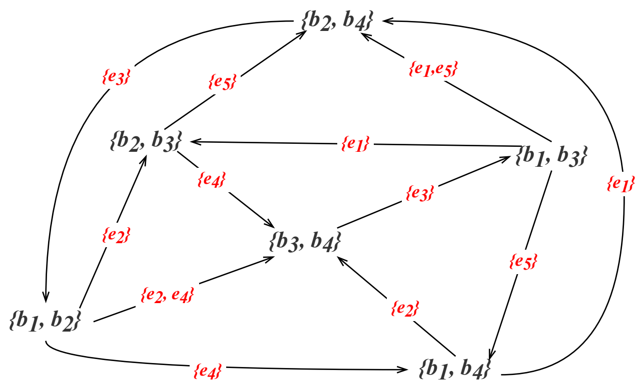
Obr. 6: C/E Systém z obrázku 5 spolu s faktami reprezentujúcimi formuly *(a)* a *(b)*.

2. *Komplementujte systém.*



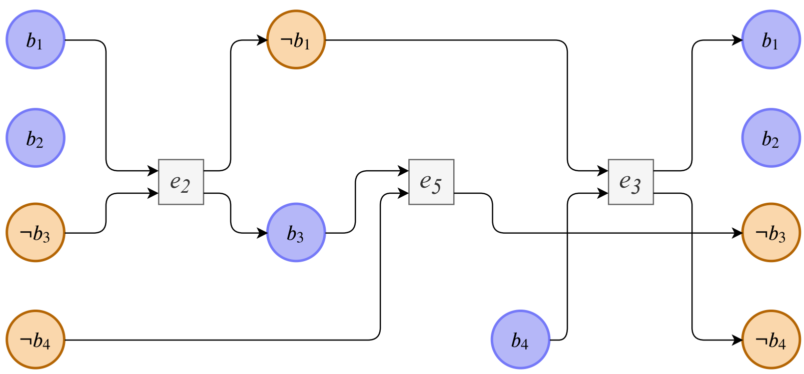
Obr. 7: Komplementovaný C/E systém z obrázka 5.

3. *Nakreslite prípadový graf.*



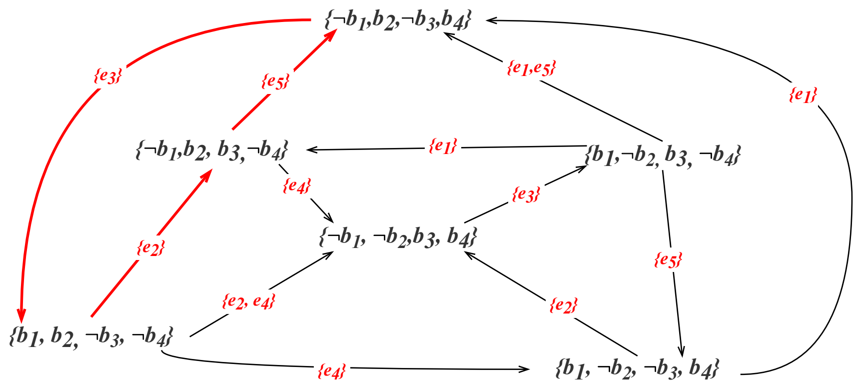
Obr. 8: Prípadový graf C/E systému z obrázka 5.

4. *Pre komplementovaný systém nakreslite najkratší proces, kde sa jeden prípad vyskytuje dvakrát (má dva rôzne S rezy, ktoré zobrazuje na rovnaký prípad).*



Obr. 9: Výskytová sieť procesu, ktorá reprezentuje nasledujúcu sekvenciu: $\{\mathbf{b1}, \mathbf{b2}, \neg\mathbf{b3}, \neg\mathbf{b4}\}\,[e2\rangle\,\{\neg b1, b2, b3, \neg b4\}\,[e5\rangle\,\{\neg b1, b2, \neg b3, b4\}\,[e3\rangle\,\{\mathbf{b1}, \mathbf{b2}, \neg\mathbf{b3}, \neg\mathbf{b4}\}$. Práve počiatočný prípad sa vyskytuje dvakrát.

5. *Ktorým cestám v prípadovom grafe odpovedá tento proces ?*



Obr. 10: Prípadový graf C/E systému z obrázka 7. Červené čiary odpovedajú procesu z obrázka 10.

Literatúra

[1] Češka, M.; Marek, V.; aj.: *Petriho sítě PES* (Studijní opora). December 2009. URL https://www.fit.vutbr.cz/study/courses/PES/public/Pomucky/PES_opora.pdf

Prílohy

A Strom dosiahnuteľných značení `Netlab`

```
M001:   0 (  2  0  2  0  2) ---t1---> M002:   1 (  1  1  1  0  2)
                            ---t2---> M003:   1 (  2  0  0  1  1)
                            ---t5---> M004:   1 (  *  0  2  0  2)
                            ---t6---> M005:   1 (  2  0  2  0  *)
M002:   1 (  1  1  1  0  2) ---t1---> M006:   2 (  0  2  0  0  2)
                            ---t4---> M001:   0 (  2  0  2  0  2)
                            ---t5---> M007:   2 (  *  1  1  0  2)
                            ---t6---> M008:   2 (  1  1  1  0  *)
M003:   1 (  2  0  0  1  1) ---t3---> M001:   0 (  2  0  2  0  2)
                            ---t5---> M009:   2 (  *  0  0  1  1)
                            ---t6---> M010:   2 (  2  0  0  1  *)
M004:   1 (  *  0  2  0  2) ---t1---> M007:   2 (  *  1  1  0  2)
                            ---t2---> M009:   2 (  *  0  0  1  1)
                            ---t5---> M004:   1 (  *  0  2  0  2)
                            ---t6---> M011:   2 (  *  0  2  0  *)
M005:   1 (  2  0  2  0  *) ---t1---> M008:   2 (  1  1  1  0  *)
                            ---t2---> M010:   2 (  2  0  0  1  *)
                            ---t5---> M011:   2 (  *  0  2  0  *)
                            ---t6---> M005:   1 (  2  0  2  0  *)
M006:   2 (  0  2  0  0  2) ---t4---> M002:   1 (  1  1  1  0  2)
                            ---t5---> M012:   3 (  *  2  0  0  2)
                            ---t6---> M013:   3 (  0  2  0  0  *)
M007:   2 (  *  1  1  0  2) ---t1---> M012:   3 (  *  2  0  0  2)
                            ---t4---> M004:   1 (  *  0  2  0  2)
                            ---t5---> M007:   2 (  *  1  1  0  2)
                            ---t6---> M014:   3 (  *  1  1  0  *)
M008:   2 (  1  1  1  0  *) ---t1---> M013:   3 (  0  2  0  0  *)
                            ---t4---> M005:   1 (  2  0  2  0  *)
                            ---t5---> M014:   3 (  *  1  1  0  *)
                            ---t6---> M008:   2 (  1  1  1  0  *)
M009:   2 (  *  0  0  1  1) ---t3---> M004:   1 (  *  0  2  0  2)
                            ---t5---> M009:   2 (  *  0  0  1  1)
                            ---t6---> M015:   3 (  *  0  0  1  *)
M010:   2 (  2  0  0  1  *) ---t3---> M005:   1 (  2  0  2  0  *)
                            ---t5---> M015:   3 (  *  0  0  1  *)
                            ---t6---> M010:   2 (  2  0  0  1  *)
M011:   2 (  *  0  2  0  *) ---t1---> M014:   3 (  *  1  1  0  *)
                            ---t2---> M015:   3 (  *  0  0  1  *)
                            ---t5---> M011:   2 (  *  0  2  0  *)
                            ---t6---> M011:   2 (  *  0  2  0  *)
M012:   3 (  *  2  0  0  2) ---t4---> M007:   2 (  *  1  1  0  2)
                            ---t5---> M012:   3 (  *  2  0  0  2)
                            ---t6---> M016:   4 (  *  2  0  0  *)
M013:   3 (  0  2  0  0  *) ---t4---> M008:   2 (  1  1  1  0  *)
                            ---t5---> M016:   4 (  *  2  0  0  *)
                            ---t6---> M013:   3 (  0  2  0  0  *)
```

```
M014:    3 (  *  1  1  0  *) ---t1---> M016:   4 (  *  2  0  0  *)
                             ---t4---> M011:   2 (  *  0  2  0  *)
                             ---t5---> M014:   3 (  *  1  1  0  *)
                             ---t6---> M014:   3 (  *  1  1  0  *)
M015:    3 (  *  0  0  1  *) ---t3---> M011:   2 (  *  0  2  0  *)
                             ---t5---> M015:   3 (  *  0  0  1  *)
                             ---t6---> M015:   3 (  *  0  0  1  *)
M016:    4 (  *  2  0  0  *) ---t4---> M014:   3 (  *  1  1  0  *)
                             ---t5---> M016:   4 (  *  2  0  0  *)
                             ---t6---> M016:   4 (  *  2  0  0  *)
Graph construction is complete!
```