Žilinská univerzita v Žiline
Fakulta riadenia a informatiky

Dizertačná práca

SYSTÉM NA PODPORU ROZHODOVANIA PRE RIADENIE DOPRAVNÝCH PROCESOV

na získanie vedecko – akademickej hodnosti doktor filozofie
v odbore doktorandského štúdia

37 – 01 – 9 Dopravná a spojová technológia

Ing. Michal Žarnay

Školiteľ: prof. Ing. Petr Cenek, CSc.
Miesto a dátum: Žilina, január 2007

POĎAKOVANIE

Moja veľká vďaka patrí

predovšetkým môjmu školiteľovi prof. Ing. Petrovi Cenkovi, CSc. za neoceniteľné rady a pripomienky, ktorými usmernil moju prácu,

kolegom a spolupracovníkom, najmä Ing. Mgr. Ľubomírovi Sadloňovi, PhD., doc. Mgr. Valentovi Klimovi, CSc., doc. Ing. Antonínovi Kavičkovi, PhD., Ing. Norbertovi Adamkovi, PhD., Ing. Milošovi Zaťkovi a Ing. Petrovi Mártonovi za inšpiráciu a cenné konzultácie, ktoré ovplyvnili moju prácu,

najbližšej rodine za veľkú trpezlivosť, toleranciu a skvelé vytvorené podmienky, navyše otcovi za bohaté pripomienky k formálnej stránke a

všetkým ostatným priateľom a známym za trpezlivosť a podporu.

1 ÚVOD DO PROBLEMATIKY 5
1.1 DOPRAVNÝ SYSTÉM A JEHO RIADENIE 6
1.1.1 Dopravný systém 6
1.1.2 Riadenie dopravného systému 8
1.1.3 Automatizácia riadenia 10
1.1.4 Riadenie simulačného modelu 11
1.1.5 Kvalita rozhodnutí 12
1.1.6 Implementácia automatického riadenia 14
1.2 IDENTIFIKÁCIA PROBLÉMU 17
1.2.1 Analýza rozhodovacích situácií 17
1.2.2 Uviaznutie 19
1.2.3 Riešený problém 21
2 SÚČASNÝ STAV RIEŠENEJ PROBLEMATIKY 22
2.1 RIEŠENIE PROBLÉMU UVIAZNUTIA 22
2.1.1 Nutné podmienky pre uviaznutie 22
2.1.2 Graf prideľovania prostriedkov 23
2.1.3 Detekcia a zotavenie z uviaznutia 23
2.1.4 Prevencia pred uviaznutím 24
2.1.5 Vyhnutie sa uviaznutiu 24
2.1.6 Ignorovanie uviaznutia 25
2.2 POUŽÍVANÉ FORMALIZMY 26
2.2.1 Systém diskrétnych udalostí (DES) 26
2.2.2 Systém prideľovania prostriedkov (RAS) 27
2.2.2.1 Kategórie RAS 28
2.2.2.2 Vyhýbanie sa uviaznutiu v RAS 30
2.2.3 Petriho siete 32
2.2.3.1 Modelovanie RAS podtriedami Petriho siete 33
2.3 EXISTUJÚCE RIEŠENIA PROBLÉMU 38
2.3.1 Algoritmus bankára 38
2.3.1.1 Základná verzia algoritmu 38
2.3.1.2 Vylepšenia a zložitosť algoritmu 40
2.3.1.3 Upravený algoritmus pre AGV 45
2.3.1.4 Zhrnutie k verziám algoritmu bankára 46
2.3.2 Ďalšie algoritmy a prístupy 46
2.3.2.1 Prístupy založené na sifónoch 46
2.3.2.2 Prístupy založené na analýze stavového priestoru 47
2.3.2.3 Iné metódy 48
2.3.2.4 Metódy pre vektorové systémy diskrétnych udalostí 48
2.3.2.5 Zložitosť algoritmov 48
2.3.3 Zhrnutie k algoritmom a prístupom 49
3 CIELE A METODIKA PRÁCE 50
4 VLASTNÉ RIEŠENIE 51
4.1 MODEL DOPRAVNÉHO SYSTÉMU VO FORME RAS V PETRIHO SIETI 51
4.1.1 Analýza prideľovania prostriedkov procesom v dopravnom systéme 51
4.1.1.1 Činnosti a procesy 51
4.1.1.2 Vykonávanie činností 53
4.1.1.3 Kategorizácia procesov 54
4.1.1.4 Detaily k prideľovaniu prostriedkov 55
4.1.2 Model dopravného systému v Petriho sieti 57
4.1.2.1 Prevod sieťového grafu na Petriho sieť 57

4.1.2.2 Modelovanie všeobecných obslužných procesov Petriho sieťou 59
4.1.2.3 Modelovanie premiestňovacích procesov Petriho sieťou 60
4.1.2.4 Vytvorenie modelu vo farbenej Petriho sieti 61
4.2 PRÍSTUPY K RIEŠENIU UVIAZNUTIA V DOPRAVNOM SYSTÉME 70
4.2.1 Detekcia a zotavenie z uviaznutia v dopravnom systéme 70
4.2.2 Prevencia pred uviaznutím v dopravnom systéme 71
4.2.3 Vyhýbanie sa uviaznutiu v dopravnom systéme 73
4.2.4 Zhrnutie k použitiu prístupov 74
4.3 MOŽNOSTI RIEŠENIA UVIAZNUTIA 74
4.3.1 Úprava metód pre sekvenčný RAS 74
4.3.2 Prevod nesekvenčného na sekvenčný proces 74
4.3.3 Algoritmy pre systém s nesekvenčnými procesmi 75
4.3.4 Výber algoritmu a východiská pre riešenie 76
4.3.5 Meranie kvality algoritmu riadenia 77
4.4 ÚPRAVY ALGORITMOV 77
4.4.1 Zavedenie paralelného spracovania procesu 78
4.4.1.1 Princíp 78
4.4.1.2 Úprava algoritmu aktualizácie údajov 80
4.4.1.3 Použitie variantov spracovania 81
4.4.1.4 Počiatočné nastavenie údajových štruktúr 82
4.4.2 Vykonanie procesu podľa značeného grafu 82
4.4.2.1 Údajové štruktúry pre vykonanie procesu 83
4.4.2.2 Algoritmus pre vykonanie procesu 84
4.4.3 Výber prostriedkov podľa profesií 85
4.4.3.1 Zmeny v údajových štruktúrach 86
4.4.3.2 Zmeny v operáciách 87
4.4.4 Výsledné verzie algoritmu bankára 89
4.4.4.1 Parametre algoritmov 93
4.4.5 Realizácia vyhýbania sa uviaznutiu 94
5 VÝSLEDKY RIEŠENIA A ICH ZHODNOTENIE 95
5.1 DEMONŠTRAČNÝ MODEL 95
5.1.1 Požiadavky na model 95
5.1.2 Všeobecný popis modelu 96
5.1.3 Model vo farbenej Petriho sieti 100
5.1.4 Reprezentácia procesu 100
5.1.5 Príklady stavov uviaznutia 101
5.2 IMPLEMENTÁCIA NAVRHNUTÉHO ALGORITMU 102
5.2.1 Údajové štruktúry a funkcie 102
5.2.2 Zapojenie do Petriho siete 103
5.3 EXPERIMENTY A ICH VÝSLEDKY 104
5.3.1 Výber experimentov 104
5.3.2 Výsledky analýzy 104
6 ZÁVER 109
6.1 ZHRNUTIE 109
6.2 PRÍNOSY DIZERTAČNEJ PRÁCE 110
6.3 ĎALŠIE PROBLÉMY 111
7 ZOZNAM POUŽITÝCH ZDROJOV 112
8 VYSVETLENIE POUŽITÝCH SKRATIEK A SYMBOLOV 119
9 PRÍLOHY 121

1 Úvod do problematiky

Tak ako vo všetkých oblastiach spoločnosti, aj v doprave je snaha stále všetko zlepšovať. Či už je to jej technické vybavenie, alebo atribúty poskytovaných služieb ako bezpečnosť, rýchlosť, spoľahlivosť, pohodlnosť alebo efektívnosť prevádzky. Toto úsilie sa nezaobíde bez zlepšovania riadenia dopravy, ktoré má na kvalitu celého systému významný vplyv. Kvalita riadenia zasa závisí od spôsobu riešenia riadiacich úloh v dopravnom systéme.

S rozvojom telematiky, t. j. informatiky, komunikačnej, výpočtovej a automatizačnej techniky, ako aj matematiky, teórie systémov a ďalších oblastí sa otvorili nové možnosti pre skvalitnenie riadiacich metód. Okrem toho významný nárast výpočtových možností v posledných rokoch umožnil aj vývoj a častejšiu aplikáciu metód náročných na výpočtový čas a pamäť.

Metódy riešenia riadiacich úloh možno v podstate rozdeliť na dve kategórie: metódy prinášajúce optimálne riešenie a tie, od ktorých možno získať riešenie bez záruky optimálnosti. Zaradenie metódy závisí okrem výpočtovej náročnosti najmä od charakteru riešenej úlohy a od zložitosti riešeného systému. S určitým zjednodušením možno konštatovať, že k optimálnemu riešeniu sa možno dopracovať najmä pri úlohách jednoduchších a v jednoduchších dopravných systémoch, prípadne použitím určitého zjednodušenia. So stúpajúcou zložitosťou úloh a systémov sa možnosti riešiť úlohy exaktne zvyčajne zmenšujú a riešiteľ sa musí uspokojiť s výsledkami bez zaručenia ich optimálnosti. Tie možno získať pomocou heuristických a simulačných metód. Heuristické metódy sú založené na pravidlách, ktoré usmerňujú hľadanie riešenia tak, aby sa získalo riešenie pokiaľ možno čo najbližšie optimálnemu v primeranom čase. Často sa používajú v kombinácii s exaktnými metódami. Simulačné metódy využívajú počítač na zaznamenanie správania sa modelu v čase a štatistické vyhodnotenie sledovaných ukazovateľov. Na rozdiel od heuristických metód neposkytujú zväčša riešenie, ale pomáhajú používateľovi pri jeho hľadaní.

A práve počítačová simulácia dopravného systému, konkrétne simulácia zriaďovacích a osobných staníc (Žarnay-Kavička, 1999, Žarnay et al., 2002, Kavička et al., 2005, Bažant-Žarnay, 2005/1-3), bola zdrojom problému a poskytla motiváciu na jeho riešenie v tejto dizertačnej práci. Ide o problém uviaznutia v dopravnom systéme, t. j. stav, ktorý sa v realite, kde dopravný systém riadi človek, vyskytuje s minimálnou pravdepodobnosťou. Avšak pri počítačovej simulácii zložitého systému s rozmanitými prvkami a vzťahmi, kde sa riadenie vykonáva počítačovým algoritmom, sa stav uviaznutia môže i napriek kvalitnému návrhu systému vyskytnúť častejšie.

Vo svojej dizertačnej práci sa venujem tomuto problému a jeho riešeniu. V prvej kapitole bližšie opíšem dopravný systém a jeho riadenie, ďalej ich modelovanie a naznačím problémy, ktoré sa môžu vyskytnúť. Z nich je potom vybratý problém uviaznutia, pri ktorom zanalyzujem, kedy a ako v dopravnom systéme vzniká a čo

spôsobuje. Druhá kapitola sa venuje opisu súčasného stavu riešenia problému v dopravných systémoch ako aj v iných oblastiach, ktoré majú podobnú charakteristiku – na základe zdrojov, ktoré som o danej problematike našiel. Analytická časť je pomerne rozsiahla, lebo si to zameranie úlohy vyžaduje.

V tretej kapitole sú tézy dizertačnej práce, ktoré som si v spolupráci so školiteľom vytýčil. Návrh môjho riešenia týkajúci sa modelu dopravného systému a algoritmov na vyhýbanie sa uviaznutiam nasleduje vo štvrtej, najrozsiahlejšej kapitole celej práce. Nasledujúcu kapitolu venujem demonštračnému modelu, ktorý dokumentuje aplikovateľnosť riešenia a poskytuje analytické výsledky pre vytvorené algoritmy. Celú prácu uzatvárajú záverečné konštatovania v šiestej kapitole.

1.1 Dopravný systém a jeho riadenie

1.1.1 Dopravný systém

Systém je usporiadaná množina prvkov navzájom prepojených väzbami, ktorá sa dá charakterizovať štruktúrou a cieľovým správaním. Možno ho opísať stavmi a ich zmenami.

Obslužný systém je systém, v ktorom možno rozlíšiť 2 základné typy prvkov, a to:

- zákazníkov so svojimi požiadavkami na obsluhu a
- linky (prostriedky) obsluhujúce zákazníkov spracovaním ich požiadaviek.

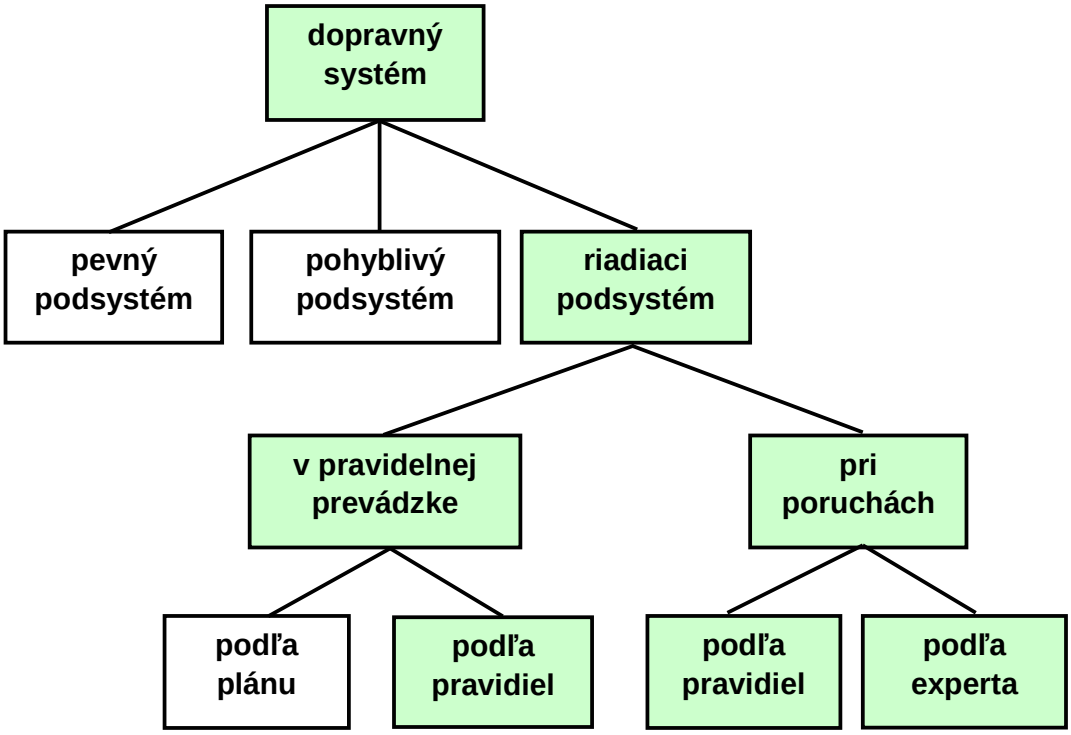
Zákazníci s požiadavkami tvoria vstupný prúd. Pokiaľ sú linky obsadené, požiadavky čakajúce na spracovanie sa môžu zoraďovať do frontu. Spracovanie požiadavky zaberá určitý čas a obslúžených zákazníkov možno považovať za výstupný prúd.

Dopravný a prepravný systém sú príkladmi obslužného systému. Prepravný systém má za cieľ premiestnenie ľudí alebo zásielok podľa prepravných požiadaviek. Premiestnenie sa realizuje v dopravnom systéme pomocou činností presunu, to znamená, že dopravný systém sa detailnejšie zaoberá technologickou stránkou prepravy. Tá je tiež predmetom tejto práce.

V terminológii obslužného systému sú prepravné požiadavky zákazníci, resp. zákazky. Prostriedkami ich obsluhy sú v tomto prípade dopravná infraštruktúra, vozidlá (v prípade potreby) a obslužný personál. Prostriedky spolu so zákazkami tvoria prvky dopravného systému, ktoré možno ďalej rozčleniť na tri podsystémy: pevný, pohyblivý a riadiaci (Obr. 1.1). (Cenek et al., 1994, str. 15)

Pevný podsystém zahŕňa prvky, ktoré nemenia v čase svoju polohu a sú základom pre prevádzku systému. Je to dopravná sieť alebo dopravná infraštruktúra.

Pohyblivý podsystém obsahuje prvky, ktoré menia svoju polohu v čase. Robia tak buď preto, aby realizovali samotný prepravný proces, ako automobily, vlaky, lode či lietadlá, alebo aby vykonali činnosti podporujúce prepravné procesy, ako personál,



Obr. 1.1 Dopravný systém a jeho podsystémy – podfarbené rámčeky označujú oblasť tejto práce.

prostriedky údržby, posunovacie rušne a iné. Každý pohyblivý prvok má určité atribúty, týkajúce sa nielen pohybu a polohy, ale aj jeho činnosti v systéme.

Riadiaci podsystém riadi prvky dopravného systému, aby sa zákazky prepravili zo svojej aktuálnej pozície do cieľa. Tu možno uviesť ako príklad činnosť dopravných dispečerov, zabezpečovacieho a komunikačného systému alebo riadiacu výpočtovú techniku.

Súčasťou pohyblivého podsystému sú aj zásielky, ktoré potrebujú na obsluhu prostriedky spravidla zo všetkých troch podsystémov. Ide o vzťah dopravného systému voči jeho okoliu, za ktorý možno považovať aj prepravný systém.

Pod pojem dopravného systému možno zaradiť rôzne reálne systémy. Mojou ambíciou je nájsť riešenie pre ľubovoľný dopravný systém. Avšak v prípade potreby konkrétneho príkladu dopravného systému budem sa orientovať na železničnú stanicu, teda uzol v železničnej sieti, kde sa vykonávajú technologické činnosti so zákazkami za účelom ich prepravy podľa objednávky prepravcu. Podľa povahy prepravnej prevádzky sa železničné stanice delia na osobné, nákladné, prekládkové a zmiešané (Flodr, 1990, str. 13).

Zákazníci uzla železničnej siete (ľudia alebo zásielky) sa v procese prepravy nachádzajú buď na miestach stacionárnej obsluhy alebo vo vlakoch, ktoré možno ešte ďalej deliť na vozne. Ďalšie delenie na konkrétne osoby, kusy alebo iné jednotky tovaru si dovolím zanedbať, nakoľko z dopravného hľadiska je vo väčšine procesov postačujúce uvažovať o vozni, prípadne o vlaku ako o najmenšej jednotke. V prípade, že by to tak nebolo a v uzle by sme skúmali aj zmenu obsahu vozňa a tok cestujúcich alebo nákladu, v texte to uvediem.

Medzi prostriedky v uzle železničnej dopravy možno uviesť najmä

- koľajisko, zabezpečovacie zariadenie, technologické zariadenia, budovy a i. v pevnom podsystéme,
- rušne, vozne a personál, potrebné pre vytvorenie vlaku, resp. posunovacie rušne a železničných zamestnancov v pohyblivom podsystéme a
- riadiacich pracovníkov (dispečerov, výpravcov), ktorých riadiace úkony sa realizujú prostredníctvom zabezpečovacieho a komunikačného zariadenia v riadiacom podsystéme.

1.1.2 Riadenie dopravného systému

Pojem riadenie možno všeobecne chápať ako vykonávanie postupnosti rozhodnutí, ktoré môžu spôsobiť zmenu stavu systému. Rozhodnutia sa vykonávajú na základe aktuálneho stavu systému a dostupných potrebných informácií o ňom. Rozhodovací cyklus pozostáva zo štyroch fáz: informačnej, poznávacej, rozhodovacej a kontrolnej (Obr. 1.3).

Z pohľadu základných funkcií možno riadenie procesov v systéme všeobecne rozdeliť do troch fáz: plánovanie, organizovanie a kontrolovanie. Vo fáze plánovania sa vopred naplánujú činnosti v systéme. V druhej fáze sa na základe známych informácií o stave systému prijímajú rozhodnutia tak, aby systém vykonal naplánované činnosti a splnil plánované ciele. Fáza kontrolovania slúži na vyhodnotenie dopadu vykonania činností, porovnanie realizácie s plánom a poskytne spätnú väzbu riadiacemu systému.

Riadiacim subjektom je jednotka vykonávajúca rozhodnutia – je to osoba, stroj alebo ich kombinácia. Budem ju nazývať **riadiacim systémom**, ktorý je vo vzťahu k celému dopravnému systému podsystémom. Riadenými objektmi sú v tomto prípade prvky ostatných podsystémov dopravného systému.

Na klasifikáciu riadiacich procesov možno použiť časové hľadisko dosahu riadenia. Podľa neho možno rozlíšiť tri úrovne: strategickú, taktickú a operatívnu.

Strategické riadenie zahŕňa rozhodnutia dlhodobého charakteru, ktoré formujú dopravný systém na dlhé obdobie. Ako príklady možno uviesť návrh trás pre relácie v železničnej sieti, návrh diaľničnej križovatky alebo obstaranie lietadiel pre prevádzku leteckej spoločnosti. Také rozhodnutia sa zvyčajne robia na úrovni vrcholového manažmentu organizácií a potrebujú dlhý čas na prípravu a realizáciu, v typickom prípade mesiace alebo roky.

Operatívne riadenie je opačným extrémom – pozostáva z krátkodobých rozhodnutí založených na aktuálnom stave systému a ovplyvňujúcich prevádzku systému zväčša v nadchádzajúcich sekundách, minútach, prípadne hodinách. Vykonávajú ich riadiace štruktúry na najnižšej úrovni, ktorá má priamy dosah na prevádzku. Ako príklad možno menovať postavenie cesty pre vlak v železničnej stanici, úpravu rýchlostného limitu na diaľnici podľa aktuálneho počasia alebo organizáciu prekládky v prístave.

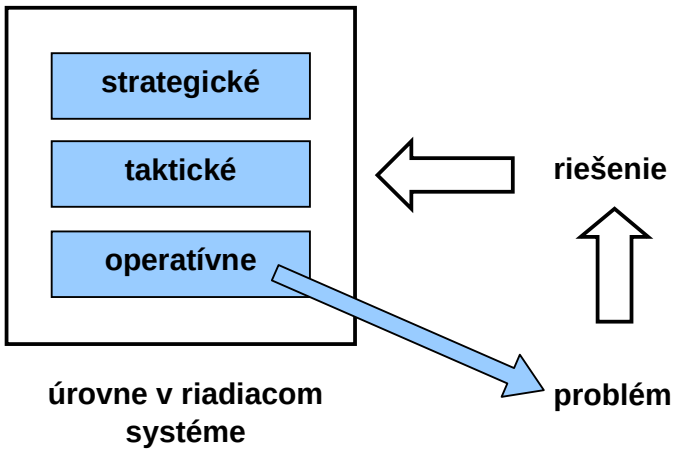
Taktické riadenie sa nachádza medzi dvoma uvedenými úrovňami. Možno tam zaradiť rozhodnutia ovplyvňujúce prevádzku systému v strednodobom horizonte, ako je grafikon verejnej dopravy alebo úprava triediacej schémy pre daný deň v zriaďovacej stanici.

Riadiaci problém spracovaný v tejto práci pochádza z úrovne operatívneho riadenia. Jeho riešenie sa však môže dotýkať nielen najnižšej, ale aj vyšších úrovní riadenia (Obr. 1.2).

Z pohľadu vzťahu plánovania procesov v prvej fáze a organizovania v druhej fáze možno rozlíšiť 2 typy riadenia:

- Riadenie **v pravidelnej prevádzke** (podľa plánu) – predpokladá opakovateľnosť procesov, ktorá dovolí vypracovanie plánu práce, podľa ktorého sa potom riadi.
- **Operatívne** riadenie odchýlok od plánu – zhodnotí konkrétny stav systému a vykoná rozhodnutie až vtedy, keď rozhodovacia situácia nastane.

V bežnom dopravnom systéme sa zvyčajne vyskytujú prvky riadenia obidvoch typov. Napríklad príchody a odchody vlakov sú určené grafikonom vlakovej dopravy, rovnako ako obeh čiat, práca staničných zamestnancov, triediaca schéma a ďalšie oblasti prevádzky stanice. Avšak náhodné vplyvy, zväčša časové odchýlky od očakávaného priebehu vyžadujú operatívne prispôsobenie riadenia aktuálnemu stavu.



Obr. 1.2 Problém vzniká na operatívnej úrovni a jeho riešenie ovplyvňuje všetky úrovne riadenia.

Podľa Obr. 1.1 možno ďalej riadenie v pravidelnej prevádzke rozdeliť na riadenie **podľa plánu** a **podľa pravidiel**. Plán sa zvyčajne postaví pre procesy, ktoré sa pravidelne opakujú, ako napr. pravidelné dopravné spoje na linkách. Pre procesy, ktoré sa vyskytujú občas, avšak ich výskyt sa očakáva, riadenie definuje vopred pravidlá (postupy, návody) ako napríklad pravidlá pre vykonanie opravy vozňov alebo rušňov.

Pri odchýlkach od plánu možno rozlíšiť medzi riadením **podľa pravidiel** a **podľa experta**. Riadenie podľa pravidiel sa uplatní vtedy, keď sa vyskytne stav, pre ktorý sú pravidlá vytýčené – podobne ako bolo vyššie uvedené. Tieto stavy nie sú síce plánované, no možno ich výskyt vďaka náhodným vplyvom v systéme predpokladať. Ako príklad možno uviesť priraďovanie náhradných nástupištných koľají zmeškaným vlakom, ktorých neskorý príchod znemožnil priradiť im koľaje určené plánom obsadenia koľají.

Avšak nie každý stav zapríčinený odchýlkou sa dá predvídať. V situáciách, kde pravidlá nie sú definované, je nutné rozhodnutie experta: buď od človeka alebo od počítačového expertného systému.

1.1.3 Automatizácia riadenia

Automatické riadenie je také riadenie, kde sa ľudský činiteľ nahradí technickým systémom, najčastejšie s podporou výpočtovej techniky. Technický systém vykonáva rozhodnutia automaticky, bez nutnosti zásahu človeka. Ideálne si to možno predstaviť tak, že technický systém úplne nahradí ľudské myslenie a vykonáva rozhodnutia minimálne v rovnakej kvalite, ako by to urobil skúsený človek.

V praxi sa automatizácia zväčša vykonáva iba do istej miery, na medzistupni na ceste k ideálnej predstave. Niekedy ani nie je cieľom úplná automatizácia riadenia, najmä v systémoch veľmi citlivých na chyby. V oblasti dopravy sa nájdu jednoduché systémy alebo časti systémov, kde sa už úplné automatické riadenie presadilo, no sú aj oblasti a zložité systémy, ktoré si bez ľudského činiteľa je zatiaľ ťažké predstaviť. Často dochádza v riadení ku kombinácii, kde technika poskytne podporné podklady, na ktorých základe urobí rozhodnutie človek.

S automatizáciou riadenia je spojené aj modelovanie a simulácia. Na jednej strane je simulácia správania systému užitočná pri odskúšaní automatického riadiaceho podsystému pripravovaného pre reálne nasadenie. Na druhej strane si samotné simulačné modely taktiež vyžadujú vlastné riadenie. A pokiaľ má používateľ simulácie záujem na využití predností simulácie, medzi ktoré patrí jej rýchlosť, tak je vhodné, aby bolo riadenie automatické s minimalizáciou zásahov človeka.

Čím je simulačný model zložitejším, tým sa zvyšujú aj nároky na jeho automatické riadenie. Tento fakt poskytol prvotnú motiváciu aj pre túto prácu. A pokiaľ získané poznatky tejto práce nájdu uplatnenie v riadení simulačných modelov, tak ich celkom isto možno využiť aj pre riadenie reálnych dopravných systémov.

V ďalšom texte sa pokúsim naznačiť určité zovšeobecnené skúsenosti z riadenia simulačných modelov dopravných systémov a jeho vzťahu k riadeniu reálneho dopravného systému.

1.1.4 Riadenie simulačného modelu

Ako som naznačil v kapitole 1.1.3, podstatný rozdiel medzi bežným riadením dopravného systému a automatickým riadením jeho simulačného modelu je ľudský činiteľ. Kým v reálnom svete je ťažisko rozhodovania vo väčšine situácií na ľuďoch, v modeli máme ambíciu nechať na počítačové algoritmy všetky rozhodnutia, aj tie, ktoré v realite vykonávajú ľudia (Žarnay, 2004/1).

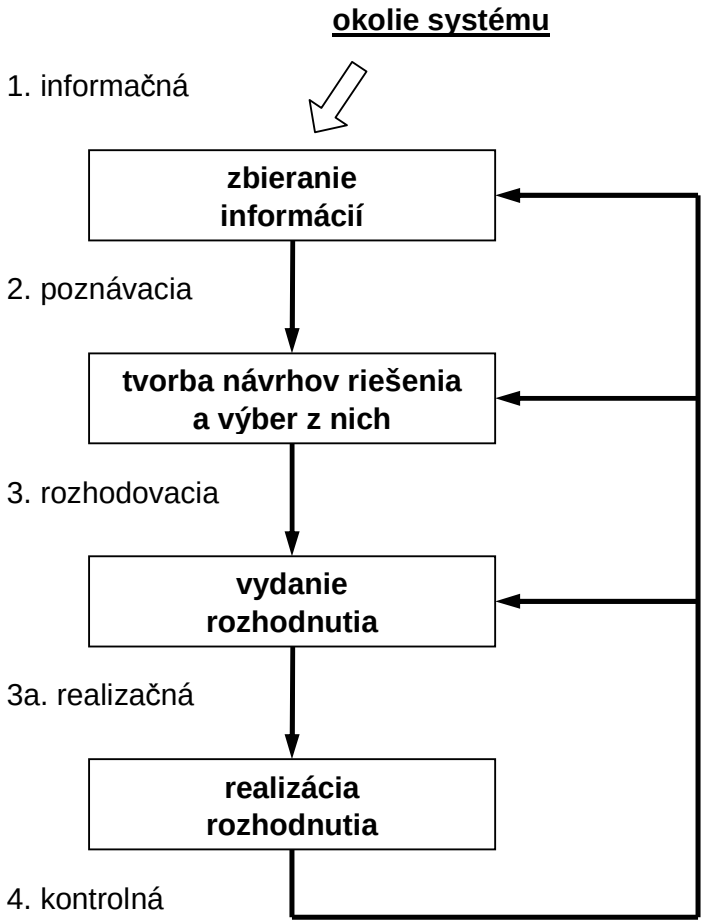
Riadiaci podsystém simulačného modelu možno takisto rozčleniť podľa schémy na Obr. 1.1 a aj z tohto pohľadu zhodnotiť skúsenosti s prácou so simulačnými modelmi. **V pravidelnej prevádzke** sa simulačný model riadi plánom alebo pravidlami, ktoré vytvoril tvorca modelu. Oboje je spravidla vytvorené vopred a model má nástroje na ich interpretovanie, čo má zvyčajne hladký priebeh. Implementácia **riadenia pri odchýlkach** je však oveľa zložitejšia, lebo vyžaduje nahradenie ľudského myslenia inteligenciou algoritmov v oveľa väčšej miere.

Ak sa táto inteligencia neimplementuje algoritmicky, tak sa rozhodovanie ponechá na operátorovi modelu. Vtedy sa pri každej rozhodovacej situácii beh simulačného modelu zastaví a čaká na odpoveď od operátora. Početné zastavenia predĺžia čas vykonania simulačného behu, najmä v modeloch zložitých systémov, a jedna z podstatných výhod simulácie sa stráca. Tento dôsledok vytvára potrebu nájsť inteligentný riadiaci podsystém, ktorý by bol schopný vyriešiť operatívne pokiaľ možno čo najviac problémov. Môže nahradiť operátora úplne a odpovedať priamo, alebo môže pomôcť operátorovi pri výbere rozhodnutia.

Okrem podobnosti v systémovom členení možno medzi riadiacimi podsystémami v simulačnom modeli a v reálnom dopravnom systéme nájsť aj rozdiel, a to z hľadiska obsahu rozhodovania. Pri simulácii dopravného procesu možno rozlišovať dve kategórie rozhodnutí, a to prvotné a druhotné.

Prvotné rozhodovanie zodpovedá rozhodovaniu človeka v reálnom systéme. Ako príklad si môžeme vziať výber nástupištnej koľaje pre vlak prichádzajúci do stanice v čase, keď „jeho“ koľaj daná plánom obsadenia koľají nie je z nejakého dôvodu práve k dispozícii.

Druhotné rozhodovanie zahŕňa stavy, ktoré sa v reálnom systéme zvlášť nevyčleňujú ako problematické, považujú sa za jednoduché, ba až triviálne, prípadne jasne vyplývajú z kontextu situácie. Na rozhodnutia postačí inteligencia výkonných prvkov. V simulačnom modeli však aj takéto rozhodovanie treba z implementačných dôvodov brať do úvahy, pričom sa stáva súčasťou riadiaceho podsystému. Do tejto kategórie patria väčšinou rozhodovacie situácie o výbere prostriedku z disponibilnej množiny a jeho priradení požiadavke, napríklad pridelenie jedného z vozmajstrov na vykonanie technickej prehliadky súpravy vlaku. V realite sú zväčša pracovníci na



Obr. 1.3 Fázy rozhodovacieho cyklu.

.

obsluhu konkrétnych vlakov určení vopred, a v momente, keď sa má prehliadka začať, dotyčný pracovník nastúpi do činnosti bez toho, aby riadiaci pracovník prijímal konkrétne rozhodnutie. V modeli má riadiaci podsystém k dispozícii skupinu všetkých pracovníkov v stanici, z nich vyberie podskupinu pracovníkov určených na danú činnosť a z nich vyberie toho, kto v danom momente nepracuje a je pripravený na danú činnosť.

1.1.5 Kvalita rozhodnutí

V riadiacom systéme postavenom na rozhodnutiach človeka dokáže operátor spozorovať odchýlky, ktoré by mohli neskôr viesť k uviaznutiu. Sleduje systém a vykonáva nevyhnutné operácie tak, aby fungoval systém v stave pokiaľ možno najbližšom k plánovanému. Využíva na to platné pravidlá a normatívy, ako aj svoje empirické skúsenosti a intuíciu pre riešenie konkrétnej konfliktnej situácie.

Pri automatizácii systému riadenia s pomocou počítačovej logiky treba vytvoriť nástroje na

- sledovanie systému a zbieranie informácií,
- tvorbu návrhov riešenia a výber z nich a
- vykonávanie operácií realizujúcich rozhodnutia.

Tieto nástroje sú pravdaže vopred definované človekom vo forme algoritmov. Algoritmus má potom v stave, ktorý rozpoznal z dostupných informácií, navrhnúť varianty riešenia a nájsť rozhodnutie.

Pri posudzovaní kvality rozhodnutí možno vychádzať z viacerých kritérií. Nie je jednoduché ich úplne presne definovať, a tak v najbližšom texte snáď pre predstavu postačí táto približná definícia: za hlavné kritérium kvality rozhodnutia považujem mieru rozdielnosti dosiahnutého stavu od plánovaného stavu systému pre daný časový bod a najbližšiu budúcnosť. Miera rozdielnosti zahŕňa rozdiely vo vybraných parametroch riadeného systému. Najbližšou budúcnosťou možno podľa kontextu uvažovať časové obdobie, kým sú zákazky dotknuté odchýlkami v danom stave prítomné v systéme.

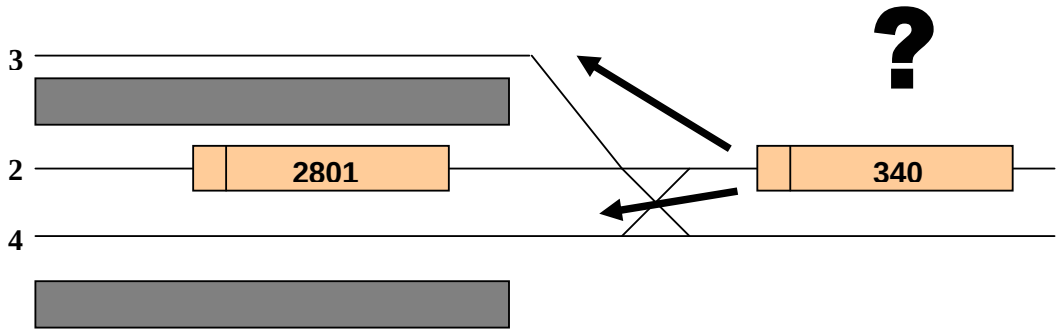
Kvalita rozhodnutia automatizovaného systému závisí od

- dostatočného množstva a kvality spracovaných informácií,
- blízkosti modelového k skutočnému systému a
- kvality rozhodovacieho algoritmu.

Čo sa týka kvality **spracovaných informácií**, algoritmus musí mať k dispozícii určitú množinu základných údajov, ktoré sú nevyhnutné pre rozhodnutie. K tomuto základu môže prijať ešte ďalšie informácie, ktoré môžu prispieť ku kvalitnejšiemu rozhodnutiu.

Pokiaľ je rozhodovací systém v simulačnom modeli, dôležitú úlohu zohráva prítomnosť potrebných prvkov, ktorých sa tieto informácie týkajú, a presnosť ich modelovania.

Pokiaľ ide o **rozhodovacie algoritmy**, tie musia byť minimálne schopné pripraviť



Obr. 1.4 Príklad konfliktnej situácie na železnici a možností jej riešenia.

požadované rozhodnutie. V záujme kvalitnejšieho riešenia by mal byť algoritmus dostatočne prepracovaný a môže obsahovať pri výbere riešenia mechanizmy na spracovanie ďalších informácií a vyhodnotenie ďalších kritérií, ktoré nie sú nevyhnutné v základnej verzii. Pri tvorbe algoritmov zohráva kľúčovú úlohu presná interpretácia pravidiel a postupov z reálneho systému, ktoré bývajú nezriedka vymedzené príliš zložito alebo vágne.

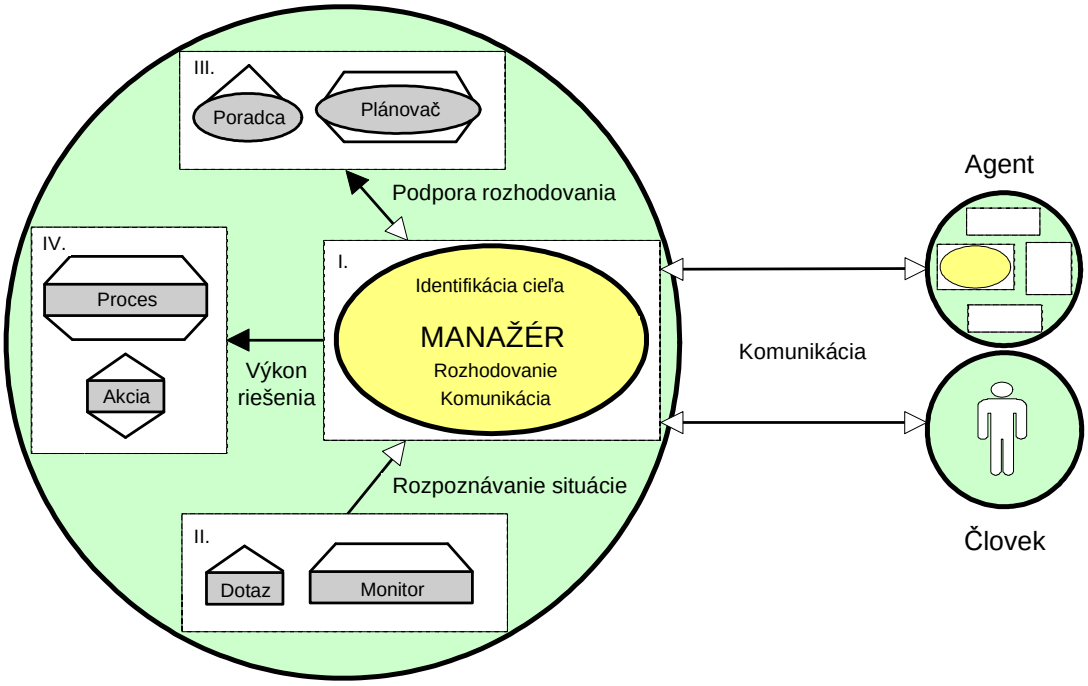
Ilustračný príklad sa nachádza na Obr. 1.4. Osobný vlak číslo 340 prichádza zmeškaný do stanice. Podľa plánu má použiť nástupištnú koľaj č. 2. Avšak v čase zmeškaného príchodu už túto koľaj zaberá vlak číslo 2801. Na riešenie tohto konfliktu treba použiť ďalšiu koľaj. Rozhodovací algoritmus nájde na základe známych dĺžok koľají a ich určenia dve možnosti riešenia: nástupištné koľaje 3 a 4. Tieto informácie (zoznam ďalších použiteľných koľají, ich dĺžka a určenie) sú nevyhnutným minimom na vyriešenie situácie. Jednoduchý algoritmus môže vybrať prvú položku v zozname a prideliť ju vlaku, a tak by vlak č. 340 použil koľaj č. 3. Avšak dôsledkom tohto rozhodnutia sa môže táto evidentná odchýlka od plánu rozšíriť na ďalšie vlaky a koľaje, lebo použitie koľaje č. 3 môže ovplyvniť spracovanie iného vlaku, pre ktorý je plánovaná, alebo prípojov, kam majú prestúpiť cestujúci. Iné rozhodnutie, v tomto prípade použitie koľaje č. 4, by mohlo mať zasa iné dôsledky. Tiež by mohlo rozšíriť odchýlku od plánu do ďalších častí modelu ako prvá alternatíva, a to možno vo väčšej, možno v menšej miere. Jednoduchý algoritmus neporovnáva dôsledky až tak podrobne.

Ak sú k dispozícii ďalšie informácie o dôsledkoch a rozhodovací algoritmus ich dokáže spracovať, môže vybrať také riešenie, ktoré privedie odchýlku najbližšie k ukončeniu v blízkej budúcnosti. V tomto príklade môže použiť informácie z plánu obsadenia nástupištných koľají 3 a 4 v najbližších 10 minútach (ak vlak č. 340 dovtedy koľaj opustí), preveriť, ako ovplyvnia upravené cesty vlakov cez stanicu ďalšie pohyby v nej a prípadne tiež zistiť, aké väzby má meškajúci vlak na prípoje, čaty alebo posunovacie rušne. Tieto informácie sa môžu porovnať s ďalšími údajmi použitia nástupištných koľají a pohybov v stanici plánovaných v blízkej budúcnosti. Výsledky takéhoto hodnotenia môžu prispieť k výberu vhodnejšej z dvoch alternatív. (Viac o formalizácii tohto problému možno nájsť v publikácii (Bažant-Žarnay, 2005/2).)

1.1.6 Implementácia automatického riadenia

Implementácia rozhodovania v systéme automatického riadenia v diskrétnej simulácii závisí od použitej simulačnej architektúry. Pre konštrukciu diskrétnych simulačných modelov sa používajú architektúry založené na udalostiach, aktivitách, stavoch, procesoch, správach alebo agentoch. Prvé štyri sa používajú najmä pre modely pomerne jednoduchých systémov, ktoré nevyžadujú zložitý riadiaci podsystém. Riadenie je zvyčajne zakódované v schéme vykonávania simulácie. Napríklad pri udalostne orientovanej simulácii sa riadi samotnými udalosťami, ktoré vytvárajú simulačné správanie modelu.

Pre simulačné modely komplexných systémov sa používa správovo orientovaná architektúra alebo jej pokročilejšia verzia založená na agentoch. Na nich sa zakladá aj príklad existujúcej implementácie riadenia, ktorý sa použil pri vývoji nástroja na simuláciu procesov v uzloch železničnej siete. Celý prístup je viac opísaný v publikácii (Kavička et al., 2005).

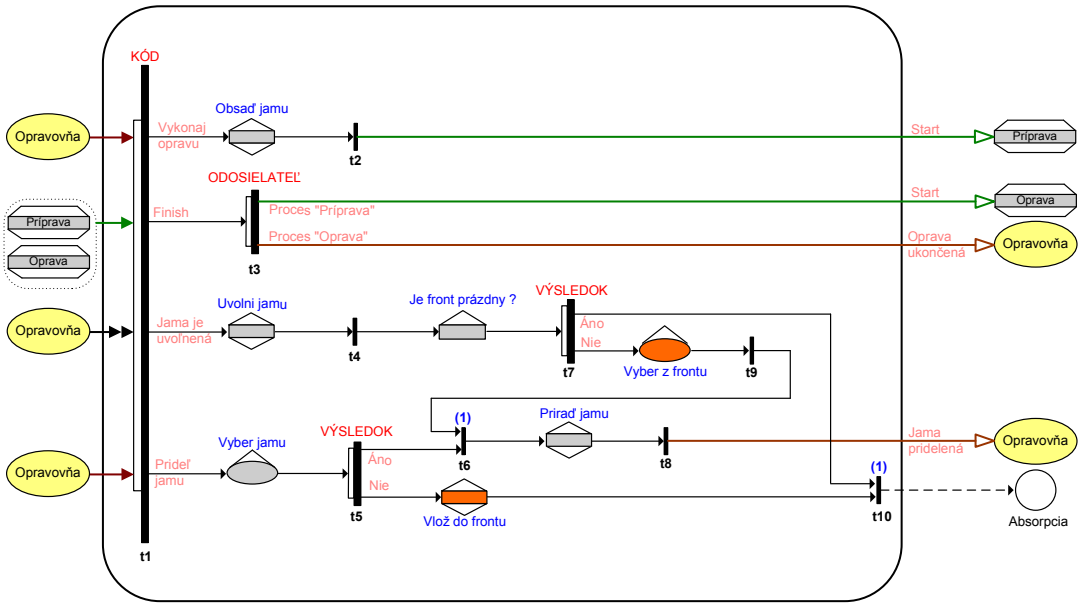


Obr. 1.5 Agent a jeho komponenty a vzťahy k prostrediu.

V tejto špecifickej architektúre, ktorá sa volá **ABASim**, sú agenti definovaní ako reaktívni, t. j. reagujú na podnety zo svojho okolia. Každý agent pozostáva zo štyroch skupín riadiacich komponentov (Obr. 1.5), ktoré vykonávajú vyššie naznačené fázy rozhodovacieho procesu (Obr. 1.3). Patria tam

- manažér – schopnosť rozhodovania,
- senzory – sprístupňovanie informácií,
- riešitelia – poskytovanie návrhov riešenia problémov,
- efektory – vykonávanie rozhodnutí.

Jednoduchý systém sa môže skladať iba z jedného agenta. Zvyčajne však táto architektúra modeluje komplexné systémy, ktoré sa vybudujú na viacerých agentoch, a tí sú usporiadaní hierarchicky a spolupracujú prostredníctvom správ. Správy si posielajú po definovaných komunikačných linkách. Každý z agentov má určenú svoju vlastnú oblasť činnosti, a tým aj automatické riadenie je rozptýlené na viacerých miestach v modeli.



Obr. 1.6 Príklad ABAgrafu pre manažéra agenta simulačného modelu.

Na definíciu správania manažéra sa dá použiť tzv. **ABAgraf**, ktorý je podtriedou Petriho siete (Obr. 1.6). Jeho úlohou je prehľadne zaznamenať všetky definované rozhodovacie a komunikačné činnosti manažéra a následnosti medzi nimi. Na uvedenom obrázku vidno vo forme vrcholov rôznych tvarov ďalšie komponenty agenta, ku ktorému manažér patrí a v elipsách manažéra agenta opravovne, s ktorým tento manažér komunikuje. Rozhodovanie sa vykonáva na rozhodovacích hradlách – to sú hrubé zvislé čiary, no iba tie, z ktorých vychádza viac hrán.

Pre bližší opis rozhodovania v ABAgrafe manažéra agenta odporúčam publikáciu (Kavička et al., 2005). Viac o Petriho sieťach a ich použití v riešení tejto práce sa nachádza v ďalších kapitolách.

Záverom možno zhrnúť, že uvedený príklad architektúry simulačného modelu ukazuje do značnej miery dekomponovaný automatický riadiaci podsystém, ktorý umožňuje prijímať rozhodnutia na lokálnej úrovni a čiastočne svojím návrhom navádza dizajnéra modelu riešiť problémy lokálne, čo nie je vždy nevyhnutnou výhodou. Niekedy sa vyskytujú riadiace otázky na úrovni presahujúcej kompetencie jedného alebo niekoľkých agentov a vtedy stojí za úvahu integrovať do tejto inak veľmi flexibilnej architektúry riešenie takýchto problémov.

1.2 Identifikácia problému

1.2.1 Analýza rozhodovacích situácií

Pri analýze činnosti vyššie uvedeného simulačného modelu som rozpoznal určité rozhodovacie situácie, ktoré sa dajú rozdeliť na skupiny podľa charakteristiky:

a) Priradenie prostriedkov požiadavke.
b) Určenie času.
c) Úprava technologického postupu.

Priraďovanie prostriedkov požiadavkám je základná a najčastejšia rozhodovacia situácia v ľubovoľnom obslužnom systéme, dopravný nevynímajúc. V ňom sa týka obidvoch skupín prostriedkov, pevných i pohyblivých.

Každá objednávka v dopravnom systéme musí mať po celý čas svojej existencie pridelený minimálne jeden prostriedok z pevného podsystému pre svoju obsluhu, a ním je dopravná cesta (napr. vodná cesta, plavebná komora, vozovka, skladovacie miesto kontajnera, koľaj, letový koridor a pod.). Mobilné prostriedky (napr. remorkér, ťahač cestných návesov, kontajnerový prekladač, posunovací rušeň, obslužný personál) sa prideľujú v priebehu technologických procesov podľa ich požadovaných parametrov, a je prípustný aj stav, že zákazka nemá priradený ani jeden pohyblivý prostriedok.

Zvyšok situácií sa týka zmien v technologických procesoch v dopravnom systéme, ktoré závisia od aktuálnych podmienok v modelovanom systéme. Zmeny môžu byť buď určením času vykonania úkonu alebo úpravou technologického postupu. Pomôžem si niekoľkými príkladmi z modelov železničných staníc.

Skupina rozhodovacích situácií, kde sa **určuje čas** vykonania úkonu alebo postupu, sa takisto týka priraďovania prostriedkov. Otázka však nestojí tak, že ktorý prostriedok komu priradiť, ale tak, že kedy daný prostriedok priradiť. Inými slovami, ako usporiadať požiadavky tak, aby sa z hľadiska výkonu v systéme postupovalo najúčinnejšie.

Ak existuje skupina vlakov čakajúcich na rozradenie vo vchodovej skupine, môže byť užitočné určiť ich poradie rozradenia tak, aby sa najskôr skompletizovali tie východiskové vlaky v smerovej skupine, ktoré čakajú na splnenie svojho limitu zhromažďovania. Vo veľkých zriaďovacích staniciach s veľkými vstupnými i výstupnými tokmi vozňov a vysokou priepustnosťou pahorka sa môže potreba rýchleho skompletizovania východiskových vlakov stať dôležitým príspevkom k dosiahnutiu vyššej efektívnosti systému.

Čas spustenia spracovania východiskového vlaku vytvoreného z vozňov zhromaždených na smerovej koľaji závisí vo veľkých zriaďovacích staniciach nielen od grafikonu, ale aj od aktuálneho stavu čiat a dĺžky vlakovej súpravy. Čím je viac vozňov v súprave, tým bude súprava vyžadovať viac času na technologické spracovanie (technická prehliadka, prepravná prehliadka, atď.) a tým skôr sa musí začať s obsluhou vlakovej súpravy, aby bola ukončená do času odchodu.

V niektorých železničných systémoch je dovolené posielať nákladné vlaky s náskokom oproti ich času odchodu podľa grafikonu. Poznajúc, že staničné zhlavie potrebné pre odchod určitého vlaku môže byť vplyvom náhodných udalostí v čase odchodu podľa grafikonu obsadené, riadiaci systém sa môže rozhodnúť poslať pripravený vlak s náskokom pred plánovaným odchodom. Podobne v iných situáciách (plán práce čiat alebo posunovacích rušňov) môže poznanie stavu v systéme v blízkej budúcnosti pomôcť pri efektívnejšom rozhodnutí v prebiehajúcom čase.

Typickou situáciou časového nastavenia v osobných staniciach je synchronizácia prípojov medzi osobnými vlakmi, keď dôjde k oneskoreniu aspoň jedného z nich oproti grafikonu. Železničné pravidlá určujú, ktoré prípoje majú čakať a ako dlho. Čakacie časy sa tiež stanovujú podľa aktuálneho stavu v stanici (počet prítomných cestujúcich, počet cestujúcich presúvajúcich sa medzi prípojmi, atď.). Inou situáciou je pridelenie náhradnej koľaje oneskorenému vlaku, ktorý nemôže ísť na plánovanú koľaj pri svojom príchode, ako som už spomenul v príklade v kap. 1.1.5.

Poslednou vytýčenou kategóriou situácií sú **úpravy technologického postupu** spracovania zákazky podľa podmienok v systéme. Pre spracovanie existuje, ako už bolo spomenuté, istý vopred stanovený plán, ktorý sa má za normálnych okolností realizovať. Ak sa však okolnosti (vplyvom náhodných javov) zmenia natoľko, že sa pôvodný plán použiť nedá, je možná alebo nevyhnutná jeho úprava.

Aj táto kategória súvisí s prideľovaním prostriedkov požiadavkám, avšak to sa rieši až v druhom kroku, po vykonaní úpravy technologického postupu.

V železničnej stanici je takáto situácia typická napríklad pri skladbe vlakovej súpravy zo skupín vozňov z rôznych vlakov. Jednotlivé vlaky, ktoré poskytujú skupiny vozňov sa nemusia v stanici nachádzať pri každom výskyte vytváraného vlaku. V takom prípade treba upraviť technologický postup obsluhy vlaku tak, aby sa poskladal iba zo skupín vozňov, ktoré sú prítomné: upraví sa cesta posunovacieho rušňa, poradie vyzdvihnutia skupín vozňov a pod.

Iný jednoduchý, no potrebný príklad modifikácie technologického procesu nastáva pred vyhlásením východiskového vlaku na relačnej koľaji zriaďovacej stanice: ak zhromaždené vozne stoja oddelene s priveľkými medzerami, treba ich stlačiť dohromady posunovacím rušňom pred spracovaním vlaku. V prípadoch, ak sú medzery minimálne, tieto operácie nie sú potrebné.

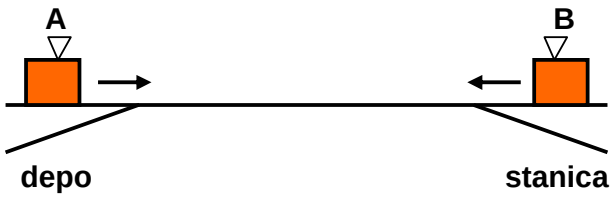
Posledný príklad úprav technologických postupov sa týka stavania ciest pre pohyb vlakov, resp. vlakových súprav v stanici. Ak pri pravidelnom presune skupiny vozňov koľajiskom stanice musí dôjsť k úvraťovej jazde, môže sa koľaj pre úvrať vybrať podľa dĺžky skupiny a aktuálnej situácie v koľajisku. Podobne v situácii triedenia zatláčaním vozňov môžu pohyby triedenej vlakovej súpravy na relačné koľaje závisieť od dĺžky skupiny vozňov určených pre každú z jednotlivých koľají tak, aby vyčlenené vozne stáli čo najbližšie ku koncu svojej relačnej koľaje, cez ktorý sa na ňu dostali.

Hoci vymenované príklady pochádzajú z prostredia železničnej stanice, pri poznaní technológie iných dopravných systémov si možno v nich predstaviť situácie, ktoré sa uvedeným princípom podobajú.

1.2.2 Uviaznutie

Po vykonanej analýze môžem prejsť k uvedeniu problému, ktorý sa pri automatickom rozhodovaní v uvedených situáciách môže vyskytnúť. Je ním tzv. uviaznutie, čo je zablokovanie prebiehajúcich procesov v systéme. Je to teda stav, keď proces v systéme čaká na jeden alebo viac prostriedkov, ktoré používa iný proces a súčasne tento iný proces čaká na prostriedky súčasne používané prvým procesom. Ani jeden z procesov nemôže pokračovať, pretože čakajú jeden na druhého. Stav uviaznutia môže nastať aj medzi viac ako dvoma procesmi, ktoré používajú rovnaké prostriedky.

Ilustračný príklad je na Obr. 1.7: rušeň A sa pohybuje z rušňového depa k vlaku v stanici, kam bol pridelený. V rovnakom čase sa pohybuje rušeň B zo stanice do rušňového depa. Obidva rušne majú použiť rovnakú posunovú cestu medzi depom a stanicou, jeden v jednom smere, druhý v opačnom smere. Každému z nich sa cesta stavia po častiach postupne, ako sa pohybujú. Obidva sa pohybujú súčasne, až kým



Obr. 1.7 Ilustrácia uviaznutia.

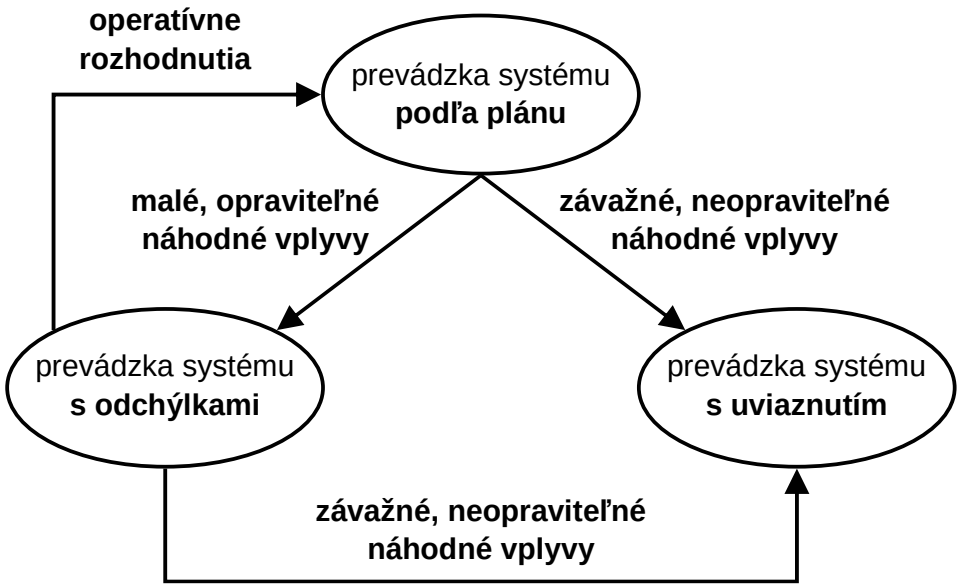
prídu do bodu, keď najbližšia časť cesty rušňa A je obsadená rušňom B a najbližšia časť cesty rušňa B je obsadená rušňom A. Obidva rušne sa zastavia, aby počkali na pridelenie požadovaných prostriedkov. A ak nemajú inú možnosť na dokončenie svojho pohybu než s požadovanou sekciou, môžu čakať navždy – nachádzajú sa v uviaznutí.

Ako som už naznačil, prevádzka v dopravných systémoch a aj ich simulačných modeloch sa **vopred plánuje**. Plán predpokladá časové okamihy, kedy sa jednotlivé udalosti majú stať a obsahuje reakcie, ktoré by mali nasledovať za udalosťami v kontexte aktuálneho stavu.

Kvôli náhodným vplyvom sa však udalosti nevyskytujú vždy presne podľa plánu, ale v určitých odchýlkach od plánu. Najčastejšie ide o odchýlky v čase alebo vo výbere použitých prostriedkov, menej často o odchýlky v plánovanom

technologickom postupe. V takej situácii je úlohou riadiaceho systému odchýlku zvládnuť a priviesť aktuálny stav v systéme do súladu s plánom. Možno tak urobiť buď úpravou plánu k aktuálnej realite alebo naopak – vykonaním takých akcií, že sa systém vráti do plánovaného stavu. Ak je dostupný dostatok prostriedkov (časové rezervy, dostupný personál alebo zariadenie), riadiaci systém môže zvládnuť odchýlky a priviesť systém naspäť k plánu. Ak niektorý z požadovaných prostriedkov zlyhá, odchýlka môže rásť a zmeniť sa prípadne až na uviaznutie (Obr. 1.8).

V železničnej stanici je napríklad plán reprezentovaný grafikonom vlakovej dopravy, ktorý predpisuje odchody a príchody vlakov. Na základe neho sú zostrojené plány zodpovedajúcich technologických procesov a obsluhy cestujúcich. Takým spôsobom sa staniční zamestnanci dozvedia, kedy treba vlakové súpravy presúvať, čistiť alebo kontrolovať. Každý zamestnanec má individuálny pracovný plán, ktorý obsahuje zoznam časových údajov s číslami vlakov, na ktorých má pracovať. Náhodné vplyvy spôsobia, že vlaky môžu meškať, nákladné vlaky môžu obsahovať viac naložených vozňov alebo nejaké vozne na špeciálnu manipuláciu, prípadne



Obr. 1.8 Náhodné vplyvy a operatívne riadenie v dopravnom systéme.

nečakané poruchy môžu spôsobiť zdržanie technologických operácií a pod.

Odchýlky v tomto príklade by mohli byť: použitie koľaje, ktorá sa pre daný vlak neplánovala, meškanie v spracovaní vlaku alebo použitie inej čaty pre určitú operáciu, než je v pôvodnom pláne.

1.2.3 Riešený problém

Vyriešenie problému uviaznutia je evidentne základnou úlohou pri implementácii automatického riadenia v dopravnom systéme. Jeho pripustením hrozí, že sa systém dostane do stavu, keď bude treba použiť prídavné operácie na ošetrenie stavu, čo spolu s plytvaním času znižuje efektivitu systému. V riadiacom systéme si to vyžaduje navyše implementovať postup ošetrenia takého stavu. A ak som konštatoval, že pri zložitejších riadiacich úlohách je algoritmizácia ľudského myslenia zložitým problémom, tak tvorba ošetrenia uviaznutia je ešte zložitejšia, ba možno snáď povedať, že sa riešenie uviaznutia bez ľudského zásahu nezaobíde. Aj preto treba venovať dostatočnú pozornosť tomuto problému.

Ak sa pozrieme na kategorizáciu rozhodovacích situácií v dopravnom systéme v kapitole 1.2.1, tak môžeme konštatovať, že uviaznutie sa priamo týka iba prvej kategórie – priradenie prostriedku požiadavke. Ďalšie dve kategórie, určenie času a úprava technologického postupu, súvisia s uviaznutím iba sprostredkovane, nakoľko k rozhodovaniu o priraďovaní prostriedkov dochádza až následne po inom rozhodnutí z týchto kategórií.

Možno súhrnne konštatovať, že implementácia rozhodovania pre druhú a tretiu kategóriu situácií ovplyvní kvalitu riadenia skôr v parametroch efektívnosti a účinnosti fungovania systému, zatiaľ čo algoritmy pre prvú kategóriu situácií sú veľmi dôležité pri určení, či môže riadiaci podsystém fungovať samostatne, bez zásahu (a iba pod dohľadom) človeka. Aj preto som sa rozhodol vo svojej práci zamerať práve na riešenie problému uviaznutia v dopravnom systéme. Tento sa z vytýčených úloh javí ako najťažšia úloha s pomerne všeobecným dosahom na ľubovoľný dopravný systém.

2 Súčasný stav riešenej problematiky

V tejto kapitole sa zameriam na teoretické východiská riešenia problému. Prvým východiskom (kapitola 2.1) sú prístupy, ako sa vysporiadať s uviaznutím v ľubovoľnom systéme, kde tento stav môže nastať. Nasleduje opis formalizmov (kapitola 2.2), ktoré sa používajú pri riešení problému v literatúre: systém diskrétnych udalostí, ďalej systém prideľovania prostriedkov ako jeho špeciálny prípad a Petriho sieť ako paradigma na modelovanie uvedených systémov. V kapitole 2.3 je prehľad spôsobov riešenia problému z literatúry, ktorý už bol v čase riešenia čiastočne ovplyvnený analýzou, ako sa možno vysporiadať s uviaznutím v dopravnom systéme, t. j. analýzou použiteľnosti prístupov z kapitoly 2.1, ktorá sa nachádza v úvode vlastného riešenia v kapitole 4.2.

2.1 Riešenie problému uviaznutia

Stav uviaznutia procesov som naznačil v kapitole 1.2.2. Ide o nežiaduci stav, keď do neho zapojené procesy nikdy neskončia svoju činnosť a viažu systémové prostriedky, čím brzdia prácu ďalších procesov a znižujú efektívnosť celého systému. Môže nastať v každom systéme s prideľovaním prostriedkov pri splnení určitých podmienok, ktoré sú uvedené v podkapitole 2.1.1.

Prístupy k riešeniu problému uviaznutia procesov možno principiálne rozdeliť do týchto oblastí:

a) Detekcia a zotavenie z uviaznutia.
b) Prevencia pred uviaznutím.
c) Vyhnutie sa uviaznutiu.
d) Ignorovanie uviaznutia.

V nasledujúcich častiach si prístupy priblížime. Uvedené informácie si môže čitateľ doplniť napríklad v publikáciách (Coffman et al., 1971), (Martincová-Grondžák, 2004) alebo (Silberschatz et al., 2002).

2.1.1 Nutné podmienky pre uviaznutie

Stav uviaznutia procesov môže nastať pri **súčasnom** splnení nasledovných štyroch (Coffmanových) podmienok v danom čase.

- **Vzájomné vylúčenie** – aspoň jeden prostriedok musí byť pridelený výlučne, t. j. nemôže byť zdieľaný medzi viacerými procesmi.
- **Vlastniť a žiadať** – musí existovať proces, ktorý má pridelený aspoň jeden prostriedok a požaduje ďalšie prostriedky, ktoré sú pridelené iným procesom.
- **Používanie bez preempcie** – prostriedok nemôže byť odňatý, t. j. proces môže uvoľniť prostriedok iba dobrovoľne, keď ho už nepotrebuje.
- **Kruhové čakanie** – musí existovať množina $\{P_0, P_1, P_2, ..., P_n\}$ čakajúcich procesov takých, že P_0 čaká na prostriedok, ktorý drží P_1, P_1 čaká na prostriedok

pridelený P_2, ..., P_{n-1} čaká na prostriedok pridelený P_n a P_n čaká na prostriedok, ktorý drží P_0.

2.1.2 Graf prideľovania prostriedkov

Ide o orientovaný graf, ktorého množina vrcholov pozostáva z dvoch podmnožín: množiny všetkých procesov v systéme $\{P_0, P_1, ..., P_n\}$ a množiny všetkých typov prostriedkov systému $\{R_0, R_1, ..., R_m\}$.

Jeho množina hrán má dva druhy orientovaných hrán, a to

- $P_i \rightarrow R_j$ – orientovaná hrana z vrcholu P_i do vrcholu R_j znamená, že proces P_i požiadal a čaká na prostriedok R_j a
- $R_j \rightarrow P_i$ – orientovaná hrana z vrcholu R_j do vrcholu P_i znamená, že prostriedok R_j je práve pridelený procesu P_i.

Keď proces požiada o prostriedok, do grafu sa vloží hrana, ktorá smeruje od procesu k prostriedku. Ak sa dá požiadavka uspokojiť, hrana sa zmení na hranu opačnej orientácie, t. j. od prostriedku k procesu. Keď proces uvoľní prostriedok, hrana sa zmaže.

Graf sa používa na detekciu kruhového čakania, ktoré je jednou z podmienok vzniku stavu uviaznutia. Preto sa mu hovorí aj čakací graf.

2.1.3 Detekcia a zotavenie z uviaznutia

Tento prístup (v anglicky písanej literatúre označovaný ako *deadlock detection and recovery*) prichádza do úvahy vtedy, keď v systéme neexistujú metódy na zabránenie uviaznutiu. V takom prípade obsahuje systém algoritmus na detekciu stavu uviaznutia a algoritmus na zotavenie systému.

Algoritmus detekcie preskúma aktuálny stav systému a určí, či nastalo uviaznutie. Všeobecne vytvorenie algoritmu závisí od toho, či sa procesom prideľuje iba jedna jednotka z každého typu prostriedkov alebo viac jednotiek.

V prvom prípade sa na detekciu môže použiť čakací graf, ktorý je variantom grafu prideľovania prostriedkov. Vytvorí sa vypustením vrcholov pre typy prostriedkov a náhradou príslušných hrán tak, že vznikne orientovaný graf, kde orientovaná hrana $P_i \rightarrow P_j$ symbolizuje vzťah čakania procesu P_i na proces P_j, aby vrátil prostriedky požadované procesom P_i. Existencia cyklu v tomto grafe potom zaručí existenciu uviaznutia v systéme.

V druhom prípade sa na detekciu môže použiť algoritmus podobný algoritmu bankára, ktorý je opísaný v kapitole 2.2.2.

Algoritmy zotavenia principiálne vychádzajú z dvoch operácií. Buď sa na prerušenie cyklu v čakacom grafe použije ukončenie jedného alebo viacerých procesov, ktoré sú uviaznutím zablokované, alebo sa odníme jeden alebo viac prostriedkov, na ktoré zablokované procesy čakajú. Obidve z týchto operácií spôsobia

v systéme straty, ktorých výška závisí od stavu postihnutých procesov a od ceny, ktorú bude stáť obnovenie operácií alebo celých procesov.

Obsluha uviaznutia pomocou detekcie a zotavenia sa používa najčastejšie tam, kde k uviaznutiam dochádza veľmi zriedka alebo kde cena za zotavenie je nižšia ako by bola cena za dodatočné operácie pri ďalších dvoch prístupoch.

V prípade použitia tohto prístupu v riadení simulácie dopravného systému, možno na zotavenie systému použiť **umelé zdroje** a do simulačného protokolu zapísať zoznam takto riešených situácií. Výhodou tohto riešenia je zbehnutie simulačných behov bez prerušenia do konca, pričom operátor na konci zistí, kde nastali stavy uviaznutia a môže im zásahom do štruktúry simulačného modelu predísť.

2.1.4 Prevencia pred uviaznutím

Preventívny prístup (v anglicky písanej literatúre označovaný ako *deadlock prevention*) vychádza zo štyroch podmienok uviaznutia uvedených vyššie. Na to, aby sa stav uviaznutia nevyskytol, treba zaručiť, aby bola sústavne porušená aspoň jedna z nich.

Ak chceme porušiť podmienku **vzájomného vylúčenia**, musíme zaručiť, že všetky prostriedky v systéme sa môžu zdieľať medzi viacerými procesmi.

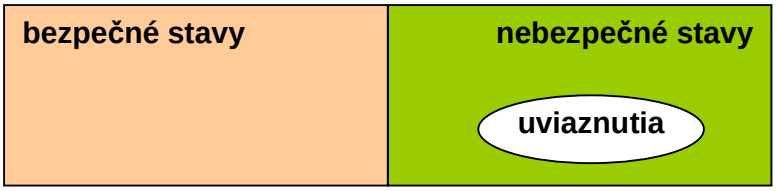
Druhú podmienku **vlastniť a žiadať** možno odstrániť tým, že pri žiadosti o prostriedok nebude mať proces priradený iný prostriedok. To sa dá dosiahnuť dvoma spôsobmi: buď každý proces požiada o všetky potrebné prostriedky naraz napríklad na začiatku svojej činnosti alebo vždy pred požiadaním o prostriedok uvoľní všetky, ktoré má pridelené.

Tretia podmienka **zákazu preempcie** sa poruší tým, že sa umožní odoberanie prostriedkov procesom. Stačí to zariadiť tak, aby sa prostriedky odoberali iba v nevyhnutných prípadoch, keď by mohlo dôjsť k stavu uviaznutia – od procesov čakajúcich na iné prostriedky.

Platnosť podmienky **kruhového čakania** možno odstrániť pravidlom, že procesy budú požadovať prostriedky podľa vzostupného poradia číslovania. Prostriedky sa zoradia do rozumného poradia (podľa toho, v akom poradí bývajú zväčša procesom prideľované) a každý proces bude mať dovolené dostať iba prostriedok s vyšším poradím, než je najvyššie poradie prostriedkov, ktoré už má. V prípade, že by potreboval prostriedok s nižším poradím, musel by uvoľniť najskôr všetky prostriedky, ktoré majú vyššie poradie, než ten, čo je požadovaný.

Pokiaľ je možné v štruktúre systému zaručiť porušenie aspoň jednej zo štyroch podmienok, tak je problém uviaznutia vyriešený. Avšak naznačené pravidlá majú vplyv na efektívnosť fungovania systému, čo je nevýhodou tohto prístupu.

2.1.5 Vyhnutie sa uviaznutiu



Obr. 2.1 Vzťah stavov v stavovom priestore systému z hľadiska bezpečnosti.

V prípade, že nie je možné zaistiť prevenciu pred uviaznutím v štruktúre systému, možno si vybrať spôsob, ako sa vyhnúť uviaznutiu (v anglicky písanej literatúre označovaný ako *deadlock avoidance*), ktorý navyše využije informácie o stave systému. Podstatou algoritmov v tejto kategórii je rozlíšiť, či daný stav systému je bezpečný alebo nie (Obr. 2.1). Algoritmy sa líšia v tom, koľko a aké informácie o stave systému potrebujú.

Stav pridelenia prostriedkov je bezpečný, ak existuje postupnosť spracovania procesov, ktorá umožní prideliť každému procesu všetky ním požadované prostriedky a dovolí ukončiť spracovanie všetkých procesov (vyhne sa stavu uviaznutia). V prípade, že stav po pridelení požadovaného prostriedku nie je identifikovaný ako bezpečný, prostriedok sa nepridelí a žiadajúci proces musí počkať, kým sa nezmení stav systému.

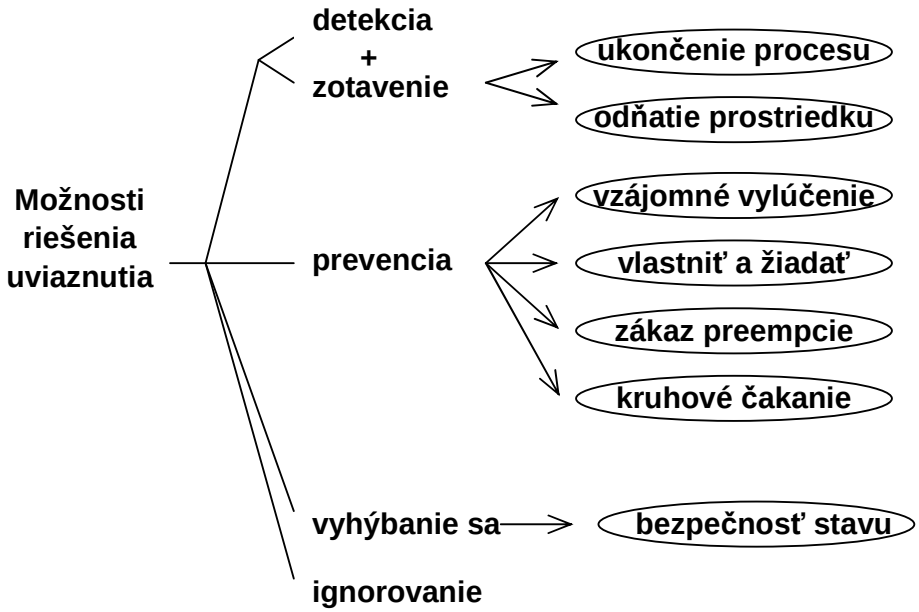
Tieto algoritmy pracujú počas behu systému, to znamená, že ich vykonanie si vyžaduje istý čas a výpočtový priestor popri bežiacom systéme. Ďalšou ich nevýhodou môže byť, že nezachytia všetky bezpečné stavy, ale niektoré z nich označia za nebezpečné, čo v praxi znamená, že systém nebude pripustený do stavu, z ktorého nehrozí uviaznutie, a tým sa zníži efektivita fungovania systému. Aj napriek týmto nevýhodám však môže algoritmus zabezpečiť zo všetkých prístupov najefektívnejšie využitie systémových prostriedkov pri zamedzení stavu uviaznutia.

2.1.6 Ignorovanie uviaznutia

Okrem troch uvedených prístupov sa dá použiť ešte jeden spôsob: vôbec uviaznutie neriešiť – tváriť sa, že sa také stavy v systéme nikdy nevyskytnú.

Tento spôsob sa dá akceptovať v systémoch, kde sa tieto stavy vyskytujú s minimálnou pravdepodobnosťou a kde sa dá systém znovu naštartovať bez väčších strát. Je zaujímavé, že práve v oblasti operačných systémov, kde teória o stavoch uviaznutia a ich riešení vznikla, sa v súčasnosti tento prístup používa zo všetkých štyroch najviac (Silberschatz et al., 2002, str. 249).

Prebraté možnosti riešenia stavu uviaznutia sumarizuje schéma na Obr. 2.2. Z troch prístupov riešiacich stav uviaznutia je pre systém najreštriktívnejšou prevencia pred uviaznutím. Naopak najviac voľnosti poskytuje prístup detekcie uviaznutia, no za



Obr. 2.2 Možnosti riešenia uviaznutia všeobecne.

cenu zvýšených nákladov na zotavenie systému. Vyhnutie sa uviaznutiu sa nachádza čo do poskytnutia voľnosti niekde medzi nimi za cenu časti výpočtového výkonu počas behu systému.

2.2 Používané formalizmy

Po úvode k problematike vysporiadania sa s uviaznutím načrtnem v tejto kapitole formalizmy, ktoré sa v literatúre používajú pri riešení problému.

Pri hľadaní spôsobov riešenia bola moja prvotná orientácia okrem dopravného systému aj na ďalšie aplikačné systémy, kde sa tento problém vyskytuje, a to najmä operačné systémy a pružné výrobné systémy. To ma doviedlo k teoretickým formalizmom týchto a ďalších aplikačných systémov: systému diskrétnych udalostí, systému prideľovania prostriedkov a Petriho sieti.

2.2.1 Systém diskrétnych udalostí (DES)

Systém diskrétnych udalostí (Discrete Event System, DES) je teoretickým základom pre pohľad na riešenie uviaznutia používaný v jednej skupine publikácií, a zároveň je aj formálnym rámcom pre špeciálny systém opísaný v ďalšej kapitole.

DES možno formálne považovať za dynamický systém, ktorý má stavový priestor a štruktúru stavov a prechodov (Wonham, online). DES obzvlášť

- je **diskrétny** v čase a (zvyčajne aj) v stavovom priestore;
- je **asynchrónny**, resp. vedený udalosťami inými, než je tikanie hodín; a
- môže byť nedeterministický.

Typicky sa DES modeluje **konečnými automatmi**, t. j. triedou grafov, ktoré pozostávajú z vrcholov reprezentujúcich stavy systému a orientovaných hrán označujúcich udalosti, spôsobujúce prechody medzi stavmi.

Takýmito všeobecnými princípmi sa vyznačuje široká oblasť aplikačných systémov, ako sú dopravné a logistické (obslužné) systémy, výrobné systémy, systémy riadenia bázy dát alebo operačné systémy.

Čitateľovi je už asi zrejmé, že všetky dosiaľ rozoberané systémy možno považovať za DES. V kapitole 1.1.1 som už inými slovami opísal analyzovaný dopravný systém ako štruktúru stavov a prechodov medzi nimi a neskôr bolo spomenuté uvažovanie aj náhodných vplyvov, t. j. nedeterministickosť systému.

Hoci to dosiaľ nebolo slovne uvedené, v kontexte uvažovaného dopravného systému predpokladám aj diskrétnosť udalostí v čase. Aj keď činnosti premiestnenia alebo obsluhy môžu byť kategorizované ako spojité procesy, mňa pri riešení úlohy zaujímajú iba diskrétne zmeny stavu systému, t. j. začiatok a koniec spojitých činností. Asynchrónnosť v dopravnom systéme vyplýva z udalostí, ktoré sa vo všeobecnosti neviažu na nijaké takty hodín.

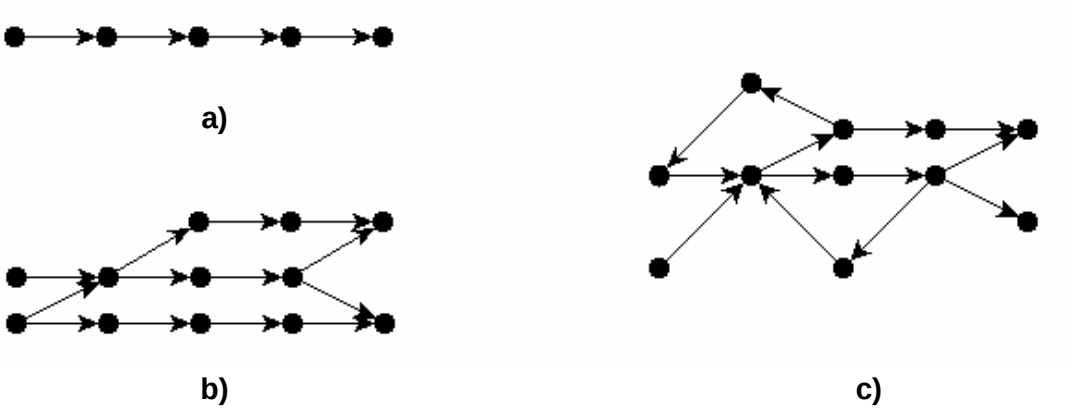
Predmet tejto práce, dopravný systém, spadá do kategórie obslužných systémov, t. j. zaoberám sa podtriedou DES, v ktorej sa identifikujú zákazky (požiadavky) a prostriedky na ich uspokojenie. Parametre stavov tejto podtriedy DES zahŕňajú stav pridelenia prostriedkov a stav prebiehajúcich technologických činností. Činnosti slúžia na prechod dopravného systému z jedného stavu do druhého.

2.2.2 Systém prideľovania prostriedkov (RAS)

Pojmom systém prideľovania prostriedkov (Resource Allocation System – RAS) sa označuje v literatúre systém pozostávajúci z množiny **súbežne prebiehajúcich procesov**, ktoré v určitých fázach na úspešné dokončenie svojho vykonávania požadujú **výlučné použitie** určitého počtu **systémových prostriedkov** (Peterson, 1981). Prostriedky sú **obmedzené** a charakterizujú sa ako **opätovne použiteľné**, keďže ich prideľovanie a uvoľňovanie žiadajúcim procesom nemení ich povahu ani množstvo.

Je evidentné, že týmto teoretickým označením možno zovšeobecniť aj tie aplikačné systémy, ktoré som v texte dosiaľ spomínal: dopravný systém, operačný systém, pružný výrobný systém – za predpokladu dodržania uvedených podmienok pre prostriedky v systéme. Zároveň možno RAS považovať za špeciálny prípad DES, v ktorom sa prechody medzi stavmi vykonávajú jedným z týchto troch spôsobov (Reveliotis et al., 1997, str. 8):

a) Do systému vstúpi nový proces.
b) Jeden z prebiehajúcich procesov postúpi do nasledujúcej fázy (o jeden krok).



Obr. 2.3 Schémy smerovania procesu v RAS: a) úplne usporiadaný, b) čiastočne usporiadaný, c) neusporiadaný.

c) Ukončený proces vystúpi zo systému.

2.2.2.1 Kategórie RAS

Sekvenčný RAS je systém, kde sú všetky procesy sekvenčné, t. j. pozostávajú zo sekvencie krokov prideľovania a uvoľňovania prostriedkov počas vykonania. Každý proces teda možno opísať postupnosťou fáz, kde každá fáza je definovaná podmnožinou prostriedkov potrebných pre jej úspešné vykonanie.

Sekvenčné RAS sa ďalej delia podľa (Tricas, 2003)

- súčasného používania prostriedkov a
- smerovania fáz procesu.

Podľa súčasného používania prostriedkov sa rozlišujú

- **jedno-jednotkový RAS** (Single Unit RAS, SU-RAS) – pri každej fáze spracovania potrebuje proces iba jednu jednotku jedného typu prostriedkov,
- **jedno-typový RAS** – (Single Type RAS, ST-RAS) – pri každej fáze spracovania môže proces potrebovať súčasne niekoľko jednotiek, no iba jedného typu prostriedkov,
- **viac-typový RAS** (Multiple Type RAS, MT-RAS) – pri každej fáze spracovania môže proces potrebovať súčasne niekoľko jednotiek niekoľkých typov prostriedkov.

Podľa smerovania procesov možno rozlíšiť (Obr. 2.3)

- **úplne usporiadaný RAS** (Totally Ordered RAS, TO-RAS) – fázy procesu nasledujú v usporiadanej postupnosti za sebou podľa predpisu,
- **čiastočne usporiadaný RAS** – (Partially Ordered RAS, PO-RAS) – fázy procesu možno pružne smerovať v jednom alebo viacerých bodoch jeho spracovania –

riadiaci prvok môže operatívne zvoliť jeden z variantov ďalšieho spracovania, pri ktorých je postupnosť alokácie a dealokácie prostriedkov rozdielna,

- **neusporiadaný RAS** (Non-Ordered RAS, NO-RAS) – pri spracovaní procesu sa pripúšťa aj možnosť opakovania niektorých fáz v cykle.

Tab. 2.1 Vzťahy medzi vymedzenými kategóriami RAS**.**

usporiadanosť / priradenie prostriedkov		úplne uspor.	čiastočne usporiadaný	neusporiadaný
1-1-ový		**SU-TO-RAS**	**SU-PO-RAS**	**SU-NO-RAS**
1-typový		**ST-TO-RAS**	**ST-PO-RAS**	**ST-NO-RAS**
viac-typový		**MT-TO-RAS**	**MT-PO-RAS**	**MT-NO-RAS**

Podľa vlastností popisovaného systému možno kombinovať príslušné skratky do výsledného označenia, napr. jedno-jednotkový úplne usporiadaný systém označíme ako SU-TO-RAS alebo viac-typový čiastočne usporiadaný systém ako MT-PO-RAS. Čitateľovi je istotne zrejmé, že definované systémy majú medzi sebou vzťahy kategória-podkategória. Prehľadne to zachytáva tab. 2.1.

V niektorých iných, najmä starších publikáciách v tejto oblasti, napr. (Lawley et al., 1998) a (Reveliotis et al., 1997), sa nachádza iná kategorizácia systémov s niektorými podobnými označeniami, a to

- **jedno-jednotkový RAS** (SU-RAS),
- **jedno-typový RAS** (ST-RAS),
- **konjunktívny RAS** (Conjunctive RAS, AND-RAS) – to isté ako viac-typový RAS vyššie,
- **konjunktívno-disjunktívny RAS** (Conjunctive/Disjunctive RAS, AND/OR-RAS) – pri každej fáze spracovania môže proces potrebovať súčasne niekoľko jednotiek niekoľkých typov prostriedkov ako pri konjunktívnom RAS, a navyše je možné pružné smerovanie spracovania procesu,
- **RAS pre systém AGV** (AGV RAS) – kombinácia jedno-jednotkového s disjunktívnym RAS pre systém automaticky riadených vozidiel (Automated Guided Vehicle – AGV), kde vozidlo potrebuje naraz iba jeden úsek siete, t. j. iba jeden prostriedok, a môže operatívne smerovať svoj pohyb po alternatívnych cestách.

Tab. 2.2 Označenie kategórií používaných systémov.

pomenovanie systému	pôvodná skratka	skratka podľa kategórie
jedno-jednotkový	SU-RAS	SU-TO-RAS*
jedno-typový	ST-RAS	ST-TO-RAS*
konjunktívny	AND-RAS	MT-TO-RAS*
konjunktívno-disjunktívny	AND/OR-RAS	MT-PO-RAS
RAS pre systém AGV	AGV RAS	SU-PO-RAS

* 2. časť skratky (TO) - uvedené systémy sa v literatúre vyskytujú s usporiadanými postupnosťami fáz

Vyššie uvedená kategorizácia z (Tricas, 2003) poskytuje prehľadnejšie a presnejšie označenie systémov, preto ju budem v tejto práci používať. Pri používaní odkazov s druhou kategorizáciou môže pomôcť tab. 2.2.

Nesekvenčný RAS je systém, kde existuje aspoň jeden nesekvenčný proces. To je taký proces, v ktorom sa sekvencie krokov prideľovania a uvoľňovania prostriedkov oproti sekvenčnému procesu navyše v jednom alebo viacerých bodoch rozdelia do vetiev a prebiehajú súbežne. Tieto vetvy sa môžu, ale nemusia do konca procesu spojiť naspäť do jednej vetvy, kde by proces mal iba jeden bod ukončenia. V publikácii (Ezpeleta-Recalde, 2004) sa takýto systém vyskytuje pod označením NS-RAS (Non-sequential RAS).

Tu je dobré si uvedomiť, že (ne)sekvenčnosť procesov je ďalšou, v poradí treťou vlastnosťou, ktorou možno RAS deliť na kategórie. To znamená, že aj pre nesekvenčný RAS možno vymedziť 9 kategórií (ako v tab. 2.1). Definícia nesekvenčného procesu v publikácii (Ezpeleta-Recalde, 2004, str. 95) naň nekladie nijaké ďalšie podmienky, preto NS-RAS zložený z takýchto procesov spadá do najvšeobecnejšej kategórie nesekvenčných MT-NO-RAS. Ak nebude uvedené inak, tak v ďalšom texte bude NS-RAS z rovnakej kategórie.

2.2.2.2 Vyhýbanie sa uviaznutiu v RAS

Vyhýbanie sa uviaznutiu v systémoch prideľovania prostriedkov možno všeobecne definovať nasledovne (upravené podľa (Reveliotis et al., 1997, str. 9)).

Definícia pojmov:

Stav s^i je **dosiahnuteľný** zo stavu s^j, označujeme $s^i \leftarrow s^j$, práve vtedy, keď existuje postupnosť udalostí, ktorá môže priviesť systém zo stavu s^j do stavu s^i. V zápise konečného automatu:

$$\forall s^i, s^j \in S, s^i \leftarrow s^j \Leftrightarrow \exists u \in E^* : f(s^j, u) = s^i$$

Navyše, stav $s^i \in S$ je v systéme dosiahnuteľný práve vtedy, keď je dosiahnuteľný z počiatočného stavu s^0, $s^i \leftarrow s^0$. Množina dosiahnuteľných (nedosiahnuteľných) stavov bude označovaná ako $S_r(S_{\bar{r}})$.

Stav s^i je **bezpečný** práve vtedy, keď stav s^0 je dosiahnuteľný zo stavu s^i. Stav, ktorý nie je bezpečný, sa volá nebezpečný stav. V zápise konečného automatu:

$$\forall s^i \in S,\ safe(s^i) \Leftrightarrow \exists u \in E^* : f(s^i, u) = s^0 \ \Leftrightarrow\ s^0 \leftarrow s^i$$

Množina bezpečných (nebezpečných) stavov bude označovaná ako $S_s(S_{\bar{s}})$.

Stavový prechod vychádzajúci z bezpečného stavu je **bezpečný** vtedy a len vtedy, keď vedie do bezpečného stavu. V zápise konečného automatu:

$$\forall s^i \in S_s,\ \forall e \in \mathbf{F}(s^i),\ safe(e/s^i) \Leftrightarrow\ f(s^i, e) \in S_s$$

Ďalej je definovaný **evidentný previs** (apparent slack) prostriedku R_j vzhľadom k množine procesov DP^i vykonávaných v systéme sa rovná kapacite C_j prostriedku R_j mínus počet jeho jednotiek pridelených procesom v DP^i. Potom stav s^i je **čiastočné uviaznutie**, ak existuje (pod)množina procesov DP^i vykonávaných v systéme taká, že pre každý proces j_j z DP^i existuje prostriedok R_k taký, že j_j požaduje z neho určitý počet jednotiek, ktorý je väčší, než evidentný previs prostriedku R_k vzhľadom k množine DP^i. Ak predchádzajúca podmienka platí pre celú množinu procesov DP^i, ide o **úplné uviaznutie**.

Stratégia vyhnutia sa uviaznutiu (Deadlock Avoidance Policy – DAP) sa snaží obmedziť fungovanie RAS do jeho dosiahnuteľného a bezpečného podpriestoru S_{rs} **zakázaním** príslušnej podmnožiny prípustných prechodov tak, že nebezpečný podpriestor $S_{r\bar{s}}$ sa stane nedosiahnuteľným zo stavu s^0. Súčasne treba zabezpečiť, že každý stav s^i v zostávajúcom grafe (t. j. dosiahnuteľný pri riadiacej stratégii) je stále bezpečný (t. j. v zostávajúcom grafe existuje orientovaná cesta vedúca z s^i do s^0). Stavy, ktoré sú dosiahnuteľné pri DAP a z ktorých je postup zakázaný stratégiou vnútenými podmienkami a nie štruktúrou RAS, sa v literatúre o uviaznutiach označujú ako **obmedzené** uviaznutia.

Tieto myšlienky sú formalizované nasledujúcimi definíciami v kontexte konečných automatov:

(1) Stratégia vyhýbania sa **P** je funkcia

$$\mathbf{P}: S \rightarrow 2^E, \mathbf{P}(s^i) = \{e \in \mathbf{F}(s^i) : e \textit{ je stratégiou vybraté}\}$$

Udalosti $e \in \bigcup_i \mathbf{P}(s^i)$ sa volajú (stratégiou) povolené udalosti.

(2) Pri danej stratégii vyhýbania **P** označme $s^i \xleftarrow{P} s^j$ fakt, že stav s^i je dosiahnuteľný zo stavu s^j postupnosťou udalostí, ktorá obsahuje iba udalosti povolené stratégiou. Ďalej nech $S_r(\mathbf{P}) = \{s^i : s^i \xleftarrow{P} s^0\}$ a $S_s(\mathrm{P}) = \{s^i : s^0 \xleftarrow{P} s^i\}$. Potom stratégia **P** je správna práve vtedy, keď

$S_r(\mathbf{P}) \subseteq S_s(\mathbf{P})$. (Je to ekvivalentné k požiadavke, aby bola stratégia bez uviaznutia a bez obmedzeného uviaznutia.)

(3) Správna stratégia vyhýbania sa uviaznutiu $\mathbf{P}^*$ je optimálna alebo maximálne permisívna (dovoľujúca) vtedy a len vtedy, keď

$$\forall s^i \in S_{rs},\ \forall e \in \mathbf{F}(s^i),\ e \in \mathrm{P}^*(s^i) \Leftrightarrow f(s^i, e) \in S_s .$$

Z uvedenej charakteristiky vyplýva:

a) Pre daný RAS je optimálna stratégia vyhýbania sa uviaznutiu $\mathbf{P}^*$ jedinečná.

b) $S_r(\mathbf{P}^*) = S_{rs}$ – zavedenie optimálnej riadiacej stratégie $\mathbf{P}^*$ zodpovedá odňatiu z grafu dosiahnuteľnosti tých prechodových hrán, ktoré patria do rezu$[S_{rs},\ S_{r\bar{s}}]$.

c) V článkoch (Araki et al., 1977) a (Gold, 1978) je ukázané, že vo všeobecnom prípade je problém určenia bezpečnosti stavu RAS NP-úplný. Keďže zahrnutie prechodu do optimálnej riadiacej stratégie $\mathbf{P}^*$ závisí od bezpečnosti nasledujúceho stavu, vychádza, že získanie stratégie $\mathbf{P}^*$ je NP-ťažký problém (Garey-Johnson, 1979).

Z posledných riadkov vyplýva, že až na niektoré špeciálne prípady systémov RAS, sú vyvinuté stratégie vyhýbania sa uviaznutiu suboptimálne, t. j. obmedzujú stavy systému viac, než je skutočný počet bezpečných stavov.

2.2.3 Petriho siete

Petriho sieť (Petri Net, PN) je označenie pre širokú triedu diskrétnych matematických modelov, ktoré umožňujú špecifickými prostriedkami popísať riadiace a informačné toky vo vnútri modelovaných systémov.

Pôvod PN sa datuje do roku 1962 k dizertačnej práci nemeckého matematika Carla Adama Petriho „Kommunikation mit Automaten“, ktorý v nej zaviedol nový koncept popisu vzájomnej závislosti medzi podmienkami a udalosťami modelovaného systému. Koncept sa neskôr rozvinul do novej oblasti operačnej analýzy, zahŕňajúcej v súčasnosti veľa rôznych tried Petriho sietí.

Triedy vznikli pridaním nových vlastností k základným prvkom a vlastnostiam základnej verzie. Motiváciou ich vzniku je získať väčšiu *modelovaciu mocnosť* formalizmu, čo je schopnosť modelu vyjadrovať základné rysy modelovaného systému, alebo väčšiu *rozhodovaciu mocnosť*, ktorá sa vzťahuje k existencii postupov na analýzu vlastností modelov. Modelovacia a rozhodovacia mocnosť triedy sú žiaľ v istom zmysle protichodné vlastnosti – zvýšením modelovacej klesá obvykle rozhodovacia mocnosť triedy modelov a naopak.

Modifikácie buď rozširujú alebo zužujú základnú triedu Petriho sietí, podľa toho ich radíme medzi podtriedy alebo rozšírenia Petriho sietí. Podtriedy majú zväčša zvýšenú rozhodovaciu mocnosť, kým rozšírenia poskytujú zvýšenú modelovaciu mocnosť formalizmu.

Medzi aplikačné oblasti Petriho sietí patrí nielen výpočtová technika, ako paralelné architektúry, komunikačné protokoly, paralelné databázové systémy, prekladače a počítačové siete, ale aj telekomunikácie, automatizované výrobné systémy alebo riadenie ľudských systémov.

Ako píše autor v publikácii (Tricas, 2003, str. 19-20), Petriho sieť je pre modelovanie systémov DES a špeciálne RAS favorizovaná oproti ďalším modelovacím paradigmám (konečný automat, konečno rekurzívne procesy, nástroje teórie grafov a i.) najmä vďaka tomu, že spolu kombinuje tieto vlastnosti ako

- ľahká interpretácia súbežnosti, zdieľania prostriedkov, konfliktov, vzájomných vylúčení a nedeterminizmu,
- rôzne úrovne abstrakcie, ktoré umožnia prijať rôzne triedy Petriho sietí v rôznych fázach skúmaného procesu,
- definovaná sémantika, ktorá umožní validáciu systému a overenie vlastností prostredníctvom analýzy modelu,
- možnosť generovania kódu z modelu Petriho siete za účelom získania prototypu riadiaceho programu,
- príjemná a intuitívna grafická reprezentácia, ktorá pomôže pri komunikácii ľuďom angažovaným v modelovaní systému.

Opis Petriho siete je, aj pri obmedzení sa iba na informácie potrebné pre spracovanie mojej úlohy, pomerne rozsiahly. Preto ho uvádzam v prílohe A, kam by som na tomto mieste odkázal každého čitateľa, ktorý Petriho siete podrobnejšie nepozná.

V jedinej podkapitole sa obmedzím iba na použitie podtried Petriho siete na modelovanie systému prideľovania prostriedkov (RAS), ktoré sa v tejto práci ďalej využívajú.

2.2.3.1 Modelovanie RAS podtriedami Petriho siete

Uvádzané definície vychádzajú z publikácií (Tricas, 2003), (Ezpeleta-Recalde, 2004), (Park-Reveliotis, 2001) a (Reveliotis, 2006).

Definícia 2-1

Procesnou Petriho sieťou (process Petri net, P^2N) nazývame takú zovšeobecnenú silno súvislú Petriho sieť bez vlastných cyklov $\mathcal{N} = (P, T, F, W, M_0)$, pre ktorú platí:

(1) P je rozložená nasledovne: $P = P_0 \cup P_S \cup P_R$, kde platí:

(a) $P_0 = \{p_0\}$,

(b) $P_S \cap P_R = P_S \cap P_0 = P_R \cap P_0 = \varnothing$,

(c) $P_S \neq \varnothing$,

(d) $P_R = \{r_1, r_2, ..., r_k\} \neq \emptyset, k > 0$.

(2) Podsieť generovaná množinou $\{p_0\} \cup P_S \cup T$, $\mathcal{N}_{(\{p_0\} \cup P_S \cup T)}$ je silno súvislý stavový stroj taký, že každý cyklus obsahuje miesto p_0.

(3) Pre $\forall r \in P_R$ existuje jedinečný minimálny p-poltok $\mathbf{Y}_r \in \mathbf{N}^{|P|}$ taký, že $\{r\} = \|\mathbf{Y}_r\| \cap P_R$, $\{p_0\} \cap \|\mathbf{Y}_r\| = \emptyset$, $P_S \cap \|\mathbf{Y}_r\| \neq \emptyset$ a $\mathbf{Y}_r[r] = 1$.

(4) $P_S = \bigcup_{r \in P_R} (\|\mathbf{Y}_r\| \setminus \{r\})$.

(5) $M_0(p_0) > 0$, $\forall p \in P_S : M_0(p) = 0$, $\forall p \in P_S, \forall r \in P_R : M_0(r) \geq \mathbf{Y}_r(p)$.

■

Procesná Petriho sieť so svojím značením, ktorej definícia vychádza z (Tricas, 2003, str. 32-35), sa používa na reprezentáciu spracovania množiny procesov rovnakého typu (v nasledujúcich riadkoch tejto kapitoly pojem proces zodpovedá jednej inštancii typu procesov). V niektorých publikáciách sa tiež nazýva *jednoduchý sekvenčný proces* (Simple Sequential Process – S^2P). Ilustráciou procesnej PN môže byť podsieť na Obr. 9.8 vytvorená miestami *p*10, *p*11, *p*12, *p*13, *p*14, *p*15, *R*1, *R*2, *R*3 a prechodmi a hranami, ktoré ich spájajú.

Podľa podmienky (1) je množina miest procesnej PN rozložená na podmnožiny obsahujúce miesto p_0, ktoré reprezentuje *nečinný stav* procesu mimo spracovania (idle state), množinu miest P_S pre jednotlivé *fázy spracovania* a množinu miest P_R, modelujúcich *stav prostriedkov* potrebných pre spracovanie procesu.

Podmienka (2) zabezpečuje sekvenčné spracovanie procesu prepojením fáz spracovania a nečinného stavu pomocou prechodov z množiny *T*, pričom varianty spracovania sú prípustné. Spracovanie procesu začína i končí vždy v nečinnom stave (resp. proces vždy prechádza nečinným stavom) – to je model reálnej situácie, kde spracovaný proces prepustí miesto v systéme novému procesu rovnakého typu pred spracovaním. Zároveň to zamedzuje situácii, v ktorej by sa proces nevykonal podľa kompletného predpisu od začiatku do konca.

Varianty umožňuje charakter stavového stroja (Definícia 9-20), kde nie je povolené paralelné spracovanie procesu (t. j. do a z prechodu vstupuje i vystupuje práve jedna značka), ale výber z variantov je možný (t. j. počet vstupných a výstupných prechodov do a z miesta môže byť aj väčší ako jeden).

Podmienka (3) opisuje skutočnosť, že každý zahrnutý prostriedok sa dá znovu použiť (nemožno ho v systéme vytvoriť ani zničiť) a je aspoň jedenkrát pridelený procesu, pričom sa prideľuje pre ľubovoľne dlhú postupnosť fáz spracovania (bez nečinného stavu). Pridelenie symbolizuje hrana z miesta prostriedku do prechodu a uvoľnenie hrana z prechodu do miesta prostriedku, pričom tieto prechody sú rôzne (v sieti sú zakázané vlastné cykly).

Štvrtá podmienka vylučuje z modelu procesu také fázy spracovania, pri ktorých by nebol priradený nijaký prostriedok zo zahrnutej množiny. Z teoretického hľadiska táto podmienka nie je pre riešenie problému v tejto práci potrebná a mohla by byť vynechaná.

Prípustné počiatočné značenie podľa poslednej podmienky symbolizuje stav, keď všetky prítomné procesy sú nečinné a ich počet pre daný typ je zachytený v mieste p_0, a kapacita prostriedkov je dostatočná na to, aby sa v každej fáze procesu dala aspoň jedna inštancia spracovať. Každé ďalšie značenie potom symbolizuje stav systému.

Definícia 2-2

Nech $I_{\mathcal{N}} = \{1, 2, ..., m\}$ je konečná neprázdna množina indexov. S^4PR je súvislá zovšeobecnená Petriho sieť bez vlastných cyklov $\mathcal{N} = (P, T, F, W, M_0)$, pre ktorú platí:

(1) $P = P_0 \cup P_S \cup P_R$ je rozložená tak, že:

(a) $P_S = \bigcup_{i \in I_{\mathcal{N}}} P_{S_i}$, kde pre každé $i \in I_{\mathcal{N}}, P_{S_i} \neq \emptyset$, a pre každé $i, j \in I_{\mathcal{N}}, i \neq j, P_{S_i} \cap P_{S_j} = \emptyset$.

(b) $P_0 = \bigcup_{i \in I_{\mathcal{N}}} \{p_{0_i}\}$.

(c) $P_R = \{r_1, r_2, ..., r_n\}, n > 0$.

(2) $T = \bigcup_{i \in I_{\mathcal{N}}} T_i$, kde pre každé $i \in I_{\mathcal{N}}, T_i \neq \emptyset$, a pre každé $i, j \in I_{\mathcal{N}}, i \neq j, T_i \cap T_j = \emptyset$.

(3) Pre každé $i \in I_{\mathcal{N}}$, podsieť $\mathcal{N}_{(P_{0_i} \cup P_{S_i}, T_i)}$ je silno súvislý stavový stroj, taký, že každý cyklus obsahuje p_{0_i}.

(4) Pre každé $r \in P_R$ existuje jedinečný minimálny p-poltok $\mathbf{Y}_r \in \mathbf{N}^{|P|}$ taký, že $\{r\} = \|\mathbf{Y}_r\| \cap P_R$, $\{p_0\} \cap \|\mathbf{Y}_r\| = \emptyset$, $P_S \cap \|\mathbf{Y}_r\| \neq \emptyset$ a $\mathbf{Y}_r[r] = 1$.

(5) $P_S = \bigcup_{r \in P_R} (\|\mathbf{Y}_r\| \setminus \{r\})$.

■

Tab. 2.3 Vzťah kategórií RAS a zodpovedajúcich podtried Petriho siete.

Systém prideľovania prostriedkov (RAS)		Podtrieda PN
sekvenčný	SU-TO-RAS, ST-TO-RAS, MT-TO-RAS SU-PO-RAS, ST-PO-RAS, MT-PO-RAS	S^4PR^*
	SU-NO-RAS, ST-NO-RAS, MT-NO-RAS	S^*PR
nesekvenčný	SU-TO-RAS, ST-TO-RAS, MT-TO-RAS SU-PO-RAS, ST-PO-RAS, MT-PO-RAS SU-NO-RAS, ST-NO-RAS, MT-NO-RAS	GS^*PR

* Pre niektoré verzie RAS možno nájsť v literatúre podtriedy PN s iným označením

Pomenovanie S^4PR podľa vysvetlenia v publikácii (Tricas, 2003, str. 43) nemá nijaký konkrétny význam. Autori ho vybrali z toho dôvodu, že táto trieda sietí je zovšeobecnením skôr zavedených tried nazvaných:

- S^3PR – System of simple sequential processes with resources (systém jednoduchých sekvenčných procesov s prostriedkami) uvedený v publikácii (Ezpeleta et al., 1995),
- ES^3PR – Extended simple sequential system of processes with resources (rozšírený jednoduchý sekvenčný systém procesov s prostriedkami) uvedený v publikácii (Reveliotis, 2006) – to je systém, ktorý požaduje, aby sa prostriedky procesom prideľovali po jednom, pričom súčasné používanie viac prostriedkov je povolené.
- L-S^3PR – System of simple linear sequential processes with resources (systém jednoduchých lineárnych sekvenčných procesov s prostriedkami), uvedený v publikácii (Ezpeleta et al., 1998).

Trieda S^4PR, prvý raz zavedená v publikácii (Tricas et al., 1999), je rovnaká ako S^3PGR^2 (System of Simple Sequential Processes with General Resource Requirements, systém jednoduchých sekvenčných procesov so všeobecnými požiadavkami na prostriedky) podľa (Park-Reveliotis, 2001). Ilustračná sieť S^4PR sa nachádza na Obr. 9.8.

Od triedy sietí S^4PR je odvodená S^*PR (napr. Tricas, 2003; Ezpeleta-Recalde, 2004), ktorá sa skladá z *rozšírených procesných Petriho sietí*. Tie sa od pôvodných líšia iba v tom, že pripúšťajú existenciu cyklu neobsahujúceho miesto p_0 (podmienka (3)) t. j. existenciu cyklu v spracovaní procesu pred jeho ukončením. Z toho vyplýva, že kým siete S^4PR môžu modelovať nanajvýš čiastočne usporiadané RAS, S^*PR sú pripravené aj na neusporiadané RAS (tab. 2.3).

Definícia 2-3

Petriho sieťou nesekvenčného procesu (non-sequential process Petri net, NP^2N) nazveme takú značenú Petriho sieť $\mathcal{N} = (P, T, F, W, M_0)$, pre ktorú platí:

(1) P je rozložená nasledovne: $P = P_0 \cup P_S \cup P_R$, kde platí:

(a) $P_0 = \{p_0\}$,

(b) $P_S \cap P_R = P_S \cap P_0 = P_R \cap P_0 = \emptyset$,

(c) $P_S \neq \emptyset$,

(d) $P_R = \{r_1, r_2, ..., r_k\} \neq \emptyset, k > 0$.

(2) Podsieť generovaná množinou $P_0 \cup P_S \cup T$, $\mathcal{N}_{(P_0 \cup P_S \cup T)}$ je obyčajná, konzervatívna a konzistentná Petriho sieť bez vlastných cyklov.

(3) Pre $\forall r \in P_R$ existuje neprázdna množina $\Upsilon_r = \{\mathbf{Y}_r^1, \mathbf{Y}_r^2, ..., \mathbf{Y}_r^{k_r}\}$ minimálnych p-poltokov taká, že pre každé $i \in \{1, 2, ..., k_r\}$ platí:

(a) $\{r\} = \|\mathbf{Y}_r^i\| \cap P_R$,

(b) $\{p_0\} \cap \|\mathbf{Y}_r^i\| = \emptyset$.

(4) $P_S = \bigcup_{r \in P_R} \bigcup_{i \in \{1, 2, ..., k_r\}} \left(\|\mathbf{Y}_r^i\| \setminus \{r\} \right)$.

(5) $M_0(p_0) = 1$, $\forall p \in P_S : M_0(p) = 0$, $\forall r \in P_R : M_0(r) \geq 0$.

■

Definícia Petriho siete nesekvenčného procesu pochádza z (Ezpeleta-Recalde, 2004), kde ju autori nazývajú *nesekvenčným procesom s priradením prostriedkov* (non-sequential resource allocation process, NS-RAP). Základný rozdiel od procesnej Petriho siete je v tom, že pri nesekvenčnom procese sa pripúšťa paralelné spracovanie, t. j. z jedného prechodu môže vychádzať, ako aj do neho vchádzať viac ako jedna hrana, čím sa počet značiek symbolizujúcich proces v sieti mení. Na druhej strane sa v nej nepočíta s pružným smerovaním, ktoré je obsiahnuté v definícii procesnej Petriho siete (dané charakterom stavového stroja podsiete v bode (2) tejto definície).

Ja sa budem v ďalšom texte odvolávať na sieť obsahujúcu nesekvenčné procesy s priradením prostriedkov značkou *GS*PR* (Generalised system of processes with resources, zovšeobecnený systém procesov s prostriedkami). V tab. 2.3 je zachytený vzťah tejto siete k príslušným RAS.

2.3 Existujúce riešenia problému

Po rámcovom načrtnutí prístupov k vysporiadaniu sa so stavom uviaznutia uvedenom v kapitole 2.1 a vymedzení používaných teoretických formalizmov v predchádzajúcej kapitole sa v tejto časti venujem prehľadu riešení stavu uviaznutia z literatúry, ktoré sa mi podarilo naštudovať. Je ich pomerne veľa a táto oblasť sa stále rozvíja, a tak aj napriek ambícii zachytiť ich čo najviac si nasledujúce časti nemôžu robiť nárok na kompletný prehľad. Ide skôr o naznačenie viacerých smerov riešenia, kde prevažujú najmä algoritmy pre vyhnutie sa uviaznutiu a čiastočne pre prevenciu pred uviaznutím. Táto orientácia súvisí so zamýšľanou aplikáciou prístupov k uviaznutiu v dopravnom systéme, o ktorej je viac uvedené v kapitole 4.2.

Pre zaujímavosť ešte doplním, že v dostupnej literatúre je riešeniu problému uviaznutia v jednotlivých aplikačných oblastiach venovaný rôzny priestor. Kým napríklad pre výslovne všeobecný dopravný systém (DS), aký bol naznačený v úvode práce, som nenašiel nijakú publikáciu venujúcu sa tejto téme, tak v oblasti operačných systémov (OS) bola rozoberaná už od konca 60. rokov minulého storočia (Coffman et al., 1971) (hoci, ako píšem v kapitole 2.1.6, sa tu už aktívny prístup k riešeniu tohto problému nevyužíva). V oblasti výrobných systémov, a najmä pružných výrobných systémov (PVS), je hľadanie metód riešenia uviaznutia aktuálne posledných cca 25 rokov. S PVS súvisí aj pohyb automaticky ovládaných vozidiel vo výrobnom systéme – sú to tzv. systémy AGV (Automated Guided Vehicle), čo bola jediná oblasť súvisiaca s dopravou, ktorú sa mi podarilo v literatúre zameranej na stav uviaznutia nájsť. Avšak, ako bude neskôr opísané, tento systém má jednoduchší charakter ako dosiaľ opisovaný dopravný systém.

2.3.1 Algoritmus bankára

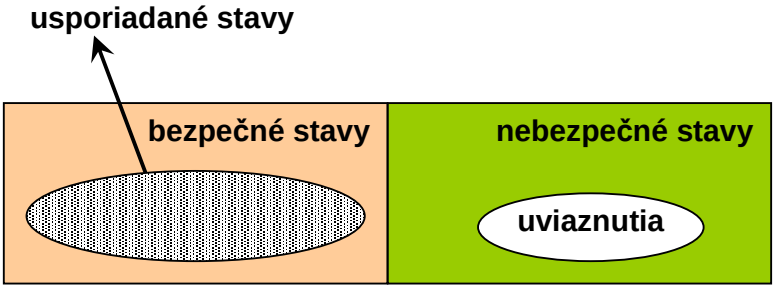
Prvou z metód používaných na riešenie problému uviaznutia je algoritmus bankára. Ide o metódu pre vyhýbanie sa uviaznutiu, to znamená, že na odlíšenie bezpečného stavu systému od nebezpečného (Obr. 2.1) používa ako informácie o štruktúre systému, tak aj informácie o jeho aktuálnom stave.

Algoritmus ako prvý publikoval Dijkstra, vtedy s motiváciou riešenia uviaznutí v počítačových operačných systémoch (Dijkstra, 1965). Dnes patrí medzi základné algoritmy na riešenie uviaznutia a používa sa aj v ďalších aplikačných oblastiach RAS.

2.3.1.1 Základná verzia algoritmu

Princíp algoritmu (Dijkstra, 1965, Lang, 1999, Lawley et al., 1998, Silberschatz et al., 2002) spočíva v zoradení prebiehajúcich procesov v systéme do takého poradia, aby každý z nich mohol ukončiť svoju činnosť s prostriedkami, ktoré

- má aktuálne priradené,
- sú práve v systéme voľné,



Obr. 2.4 Vzťah bezpečnosti a usporiadanosti stavov v stavovom priestore

- sú práve priradené procesom uvedeným vo vytváranom poradí pred zaraďovaným procesom.

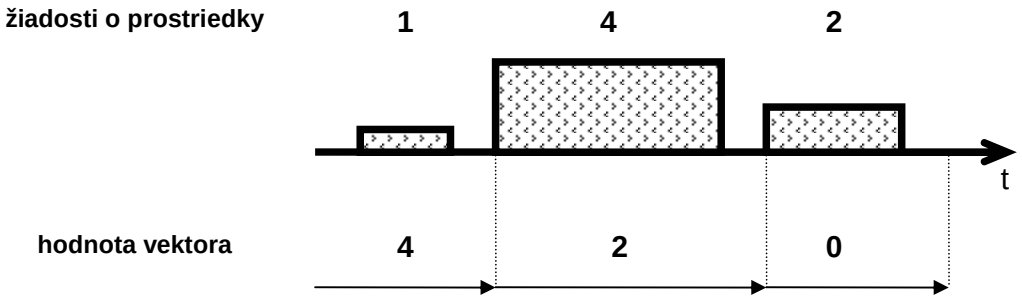
Ak sa takto podarí aktuálne procesy zoradiť, stav sa označí ako usporiadaný. Každý usporiadaný stav je zároveň aj stavom bezpečným (Obr. 2.4), pretože ak dokončíme spracovanie jednotlivých procesov v uvedenom poradí, zaručene sa každý proces úspešne dokončí a nedôjde k uviaznutiu. Toto poradie však neznamená, že by bolo nutné spracovať všetky procesy v uvedenom poradí. Je však poistkou pre prípad, že by všetky ostatné sekvencie spracovania procesov viedli do stavov uviaznutia, a teda zaručí aspoň jeden spôsob, ako bezpečne ukončiť všetky pracujúce procesy.

Ak sa procesy zoradiť nepodarí, t. j. stav je neusporiadaný, riadiaci systém nemá istotu, že existuje spôsob bezpečného ukončenia pracujúcich procesov, a tak nevyhlási stav za bezpečný. Neznamená to ešte, že je stav automaticky nebezpečný, pretože zvyčajne existujú stavy, ktoré sú bezpečné, no usporiadaná postupnosť procesov sa podľa algoritmu zostaviť nedá.

Algoritmus sa používa pri riešení otázky, či možno splniť požiadavku na pridelenie prostriedkov žiadajúcemu procesu. Riadiaci systém si namodeluje situáciu, ako by už prostriedky boli pridelené a algoritmom si otestuje usporiadanosť nového stavu. Ak vyjde stav usporiadaný, tak sa prostriedky môžu prideliť, v opačnom prípade sa žiadosť nesplní a zaradí sa do zoznamu žiadostí čakajúcich na splnenie. Zostane tam dovtedy, kým pri niektorej z budúcich zmien systému, pri ktorých sa uvoľnia prostriedky, nedôjde k stavu, kedy jej uspokojenie povedie k bezpečnému stavu.

Algoritmus vo svojej základnej verzii teda vyžaduje (podľa typov prostriedkov)

- počet voľných prostriedkov v systéme (údajovú štruktúru označím *Volne*),
- počet prostriedkov priradených jednotlivým procesom (*Priradene*),
- maximálny počet prostriedkov potrebný pre dokončenie jednotlivých procesov, ktorý deklaruje každý proces na začiatku svojej činnosti (*Potrebne*).



Obr. 2.5 Ukážka zmeny hodnoty vo vektore *ZostavajucePotrebne* v závislosti od naplánovaných žiadostí o prostriedky jedného typu v čase.

Informácie sa týkajú aktuálneho stavu systému, tretí typ vychádza aj zo znalosti štruktúry systému, pokiaľ sa procesy opakujú podľa určitých štandardných schém a ich požiadavky na prostriedky sú známe.

2.3.1.2 Vylepšenia a zložitosť algoritmu

Ako uvádzajú autori v publikácii (Lawley et al., 1998), prostredie počítačového operačného systému, kde algoritmus pôvodne vznikol, postráda apriórnu informáciu o postupnostiach žiadostí angažovaných procesov. Z toho vyplýva, že pôvodná verzia algoritmu navrhuje riešenie pre najhorší prípad, ktorým je súčasné priradenie maximálneho deklarovaného počtu prostriedkov zo všetkých typov každému procesu, a teda nutnosť mať pre vykonanie daného procesu vždy tento maximálny počet k dispozícii. Je zrejmé, že taký prístup zanedbáva sekvenčnú charakteristiku požiadaviek procesu a vedie k fungovaniu systému s väčšími obmedzeniami než je nutné.

V prípade, že je známa aj štruktúra postupnosti žiadostí procesov o prostriedky, možno namiesto maximálneho počtu prostriedkov jednotlivých typov potrebného pre dokončenie procesov (*Potrebne*) použiť **maximálny počet súčasne potrebných prostriedkov jednotlivých typov** (nazvime to *ZostavajucePotrebne*) počas vykonávania **zvyšnej časti** procesu do konca. To je počet prostriedkov, ktoré budú pridelené z daného typu súčasne, pričom sa zo všetkých zostávajúcich súčasných pridelení uchováva maximálna hodnota. Ak napríklad proces požiada o prostriedky jedného typu trikrát v disjunktívnych častiach vykonania procesu (Obr. 2.5) tak, že najprv potrebuje jeden, neskôr štyri a potom ešte raz dvojicu prostriedkov, tak hodnota vo vektore *ZostavajucePotrebne* sa bude meniť v príslušných časových okamihoch klesajúc z hodnoty štyri až na nulu.

Uvedenú novinku do algoritmu vnášajú autori v publikácii (Tricas et al., 2000/1) a (Lawley et al., 1998). Novinka umožní prijať väčší počet stavov za bezpečné, a tým zmenšiť obmedzenia fungovania systému. Vyžaduje si však zaznamenanie väčšieho

množstva údajov o každom procese oproti pôvodnému algoritmu, a to rádovo toľkokrát, koľko je zmien stavu procesu v systéme s pridelením prostriedku (resp. zmien stavu s uvoľnením prostriedku).

Pre zaujímavosť táto úprava môže byť teoreticky zahrnutá aj v algoritmoch uvedených v publikáciách (Martincová-Grondžák, 2004) a (Silberschatz et al., 2002), nakoľko sa tam nikde nespomína, či a ako sa zmenia údajové štruktúry pri uvoľňovaní prostriedkov. Ak riadiaci systém pri uvoľnení zmenšuje obsah štruktúr pre maximálny počet ešte potrebných prostriedkov, tak implementuje aj túto úpravu a používa údajovú štruktúru *ZostavajucePotrebne*. V opačnom prípade ide o základnú opisovanú verziu a údajovú štruktúru *Potrebne*.

Ďalšie vylepšenia algoritmu prinášajú autori v publikácii (Lawley et al., 1998). Uvádzajú štyri algoritmy (ich prepis je v prílohe C.1), z ktorých prvý je základnou verziou vylepšenou o vyššie spomenutú znalosť štruktúry systému.

Druhý algoritmus vychádza z nového definovaného pojmu **čiastočne usporiadaného stavu systému vzhľadom k P_{km},** kde P_{km} označuje m-tú fázu (stav) procesu k-teho typu a jeho inštancia sa označuje π_{km} (v polohe P_{km} môže byť viac procesov π_{km}). Je to taký stav, kde stačí preveriť, či je možné proces zo stavu P_{km} dokončiť, a len čo sa to podarí, algoritmus končí bez ohľadu na to, či sa podarí preveriť dokončenie ostatných bežiacich procesov alebo nie, teda stačí vytvoriť poradie dokončených procesov po proces v stave P_{km}.

Citovaní autori dokázali, že pokiaľ predchádzajúci stav systému bol bezpečný, tak aj nový čiastočne usporiadaný stav vzhľadom k stavu procesu P_{km} bude bezpečný, a ak predchádzajúci stav bol usporiadaný, tak aj nový stav bude usporiadaný. V praxi to znamená, že ak potrebujeme preveriť usporiadanosť stavu po priradení žiadaných prostriedkov procesu π_{km} a zistíme, že tento nový stav je čiastočne usporiadaný vzhľadom k stavu procesu P_{km}, môžeme ho vyhlásiť za bezpečný a prostriedky priradiť. Prínos tohto algoritmu je v skrátení výpočtu pri hľadaní usporiadanosti stavu vo väčšine behov algoritmu (okrem tých, kde v poradí procesov je P_{km} na konci) a vo zväčšení počtu stavov prijatých ako bezpečných – podľa simulačných testov sekcii 5 v publikácii (Lawley et al., 1998).

Tretí a štvrtý algoritmus v uvedenej publikácii vychádzajú z ďalšieho nového pojmu: **V_m-usporiadaný stav**. Autori ho zavádzajú v snahe pokryť väčšiu množinu bezpečných stavov, aj tých, ktoré sú podľa pôvodnej definície neusporiadané a teda pôvodným algoritmom neprijaté za bezpečné (Obr. 2.4).

V_m-usporiadaný stav je taký stav, kde sa dajú pracujúce procesy usporiadať len s pomocou posunov π_{pq} (p a q reprezentujú to, čo k a m vyššie) o konečný počet krokov dopredu. Vychádza z toho, že pri niektorých neusporiadaných stavoch existuje podmnožina procesov, ktoré sa usporiadať dajú (no neobsahuje všetky procesy), a zo zostávajúcich procesov sa dá nájsť aspoň jeden taký, ktorý možno posunúť dopredu (po odstránení usporiadaných procesov zo systému a uvoľnení nimi držaných prostriedkov) o k krokov (k je kladné celé číslo väčšie ako 1) a jeho posunom sa

zostávajúce procesy odblokujú, dajú sa dokončiť a tým sa dostanú do usporiadanej postupnosti. V prípade, že tieto kroky (usporiadanie procesov a odblokovanie zostávajúcich posunom jedného z nich o *k* krokov) povedú k usporiadaniu všetkých prítomných rozpracovaných procesov, stav sa vyhlási za V_m-usporiadaný stav. Hodnota indexu *m* závisí od toho, koľkokrát je potrebné vykonať posun nejakého procesu. To znamená, že usporiadané stavy podľa pôvodnej definície sú V_0-usporiadané, pri posune jedného procesu o *k* krokov V_1-usporiadané, pri posune dvoch procesov o *k* krokov V_2-usporiadané, atď.

Prínos tejto úpravy je evidentný vo zväčšení množiny stavov, ktoré možno prijať za bezpečné. Platbou za neho je však vyššia časová zložitosť výpočtu, pri niektorých stavoch vedúca až k exponenciálnej, čo závisí od toho, do akej hodnoty *m* dovolíme algoritmu počítať. Autori v simulačných testoch v publikácii (Lawley et al., 1998) dosiahli 90%-né pokrytie množiny bezpečných stavov algoritmom, ktorý hľadal V_1-usporiadané stavy, t. j. 9 z 10 generovaných bezpečných stavov vyhlásil za bezpečné. Teoreticky možno predpokladať, že výpočet algoritmu max. do počtu všetkých prítomných pracujúcich procesov by mohol priniesť identifikáciu všetkých 100% bezpečných stavov v systéme.

Uvedené algoritmy sú v článku (Lawley et al., 1998) odvodené pre najjednoduchšiu kategóriu RAS, ktorú som uviedol vyššie, teda systém, kde majú procesy pridelenú iba jednu jednotku prostriedkov naraz (SU-TO-RAS). Zostáva preveriť použiteľnosť algoritmov pre ďalšie kategórie systémov, včítane NS-RAS.

Z ďalších vylepšení algoritmu bankára je pre túto prácu zaujímavý druhý návrh autorov z článku (Tricas et al., 2000/1)0, ktorý sa týka **len procesov**, ktoré majú **pružné smerovanie** svojho vykonania (procesy v PO-RAS alebo NO-RAS). Pri týchto procesoch sa v jednotlivých fázach spracovania môžu vo variantoch nachádzať rôzne požiadavky na *ZostavajucePotrebne* prostriedky, a to možno využiť v algoritme v situácii, keď proces možno zaradiť do usporiadanej postupnosti iba pri vykonaní niektorých, nie všetkých nasledujúcich variantov. To znamená, že sa pri procese s pružným smerovaním nebude hľadieť na dokončenie všetkých variantov v danom stave, ale bude stačiť dokončenie jedného z nich. Platbou za túto výhodu je väčšia časová zložitosť algoritmu v situáciách, keď varianty prinášajú viac možností nájdenia výsledného usporiadania. Túto myšlienku podrobnejšie rozvádza autor v publikácii (Tricas, 2003), ktorý vysvetľuje rozdiely medzi rôznymi prístupmi a porovnáva časovú zložitosť ich výpočtov (tab. 2.4). Vychádza z dvoch základných prístupov: statického (riadky 1-4 v tab. 2.4) a dynamického (riadok 5).

Statický prístup znamená, že pre každý stav procesu sa vopred vypočíta vektor požiadaviek *ZostavajucePotrebne* (na zostávajúce potrebné prostriedky pre jeho dokončenie) a pri overovaní bezpečnosti stavu sa tieto údaje použijú. Položky vektora možno vypočítať viacerými spôsobmi – tak, že obsahuje buď maximálny počet prostriedkov potrebný pre celý proces (v každom stave, t. j. ako v klasickom algoritme bankára, v tab. 2.4 riadok 1) alebo počet prostriedkov potrebný pre dokončenie

procesu z daného stavu. Ak je na výber viac variantov dokončenia, tak v druhom prípade môže položka vektora byť

- maximum zo všetkých variantov, t. j. jeden záznam (riadok 2 v tab. 2.4),
- presné údaje pre každý z variantov dokončenia, t. j. zoznam záznamov, ktorých počet zodpovedá počtu variantov dokončenia (3),
- presné údaje pre jeden z variantov dokončenia procesu (4), ktorý sa vyberie podľa zvoleného algoritmu (napr. najmenej fáz z aktuálneho stavu do konca procesu).

Dynamický prístup (riadok 5) využíva prítomnosť formalizmu S*PR (podtriedy Petriho siete uvedenej v kapitole 2.2.3.1), ktorá pre každý stav procesu ($p \in P_S$) pozná počet prostriedkov jednotlivých typov v ňom priradených a s využitím mechanizmu Petriho siete na prechod medzi miestami hľadá, či možno nájsť cestu z miesta aktuálneho stavu do miesta nečinného stavu. Tento prístup nevyžaduje nijaké špeciálne výpočty vopred (použitie prostriedkov v jednotlivých stavoch sa považuje za štandardnú informáciu v modeli S*PR). Na druhej strane je výpočet algoritmu on-line časovo náročnejší.

Z empirického porovnania prístupov autor v publikácii (Tricas, 2003) zistil, že prístup podľa riadka (3) (v uvedenej publikácii nazývaný tiež partial look-ahead, čiastočné predvídanie) nájde všetky bezpečné stavy a v porovnaní prvých štyroch prístupov podľa percentuálneho podielu nájdenia bezpečných stavov vyšlo poradie (3) – (4) – (2) – (1). Z tabuľky je však evidentné, že výpočet algoritmu v tomto prístupe je exponenciálny ako v jedinom, kým v ostatných polynomiálny.

Opísané varianty BA boli navrhnuté pre kategóriu neusporiadaných viac-typových systémov prideľovania prostriedkov (MT-NO-RAS), čo predpokladá použiteľnosť algoritmu pre všetky kategórie sekvenčného RAS.

Pred záverom kapitoly sa zmienim ešte o publikácii (Lang, 1999). Uvádza podobnú myšlienku, ako som opísal na začiatku tejto kapitoly, t. j. použitie stavovej informácie o zostávajúcich potrebných prostriedkoch namiesto maximálnych potrieb deklarovaných na začiatku procesu s variantmi smerovania. Autor navyše k opísanému ešte dekomponuje priebeh procesu na regióny, ktoré sú ohraničené bodmi s nulovým priradením prostriedkov procesu. Toto vylepšenie prináša úžitok najmä pre procesy, ktoré počas svojho vykonania aspoň raz odovzdajú všetky prostriedky.

Tab. 2.4 Zložitosť algoritmu bankára pri použití rôznych prístupov.

	prístup	záznamy			zložitosť výpočtu	
		získané pre		počet pre jeden stav	on-line	off-line
1	**statický**	celý proces		1	$\lvert P_S\rvert.\mathbf{log}(\lvert P_S\rvert).\lvert P_R\rvert$	$\lvert P_R\rvert.\lvert P_S\rvert$
2		zvyšok procesu do konca	maximum zo všetkých možných vetiev	1	$\lvert P_S\rvert.\mathbf{log}(\lvert P_S\rvert).\lvert P_R\rvert$	$\lvert P_R\rvert.(\sum_{i=1}^{n}(\lvert P_{S_i}\rvert.(\lvert P_{S_i}\rvert+\lvert T_i\rvert)))$
3			presné údaje pre každý variant dokončenia	$\lvert\mathcal{T}_S(p)\rvert$	$\lvert P_S\rvert.\mathbf{log}(\lvert P_S\rvert).\lvert P_R\rvert.\prod_{p\in P_S}\lvert f(p)\rvert$	$\sum_{i=1}^{n}\left((\lvert P_{S_i}\rvert+\lvert T_i\rvert).\sum_{p\in P_{S_i}}\lvert\mathcal{T}_S(p)\rvert\right)$
4			presné údaje pre jeden z variantov dokončenia	1	$\lvert P_S\rvert.\mathbf{log}(\lvert P_S\rvert).\lvert P_R\rvert$	$\sum_{i=1}^{n}(\lvert P_{S_i}\rvert.(\lvert P_{S_i}\rvert+\lvert T_i\rvert))$
5	**dynamický**	žiadne údaje vypočítané vopred		–	$\lvert P_S\rvert^2.\lvert P_R\rvert.\lvert T\rvert$	–

$\lvert P_S\rvert$ počet miest zodpovedajúcich stavom daného typu procesov,

$\lvert P_R\rvert$ počet miest zodpovedajúcich typom prostriedkov

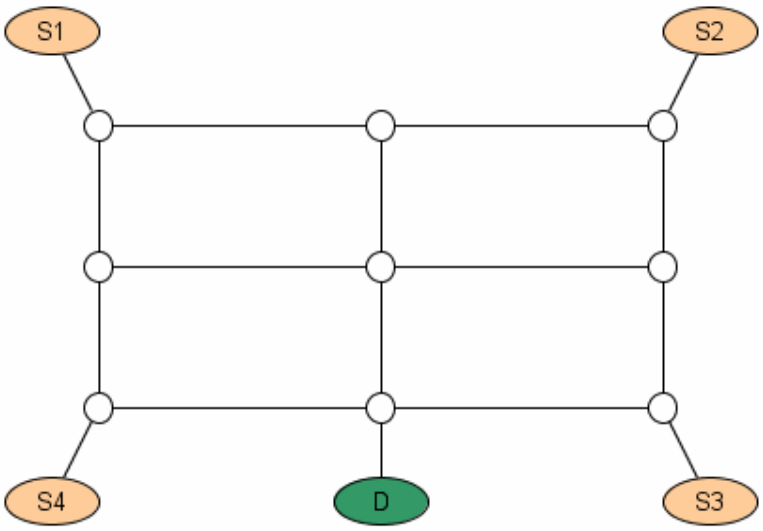
$\lvert T\rvert$ počet prechodov v sieti daného typu procesov

$\lvert T_i\rvert$ počet prechodov v *i*-tej ceste daného typu procesov

$f(p)$ množina vektorov, ktorých položky obsahujú počet prostriedkov podľa typov potrebných pre dokončenie procesu z miesta *p*

$\prod_{p\in P_S}\lvert f(p)\rvert$ počet možností rôznych priradení vektorov prostriedkov podľa typov potrebných pre dokončenie procesu z miesta *p*

$\lvert\mathcal{T}_S(p)\rvert$ počet rôznych ciest medzi miestami *p* a p_0 (počet variantov dokončenia procesu)



Obr. 2.6 Príklad systému AGV so 4 stanoviskami, depom a sieťou medzi nimi.

2.3.1.3 Upravený algoritmus pre AGV

Z hľadiska použitia v dopravnom systéme sa veľmi prínosnou javí aplikácia algoritmu bankára v systéme automaticky riadených vozidiel (Automated Guided Vehicle – AGV) v publikácii (Reveliotis, 2000).

Ide o dopravný systém, kde sa vozidlá pohybujú po obojsmerných dopravných cestách, aby prepravili náklad zo zdroja do cieľa. Vymedzený systém v uvedenej publikácii ďalej predpokladá, že vozidlá

- sa pohybujú medzi vytýčenými uzlami, ktoré zahŕňajú stanoviská (workstation) a depo (docking station),
- pohybujú sa iba jedným smerom, pričom zmeniť smer na opačný môžu iba v menovaných uzloch,
- štartujú i končia svoju „misiu“ v depe (docking station), pričom
- počas pohybu môžu byť znovu pridelené ďalšej požiadavke, ktorú začnú plniť po splnení aktuálnej.

To znamená, že jedna „misia“ AGV pozostáva z usporiadanej trojice vytýčených uzlov $<n_s, n_d, d_s>$, v ktorej symboly po rade reprezentujú zdrojové, cieľové stanovisko a depo. Predpokladá sa, že depo má dostatočnú kapacitu na prichýlenie všetkých vozidiel AGV v systéme. Jednotlivé stanoviská v systéme a depo sú prepojené sieťou ciest s križovatkami a na presun medzi dvoma vytýčenými uzlami má vozidlo výber z viacerých variantov (Obr. 2.6).

Riadenie opísaného dopravného systému je zónové, ktoré predpokladá prítomnosť nanajvýš jedného vozidla v jednej zóne. Pohyb vozidla je vykonaný zmenou priradenia vozidla z aktuálnej do susednej zóny. Zóny sú priradené úsekom medzi uzlami v sieti. V prípade, že je úsek dostatočne dlhý, môže byť rozdelený na sériu zón tak, aby sa do jednej zóny zmestilo jedno vozidlo.

Systém sa označuje AGV RAS (podľa používanej kategorizácie SU-PO-RAS), kde procesy sú pohyby jednotlivých vozidiel AGV medzi zónami a prostriedky sú zóny. Stav systému je určený priradením zón vozidlám a prechod medzi stavmi sa realizuje presunom jedného vozidla medzi susednými zónami v smere jeho pohybu.

Po vymedzení systému AGV RAS Reveliotis v publikácii (2000) navrhuje úpravy algoritmu bankára (príloha C.2), ktoré zodpovedajú faktu, že prostriedkami sú zóny a vozidlá potrebujú pre dokončenie voľné cesty, t. j. postupnosti voľných zón, ktoré spolu vytvoria potrebné cesty. Preto sa namiesto zoznamu voľných prostriedkov používa zoznam súvislých podsietí, ktoré tvoria voľné zóny, a pri preverovaní dokončenia procesu je kritériom voľná cesta pre vozidlo k jeho postupným cieľom, ktoré má ešte navštíviť. Možno tu vidieť analógiu s pôvodným algoritmom, ako i s vylepšením pre procesy s pružným smerovaním (Tricas et al., 2000/1).

2.3.1.4 Zhrnutie k verziám algoritmu bankára

Možno konštatovať, že algoritmus bankára je klasická a základná metóda pre riešenie problémov uviaznutia s pomocou znalosti stavu a štruktúry systému. V preštudovaných publikáciách sa nachádzajú okrem základnej verzie aj niektoré vylepšenia, ktoré boli odvodené buď pre prostredie operačných systémov alebo pre sekvenčné RAS, pričom dve vylepšenia rozširujúce množinu stavov algoritmom prijímaných za bezpečné boli odvodené pre najjednoduchšiu kategóriu, SU-TO-RAS, a ďalšie dve vylepšenia sa týkajú najzložitejšej kategórie, MT-NO-RAS.

Pre tému tejto práce je zaujímavá aj modifikácia algoritmu pre AGV RAS pre zónové riadenie pohybu vozidiel, ktoré však predpokladá priradenie jednej zóny vozidlu naraz. V tomto prípade ide o SU-PO-RAS.

2.3.2 Ďalšie algoritmy a prístupy

Okrem algoritmu bankára sa v literatúre nájdu aj ďalšie algoritmy na riešenie uviaznutia. Mnohé z nich využívajú podtriedy Petriho siete pre modelovanie RAS, najmä na modelovanie pôvodného systému a niekedy aj stratégia vyhýbania sa uviaznutiu je vytvorená vo forme Petriho siete.

2.3.2.1 Prístupy založené na sifónoch

Prvá skupina algoritmov je založená na **sifónoch**. Je to štrukturálny prvok Petriho siete (uvedený v definícii 9-15 v prílohe A.1.2) so špecifickou vlastnosťou, že keď sa raz vyprázdni, tak až do konca fungovania modelu zostane prázdnym. Algoritmy hľadajú a odstraňujú sifóny, ktoré obsahujú miesta pre typy prostriedkov ($p \in P_R$) a pre

stavy procesu ($p \in P_S$). Ak sa takýto sifón vyprázdni, tak stavové miesta, ktoré obsahuje, už nikdy nedostanú značku a procesy, ktorým tieto miesta prislúchajú sa zablokujú. To platí pre jedno-jednotkové systémy (SU-RAS). V prípade systémov s viacerými prostriedkami pridelenými naraz (ST-RAS alebo MT-RAS) sa zaviedol tzv. **mŕtvo označený sifón**, ktorý nemusí byť prázdny, no počet značiek v ňom nestačí pre spustenie aspoň jedného jeho prechodu. Príklad sifónu sa nachádza na Obr. 9.9.

Autori v publikácii (Park-Reveliotis, 2001) navrhujú pridať do modelu S^4PR (v článku označovanom ako S^3PGR^2, príklad na Obr. 9.8) množinu miest, ktorých počet i počiatočné značenie zodpovedá miestam prostriedkov v pôvodnom systéme. Ich pripojenie k stavovým strojom procesov vychádza z nerovností, ktoré v publikácii odvodili. Vzniká tzv. riadený systém S^3PGR^2 (Controlled S^3PGR^2 – CS^3PGR^2), ktorý zabezpečí, aby sa nevykonali prechody, ktoré spôsobujú únik značiek zo sifónov (Obr. 9.10). Okrem toho autori ukazujú, ako možno pomocou celočíselného matematického modelu vypočítať také obsadenie riadiacich miest značkami, že stratégia umožní prijatie väčšieho počtu stavov systému za prípustné (t. j. bezpečné) než je tomu pri počiatočnom nastavení obsadenia riadiacich miest podľa kapacity typov prostriedkov. Tak vzniká počiatočné značenie kontrolných miest na obrázku.

Táto stratégia vyhýbania sa uviaznutiu je určená pre C/D-RAS (MT-PO-RAS) a nazýva sa C/D-RUN DAP (RUN – Resource Upstream Neighborhood, okolie prostriedku proti prúdu). Ide o zovšeobecnenie RUN DAP, pôvodne vytvorenej pre SU-TO-RAS a neskôr zovšeobecnenej pre MT-TO-RAS (Reveliotis et al., 1997).

2.3.2.2 Prístupy založené na analýze stavového priestoru

V publikácii (Reveliotis-Choi, 2006) autori navrhujú vytvoriť riadiaci podsystém modelu v Petriho sieti na základe analýzy jeho stavového priestoru. Vychádzajú z **teórie regiónov** a navrhujú úlohu zmiešaného celočíselného programovania, ktorá maximalizuje rozsah regiónu stavového priestoru obsahujúceho bezpečné stavy pri danom obmedzení počtu riadiacich podmienok. Pomocou tejto úlohy vypočítané prípustné stavy potom dohliadací program (supervisor) počas fungovania systému povoľuje.

Výhodou tohto prístupu je, že nekladie nijaké podmienky na analyzovanú sieť z pohľadu kategorizácie RAS, a teda ho možno použiť aj pre najzložitejší nesekvenčný RAS. Túto výhodu využívajú aj autori v publikácii (Ezpeleta-Recalde, 2004), ktorí k zadefinovanému nesekvenčnému procesu s priradením prostriedkov NS-RAP definujú pridruženú sieť $\mathbf{S^*PR}$, ktorá vzniká zo stavového priestoru siete procesu NS-RAP. Ďalej je možné pomocou metód známych pre S^*PR vyhodnocovať, ktoré stavy možno prijať za bezpečné.

Kľúčovou podmienkou pre úspech týchto metód je rozsah stavového priestoru systému po určitú hranicu. To ich zatiaľ diskvalifikuje pri aplikácii v zložitých systémoch.

2.3.2.3 Iné metódy

Autori v publikácii (Reveliotis et al., 1997) okrem iného uvádzajú stratégiu Resource Ordering (RO, usporiadanie prostriedkov) pre systém ST-TO-RAS. Keďže ide o systém, ktorý umožňuje procesu viazať naraz prostriedky iba jedného typu, je možné ich usporiadať vzhľadom na poradie ich použitia v procese. Z toho vychádza stratégia RO, ktorá na prideľovanie prostriedkov kladie podmienky vzhľadom na zvolené zoradenie prostriedkov.

Publikácia (Fanti et al., 1997) sa zaoberá zavedením metód vyhýbania sa uviaznutiu na princípe orientovaných čakacích grafov, ktoré autori odvodili pre systém SU-TO-RAS.

2.3.2.4 Metódy pre vektorové systémy diskrétnych udalostí

Vektorový systém diskrétnych udalostí (VDES) je štruktúrovaný model systému diskrétnych udalostí, ktorý efektívne modeluje systémy s prirodzenou aditívnou štruktúrou (pružné výrobné systémy, niektoré počítačové a komunikačné systémy). Zaviedli ho autori v publikáciách (Li-Wonham, 1993) a (Li-Wonham, 1994) a vychádza z tzv. teórie RW o riadení systémov diskrétnych udalostí (RW je podľa iniciálok autorov Ramadge-Wonham). Viac o DES všeobecne sa možno dočítať v publikácii (Wonham, online), zbežné porovnanie DES v kontexte s Petriho sieťami, modulárnymi konečnými automatmi a štruktúrami stavových stromov sa nachádza vo (Wonham, 2003).

V publikácii (Li-Wonham, 1993) uviedli základné myšlienky teórie stavovej spätnej väzby pre VDES, ktorá zahŕňa statické i dynamické riadenie VDES. Keď v systéme neexistuje slučka (spätná väzba), riadiaci prvok možno vytvoriť vyriešením úlohy celočíselného lineárneho programovania a zapísať ho vo forme buď konjunkcie lineárnych predikátov alebo konjunktívnej normálnej formy (Li-Wonham, 1994).

Zo zbežného preštudovania teórie a príkladov usudzujem, že táto teória je určená pre najjednoduchší RAS: SU-TO-RAS.

2.3.2.5 Zložitosť algoritmov

Zložitosti algoritmov pre vyhýbanie sa uviaznutiu sa venujú autori v publikácii (Lawley-Reveliotis, 2001). Pripomínajú všeobecne známy fakt, že optimálnu stratégiu pre vyhýbanie sa uviaznutiu nemožno používať v systéme pracujúcom v reálnom čase, pretože určenie všetkých bezpečných stavov v určitých širokých triedach RAS je NP-ťažký problém. Vtedy sa treba uspokojiť so suboptimálnou stratégiou, ktorá neprijme niektoré bezpečné stavy, a tým obmedzí voľnosť systému. V súlade s týmto konštatovaním je aj porovnanie zložitosti variantov algoritmu bankára v publikácii (Tricas, 2003), ktoré som uviedol v tab. 2.4.

Avšak v niektorých špeciálnych prípadoch systémov SU-TO-RAS, ktoré medzi svojimi dosiahnuteľnými stavmi neobsahujú nijaké nebezpečné stavy bez uviaznutia, je stratégia preverenia bezpečnosti najbližšieho stavu, ktorá sa vypočíta polynomiálne,

optimálna. Autori ďalej rozoberajú, ktoré systémy uvedenej kategórie podľa vlastností spadajú medzi tie špeciálne prípady.

2.3.3 Zhrnutie k algoritmom a prístupom

V tab. 2.5 som znázornil použiteľnosť spomenutých metód. Okrem kategórie RAS, pre ktorú bola metóda v publikácii priamo uvedená, som zaškrtol aj tie kategórie, ktoré sú jednoduchšie a taktiež spĺňajú podmienky menovanej kategórie.

tab. 2.1 Použiteľnosť metód v jednotlivých kategóriách RAS.

		BA podľa (Lawley et al., 1998)	riadenie VDES	čakací graf	statégia RO	C/D-RUN DAP	BA pre AGV	základný BA a jeho 2 vylepšenia podľa (Tricas, 2003)	teória regiónov a stavový priestor NS-RAP
sekvenčný	SU-TO-RAS	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**
	ST-TO-RAS				**x**	**x**		**x**	**x**
	MT-TO-RAS					**x**		**x**	**x**
	SU-PO-RAS					**x**	**x**	**x**	**x**
	ST-PO-RAS					**x**		**x**	**x**
	MT-PO-RAS					**x**		**x**	**x**
	SU-NO-RAS							**x**	**x**
	ST-NO-RAS							**x**	**x**
	MT-NO-RAS							**x**	**x**
nesekvenčný	SU-TO-RAS								**x**
	ST-TO-RAS								**x**
	MT-TO-RAS								**x**
	SU-PO-RAS								**x**
	ST-PO-RAS								**x**
	MT-PO-RAS								**x**
	SU-NO-RAS								**x**
	ST-NO-RAS								**x**
	MT-NO-RAS								**x**

3 Ciele a metodika práce

Dizertačná práca má štyri hlavné ciele, ktoré môžem zhrnúť do nasledujúcich téz.

V oblasti riadenia procesov v uzle dopravnej siete

- analyzovať problémy riadenia,
- popis a formalizácia problému uviaznutia,
- prehľad a analýza algoritmov na riešenie problému uviaznutia

Algoritmus pre analýzu situácií a riešenie problému uviaznutia

- navrhnúť model dopravného systému s použitím Petriho siete,
- navrhnúť a implementovať vybrané algoritmy riadenia,
- overiť funkčnosť a kvalitu navrhnutých algoritmov.

Pre **meranie kvality** algoritmov riadenia

- definovať kritéria kvality,
- zhodnotiť úroveň kvality navrhnutých algoritmov a porovnať so súčasným stavom,
- porovnať rôzne verzie navrhnutých algoritmov,
- sformulovať odporúčania pre výber algoritmu riadenia.

Experimentálne **overenie** navrhnutých algoritmov na vzorovom príklade modelu uzla železničnej siete.

Použitá metodika je z teoretickej oblasti systémov prideľovania prostriedkov s použitím formalizmu Petriho sietí, a to pri vytvorení modelu, overení jeho správnosti, implementácii algoritmov, ako aj výpočte parametrov porovnania pomocou nástrojov tohto formalizmu.

4 Vlastné riešenie

Zo cieľov vyplýva, že predmetom tejto dizertačnej práce je riešenie stavu uviaznutia v dopravných systémoch, ktorý je v reálnej prevádzke neprípustný. Vychádzam z teoretického základu daného systémom prideľovania prostriedkov (RAS), pričom pracujem s jeho modelom v Petriho sieti.

Prevodu dopravného systému do tohto kontextu sa venuje prvá podkapitola. Po analýze činností, procesov a prideľovania prostriedkov v dopravnom systéme opisujem proces vytvorenia modelu v Petriho sieti.

V druhej podkapitole hodnotím možnosti riešenia problému z hľadiska prístupov k vysporiadaniu sa s uviaznutím opísaných v kapitole 2.1. V ďalšej podkapitole po preverení možností riešenia, ktoré sa črtajú vo zvolenom kontexte zdôvodním výber algoritmu bankára pre ďalšie riešenie. Za tým nasleduje vymedzenie nárokov na algoritmus a kritérií hodnotenia jeho kvality.

Kľúčovou sa javí kapitola 4.4, kde podrobne opisujem údajové štruktúry , čiastkové algoritmy a navrhované úpravy 3 verzií algoritmu bankára tak, aby sa dal vo vymedzenom systéme použiť.

4.1 Model dopravného systému vo forme RAS v Petriho sieti

Proces tvorby modelu pozostáva z analýzy dopravného systému z pohľadu prideľovania prostriedkov a následného vytvorenia modelu v Petriho sieti. Podľa toho sa aj člení táto kapitola na dve časti.

4.1.1 Analýza prideľovania prostriedkov procesom v dopravnom systéme

Jadro analýzy spočíva v zaradení činností a procesov do kategórií systémov prideľovania prostriedkov a posúdenie významu ďalších detailov v dopravnom systéme pre vytváraný model.

4.1.1.1 Činnosti a procesy

Pod dopravným procesom rozumiem technologický proces, v ktorom sa pomocou prostriedkov obsluhuje zákazka s cieľom jej premiestnenia. Tento proces možno ďalej rozčleniť na technologické činnosti ako najmenšie, ďalej už nedeliteľné jednotky. Napríklad procesom môže byť preprava kontajnera od dodávateľa k odberateľovi, ktorá pozostáva z činností prepravy cestným vozidlom a plavidlom, prekládky v termináloch kombinovanej dopravy a z činností s tým súvisiacich.

Technologické činnosti možno z **hľadiska povahy** rozdeliť na premiestňovacie a ostatné obslužné.

Premiestňovacie činnosti sú základnými činnosťami v dopravnom systéme. Keby sa v ňom nenachádzali, už by to nebol dopravný systém. Jednak ide o činnosti premiestnenia požiadaviek medzi miestom vzniku požiadavky a jej cieľom (napr. jazda kamiónu so zásielkou medzi distribučným skladom a zákazníkom), a taktiež o pomocný presun pohyblivých prostriedkov bez priameho pôsobenia na premiestnenie požiadavky, no s cieľom k tomu prispieť (napríklad presun prázdneho nákladného vozidla z miesta údržby do distribučného skladu k zásielke).

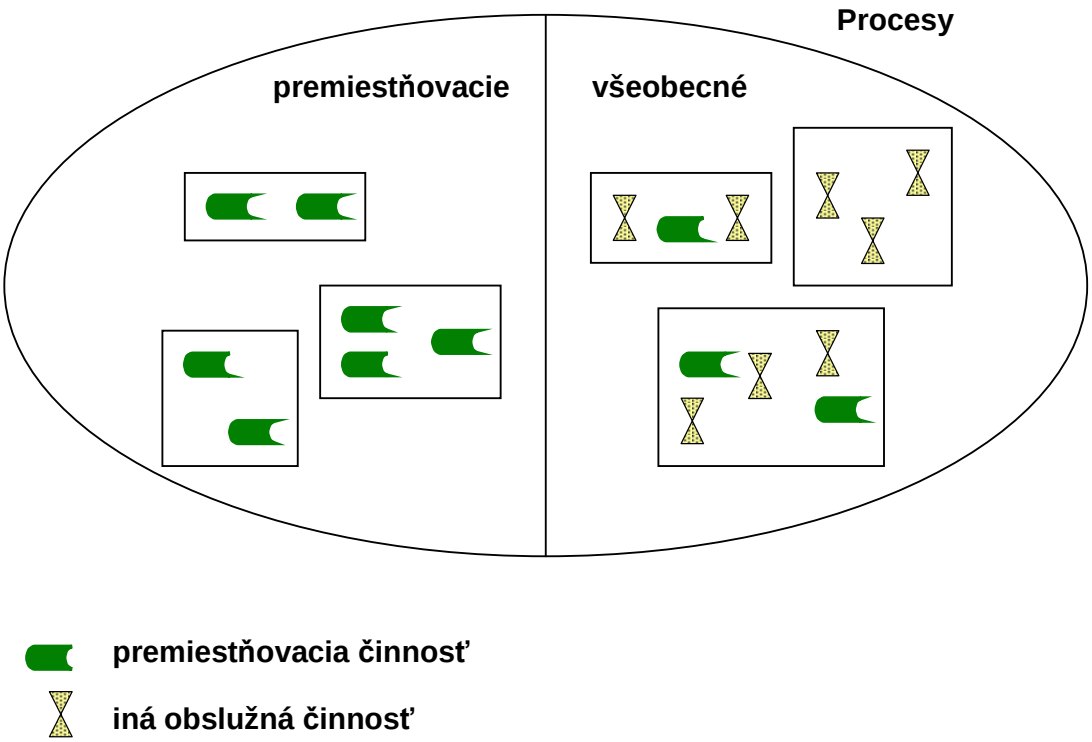
Medzi **ostatné obslužné činnosti** možno zahrnúť činnosti, ktoré sú nevyhnutné pre vykonanie procesu premiestnenia, ako je nástup a výstup cestujúcich, nakladanie a vykladanie nákladu, oprava a údržba dopravných prostriedkov, kontrola ich technického stavu, cestujúcich alebo zásielok a pod. Tieto činnosti sa vyskytujú medzi činnosťami presunu, pričom sa zaraďujú mimo alebo počas plnenia požiadavky na premiestnenie.

Na splnenie jednej požiadavky na premiestnenie sa možno pozrieť **z rôznych úrovní detailnosti**. Jeden pohľad to môže ukázať ako činnosť trvajúcu od štartu plnenia požiadavky až do jeho skončenia (napríklad cesta autobusového spoja od odchodu z autobusovej stanice v Žiline po príchod na autobusovú stanicu v Bratislave). Podrobnejší pohľad nám túto činnosť môže zobraziť ako sériu presunov, ktoré sú prerušené zastávkami a doplnené inými obslužnými činnosťami. Zastávky môžu byť buď plánované alebo vynútené (napríklad autobus sa po ceste zastaví na autobusovej stanici v Považskej Bystrici podľa cestovného poriadku a niekoľkokrát ho donúti premávka na ceste zastaviť na križovatkách). V druhom prípade by sme to mohli vnímať tiež ako proces skladajúci sa z niekoľkých činností premiestnenia, prípadne činností nástupu a výstupu cestujúcich a činností zabezpečujúcich plynutie cestnej premávky. Rozlíšenie, čo sa považuje za činnosť a čo už za proces zložený z viacerých činností závisí od toho, aký cieľ a akú úroveň podrobností chce dizajnér riadiaceho systému dosiahnuť.

Pre ďalšiu analýzu potrebujem ešte rozlíšiť uvažované technologické procesy na premiestňovacie a všeobecné (Obr. 4.1).

Za **premiestňovacie procesy** považujem procesy, ktoré sa skladajú výhradne z premiestňovacích činností doplnených nanajvýš vynútenými technologickými prerušeniami, ako sú napríklad vyššie spomenuté zastávky na križovatkách v cestnej doprave alebo zmena smeru jazdy pri presune železničného vozidla v stanici. Tieto procesy však neobsahujú iné obslužné činnosti než premiestňovacie.

Medzi **všeobecné procesy** zaraďujem všetky tie, ktoré pozostávajú z ľubovoľných obslužných činností. Spravidla zahŕňajú okrem ostatných aj premiestňovacie činnosti, hoci to nie je podmienkou. Príkladom procesu takéhoto druhu môžu byť už spomenuté autobusové spoje zastavujúce na viacerých miestach kvôli obsluhe cestujúcich alebo



Obr. 4.1 Delenie procesov podľa činností, ktoré obsahujú.

prepravy kontajnerov od dodávateľa k odberateľovi, ktoré po ceste podstúpia prekládky a prepravné kontroly.

4.1.1.2 Vykonávanie činností

Pre ďalšiu analýzu potrebujem poznať vzťahy medzi vytýčenými činnosťami z dvoch pohľadov, a to z hľadiska

- času – činnosti sa môžu vykonať postupne za sebou alebo súbežne vedľa seba,
- vzájomných vzťahov – činnosti sú vo vzťahu závislom, t. j. začiatok jednej závisí od stavu spracovania druhej činnosti, alebo nezávislom.

Typickým príkladom vykonania činností **za sebou so závislosťou** sú premiestňovacie procesy, kde jednotlivé etapy cesty nastávajú za sebou závisle: kým nie je dokončená jedna, nemôže začať nasledujúca. Na druhej strane príkladmi **nezávislých** činností sú technická prehliadka plavidla a prekládka nákladu medzi plavidlom a pevninou počas jeho pobytu v prístave. Tieto činnosti môžu prebiehať buď **súbežne** alebo **za sebou** podľa okolností. Pri minimalizácii celkového času spracovania požiadavky je snaha nezávislé činnosti spracovať súbežne.

Pre modelovanie časových závislostí medzi činnosťami sa z teórie grafov používajú sieťové grafy, resp. acyklické digrafy pre časové plánovanie (Palúch,

online). V modelovaní sieťovými grafmi existujú v zásade dva prístupy, a to keď sa na reprezentáciu činností používajú vrcholy a závislosti vyjadrujú hrany, alebo opačne.

Prvý prístup je vrcholovo-orientovaný a má výhodu v intuitívnejšom zachytení závislostí a synchronizácie činností. Nepotrebuje na to nijaké nadbytočné prvky. Druhý, hranovo-orientovaný prístup je zasa úspornejší v počte použitých prvkov v grafe, najmä pri takých grafoch, kde sa vyskytujú miesta zbiehania a rozbiehania paralelných vetiev, hoci na synchronizáciu musí v niektorých situáciách použiť nadbytočné tzv. fiktívne hrany (ktoré reprezentujú iba závislosť, nie činnosť). Ilustráciu hranovo orientovaného prístupu, ktorý používam vo svojej práci, možno nájsť na Obr. 5.2 a na Obr. 5.3.

4.1.1.3 Kategorizácia procesov

Z pohľadu RAS (ten je uvedený v kapitole 2.2.2) možno nájsť medzi premiestňovacími a všeobecnými procesmi niektoré typické odlišnosti.

V **premiestňovacom procese** sa činnosti vykonávajú v typických prípadoch sekvenčne, kde jeden úsek premiestnenia nasleduje za druhým. Zákazka má po celý čas pridelený minimálne jeden prostriedok, a to dopravnú cestu, na ktorej stojí alebo sa pohybuje. Za predpokladu, že dĺžka dopravného prostriedku alebo kompletu v zákazke je taká, že sa zmestí celý na každý úsek dopravnej cesty v systéme (a pre jednoduchosť predpokladáme, že presun medzi úsekmi je jedna nedeliteľná udalosť / prechod v Petriho sieti), tak proces nepotrebuje mať pridelený viac ako jeden prostriedok naraz. Ak by navyše boli všetky cesty pre zákazky dané, tak systém presúvania takých zákazok by sa dal modelovať najjednoduchším systémom prideľovania prostriedkov: jedno-jednotkovým úplne usporiadaným RAS. Ak existujú variantné cesty pre proces, na popísanie by bol vhodný AGV RAS (jedno-jednotkový čiastočne usporiadaný RAS, viď. kapitola 2.2.2).

Pokiaľ pripustíme, že dĺžka presúvanej jednotky je dlhšia ako niektorý úsek dopravnej cesty v systéme, potom treba jednotke pri použití kratších úsekov prideľovať viac úsekov cesty naraz. Čitateľovi je asi zrejmé, že bezprostredne susediace úseky dopravnej cesty treba považovať za rozličné typy prostriedkov, a teda ide o súčasné pridelenie prostriedkov z viacerých typov. Za predpokladu presne vytýčených ciest pre zákazky na popísanie takého systému potrebujeme **sekvenčný** viac-typový úplne usporiadaný **RAS** (MT-TO-RAS), pri existencii variantov ciest by to bol sekvenčný viac-typový čiastočne usporiadaný RAS (MT-PO-RAS).

Neusporiadaný RAS v tomto prípade len zriedka pripadá do úvahy, keďže opakovanie činností pri presune znamená pohyb aspoň jeden krát do kruhu, čo nie je efektívne vykonanie procesu. Niekedy sa to môže vyskytnúť z dôvodu bezpečnosti (napríklad lietadlo krúži vo vzduchu nad cieľovým letiskom čakajúc na pokyn pristáť), no ide o zriedkavý, neplánovaný jav.

Všeobecné dopravné **procesy** majú takisto spravidla pridelenú dopravnú cestu, kde sa objekt obsluhy nachádza. Na vykonanie obsluhy alebo pohybu však treba zvyčajne prideliť ďalšie prostriedky, ako technické zariadenie alebo personál, pričom

súčasne môže proces vlastniť viac ako jednu jednotku daného typu prostriedkov (napr. dvaja vozmajstri pri technickej prehliadke vlaku). To znamená, že proces môže potrebovať súčasne niekoľko jednotiek niekoľkých typov prostriedkov.

Z hľadiska smerovania sa môžu v spracovaní všeobecných procesov vyskytovať aj varianty, napríklad podľa obsahu zákazky – pri obsluhe vlaku rôzny počet skupín vozňov alebo výskyt, resp. absencia vozňov so zákazom jazdy cez pahorok. Spravidla však už v momente výberu je podľa situácie v systéme rozhodnuté, ktorý variant sa použije. Nevybraté varianty môžu byť znevýhodnené podľa určitého kritéria (napr. poloha uvažovaných prvkov v systéme) alebo diskvalifikované porušením podmienok ich použitia (napr. neprítomnosť potrebnej skupiny vozňov v súprave vlaku).

Opakovanie činností do kruhu sa vo všeobecných dopravných procesoch môže takisto vyskytnúť, napr. nákladné vozidlá prevážajúce sypký materiál zo skladu na stavenisko. Vo všeobecnom prípade teda uvažujeme o neusporiadanom viac-typovom systéme MT-NO-RAS. Opakovaniu činností sa možno niekedy vyhnúť napríklad tým, že sa celý cyklus nahradí jednou agregovanou činnosťou. Potom by stačila kategória MT-PO-RAS, teda čiastočne usporiadaný viac-typový RAS. Keďže aj variantné smerovanie sa vyskytuje zriedka, tak stačí spravidla MT-TO-RAS, teda úplne usporiadaný viac-typový RAS.

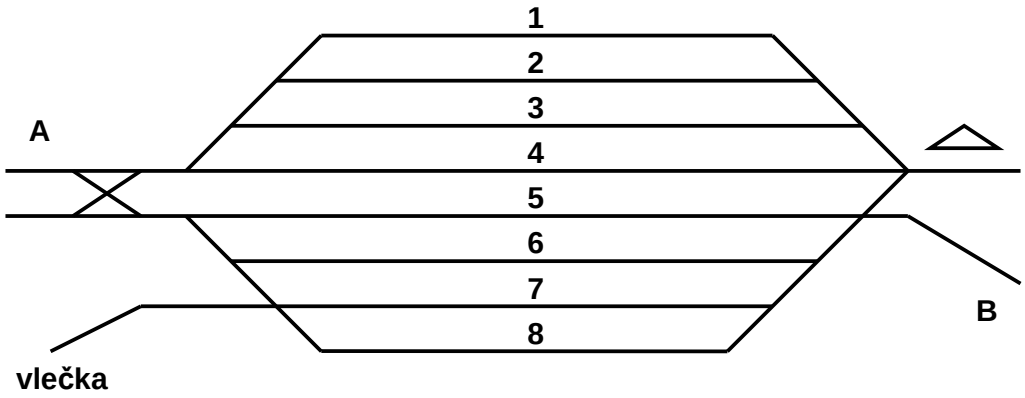
Ako som už naznačil v predchádzajúcej časti, činnosti môžu pri spracovaní zákazky prebiehať aj súbežne, čo rozdelí priebeh procesu na viac ako jednu vetvu. Typicky sa činnosti po začiatku procesu rozdelia do paralelných vetiev, ktoré sú na niektorých miestach synchronizované a na konci sa zlúčia do jedného bodu, kde vykonanie procesu končí (príklad na Obr. 5.2 alebo Obr. 5.3). V takom prípade sa proces už označuje ako nesekvenčný a na popísanie systému, kde je aspoň jeden taký proces, potrebujeme **nesekvenčný RAS**.

Z uvedeného vychádza, že dopravný systém všeobecne patrí do kategórie **nesekvenčných viac-typových neusporiadaných RAS (nesekvenčných MT-NO-RAS)** a podľa absencie spomínaných vlastností možno konkrétny systém zaradiť aj do niektorej kategórie pre menej zložité systémy. Otvorenou otázkou zostáva, čo by prinieslo oddelené modelovanie premiestňovacích procesov v rámci jednoduchšieho sekvenčného a ostatných procesov v nesekvenčnom MT-NO-RAS (prípadne MT-TO-RAS).

V ďalšom riešení som sa zameral na úplný model dopravného systému ako nesekvenčný MT-PO-RAS (teda s vylúčením cyklického opakovania sa činností), ktorého považujem za typického predstaviteľa všeobecného dopravného systému.

4.1.1.4 Detaily k prideľovaniu prostriedkov

Aby bola analýza úplná, treba sa ešte zmieniť aj o spôsoboch prideľovania prostriedkov v dopravnom systéme. Vyššie už bolo spomenuté, že procesy všeobecne môžu potrebovať viac prostriedkov viacerých typov naraz. Okrem toho treba zohľadniť ešte dve okolnosti.



Obr. 4.2 Schéma koľajiska pre aplikáciu použitia profesií koľají.

Prvou okolnosťou je fakt, že prostriedky môžu byť jednému procesu pridelené a uvoľnené aj viac ako jedenkrát počas jeho spracovania. Napríklad v malom prístave, ktorý má úzky vjazd do prístavného bazéna prispôsobený na plavbu iba jednej lode, potrebuje dostať plavidlo tento vjazd na vplávanie do prístavného bazéna ako aj na neskoršie vyplávanie z neho. Na začiatku ho dostane pridelený a po vplávaní ho uvoľní pre ďalšie plavidlá. Po skončení obsluhy ho plavidlo opäť použije na opustenie prístavného bazéna. To znamená, že proces tento prostriedok postupne dostane pridelený, uvoľní, dostane pridelený a opäť uvoľní. Na túto okolnosť sa budem v ďalšom texte odvolávať výrazom **opakované pridelenie prostriedku procesu**.

Druhou okolnosťou sú vlastnosti, ktoré môžu prostriedky rozdeliť do viacerých množín, pričom jeden prostriedok môže patriť do viac ako jednej množiny. K rozdeleniu prostriedkov dochádza podľa tzv. **profesií**, čo sú pomenovania ich príslušnosti k množinám. Typické použitie je napríklad v železničných staniciach, kde koľaje možno označiť profesiami podľa ich určenia, napríklad staničná koľaj, traťová koľaj, odstavná koľaj a pod.

Majme napríklad 8-koľajovú vchodovú skupinu zriaďovacej stanice – koľaje 1–8 (Obr. 4.2). Kým zo smeru A sú dostupné všetky koľaje skupiny, z trate zo smeru B ústiacej zo strany pahorka je dostupná iba polovica z nich: koľaje 5–8. Ďalej koľaje 7 a 8 sú vyhradené aj na výmenu vozňov s priľahlou vlečkou a koľaje 4 a 5 slúžia na presun posunovacej zálohy z pahorka ku vlakovej súprave pred rozradením. Pri rôznych žiadostiach o pridelenie koľaje tak možno vyberať vždy z príslušnej množiny koľají podľa profesie. To spôsobuje, že niektoré koľaje sa vyskytujú vo viac ako jednej množine, napr. koľaj č. 5 až v troch.

Opakované pridelenie a použitie profesií pri výbere sú okolnosti, ktoré sú v zložitých systémoch bežné, ba priam nevyhnutné, a treba ich pri spravovaní prostriedkov v dopravnom systéme zohľadniť.

4.1.2 Model dopravného systému v Petriho sieti

Petriho sieť je vhodný modelovací nástroj pre analýzu prideľovania prostriedkov. A tak jednou z prvých úloh je vytvorenie Petriho siete pre skúmaný dopravný systém. Túto úlohu možno rozdeliť na podúlohy, a to modelovanie premiestňovacích procesov a modelovanie všeobecných obslužných procesov.

Podľa tab. 2.3 a kapitoly 4.1.1.3 stačí v prvom prípade na modelovanie procesu procesná Petriho sieť (P^2N) a celý systém je v sieti S^4PR (uvedená v kapitole 2.2.3.1), ktorá vzniká zlúčením všetkých P^2N. V druhom prípade sa predpokladá prítomnosť aspoň jedného nesekvenčného procesu, modelovaného Petriho sieťou nesekvenčného procesu (NP^2N) a systém, ktorý ich obsahuje, potrebuje pre svoj model sieť GS^*PR.

V nasledujúcich častiach ukážem, ako možno zapísať obidva druhy dopravných procesov v Petriho sieti. Keďže pri všeobecných obslužných procesoch ide o zložitejšiu operáciu, pozostávajúcu z prevodu sieťového grafu na Petriho sieť a následného doplnenia ostatných náležitostí pre NP^2N, začnem od nich. Jednoduchšia operácia zápisu premiestňovacích procesov nasleduje za tým.

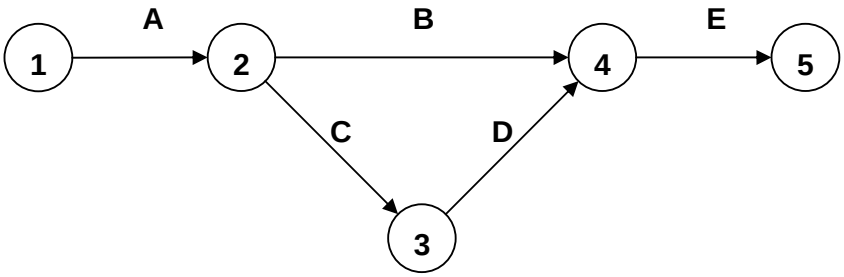
4.1.2.1 Prevod sieťového grafu na Petriho sieť

Štartovacím bodom pre všeobecné obslužné procesy je sieťový graf (SG), ako som spomenul v kapitole 4.1.1.2. Prvou úlohou teda je previesť SG na Petriho sieť.

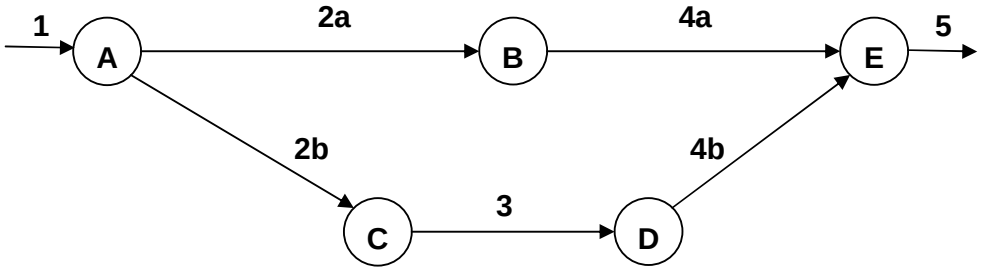
Sieťový graf, ako je všeobecne známe, pozostáva z vrcholov jedného druhu a orientovaných hrán, kde jeden vrchol pôsobí ako zdroj a iný vrchol ako ústie (Palúch, online). Jeho vykonanie sa v zásade zhoduje s princípom metódy kritickej cesty (Critical Path Method, CPM), t. j. obsluha začína v zdroji grafu a končí v ústí, pričom sa všetky činnosti medzi zdrojom a ústím vykonávajú v čase. Podľa toho, ktorý prvok znázorňuje činnosť, môžu byť dve verzie (kapitola 4.1.1.2). Obidve však zaisťujú, že v bode rozvetvenia sa vyberie vykonávanie súbežne všetkými vetvami a nie iba jednou (ako by to bolo v prípade stavového stroja). Príklad možno nájsť na Obr. 4.3 a) a b).

Sieťový graf je principiálne to isté ako **značený graf**, ktorý je podtriedou PN, ako sa uvádza v kapitole A.1.3. Z toho priamo vyplýva, že SG možno znázorniť prvkami PN. V prípade hranovo orientovanej verzie sa pri prevode vrchol SG nahradí prechodom a hrana SG postupnosťou hrana-miesto-hrana v Petriho sieti. V prípade druhej verzie je to analogické s malou úpravou: vrcholy sa zmenia na miesta a hrany na (hrana-prechod-hrana), pričom výstup viac hrán i vstup viac hrán v SG sa vždy nahradí príslušným výstupom/vstupom z/do miesta iba jednej hrany.

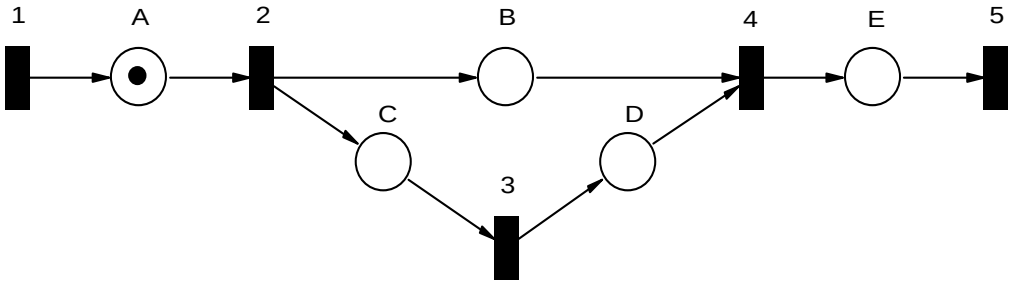
V príklade prevodu technologického postupu s činnosťami A až E na Obr. 4.3 je zdrojom vrchol 1 (a) , resp. hrana 1 (b) v SG a prechod 1 v PN (c). Analogicky ústie reprezentujú prvky s označením 5. Spomenutú úpravu si možno všimnúť v realizácii synchronizačného bodu 4, ktorý je vo vrcholovo orientovanom SG tvorený hranami 4a a 4b.



a)



b)



c)

Obr. 4.3 Prevod sieťového grafu časového plánovania na Petriho sieť: a) hranovo-orientovaný sieťový graf, b) uzlovo-orientovaný sieťový graf, c) Petriho sieť. Písmená A-E predstavujú činnosti, očíslované prvky zabezpečujú ich synchronizáciu.

Vykonanie technologického postupu v takejto Petriho sieti sa zabezpečí postupným presunom značiek od začiatku siete (zdroja) na koniec (ústie) v smere šípok. Z definície vyplýva, že značený graf môže vytvárať nové značky alebo rušiť už

vytvorené. V sledovaných sieťových grafoch to znamená rozvetvenie vykonávania technologických činností do paralelných vetiev grafu, ktoré sa tak môžu vykonávať súbežne, a neskôr ich zlúčenie. Na druhej strane značený graf neobsahuje konflikty, t. j. každé miesto má práve jeden vstupný a jeden výstupný prechod.

4.1.2.2 Modelovanie všeobecných obslužných procesov Petriho sieťou

Aby sa z vytvorenej siete stala PN nesekvenčného procesu podľa poslednej definície v kapitole 2.2.3.1, treba ešte urobiť niekoľko úkonov. Vytvorená sieť z predchádzajúcej časti obsahuje dosiaľ množinu miest pre fázy procesu P_S a množinu prechodov T z uvedenej definície.

Najjednoduchším úkonom je zavedenie miesta pre nečinné procesy p_0 a jeho pripojenie hranami z prechodu ústia siete a do prechodu zdroja. Tým sa doplní množina $P_0 = \{ p_0\}$ a zabezpečí sa plynulý prechod od vykonanej inštancie procesu (podľa technologického postupu) k ďalšej, novej inštancii na spracovanie.

Ďalej nasleduje namodelovanie prideľovania a uvoľňovania prostriedkov procesu. Tieto úkony možno nazvať **riadiace činnosti** a modelujú sa odlišne v sieťovom grafe obslužnej technológie, z ktorého vychádzame, a v Petriho sieti, ku ktorej smerujeme. V sieťovom grafe sú explicitne zachytené iba riadiace činnosti, ale nie prostriedky. Ich modelovanie sa predpokladá v oddelených údajových štruktúrach, ktoré sa používajú v algoritmoch riadiacich činností technologického grafu. To by podľa opísaného prevodu do Petriho siete znamenalo zmeniť stav prostriedkov v rámci miesta, čo odporuje princípu Petriho siete (zmeny stavu sa dejú vykonávaním prechodov).

Preto treba pri tvorbe PN nesekvenčného procesu urobiť dve operácie: zaviesť miesta pre prostriedky a vybrať prechody, ktoré vykonajú pridelenie/uvoľnenie prostriedku k/od procesu. Pre každý typ prostriedku sa vytvorí jedno oddelené miesto $p_i \in P_R$, pričom každé miesto bude v počiatočnom značení obsahovať toľko značiek, koľko prostriedkov daného typu sa v systéme nachádza. Pripojenie týchto miest k prechodom množiny T je nasledovné:

a) Pre riadiacu činnosť modelovanú miestom p_j, ktorá žiada pridelenie prostriedku typu r_i, sa zavedie hrana z miesta p_i do prechodu $p_j^{\bullet}$, t. j. do výstupného prechodu, ktorý je iba jeden (vzhľadom na pôvod PN zo sieťového grafu).
b) Pre riadiacu činnosť modelovanú miestom p_j, ktorá uvoľňuje prostriedok typu r_i, sa do miesta p_i zavedie hrana z prechodu ${}^{\bullet}p_j$, t. j. zo vstupného prechodu, ktorý je taktiež iba jeden.

Aby vytváraná PN zodpovedala presne definícii PN nesekvenčného procesu, zostáva z nej naplniť podmienky (2) až (5).

Keďže značený graf je obyčajná, konzervatívna a konzistentná Petriho sieť bez vlastných cyklov, spĺňa vlastnosti v podmienke (2).

Ďalšia podmienka vyjadruje skutočnosť, že každý prostriedok, ktorý sa procesu počas spracovania pridelí, sa neskôr aj vráti do množiny voľných prostriedkov. To by

malo byť splnené z charakteru procesu modelovaného pôvodným sieťovým grafom, no prakticky to závisí od správnosti vytvoreného grafu. Keďže v reálnom dopravnom systéme uvažujeme výhradne o prostriedkoch, ktoré sa nespotrebovávajú, ani nevytvárajú, je táto podmienka (pri správnom modeli sieťovým grafom) splnená.

Podmienka (4) požaduje, aby každá fáza procesu okrem nečinného stavu mala pridelený aspoň jeden prostriedok. Keďže, ako som uviedol pri rozbore Coffmanových podmienok v kapitole 4.2.2, každý proces v dopravnom systéme obsahuje počas spracovania ako prostriedok minimálne úsek dopravnej cesty, na ktorom sa nachádza, je táto podmienka automaticky splnená.

Posledná podmienka nastoľuje, ako má vyzerať počiatočné značenie vytvorenej PN – to sa týka dynamickej dimenzie Petriho siete, ktorá sa nenachádza v pôvodnom SG. Ten opisuje postup spracovania jednej inštancie procesu, je to statický model. To by v Petriho sieti zodpovedalo jednej značke v mieste p_0 (v definícii je tiež uvedená jednotka, lebo ide o definíciu jedného procesu). Keďže však v PN môžeme modelovať vďaka značkám viac procesov, t. j. viac inštancií rovnakého postupu spracovania procesu naraz, možno tento fakt zohľadniť v obsahu miesta p_0 na začiatku a uviesť tam počet značiek zodpovedajúci počtu procesov v reálnom systéme. Zvyšok tejto podmienky nastoľuje počiatočné podmienky dynamiky systému, t. j. nijaký proces nie je rozpracovaný a každý typ prostriedkov je zastúpený neprázdnou množinou.

Záverom teda možno zhrnúť, že všeobecnému obslužnému procesu zodpovedá Petriho sieť nesekvenčného procesu, v ktorej podsieť $(P_S \cup P_0)\times T$ je **značeným grafom**, ak má proces iba jeden variant vykonania. Ak má viac variantov vykonania (pružného smerovania procesu), tak podsieť tvorí **PN s voľným výberom**. To sa v nej prejaví tak, že aspoň z jedného miesta podsiete vychádza viac ako jedna hrana a aspoň do jedného miesta (nemusí byť to isté) vchádza viac ako jedna hrana.

4.1.2.3 Modelovanie premiestňovacích procesov Petriho sieťou

Premiestňovacie procesy sa v zásade modelujú analogicky ako všeobecné obslužné procesy v predchádzajúcej časti.

Prostriedkami pre presun sú úseky dopravnej cesty. Pod úsekom dopravnej cesty si možno predstaviť najmenšiu, ďalej už nedeliteľnú jednotku dopravnej cesty takej dĺžky, že ju môže obsadiť iba jeden proces (vozidlo, komplet). Napríklad v mestskej cestnej sieti môžu mať úseky ciest dĺžku zodpovedajúcu dĺžke priemerného osobného vozidla a medzere medzi vozidlami pri jazde, alebo v železničnej doprave môže byť úsekom dopravnej cesty jeden traťový oddiel, staničná koľaj alebo výhybka.

Tak, ako pri všeobecných prostriedkoch, aj úseky rozdeľujeme do množín R_i, i = 1, 2, ..., k, kde jedna množina reprezentuje jeden typ prostriedkov a k počet takýchto množín. V tvorenej sieti P^2N sa budú voľné prostriedky daného typu sústrediť do jedného miesta z množiny miest pre typy prostriedkov P_R. Rozdelenie úsekov do množín R_i závisí od presnosti modelovania dopravnej cesty zvolenej tvorcom modelu. V krajnom prípade každý úsek tvorí samostatnú množinu (typ prostriedku) a počet

miest v množine P_R potom zodpovedá počtu všetkých úsekov v sieti. Často je však žiaduce zoskupiť niektoré úseky do jednej množiny prostriedkov, kde každý úsek reprezentuje jeden prostriedok a procesy si môžu vyžiadať ktorýkoľvek z nich (napríklad paralelné staničné koľaje pre osobnú vlakovú prepravu alebo paralelné odbavovacie miesta pre nákladné vozidlá na cestnom hraničnom priechode). Taktiež sa ako veľmi výhodné javí súvislú podsieť, ktorej úseky sa vždy prideľujú najviac jednému procesu a netreba ich ďalej deliť, nahradiť jedným z množiny prostriedkov. Takým je napríklad koľajové zhlavie malej železničnej stanice, ktoré sa vždy môže obsadiť naraz iba jedným vlakom. Ak možno na zhlaví postaviť paralelné cesty, do úvahy prichádza rozdelenie zhlavia na toľko častí, koľko ciest možno naraz postaviť. Každá časť bude samostatným prostriedkom.

V príklade schematického koľajiska na Obr. 5.1 sa vytýčili všetky koľaje, ktoré môžu byť cieľom pohybu a umiestnili sa do množín podľa svojho určenia – tak vznikli tri množiny prostriedkov: *traťová koľaj*, *staničné koľaje* a *depo* obsahujúce jednu, tri, resp. jednu koľaj. Zostávajúce spojovacie koľaje a výhybky medzi nimi tvoria množiny *zhlavie 1* a *zhlavie 2*.

Rozdelenie procesu premiestnenia na fázy takisto závisí od konkrétneho modelovaného dopravného systému a od potrebnej presnosti modelovania. Napr. v železničnej doprave možno vyhradiť dlhú cestu pre jazdu vlaku, pozostávajúcu z k prostriedkov, t. j. prideliť procesu naraz všetkých k prostriedkov, alebo ju rozdeliť na n čiastkových ciest, obsahujúcich postupne k_1, k_2, ..., k_n prostriedkov ($k = k_1 + k_2 + ... + k_n$) a postupne ich procesu prideľovať a uvoľňovať. Jedna čiastková cesta (fáza procesu) môže potrebovať jeden alebo viac prostriedkov naraz.

Doplnenie miesta p_0 a splnenie ďalších podmienok siete P^2N je analogické ako pri všeobecných obslužných procesoch. Potom premiestňovaciemu procesu zodpovedá procesná Petriho sieť, v ktorej podsieť $(P_S \cup P_0) \times T$ je **cyklus**, ak má proces iba jeden variant vykonania. V prípade viac variantov vykonania (pružného smerovania procesu) je táto podsieť **stavovým strojom**, t. j. obsahuje aspoň jedno miesto, z ktorého vychádza viac ako jedna hrana a aspoň jedno miesto, do ktorého vchádza viac ako jedna hrana.

4.1.2.4 Vytvorenie modelu vo farbenej Petriho sieti

Pred vytvorením modelu dopravného systému v Petriho sieti si tvorca potrebuje ujasniť, akej povahy sú technologické procesy, ktoré ide zobrazovať. Podľa toho si zvolí pre každý z nich jeden z postupov uvedených v predchádzajúcich častiach. Výsledkom budú procesné Petriho siete (P^2N) pre premiestňovacie procesy a Petriho siete nesekvenčných procesov (NP^2N) pre všeobecné procesy. Vytvorené siete budú podsieťami v Petriho sieti reprezentujúcej celý systém. Počet podsietí sa rovná počtu rozličných modelovaných technologických postupov, resp. premiestnení. Navzájom prepojené budú pomocou miest z množiny P_R, ktoré modelujú prostriedky v systéme zdieľané všetkými procesmi.

Niektoré dopravné systémy svojou veľkosťou môžu viesť k rozľahlým a na prvky bohatým modelom v Petriho sieti. Preto sa javí relevantnou otázka, či použitie formalizmu Petriho siete vyššej úrovne neprinesie zjednodušenie modelu (viď kapitola A.1.4). Z možností širokej škály PN vyššej úrovne sa na riešenie tejto otázky perspektívnou javí **hierarchická farbená Petriho sieť** (hierarchická CPN). Pri skúmaní tejto otázky som došiel k nasledujúcim zisteniam.

Predpokladajme, že máme v systéme

- n typov procesov (t. j. unikátnych predpisov), tvoriacich n podsietí P^2N alebo NP^2N ($i = 1..n$), a
- k typov prostriedkov ($j = 1..k$).

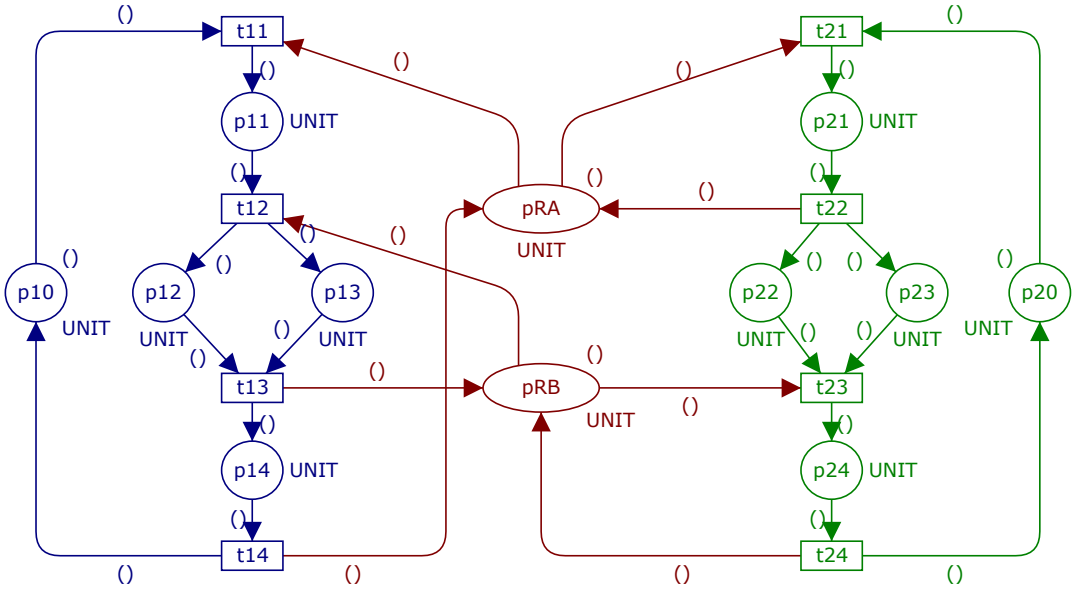
Ak pre každý typ procesu i a pre každý typ prostriedku j zavedieme farby, potom možno vykonať nasledovné:

1) Zlúčiť miesta p_0^i pre nečinnú fázu procesov i-teho typu do jedného miesta p_0 pre všetky typy procesov (resp. nahradiť n množín P_0^i jednou množinou P_0).
2) Zlúčiť miesta p_R^j pre voľné prostriedky j-teho typu do jedného miesta p_R pre všetky typy prostriedkov (resp. nahradiť k prvkov v množine P_R jedným prvkom: $P_R = \{p_R\}$).
3) Zlúčiť miesta $p \in P_S^i$ pre fázy a im zodpovedajúce prechody $t \in T^i$, ktoré spolu reprezentujú štruktúru spracovania i-teho typu procesov za predpokladu, že sa vo viacerých procesoch vyskytne **zhodná štrukturálna vzorka**.

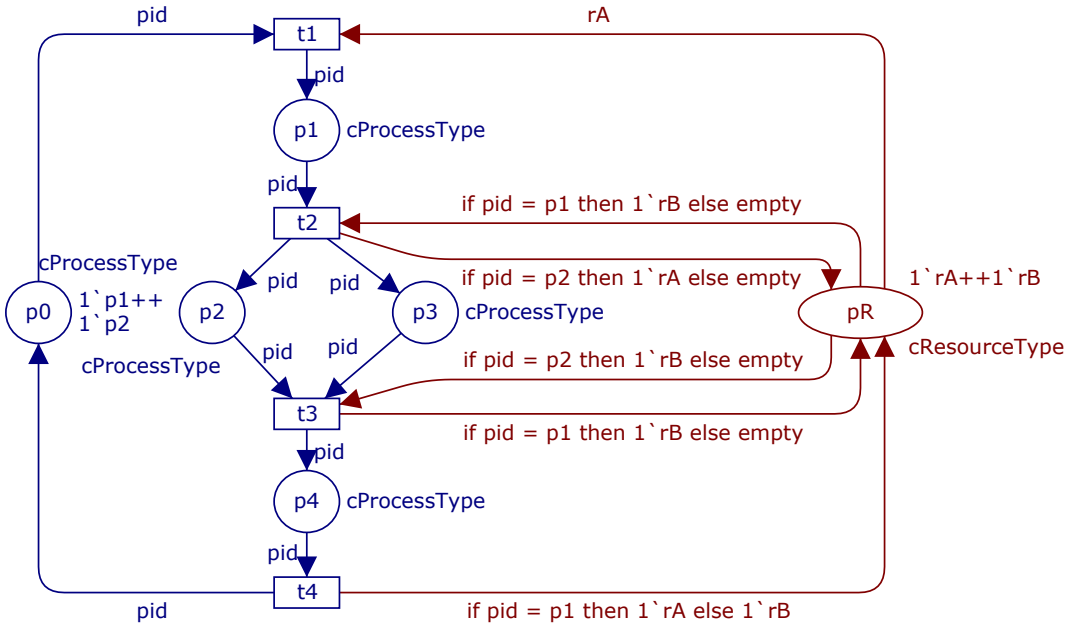
Vykonanie operácie podľa bodu 3 teda predpokladá, že existujú podsiete generované $P_R \cup P_S^1 \cup T^1$ pre typ procesov 1 a $P_R \cup P_S^2 \cup T^2$ pre typ procesov 2, ktoré majú zhodnú štruktúru miest, prechodov a hrán medzi nimi. Potom možno spoločnú podsieť vytvoriť iba jedenkrát a pripojiť ju do všetkých podsietí i, v ktorých sa táto štruktúra vyskytuje. Možno tu využiť modularitu v rámci hierarchickej štruktúry farbenej Petriho siete a opakujúcu sa podsieť zostaviť ako samostatný modul, ktorý sa v rámci hierarchie použije na príslušných miestach modelu.

Čo však urobiť v situácii, kde štruktúra viacerých podsietí nie je zhodná, ale podsiete sa podobajú? Napríklad na Obr. 4.4a sa nachádza sieť GS^*PR dvoch typov procesov (zobrazená prostriedkami PN nižšej úrovne), ktoré majú zhodnú štruktúru fáz vykonania, avšak rozdielne poradie pridelenia a uvoľnenia prostriedkov. Obidva sa spracúvajú najskôr vo fáze 1, na čo potrebujú prostriedok typu A. Potom prebiehajú súbežne fázy 2 a 3, kde typ 1 potrebuje obidva prostriedky, typ 2 ani jeden. Napokon je fáza 4, v ktorej obidva typy procesov využívajú prostriedok typu B.

Iná situácia je na Obr. 4.5a, kde je pridelenie a uvoľnenie prostriedkov jednotlivým fázam zhodné, avšak postupy sa líšia v štruktúre fáz (proces typu 2 nemá súbežné spracovanie v strednej časti).

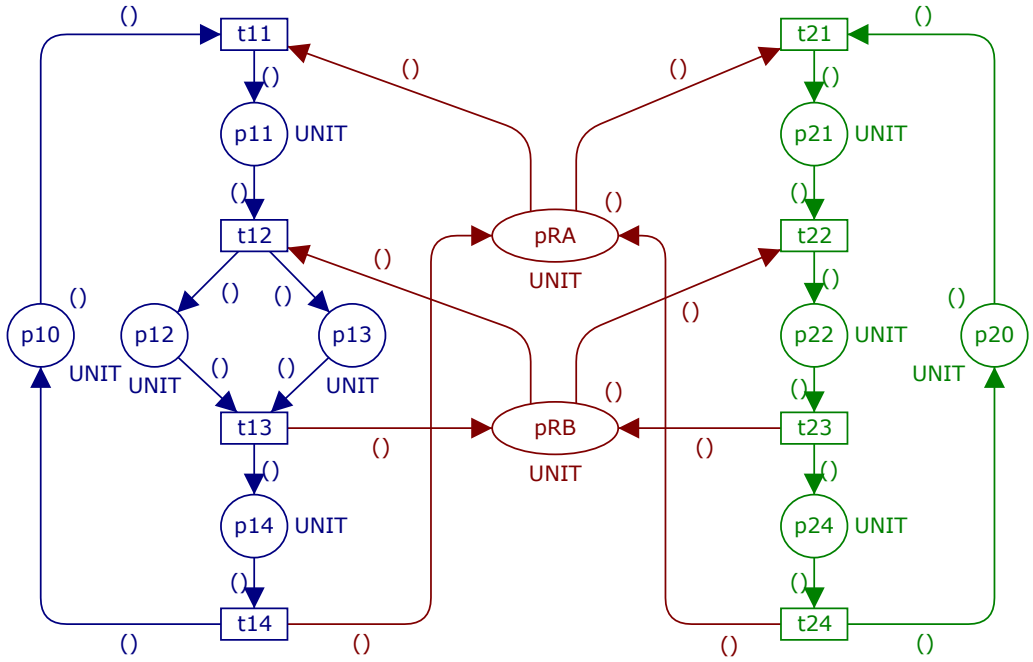


a)

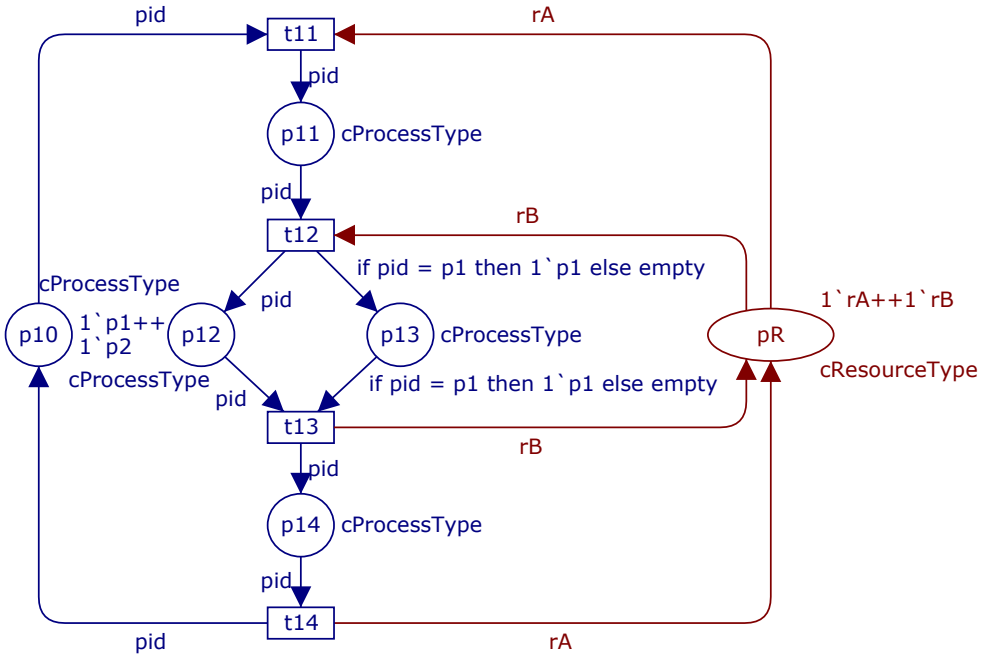


b)

Obr. 4.4 Dva procesy rovnakej štruktúry s rôznym poradím pridelenia a uvoľnenia prostriedkov: a) GS*PR, b) SNP-CPN.



a)



b)

Obr. 4.5 Dva procesy podobnej štruktúry s rovnakým poradím pridelenia a uvoľnenia prostriedkov: a) GS*PR, b) SNP-CPN.

Intuitívne možno predpokladať, že v obidvoch prípadoch sú štruktúry obidvoch typov procesov natoľko podobné, že ich ešte možno zlúčiť, ako je to znázornené vo výsledných farbených Petriho sieťach na Obr. 4.4b a Obr. 4.5b, za pomoci podmienkových výrazov na hranách. Čím sú štrukturálne vzorky odlišnejšie, tým viac, príp. tým zložitejšie podmienkové výrazy sa musia vyskytovať na hranách. Od určitého bodu môže ich zložitosť prevýšiť zjednodušenie získané zjednocovaním vzoriek. Preto možno zaviesť pojem **podobná štrukturálna vzorka**, ktorého presné vymedzenie závisí od situácie a zámerov tvorcu modelu. Nie je vhodné určovať túto hranicu všeobecne pre ľubovoľný prípad.

Zavedením farieb do Petriho siete teda možno zmenšiť počet miest v modeli určite o $n + k - 2$, kde n a k je počet modelovaných typov procesov, resp. typov prostriedkov – to je podľa prvých dvoch krokov, ktoré nemajú nijaký vplyv na počet prechodov a hrán v sieti. Ak predpokladáme, že každý proces je popísaný štruktúrou aspoň z 10 prvkov (miesta, prechody a hrany) a typicky niekoľkonásobne viac prvkov, tak je zmenšenie siete minimálne.

A tak najväčšiu úsporu v modeli možno dosiahnuť krokom 3, ktorá však závisí najmä od počtu a podobnosti štrukturálnych podsietí a od zámeru tvorcu modelu pri zlučovaní podsietí. Pri nepriaznivej štruktúre siete (t. j. veľa rozdielnych typov procesov) môže byť úspora malá. Čím viac sa typy procesov podobajú, tým je možnosť zlučovania väčšia a zväčšuje sa aj efekt zavedenia CPN.

V každom prípade prináša CPN výhodu v sprehľadnení organizácie hrán v podsieti danej $P_R \cup T_i$, keď sa redukuje počet nútených križovaní hrán oproti základnej PN.

Na spresnenie postupu nasledujú definície spomínaných sietí.

Definícia 4-1

Petriho sieťou nesekvenčného procesu s pružným smerovaním nazveme takú značenú Petriho sieť $\mathcal{N} = (P, T, F, W, M_0)$, pre ktorú platí:

(1) P je rozložená nasledovne: $P = P_0 \cup P_S \cup P_R$, pričom platí:

(a) $P_0 = \{p_0\}$,

(b) $P_S \cap P_R = P_S \cap P_0 = P_R \cap P_0 = \emptyset$,

(c) $P_S \neq \emptyset$,

(d) $P_R = \{r_1, r_2, ..., r_k\} \neq \emptyset, k > 0$.

(2) Podsieť generovaná množinou $P_0 \cup P_S \cup T$, $\mathcal{N}_{(P_0 \cup P_S \cup T)}$ je obyčajná, konzervatívna a čiastočne konzistentná Petriho sieť bez vlastných cyklov.

(3) Pre $\forall r \in P_R$ existuje neprázdna množina $\Upsilon_r = \{\mathbf{Y}_r^1, \mathbf{Y}_r^2, ..., \mathbf{Y}_r^{k_r}\}$ minimálnych p-poltokov taká, že pre každé $i \in \{1, 2, ..., k_r\}$ platí:

(a) $\{r\} = \left\| \mathbf{Y}_r^i \right\| \cap P_R$,

(b) $\{p_0\} \cap \left\| \mathbf{Y}_r^i \right\| = \varnothing$.

(4) $P_S = \bigcup_{r \in P_R} \bigcup_{i \in \{1, 2, ..., k_r\}} \left(\left\| \mathbf{Y}_r^i \right\| \setminus \{r\} \right)$.

(5) $M_0(p_0) = 1,\ \forall p \in P_S : M_0(p) = 0,\ \forall r \in P_R : M_0(r) \geq 0$.

■

Petriho sieť nesekvenčného procesu s pružným smerovaním má definíciu takmer zhodnú s definíciou autorov v publikácii (Ezpeleta-Recalde, 2004), ktorú som uviedol v kapitole 2.2.3.1 pod upraveným názvom PN nesekvenčného procesu. Jediný rozdiel spočíva v tom, že zavedením pružného smerovania sa podmienka *konzistentnosti* podsiete v bode (2) mení na *čiastočnú konzistentnosť*. Tento malý detail je už zahrnutý v nasledujúcej definícii, ktorá všeobecne umožňuje procesu paralelné spracovanie a pružné smerovanie zároveň.

Definícia 4-2

Farbenou Petriho sieťou nesekvenčného procesu (Coloured Petri Net of Non-sequential Process, NP-CPN) nazveme takú farbenú Petriho sieť $\mathcal{CPN} = (\sum, P, T, A, N, C, G, E, I)$, pre ktorú platí:

(1) Množina typov (množín farieb) $\sum$ je rozložená nasledovne: $\Sigma = \Sigma_O \cup \Sigma_R$, pričom platí:

(a) $\Sigma_O = \{o\}$, kde o je farba typu procesu,

(b) $\Sigma_R = \{r_1, r_2, ..., r_k\} \neq \varnothing, k > 0$, kde r_1, r_2, ..., r_k sú typy prostriedkov v systéme.

(2) Množina miest P je rozložená nasledovne: $P = P_0 \cup P_S \cup P_R$, pričom platí:

(a) $P_0 = \{p_0\}$, kde p_0 je miesto pre nečinnú fázu procesu

(b) $P_S \neq \varnothing$,

(c) $P_R = \{p_R\}$, kde p_R je miesto pre prostriedky všetkých typov z množiny Σ_R,

(d) $P_S \cap P_R = P_S \cap P_0 = P_R \cap P_0 = \varnothing$.

(3) Množina hrán A je rozložená na dve disjunktívne podmnožiny: $A = A_{0S} \cup A_R$.

(4) Pre funkciu vrcholov N platí:

$$N: \begin{cases} A_{0S} \rightarrow ((P_0 \cup P_S) \times T) \cup (T \times (P_0 \cup P_S)) \\ A_R \rightarrow (P_R \times T) \cup (T \times P_R) \end{cases}$$

(5) Pre funkciu farieb *C* platí:

$$C: \begin{cases} (P_0 \cup P_S) \to \Sigma_O \\ P_R \to \Sigma_R \end{cases}$$

(6) Pre funkciu hranových výrazov *E* platí:

$$E: \begin{cases} A_{0S} \to \{\forall a \in A_{0S} : (E(a) = o \vee Type(Var(E(a))) = \sum_O)\} \\ A_R \to \{\forall a \in A_R : (Type(E(a)) = (\sum_R)_{\mathrm{MS}} \wedge Type(Var(E(a))) = \sum_R)\} \end{cases}$$

(7) Pre inicializačnú funkciu *I* platí:

$$I: \begin{cases} P_0 \to \{o\}_{\mathrm{MS}} \\ P_S \to \varnothing \\ P_R \to \{r_1, r_2, ..., r_k\}_{\mathrm{MS}}, k > 0 \end{cases}$$

(8) Podsieť generovaná množinou $P_0 \cup P_S \cup T$, $\mathcal{CPN}_{(P_0 \cup P_S \cup T)}$ = (Σ_O, $P_0 \cup P_S$, *T*, A_{0S}, *N*, *C*, *G*, *E*, *I*), je konzervatívna a čiastočne konzistentná Petriho sieť bez vlastných cyklov.

(9) Pre $\forall r \in \Sigma_R$ existuje neprázdna množina $\Upsilon_r = \{\mathbf{Y}_r^1, \mathbf{Y}_r^2, ..., \mathbf{Y}_r^{k_r}\}$ minimálnych p-poltokov taká, že pre každé $i \in \{1, 2, ..., k_r\}$ platí:

(a) $\{p_R\} = \|\mathbf{Y}_r^i\| \cap P_R$,

(b) $\{p_0\} \cap \|\mathbf{Y}_r^i\| = \varnothing$,

(c) $P_S \cap \|\mathbf{Y}_r^i\| \neq \varnothing$,

(d) $\mathbf{Y}_r^i[r] = 1$.

(10) $P_S = \bigcup_{r \in P_R} \bigcup_{i \in \{1, 2, ..., k_r\}} (\|\mathbf{Y}_r^i\| \setminus \{p_R\})$

■

(1) V tejto Petriho sieti sa nachádzajú iba dve množiny farieb: množina pre typ modelovaného procesu $\sum_O$ s jedinou farbou *o* a množina pre typy prostriedkov v systéme $\sum_R$, ktorá obsahuje toľko farieb, koľko rôznych typov prostriedkov sa v systéme nachádza.

(2) Množina miest *P* sa podobne ako pri sieťach nižšej úrovne (P^2N a NP^2N) rozloží na tri disjunktívne podmnožiny: podmnožinu P_0 s jediným miestom pre procesy v nečinnej fáze, neprázdnu podmnožinu P_S pre fázy činnosti procesu a podmnožinu P_R pre modelovanie typov prostriedkov. Na rozdiel od P^2N a NP^2N však stačí pre všetky typy prostriedkov z $\sum_R$ jediné miesto p_R – vďaka použitiu farieb.

(3) + (4) Množina hrán *A* je tvorená dvoma disjunktívnymi podmnožinami, ktoré delia hrany na dva druhy: jeden spája miesta pre fázy procesu s prechodmi, druhý

miesta pre typy prostriedkov s prechodmi (ako vidno z definície funkcie vrcholov). Rozdelenie množiny hrán sa s výhodou používa v ďalších bodoch definície.

(5) + (6) Miesta pre nečinnú fázu P_0 a činné fázy procesu P_S obsahujú značky iba jednej farby pre typ modelovaných procesov a miesto pre typy prostriedkov obsahuje iba značky farieb pre typy prostriedkov. Podobne sa vyhodnocujú hranové výrazy v týchto podsieťach – môžu byť definované ako jedna značka typu procesu alebo premenná takéhoto typu. To zároveň zabezpečuje vlastnosť ekvivalentnú obyčajnosti siete NP^{2}N. Na hranách z a do miest pre typy prostriedkov môže byť značiek pre prostriedky i pre typy prostriedkov viac.

(7) Inicializácia siete je rovnaká ako pri sieti NP^{2}N, teda všetky inštancie procesov (vyjadrené multi-množinou prvku *o*) sa nachádzajú v nečinnom stave a každý typ prostriedkov je zastúpený aspoň jednou inštanciou. Ak má mať popísaný typ procesu šancu na dokončenie, musí byť značenie popisujúce typ prostriedkov dostatočné pre vykonanie aspoň jedného behu spracovania typu procesu.

(8) V tejto podmienke je zachytená už aj možnosť pružného smerovania procesu, ktorá sa explicitne v názve novej siete nenachádza. Je daná podmienkou iba *čiastočnej* konzistentnosti podsiete pre fázy procesu. Oproti definícii NP^{2}N vypadla podmienka obyčajnosti, ktorá je zachytená v bode (6). Konzervatívnosť a konzistentnosť platia pre farbené Petriho siete pri použití pravidiel operácií s multi-množinami rovnako ako pre P/T Petriho siete.

(9) a (10) sú rovnaké ako pri sieti NP^{2}N.

Definícia 4-3

Nech $I_{\mathcal{N}} = \{1, 2, ..., n\}$ je konečná neprázdna množina indexov a $\mathcal{NP}\text{-}\mathcal{CPN}^i$ = ($\sum^i$,P^i, T^i, A^i, N^i, C^i, G^i, E^i, I^i), $i \in I_{\mathcal{N}}$ sú farbené Petriho siete nesekvenčných procesov.

Farbenou Petriho sieťou systému s nesekvenčnými procesmi (Coloured Petri Net of System with Non-sequential Processes, SNP-CPN) budem nazývať súvislú farbenú Petriho sieť bez vlastných cyklov $\mathcal{CPN}$ = ($\sum$,*P*, *T*, *A*, *N*, *C*, *G*, *E*, *I*), pre ktorú platí:

(1) Množina typov $\sum$ je rozložená nasledovne: $\Sigma = \Sigma_O \cup \Sigma_R$, kde platí:

(a) $\Sigma_O = \bigcup_{i \in I_{\mathcal{N}}} \Sigma_O^i = \{o_1, o_2, ..., o_n\}$, kde o_i je farba *i*-teho typu procesu,

(b) $\Sigma_R = \{r_1, r_2, ..., r_k\} \neq \emptyset, k > 0$, kde r_j je farba *j*-teho typu prostriedkov v systéme.

(2) Množina miest *P* je rozložená nasledovne: $P = P_0 \cup P_S \cup P_R$, kde platí:

(a) $P_0 = P_0^i = \{p_0\}, i \in I_{\mathcal{N}}$, kde p_0 je miesto pre nečinné fázy všetkých prítomných procesov,

(b) $P_S = \bigcup_{i \in I_{\mathcal{N}}} P_S^i$, a pre každé $i, j \in I_{\mathcal{N}}, i \neq j, P_S^i \cap P_S^j = \emptyset$

(c) $P_R = P_R^i = \{p_R\}, i \in I_{\mathcal{N}}$, kde p_R je miesto pre prostriedky všetkých typov z množiny Σ_R,

(d) $P_S \cap P_R = P_S \cap P_0 = P_R \cap P_0 = \emptyset$.

(3) Množina hrán $A = \bigcup_{i \in I_{\mathcal{N}}} A_{0S}^i \cup A_R^i$.

(4) Množina prechodov $T = \bigcup_{i \in I_{\mathcal{N}}} T^i$ a pre každé $i, j \in I_{\mathcal{N}}, i \neq j, T^i \cap T^j = \emptyset$.

(5) Pre funkciu vrcholov *N* platí:

$$N: \begin{cases} A_{0S}^i \to \left(\left(P_0 \cup P_S^i\right) \times T^i\right) \cup \left(T^i \times \left(P_0 \cup P_S^i\right)\right) \\ A_R^i \to \left(P_R \times T^i\right) \cup \left(T^i \times P_R\right) \end{cases}$$

(6) Pre funkciu farieb *C* platí:

$$C: \begin{cases} \left(P_0 \cup P_S\right) \to \Sigma_O \\ P_R \to \Sigma_R \end{cases}$$

(7) Pre funkciu hranových výrazov *E* platí:

$$E: \begin{cases} A_{0S}^i \to \begin{cases} \forall a \in A_{0S}^i, N(a) = \left(P_0 \times T^i\right) : E(a) = o_i \\ \forall a \in A_{0S}^i, N(a) \neq \left(P_0 \times T^i\right) : \left(E(a) = o_i \vee Type(Var(E(a))) = \Sigma_O\right) \end{cases} \\ A_R^i \to \left\{\forall a \in A_R^i : \left(Type(E(a)) = (\Sigma_R)_{\mathrm{MS}} \wedge Type(Var(E(a))) = \Sigma_R\right)\right\} \end{cases}$$

(8) Pre inicializačnú funkciu *I* platí:

$$I: \begin{cases} P_0 \to \{o_1, o_2, ..., o_n\}_{\mathrm{MS}} \\ P_S \to \emptyset \\ P_R \to \{r_1, r_2, ..., r_k\}_{\mathrm{MS}}, k > 0 \end{cases}$$

■

(1) Množina farieb pre typy procesov $\sum_O$ vzniká zjednotením množín farieb pre typy procesov jednotlivých NP-CPN, kým množina farieb pre typy prostriedkov $\sum_R$ by mala byť u všetkých NP-CPN rovnaká.

(2) – (4) Miesto pre nečinné procesy p_0 vzniká zlúčením miest pre nečinnú fázu všetkých typov procesov, podobne aj miesto pre všetky typy prostriedkov p_R. Množina miest pre fázy procesov P_S, množina hrán *A* a množina prechodov *T* sú zjednotením príslušných množín jednotlivých NP-CPN.

(5) – (8) Funkcie sú definované analogicky ako pri NP-CPN. Za zmienku stojí podmienka v bode (7), že hrany vychádzajúce z miesta p_0 do prechodov z T^i musia byť striktne označené farbou príslušného typu procesu o_i, aby sa jasne vymedzilo, ktorou podsieťou je *i*-ty typ procesu popísaný.

Táto definícia požaduje striktnú disjunktívnosť množín miest pre fázy, množín prechodov a hrán medzi jednotlivými NP-CPN pri zlučovaní do SNP-CPN. Zlučuje množinu NP-CPN v súlade s prvými dvoma bodmi opisu uvedeného v úvode tejto kapitoly. Na to, aby vznikla SNP-CPN aj po vykonaní tretieho bodu, treba ešte definovanú SNP-CPN ďalej racionalizovať podľa vytýčených pravidiel. Výsledná sieť sa preto môže od definovanej SNP-CPN čiastočne líšiť – byť jednoduchšia.

Systém môže obsahovať aj sekvenčné procesy, ktoré evidentne možno považovať za jednoduchšiu verziu nesekvenčného procesu, a teda pri definícii SNP-CPN možno procesné Petriho siete zahrnúť vo forme NP-CPN a netreba uvažovať zvlášť o integrácii P^2N.

$\mathcal{NP}\text{-}\mathcal{CPN}^i$ popisuje *i*-ty typ procesu. Jeho realizáciu modelujú značky farby o_i, ktoré sa v podsieti pre $\mathcal{NP}\text{-}\mathcal{CPN}^i$ môžu pohybovať a ktoré nazývame *inštancie procesu*.

4.2 Prístupy k riešeniu uviaznutia v dopravnom systéme

Nie všetky z prístupov k vysporiadaniu sa s uviaznutím uvedených v kapitole 2.1 sú použiteľné v riadení dopravného systému. Ignorovanie uviaznutia evidentne neprichádza do úvahy. Aj keby sa stavy uviaznutia vyskytovali s pravdepodobnosťou blízkou nule, znovu naštartovať systém z nulového stavu po nich by bolo zbytočne príliš drahé vysporiadanie sa s uviaznutím. Ostatné tri prístupy stoja za ďalšiu analýzu v tejto časti.

4.2.1 Detekcia a zotavenie z uviaznutia v dopravnom systéme

Detekcia stavu uviaznutia sa môže v dopravnom systéme udiať principiálne rovnakým spôsobom, ako bolo popísané v kapitole 2.1.3. Problematickejším je pri tomto prístupe zotavenie systému z uviaznutia.

Ako je uvedené v spomenutej kapitole, na zotavenie systému treba aspoň jednu z operácií: ukončovanie procesov alebo odnímanie prostriedkov, ktoré sú zablokované v čakacom grafe.

Násilné **ukončovanie procesov** v dopravnom systéme sa dá sotva realizovať, nakoľko sú procesom vykonané zmeny zväčša nevratné. Ak by sa mal proces zrušiť, resp. vrátiť na začiatok svojho vykonávania, mohlo by to stáť dodatočné náklady, ktoré sú zvyčajne v porovnaní s inými metódami zbytočne vysoké.

Odňatie prostriedku od bežiaceho procesu je prijateľnejšia možnosť, hoci na to zvyčajne treba použiť mimoriadny zásah v systéme. Po určitom čase, keď už je uviaznutie odstránené, môže proces po návrate odňatého prostriedku späť pokračovať z určitého bodu ďalej. Vybrať možno ten prostriedok, ktorého odobratie spôsobí najmenšiu stratu. Napr. odňatie obslužného personálu od vykonávanej operácie je

ďaleko menej nákladné, ako zastavenie vozidla alebo iného technického zariadenia a jeho presun k inému procesu. Na druhej strane existujú prostriedky ako dopravná cesta, na ktorej zákazka stojí, ktorých reálne odňatie si možno predstaviť iba veľmi ťažko.

Z úvahy je evidentné, že zotavenie systému prostredníctvom odňatia prostriedku je v dopravnom systéme možné iba čiastočne, pričom cena závisí od kategórie prostriedku.

4.2.2 Prevencia pred uviaznutím v dopravnom systéme

Ako uvádza kapitola 2.1.4, na prevenciu pred stavom uviaznutia treba porušiť aspoň jednu zo štyroch Coffmanových podmienok.

Za normálnych okolností sa prostriedky prideľujú **výlučne**, to znamená, že ich nemožno zdieľať medzi viacerými procesmi. Môžu sa vyskytnúť zriedkavé výnimky, keď možno prostriedok zdieľať (napr. dva vlaky stojace na jednej koľaji), ale ide buď o dobre naplánované a zorganizované alebo o mimoriadne situácie. Túto podmienku však nemožno porušiť ľubovoľne a za hocijakých okolností.

Porušenie druhej podmienky **vlastniť a žiadať** vyžiadaním všetkých prostriedkov na začiatku technologickej obsluhy vedie zväčša k rapídne zníženej efektivite využívania prostriedkov, keď niektoré z nich môžu byť pridelené procesu príliš dlho vzhľadom na čas ich využitia. Druhý prístup v odnímaní prostriedkov môže byť prípustný pre niektoré kategórie prostriedkov (obslužný personál), ale pre iné veľmi nákladný (napr. posunovací rušeň) alebo nesplniteľný (napr. dopravná cesta). Takže porušenie tejto podmienky je možné iba čiastočne.

V danej situácii by sa mohol prijať kompromis: snažiť sa minimalizovať počet situácií, keď proces "vlastní a žiada" a povoliť výnimku na dopravnú cestu, kde zákazka práve stojí, teda: cesta by nemusela byť uvoľnená pred pridelením ďalších prostriedkov. Tento kompromis nezaručí neplatnosť podmienky "vlastniť a žiadať", a teda tým nie je vylúčená ani možnosť výskytu uviaznutia. Avšak môže mať vplyv na zníženie pravdepodobnosti jeho výskytu.

Odstrániť tretiu Coffmanovu podmienku, ktorou je **zákaz preempcie**, znamená zaviesť možnosť odobrať bežiacemu procesu prostriedky počas jeho behu. Tu možno rozlíšiť dve situácie podľa toho, či je prostriedok práve použitý priamo v nejakej činnosti alebo nie.

V prvom prípade to znamená odobrať technické zariadenie alebo obslužný personál v čase ich činnosti pri spracovaní zákazky alebo odobrať dopravnú cestu, na ktorej zákazka stojí alebo po ktorej sa práve pohybuje. Toto je zrejme nemožné bez porušenia základných pravidiel reálnej prevádzky. Snáď len v prípade, keď činnosť je prerušiteľná, a teda je možné preradiť prostriedok k druhej požiadavke, odkiaľ sa po skončení vráti späť a dokončí svoju prerušenú činnosť.

Ak prostriedok nie je práve použitý, iba je pridelený, tak ide o zariadenie alebo pracovníka čakajúceho pri zákazke buď na obsluhu alebo na uvoľnenie potrebnej

dopravnej cesty pre obsluhovanú zákazku, ktorú ešte táto neobsadila, ale sa k nej blíži. V tomto prípade je možné pridelený prostriedok odobrať – samozrejme v prípade, ak je to možné z prevádzkového a bezpečnostného hľadiska.

Možno si všimnúť, že okolnosti výskytu druhého prípadu sa viažu do istej miery k predchádzajúcej Coffmanovej podmienke, vlastniť a žiadať. Prostriedok môže čakať na zaradenie do procesu pravdepodobne z dvoch dôvodov:

1) Prostriedok bol priradený procesu skôr, než je potrebný a je len otázkou času, kedy sa začne používať.
2) Pri platnosti spomenutej podmienky daný prostriedok čaká na iný prostriedok, ktorý proces potrebuje a ešte ho nedostal - to môže byť takisto otázka času, ale aj nebezpečenstvo uviaznutia.

Pokiaľ zaručíme neplatnosť podmienky „vlastniť a žiadať", tak všetky prípady budú mať výlučne prvý menovaný dôvod, a vtedy netreba prostriedok odňať, uviaznutie nehrozí. Pokiaľ však tá podmienka môže platiť čo len pri jednej výnimke, prípad z druhého dôvodu sa môže vyskytnúť, a aj s hrozbou uviaznutia.

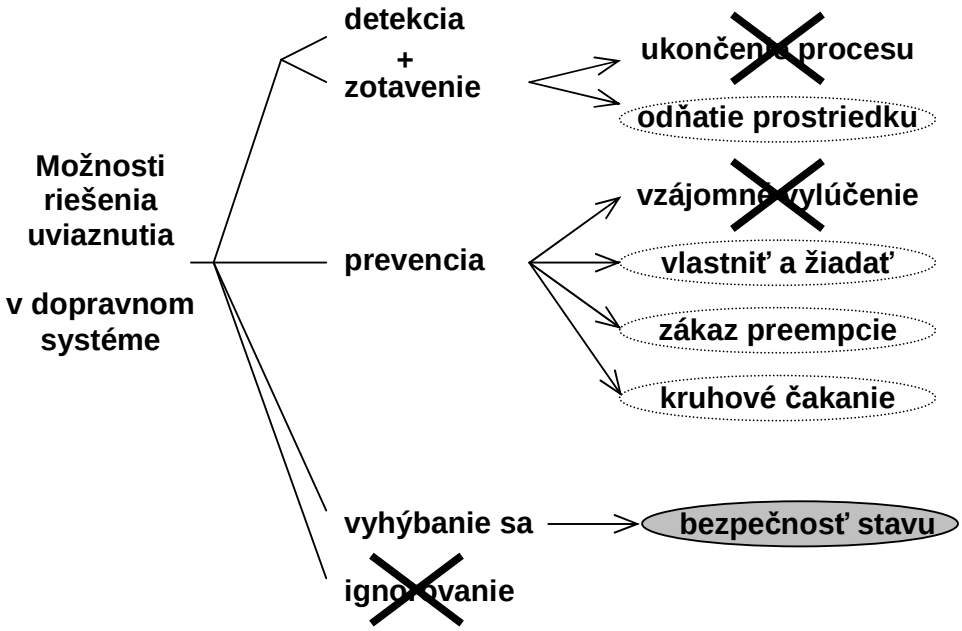
Z uvedeného vyplýva, že odstránenie podmienky neodoberania prostriedkov nie je úplne možné pri všetkých typoch prostriedkov. Tam, kde je to možné a prostriedok je priamo použitý alebo je len otázkou času, kedy bude použitý, tak ho netreba odoberať, ale len počkať na dokončenie jeho využitia. Ak je prostriedok iba priradený procesu, ktorý čaká na ďalšie prostriedky, vtedy ho možno odňať bez vážnych dôsledkov na technológiu.

Zamedziť platnosti poslednej Coffmanovej podmienky **kruhové čakanie** v dopravnom systéme možno zoradením všetkých typov prostriedkov do určitej hierarchie a ich prideľovaním podľa tohto poradia. Hierarchia môže byť založená na tom, aké drahé prostriedky sú, teda koľko nás bude stáť ich prípadné odobratie. V prípade dopravného systému všeobecne sa takým javí poradie: dopravná cesta, pohyblivé technické zariadenie, personál, pričom najdrahší prostriedok sa bude prideľovať ako posledný. To znamená, že koľaj by sa mala prideľovať až vtedy, keď všetky prostriedky nižšej priority pre danú činnosť sú k dispozícii na pridelenie.

Cena za prostriedky sa môže prejaviť aj z pohľadu množstva voľných prostriedkov. Napríklad ak máme prázdnu stanicu s 20 koľajami, no iba jedného posunovača, bude zrejmé drahý posunovač. Ak máme stanicu plnú a 20 posunovačov, bude to naopak. V každom prípade treba poradie stanoviť veľmi starostlivo na základe podmienok v stanici a postrádateľnosti prostriedkov na určitú dobu.

Porušiť podmienku kruhového čakania možno aj tým, že sa spracováva len jeden proces, resp. len vzájomne disjunktívne procesy naraz. Toto opatrenie však môže byť zbytočne priveľkým zásahom do efektívnosti fungovania systému.

Ako vyplýva z predchádzajúcich odsekov, zaručiť úplnú neplatnosť ktorejkoľvek zo štyroch Coffmanových podmienok v dopravnom systéme nie je možné za ľubovoľných okolností.



Obr. 4.6 Možnosti riešenia uviaznutia v dopravnom systéme.

Tri podmienky je možné obmedziť pre prostriedky dvoch druhov, nie však pre dopravnú cestu, ktorú práve obsluhovaná entita zaberá. Znamená to, že prostredníctvom uvedených pravidiel sa nemožno zaručene vyhnúť uviaznutiu v skúmanom dopravnom systéme. Je však možné jeho pravdepodobnosť za cenu zníženia efektivity modelu zmenšiť.

Niektoré obmedzenia podmienok si vyžadujú znalosť nielen štruktúry systému, ale aj jeho aktuálneho stavu. V takom prípade však už nejde o prevenciu pred uviaznutím, ale o ďalšiu kategóriu riešenia stavu uviaznutia: vyhýbanie sa uviaznutiu.

4.2.3 Vyhýbanie sa uviaznutiu v dopravnom systéme

Ako som už naznačil v predchádzajúcej kapitole, už samotná snaha o porušenie Coffmanových podmienok pri prevencii pred uviaznutím môže dizajnéra riadiaceho podsystému často doviesť k potrebe použiť informácie o stave systému. A tie sa využívajú práve v tomto prístupe, ktorého obmedzenia aplikácie závisia od konkrétnych algoritmov.

Nemožno teda vo všeobecnej rovine určiť, nakoľko je stratégia vyhýbania sa uviaznutiu použiteľná v dopravnom systéme. A priori neexistujú nijaké prekážky použitia v dopravnom systéme. Závisí to iba od implementácie algoritmu a potrebných údajov pre neho.

4.2.4 Zhrnutie k použitiu prístupov

Z rozboru troch možných prístupov riešenia stavu uviaznutia sa najsľubnejšie javí vyhýbanie sa uviaznutiu. Zvyšné prístupy sa nedajú alebo by sa dali použiť iba čiastočne – neposkytujú úplné riešenie, resp. výraznejšie znižujú efektívnosť práce systému. Sumarizácia použitia prístupov k vysporiadaniu sa s uviaznutím v dopravnom systéme je prehľadne znázornená na Obr. 4.6.

V ďalšej práci sa teda zameriam na vyhýbanie sa uviaznutiu pomocou informácií o štruktúre systému i o jeho aktuálnom stave.

4.3 Možnosti riešenia uviaznutia

Z tab. 2.5 na konci kapitoly 2.3 vidno, že väčšina metód je vytvorená pre sekvenčné RAS, len časť z nich sa dá použiť pre nesekvenčný RAS. Ich použiteľnosť je teda pre uvažovaný nesekvenčný MT-NO-RAS (odvodený v kap. 4.1.1.3) rôzna. Principiálne existujú tri možné cesty pre skúmanie:

1) Upraviť metódy pre sekvenčný RAS aj na použitie pre nesekvenčný RAS.
2) Previesť nesekvenčný RAS na sekvenčný a použiť metódy pre sekvenčný RAS.
3) Použiť metódy určené pre nesekvenčný RAS.

Perspektívy ciest stručne naznačím v nasledujúcich podkapitolách.

4.3.1 Úprava metód pre sekvenčný RAS

Z metód sa na použitie aj pre nesekvenčné RAS najperspektívnejšie javí algoritmus bankára. Ten vo svojom princípe nezávisí od štruktúry preverovaných procesov a je odskúšaný pre všetky kategórie sekvenčného RAS.

Možnosti ďalších metód vyzerajú byť obmedzené tým, že sú viazané na štruktúru systému príslušnej kategórie RAS. Odvodenie verzií pre nesekvenčný RAS zložitejšieho charakteru si vyžaduje rozsiahlejší výskum.

4.3.2 Prevod nesekvenčného na sekvenčný proces

Pri prevode nesekvenčného na sekvenčný systém ide o prevedenie všetkých nesekvenčných procesov v systéme na sekvenčné, t. j. o odstránenie paralelného spracovania v nesekvenčných procesoch. Terminológiou Petriho siete sa jedná o transformáciu všetkých podsietí NP^2N, ktoré sú dané množinami miest stavov procesu P_S a nečinného stavu P_0 zo značeného grafu (ak proces nemá pružné smerovanie) a z PN s voľným výberom (pri výbere z variantov spracovania) na stavový stroj.

Ako prvá možnosť sa intuitívne naskytá tento postup ($\mathcal{N}$ je pôvodná NP^2N s počiatočným značením pre jeden nečinný proces, $\mathcal{N}_{S0}$ je podsieť NP^2N daná $(P_S \cup P_0)\times T$):

1) Nájdenie množiny všetkých možných vykonaní procesu (postupností vykonania prechodov v $\mathcal{N}$).
2) Vytvorenie cyklu $\mathcal{N}_{S0}^{\,i}$ pozostávajúceho z prechodov $\mathcal{N}_{S0}$ a miest, ktoré zodpovedajú stavom procesu (značeniam $\mathcal{N}$) pre každú postupnosť vykonania prechodov.
3) Zoskupenie vytvorených cyklov $\mathcal{N}_{S0}^{\,i}$ do novej P^2N $\mathcal{N}_{S0}^{\,*}$ zlúčením miest reprezentujúcich počiatočné značenie siete $\mathcal{N}_{S0}$.
4) Nahradenie pôvodnej podsiete $\mathcal{N}_{S0}$ v $\mathcal{N}$ novou podsieťou $\mathcal{N}_{S0}^{\,*}$, čím vzniká $\mathcal{N}^*$.

Tým sa namiesto pôvodnej NP^2N získa P^2N, stavový stroj reprezentujúci sekvenčný proces s pružným smerovaním. Úskalím tohto postupu je fakt, že množina všetkých možných vykonaní procesu môže byť i pre malé nesekvenčné procesy značne veľká.

Tento problém však možno významne zmenšiť, ak sa pri zoskupení cyklov $\mathcal{N}_{S0}^{\,i}$ zlúčia nielen miesta pre počiatočné značenie, ale všetky miesta s rovnakým značením v pôvodnej $\mathcal{N}_{S0}$. Po tejto operácii sa môžu nájsť v sieti dvojice miest, ktoré budú spojené viac ako jednou trojicou prvkov hrana-prechod-hrana – tie treba odstrániť tak, aby tam zostala iba jedna trojica hrana-prechod-hrana. Tým sa dosiahne procesná Petriho sieť zodpovedajúca stavovému priestoru pôvodnej PN nesekvenčného procesu, ktorú odvádzajú aj autori v publikácii (Ezpeleta-Recalde, 2004), kde na nahradenie používajú stavový stroj vytvorený z grafu dosiahnuteľnosti.

Výsledný sekvenčný proces s pružným smerovaním obsahuje niekoľkonásobne viac miest, prechodov a hrán ako pôvodný nesekvenčný proces. Pomer závisí od počtu paralelných vetiev a ich dĺžky v pôvodnom nesekvenčnom procese. Ako príklad možno použiť nesekvenčný proces popísaný sieťovým grafom na Obr. 5.2, ktorý má 20 vrcholov a 26 hrán. Jemu zodpovedajúca podsieť $\mathcal{N}_{S0}$ obsahuje 20 prechodov a 27 miest (1 miesto zodpovedá nečinnému stavu procesu, ktorý v sieťovom grafe nie je zaznamenaný). Pre tento proces vyšiel graf dosiahnuteľnosti o veľkosti 163 vrcholov a 375 hrán, čo zodpovedá 163 miestam a 375 prechodom Petriho siete pre sekvenčný proces.

Po nahradení všetkých nesekvenčných procesov sekvenčnými s pružným smerovaním možno použiť metódy pre sekvenčné RAS podľa príslušnej kategórie.

4.3.3 Algoritmy pre systém s nesekvenčnými procesmi

Podľa prehľadu v tab. 2.5 pre systém s nesekvenčnými procesmi bol odvodený jediný prístup – založený na analýze stavového priestoru a vytváraní regiónov v ňom. Má podstatnú nevýhodu vo veľkosti stavového priestoru systému, ktorý pre zložitejšie

systémy je značný. Napríklad demonštračný model jednoduchého dopravného systému vybudovaný v rámci tejto práce obsahuje pre 3 inštancie procesov (tab. 5.2) takmer 180 tisíc stavov a vyše 600 tisíc prechodov medzi nimi.

Podotýkam, že ide o inú situáciu, ako predchádzajúcej kapitole, kde som stavový priestor počítal pre jeden proces a jednu inštanciu procesu. V tomto prípade ide o stavový priestor systému všetkých procesov a všetkých ich uvažovaných inštancií.

4.3.4 Výber algoritmu a východiská pre riešenie

Ako najschodnejšia z naznačených ciest sa javí tretia, t. j. použitie algoritmu pre systém s nesekvenčnými procesmi, a v rámci nej algoritmus bankára (BA). Ďalšou motiváciou na jeho výber bolo porovnanie účinnosti tohto algoritmu a algoritmu založeného na odstraňovaní sifónov v Petriho sieti v publikácii (Lawley et al., 1998), kde algoritmus bankára povolil viac bezpečných stavov. Východiská pre jeho použitie opíšem v tejto kapitole a jeho úpravám a aplikácii sa venujem vo zvyšku tejto práce.

Ako je uvedené v kap. 2.3.1.4, v dostupnej literatúre som našiel riešenia BA iba pre sekvenčný RAS – niektoré pre najjednoduchšiu kategóriu SU-TO-RAS a iné pre najzložitejšiu MT-NO-RAS.

Na to, aby sa dal algoritmus použiť pre vymedzený nesekvenčný MT-NO-RAS so špecifikami prideľovania prostriedkov (kap. 4.1.1), treba existujúce verzie upraviť tak, aby dokázali aj

- paralelné spracovanie procesu pri súčasnom dodržaní pružného smerovania procesu a možného opakovaného použitia prostriedkov v rámci jedného procesu,
- použitie profesií pri výbere prostriedkov.

Za čiastkové ciele som si stanovil úpravu a preverenie všetkých verzií algoritmu uvedených v publikácii (Lawley et al., 1998) s využitím vylepšení zavedených v publikácii (Tricas, 2003), ktoré bližšie opisujem v kapitolách 2.3.1.1, 2.3.1.2 a prílohe C.1.

Pri výbere algoritmu som takisto zvažoval použitie úpravy BA pre systém AGV (kap. 2.3.1.3 a príloha C.2), pričom jeho použiteľnosť pre všeobecný dopravný systém ako MT-NO-RAS by si vyžadovala uvoľnenie jedného z predpokladov tak, aby vozidlo mohlo zabrať viac ako jednu zónu (úsek) dopravnej siete naraz.

Nakoľko je úprava BA pre systém AGV postavená výlučne na prostriedkoch dopravnej siete (úsekoch), usudzujem, že jej aplikácia na systém s rôznymi druhmi prostriedkov, včítane pohyblivých, by bola vhodná pri dodržaní jednej z týchto podmienok:

a) Pohyb vozidiel je hlavným modelovaným procesom a nevyskytujú sa ostatné technologické činnosti.

b) Ostatné činnosti sú zoskupené do všeobecných obslužných procesov, ktoré sa modelujú autonómne, kým je vozidlo v uzle siete, pričom tieto procesy v uzloch by používali disjunktívne množiny ostatných prostriedkov.

Tab. 4.1 Zmena označenia algoritmov bankára.

nové	pôvodne	základná charakteristika
A	algoritmus 1	Základný BA – hľadá usporiadané stavy na základe usporiadania všetkých aktívnych procesov v systéme.
B	algoritmus 2	Vylepšenie základného BA – hľadá čiastočne usporiadané stavy. Jeho zložitosť je v najhoršom prípade rovnaká ako u základného, no zvyčajne odpovie za kratší čas.
C	algoritmy 3 a 4	Vylepšenie predch. verzií BA – hľadá V_m-usporiadané stavy. Využíva predchádzajúce algoritmy, vyžaduje navyše aj informácie o spracovaní procesov. Podľa veľkosti parametra m sa môže jeho zložitosť blížiť až k exponenciálnej.

Vzhľadom na to, že dopravný systém vymedzený v tejto práci má iný, všeobecnejší charakter, usúdil som, že úprava algoritmu pre systém AGV nie je preň vhodná a preverenie tohto variantu riešenia som si preto nedal medzi hlavné priority.

4.3.5 Meranie kvality algoritmu riadenia

Hlavným kritériom kvality implementovaného riadiaceho algoritmu je **absencia uviaznutí** v riadenom systéme. Dosiahnutie tohto kritéria je hlavným cieľom práce a rozhoduje o úspešnosti výsledkov.

Keďže algoritmus pracuje na princípe delenia stavov na usporiadané a neusporiadané, pričom povolí iba usporiadané stavy, a podľa kapitoly 2.3.1.1 a Obr. 2.4 vieme, že nie všetky bezpečné stavy sú algoritmom vyhlásené za usporiadané, bude vedľajším kritériom kvality implementovaného algoritmu **počet stavov**, ktoré **prijal za usporiadané**. Nakoľko optimálna stratégia, ktorá pripustí všetky bezpečné stavy je NP-ťažká (kap. 2.3.2.5), pri výpočte v polynomiálnom čase sa treba uspokojiť iba so suboptimálnymi riešeniami. Algoritmy rôznej zložitosti môžu viesť k rôznym výsledkom, preto bude zaujímavé porovnať aj ich zložitosť a výsledky.

4.4 Úpravy algoritmov

Algoritmy, ktoré sú výsledkom tejto práce, rozdelím na 3 hlavné a 4 čiastkové.

Hlavné sú verzie algoritmu bankára. Vychádzal som z publikácie (Lawley et al., 1998), kde uvádzajú štyri algoritmy pre SU-TO-RAS (príloha C.1, komentár v kap. 2.3.1.2). Ide však v podstate o tri verzie algoritmu bankára, ktoré riešia otázku usporiadanosti stavu. Pre potreby svojej práce som ich preznačil ako uvádza tab. 4.1.

Čiastkové algoritmy vychádzajú z návrhov, ktoré v tejto práci uvádzam a ktorých definícia dopĺňa alebo spresňuje hlavné algoritmy.

Identifikátory použité v navrhnutých algoritmoch vychádzajú z programovacieho jazyka Pascal. Použité zoznamy sú necyklické, dvojsmerne lineárne zreťazené a predpokladám u nich a u ich prvkov existenciu operácií podobných ako v publikácii (Cenek et al., 1994, str. 62-65). Tie, ktoré sa vyskytnú v mojom návrhu zhŕňa tab. 4.2.

4.4.1 Zavedenie paralelného spracovania procesu

Kvôli jasnejšiemu vysvetleniu v nasledujúcom texte zavádzam dva pojmy, a to

- **variant** spracovania procesu – jedna z ciest spracovania procesu (sekvenčného alebo nesekvenčného), ktorá bola vybratá pri jeho pružnom smerovaní, je to cesta v Petriho sieti stavového stroja,
- **vetva** spracovania procesu – časť procesu, ktorá sa spracováva súbežne s inou časťou (vetvou) procesu, t. j. cesta v Petriho sieti značeného grafu.

U nesekvenčných procesov s pružným smerovaním sa používajú obidva výrazy súčasne – proces má viacero variantov spracovania a v rámci nich môže byť viac vetiev, prípadne súčasťou vetiev môžu byť varianty spracovania.

4.4.1.1 Princíp

Údajové štruktúry, matice *Pridelene*, *Potrebne* a vektor *Volne*, ktoré používa pôvodná verzia algoritmu, sa nemenia. Dovolím si len premenovať maticu potrebných prostriedkov na *ZostavajucePotrebne*, aby som presnejšie vyjadril, že obsah tejto údajovej štruktúry sa počas spracovania procesu mení. Prostriedky, ktoré už proces nebude opakovane požadovať, sú po skončení ich používania z jeho vektora odčítané, a teda v ňom zostanú iba tie prostriedky, o ktoré bude proces v zostávajúcej časti spracovania ešte žiadať. Tento zdanlivo samozrejmý úkon nie je evidentný z pôvodnej verzie BA a v niektorých systémoch nemusí byť realizovateľný. V dopravnom

Tab. 44.2 Operácie so zoznamom a jeho prvkami.

meno metódy	**stručný opis**
zoznam.Prvy	sprístupní prvý prvok daného zoznamu
prvok.Dalsi	sprístupní nasledovníka daného prvku v zozname
zoznam.Zrus(*prvok*)	zruší daný prvok z daného zoznamu
zoznam.PridajKopiu(*prvok*)	vytvorí kópiu daného prvku a pridá ju do daného zoznamu
zoznam.PocetPrvkov	vráti počet prvkov daného zoznamu
zoznam.JePrazdny	vráti *True*, ak je daný zoznam prázdny, inak *False*

systéme sa očakáva, že priebeh procesu vopred poznáme, a teda poznáme aj stavy, kedy sa hodnoty tohto vektora môžu zmeniť.

Zavedenie zmien hodnôt vektora *ZostavajucePotrebne* počas spracovania procesu je prvou úlohou, ktorú treba vykonať. Tricas v publikácii (2003, str. 124) navrhuje, aby sa hodnoty vektora *ZostavajucePotrebne* priradili jednotlivým stavom procesu na ceste do konca spracovania. Ak je na výber viac variantov spracovania (t. j. spracovanie procesu je pružne smerované), tak tento návrh elegantne rieši problém úpravy vektora po výbere jedného z variantov – tak, aby vektor obsahoval presnú zostávajúcu potrebu prostriedkov do konca daného variantu.

Pri súbežnom spracovaní nesekvenčného procesu vo vetvách to však nefunguje. Kým v sekvenčnom procese je aktuálny stav procesu reprezentovaný značkou v jednom mieste v PN, v nesekvenčnom procese je reprezentovaný značkami vo viacerých miestach. Keďže sa vetvy vykonávajú súbežne, poradie presunu značiek medzi miestami je nedeterministické. Tým pádom nie je možné všeobecne pre každé miesto určiť (v momente, keď sa v ňom nachádza značka procesu), koľko prostriedkov má proces aktuálne pridelených a koľko ich ešte bude potrebovať až do konca spracovania.

Ako vhodné riešenie sa ukázalo zavedenie **vektorov relatívnych zmien**, ktoré zmenia vektor procesu *ZostavajucePotrebne* pri každej zmene stavu príslušným spôsobom. Novozavedený vektor budem nazývať *ZiadanaZmena*. Ak sa pri relatívnej zmene prideľujú prostriedky procesu, vo vektore *ZiadanaZmena* je ich pridelenie zapísané kladnou hodnotou ich počtu. Ak sa uvoľňujú, tak zápornou.

Okrem toho však treba ešte pri uvoľnení prostriedku zohľadniť, či sa bude prostriedok neskôr opätovne prideľovať, a teda, či má byť vektor *ZostavajucePotrebne* o túto potrebu aktualizovaný. Na zakódovanie tejto informácie som sa rozhodol využiť polovicu bitov čísla vo vektore *ZiadanaZmena*. Počet, ktorý je zakódovaný v hornej polovici bitov, sa opätovne zapíše do vektora *ZostavajucePotrebne*. Počet v dolnej časti bitov sa už nezapíše. Opätovný zápis sa koná iba pri uvoľnení prostriedkov. Naopak, pri pridelení sa vektor *ZostavajucePotrebne* musí o prideľovaný počet prostriedkov automaticky znížiť, aby sa dodržala podmienka, že proces už nebude potrebovať viac, než bolo v úvode procesom deklarované.

Je evidentné, že počet bitov musí byť zvolený tak, aby bol dostatočný pre zápis maximálneho počtu prostriedkov jedného typu prideľovaného, resp. uvoľňovaného naraz. Intuitívne sa ponúka rozdelenie bitov na polovicu, no nie je nevyhnutné to vždy dodržať. Vo výpočtoch sa obsah bitov sprístupňuje použitím konštanty, ktorú nazývam *Rozdelovnik* (jej veľkosť zodpovedá počtu bitov v dolnej časti bytu). Horná časť bitov sa dosiahne pri kódovaní vynásobením konštantou *Rozdelovnik* a pri dekódovaní celočíselným delením (používam znak „/“) tým istým číslom. Obsah dolnej časti sa naplní prirodzeným pričítaním a sprístupní zo zvyšku po celočíselnom delení (znak „\“) konštantou *Rozdelovnik*.

Tab. 4.3 Príklady kódovania v čísle vektora *ZiadanaZmena*.

stĺpec	1	2	3	4	5	6	7
číslo vektora *ZiadanaZmena*	**1**	**16**	**17**	**–1**	**–16**	**–17**	**–54**
operácia (pridelenie, uvoľnenie)	p	p	p	u	u	u	u
zmena vektora *Volne*	–1	–1	–2	+1	+1	+2	+9
zmena vektora *Pridelene*	+1	+1	+2	–1	–1	–2	–9
zmena vektora *ZostavajucePotrebne*	–1	–1	–2	0	+1	+1	+3

V tab. 4.3 je sedem príkladov kódovania v čísle vektora *ZiadanaZmena* pre jeden typ prostriedkov o veľkosti 1 byte, ktoré rozdelím na horné a dolné štyri bity, t. j. *Rozdelovnik* je rovný 16. Z príkladu vidieť, že ak sa uvoľňuje viac ako jeden prostriedok, ale iba niektoré z nich bude neskôr opäť treba, možno ich počet rozdeliť do horných a dolných bitov – procesu sa odoberie ich súčet, no pre budúcnosť sa zachová len časť v horných bitoch (ako vidno v stĺpcoch 6 a 7). Taktiež si možno všimnúť, že pri pridelení sa medzi použitím horných a dolných bitov nerozlišuje (stĺpce 1 a 2).

4.4.1.2 Úprava algoritmu aktualizácie údajov

Pri aktualizácii údajových štruktúr algoritmu sa nasledujúci blok príkazov rozširuje o príkazy vetvenia a ďalšie vzťahy, ako uvádzam v čiastkovom algoritme 1.

```
for j = 1 to PocetProfesii do begin
   Volne[j] = Volne[j] – ZiadanaZmena[j];
   Pridelene[i][j] = Pridelene[i][j] + ZiadanaZmena[j];
   ZostavajucePotrebne [i][j] = ZostavajucePotrebne [i][j] – ZiadanaZmena[j];
end;
```

Čiastkový algoritmus 1: Aktualizácia údajových štruktúr
vstup: *ZiadanaZmena*, *Rozdelovnik*, *Volne*, *Pridelene* a *ZostavajucePotrebne*
výstup: *Volne*, *Pridelene* a *ZostavajucePotrebne*

```
begin
   for j := 1 to PocetProfesii do
      if ZiadanaZmena[j] <> 0 then begin
         if ZiadanaZmena[j] < 0
         then HodnotaZmeny := – (abs(ZiadanaZmena[j]) / Rozdelovnik +
         abs(ZiadanaZmena[j]) \ Rozdelovnik)
```

```
            else HodnotaZmeny := ZiadanaZmena[j] / Rozdelovnik + ZiadanaZmena[j] \
            Rozdelovnik;
            Volne[j] := Volne[j] – HodnotaZmeny;
            Pridelene[i][j] := Pridelene[i][j] + HodnotaZmeny;

            if ZiadanaZmena[j] < 0
            then HodnotaZmeny := – abs(ZiadanaZmena[j]) \ Rozdelovnik
            else HodnotaZmeny := ZiadanaZmena[j] \ Rozdelovnik;
            ZostavajucePotrebne[i][j] := ZostavajucePotrebne[i][j] – HodnotaZmeny
        end;
    return (Volne, Pridelene, ZostavajucePotrebne)
end;
```

■

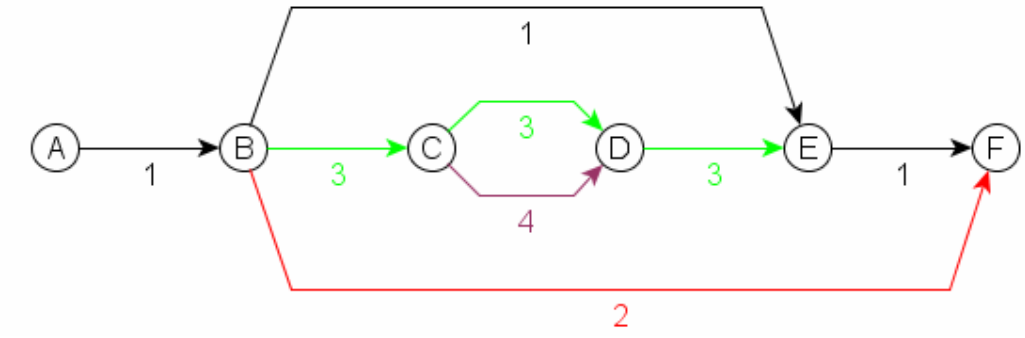
4.4.1.3 Použitie variantov spracovania

Ďalej zostáva vyriešiť obsah vektora *ZostavajucePotrebne* pre jednotlivé varianty spracovania. Tu možno takisto použiť relatívne zmeny vektorov *Pridelene* a *Volne*, ktoré sa pri tvorbe modelu vytvoria nasledovne:

1) Zoradiť všetky varianty spracovania jedného procesu do poradia a prvý z variantov určiť ako **primárny** – vektor potrebných prostriedkov pre tento variant bude štartovací pre každý proces.

2) V každom bode, kde dochádza k zmene variantu na iný, treba definovať rozdielový vektor, o ktorý sa obidva varianty líšia – definujme ho ako výsledok funkcie *Rozdiel*(*i*, *j*), ktorá vzniká odčítaním vektora *ZostavajucePotrebne*[*j*] od vektora *ZostavajucePotrebne*[*i*] v danom bode.

Na Obr. 4.7 máme príklad spracovania procesu s variantnými cestami zoradenými od 1 po 4. Podľa naznačeného postupu bude obsahovať na začiatku spracovania



Obr. 4.7 Príklad spracovania procesu v 4 rôznych variantoch, z ktorých sa vyberá jeden.

počiatočné hodnoty maximálnych potrebných prostriedkov pre variant 1. Ak sa v bode B zmení smerovanie na variant 2, vektor *ZostavajucePotrebne* sa sčíta s vektorom *Rozdiel*(1, 2). Podobne treba zaviesť rozdielové vektory *Rozdiel*(1, 3) v rovnakom bode a *Rozdiel*(3, 4) v bode C. V bodoch D a E nie sú potrebné nijaké zmeny, nakoľko zostávajúce potrebné prostriedky vo zvyšku spracovania sú pre zlučujúce sa varianty rovnaké a musia byť zakódované už v nastavení primárneho vektora na začiatku. Rozdielové vektory medzi variantmi nemajú na tieto prostriedky vplyv.

4.4.1.4 Počiatočné nastavenie údajových štruktúr

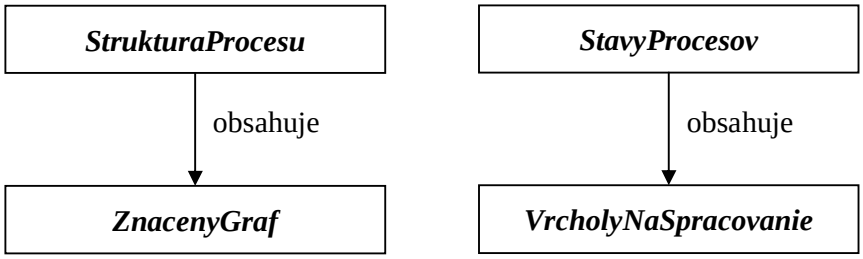
Oproti základnej verzii BA sa počiatočné nastavenie údajových štruktúr algoritmu nemení. Je však vhodné upozorniť na pár detailov.

Vektor *Volne* obsahuje počet voľných prostriedkov pri štarte systému. Matica *Pridelene* pri štarte systému z nulového stavu (všetky procesy sú nečinné) tomu zodpovedá, zvyčajne teda obsahuje samé nuly. Ak procesy vstupujú s priradenými prostriedkami, toto treba zohľadniť. Napríklad v demonštračnom modeli (ako opíšem neskôr) vstupujú vlaky do stanice s vlakovým rušňom v čele, ktorý sa tiež považuje za sledovaný prostriedok. Preto vektory procesov obsahujú na príslušnom mieste jednotku.

Nastavenie vektora *ZostavajucePotrebne*[*i*] pre *i*-ty proces závisí od opisu spracovania procesu, ako som už opísal. Pre každý typ prostriedkov obsahuje maximálny počet, ktorý môže proces vlastniť naraz v ľubovoľnom okamihu spracovania vo variante, ktorý bol vybratý za primárny. Ak sa niektorý prostriedok používa opakovane a v počte jednotiek väčšom ako jedna, treba dôsledne rozlíšiť, ktoré jednotky sa budú používať určite v disjunktívnych časových obdobiach a kde môže dôjsť k súbežnému použitiu, zvlášť pri zložitejších opisoch spracovania zložených z kombinácie viacerých vetiev a variantov. V prípade nesprávneho určenia tejto hodnoty môže dôjsť k nesprávnym výsledkom algoritmu, ba až k vzniku stavu uviaznutia – to je v prípade, keď sa nastaví nižšia hodnota ako je potrebné maximum. Ak sa nastaví vyššia hodnota, dochádza k zníženiu efektivity fungovania systému, keď algoritmus niektoré bezpečné stavy označí zbytočne za neusporiadané. Skúškou správnosti nastavených hodnôt vektora *ZostavajucePotrebne*[*i*] pre *i*-ty proces je jeho obsah na konci spracovania procesu – po skončení každého z definovaných variantov obsahuje samé nuly.

4.4.2 Vykonanie procesu podľa značeného grafu

Algoritmus C potrebuje navyše aj údaje o štruktúre spracovania procesu, nakoľko sa pri svojom vykonaní pokúša posúvať procesy. Keďže proces môže byť všeobecne nesekvenčný, treba údajovú štruktúru pripraviť aj na zachytenie štruktúry značeného grafu a k nej aj príslušný algoritmus spracovania. V nasledujúcich podkapitolách opisujem svoj návrh.



Obr. 4.8 Vzťahy medzi definovanými údajovými štruktúrami.

4.4.2.1 Údajové štruktúry pre vykonanie procesu

Značený graf je zapísaný v usporiadanom poli prvkov ***ZnacenyGraf***, kde jeden prvok *ZnacenyGraf*[*i*] obsahuje údaje o jednom vrchole grafu V_i, a to konkrétne identifikátor vrcholu V_i, počet hrán značeného grafu, ktoré do vrcholu V_i vchádzajú (t. j. počet predchodcov), zoznam smerníkov na nasledovníkov, do ktorých z vrcholu V_i hrany vychádzajú a smerník na vektor *ZiadanaZmena* pre vrchol V_i.

Ak existuje viac variantov spracovania procesu, každý z nich je reprezentovaný svojou vlastnou štruktúrou značeného grafu. A tak vzniká nadradené pole ***StrukturaProcesu*** s počtom prvkov rovným počtu variantov procesu. Uvedená štruktúra údajov je pre systém konštantná – vytvorí sa na začiatku pri zostavení systému a počas jeho činnosti sa nemení.

Ďalej potrebujeme premennú údajovú štruktúru, ktorá registruje **aktuálny stav všetkých (inštancií) procesov**, ktoré pracujú podľa svojej štruktúry procesu. Tá v čase mení svoj obsah podľa posunu inštancií procesov v systéme. Nazvime ju ***StavyProcesov***. Je to zoznam prvkov, v ktorom jeden prvok zastupuje stav jednej inštancie procesu v systéme. Obsahuje identifikátor inštancie procesu, zoznam

Tab. 4.1 Údajové štruktúry pre modelovanie vykonávania procesov.

prvok poľa *StrukturaProcesu*	**prvok poľa *ZnacenyGraf***
VariantID (index) *ZnacenyGraf*	*VrcholID* (index) *PocetPredchodcov* *Nasledovnici* *ZiadanaZmena*

prvok zoznamu *StavyProcesov*	**prvok zoznamu *VrcholyNaSpracovanie***
InstanciaProcesuID *VrcholyNaSpracovanie* *VariantID*	*VrcholID* *PocetSpracovanychPredch*

vrcholov značeného grafu, ktoré sa majú v najbližšej budúcnosti spracovať a identifikátor variantu procesu, ktorý jeho značený graf opisuje.

Lineárny zoznam vrcholov ***VrcholyNaSpracovanie*** obsahuje tie vrcholy značeného grafu (pre každú vetvu, v ktorej sa spracovanie inštancie procesu nachádza), ktoré sú navrhnuté na spracovanie, no ešte nemusia mať na to splnené všetky podmienky. Každý prvok zoznamu *VrcholyNaSpracovanie* obsahuje identifikátor vrchola značeného grafu a počet predchodcov tohto vrchola, ktorí boli dosiaľ v systéme spracovaní.

4.4.2.2 Algoritmus pre vykonanie procesu

Proces napreduje postupným spracovávaním vrcholov značeného grafu. Posun procesu z aktuálneho stavu do ďalšieho sa realizuje spracovaním jedného vrchola, ktorý je už na to pripravený. To zachytáva nasledujúci čiastkový algoritmus 2.

Pre svoju činnosť potrebuje zoznam vrcholov na spracovanie daného procesu a príslušný značený (sieťový) graf, podľa ktorého sa proces vykonáva. Výsledkom algoritmu je aktualizovaný zoznam vrcholov na spracovanie po vykonaní posunu a vektor navrhnutej zmeny v obsadení prostriedkov pri vykonanom kroku, ktorý sa získa z údajovej štruktúry značeného grafu.

Čiastkový algoritmus 2: Posun daného procesu do ďalšieho stavu
input: *VrcholyNaSpracovanie*, *ZnacenyGraf* (pre daný proces)
output: aktualizované *VrcholyNaSpracovanie*, *NavrhnutaZmena*

```
begin
   // Nájdi v zozname VrcholyNaSpracovanie prvý prvok, ktorý možno spracovať. //
   AktualnyVrchol := VrcholyNaSpracovanie.Prvy;
   while AktualnyVrchol.PocetSpracovanychPredch <
   ZnacenyGraf[AktualnyVrchol].PocetPredchodcov do
      AktualnyVrchol := AktualnyVrchol.Dalsi;
   // Nájdený vrchol sprístupní zoznam svojich nasledovníkov v značenom grafe. //
   AktualnyNasledovnik :=
   ZnacenyGraf[AktualnyVrchol.VrcholID].Nasledovnici.Prvy;
        // Podľa vybratého vrchola sa nastaví aj zmenový vektor na výstup. //
   NavrhnutaZmena := ZnacenyGraf[AktualnyVrchol.VrcholID].ZiadanaZmena;
        // Nájdený vrchol sa zruší. //
   VrcholyNaSpracovanie.Zrus(AktualnyVrchol);

        // Prehľadaj predpísaných nasledovníkov akt. vrchola v značenom grafe. //
   while AktualnyNasledovnik <> NIL do begin
        // Zisti, či sa v zozname VrcholyNaSpracovanie akt. nasledovník nachádza. //
      PomocnyVrchol := VrcholyNaSpracovanie.Prvy;
```

```
        while (PomocnyVrchol <> NIL) and
        (PomocnyVrchol.VrcholID <> AktualnyNasledovnik.VrcholID) do
        PomocnyVrchol := PomocnyVrchol.Dalsi;
                // Ak áno, tak zvýš počet jeho spracovaných predchodcov. //
        if PomocnyVrchol.VrcholID = AktualnyNasledovnik.VrcholID
        then PomocnyVrchol.PocetSpracovanychPredch :=
        PomocnyVrchol.PocetSpracovanychPredch + 1
                // Ak nie, zaraď ho tam. //
        else VrcholyNaSpracovanie.PridajKopiu(AktualnyNasledovnik);
                // Ďalší nasledovník aktuálneho vrchola v značenom grafe. //
        AktualnyNasledovnik := AktualnyNasledovnik.Dalsi
    end;
    return (VrcholyNaSpracovanie, NavrhnutaZmena)
end
```

■

Napríklad pre proces podľa značeného grafu na Obr. 5.2, v situácii, keď sa už vykonali vrcholy 1, 2 a 3, bude zoznam *VrcholyNaSpracovanie* na vstupe čiastkového algoritmu 2 obsahovať hodnoty: ((4,1), (7,1), (11,1), (5,1)).

Algoritmus vyberie na vykonanie posunu hneď prvý vrchol č. 4, keďže viac ako jedného predchodcu nemá. Po skončení algoritmu obsahuje zoznam *VrcholyNaSpracovanie* hodnoty: ((7,2), (11,1), (5,2)).

4.4.3 Výber prostriedkov podľa profesií

Zavedenie profesií pre výber prostriedkov ovplyvňuje údajové štruktúry a operácie týkajúce sa prostriedkov používaných v systéme. Ide nielen o úpravy v algoritme bankára, ale aj o ďalšie operácie súvisiace s údajmi pre algoritmus. V nasledujúcich podkapitolách k tomu uvádzam podrobnosti.

Pre ľahšie vysvetlenie v ďalšom texte upresním 2 používané pojmy:

- **profesia** prostriedku – typ prostriedku, podľa ktorého je žiadaný,
- **množina** prostriedku – typ prostriedku, kam v systéme patrí podľa jedinečnej kombinácie profesií, ktorých žiadosti môže splniť – každý prostriedok v systéme sa nachádza práve v jednej množine, t. j. množiny sú disjunktívne.

Pred uvažovaním o kombináciách profesií u prostriedkov zodpovedala v systéme jedna profesia jednej množine, t. j. vzťah bol 1:1. V momente zavedenia prvého prostriedku, ktorý sa môže priraďovať podľa viac než jednej profesie, sa ruší vzťah 1:1 a aspoň jedna profesia môže mať vzťah k viac ako jednej množine alebo aspoň jedna množina vzťah k viac ako jednej profesii. Všeobecne vzniká vzťah m:n.

V príklade v kapitole 4.1.1.4 je počet profesií štyri:
I) Koľaje prístupné zo smeru A.
II) Koľaje prístupné zo smeru B.
III) Koľaje určené pre presun pahorkového rušňa.
IV) Koľaje pre obsluhu vlečky.

Počet množín je však päť:
1) Koľaje č. 1, 2 a 3 – profesia I.
2) Koľaj č. 4 – profesie I a III.
3) Koľaj č. 5 – profesie I, II a III.
4) Koľaj č. 6 – profesie I a II.
5) Koľaje č. 7 a 8 – profesie I, II a IV.

4.4.3.1 Zmeny v údajových štruktúrach

Dosiaľ používaných údajových štruktúr pre prostriedky v BA sa zmeny dotýkajú nasledovne:
a) Vektory *ZostavajucePotrebne*, *ZiadanaZmena*, ako aj rozdielové vektory medzi variantmi zaznamenávajú profesie prostriedkov, teda majú dĺžku rovnú počtu profesií.
b) Vektory *Volne* a *Pridelene* majú prostriedky rozdelené podľa množín a ich dĺžka sa rovná počtu všetkých disjunktívnych množín prostriedkov.

Okrem toho treba zaviesť dve nové údajové štruktúry – vektor pre skutočnú zmenu a tabuľku množín pre profesie.

Vektor ***SkutocnaZmena*** na rozdiel od vektora *ZiadanaZmena* obsahuje informáciu o skutočnej zmene pri pridelení alebo uvoľnení prostriedkov podľa množín, ktorá je potrebná pri aktualizácii vektorov *Volne* a *Pridelene* v modeli.

Predpokladajme, že vo vyššie uvedenom príklade prichádza vlak zo smeru B a žiada si koľaj v prázdnej vchodovej skupine. Vektor *ZiadanaZmena* vtedy obsahuje hodnoty: (0, 1, 0, 0). Riadiaci podsystém mu môže prideliť ľubovoľnú z koľají 5–8, a tak vektor *SkutocnaZmena* môže mať jednu z troch ľubovoľných podôb:
(0, 0, 1, 0, 0) alebo (0, 0, 0, 1, 0) alebo (0, 0, 0, 0, 1).

Vzťahy medzi profesiami a množinami určuje tabuľka množín pre profesie ***MnozinyPreProfesie***. Jedna položka tabuľky obsahuje identifikátor profesie a skupinu identifikátorov množín, kde sa nachádzajú prostriedky danej profesie. Obsah tabuľky sa nemení počas fungovania modelu. Je jednou z jeho charakteristík.

Pre vyššie uvedený príklad by taká tabuľka obsahovala
- pre profesiu I množiny 1, 2, 3, 4 a 5,
- pre profesiu II množiny 3, 4, 5,
- pre profesiu III množinu 5,
- pre profesiu IV množiny 2 a 3.

Alebo zapísané vektorovo (identifikátor profesie je index):
((1, 2, 3, 4, 5), (3, 4, 5), (5), (2, 3)).

4.4.3.2 Zmeny v operáciách

Zmeny sa týkajú dvoch operácií:
1) Overenie, či možno splniť požiadavku na pridelenie prostriedkov.
2) Obnova údajových štruktúr.

V prvom prípade treba nerovnosť *ZostavajucePotrebne*[*i*][*j*] ≤ *Volne*[*j*] nahradiť ďalším čiastkovým algoritmom. Ten preveruje, či danú žiadosť o prostriedky možno pokryť dostupnými voľnými prostriedkami. Žiadosť je vo forme vektora podľa profesií a voľné prostriedky vo forme vektora podľa množín. Na prevod slúži tabuľka *MnozinyPreProfesie*.

Okrem odpovede na základnú otázku, či možno pokryť žiadosť voľnými prostriedkami, vráti algoritmus aj vektor prostriedkov *Pokrytie* (jeho veľkosť zodpovedá počtu množín prostriedkov), ktorý v prípade kladnej odpovede na základnú otázku obsahuje presné hodnoty počtu prostriedkov, ktoré možno z dostupných množín odčítať. Táto informácia sa využíva v špecifickom prípade – vo verzii C algoritmu bankára (kap. 4.4.4).

Čiastkový algoritmus 3: Dá sa žiadosť pokryť voľnými prostriedkami?
vstup: *Ziadost, Volne, MnozinyPreProfesie*
výstup: *Ano* / *Nie, Pokrytie*

```
begin
          // Na začiatku je vektor nulový. //
   for i := 1 to PocetMnozin do Pokrytie[i] := 0;
   for i := 1 to PocetProfesii do
      if Ziadost[i] > 0 then begin
         // Prvá množina, z ktorej možno uspokojiť žiadané prostriedky i-teho typu. //
         AktualnaMnozina := MnozinyPreProfesie[i].Prvy
         while AktualnaMnozina <> NIL do
            if Ziadost[i] <= Volne[AktualnaMnozina]
            // Ak je voľných prostriedkov v aktuálnej množine dostatok... //
            then begin
               // Odober z nich. //
               Volne[AktualnaMnozina] := Volne[AktualnaMnozina] – Ziadost[i];
               Pokrytie[AktualnaMnozina] := Ziadost[i];
               // I-ty typ prostriedkov je pokrytý a cyklus možno skončiť. //
               break
            end
            else begin
               // Zober všetky voľné prostriedky z aktuálnej množiny. //
               Ziadost[i] := Ziadost[i] – Volne[AktualnaMnozina];
               Pokrytie[AktualnaMnozina] := Volne[AktualnaMnozina];
               Volne[AktualnaMnozina] := 0;
```

```
            // Posuň sa na ďalšiu množinu s prostriedkami i-teho typu. //
            AktualnaMnozina := AktualnaMnozina.Dalsi;
        end;
      // Preverili sa všetky prípustné množiny a požiadavka nebola pokrytá. //
      if AktualnaMnozina = NIL then return (Nie, Pokrytie);
    end;
  // Všetky požiadavky boli množinami dostupných prostriedkov pokryté. //
  return (Ano, Pokrytie)
end
```

■

K uvedenému algoritmu podotknem, že pre niektoré špecifické konfigurácie (obsahujúce najmä veľa množín zložených z viac ako jednej profesie) a situácie môže dať odpoveď *Nie*, aj keď by mohla byť *Ano*. Môže sa tak stať preto, lebo spôsob výberu skúma iba jedno poradie pokrývania prostriedkov v žiadosti, pričom pri inom poradí testovania pokrytia by mohol dôjsť k inému výsledku. Dôkladné riešenie tohto problému však považujem za menej prioritné v tejto práci, preto som sa mu viac nevenoval, pričom tento jednoduchý návrh mi na demonštráciu postačuje.

V druhom prípade – pri obnove údajových štruktúr sa vo vzťahoch uvedených v kapitole 4.4.1.2 (najmä čiastkový algoritmus 1) nahradí vektor *ZiadanaZmena* vektorom *SkutocnaZmena*.

Čiastkový algoritmus 4: Aktualizácia údajových štruktúr pri použití profesií

vstup: *ZiadanaZmena*, *SkutocnaZmena*, *Rozdelovnik, Volne, Pridelene* a *ZostavajucePotrebne*
výstup: aktualizované *Volne*, *Pridelene* a *ZostavajucePotrebne*

```
begin
  for j := 1 to PocetProfesii do begin
    if SkutocnaZmena[j] <> 0 then begin
      if SkutocnaZmena[j] < 0
      then HodnotaZmeny := – (abs(SkutocnaZmena[j]) / Rozdelovnik +
      abs(SkutocnaZmena[j]) \ Rozdelovnik)
      else HodnotaZmeny := SkutocnaZmena[j] / Rozdelovnik + SkutocnaZmena[j]
      \ Rozdelovnik;
      Volne[j] := Volne[j] – HodnotaZmeny;
      Pridelene[i][j] := Pridelene[i][j] + HodnotaZmeny
    end;

    if ZiadanaZmena[j] <> 0 then begin
```

```
        if ZiadanaZmena[j] < 0
        then HodnotaZmeny := – abs(ZiadanaZmena[j]) \ Rozdelovnik
        else HodnotaZmeny := ZiadanaZmena[j] \ Rozdelovnik;
        ZostavajucePotrebne[i][j] := ZostavajucePotrebne[i][j] – HodnotaZmeny
      end
   end;
   return (Volne, Pridelene, ZostavajucePotrebne)
end;
```

■

4.4.4 Výsledné verzie algoritmu bankára

Ako som spomenul v úvode kapitoly 4.4, pracujem s troma verziami algoritmu bankára, ktoré vychádzajú z algoritmov publikácie (Lawley et al., 1998). Pôvodné algoritmy som upravil tak, aby používali navrhnuté údajové štruktúry, a spresnil čiastkovými algoritmami odvodenými v predchádzajúcich kapitolách. Niektoré podstatné úpravy vysvetlím po uvedení každého algoritmu.

Vysvetlenie niektorých použitých symbolov:

m	počet typov prostriedkov v systéme
ciastkovy_alg_X	čiastkový algoritmus X opísaný v doterajšom texte
Algoritmus_Y	hlavný algoritmus Y – jeden z tých, čo sú uvedené nižšie
AktualnyProces	inštancia procesu, ktorý sa nachádza v rozpracovanom stave
ZiadajuciProces	inštancia procesu, ktorého žiadosť o prostriedky sa algoritmom preveruje

Algoritmus A: Je daný stav usporiadaný?
vstup: *StavyProcesov*, *Volne*, *Pridelene* a *ZostavajucePotrebne* pre daný stav systému.
výstup: *Ano* / *Nie*

```
begin
   while not StavyProcesov.JePrazdny do begin
                  // Nájdi proces, ktorý možno spracovať. //
      AktualnyProces := StavyProcesov.Prvy
      while AktualnyProces <> NIL and
      ciastkovy_alg_3(ZostavajucePotrebne[AktualnyProces.InstanciaProcesuID],
      Volne, MnozinyPreProfesie) = Nie
      do AktualnyProces := AktualnyProces.Dalsi;
      if AktualnyProces = NIL then return Nie;       // Ak sa nenašiel, odmietni stav. //
                  // Inak spracuj nájdený proces. //
      i := AktualnyProces.InstanciaProcesuID;
```

```
      for j := 1 to m do begin
         Volne[j] = Volne[j] + Pridelene[i][j];
         Pridelene[i][j] = 0;
         ZostavajucePotrebne[i][j] = 0
      end;
      StavyProcesov.Zrus(AktualnyProces)          // Proces je spracovaný. //
   end;
            // Všetky procesy boli spracované, teda prijmi stav. //
   return Ano
end
```

■

Tak ako pôvodná verzia algoritmu, aj algoritmus A pracuje na princípe usporiadania práve bežiacich procesov v systéme tak, aby každý mohol využiť okrem u6 pridelených všetky voľné prostriedky. Teda hľadá sa aspoň jedna postupnosť takýchto procesov. Ak sa to nepodarí, daný stav je označený ako neusporiadaný.

Pomocou čiastkového algoritmu 3 sa spresňuje overovanie, či možno požiadavku zostávajúcich potrebných prostriedkov aktuálneho preverovaného procesu pokryť práve voľnými prostriedkami pri zohľadnení mechanizmu použitia profesií.

Algoritmus B: Je daný stav čiastočne usporiadaný vzhľadom k danej inštancii procesu?

vstup: *ZiadajuciProces*, *StavyProcesov*, *Volne*, *Pridelene* a *ZostavajucePotrebne* pre daný stav systému.
výstup: *Ano* / *Nie*

```
begin
   while not StavyProcesov.JePrazdny do begin
            // Ak možno spracovať daný proces, prijmi stav. //
      if ciastkovy_alg_3(ZostavajucePotrebne[ZiadajuciProces.InstanciaProcesuID],
      Volne, MnozinyPreProfesie) = Ano
      then return Ano
            // Inak nájdi ľubovoľný proces, ktorý možno spracovať. //
      AktualnyProces := StavyProcesov.Prvy
      while AktualnyProces <> NIL and
      ciastkovy_alg_3(ZostavajucePotrebne[AktualnyProces.InstanciaProcesuID],
      Volne, MnozinyPreProfesie) = Nie do
         AktualnyProces := AktualnyProces.Dalsi;
      if AktualnyProces = NIL then return Nie;      // Ak sa nenašiel, odmietni stav. //
            // Inak spracuj nájdený proces. //
      i := AktualnyProces.InstanciaProcesuID;
      for j := 1 to m do begin
         Volne[j] = Volne[j] + Pridelene[i][j];
```

```
        Pridelene[i][j] = 0;
        ZostavajucePotrebne[i][j] = 0
      end;
      StavyProcesov.Zrus(AktualnyProces)          // Proces je spracovaný. //
   end;
end
```

■

Algoritmus B sa snaží oproti algoritmu A zrýchliť výpočet tým, že sa pokúsi čiastočne usporiadať bežiace procesy vzhľadom k danej inštancii procesu, žiadajúcej o prostriedky. Akonáhle sa mu podarí do vytváranej usporiadanej postupnosti vložiť túto inštanciu, tak končí bez ohľadu na usporiadanosť ostatných, dovtedy neusporiadaných procesov.

V tomto algoritme, tak ako v predchádzajúcom, sa pomocou čiastkového algoritmu 3 spresňuje overovanie, či možno požiadavku zostávajúcich potrebných prostriedkov daného žiadajúceho procesu, a neskôr aj ostatných bežiacich procesov v cykle pokryť práve voľnými prostriedkami pri zohľadnení mechanizmu použitia profesií.

Algoritmus C: Je daný stav V0 alebo V1-usporiadaný?

vstup: *ZiadajuciProces*, *StavyProcesov*, *Volne*, *Pridelene* a *ZostavajucePotrebne* pre daný stav systému.
výstup: *Ano* / *Nie*

```
begin
      // Prever, či daný stav je čiastočne usporiadaný vzhľadom k žiadajúcemu
   procesu. //
   if Algoritmus_B (ZiadajuciProces, StavyProcesov, Volne, Pridelene,
   ZostavajucePotrebne) = Ano
   then return Ano;     // Stav je čiastočne usporiadaný vzhľadom k žiadajúcemu
   procesu. //
            // Inak prever, či je daný stav V1-usporiadaný. //
   PreverovanyProces := StavyProcesov.Prvy;
   while PreverovanyProces <> NIL do begin
            // Okopíruj údajové štruktúry. //
      docas0 = PreverovanyProces.VrcholyNaSpracovanie;
      docas1 = Volne;
      docas2 = Pridelene;
      docas3 = ZostavajucePotrebne;
      repeat    // Aktualizuj údajové štruktúry pokusom posunúť aktuálny proces
      dopredu. //
            //  Navrhni krok pre posun preverovaného procesu do ďalšieho stavu
         z pohľadu značeného grafu. //
```

```
        (docas0, NavrhnutaZmena) := ciastkovy_alg_2(docas0,
        StrukturaProcesu[PreverovanyProces.VariantID].ZnacenyGraf);
              //  Over, či je krok realizovateľný z hľadiska voľných prostriedkov. //
         (vysledok, Pokrytie) := ciastkovy_alg_3(NavrhnutaZmena, docas1,
        MnozinyPreProfesie) ;
        if vysledok = Ano then begin     // Proces bol úspešne posunutý dopredu. //
              //  Aktualizácia údajových štruktúr po vykonaní kroku. //
           (docas1, docas2, docas3) := ciastkovy_alg_4(NavrhnutaZmena, Pokrytie,
           Rozdelovnik, docas1, docas2, docas3);
              // Je výsledný stav usporiadaný? //
           if Algoritmus_A (docas0, docas1, docas2, docas3) = Ano
           then return Ano;              // Daný stav je V1-usporiadaný. //
        end;
     until vysledok = Nie;     // Proces je blokovaný a nemôže sa pohnúť. //
              // Skús ďalší proces v poradí. //
     PreverovanyProces := PreverovanyProces.Dalsi;
  end
  return Nie
      // Daný stav nie je ani V0 ani V1-usporiadaný. Nepovoľ navrhovaný krok. //
end
```

■

V algoritme C sú integrované dohromady dva algoritmy z pôvodného zdroja (tam označované ako algoritmus 3 a algoritmus 4). Algoritmus C používa obidva hlavné a aj 3 čiastkové algoritmy navrhnuté doposiaľ.

Hlavný algoritmus B sa na začiatku pokúsi zistiť, či možno stav usporiadať čiastočne vzhľadom k žiadajúcemu procesu. Ak sa to nepodarí, teda stav nie je V_0-usporiadaný, nasleduje overovanie, či je stav V_1-usporiadaný. To sa vykoná opakovaním pokusov o posun niektorého bežiaceho procesu do ďalšieho stavu, až kým sa takto nenájde nejaký usporiadaný stav (overenie algoritmom A). Ak sa nenájde po vyskúšaní všetkých bežiacich procesov, algoritmus končí odpoveďou *Nie*.

Keďže posun procesov do ďalšieho stavu mení premenné o stave systému a pri pokusoch o posun sú tieto zmeny iba dočasné, zaznamenávajú sa v premenných *docas0, docas1*, *docas2* a *docas3*. Nastavujú sa vždy na začiatku pokusov o posun prvého a potom vždy ďalšieho bežiaceho procesu a počas pokusov o posun registrujú aktuálny stav systému po vykonaní krokov.

Čiastkové algoritmy postupne

- nájdu krok u práve preverovaného procesu, ktorý možno vykonať (čiastk. alg. 2),
- overia, či sú naň voľné prostriedky (3) a ak áno,
- vykonajú vybratý krok (4).

Ak nový stav nie je usporiadaný, iterácia sa opakuje, pokiaľ sa preverovaný proces dá posúvať.

4.4.4.1 Parametre algoritmov

Zavedením úprav súvisiacich s použitím profesií a posunom procesu podľa značeného grafu do algoritmov sa vniesli parametre, ktorých nastavenie môže vplývať na účinnosť algoritmov. Najviac je to viditeľné v rámci **algoritmu C**, pri ktorom som identifikoval nasledovné parametre.

(1) Možnosti zaradenia do vytváranej usporiadanej postupnosti a možnosti posunu sa preverujú u všetkých bežiacich inštancií procesov jednotlivo, teda aj u tých, ktoré sú navzájom v rovnakom stave vykonávania. To znižuje efektivitu výpočtu, keďže pre inštancie procesov v rovnakom stave platí rovnaký výsledok, či možno zaradiť alebo nezaradiť do usporiadanej postupnosti, bez potreby preverovania všetkých rovnakých.

(2) Pri preverení pokrytia žiadosti voľnými prostriedkami sa skúša len jeden spôsob pokrytia podľa profesií, t. j. množiny sa z konštantnej údajovej štruktúry *MnozinyPreProfesie* vyberajú iba jedným spôsobom, nie všetkými možnými.

(3) Pri pokuse o posun skúša algoritmus pre danú inštanciu iba jednu možnú cestu značeným grafom; ostatné možné cesty (t. j. stavy, do ktorých sa možno posunúť) sa nepreverujú.

(4) Algoritmus A preverujúci usporiadanosť stavu po vykonaní kroku možno nahradiť algoritmom podobným algoritmu C – označím ho algoritmus C'. Ten by sa líšil od algoritmu C iba v dvoch príkazoch: namiesto algoritmu B by mal zaradený algoritmus A a namiesto algoritmu A sám seba, t. j. algoritmus C'. Prvá úprava je daná podmienkami použitia vlastnosti *čiastočná usporiadanosť stavu vzhľadom k danému procesu* namiesto vlastnosti *usporiadanosť stavu* – tie podmienky by v tomto prípade neboli zaručené. Podrobnosti nájde čitateľ v publikácii (Lawley et al., 1998, str. 16-18). Nahradenie algoritmu A algoritmom C' vedie k hľadaniu V_n-usporiadaných stavov (podrobnosti v rovnakej publikácii na str. 18-25), kde *n* je počet úrovní vnorenia do seba, ktoré algoritmus C' vykoná.

Zo štyroch parametrov algoritmov, ktoré v navrhnutej implementácii vďaka svojej jednoduchosti pôsobia obmedzujúco, platia **prvé dva aj pre algoritmy A a B**.

Intuitívne možno predpokladať, že v prípade (1) sa zložitosť algoritmu môže zvýšiť aj znížiť. Závisí od faktu, v koľkých stavoch systému koľko procesov býva rovnakých (teda od parametrov modelovaného systému) a nakoľko by sa predĺžila réžia spravovania údajových štruktúr zavedením dodatočných informácií o procesoch v rovnakých stavoch.

Uvoľnenie obmedzení algoritmu podľa bodov (2) – (4) umožní nájsť viac usporiadaných stavov systému. Platbou za to je zvýšenie zložitosti algoritmu. Tá sa však prejaví len pri preverovaní niektorých stavov.

Uvedená úvaha môže byť predmetom ďalšieho skúmania správania sa algoritmu C pre systémy s rôznymi parametrami.

4.4.5 Realizácia vyhýbania sa uviaznutiu

V tejto kapitole zhrniem svoj návrh predložený v kapitole 4.4. Na to, aby riadiaci podsystém zabránil výskytu uviaznutí pomocou navrhnutých algoritmov, musí obsahovať isté údajové štruktúry a vykonať určité kroky.

Údajové štruktúry sú

- vektor *Volne* rozmeru *m* pre počet neobsadených prostriedkov zapísaných podľa ich typov, kde *m* je počet množín prostriedkov v systéme,
- matica *Pridelene* rozmerov *n***m* pre počet pridelených prostriedkov jednotlivým procesom, kde *n* je počet inštancií procesov (činných i nečinných) a *m* je počet množín prostriedkov v systéme,
- matica *ZostavajucePotrebne* rozmerov *n***p* pre počet prostriedkov, ktoré jednotlivé inštancie procesov ešte budú potrebovať do skončenia svojho spracovania, kde *n* je počet inštancií procesov (činných i nečinných) a *p* je počet profesií (tentoraz nie množín) prostriedkov v systéme,
- necyklický dvojsmerne lineárne zreťazený zoznam *StavyProcesov* pre inštancie procesov v stave spracovania, kde každá inštancia obsahuje svoje identifikačné číslo, zoznam vrcholov značeného grafu *VrcholyNaSpracovanie*, ktoré určujú stav spracovania inštancie a číslo variantu značeného grafu, podľa ktorého sa vykonáva,
- vektor *StrukturaProcesu* rozmeru *q*, ktorý obsahuje odkazy na zápisy sieťových grafov, podľa ktorých sa môžu procesy spracovať – všetky údaje v tejto štruktúre sú pre daný systém konštantné a číslo *q* zodpovedá počtu rôznych použitých značených grafov, pričom svoj vlastný značený graf má nielen každý typ procesu, ale každý jeho variant, ak používa proces pružné smerovanie,
- vektor *MnozinyPreProfesie* rozmeru *p*, ktorý poskytuje v každom prvku zoznam množín, z ktorých možno pre profesiu daného indexu získať prostriedky – je to konštantná údajová štruktúra pre daný systém,
- konštanta *Rozdelovnik*, ktorá sa používa na oddelenie dolných a horných bitov čísel určujúcich počet prideľovaných alebo uvoľňovaných prostriedkov.

Vykonávané kroky sú nasledovné:

1) Pri každej žiadosti o prostriedky systému spustiť jeden z algoritmov A, B alebo C, ktorý splní alebo odmietne žiadosť posúdením stavu, ktorý by splnením žiadosti nasledoval.
2) Pri každom kroku pridelenia alebo uvoľnenia prostriedku v systéme aktualizovať stav údajových štruktúr *Volne*, *Pridelene* a *ZostavajucePotrebne* čiastkovým algoritmom 4 (pri nepoužití profesií možno použiť aj čiastkový algoritmus 1) a *VrcholyNaSpracovanie* príslušného procesu v údajovej štruktúre *StavyProcesov* čiastkovým algoritmom 2.

5 Výsledky riešenia a ich zhodnotenie

Upravené algoritmy popísané v predchádzajúcej časti som vyskúšal na modeli jednoduchého dopravného systému, ktorý spĺňa také základné požiadavky, aby bolo možné preveriť všetky podstatné vlastnosti vytýčené v doterajšej práci. Model som vytvoril v hierarchickej farbenej Petriho sieti s použitím nástroja CPN Tools, aj keď pre ukážku fungovania algoritmu bankára v riadení dopravného systému netreba **Petriho sieť**. Samotný algoritmus možno implementovať pomocou ľubovoľného programovacieho nástroja. Avšak vzhľadom na merateľnosť hlavného kritéria kvality algoritmu, ktorým bola absencia uviaznutí v systéme, som potreboval analyzovať kompletný stavový priestor systému so súbežným vykonávaním činností. A to sa dá realizovať práve v Petriho sieti bez nutnosti rozsiahlej implementácie modelu.

Ukážka riešenia taktiež primárne nevyžaduje **farbenú** Petriho sieť. Pre tento formalizmus som sa však rozhodol z dvoch dôvodov. Prvým je vyššia modelovacia schopnosť CPN, čo umožňuje vytvoriť jednoduchší a prehľadnejší model, ako by to bolo na úrovni P/T Petriho siete. Druhým dôvodom je existencia nástroja CPN Tools, ktorý je voľne dostupný a okrem hierarchickej farbenej Petriho siete poskytuje tiež schopnosť analyzovať stavový priestor PN a manipulačný jazyk pre realizáciu riadiacich funkcií algoritmu bankára.

Tieto zámery viedli napokon aj k odvodeniu teoretických definícií farbenej Petriho siete nesekvenčného procesu (NP-CPN) a systému nesekvenčných procesov (SNP-CPN), ktoré som uviedol v kapitole 4.1.2.4. Demonštračný model pomohol v praxi overiť aj tieto nové poznatky.

V prvej podkapitole opíšem demonštračný model, v ďalšej implementáciu algoritmu a na záver sa venujem opisu experimentov a ich výsledkov.

5.1 Demonštračný model

5.1.1 Požiadavky na model

Demonštračný model mal za cieľ **zachytiť podstatné rysy dopravného systému**, opísané v doterajšom texte. To zahŕňa tieto podmienky:

a) Sú prítomné činnosti premiestnenia a ostatné obslužné činnosti.
b) Procesy sa realizujú paralelným spracovaním vo vetvách (t. j. sú nesekvenčné) a súčasne využívajú pružné smerovanie prostredníctvom variantov spracovania.
c) Technologické postupy požadujú aspoň jeden prostriedok viac ako jedenkrát počas spracovania, čo znamená jeho opakované pridelenie a uvoľnenie.
d) Na výber prostriedkov sa používajú profesie.

Ďalším cieľom bola možnosť **analyzovať správanie modelu**, to znamená umožniť zvolenému nástroju v rozumnom čase (max. niekoľko dní) vypočítať stavový

priestor, z ktorého odvodzuje analytické vlastnosti. Podmienkou na splnenie tohto cieľa je primeraná veľkosť systému, t. j. primeraný počet prostriedkov a prebiehajúcich procesov primeranej zložitosti.

Medzi ciele modelu nepatrilo presné modelovanie reality v niektorom z existujúcich železničných systémov. Keď to bude prioritou, predpokladám, že prípadné diskutabilné detaily tohto demonštračného modelu možno odstrániť bez väčších ťažkostí.

5.1.2 Všeobecný popis modelu

Model reprezentuje **malú železničnú stanicu**, ktorá obsluhuje osobné vlaky na konci jednokoľajnej trate lokálneho významu. Koľajisko pozostáva z troch staničných koľají, ktoré sú na jednej strane napojené na traťovú koľaj a na druhej strane na rušňové depo s piatimi koľajami (Obr. 5.1).

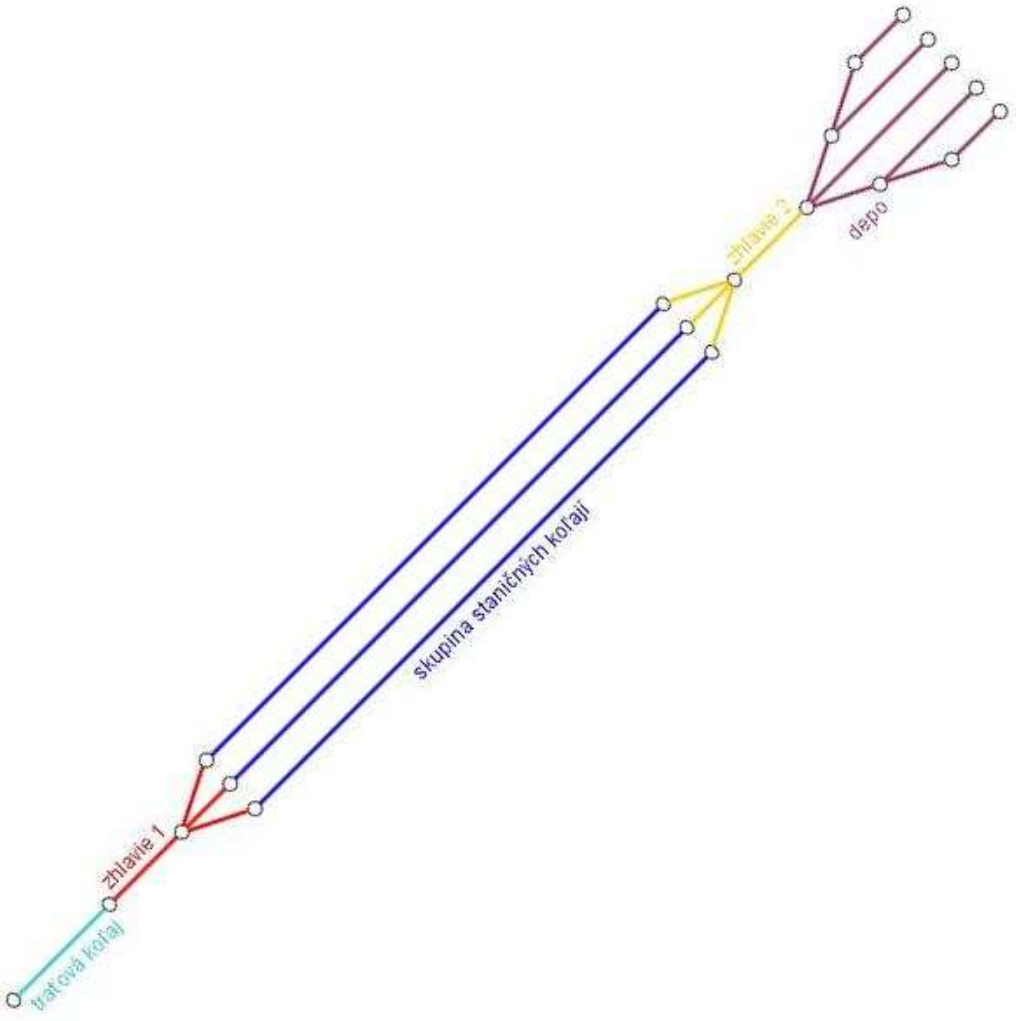
Vlaky po príchode do stanice okrem nástupu a výstupu cestujúcich absolvujú technickú prehliadku a prestavenie rušňa. Tu je na výber buď prestavenie rušňa, ktorý vlak do stanice priviedol alebo výmena rušňa s iným v depe. V obidvoch prípadoch sa rušeň po príchode odpája na strane smerom do depa a pred odchodom pripája na druhej strane smerom na trať. Pohyb vlakových rušňov pri úvraťovej prestavovacej jazde vyžaduje ďalšiu staničnú koľaj, obidve zhlavia a traťovú koľaj. V depe sa na začiatku nachádzajú dva voľné rušne a tri voľné koľaje.

V technologickom procese sa ďalej modelujú **dve profesie staničných zamestnancov**:

a) posunovač-sprievodca rušňov, ktorý odpája a pripája rušne,
b) vozmajster, ktorý vykonáva technickú prehliadku.

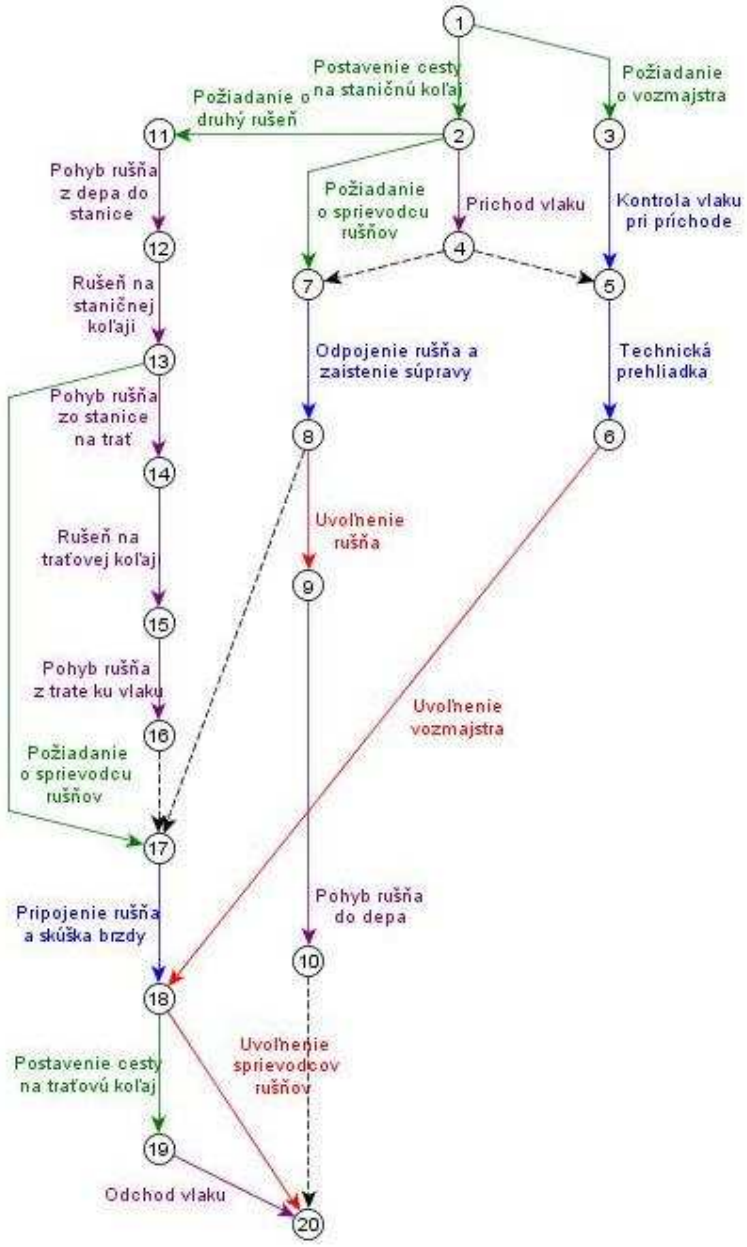
Správanie som popísal všeobecným dopravným procesom, ktorý obsahuje aj činnosti premiestnenia aj ostatné obslužné činnosti. Je zapísaný sieťovými (značenými) grafmi, kde sú farebne odlíšené premiestňovacie činnosti (fialová), ostatné obslužné činnosti (modrá), žiadosti o mobilné prostriedky (zelená), uvoľnenie mobilných prostriedkov (červená) a fiktívne činnosti (čierna čiarkovaná čiara).

Spracovanie vlaku má **dva varianty**. Zložitejší prvý variant (Obr. 5.2) predpisuje výmenu rušňov a používa o jedného posunovača-sprievodcu rušňov viac, aby mohli obaja pracovať súbežne, vykonať spoločne jednoduchú skúšku brzdy a aby vlak mohol byť rýchlejšie pripravený na odchod. Jednoduchší druhý variant (Obr. 5.3) predpokladá dlhší pobyt vlaku v stanici, počas ktorého pôvodný rušeň zmení svoju polohu pri súprave a celú procedúru obslúži iba jeden posunovač. V obidvoch variantoch sa vykonáva technická prehliadka, a taktiež nástup a výstup cestujúcich, ktorý nie je explicitne zapísaný v technologickom predpise činností procesu a jeho trvanie sa predpokladá počas celého pobytu vlaku na staničnej koľaji.



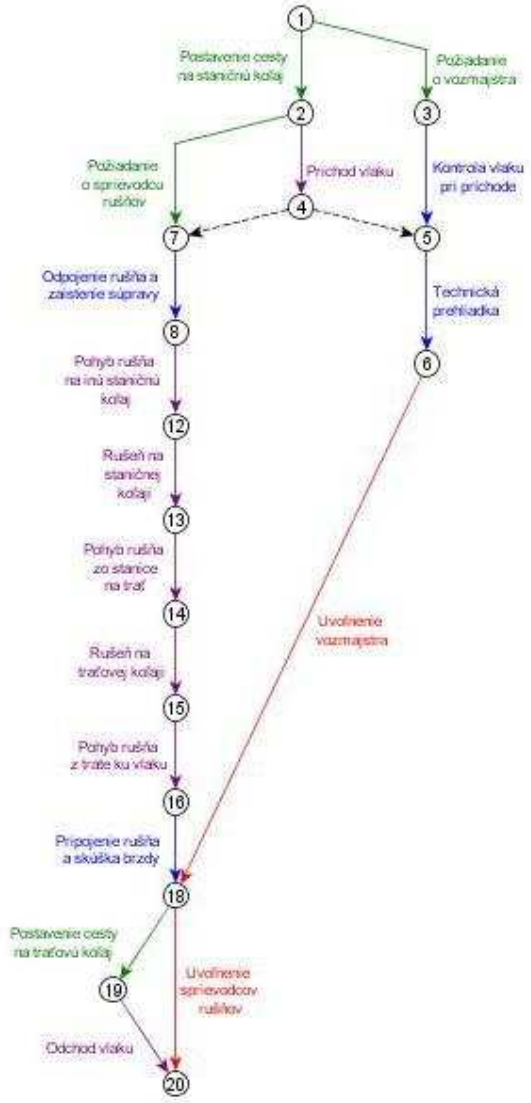
Obr. 5.1 Koľajisko železničnej stanice v testovacom modeli s piatimi skupinami koľají.

Obslužná čata v stanici zahŕňa 3 pracovníkov, z ktorých dvaja môžu vykonávať funkciu posunovača-sprievodcu rušňov a jeden funkciu vozmajstra. Tieto parametre platia pre prvú sledovanú verziu modelu. Možno s nimi experimentovať, tak ako aj s počtom staničných, traťových, depových koľají, počtom rušňov v depe alebo celkovým počtom vlakov v systéme. Ja som sa rozhodol vytvoriť ešte jednu konfiguráciu s odlišnou skladbou pracovnej čaty – zloženej z troch zamestnancov, ktorí môžu vykonávať aj úlohy posunovača-sprievodcu rušňov, aj úlohy vozmajstra. Pri ich prideľovaní je už nevyhnutné používať profesie.



Obr. 5.2 Sieťový graf technologického postupu obsluhy osobného vlaku v stanici – zložitejší variant 1.

Vytvorený model možno v oblasti systémov pridelenia prostriedkov kategorizovať ako **nesekvenčný viac-typový čiastočne usporiadaný RAS**. K maximálnej vymedzenej zložitosti mu chýba ešte opakovanie činností do kruhu, ktoré nepovažujem za tak často sa vyskytujúci jav v dopravnom systéme, ani za tak problematický jav z hľadiska vyhýbania sa uviaznutiu, že by sa mu musela venovať špeciálna pozornosť.



Obr. 5.3 Sieťový graf technologického postupu obsluhy osobného vlaku v stanici – jednoduchší variant 2.

Model však spĺňa všetky vlastnosti vytýčené v predchádzajúcej podkapitole: používa obidva druhy činností, procesy sú nesekvenčné a zároveň s pružným smerovaním. Opakované priradenie prostriedkov vidno pri pohyboch, keď napríklad zhlavie 1 na strane trate je každým procesom použité trikrát po sebe. Profesie sa používajú pri obslužnej čate, kde nastavenie vlastností pracovníkov určuje, či sú schopní vykonávať úlohy jednej alebo viacerých profesií.

5.1.3 Model vo farbenej Petriho sieti

Farbená Petriho sieť opísaného modelu sa nachádza na obrázkoch v prílohe D.1. Ide o **farbenú Petriho sieť systému s nesekvenčnými procesmi** (SNP-CPN), v ktorom sa nachádza iba jeden nesekvenčný proces, ako vidno na Obr. 9.11 zobrazujúcom hlavnú stránku modelu.

PN tohto procesu je farbená Petriho sieť nesekvenčného procesu (NP-CPN), rozdelená na hlavnú stránku a 2 podstránky. Činnosti reprezentujú prvky čiernej farby. Prostriedky by mohli byť umiestnené do jedného miesta, no sú z dôvodu prehľadnosti rozdelené do dvoch: tyrkysová farba sa používa pre pevný podsystém (koľaje, množina farieb *cTracksOcc*) a tmavohnedá farba pre pohyblivý podsystém (vlakové rušne a staniční zamestnanci, množina farieb *cMobileResourcesOcc*).

Na Obr. 9.12 je spoločná časť technologického predpisu pre obidva varianty. Zo sieťových grafov na Obr. 5.2 a na Obr. 5.3 obsahuje vrcholy 1 až 6 s priľahlými hranami. V CPN ide o prechody T1, T3, T4, T5 a T6 s priľahlými miestami a hranami a dva substitučné prechody na mieste vrchola 2, ktoré sú bránami k podstránkam zobrazujúcim varianty procesu. Na Obr. 9.13 je zložitejší variant 1 a na Obr. 9.14 jednoduchší variant 2 – obidva obsahujú reprezentáciu ostatných vrcholov a hrán príslušného sieťového grafu.

Mechanizmus profesií pri výbere prostriedku sa nachádza vo funkcii *CheckProfession*, ktorá má v tele implementovaný vzťah medzi používanými profesiami a modelovanými prostriedkami a vracia hodnotu *boolean* podľa toho, či prostriedok v argumente spĺňa požiadavku danej profesie. Funkcia sa zavolá vždy pri priradení prostriedku, a to v strážnej podmienke príslušného prechodu CPN. Ukážku možno vidieť pri prechode T3 v sieti na Obr. 9.12.

Model je uzavretý z analytických dôvodov (viď definícia Petriho siete pre RAS v kapitole 2.2.3.1), to znamená, že vlaky po opustení stanice a pobyte v jej okolí (miesto *Environment*) sa do nej opäť môžu vrátiť.

5.1.4 Reprezentácia procesu

Inštanciou procesu je v modeli zákazka vlaku, ktorý sa v stanici obslúži. Z implementačných možností vlakov sa v popredí ponúkajú najmä dve, ktoré porovnám. Prvá modeluje procesy rovnakými značkami. Druhá dáva každému

Tab. 5.1 Veľkosť stavového priestoru podľa označenia procesov.

	stavový priestor		
označenie vlakov	počet stavov	počet prechodov	čas výpočtu
2`1	**4883**	**14935**	**12**
1`1+1`2	**18002**	**59444**	**171**

procesu jedinečnú značku. Rozdiel medzi nimi je najmä v **rozsahu stavového priestoru** vytvoreného modelu.

V príklade pre dva procesy (vlaky), ako uvádza tab. 5.1, je v prvej verzii počet vrcholov aj počet hrán stavového priestoru (ktoré reprezentujú počet stavov a prechodov pôvodnej PN) takmer 4-krát menší a čas potrebný na výpočet stavového priestoru dokonca cca 14-krát menší. Preto sa výhodnejším javí používanie rovnakých značiek pre všetky procesy. Avšak vzhľadom na to, že implementovaný algoritmus bankára funguje **iba pri jedinečnom rozlišovaní značiek**, budem ďalej používať iba to – s neželaným dôsledkom nárastu stavového priestoru a času jeho výpočtu.

5.1.5 Príklady stavov uviaznutia

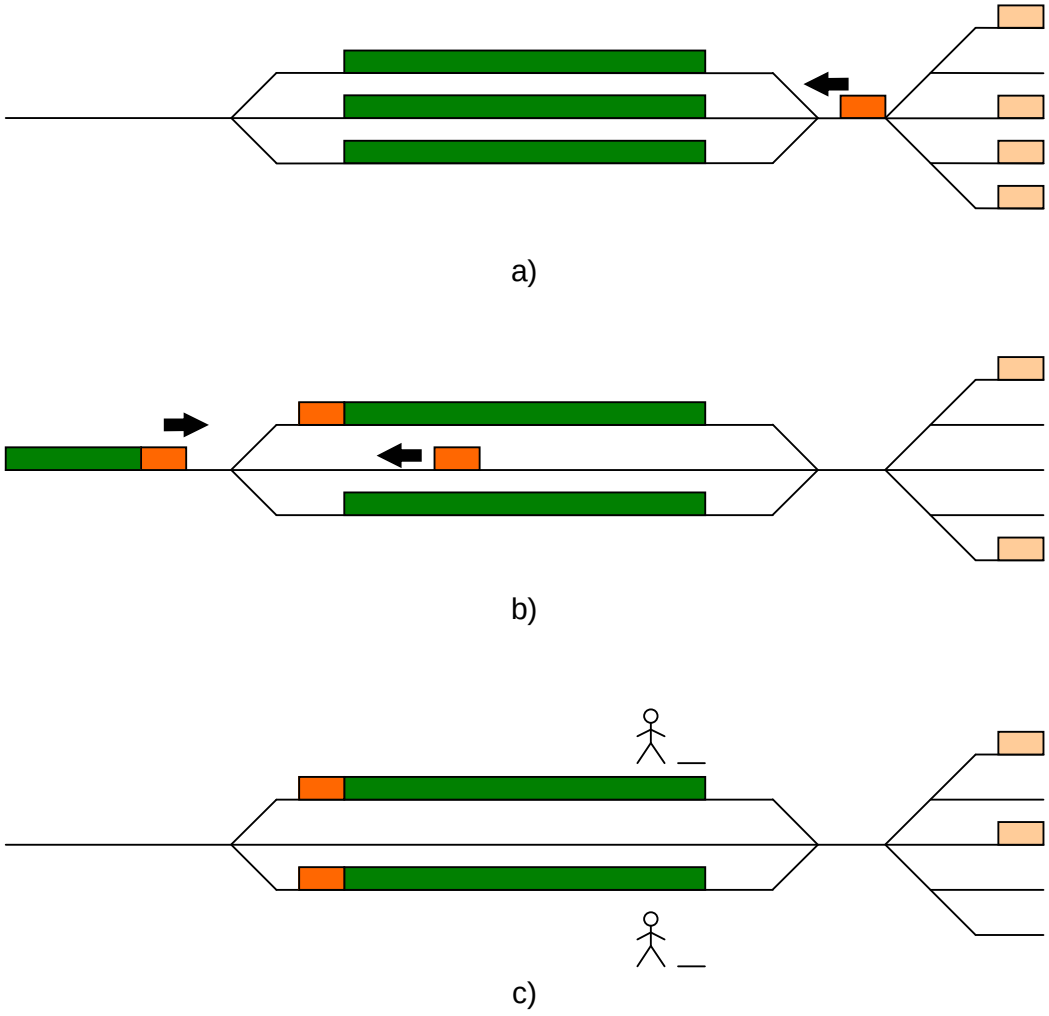
V protokole analýzy základnej verzie modelu (t. j. čata pozostáva z dvoch posunovačov a jedného vozmajstra) pre 3 vlaky sa nachádza 306 stavov uviaznutia. Mnohé z nich sú princípom rovnaké, líšia sa iba v tom, ktorý vlak sa kde nachádza vďaka používaniu jedinečných značiek pre vlaky. Keď som použil rovnakú hodnotu značky pre každý vlak (čo pri nepoužití algoritmu bankára neprekáža), analýza ukázala 24 rôznych stavov uviaznutia, ktoré môžu v danom modeli nastať.

Uvádzam z nich niekoľko príkladov, ktoré sú zobrazené na Obr. 5.4. Niektoré situácie sa odborníkovi z praxe môžu zdať ako nereálne, pokiaľ systém riadi človek. Treba si však uvedomiť, že tu ide o riadenie systému počítačom, ktorý môže bez algoritmu na vyhnutie sa uviaznutiu pripustiť ľubovoľnú situáciu.

V prvej situácii na Obr. 5.4a vidno všetky tri koľaje v stanici obsadené vlakovými súpravami bez rušňov, pričom každý z nich čaká na rušeň z depa, aby sa pripojil k súprave na stranu východu zo stanice. Prvý z rušňov tak chce vykonať, no nemá voľnú staničnú koľaj, kadiaľ by prešiel na druhú stranu stanice.

V druhej situácii na Obr. 5.4b je jedna koľaj v stanici obsadená vlakom pripraveným na odchod a jedna vlakovou súpravou, ktorá čaká na rušeň prichádzajúci z depa. Tento rušeň obsadzuje práve tretiu staničnú koľaj a čaká na traťovú koľaj, aby sa mohol úvraťovou jazdou dostať k pridelenej súprave. Traťová koľaj je však už obsadená ďalším vlakom, ktorý prichádza do stanice.

Tretia situácia na Obr. 5.4c má dokonca jednu staničnú koľaj voľnú a vlakové súpravy na ostatných dvoch koľajách majú už pristavené rušne z depa. Avšak obidve majú prideleného jedného posunovača a na dokončenie prípravy na odchod potrebujú ešte jedného, pričom nijaký už nie je voľný. Už v tejto situácii je to lokálne uviaznutie. Úplné bude vtedy, keď sa aj tretia staničná koľaj obsadí tretím vlakom, z depa sa začne pohybovať rušeň do stanice a vznikne situácia podobná tej na Obr. 5.4a.



Obr. 5.4 Príklady stavov uviaznutia v demonštračnom modeli.

5.2 Implementácia navrhnutého algoritmu

5.2.1 Údajové štruktúry a funkcie

Navrhnutý algoritmus je v modeli napísaný v jazyku CPN ML. Vzhľadom na jeho principiálnu odlišnosť od bežných sekvenčných jazykov (orientácia na funkcie a rekurziu) je aj implementácia algoritmu odlišná od verzií naznačených v kapitole 4.4.

Výsledná verzia v CPN obsahuje cca 30 funkcií, ktoré realizujú všetky hlavné a čiastkové algoritmy a nimi využívané pomocné operácie. Funkcie sa delia na

- všeobecné funkcie narábajúce so zoznamami údajov,
- funkcie pre obsluhu stavu systému,

- funkcie algoritmu bankára.

Všetky viacprvkové údajové štruktúry (polia) sú realizované ako zoznamy prvkov, ktoré môžu na mieste prvku mať takisto zoznam.

5.2.2 Zapojenie do Petriho siete

V Petriho sieti sa nachádzajú dve miesta, ktoré uchovávajú údaje potrebné ako pre algoritmus bankára – *Banker's Alg Data*, tak aj pre stav systému – *Process States* (viď Obr. 9.15). Vďaka princípu zlučovania miest sa vyskytujú na všetkých stránkach SNP-CPN kópie týchto miest obsahujúce rovnaké značky, čím je možný prístup k rovnakej kópii údajov na jednom mieste. Kópie miest sú na stránkach zapojené na všetky prechody, v ktorých sa pristupuje k ich údajom. Zapojenie je vykonané pomocou dvojice hrán: jedna hrana privádza značku s údajmi z miesta do prechodu pred výpočtom a druhá hrana vedie značku s aktualizovanými údajmi z prechodu naspäť do miesta.

Najdôležitejšou je funkcia, ktorá rozhoduje o bezpečnosti stavu po pridelení prostriedkov. Volá sa v strážnej podmienke každého prechodu v SNP-CPN, kde sa má splniť žiadosť o pridelenie prostriedkov. Ak je nový stav bezpečný, výsledok funkcie povolí prechod a ten sa v CPN môže vykonať. V prípade negatívnej odpovede sa prechod neaktivuje, no v budúcnosti ho možno aktivovať, ak sa stav systému zmení. Tu sa prejavuje silná stránka simulátora formalizmu Petriho siete, ktorý po zmene stavu v CPN automaticky preveruje možnú aktiváciu všetkých prechodov, a tak môže aktivovať aj predtým zakázaný prechod.

Ďalšie dve podstatné funkcie slúžia na aktualizáciu údajových štruktúr pre algoritmus po vykonaní zmeny stavu systému. Jedna aktualizuje premenné algoritmu bankára a realizuje sa iba vtedy, keď dochádza k zmene stavu v pridelení prostriedkov procesom (čo nie je napríklad na prechode T5 alebo T8). Druhá mení stav systému v premennej podľa zakódovaných sieťových grafov pri vykonaní každého prechodu, nakoľko každý vykoná zmenu systému.

Tieto dve funkcie sa volajú v kódovom segmente každého prechodu (kódové segmenty sú 4- až 6-riadkové fragmenty textu okolo siete na Obr. 9.15), ktorý sa vykonáva len pri aktivácii prechodu. To je podstatný rozdiel od strážnej podmienky, ktorá sa testuje vždy pri pokuse aktivovať prechod. To znamená, že vlastný algoritmus bankára sa vykoná pri každom pokuse aktivovať príslušný prechod, aktualizačné funkcie až pri prechode do nového stavu.

Všetky výpočty sa realizujú aj pri simulácii CPN, aj pri výpočte jej stavového priestoru.

5.3 Experimenty a ich výsledky

5.3.1 Výber experimentov

Pre overenie navrhnutého algoritmu som pripravil súbory výpočtov vychádzajúce z dvoch konfigurácií modelu, ktoré majú spoločné

- koľajisko – jedna traťová, tri staničné a päť depových koľají (Obr. 5.1), na ktorých sa na začiatku nachádzajú dva voľné vlakové rušne,
- technologické postupy podľa definovaných sieťových grafov v dvoch variantoch (Obr. 5.2 a Obr. 5.3).

Konfigurácie sa líšia v týchto vlastnostiach:

a) **Konfigurácia I** obsahuje 2 posunovačov-sprievodcov rušňov a 1 vozmajstra.
b) **Konfigurácia II** obsahuje 3 pracovníkov schopných vykonávať obidve profesie.

Zámerom rozličných konfigurácií bolo porovnanie 2 systémov: na jednej strane systému, kde majú profesie staničných zamestnancov (alias prostriedkov systému) presne vymedzené úlohy, na druhej strane systému, kde sú všetci zamestnanci schopní vykonať všetky úlohy a pri výbere z nich sa musí uplatniť mechanizmus profesií.

Každú konfiguráciu som analyzoval najskôr bez a potom s ovplyvňovaním prideľovania prostriedkov algoritmom bankára, pričom som vyskúšal všetky tri verzie. Dohromady to tvorí 8 rôznych verzií modelu:

1) Konfigurácia I bez algoritmu bankára.
2) Konfigurácia I s verziou algoritmu A.
3) Konfigurácia I s verziou algoritmu B.
4) Konfigurácia I s verziou algoritmu C.
5) Konfigurácia II bez algoritmu bankára.
6) Konfigurácia II s verziou algoritmu A.
7) Konfigurácia II s verziou algoritmu B.
8) Konfigurácia II s verziou algoritmu C.

Následne som každú z verzií analyzoval postupne pre 2, 3 a 4 vlaky (alias procesy) prítomné v (uzavretom) systéme, čím som chcel sledovať, ako sa zmenia jednotlivé sledované veličiny pri zmene veľkosti.

Iný počet vlakov nemá pri danej konfigurácii zmysel. Pri jednom vlaku by chýbala interakcia medzi procesmi a aj hrozba uviaznutí v systéme. Viac ako štyri vlaky na druhej strane sa nemôžu v modeli vyskytovať v činnom stave, nakoľko maximálny počet koľají, kde môžu stáť vlaky naraz, je štyri (3 staničné a 1 traťová). A ako vidno intuitívne aj z príkladov uviaznutí, súčasná prítomnosť štyroch vlakov v modeli v činnom stave končí nevyhnutne uviaznutím.

5.3.2 Výsledky analýzy

Pred utvorením analytických záverov sa počíta stavový priestor, ktorým je **graf dosiahnuteľnosti** značení z počiatočného značenia Petriho siete. Jednotlivé stavy

systému sa v grafe reprezentujú jeho vrcholmi a prechody medzi stavmi zasa hranami medzi vrcholmi. Vo verziách s algoritmom bankára sa pre každý prechod počítajú príslušné algoritmy tak, ako som ich opísal v kapitole 4.4. To znamená, že rovnaký algoritmus sa spustí rádovo tisíc-, desaťtisíc-, ba i stotisíckrát – podľa výskytu možných prechodov medzi rôznymi stavmi v systéme. Ak algoritmus nepovolí priradenie prostriedku, teda vykonanie potenciálneho prechodu, tento prechod sa do grafu dosiahnuteľnosti nezapíše. To znamená, že pričinením algoritmu sa môže zmenšiť počet stavov a prechodov v systéme.

Z grafu dosiahnuteľnosti sa odvádzajú vlastnosti modelu. Informácie o grafe a odvodené vlastnosti sa zapíšu do **protokolu** produkovaného v CPN Tools štandardnej forme. Ukážka takého protokolu so stručným objasnením jeho častí sa nachádza v prílohe D.2. Z informácií zapísaných v protokole sú pre túto prácu potrebné iba niektoré. Zo základných informácií ma zaujímal počet vrcholov, počet hrán a čas výpočtu grafu dosiahnuteľnosti a z odvodených vlastností CPN je pre hodnotenie významná živosť – presnejšie počet stavov s mŕtvymi značeniami.

Výsledky zhŕňajú dve tabuľky. Jeden riadok v tabuľke zachytáva výsledky z jedného protokolu. Kým ostatné parametre sú presné a nemenné aj pri opakovaní výpočtov pre rovnakú verziu modelu, **čas výpočtu** závisí od externých vplyvov počítačového systému na výpočet nástroja a môže sa pri rôznych výpočtoch meniť. Preto tabuľky poskytujú iba jeho orientačné hodnoty zapísané kurzívou.

V tab. 5.2 sa nachádzajú výsledky analýzy navrhnutých konfigurácií. Niektoré polia obsahujú namiesto výsledkov hviezdičky – je to tam, kde sa mi nepodarilo vypočítať úplný stavový priestor. Použitý nástroj CPN Tools po určitom čase výpočtu skončil v stave, keď signalizoval prebiehajúci výpočet, no vyťaženie systémových prostriedkov naznačovalo, že sa už nič nepočíta. Prípadne sa mal výpočet v určitom časovom limite prerušiť, no nestalo sa tak a jediným možným ukončením bolo uzavretie nástroja bez možnosti zápisu získaných výsledkov. Súvisí to pravdepodobne s obmedzením výpočtových možností nástroja, prípadne použitého počítačového prostredia. Ide o riadky 4-1, 4-4 a 4-5.

Riadky 4-1 a 4-5 sa týkajú konfigurácií bez algoritmu bankára. Pre túto verziu modelu možno vypočítať stavový priestor aj pre nejednoznačné značky procesov (podrobnosti v kap. 5.1.4). V záujme dokumentovať, že v modeli bez algoritmu bankára určite nastávajú stavy uviaznutia aj pre tieto konfigurácie, pripájam v tab. 5.3 výsledky výpočtov pre nejednoznačné označenie procesov (označenie riadkov korešponduje s označením v tab. 5.2). Tu sa výrazne zmenšuje stavový priestor modelu, a tak uvádzam výpočty nielen pre konfigurácie, ktorým chýbajú výsledky v tab. 5.2, ale aj pre ostatné konfigurácie bez BA pre porovnanie.

V tab. 5.3 vidno, že v každej z konfigurácií bez BA sa nachádza aspoň jeden stav uviaznutia, čo sa po zavedení jednoznačných značiek pre procesy zachová, prípadne počet týchto stavov sa zväčší. Pre neúspešne analyzované verzie bez BA 4-1 a 4-5 v tab. 5.2 možno teda označiť počet uviaznutí kladným celým číslom *p*.

Pre neúspešne analyzovanú verziu 4-4 v tab. 5.2 som si dovolil počet uviaznutí označiť nulou. Vychádza to zo simulácie Petriho siete, ktorá sa v tomto prípade správala analogicky k verziám, kde sa analytické výsledky podarilo získať: v prípade modelu s výskytom uviaznutí sa v simulácii vyskytol stav uviaznutia vždy najneskôr po rádovo niekoľko tisíc krokoch, kým v modeli bez uviaznutí sa taký stav nevyskytol ani po vykonaní milióna krokov (vykonaných niekoľko desiatok pokusov).

Zo získaných hodnôt si možno všimnúť niekoľko záverov:

1) Všetky verzie modelu, ktoré **neobsahujú BA** na vyhýbanie sa uviaznutiu (napr. 2-1, 2-5, 3-1 alebo 3-5) majú **aspoň jeden stav s mŕtvym značením**, t. j. v systéme sa môže vyskytnúť stav uviaznutia. Naopak vo všetkých ostatných verziách, kde sa BA použil, nie je hlásený nijaký stav uviaznutia. Teda BA plní svoj účel.
2) Verzie s BA (tam, kde to možno porovnať) majú **niekoľkokrát menší** stavový priestor ako verzie bez riadenia algoritmom. To potvrdzuje, že BA vylučuje nebezpečné stavy a časť bezpečných stavov, ktoré nedokáže zhodnotiť ako bezpečné. Existuje tiež **rozdiel medzi algoritmami** A a B na jednej strane a algoritmom C na druhej strane, ktorého stavový priestor je väčší. To potvrdzuje očakávanie: alg. C má prijať všetky stavy ako alg. A a B a vďaka zložitejšiemu výpočtu ešte ďalšie stavy.
3) Taktiež sa potvrdzuje očakávanie, že počet prijatých stavov algoritmami A a B je rovnaký, akurát **alg. B** má pracovať **rýchlejšie**. Vo všetkých relevantných porovnaniach je tento fakt vidieť okrem konfigurácie II pre 4 vlaky. Tam sa mohli prejaviť externé vplyvy na výpočet. Celkovo možno predpokladať, že časový rozdiel medzi verziami s alg. A a s alg. B sa v systémoch s väčším počtom procesov (rádovo stovky) prejaví oveľa viac ako pri 2-4 procesoch.
4) Z **porovnania** výsledkov relevantných verzií medzi **konfiguráciami I a II** vychádza, že konfigurácia II s 3 univerzálnymi pracovníkmi má asi o 20-30% väčší stavový priestor, čo sa dá očakávať.
5) So **stúpajúcim počtom procesov** stúpa aj počet stavov systému. Pre 5 a viac procesov už tento trend nepokračuje (preveril som bez uvedenia výsledkov v tab.) – v súlade s vyššie uvedeným zdôvodnením o počte vlakov v systéme.
6) Aj pre **pomerne malý systém** je stavový priestor pomerne veľký a analýza trvá relatívne dlhý čas.

Tab. 55.2 Výsledky analytických výpočtov pre navrhnuté konfigurácie modelu – pre jednoznačné značky procesov.

	konfig.	BA	počet			čas výpočtu [s]
			stavov	prechodov	stavov uviaznutia	
< < < 2 vlaky > > >						
2-1	**I** (2 + 1)	–	**18002**	59444	**1**	*163*
2-2		A	**7986**	21942	**0**	*61*
2-3		B	**7986**	21942	**0**	*59*
2-4		C	**11142**	33010	**0**	*126*
2-5	**II** (3)	–	**20798**	72008	**6**	*230*
2-6		A	**9978**	27888	**0**	*98*
2-7		B	**9978**	27888	**0**	*94*
2-8		C	**12172**	35524	**0**	*144*
< < < 3 vlaky > > >						
3-1	**I** (2 + 1)	–	**179408**	605634	**306**	*13252*
3-2		A	**23362**	64521	**0**	*497*
3-3		B	**23362**	64521	**0**	*491*
3-4		C	**83920**	256194	**0**	*6323*
3-5	**II** (3)	–	**228500**	810426	**760**	*23963*
3-6		A	**29338**	82359	**0**	*797*
3-7		B	**29338**	82359	**0**	*773*
3-8		C	**35920**	105267	**0**	*1185*
< < < 4 vlaky > > >						
4-1	**I** (2 + 1)	–	*****	*	***p > 0***	*
4-2		A	**46327**	128172	**0**	*1937*
4-3		B	**46327**	128172	**0**	*1843*
4-4		C	*****	*	***0***	*
4-5	**II** (3)	–	*****	*	***p > 0***	*
4-6		A	**58279**	163848	**0**	*3052*
4-7		B	**58279**	163848	**0**	*3270*
4-8		C	**71443**	209664	**0**	*4962*

Výpočty: procesor Intel® Pentium® D CPU 3 GHz, 1 GB vnútornej pamäti.

Tab. 5.3 Výsledky analytických výpočtov pre model bez algoritmu bankára a s nejednoznačnými značkami procesov.

	konfig.	vlaky	počet			čas výpočtu [s]
			stavov	prechodov	stavov uviaznutia	
2-1	**I** (2 + 1)	2	**4883**	14935	**1**	*11*
3-1		3	**13605**	43760	**24**	*62*
4-1		4	**17243**	55690	**40**	*99*
2-5	**II** (3)	2	**5741**	18223	**2**	*16*
3-5		3	**17420**	57833	**64**	*107*
4-5		4	**22875**	76185	**98**	*184*

Výpočty: procesor Intel® Pentium® D CPU 3 GHz, 1 GB vnútornej pamäti.

6 Záver

Vo svojej dizertačnej práci som sa venoval vývoju prostriedkov pre podporu rozhodovania pri riadení dopravných procesov so zameraním na riešenie stavov uviaznutia, ktoré môžu pri automatickom riadení dopravného systému (DS) vznikať. V závere zhrniem prácu, ktorú som urobil, vymenujem, aké prínosy vidím v jej výsledkoch, a naznačím, ktorým problémom sa možno v ďalšom výskume venovať.

6.1 Zhrnutie

V **analytickej časti** práce som pri analýze súčasného stavu v kapitole 2 čerpal najviac z literatúry v oblasti operačných systémov a pružných výrobných systémov, kde sa problematika stavov uviaznutia rieši najviac. Z formalizmov som sa zameral najmä na systémy diskrétnych udalostí (DES), systémy prideľovania prostriedkov (RAS) a Petriho siete (PN). Pokúsil som sa urobiť stručný, no zároveň dostatočný prehľad v tejto problematike a výsledkom je pomerne rozsiahla analytická časť práce.

Vlastné riešenie úlohy v kapitole č. 4 som začal popisom všeobecného dopravného systému z hľadiska systému prideľovania prostriedkov a vytvorením jeho modelu v Petriho sieti. Z analýzy činností a procesov som usúdil, že všeobecný dopravný systém najlepšie vystihuje nesekvenčný viac-typový čiastočne usporiadaný RAS bez cyklického opakovania činností. Je to kategória, ktorá kombinuje súbežné činnosti s pružným smerovaním spracovania procesu. Okrem nich je typickým javom vo všeobecnom dopravnom systéme opakované prideľovanie a uvoľňovanie prostriedkov počas spracovania jedného procesu a narábanie s prostriedkami prostredníctvom profesií.

Tvorba modelu DS v kapitole 4.1.2 začala ukážkou postupu, ako možno v P/T Petriho sieti vytvoriť model premiestňovacích procesov z dopravnej siete a model všeobecných procesov zo sieťových grafov, ktoré sa v praxi používajú. Pri snahe zjednodušiť vznikajúci model systému a vďaka dostupnosti počítačovej podpory pre tvorbu a analýzu farbených Petriho sietí som poukázal na možnosti zavedenia atribútov farbenej Petriho siete v tomto procese a to ma doviedlo k vytvoreniu dvoch definícií v kapitole 4.1.2.4: farbenej Petriho siete nesekvenčného procesu (NP-CPN) a farbenej Petriho siete systému s nesekvenčnými procesmi (SNP-CPN).

Druhá časť tvorivej práce sa orientovala na **riešenie stavu uviaznutia** v rozoberanom systéme. Z analýzy známych prístupov v kapitolách 4.2 a 4.3 vyšlo ako najvhodnejšie pre dopravný systém vyhýbanie sa uviaznutiu pomocou informácií o stave systému a v ňom rozšírený **algoritmus bankára**, ktorý dokáže úplne zabrániť uviaznutiam pri pomerne malom obmedzení práce systému a navyše sa javil ako najperspektívnejší pri použití pre systém s nesekvenčnými procesmi. Upravil som ho pre potreby všeobecného DS definovaním potrebných údajových štruktúr, zostavením čiastkových algoritmov a ich zahrnutím do troch hlavných algoritmov známych z literatúry (kapitola 4.4).

Na demonštráciu použiteľnosti svojich návrhov som vytvoril **demonštračný model** vo forme SNP-CPN, zahŕňajúci vytýčené vlastnosti všeobecného dopravného systému ako súbežné spracovanie procesu, pružné smerovanie, opakované prideľovanie a použitie profesií pri výbere prostriedkov (kap. 5). Implementácia modelu taktiež obsahuje navrhnuté verzie algoritmu bankára na overenie jeho účinnosti pri zabránení vzniku stavov uviaznutia v systéme.

Analýzou experimentov modelu, ktorej výsledky prináša kap. 5.3, sa potvrdilo, že algoritmus bankára je schopný zabrániť uviaznutiam, pričom jeho jednotlivé verzie obmedzia stavový priestor modelu v rozličnej miere. Algoritmy A a B ho obmedzia rovnako, no algoritmus B môže dôjsť k riešeniu v niektorých prípadoch za kratší čas. Algoritmus C vďaka zložitejšej implementácii prijme väčší počet stavov ako algoritmy A a B.

6.2 Prínosy dizertačnej práce

Prínosy tejto práce možno rozdeliť na teoretické a praktické.

Medzi **teoretické** prínosy svojej dizertačnej práce radím nasledovné aspekty:

1) Vytvoril som všeobecný **formalizovaný model** dopravného systému vo farbenej Petriho sieti na princípe systému prideľovania prostriedkov, ktorý zahŕňa všetky podstatné rysy DS ako súbežný priebeh činností v rámci jedného procesu, možnosť pružného smerovania pri jeho spracovaní, opakované prideľovanie a použitie profesií pri výbere prostriedkov.
2) Vytvoril som **teoretickú definíciu** farbenej Petriho siete nesekvenčného procesu (NP-CPN) a následne aj farbenej Petriho siete systému s nesekvenčnými procesmi (SNP-CPN), ktoré umožňujú zjednodušiť modely z P/T Petriho sietí pre zložitejšie systémy – pričom pri študovaní literatúry som našiel pomerne málo príspevkov o modelovaní nesekvenčných procesov a dosiaľ som nenašiel nič o použití farbenej Petriho siete za týmto účelom.
3) Algoritmus bankára som upravil a úspešne overil pre systém prideľovania prostriedkov s **nesekvenčnými** procesmi (NS-RAS), o čom som dosiaľ nevidel v literatúre nijakú publikáciu. Rozšíril som ho o možnosť používať profesie pri výbere prostriedkov pre procesy a taktiež pracovať s nesekvenčnou štruktúrou procesu. Ukázalo sa, že algoritmus je schopný fungovať aj v takomto systéme.

Po **praktickej** stránke vidím prínos výsledkov tejto práce predovšetkým v rozvoji modelovania automatického riadenia dopravných systémov, a to v dvoch smeroch:

1) Predovšetkým je to riadenie komplexných simulačných modelov, ktoré majú zrejmý prínos pre reálne systémy. Detailnejším pochopením princípov prideľovania a uvoľňovania prostriedkov možno skvalitniť tvorbu simulačných modelov a zavedením algoritmu bankára do riadiaceho systému zvýšiť robustnosť simulačného modelu, čo môže zefektívniť jeho výstavbu a rozšíriť jeho použiteľnosť.

2) Pri riadení reálneho dopravného systému sa môže vytvorený model s algoritmom bankára použiť ako pomocný nástroj pri rozhodovaní človeka.

6.3 Ďalšie problémy

Okrem opísaného riešenia táto práca poukázala aj na niektoré otázky a úlohy pre ďalšie skúmanie.

Ako som to naznačil pri praktických prínosoch, prvou úlohou sa javí overenie navrhnutých údajových štruktúr a verzií algoritmu bankára v nástroji na simuláciu uzlov dopravnej siete Villon, pomocou ktorého sa budujú komplexné simulačné modely s rádovo desiatkami procesov a stovkami typov prostriedkov. Vzhľadom na zložitosť modelov i samotného nástroja je to pomerne náročná úloha, ktorá sa nezmestila do rámca tejto práce. Nasadenie riešenia v rozsiahlejších modeloch (pravdepodobne už iba simuláciou, nie kompletnou analýzou stavového priestoru) môže tiež pomôcť overiť hypotézu, že čím je viac procesov a prostriedkov v systéme, tým sa rozdiel v trvaní výpočtu algoritmu A a algoritmu B zväčšuje.

Druhou úlohou je preverenie iných prístupov k riešeniu stavov uviaznutia ako vyhýbanie sa mu. Z analýzy v kapitole 4.2 vyplýva, že by sa dali použiť aj niektoré ďalšie prístupy, avšak za cenu zníženia efektívnosti systému. Napríklad detekciu uviaznutia a zotavenie z neho možno aplikovať pre niektoré prostriedky dopravného systému (napr. personál) viac, pre iné (napr. koľaje) menej, resp. závisí to od ceny za zotavenie systému pri ich odňatí procesu. Rozvinutie tejto myšlienky a overenie na konkrétnych modeloch zostáva zatiaľ otvorené.

Ďalšou myšlienkou na preverenie je obmedzenie počtu používaných typov systémových prostriedkov v algoritme bankára. Keďže typov prostriedkov v zložitom dopravnom systéme môžu byť až stovky, pričom mnohé z nich sa (vďaka svojmu určeniu) nemôžu nikdy dostať navzájom do konfliktu v stave uviaznutia, je vítaná ich redukcia iba na tie typy, ktoré sa môžu dostať do konfliktu. To by mohlo zefektívniť výpočet algoritmu bankára. Redukcia sa môže vykonať prostredníctvom prídavnej analýzy predpisov spracovania procesov (ich sieťových grafov) pred spustením systému do prevádzky a po každej zmene predpisov.

Posledná oblasť skúmania sa týka implementácie algoritmu bankára. V publikácii (Lawley et al., 1998) autori simuláciou zistili, akú časť bezpečných stavov prijmú jednotlivé verzie algoritmu za usporiadané, t. j. aká je ich účinnosť pri určovaní bezpečnosti stavu. V tejto práci sa zavedením profesií a súbežného spracovania procesov do algoritmu bankára vniesli ďalšie parametre, ktoré ovplyvňujú účinnosť algoritmov, ako opisujem v kapitole 4.4.4.1. Tým sa otvára priestor pre tvorbu ďalších úprav verzií algoritmu a preverenie, aký efekt sa tým v modeli dosiahne: ako sa zmení stavový priestor prijatý algoritmami a ako sa predĺži čas pre jeho analýzu.

7 Zoznam použitých zdrojov

(Adamko-Sadloň, 2003) ADAMKO, Norbert – SADLOŇ, Ľubomír: Aplikácia sieťových grafov v simulačných modeloch železničných uzlov. In: Zborník konferencie *Infotrans 2003*, Pardubice, 2003.

(Araki et al., 1977) ARAKI, Toshiro – SUGIYAMA, Yuji – KASAMI, Tadao: Complexity of the deadlock avoidance problem [Zložitosť problému vyhýbania sa uviaznutiu]. In: *2nd IBM Symposium on Mathematical Foundations of Computer Science*, str. 229-257, IBM, 1977. Citované v publikácii (Reveliotis et al., 1997).

(Bažant-Žarnay, 2005/1) BAŽANT, Michael – ŽARNAY, Michal: Formalizace řešení přidělení náhradní nástupištní koleje pro zpožděný vlak. In: zborník príspevkov medzinárodnej konferencie *Informační technologie v dopravě – INFOTRANS 2005*, Univerzita Pardubice, Pardubice, 2005, str. 115-133. ISBN 80–7194–792–X.

(Bažant-Žarnay, 2005/2) BAŽANT, Michael – ŽARNAY, Michal: Modelování situací vznikajících při zpoždění vlaků v osobních železničních stanicích. In: zborník konferencie *Dopravní systémy 2005*, Dopravní fakulta Jana Pernera, Univerzita Pardubice, Pardubice, 2005, str. 248-256. ISBN 80–7194–805–5.

(Bažant-Žarnay, 2005/3) BAŽANT, Michael – ŽARNAY, Michal: Simulation model of Train Connections for Delayed Trains in Passenger Stations [Simulačný model prípojov pre zmeškané vlaky v osobných staniciach]. In: zborník príspevkov medzinárodnej konferencie *Žel 2005*, Žilinská univerzita, Žilina, 2005, 2. diel, str. 142-147. ISBN 80-8070-400-7.

(Cenek, 1996) CENEK, Petr: *Modelování a optimalizace procesů na dopravních sítích*. Habilitačná práca, Žilinská univerzita v Žiline, 1996.

(Cenek et al., 1994) CENEK, Petr – KLIMA, Valent – JANÁČEK, Jaroslav: *Optimalizace dopravních a spojových procesů*. Vysoká škola dopravy a spojov, Žilina, 1994. ISBN 80–7100–197–X.

(Coffman et al., 1971) COFFMAN, Edward G. – ELPHICK, M. J. – SHOSHANI, Arie: System deadlocks [Uviaznutia systému]. In: *Computing Surveys*, vol. 3, no. 2, 1971, str. 67-78. Dostupné na:
<http://www.cs.umass.edu/~mcorner/courses/691J/papers/TS/coffman_deadlocks/coffman_deadlocks.pdf#search=%22coffman%20deadlock%20paper%22>

(CPN Tools, online) *CPN Tools home page* [Domovská webová stránka k CPN Tools]. [online]. [citované 25. novembra 2006]
Dostupné na: <http://www.daimi.au.dk/CPNTools/>.

(CPN Tools help, online) *CPN Tools help* [Nápoveda k CPN Tools]. [online]. [citované 25. novembra 2006]
Dostupné na: <http://wiki.daimi.au.dk/cpntools-help/>.

(Češka, 1994) ČEŠKA, Milan: *Petriho sítě. Úvod do teorie a nástrojů pro aplikaci Petriho sítí.* CERM, Brno, október 1994. ISBN 80-85867-35-4

(Dijkstra, 1965) DIJKSTRA, Edsger Wybe: *Cooperating sequential processes* [Spolupracujúce sekvenčné procesy]. Technical report, Technological University, Eindhoven, Netherlands, September 1965. Znovu vydané v: F. Genuys, editor: *Programming Languages*. Academic Press, New York, 1968, str. 43-112.

(Ezpeleta et al., 1995) EZPELETA, Joaquín – COLOM, José Manuel – MARTÍNEZ, Javier: A Petri net based deadlock prevention policy for flexible manufacturing systems [Stratégia predchádzania uviaznutiu založená na Petriho sieti pre pružné výrobné systémy]. In: *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, apríl 1995, str. 173-184. Podľa: (Tricas, 2003).

(Ezpeleta et al., 1998) EZPELETA, Joaquín – GARCÍA-VALLÉS, Fernando – COLOM, José Manuel: A class of well structured Petri nets for flexible manufacturing systems [Trieda štrukturovaných Petriho sietí pre pružné výrobné systémy]. In: *Application and Theory of Petri Nets 1998*, vol. 1420 of Lecture Notes on Computer Science, Ed. Jörg Desel and Manuel Silva, Springer-Verlag, 1998, pp. 64-83.

(Ezpeleta-Recalde, 2004) EZPELETA, Joaquín – RECALDE, Laura: A deadlock avoidance approach for non-sequential resource allocation systems [Prístup k vyhýbaniu sa uviaznutiu pre nesekvenčné systémy prideľovania prostriedkov]. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A: Systems and Humans, vol. 34, no. 1, January 2004, pp. 93-101.

(Ezpeleta et al., 2002) EZPELETA, Joaquín – TRICAS, Fernando – GARCÍA-VALLÉS, Fernando – COLOM, José Manuel: A Banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states [Bankárovo riešenie pre vyhýbanie sa uviaznutiu v PVS s pružným smerovaním a viac-prostriedkovými stavmi]. In: *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, August 2002, str. 621-625.

(Fanti et al., 1997) FANTI, Maria Pia – MAIONE, Bruno – MASCOLO, Saverio – TURCHIANO, Biagio: Event-based feedback control for deadlock avoidance in flexible production systems [Udalostné spätnoväzobné riadenie na vyhýbanie sa uviaznutiu v pružných výrobných systémoch]. *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, June 1997, USA, str. 347-363.

(Flodr, 1990) FLODR, František: *Dopravní provoz železnic. Technologie železničních stanic.* Alfa, Bratislava, 1990. ISBN 80-05-00598-9

(Garey-Johnson, 1979) GAREY, Michael R. – JOHNSON, David S.: Computers and Intractability: A Guide to the Theory of NP-Completeness [Počítače a neopracovateľnosť: Sprievodca k teórii NP-úplnosti.]. W. H. Freeman and Co., New York, NY, 1979. ISBN 0-7167-1044-7. Citované v publikácii (Reveliotis et al., 1997).

(Gold, 1978) GOLD, E. Mark: Deadlock prediction: Easy and difficult cases [Predpovedanie uviaznutia: jednoduché a zložité prípady]. In: *SIAM Journal of Computing*, 7: 320-336, 1978. Citované v publikácii (Reveliotis et al., 1997).

(Jensen, 1994) JENSEN, Kurt: An Introduction to the Theoretical Aspects of Coloured Petri Nets [Úvod do teoretických aspektov farbených Petriho sietí]. In: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): *A Decade of Concurrency*, Lecture Notes in Computer Science vol. 803, Springer-Verlag 1994, 230-272.
Dostupné na: < http://www.daimi.au.dk/~kjensen/papers_books/rex.pdf >

(Jensen, 1997/1) JENSEN, Kurt: A Brief Introduction to Coloured Petri Nets [Krátky úvod do farbených Petriho sietí]. In: E. Brinksma (ed.): *Tools and Algorithms for the Construction and Analysis of Systems*. Proceeding of the TACAS'97 Workshop, Enschede, The Netherlands 1997, Lecture Notes in Computer Science Vol. 1217, Springer-Verlag 1997, 203-208.
Dostupné na: < http://www.daimi.au.dk/~kjensen/papers_books/brief.pdf >

(Jensen, 1998) JENSEN, Kurt: An Introduction to the Practical Use of Coloured Petri Nets [Úvod do praktického použitia farbených Petriho sietí]. In: W. Reisig and G. Rozenberg (eds.): *Lectures on Petri Nets II: Applications*, Lecture Notes in Computer Science vol. 1492, Springer-Verlag 1998, 237-292.
Dostupné na: < http://www.daimi.au.dk/~kjensen/papers_books/use.pdf >

(Jensen, 1997/2) JENSEN, Kurt: Coloured Petri Nets. *Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts.* [Farbené Petriho siete. Základné pojmy, analytické metódy a praktické použitie. Prvý zväzok, Základné pojmy.] Monographs in Theoretical Computer Science, Springer-Verlag, 2nd corrected printing 1997. ISBN: 3-540-60943-1.

(Jensen, 1997/3) JENSEN, Kurt: Coloured Petri Nets. Basic *Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods.* [Farbené Petriho siete. Základné pojmy, analytické metódy a praktické použitie. Druhý zväzok, Analytické metódy.] Monographs in Theoretical Computer Science, Springer-Verlag, 2nd corrected printing 1997. ISBN: 3-540-58276-2.

(Jensen, 1997/4) JENSEN, Kurt: Coloured Petri Nets. Basic *Concepts, Analysis Methods and Practical Use. Volume 3, Practical Use.* [Farbené Petriho siete. Základné pojmy, analytické metódy a praktické použitie. Tretí zväzok, Praktické použitie.] Monographs in Theoretical Computer Science, Springer-Verlag, 1997. ISBN: 3-540-62867-3.

(Kavička et al., 2005) KAVIČKA, Antonín – KLIMA, Valent – ADAMKO, Norbert: Agentovo orientovaná simulácia dopravných uzlov. Žilinská univerzita v Žiline, 2005. ISBN 80-8070-477-5.

(Lang, 1999) LANG, Sheau-Dong: An extended Banker's algorithm for deadlock avoidance [Rozšírený algoritmus bankára pre vyhýbanie sa uviaznutiu]. *IEEE Transactions on software engineering*, vol. 25, no. 3, 1999, str. 428-432.

(Lawley-Reveliotis, 2001) LAWLEY, Mark A. – REVELIOTIS, Spyros A.: Deadlock avoidance for sequential resource allocation systems: hard and easy cases [Vyhýbanie sa uviaznutiu pre sekvenčné systémy prideľovania zdrojov: zložité a jednoduché prípady]. *The International Journal of Flexible Manufacturing Systems*, vol. 13, no. 4, str. 385-404, 2001.

(Lawley et al., 1998) LAWLEY, Mark A. – REVELIOTIS, Spyros A. – FERREIRA, Placid M.: The application and evaluation of Banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems [Aplikácia a vyhodnotenie algoritmu bankára pre alokáciu zásobníkového priestoru bez uviaznutia v pružných výrobných systémoch]. In: *The International Journal of Flexible Manufacturing Systems*, vol. 10, 1998, str. 73-100.

(Li-Wonham, 1993) LI, Yong – WONHAM, W. Murray: Control of Vector Discrete-Event Systems – Part I: The Base Model [Riadenie vektorových systémov diskrétnych udalostí – časť I: základný model]. In: *IEEE Transactions on Automatic Control*, vol. 38, 1993, no. 8, str. 1214-1227. ISSN: 0018-9286.

(Li-Wonham, 1994) LI, Yong – WONHAM, W. Murray: Control of Vector Discrete-Event Systems – Part II: Controller Synthesis [Riadenie vektorových systémov diskrétnych udalostí – časť II: syntéza ovládača]. In: *IEEE Transactions on Automatic Control*, vol. 39, 1994, no. 3, str. 512-531. ISSN: 0018-9286.

(Martincová-Grondžák, 2004) MARTINCOVÁ, Penka – GRONDŽÁK, Karol: *Operačné systémy*. Žilinská Univerzita, Žilina, 2004. ISBN 80-8070-242-X.

(Márton et al., 2004) MÁRTON, Peter – ŽARNAY, Michal – WENJIAN, Li: Development of computer simulation model of Mudanjiang railway junction and its results [Vývoj počítačového simulačného modelu železničného uzla Mudanjiang a jeho výsledky]. In: zborník medzinárodnej konferencie *Žel 2003*, 27.-28.5.2003, Žilina, 2003, 1. diel, str. 174-178. ISBN 80-7135-062-1.

(Murata, 1989) MURATA, Tadao: Petri Nets: Properties, Analysis and Applications [Petriho siete: vlastnosti, analýza a aplikácie]. In: *Proceedings of the IEEE*, vol. 77, no. 4, str. 541-580, April 1989.

(Palúch, online) PALÚCH, Stanislav: *Skripta z teorie grafov*. [online]. [citované 16. augusta 2006] Kapitola 5.3. Metódy časového plánovania. Kapitola 5.4. Klasická interpretácia metódy CPM a jej nedostatky. Str. 73-79. Súbory vo formáte PostScript. Dostupné na: <http://frcatel.fri.utc.sk/users/paluch/index.php>

(Park-Reveliotis, 2001) PARK, Jonghun – REVELIOTIS, Spyros A.: Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings [Vyhýbanie sa uviaznutiu v sekvenčných systémoch prideľovania prostriedkov s viacerými prideleniami prostriedkov a pružnými smerovaniami]. In: IEEE Transactions on Automatic Control, vol. 46, no. 10, 2001, str. 1572-1583.

(Peterson, 1981) PETERSON, James Lyle: Operating System Concepts. Addison-Wesley, 1981. Citované v publikácii (Reveliotis et al., 1997).

(Petri Nets World, online) *Petri Nets World: Online services for the International Petri nets community* [Svet Petriho sietí: Online služby pre medzinárodnú komunitu Petriho sietí]. [online]. [citované 25. augusta 2006]
Dostupné na: <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>.

(Reveliotis, 2000) REVELIOTIS, Spyros A.: Conflict resolution in AGV systems [Riešenie konfliktu v systémoch AGV]. IIE Transactions, vol. 32, no. 7, 2000, str. 647-659.

(Reveliotis, 2006) REVELIOTIS, Spyros A.: Implicit Siphon Control and its Role in the Liveness Enforcing Supervision of Sequential Resource Allocation Systems [Implicitné riadenie pomocou sifónu a jeho úloha v živosť uplatňujúcom dohliadacom programe sekvenčného systému prideľovania prostriedkov]. In: *IEEE Transactions on Systems, Man & Cybernetics* – Part A (pred vydaním). Dostupné aj na:
<http://www2.isye.gatech.edu/~spyros/publications/Implicit-Siphon.pdf>

(Reveliotis et al., 1997) REVELIOTIS, Spyros A.– LAWLEY, Mark A. – FERREIRA, Placid M.: Polynomial-complexity deadlock avoidance policies for sequential resource allocation systems [Stratégia vyhýbania sa uviaznutiu polynomiálnej zložitosti pre sekvenčné systémy prideľovania prostriedkov]. In: *IEEE Transactions on Automatic Control*, vol. 42, str. 1344-1357, 1997.
K tomu: PARK, Jonghun – REVELIOTIS, Spyros A.– LAWLEY, Mark A. – FERREIRA, Placid M.: A correction to the RUN DAP for conjunctive RAS presented in 'Polynomial complexity deadlock avoidance policies for sequential resource allocation systems' [Oprava RUN DAP pre konjunktívny RAS prezentovaný v článku 'Stratégia vyhýbania sa uviaznutiu polynomiálnej zložitosti pre sekvenčné systémy prideľovania prostriedkov']. In: *IEEE Transactions on Automatic Control*, vol. 46, str. 672, 2001.

(Reveliotis-Choi, 2006) REVELIOTIS, Spyros A. – CHOI, Jin Young: Designing reversibility-enforcing supervisors of polynomial complexity for bounded Petri nets through the theory of regions [Navrhovanie vratnosť uplatňujúcich dohliadacích programov polynomiálnej zložitosti pre obmedzené Petriho siete prostredníctvom teórie regiónov]. In: *27th International Conference on the Application and Theory of Petri Nets and Other Models of Concurrency (ATPN 2006)* , June 2006, Turku, Finland, LNCS 4024 (S. Donatelli and P. S. Thiagarajan, eds), Springer, 2006, str. 322-341.

(Silberschatz et al., 2002) SILBERSCHATZ, Abraham – GALVIN, Peter Baer – GAGNE, Greg: *Operating system concepts* [Koncepcie operačného systému]. 6. vydanie, John Wiley & Sons, Inc., New York, NY, USA, 2002. ISBN 0-471-41743-2.

(Tricas, 2003) TRICAS, Fernando: *Deadlock Analysis, Prevention and Avoidance in Sequential Resource Allocation Systems* [Analýza, prevencia a vyhýbanie sa uviaznutiu v sekvenčných systémoch prideľovania prostriedkov]. Ph.D. Thesis. Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, máj 2003.

(Tricas et al., 1999) TRICAS, Fernando – COLOM, José Manuel – EZPELETA, Joaquín: A solution to the problem of deadlocks in concurrent systems using Petri nets and integer linear programming. In: *Proc. of the 11th European Simulation Symposium*, Ed. G. Horton, D. Moller and U. Rude, Erlangen, Germany, October 1999, pp. 542-546.

(Tricas et al., 2000/1) TRICAS, Fernando – COLOM, José Manuel – EZPELETA, Joaquín: Some improvements to the Banker's algorithm based on the process structure [Niekoľko vylepšení algoritmu bankára založené na štruktúre procesu]. In: Proceedings of *IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, 2000, Vol. 3, str. 2853-2858. ISBN 0-7803-5886-4.

(Tricas et al., 2000/2) TRICAS, Fernando – GARCÍA-VALLÉS, Fernando – COLOM, José Manuel – EZPELETA, Joaquín: New methods for deadlock prevention and avoidance in concurrent systems [Nové metódy na prevenciu a vyhýbanie sa uviaznutiu v súbežných systémoch]. In: Proceedings of *VIII Jornadas de Concurrencia*, 2000, str. 97-110.

(Wonham, 2003) WONHAM, W. Murray: Supervisory control theory: models and methods [Teória kontrolného riadenia: modely a metódy]. In: Proc. *ATPN - Workshop on Discrete Event Systems Control, 24th International Conference on Application and Theory of Petri Nets* (ATPN 2003), Eindhoven, The Netherlands, str.1-14, June 2003.

(Wonham, online) WONHAM, W. Murray: *Supervisory control of discrete event systems* [Teória kontrolného riadenia diskrétnych udalostných systémov]. ECE 1636F/1637S 2006-07. [online] University of Toronto, 1997-2006, Posledná korektúra 1.7.2006 [citované 19. augusta 2006] Introduction. Str. vii-viii. Súbor vo formáte PDF. Dostupné na: <http://www.control.toronto.edu/cgi-bin/dldes.cgi>

(Žarnay, 2001) ŽARNAY, Michal: *Analýza metód riadenia v dopravných systémoch.* Písomná práca na dizertačnú skúšku, Žilinská univerzita v Žiline, Žilina, 2001.

(Žarnay, 2006) ŽARNAY, Michal: Coloured Petri net model of train handling in Marschalling yard [Model farbenej Petriho siete pre spracovanie vlaku v zriaďovacej stanici]. In: 14. medzinárodné sympózium *EURNEX - Žel 2006*, "Ku konkurencieschopným železničným systémom v Európe", 30. - 31.5.2006, Žilina, Zborník prednášok, 2. diel, EDIS – vydavateľstvo ŽU, Žilina, 2006, str. 168-173, ISBN 80-8070-550-X.

(Žarnay, 2003) ŽARNAY, Michal: Deadlock solving in transport system with methods from computer operating system [Riešenie uviaznutia v dopravnom systéme metódami operačných systémov]. In: *Komunikácie*, vedecké listy ŽU, EDIS – vydavateľstvo ŽU, Žilina, 4/2003, str. 74-77. ISSN 1335-4205.

(Žarnay, 2004/1) ŽARNAY, Michal: Human factor in decision-making in simulation model of transportation system and approaches to its modelling [Ľudský činiteľ v rozhodovaní v simulačnom modeli dopravného systému a prístupy k jeho modelovaniu]. In: *Komunikácie*, vedecké listy Žilinskej univerzity, EDIS – vydavateľstvo ŽU, Žilina, 1/2004, str. 50-53. ISSN 1335-4205.

(Žarnay, 2004/2) ŽARNAY, Michal: Use of Petri Net for Modelling of Traffic in Railway Stations [Použitie Petriho siete na modelovanie prevádzky v železničných staniciach]. In: zborník medzinárodnej konferencie *Infotrans 2004*, 4.-5.2.2004, Pardubice.

(Žarnay-Kavička, 1999) ŽARNAY, Michal – KAVIČKA, Antonín: Simulačný model prevádzky osobnej železničnej stanice. In: zborník medzinárodnej konferencie *Žel '99*, Žilina, 1999, str. 128-133. ISBN 80-7135-053-2.

(Žarnay et al., 2002) ŽARNAY, Michal – MÁRTON, Peter – DUPKALA, Michal: Vytvorenie a použitie simulačného modelu železničnej stanice Žilina-Teplička. In: zborník medzinárodnej konferencie *Žel 2002*, 28.-29.5.2002, Žilina, 2002, 2. diel, str. 113-118. ISBN 80-7135-059-1.

(Žarnay-Sadloň, 2006) ŽARNAY, Michal – SADLOŇ, Ľubomír: Prevod sieťového grafu technologického postupu spracovania vlaku na Petriho sieť. In: *Horizonty dopravy*, časopis pre vedu a výskum v doprave, Výskumný ústav dopravný, a. s., Žilina, 1/2006, str. 19-25. ISSN 1210-0978.

(Žarnay et al., 2006) ŽARNAY, Michal – SLÁDEČEK, Karol – CENEK, Petr : Modelling of marschalling yard technology with help of Petri net [Modelovanie technológie zriaďovacej stanice pomocou Petriho siete]. In: *Journal of Information, Control and Management Systems*, University of Žilina, Faculty of Management Science and Informatics, Vol. 4, No. 1, 2006, str. 49-62. ISSN 1336-1716.

8 Vysvetlenie použitých skratiek a symbolov

BA	Banker's Algorithm [Algoritmus bankára]
CPN	Coloured Petri Net [farbená Petriho sieť]
CPN ML	Coloured Petri Nets ML – jazyk ML pre farbené Petriho siete založený na štandardnom jazyku ML vo verzii New Jersey
DAP	Deadlock Avoidance Policy [stratégia vyhýbania sa uviaznutiu]
DES	Discrete-Event System [systém diskrétnych udalostí]
DS	dopravný systém
F^*	iterácia množiny F
FMS	Flexible Manufacturing System [pružný výrobný systém, PVS]
GS^*PR	model zovšeobecneného systému procesov s prostriedkami v Petriho sieti (sekvenčné aj nesekvenčné procesy)
MT-NO-RAS	Multiple Type – Non-Ordered – Resource Allocation System [viac-typový neusporiadaný systém prideľovania prostriedkov]
MT-PO-RAS	Multiple Type – Partially Ordered – Resource Allocation System [viac-typový čiastočne usporiadaný systém prideľovania prostriedkov]
MT-TO-RAS	Multiple Type – Totally Ordered – Resource Allocation System [viac-typový úplne usporiadaný systém prideľovania prostriedkov]
NP^2N	Non-sequential Process Petri Net [Petriho sieť nesekvenčného procesu]
NP-CPN	Coloured Petri Net of Non-sequential Process [farbená Petriho sieť nesekvenčného procesu]
NS-RAS	Non-sequential RAS [nesekvenčný systém prideľovania prostriedkov, resp. systém prideľovania prostriedkov s nesekvenčnými procesmi]
OS	operačný systém
P^2N	Process Petri net [procesná Petriho sieť]
PN	Petri Net [Petriho sieť]
PVS	pružný výrobný systém
RAS	Resource Allocation System [systém prideľovania prostriedkov]
S^*PR	Petriho sieť pre systém sekvenčných procesov s prostriedkami pripúšťajúci existenciu neusporiadaných procesov (MT-NO-RAS)

S^3PGR^2	System of Simple Sequential Processes with General Resource Requirements [systém jednoduchých sekvenčných procesov so všeobecnými požiadavkami na prostriedky] – v podstate S^4PR
S^4PR	Petriho sieť pre úplne alebo čiastočne usporiadaný systém sekvenčných procesov s prostriedkami (MT-PO-RAS)
SG	sieťový graf
SNP-CPN	Coloured Petri Net of System with Non-sequential Processes [farbená Petriho sieť systému s nesekvenčnými procesmi]
ST-NO-RAS	Single Type – Non-Ordered – Resource Allocation System [jedno-typový neusporiadaný systém prideľovania prostriedkov]
ST-PO-RAS	Single Type – Partially Ordered – Resource Allocation System [jedno-typový čiastočne usporiadaný systém prideľovania prostriedkov]
ST-TO-RAS	Single Type – Totally Ordered – Resource Allocation System [jedno-typový úplne usporiadaný systém prideľovania prostriedkov]
SU-NO-RAS	Single Unit – Non-Ordered – Resource Allocation System [jedno-jednotkový neusporiadaný systém prideľovania prostriedkov]
SU-PO-RAS	Single Unit – Partially Ordered – Resource Allocation System [jedno-jednotkový čiastočne usporiadaný systém prideľovania prostriedkov]
SU-TO-RAS	Single Unit – Totally Ordered – Resource Allocation System [jedno-jednotkový úplne usporiadaný systém prideľovania prostriedkov]
VDES	Vector Discrete-Event System [vektorový systém diskrétnych udalostí]

9 Prílohy

A Doplnenie k Petriho sieťam

B Riešenie uviaznutia pomocou sifónu

C Algoritmus bankára a jeho vylepšenia

D Demonštračný model a jeho analýza

A Doplnenie k Petriho sieťam

Cieľom tejto kapitoly je zachytiť všetky pojmy a detaily z oblasti Petriho sietí (PN), ktoré sa dotýkajú tejto práce. Člení sa na dve podkapitoly venujúce sa základnej triede Petriho sietí a jednému jej rozšíreniu – farbeným Petriho sieťam.

Prvá podkapitola opisuje postupne základy, analytické vlastnosti, niektoré podtriedy a rozšírenia základnej triedy Petriho sietí. Druhá podkapitola sa postupne venuje definícii farbenej Petriho siete (CPN), opisu hierarchickej a časovanej CPN, jej analýze a uzaviera ju stručný opis nástroja CPN Tools, z ktorého pochádzajú obrázky v tejto práci.

A.1 P/T Petriho siete

Výber zo základných pojmov z oblasti Petriho sietí uvádzam v matematickej forme podľa (Češka, 1994) a čiastočne podľa (Park-Reveliotis, 2001) a (Murata, 1989). Pre ďalšie podrobnosti možno tiež nahliadnuť do (Petri Nets World, online).

A.1.1 Definícia pojmov

Definícia 9-1

Trojicu $\mathcal{N} = (P, T, F)$ nazývame *sieťou,* ak

(1) P a T sú disjunktívne množiny a

(2) $F \subseteq (P \times T) \cup (T \times P)$ je binárna relácia.

Množina P sa nazýva *množinou miest* (places) siete $\mathcal{N}$, množina P *množinou prechodov* (transitions) siete $\mathcal{N}$ a relácia F *tokovou reláciou* (flow relation) siete $\mathcal{N}$.

■

Grafom siete nazývame bipartitný orientovaný graf, ktorý vznikne grafovou reprezentáciou relácie F. Množina $P \cup T$ je množinou vrcholov grafu siete. Miesta sa obvykle kreslia v tvare kružníc, prechody v tvare obdĺžnikov alebo úsečiek. Príklad grafu Petriho siete možno nájsť na Obr. 9.1.

Definícia 9-2

Nech $\mathcal{N} = (P, T, F)$ je sieť.

(1) Pre všetky $x \in (P \cup T)$

$^\bullet x = \{y \mid yFx\}$ sa nazýva *vstupnou množinou* (preset) *prvku*.

$x^\bullet = \{y \mid xFy\}$ sa nazýva *výstupnou množinou* (postset) *prvku x*.

Pre $X \subseteq (P \cup T)$ je

$${}^{\bullet}X = \bigcup_{x \in X} {}^{\bullet}x \text{ a } X^{\bullet} = \bigcup_{x \in X} x^{\bullet}$$

Zrejme pre každé $x, y \in (P \cup T)$ platí: $x \in {}^{\bullet}y \Leftrightarrow y \in x^{\bullet}$

(2) Usporiadaná dvojica $\langle p,t \rangle \in P \times T$, sa nazýva *vlastný cyklus* (self-loop), ak $pFt \wedge tFp$. Ak $\mathcal{N}$ neobsahuje vlastný cyklus, potom sa nazýva *čistou sieťou* (pure net).

(3) Prvok $x \in P \cup T$ sa nazýva *izolovaný*, ak ${}^{\bullet}x \cup x^{\bullet} = \emptyset$.

■

Definícia 9-3

Šesticu $\mathcal{N}$ = (P, T, F, W, K, M_0) nazývame *P/T Petriho sieťou* (Place/Transition Petri Net), ak

(1) (P, T, F) je konečná sieť

(2) $W : F \to \mathbf{Z}^+$ je ohodnotenie hrán grafu siete určujúce váhu každej hrany

(3) $K : P \to \mathbf{Z}^+ \cup \{\infty\}$ je zobrazenie špecifikujúce kapacitu (aj neobmedzenú) každého miesta

(4) $M_0 : P \to \mathbf{Z}^+ \cup \{\infty\}$ je *počiatočné značenie* miest siete, rešpektujúce kapacitu miest,
t. j. $M_0(p) \leq K(p)$ pre všetky $p \in P$.

■

V ďalšom texte tejto časti sa bude Petriho sieťou rozumieť P/T Petriho sieť.

Definícia 9-4

Petriho sieť $\mathcal{N}$ = (P, T, F, W, K, M_0) sa nazýva *obyčajná* (ordinary), ak $W : F \to \{1\}$.

■

Definícia 9-5

Nech P je množina miest a T množina prechodov Petriho siete. Usporiadaná množina $X = \langle x_1, \ldots, x_n \rangle \subseteq P \cup T$ sa nazýva *cesta* (path) práve vtedy, keď $x_{i+1} \in x_i^{\bullet}, i = 1, \ldots, n-1$. Cesta sa nazýva *cyklus* (circuit) práve vtedy, keď $x_1 \equiv x_n$.

■

Definícia 9-6

Nech $\mathcal{N}$ = (P, T, F, W, K, M_0) je Petriho sieť.

(1) Zobrazenie $M : P \to \mathbf{Z}^+ \cup \{\infty\}$ sa nazýva *značenie* (marking) Petriho siete $\mathcal{N}$, ak $\forall p \in P : M(p) \leq K(p)$

Nech M je značenie Petriho siete $\mathcal{N}$.

(2) Prechod $t \in T$ je vykonateľný (enabled) pri značení M, stručnejšie *M-vykonateľný* (M-enabled), ak
$\forall p \in {}^\bullet t : M(p) \geq W(p,t)$
$\forall p \in t^\bullet : M(p) \leq K(p) - W(t,p)$

(3) Ak je prechod $t \in T$ vykonateľný pri značení M, potom jeho vykonaním získame *následné značenie M'* k značeniu M, ktoré je definované takto:

$$M'(p) = \begin{cases} M(p) - W(p,t), & \text{ak } p \in {}^\bullet t \setminus t^\bullet \\ M(p) + W(t,p), & \text{ak } p \in t^\bullet \setminus {}^\bullet t \\ M(p) - W(p,t) + W(t,p), & \text{ak } p \in {}^\bullet t \cap t^\bullet \\ M(p), & \text{inak} \end{cases}$$

Vykonanie prechodu t (transition firing) zo značenia M do značenia M' symbolicky zapisujeme: $M [t \rangle M'$

(4) Označme symbolom $[\mathbf{M}\rangle$ najmenšiu množinu rôznych značení siete $\mathcal{N}$ takú, že platí
(a) $M \in [\mathbf{M}\rangle$
(b) ak je $M_1 \in [\mathbf{M}\rangle$ a pre nejaké $t \in T$ platí $M_1 [t \rangle M_2$, tak potom $M_2 \in [\mathbf{M}\rangle$

Množina $[\mathbf{M}\rangle$ sa nazýva *množinou dosiahnuteľných značení* (reachability set) *zo značenia M*; množina $[M_0\rangle$ dosiahnuteľných značení z počiatočného značenia sa nazýva *množinou dosiahnuteľných značení siete* $\mathcal{N}$.

■

Konvencia

V grafickej reprezentácii Petriho siete sa zvyčajne hrana f označuje ohodnotením $W(f)$ len v prípade, keď $W(f) > 1$. Kapacita miesta sa označuje príslušnou hodnotou $K(p)$; v prípade, že je nekonečná, hodnota sa neuvádza. Hodnota značenia miesta p sa zvyčajne vykresľuje počtom $M(p)$ čiernych bodiek vo vnútri miesta p; ak je zobrazenie veľkého počtu bodiek zložité, tak príslušným číslom alebo symbolom ∞.

Ak je $P = \{p_1, \ldots, p_n\}$ množina miest Petriho siete $\mathcal{N}$ a M nejaké značenie Petriho siete $\mathcal{N}$, potom sa toto značenie zvyčajne zapisuje vektorom $(M(p_1), M(p_2), \ldots, M(p_n))$.

■

Definícia 9-7

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť a $[M_0\rangle$ jej množina dosiahnuteľných značení. *Prechodovou funkciou* Petriho siete $\mathcal{N}$ nazveme (parciálnu) funkciu δ:

$$\delta : [M_0\rangle \times T \rightarrow [M_0\rangle,$$

pre ktorú

$$\forall t \in T : \forall M, M' \in [M_0\rangle : \delta(M,t) = M' \Leftrightarrow M[t\rangle M'$$

Prechodová funkcia δ môže byť zovšeobecnená na postupnosti prechodov:

$$\delta : [M_0\rangle \times T^{**} \rightarrow [M_0\rangle$$

takto:

$$\delta(M, t\tau) = \delta(\delta(M,t),\tau), \quad \tau \in T^{**}$$

$$\delta(M, e) = M \text{ , } e \text{ je prázdny symbol}$$

Postupnosť $\tau \in T^{**}$ nazveme *výpočtovou postupnosťou* Petriho siete $\mathcal{N}$, ak je pre ňu definovaná prechodová funkcia $\delta(M_0, \tau)$. Množina všetkých výpočtových postupností charakterizuje chovanie Petriho siete.

■

Definícia 9-8

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť.

(1) Pre prechod $t \in T$ nech je definovaný vektor $\underline{\mathbf{t}} : P \rightarrow \mathbf{Z}$ takto:

$$\underline{\mathbf{t}}(p) = \begin{cases} W(t,p), & \text{ak } p \in t^{\bullet} \setminus {}^{\bullet}t \\ -W(p,t), & \text{ak } p \in {}^{\bullet}t \setminus t^{\bullet} \\ W(t,p) - W(p,t), & \text{ak } p \in {}^{\bullet}t \cap t^{\bullet} \\ 0, & \text{inak} \end{cases}$$

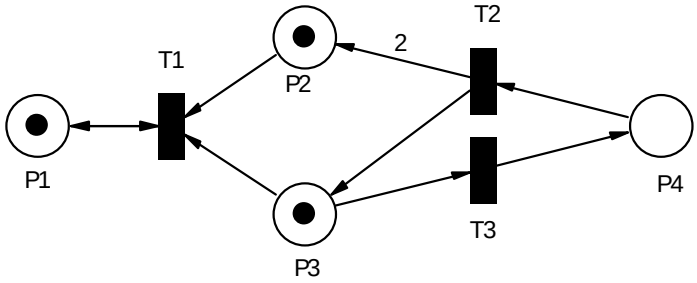
(2) Matica Petriho siete $\underline{\mathbf{N}}$ je definovaná ako matica
$\underline{\mathbf{N}} : P \times T \rightarrow \mathbf{Z}$, kde $\underline{\mathbf{N}}(p,t) = \underline{\mathbf{t}}(p)$

■

Definícia 9-9

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. *Tokovou maticou* (flow matrix) Petriho siete $\mathcal{N}$ sa nazýva matica $\underline{\mathbf{F}}$, $\underline{\mathbf{F}} : P \times T \rightarrow \mathbf{N} \times \mathbf{N}$,

$$\underline{\mathbf{F}}(p,t) = \langle w(p,t), w(t,p)\rangle, \text{ kde } w(x,y) = \begin{cases} W(x,y), & \text{ak } \langle x, y\rangle \in F \\ 0, & \text{inak} \end{cases}$$



$\underline{\mathbf{F}}$ =

	T1	T2	T3
P1	(1, 1)	(0, 0)	(0, 0)
P2	(1, 0)	(0, 2)	(0, 0)
P3	(1, 0)	(0, 1)	(1, 0)
P4	(0, 0)	(1, 0)	(0, 1)

$\underline{\mathbf{N}}$ =

	T1	T2	T3
P1	0	0	0
P2	-1	2	0
P3	-1	1	-1
P4	0	-1	1

Obr. 9.1 Petriho sieť a jej matice.

Toková matica je maticovou reprezentáciou váženej tokovej relácie *F*. Vzťah medzi maticou $\underline{\mathbf{N}}$ a $\underline{\mathbf{F}}$ je zrejmý:

$$\forall p \in P, \forall t \in T : \underline{\mathbf{N}}(p,t) = w(t,p) - w(p,t)$$

Matica Petriho siete $\underline{\mathbf{N}}$ sa pre odlíšenie od matice $\underline{\mathbf{F}}$ nazýva niekedy *maticou zmien* (change matrix) Petriho siete $\mathcal{N}$.

■

Poznámka

V niektorých anglicky písaných publikáciách, napr. v publikácii (Park-Reveliotis, 2001) sa matica zmien pre čistú Petriho sieť označuje pojmom *toková matica* (flow matrix) – v rozpore s definíciou tokovej matice v publikácii (Češka, 1994), ktorá je tam však zavedená pre všeobecnú Petriho sieť a ktorú som prebral aj do tejto práce.

■

Príklad Petriho siete a jej matíc je na Obr. 9.1.

A.1.2 Analytické vlastnosti Petriho siete

Vlastnosti Petriho sietí sa používajú na analýzu správania modelovaného systému. Pre potreby tejto práce sú dôležité najmä bezpečnosť, obmedzenosť, konzervatívnosť, živosť, sifón, pasca, p- a t-invarianty, resp. poltoky.

Definícia 9-10

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. Miesto $p \in P$ sa nazýva *bezpečné* (safe), ak platí:

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

Ak je každé miesto Petriho siete $\mathcal{N}$ bezpečné, potom sa $\mathcal{N}$ nazýva *bezpečnou Petriho sieťou* (safe Petri net).

■

Definícia 9-11

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. Miesto $p \in P$ sa nazýva *k-obmedzené* (k-bounded), ak platí:

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

Miesto *p* sa nazýva *obmedzené*, ak je *k*-obmedzené pre určité $\exists k \in \mathrm{N}$. Ak je každé miesto Petriho siete $\mathcal{N}$ obmedzené, potom $\mathcal{N}$ sa nazýva *obmedzenou Petriho sieťou* (bounded Petri net). Ak je Petriho sieť $\mathcal{N}$ obmedzená pre ľubovoľné počiatočné značenia M_0, nazýva sa *štrukturálne obmedzená* (structurally bounded) Petriho sieť.

■

Bezpečnosť Petriho siete je špeciálnym prípadom obmedzenosti, kde k = 1.

Definícia 9-12

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. $\mathcal{N}$ je *striktne konzervatívna* (strictly conservative), ak platí:

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

■

Definícia 9-13

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. $\mathcal{N}$ je *striktne konzervatívna vzhľadom k váhovému vektoru* $\underline{\mathbf{v}}$, ak platí:

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

Petriho sieť $\mathcal{N}$ nazveme *konzervatívnou*, ak je konzervatívna vzhľadom k váhovému vektoru $\underline{\mathbf{v}}$ **(** $\underline{\mathbf{v}} > 0 \Leftrightarrow \forall p \in P : \sum_{p \in P} v(p) > 0$ **)**.

■

Definícia 9-14

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť.

(1) Prechod $t \in T$ sa nazýva *živý* (live), ak pre $\forall M \in [M_0\rangle$ existuje značenie $M' \in [M\rangle$ také, že prechod *t* je M'-vykonateľný.

(2) Prechod $t \in T$ sa nazýva *mŕtvy* (dead) pri značení $M \in [M_0\rangle$, ak neexistuje značenie $M' \in [M\rangle$ také, že prechod *t* je M'-vykonateľný.

(3) Značenie $M \in [M_0\rangle$ sa nazýva (úplným) *uviaznutím* (deadlock), ak všetky prechody $t \in T$ sú mŕtve.

(4) Petriho sieť $\mathcal{N}$ sa nazýva *kvázi-živá* (quasi-live), ak pre všetky prechody $t \in T$ existuje $M \in [M_0\rangle$ také, že prechod *t* je *M*-vykonateľný.

(5) Petriho sieť $\mathcal{N}$ sa nazýva *slabo živá* (weakly live), ak pre všetky značenia $M \in [M_0\rangle$ existuje prechod $t \in T$ taký, že prechod *t* je *M*-vykonateľný.

(6) Petriho sieť $\mathcal{N}$ sa nazýva *živá* (live), ak všetky prechody $t \in T$ sú živé.

■

Definícia 9-15

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. Množina miest $S \subseteq P$ taká, že ${}^\bullet S \subseteq S^\bullet$, sa nazýva *sifón* (siphon) a množina miest $R \subseteq P$ taká, že $R^\bullet \subseteq {}^\bullet R$, sa nazýva *pasca* (trap).

(1) Sifón *S* (resp. pasca *R*) sa nazýva *minimálny* (resp. *minimálna*), ak neexistuje sifón S' taký, že $S' \subset S$ (resp. pasca R' taká, že $R' \subset R$).

(2) Sifón *S* (resp. pasca *R*) sa nazýva *prázdny* (resp. *prázdna*) pri značení *M*, ak $M(S) \equiv \sum_{p \in S} M(p) = 0$ (resp. $M(R) \equiv \sum_{p \in R} M(p) = 0$).

■

Sifón v Petriho sieti, v publikácii (Češka, 1994) nazývaný deadlock, vďaka svojej štruktúre funguje takým spôsobom, že keď sa raz vyprázdni, tak až do konca fungovania systému zostane prázdnym. Pasca funguje opačne: keď sa v nej zachytí aspoň jedna značka z okolia, pasca sa už až do konca nevyprázdni.

Definícia 9-16

Nech $\mathcal{N} = (P, T, F, W, K, M_0)$ je Petriho sieť. Vektor miest $i: P \rightarrow \mathbf{Z}$ nazývame *p-invariantom* Petriho siete $\mathcal{N}$, ak platí $\underline{\mathbf{N}}^{\mathrm{T}} . i = \mathbf{0}$ ($\underline{\mathbf{N}}^{\mathrm{T}}$ je transponovaná matica Petriho

siete $\mathcal{N}$). Ak $i(p) \in \{0,1\}$ pre všetky $p \in P$, potom sa *i* nazýva *binárnym p-invariantom* siete $\mathcal{N}$.

■

Definícia 9-17

Nech $\mathcal{N}$ = (*P*, *T*, *F*, *W*, *K*, M_0) je Petriho sieť. Vektor prechodov *i*: $T \rightarrow$ **Z** nazývame *t-invariantom* Petriho siete $\mathcal{N}$, ak platí $\underline{\mathbf{N}}.i = \mathbf{0}$.

■

Viac o vlastnostiach, výpočte a použití invariantov sa čitateľ dozvie napríklad v publikácii (Češka, 1994).

Definícia 9-18

P-invariant (resp. t-invariant) Petriho siete, ktorý má všetky vektorové zložky nezáporné, sa nazýva *p-poltok* (resp. *t-poltok*) (anglicky p-semiflow, resp. t-semiflow).

P-poltok *y* (resp. t-poltok *x*) sa nazýva *minimálny*, ak $\neg\exists$ p-poltok y' (resp. t-poltok x'), taký, že $\|y'\| \subset \|y\|$ (resp. $\|x'\| \subset \|x\|$), $\|y\| = \{p \in P \mid y(p) > 0\}$ (resp. $\|x\| = \{t \in T \mid x(t) > 0\}$).

■

Poltok sa tiež podľa (Tricas, 2003, str. 194) nazýva *prirodzeným prvkom jadra lineárneho zobrazenia daného maticou* $\underline{\boldsymbol{N}}$ (natural annuller). *Celočíselným prvkom* (integer annuller) je *tok*. P-(pol)toky, resp. t-(pol)toky sú ľavými resp. pravými prvkami jadra lineárneho zobrazenia daného maticou $\underline{\mathbf{N}}$.

Definícia 9-19

Petriho sieť $\mathcal{N}$ je *(čiastočne) konzistentná* ((partially) consistent) práve vtedy, keď jej t-invariant obsahuje kladné (nezáporné) zložky.

■

V (čiastočne) konzistentnej Petriho sieti sa vo vykonávacej postupnosti prechodov σ začínajúcej a končiacej v počiatočnom značení M_0 vyskytujú všetky (niektoré) prechody aspoň jeden raz (Murata, 1989).

A.1.3 Podtriedy Petriho siete

Podtriedy Petriho siete predstavujú zúženie základnej triedy Petriho siete (P/T Petriho siete). Vznikajú spravidla za účelom zvýšenia rozhodovacej mocnosti. V tejto kapitole uvádzam podtriedy so širšou platnosťou na základe publikácie (Češka, 1994).

Definícia 9-20

Petriho sieť $\mathcal{N} = (P, T, F, W, M_0)$ sa nazýva *stavový stroj* (state machine), ak spĺňa podmienku:

$$\forall t \in T : \left(\left| {}^\bullet t \right| = \left| t^\bullet \right| = 1 \wedge W\left({}^\bullet t, t \right) = W\left(t, t^\bullet \right) = 1 \right)$$

■

Stavové stroje majú výbornú rozhodovaciu mocnosť. Sú striktne konzervatívne, a teda ich stavový priestor (strom dosiahnuteľných značení) je konečný a umožňuje rozhodnúť všetky otázky analýzy. Ich modelovacia mocnosť je však rovnaká ako mocnosť konečných automatov, a tak ich samostatné používanie je obmedzené.

Definícia 9-21

Petriho sieť $\mathcal{N} = (P, T, F, W, M_0)$ sa nazýva *značený graf* (marked graph), ak spĺňa podmienky:

(1) $\forall t_1, t_2 \in T : t_1 F^* t_2$ ($\mathcal{N}$ je silno súvislý) [F^* symbolizuje iteráciu množiny F]

(2) $\forall p \in P : \left(\left| {}^\bullet p \right| = \left| p^\bullet \right| = 1 \wedge W\left({}^\bullet p, p \right) = W\left(p, p^\bullet \right) = 1 \right)$

■

Trieda značených grafov môže byť z hľadiska grafov chápaná ako duálna k triede stavových strojov. Modelovacia mocnosť oboch tried sa líši. Značené grafy, na rozdiel od stavových strojov, môžu pri vykonávaní vytvárať nové značky alebo rušiť už vytvorené značky, čo je podmienka pre modelovanie paralelizmu a synchronizácie (čakanie na vytvorenú značku). Nemôžu však modelovať konflikty (nedeterministické javy) alebo rozhodnutia závislé na údajoch.

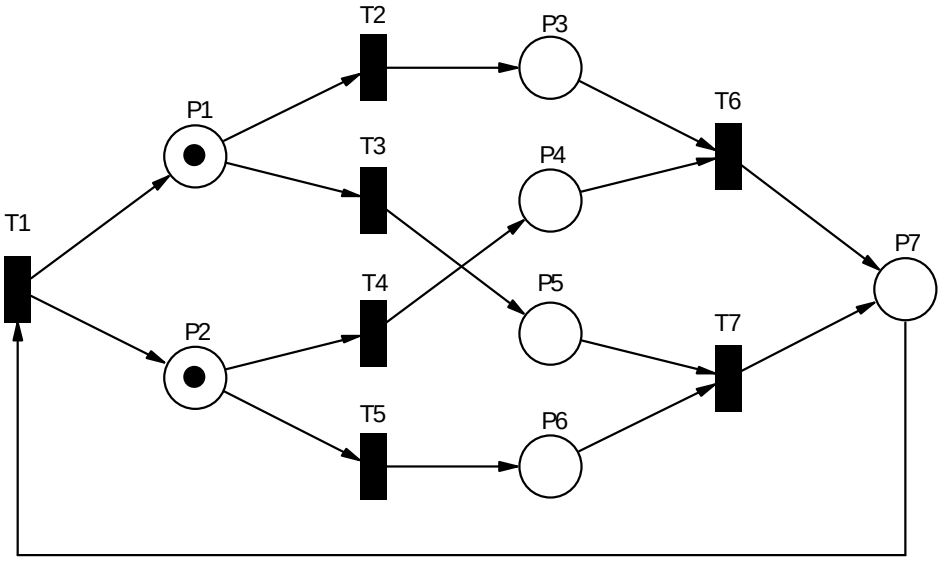
Definícia 9-22

Petriho sieť $\mathcal{N} = (P, T, F, W, M_0)$ sa nazýva *Petriho sieť s voľným výberom* (free-choice Petri net), ak spĺňa podmienky:

(1) $\forall (p,t) \in F \cap (P \times T) : \left(p^\bullet = \{t\} \vee {}^\bullet t = \{p\} \right)$

(2) $\forall z \in F : W(z) = 1$

■



Obr. 9.2 Petriho sieť s voľným výberom.

Splnenie podmienok v Petriho sieti s voľným výberom (Obr. 9.2) znamená, že možno riešenie konfliktov určitým spôsobom riadiť. Ak je nejaké miesto vstupným miestom pre niekoľko prechodov (je možným zdrojom konfliktu), tak všetky tieto prechody majú iba jedno vstupné miesto. Preto sú pri značení *M* vykonateľné buď všetky alebo ani jeden z nich. To umožňuje slobodný výber prechodu z množiny konfliktných prechodov; prítomnosť značiek v iných miestach netreba brať do úvahy.

A.1.4 Rozšírenia Petriho siete

Ďalšie triedy Petriho siete predstavujú rozšírenie základnej triedy Petriho siete (P/T Petriho siete). Vznikajú spravidla za účelom zvýšenia modelovacej mocnosti.

Časové Petriho siete (Time Petri nets) a ich špeciálny prípad *časované Petriho siete* (Timed Petri nets) sú rozšírením základnej triedy o možnosť popisu časových vzťahov medzi operáciami v modelovanom systéme. K nim príbuzné sú *stochastické Petriho siete*, ktoré ohodnocujú prechody siete stochastickým atribútom, ktorý reprezentuje dĺžku trvania operácie.

Petriho siete vyššej úrovne (higher-level Petri nets) umožňujú kategorizovať a individualizovať jednotlivé značky značenia miest a stanoviť podmienky prechodov rešpektujúce atribúty značiek. Medzi ne patria *farbené Petriho siete* (Coloured Petri nets, CPN) a ich špeciálny prípad – *hierarchické farbené Petriho siete*, ktoré poskytujú navyše aj možnosť explicitne štrukturovať graf PN. Ich výhodou je značná redukcia rozsahu siete v porovnaní so základnou triedou PN, a to tak, že časť miest

a prípadne aj prechodov je nahradená ich menším počtom, pričom sa v nich pohybujú farbené značky zachovávajúce význam nahradených miest a prechodov. Redukcia však závisí od konkrétnej štruktúry siete a najmä od faktu, či sa v sieti vyskytujú opakujúce sa štrukturálne vzorky (podsiete obsahujúce miesta, prechody a hrany v rovnakej štruktúre) a koľkokrát sa opakujú.

Z rozšírení sa ďalej venujem farbeným Petriho sieťam, ktoré sa používajú v tejto práci.

A.2 Farbené Petriho siete

Nasledujúce definície sú prevzaté a upravené podľa (Jensen, 1997/2, str. 66-78). Ich vysvetlenie demonštrujem na príklade CPN jednoduchého dopravného systému, implementovanom v nástroji CPN Tools.

A.2.1 Definícia pojmov

Definícia 9-23

Multi-množinou m nad neprázdnou množinou prvkov *S* nazývame funkciu $m \in S \rightarrow \mathbf{N}$. Nezáporné číslo $m(s) \in \mathbf{N}$ je *počet výskytov* prvku *s* v multi-množine *m*.

Multi-množina sa zvyčajne reprezentuje ako formálna suma:

$$\sum_{s \in S} m(s)`s$$

Symbolom S_{MS} označíme množinu všetkých multi-množín nad množinou *S*. Nezáporné celé čísla $\{m(s) \mid s \in S\}$ sa nazývajú *koeficienty* multi-množiny *m*, a $m(s)$ sa volá *koeficient* prvku *s*. Hovoríme, že prvok $s \in S$ patrí do multi-množiny *m* práve vtedy, keď $m(s) \neq 0$ a potom píšeme $s \in m$.

■

Tak, ako existuje prázdna množina, hovoríme aj o prázdnej multi-množine a značíme ju $\varnothing$.

Medzi podmnožinami množiny *S* a tými multi-množinami, ktoré majú všetky koeficienty nula-jednotkové, existuje vzťah 1:1. Preto sa používa množina $A \subseteq S$ na označenie tých multi-množín, ktoré obsahujú jeden výskyt každého prvku z množiny *A*. Analogicky sa prvok $s \in S$ používa na označenie multi-množiny 1`*s*.

Pre multi-množiny existuje niekoľko štandardných operácií, ktoré sú až na operáciu $|m|$ všetky odvodené od štandardných operácií pre funkcie.

Definícia 9-24

Pre všetky multi-množiny $m, m_1, m_2 \in S_{MS}$ a všetky čísla $n \in \mathbf{N}$ platí:

(1) $m_1 + m_2 = \sum_{s \in S} (m_1(s) + m_2(s))`s$ (súčet)

(2) $n * m = \sum_{s \in S} (n * m(s))`s$ (násobenie skalárom)

(3) $m_1 \neq m_2 = \exists s \in S : m_1(s) \neq m_2(s)$ (porovnanie)

$m_1 \leq m_2 = \forall s \in S : m_1(s) \leq m_2(s)$ (≥ a = sú definované analogicky ako ≤)

(4) $|m| = \sum_{s \in S} m(s)$ (veľkosť)

(5) ak $m_1 \leq m_2$, tak

$m_2 - m_1 = \sum_{s \in S} (m_2(s) - m_1(s))`s$ (rozdiel)

■

Keď $|m| = \infty$, hovoríme, že *m* je *nekonečná*. Inak *m* je *konečná*.

V ďalších definíciách sa vyskytujú nasledovné výrazy CPN, ktoré v konkrétnej implementácii farbenej Petriho siete môžu mať definovanú syntax a sémantiku inak.

- *Prvky typu T* – množina všetkých prvkov v *T* sa označuje menom typu *T*.
- *Typ premennej v* sa označuje ako *Type*(*v*).
- *Typ výrazu expr* sa označuje ako *Type*(*expr*).
- *Množina premenných* vo výraze *expr* sa označuje ako *Var*(*expr*).
- *Väzba množiny premenných V* spája s každou premennou $v \in V$ prvok $b(v) \in Type(v)$.
- *Hodnota* získaná vyhodnotením výrazu *expr* vo väzbe *b* sa označuje ako *expr*<*b*>. Požaduje sa, aby *Var*(*expr*) bola podmnožinou premenných väzby *b*. Vyhodnotenie sa vykonáva nahradením každej premennej $v \in Var(expr)$ hodnotou $b(v) \in Type(v)$ určenou danou väzbou.

Výraz bez premenných sa nazýva *zatvorený výraz*. Môže byť vyhodnotený vo všetkých väzbách, a všetky vyhodnotenia dajú rovnakú hodnotu, ktorá sa často značí samotným výrazom. To znamená, že sa píše jednoducho *expr* namiesto dôkladnejšieho *expr*<*b*>.

B označuje typ boolean (s prvkami {false, true} a štandardnými operáciami).

Ešte rozšírime notáciu pre typ premennej *Type*(*v*) na typ premenných z množiny *A*:
Type(*A*) = {*Type*(*v*) | $v \in A$}, kde *A* je množina premenných.

Definícia 9-25

Farbenou Petriho sieťou (CPN) nazývame deväticu $CPN = (\sum, P, T, A, N, C, G, E, I)$, v ktorej:

(1) $\sum$ je konečná množina neprázdnych typov, tiež nazývaných množiny farieb.

(2) P je konečná množina miest.

(3) T je konečná množina prechodov.

(4) A je konečná množina hrán, pre ktorú platí:

$$P \cap T = P \cap A = T \cap A = \varnothing$$

(5) $N: A \rightarrow (P \times T) \cup (T \times P)$ je funkcia vrcholov.

(6) $C: P \rightarrow \sum$ je funkcia farieb.

(7) G je funkcia strážnych podmienok, pre ktorú platí:
$G: T \rightarrow \{\forall t \in T : (Type(G(t)) = \mathbf{B} \wedge Type(Var(G(t))) \subseteq \sum)\}$

(8) E je funkcia hranových výrazov, pre ktorú platí:
$E: A \rightarrow \{\forall a \in A : (Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \sum)\}$,
pričom p je miestom $N(a)$.

(9) I je inicializačná funkcia, pre ktorú platí:
$I: P \rightarrow \{\forall p \in P : Type(I(p)) = C(p)_{MS}\}$

■

(1) Množina typov $\sum$ určuje hodnoty údajov, operácie a funkcie, ktoré možno použiť v sieťových výrazoch (napr. hranové výrazy, strážne podmienky a inicializačné výrazy). Predpokladáme, že každý typ obsahuje aspoň jeden prvok. Prvky môžu byť jednoduché alebo štruktúrované údajové objekty (napr. n-tice, záznamy, zoznamy).

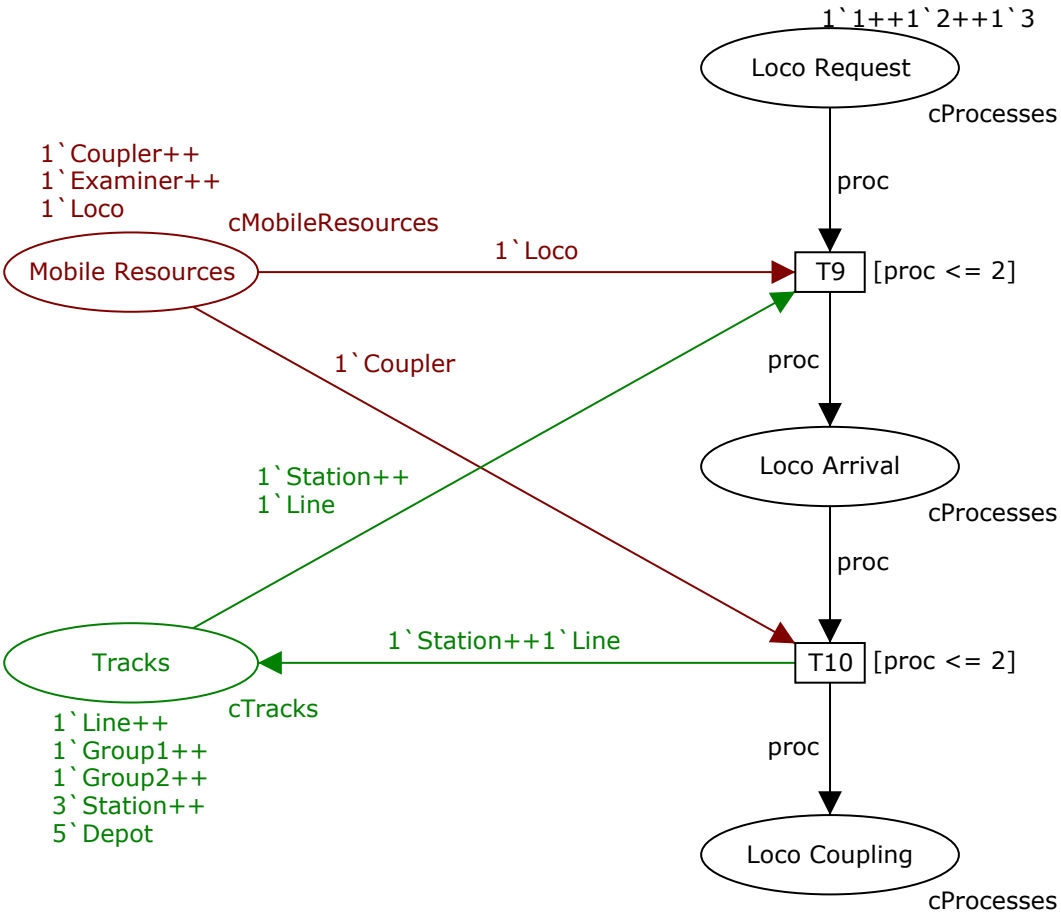
(2) + (3) + (4) Miesta, prechody a hrany sú opísané troma množinami P, T a A, ktoré musia byť konečné a po dvojiciach disjunktívne. Požiadavkou konečnosti množín sa vyhneme viacerým technickým problémom ako napríklad možnosť mať nekonečný počet hrán medzi dvoma vrcholmi.

V príklade podľa Obr. 9.3 tvoria množinu $\sum$ 3 typy: 2 vymenované typy *cTracks* a *cMobileResources* a celočíselný typ *cProcesses*. Množina miest P obsahuje 5 prvkov, a to miesta s pomenovaniami: *Loco Request*, *Loco Arrival*, *Loco Coupling*, *Mobile Resources* a *Tracks*. Množina prechodov T je dvojprvková: prechody *T9* a *T10*. Množina A zahŕňa 8 hrán medzi vrcholmi (viď Obr. 9.3).

(5) Funkcia vrcholov N mapuje každú hranu na usporiadanú dvojicu, ktorej prvý prvok je zdrojový a druhý je cieľový vrchol hrany. Obidva vrcholy musia byť rôzneho druhu (t. j. jeden musí byť miestom, druhý prechodom). Táto definícia tiež umožňuje mať viac hrán medzi vrcholmi rovnakej usporiadanej dvojice, čo zvyšuje pohodlie pri modelovaní. V príklade na Obr. 9.3b prideľuje funkcia N hornej vodorovnej hrane usporiadanú dvojicu [*Mobile Resources*, *T9*], dolnej vodorovnej hrane dvojicu [*T10*, *Tracks*] atď.

Declarations
colset cTracks = with Line | Group1 | Group2 | Station | Depot;
colset cMobileResources = with Loco | Coupler | Examiner;
colset cProcesses = INT;
var proc: cProcesses;

a)



b)

Obr. 9.3 Ilustračný model vo farbenej Petriho sieti: a) deklaračná časť, b) diagram CPN.

(6) Funkcia farieb *C* zobrazuje každé miesto *p* na typ *C*(*p*). To intuitívne znamená, že každá značka v mieste *p* musí mať hodnotu typu *C*(*p*). V príklade majú miesta *Loco Request*, *Loco Arrival* a *Loco Coupling* priradený typ *cProcesses*, miesto *Mobile Resources* typ *cMobileResources* a miesto *Tracks* typ *cTracks*.

(7) Funkcia strážnych podmienok *G* prevádza každý prechod *t* do booleovského výrazu, v ktorom všetky premenné majú typy z množiny $\sum$. Pri kreslení CPN sa neuvádzajú tie výrazy, v ktorých vždy vyjde výsledok true. Príkladom na Obr. 9.3 je výraz *proc <= 2* pri obidvoch prechodoch – výraz zaručí aktiváciu prechodov iba vtedy, keď premennú *proc* možno vo vstupnom mieste naviazať na značku s hodnotou jeden alebo značku s hodnotou dva.

(8) Funkcia hranových výrazov *E* mapuje každú hranu a na výraz typu $C(p)_{MS}$. To znamená, že každý hranový výraz sa musí vyhodnotiť do multi-množín nad typom priľahlého miesta *p*. Diagram CPN môže obsahovať aj hranový výraz *expr* typu *C*(*p*), čo sa považuje za skrátenie zápisu 1`(*expr*). Výrazy *1`Loco* a *1`Coupler* sú hodnoty z multi-množiny $cMobileResources_{MS}$, obdobne výraz *1`Station++1`Line* na 2 hranách dáva hodnoty z multi-množiny $cTracks_{MS}$. Výraz *proc* na ostatných hranách je skrátením zápisu *1`proc* a ide o premennú vyhodnotenú do multi-množiny $cProcesses_{MS}$.

(9) Inicializačná funkcia *I* zobrazuje každé miesto *p* na uzavretý výraz, ktorý musí byť typu $C(p)_{MS}$. Pri kreslení CPN sa vynechávajú inicializačné výrazy vyhodnocované ako *Ø*. V príklade sa nachádzajú inicializačné výrazy (počiatočné značenia) pri troch miestach: *Loco Request*, *Mobile Resources* a *Tracks*; zvyšné 2 miesta majú tieto výrazy vynechané, teda po inicializácii neobsahujú nijakú značku. Výraz *1`1++1`2++1`3* znamená prítomnosť troch značiek na začiatku, jednej značky s hodnotou 1, jednej s hodnotou 2 a jednej značky s hodnotou 3.

Po zadefinovaní štruktúry nasleduje opis správania CPN, k čomu treba pre všetky prechody $t \in T$ a pre všetky dvojice vrcholov $(x_1, x_2) \in (P \times T) \cup (T \times P)$ ešte túto notáciu:

- $A(t) = \{\forall a \in A \mid N(a) \in (P \times \{t\}) \cup (\{t\} \times P)\}$ – množina všetkých hrán incidentných s prechodom *t*.
- $Var(t) = \{v \mid v \in Var(G(t)) \vee \exists a \in A(t) : v \in Var(E(a))\}$ – množina všetkých premenných, ktoré sa nachádzajú v strážnej podmienke prechodu *t* alebo na jeho incidentnej hrane.
- $A(x_1, x_2) = \{a \in A \mid N(a) = (x_1, x_2)\}$– množina hrán medzi vrcholmi x_1 a x_2.
- $E(x_1, x_2) = \sum_{a \in A(x_1, x_2)} E(a)$ súčet multi-množín z výrazov na hranách medzi vrcholmi x_1 a x_2.

Definícia 9-26

Väzbou prechodu *t* nazývame takú funkciu *b* definovanú na *Var*(*t*), pre ktorú platí:

(1) $\forall v \in Var(t) : b(v) \in Type(v)$.

(2) *G*(*t*)<*b*>.

B(*t*) označuje množinu všetkých väzieb pre *t*.

■

Pre prechod *T9* v ilustračnom príklade je *Var*(*T9*) = {*proc*} a keďže *G*(*T9*) je podmienka *proc*<=*2*, tak *proc* = 1 a *proc* = 2 sú jediné dve možné väzby prechodu *T9*, čiže *B*(*T9*) = {*proc* = 1, *proc* = 2}.

Definícia 9-27

Značkový prvok je dvojica (*p*,*c*), kde $p \in P$ a $c \in C(p)$, kým *väzbový prvok* je dvojica (*t*,*b*), kde $t \in T$ a $b \in B(t)$. Množina všetkých značkových prvkov sa označuje *TE*, kým množina všetkých väzbových prvkov sa označuje *BE*.

Značenie je multi-množina nad *TE*, pokým *krok* je neprázdna a konečná multi-množina nad *BE*. *Počiatočné značenie* M_0 je značenie, ktoré sa získa vyhodnotením inicializačných výrazov:

$$\forall (p,c) \in TE : M_0(p,c) = (I(p))(c).$$

Množiny všetkých značení a krokov sa označujú **M**, resp. **Y**.

■

Každé značenie $M \in TE_{MS}$ určuje jedinečnú funkciu M^* definovanú na množine *P* takú, že $M^*(p) \in C(p)_{MS}$:

$$\forall p \in P \; \forall c \in C(p) : (M^*(p))(c) = M(p,c).$$

Na druhej strane, každá funkcia M^* definovaná na množine *P* taká, že $M^*(p) \in C(p)_{MS}$ pre všetky $p \in P$, určuje jedinečné značenie *M*:

$$\forall (p,c) \in TE : M(p,c) = (M^*(p))(c).$$

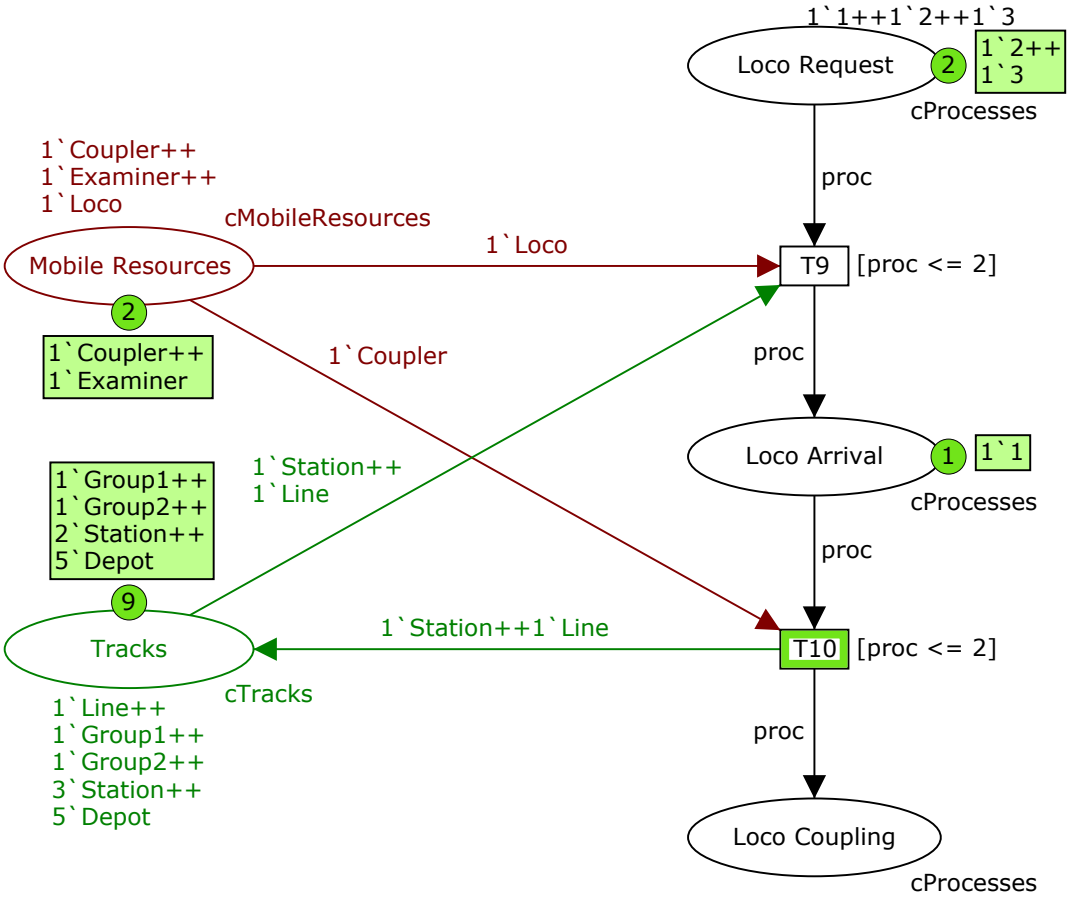
Preto sa často značenia reprezentujú funkciami definovanými na množine *P*.

Značkovým prvkom v ilustračnom príklade je napríklad dvojica (*Tracks*, *Line*), (*Loco Request*, 2) alebo (*Loco Request*, 6). Väzbovým prvkom je (*T9*, *proc* = 1) alebo (*T10*, *proc* = 2), ale už nie napríklad (*T9*, *proc* = 4).

Definícia 9-28

Krok *Y* je *aktivovaný* v značení *M* práve vtedy, keď je splnená vlastnosť:

$$\forall p \in P : \sum_{(t,b) \in Y} E(p,t)\langle b \rangle \leq M(p).$$



Obr. 9.4 Ilustračný model v CPN po vykonaní jedného kroku.

Nech krok Y je aktivovaný pri značení M. Keď $(t,b) \in Y$, hovoríme, že prechod t je *aktivovaný* pri značení M pre väzbu b. Tiež hovoríme, že väzbový prvok (t,b) je aktivovaný pri značení M, rovnako ako prechod t. Keď (t_1,b_1), $(t_2,b_2) \in Y$ a $(t_1,b_1) \neq (t_2,b_2)$, hovoríme, že väzbové prvky (t_1,b_1) a (t_2,b_2) sú *súbežne aktivované*, tak ako aj t_1 a t_2. Keď $|Y(t)| \geq 2$, hovoríme, že prechod t je súbežne aktivovaný sám so sebou. Keď $Y(t,b) \geq 2$, hovoríme, že väzbový prvok (t,b) je aktivovaný sám so sebou.

■

Definícia 9-29

Keď krok Y je aktivovaný v značení M_1, môže sa *vykonať*, pričom zmení značenie M_1 na iné značenie M_2, definované:

$$\forall p \in P : M_2(p) = \left(M_1(p) - \sum_{(t,b) \in Y} E(p,t)\langle b \rangle \right) + \sum_{(t,b) \in Y} E(t,p)\langle b \rangle .$$

Prvá suma sa nazýva *odstránené* značky, kým druhá sa nazýva *pridané* značky. Navyše hovoríme, že M_2 je *priamo dosiahnuteľné* z M_1 výskytom kroku Y, čo sa zapisuje: $M_1 [Y> M_2$.

■

Aktiváciu prechodu možno ilustrovať na Obr. 9.3 na prechode *T9*. Ten je aktivovaný pre väzbový prvok (*T9*, *proc* = 1), nakoľko všetky vstupné miesta obsahujú značky požadované na hranách: miesto *Mobile Resources* má v značení značku *1`Loco*, miesto *Tracks* má v značení požadované značky *1`Station* a *1`Line* a miesto *Loco Request* môže poskytnúť značku *1`1*, ktorá určí väzbový prvok. Po vykonaní prechodu s daným väzbovým prvkom vznikne nové značenie, ktoré možno vyčítať z Obr. 9.4: zmenil sa obsah všetkých miest okrem miesta *Loco Coupling*. Nové značenia miest sú zapísané v zelených obdĺžnikoch pri miestach (značenia mimo zelených obdĺžnikov reprezentujú počiatočné značenia jednotlivých miest).

Definícia 9-30

Konečná postupnosť vykonania prechodov je postupnosť značení a krokov:

$$M_1 [Y_1> M_2 [Y_2> M_3 \ldots M_n [Y_n> M_{n+1}$$

taká, že $n \in \mathbf{N}$ a $M_i [Y_i> M_{i+1}$ pre všetky $i \in \{1, 2, \ldots, n\}$, M_1 je *počiatočné značenie*, M_{n+1} je *koncové značenie* a n je *dĺžka*.

Analogicky sa *nekonečnou postupnosťou vykonania prechodov* nazýva postupnosť značení a krokov:

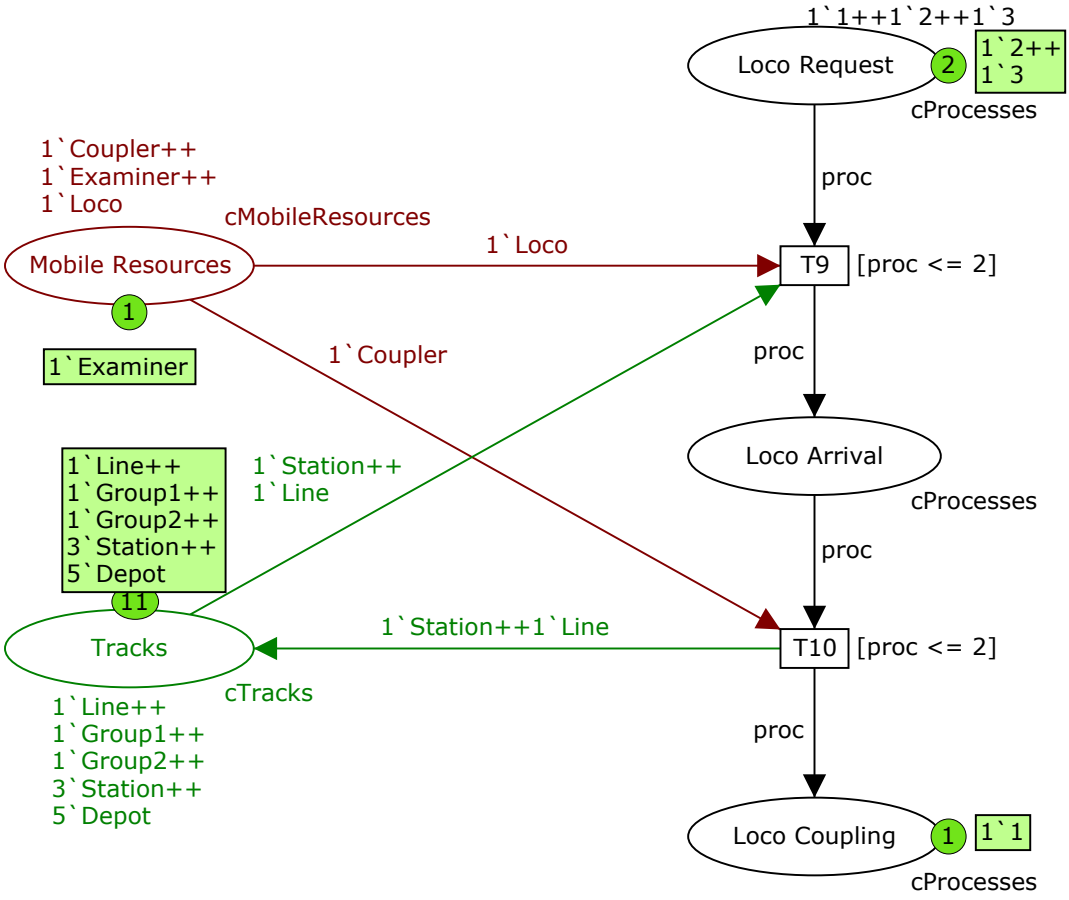
$$M_1 [Y_1> M_2 [Y_2> M_3 \ldots$$

taká, že $M_i [Y_i> M_{i+1}$ pre všetky $i \geq 1$.

Značenie *M"* je *dosiahnuteľné* zo značenia *M'*, ak existuje konečná postupnosť vykonania prechodov, začínajúca v *M'* a končiaca v *M"*. Množina značení dosiahnuteľných zo značenia *M'* sa označuje $[M'>$. Značenie je dosiahnuteľné, ak patrí do $[M_0>$.

■

Ak v sieti na Obr. 9.3 – označme jej značenie M_1 – vykonáme prechod *T9* s väzbou *proc* = 1, označme krok Y_1, dostaneme značenie M_2 zobrazené na Obr. 9.4. Ako vidno, jediným aktivovaným je prechod *T10* s väzbou *proc* = 1. Jeho vykonaním (krok Y_2) dostaneme značenie M_3 zobrazené na Obr. 9.5. Toto značenie je zároveň aj koncovým, keďže nijaký ďalší prechod nie je aktivovaný. Ilustračný príklad má teda konečnú postupnosť vykonania prechodov $M_1 [Y_1> M_2 [Y_2> M_3$.



Obr. 9.5 Ilustračný model po vykonaní prechodov T9 a T10 s väzbou proc = 1.

Ak v sieti na Obr. 9.3 vykonáme prechod *T9* s inou väzbou, *proc* = 2, pôjde o krok odlišný od Y_1, označme ho Y_3, a taktiež dostaneme nové značenie M_4, podobné značeniu M_2 zobrazenému na Obr. 9.4. Odlišnosť bude vo výmene značiek medzi dvoma miestami: *Loco Request* vtedy bude obsahovať značenie *1`1++1`3* a *Loco Arrival* bude obsahovať značku *1`2*. Pri značení M_4 bude jediným aktivovaným prechod *T10* s väzbou *proc* = 2. Jeho vykonaním (krok Y_4) dostaneme značenie M_5 analogické so značením M_3. Aj toto značenie je koncovým pre postupnosť vykonania prechodov M_1 [Y_3>M_4 [Y_4>M_5, ktorá je odlišná od vyššie uvedenej. Z tohto príkladu vidno, že postupnosť vykonania prechodov nezávisí iba od poradia vykonania prechodov, ale pri farbených Petriho sieťach aj od farieb značiek, ktorá sa nachádzajú vo väzbových prvkoch.

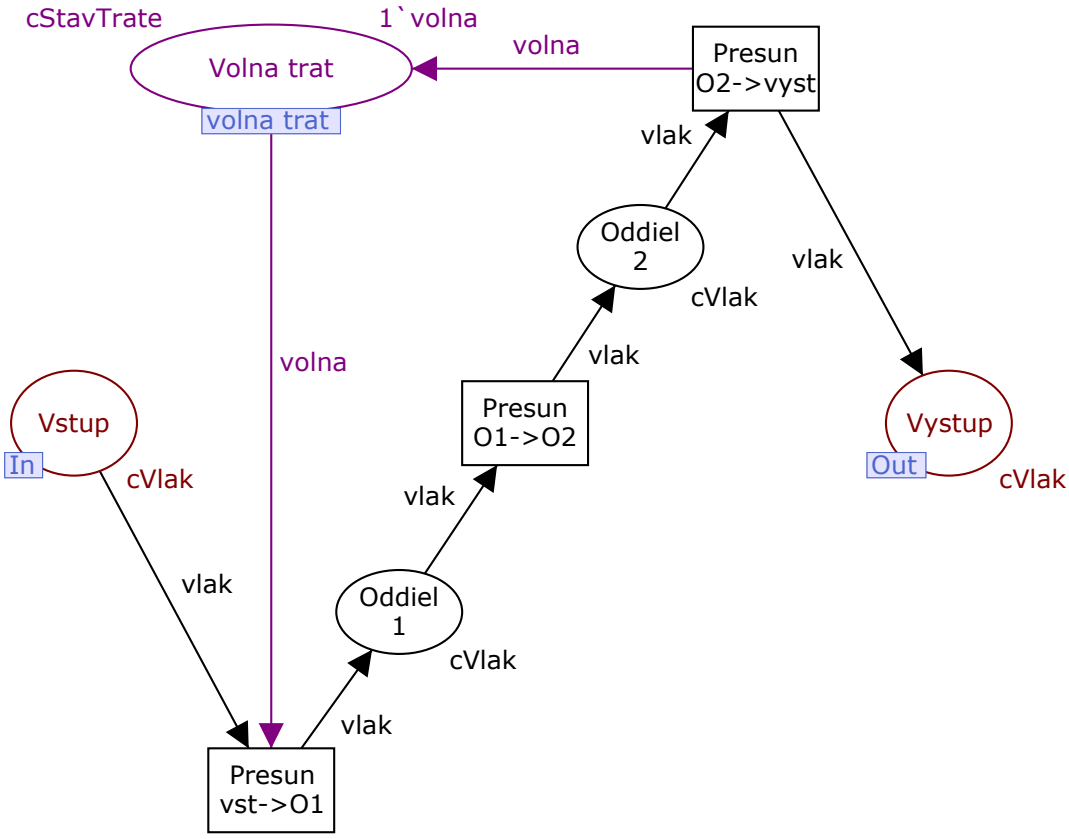
A.2.2 Hierarchická štruktúra

Hierarchická štruktúra siete sa dosiahne možnosťou vytvárania podsietí (zvaných stránky) a mechanizmov ich prepojenia na uzly nadradenej siete, ktoré zabezpečujú prenos značiek medzi modulmi siete. Tieto mechanizmy obsahujú

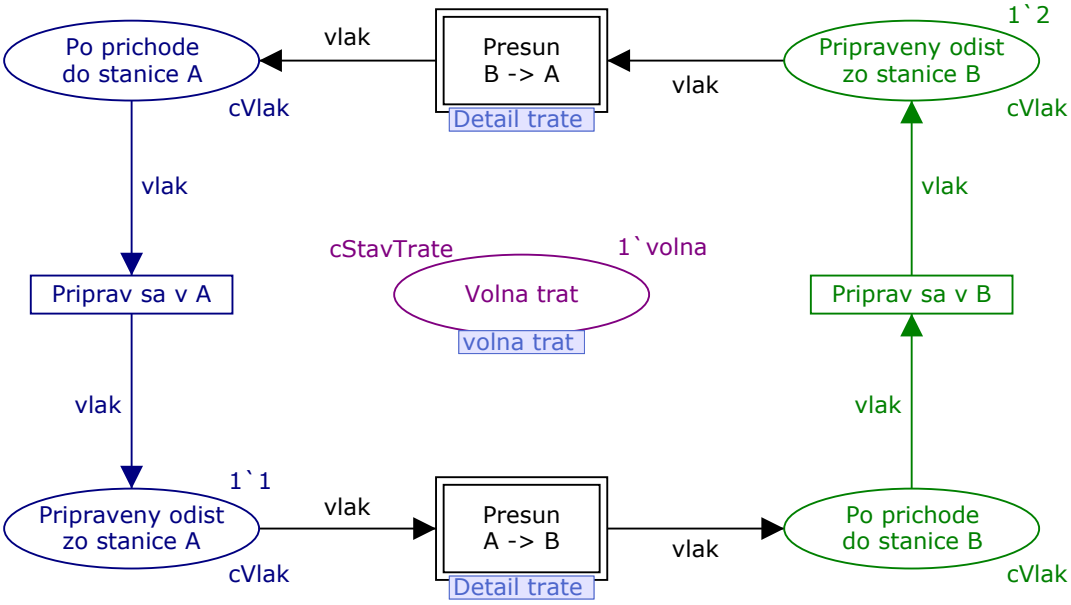
(1) *substitučné prechody*, ktoré v nadradenej sieti nahrádzajú podradenú sieť,

(2) *sokety a brány*, ktoré tvoria rozhranie medzi nadradenou a podradenou sieťou pre prenos značiek medzi úrovňami – všetky miesta v nadradenej sieti spojené so substitučným prechodom sa nazývajú sokety a zodpovedajú im brány v podsieti, ktoré môžu byť vstupné, výstupné alebo vstupno-výstupné.

Na Obr. 9.6 a Obr. 9.7 možno vidieť príklad použitia hierarchie na modelovanie pohybu vlakov po 1-koľajnom traťovom úseku s dvomi oddielmi medzi stanicami A a B. Na Obr. 9.7 je sieť modelujúca presuny medzi stanicami obidvoma smermi, pričom prechody modelujúce tieto jazdy *Presun* $A \rightarrow B$ a *Presun* $B \rightarrow A$ sú substitučné prechody, ktoré sú detailne opísané podstránkou *Detail trate* (Obr. 9.6). Podstránka



Obr. 9.6 Ilustrácia hierarchickej CPN – podstránka Detail trate.



Obr. 9.7 Ilustrácia hierarchickej CPN – hlavná stránka Prehlad.

má dve brány: vstupnú nazvanú *Vstup* a výstupnú nazvanú *Výstup*. Tie sú napojené príslušným spôsobom na sokety na hlavnej stránke. Napr. pre substitučný prechod *Presun* $A \rightarrow B$ je vstupným soketom miesto *Pripraveny odist zo stanice A* a výstupným soketom miesto *Po prichode do stanice B*. Z príkladu taktiež vidno významný efekt hierarchie, kde obidva substitučné prechody sú popísané rovnakou podstránkou, pričom pre simuláciu sa vytvoria dve inštancie rovnakej štruktúry. Teda netreba vytvoriť dve podstránky.

Okrem toho môže nástroj na farbené Petriho siete ponúknuť aj tzv. *zlučujúce miesta*, ktoré umožňujú prístup k miestu s rovnakým obsahom značiek a zmenu tohto obsahu z ľubovoľného miesta Petriho siete. Tieto prvky možno použiť nezávisle od existencie hierarchického usporiadania v modeli CPN, t. j. aj v rámci jednostránkového modelu. Ich hlavná výhoda sa však prejaví najmä pri prepojení miest medzi rôznymi stránkami po zavedení hierarchie.

V príklade je zlučujúcim miestom *Volna trat*, ktoré sa nachádza na obidvoch stránkach, pričom do štruktúry je zapojené iba na podstránke. Jeho prítomnosť na hlavnej stránke je najmä z dôvodu sledovania modelu.

Hierarchická vlastnosť Petriho siete sa používa najmä v takých modeloch, kde Petriho sieť je svojou zložitosťou na jednej stránke ťažko čitateľná a možno ju ďalej logicky rozdeliť. Vzniknú tak moduly, ktorých zavedenie sprehľadní model, uľahčí manipuláciu so sieťou a poukáže na modularitu modelu. Nie je to vlastnosť, ktorá by

zvýšila modelovaciu schopnosť Petriho siete, ale zavádza lepší prehľad pre používateľa.

A.2.3 Časovanie farbenej Petriho siete

Čas v modeli CPN sa reprezentuje pomocou

- časových pečiatok pridelených značkám,
- výrazov upravujúcich časové pečiatky,
- vnútorného počítadla zvaného hodiny a
- podmienok sprístupnenia časovaných značiek.

Časovaná Petriho sieť sa vytvorí vtedy, keď aspoň jeden typ je označený ako *časovaný*, pričom nie všetky typy v časovanom modeli musia mať takéto označenie.

Všetkým značkám časovaného typu sa v modeli priradia časové pečiatky, ktorých počiatočná hodnota sa rovná hodnote hodín v čase vzniku značky. Hodnota pečiatky sa môže zmeniť vo výrazoch na prechodoch a hranách. Jej efekt sa prejaví pri vyhodnotení podmienky sprístupnenia prechodu, kedy sa už berie do úvahy aj čas: značka bude dostupná pre prechod iba vtedy, keď aktuálny čas hodín je väčší alebo rovný jej časovej pečiatke. V opačnom prípade čaká značka v mieste minimálne do splnenia tejto podmienky.

Čas v Petriho sieťach umožňuje modelovať trvanie operácií v systéme, čo rozširuje uplatnenie Petriho siete v takejto forme aj pre diskrétne simulačné modely.

A.2.4 Analýza farbenej Petriho siete

Analytické metódy pre farbené Petriho siete sú analogické metódam pre základnú triedu P/T Petriho sietí. Vyšetrením stavového priestoru (stromu dosiahnuteľných značení) možno získať informácie o dynamických vlastnostiach modelu: ohraničenosti, živosti, návratnosti a prijateľnosti. Zostavenie invariantov miest a prechodov zasa pomôže pochopiť statické vlastnosti, ktoré má každý model špecifické.

V mojej práci nachádza uplatnenie najmä analýza stavového priestoru. Stav v stavovom priestore reprezentuje značenie v príslušnej Petriho sieti. Orientovaná hrana zo stavu S_i do stavu S_j znamená, že v značení M_i, zodpovedajúcom stavu S_i, existuje prechod, ktorý je aktivovaný a keď sa vyskytne, Petriho sieť prejde do značenia M_j, zodpovedajúceho stavu S_j. Preto sa stavový priestor tiež nazýva graf dosiahnuteľnosti alebo graf výskytu.

Ohraničenosť sa určuje pre každé miesto z dvoch pohľadov. Prvý pohľad, podobne ako pri PN nižšej úrovne obmedzenosť (kapitola A.1.2), sa zameriava na počet značiek v mieste. Napríklad v ilustračnom modeli na Obr. 9.3 až Obr. 9.5 je počet značiek v mieste *Loco Request* z intervalu <2, 3>, t. j. miesto v danej sieti obsahuje v každom stave modelu najmenej 2 značky a najviac 3. Pre miesto *Mobile Resources* je ohraničením interval <1, 3> a pre miesto *Tracks* interval <9, 11>.

Druhý pohľad opisuje multi-množinové ohraničenia. Dolné multi-množinové ohraničenie je množina značiek, ktoré sa v mieste vyskytujú v každom stave. Napr. pre miesto *Tracks* je to multi-množina *1`Group1++1`Group2++2`Station++5`Depot* a pre miesto *Loco Coupling* prázdna množina. Horným multi-množinovým ohraničením je zjednotenie všetkých značiek (pre každú značku s maximálnym počtom výskytov), ktoré sa vo všetkých stavoch modelu v mieste nachádzajú. Pre miesta *Mobile Resources* a *Tracks* sa multi-množina rovná počiatočnému značeniu a pre miesto *Loco Coupling* to je množina *1`1++1`2*.

Živosť Petriho siete charakterizuje správanie jednotlivých prechodov z hľadiska podmienenosti ich výskytu počas simulácie zmien stavov Petriho siete. Táto vlastnosť je rovnaká ako pre PN nižšej úrovne (kap. A.1.2). Pre živý prechod platí, že sa počas ďalšej simulácie Petriho siete môže vykonať. Naopak mŕtvy prechod sa už ďalej nemôže vykonať pre nijakú z možných postupností vykonávania prechodov. Od vlastností prechodov sa ďalej odvíja aj živosť celej siete, ktorá môže byť na rôznych úrovniach.

Siete v rozoberaných príkladoch sú z tohto hľadiska odlišné. CPN na Obr. 9.3 až Obr. 9.5 je kvázi-živá, pričom značenie na Obr. 9.5 je uviaznutím (obidva prechody sú mŕtve). Je to evidentné aj z postupnosti vykonávania prechodov v tejto sieti, ktorá je konečná, t. j. príde stav, keď už ďalšie vykonávanie nie je možné, t. j. uviaznutie. V tomto prípade sú uviaznutia v sieti dve – sú to koncové stavy vyššie rozobratých postupností vykonávania prechodov líšiacich sa v použitej väzbe.

Hierarchická CPN na Obr. 9.6 a na Obr. 9.7 je živá, čo je zabezpečené tým, že značky reprezentujúce vlaky majú možnosť cirkulovať do kruhu a vďaka lineárnemu charakteru modelu a jednotkovému obmedzeniu vstupu na trať (to je existencia iba jednej povoľujúcej značky) je vylúčená možnosť výskytu uviaznutia, a teda každý prechod v sieti sa môže pri ďalšej simulácii vyskytnúť.

Návratnosť Petriho siete vypovedá o možnosti dosiahnuť počiatočné značenie PN z ľubovoľného iného značenia, v ktorom sa PN môže nachádzať. V prípade, že sa dá vrátiť do iného ako počiatočného značenia, hovorí sa mu *domovský stav*. V prvom príklade neexistuje nijaké domovské značenie a v druhom príklade sú domovskými značeniami všetky.

Vlastnosť *prijateľnosti* súvisí s frekvenciou výskytu prechodov v postupnostiach vykonania prechodov a s ich vzájomným ovplyvnením. Dá sa vyšetriť iba v modeloch s nekonečnou postupnosťou vykonania prechodov.

Podrobnejšie o analytických vlastnostiach CPN sa možno dozvedieť v publikácii (Jensen, 1997/2, str. 127-137).

A.2.5 CPN Tools

CPN Tools je počítačový nástroj na editáciu, simuláciu a analýzu farbenej Petriho siete, ktorá navyše môže byť časovaná a hierarchická (CPN Tools, online). Obrázky Petriho sietí v tejto práci boli vytvorené práve v prostredí tohto nástroja.

Na editáciu poskytuje CPN Tools interaktívne techniky vyššej úrovne: palety nástrojov (toolglasses, palettes) a pečiatkové ponuky (marking menus). Netradičné sú aj prostriedky spätnej väzby aplikácie – poskytujú kontextové chybové hlásenia a uvádzajú vzťahy závislosti medzi sieťovými prvkami.

CPN Tools počas tvorby siete inkrementačne kontroluje syntax a generuje kód. Samotná simulácia sa teda dá pustiť v ľubovoľnom bode editácie za predpokladu absencie chýb v skladbe modelu.

Analýzu vykonáva prostredníctvom grafu dosiahnuteľnosti (stavového priestoru), ktorý možno vypočítať úplne alebo čiastočne. Z vypočítaného grafu sa odvádzajú vlastnosti správania sa modelu: obmedzenosť, návratnosť (domovský stav), živosť a prijateľnosť, ktorých výsledky sa dajú spolu so základnými informáciami o stavovom priestore a jeho výpočte zapísať do protokolu.

Model systému vo forme hierarchickej farbenej Petriho siete sa skladá z dvoch zložiek: grafickej sieťovej štruktúry a jej formálneho zápisu (inscription). Tento zápis v súlade s doteraz uvedeným opisom obvykle zahŕňa množiny farieb, počiatočné značenia, hranové výrazy a strážne podmienky. Okrem toho existujú *procedúry* (kódové segmenty) viazané na prechody siete, ktoré môžu vykonávať zložitejšie výpočty s údajovými objektmi značiek, prípadne vedľajšie efekty ako komunikáciu so súbormi.

Zápis sa realizuje v jazyku CPN ML (jazyk ML pre farbené Petriho siete) založenom na štandardnom jazyku ML vo verzii New Jersey. Ide o všeobecný predikátový (funkcionálny) jazyk.

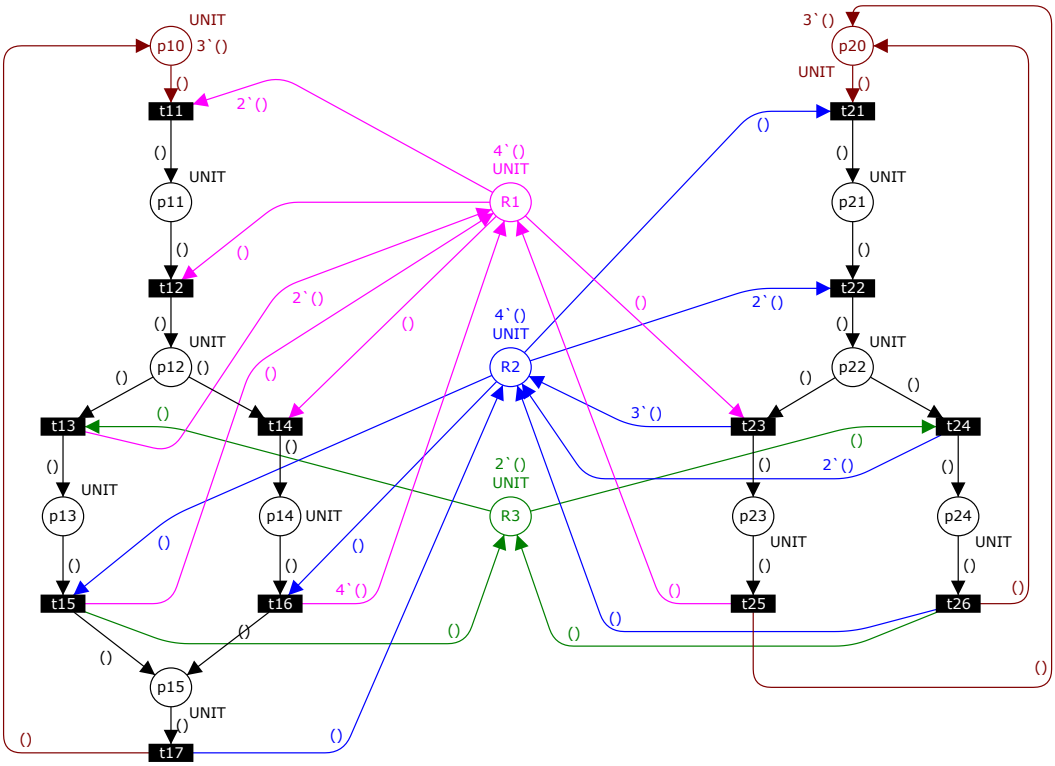
B Riešenie uviaznutia pomocou sifónu

Zabránenie výskytu prázdneho (mŕtvo označeného) sifónu:

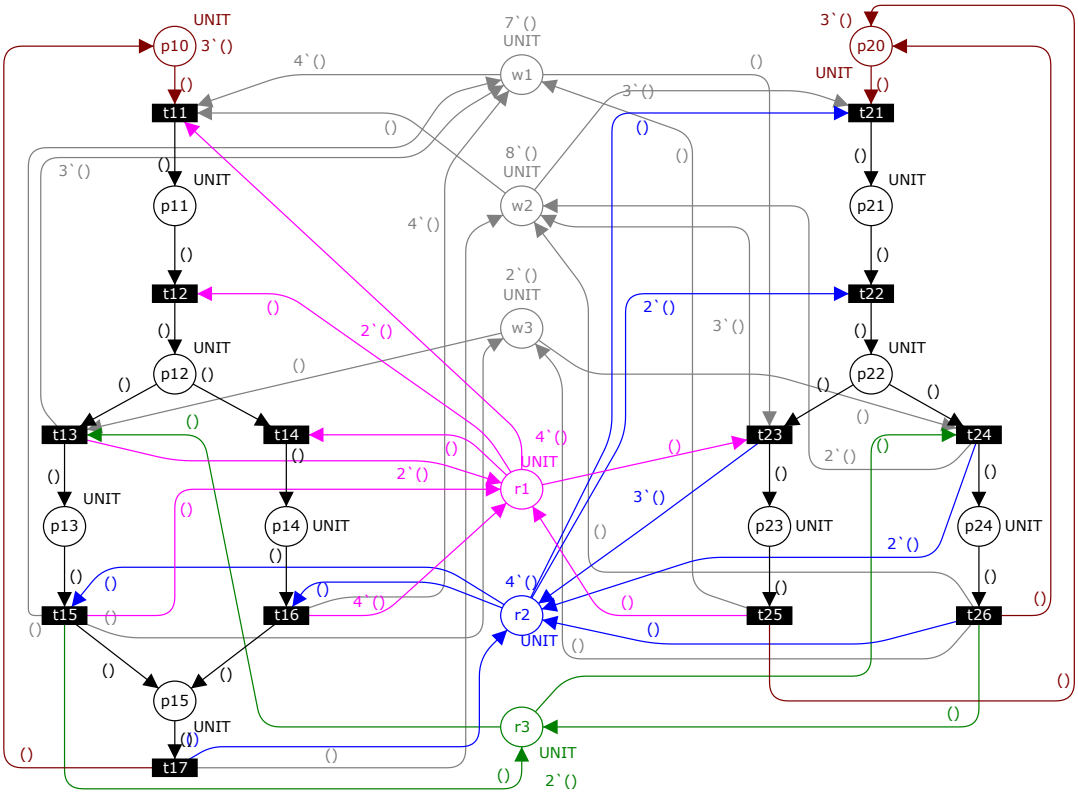
Obr. 9.8 uvádza riešenú sieť S^4PR

Obr. 9.10 uvádza systém s riadiacou podsieťou

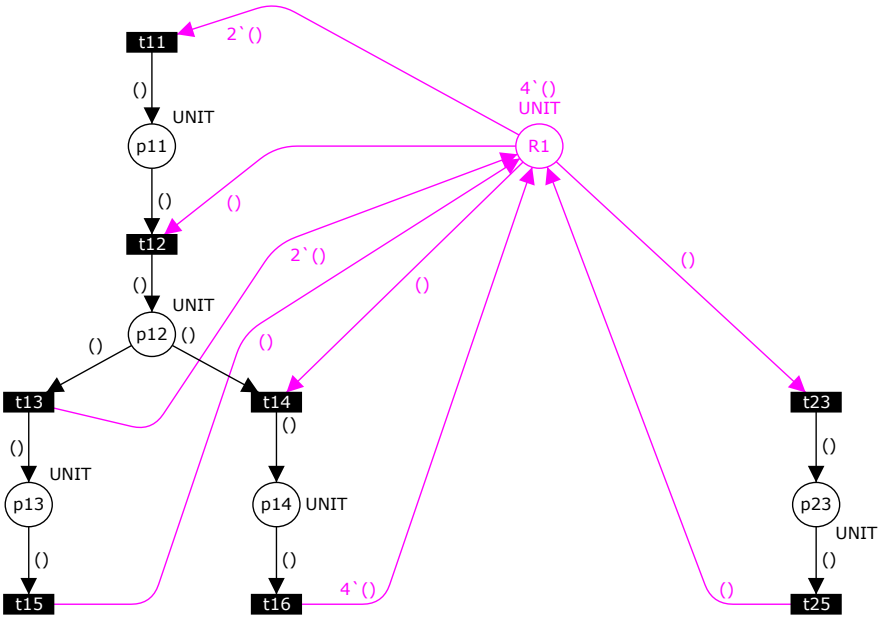
Obr. 9.9 zobrazuje jeden z dvoch sifónov, ktoré sa v sieti nachádzajú a ktorým treba zabrániť



Obr. 9.8 Model S^4PR pre RAS s 2 procesmi a 3 typmi prostriedkov podľa (Park-Reveliotis, 2001).



Obr. 9.10 Podsieť z Obr. 9.8 s kontrolnou podsieťou tvorenou miestami *w1*, *w2*, *w3*.



Obr. 9.9 Podsieť siete z Obr. 9.8 obsahujúca sifón: miesta *p12*, *p13*, *p14*, *p23* a *R1*. Prechod *t11* môže sifón zbaviť značiek.

C Algoritmus bankára a jeho vylepšenia

V nasledujúcich podkapitolách uvádzam algoritmy z dvoch zdrojov.

V prvej podkapitole sú algoritmy z článku (Lawley et al., 1998) pre jedno-jednotkový systém prideľovania prostriedkov (SU-TO-RAS) na doplnenie opisu v kapitole 2.3.1.2. Systém je definovaný v kapitole 2.2.2.

V druhej podkapitole sú algoritmy z publikácie (Reveliotis, 2000) pre AGV RAS (čo je principiálne SU-PO-RAS so špecifikom, že prostriedky sú úseky siete) na doplnenie opisu v kapitole 2.3.1.3, kde je tento systém aj priblížený.

Algoritmy sú ponechané s pôvodným označením z publikácií, pričom prvý algoritmus pre AGV RAS zodpovedá prvému algoritmu pre SU-TO-RAS, čo je pôvodná verzia algoritmu bankára.

C.1 Algoritmy bankára pre SU-TO-RAS

Algoritmy, ktoré ďalej nasledujú, používajú nasledovné symboly:

C_j	počet prostriedkov typu j
p(i)	fáza procesu na i-tom mieste množiny $\mathbf{S}^+$ (p je indexová funkcia)
P_{km}	m-tá fáza spracovania procesu typu k
P_{pq}	q-tá fáza spracovania procesu typu p
$\mathbf{R}$	množina všetkých prostriedkov systému
$\mathbf{S}_u$	stavový vektor systému prislúchajúci vrcholu u v stavovom priestore
$\mathbf{S}_u^+$	množina fáz spracovania procesu, ktoré sú v stave u obsadené aspoň jednou inštanciou procesu (podmnožina stavového vektoru pre vrchol u v stavovom priestore)
$\mathbf{V}_0$	množina usporiadaných stavov
$\mathbf{V}_1$	množina stavov usporiadaných na úrovni $\mathbf{V}_1$ (podrobnosti v hlavnom texte)
$\mathbf{\Pi}$	množina všetkých inštancií procesov v systéme
π_{km}	proces typu k v m-tej fáze spracovania
ξ_j	množina všetkých fáz procesu spracovaných prostriedkom j

Na reprezentáciu prostriedkov a ich vzťahov k procesom sa používajú 3 hlavné údajové štruktúry: matice *Potrebne* a *Pridelene* a vektor *Volne*. Ich naplnenie:

Pre $i = 1, 2, \ldots, |\mathbf{S}_u^+|$ a $j = 1, 2, \ldots, |\mathbf{R}|$,

Potrebne[*i*][*j*] = 1, ak sa prostriedok R_j nachádza vo zostávajúcej ceste p(*i*) = $P_{km} \in \mathbf{S}_u^+$, 0 inak;

Pridelene[*i*][*j*] = $|\pi_{km}|$, ak p(*i*) = $P_{km} \in \xi_j$, 0 inak;

Volne[*j*] = $C_j - \sum_i$ *Pridelene*[*i*][*j*].

SU-TO-RAS – algoritmus 1: Je $\mathbf{S}_u \in \mathbf{V}_0$, t. j. je daný stav usporiadaný?

vstup: reprezentácia *Pridelene, Potrebne, Volne* pre stav $\mathbf{S}_u$.

výstup: *Prijat* / *Odmietnut*

```
begin
   Π = {1, 2, …, |S_u^+|};
   loop
            // Ak všetky P_pq ∈ S_u^+ boli spracované, prijmi stav. //
      if Π = Ø then return Prijat
            // Inak nájdi p(i), ktoré možno spracovať. //
      else find i ∈ Π such that
         Potrebne[i][j] ≤ Pridelene[i][j] + Volne[j],  ∀j = 1..|R| ;
      if no such i exists, return Odmietnut;
            // Inak spracuj p(i). //
      for j = 1 to |R| do begin
         Volne[j] = Volne[j] + Pridelene[i][j];
         Pridelene[i][j] = 0;
         Potrebne[i][j] = 0;
      end;
      Π = Π \ {i}
   end loop
end
```

■

SU-TO-RAS – algoritmus 2: Je S_v čiastočne usporiadaný vzhľadom k $P_{km} \in S_v^+$?

vstup: P_{km} a reprezentácia *Pridelene*, *Potrebne*, *Volne* pre stav $\mathbf{S}_v$.

výstup: *Prijat* / *Odmietnut*

```
begin
   Π = {1, 2, …, |S_v^+| };
   k = p^-1(P_km) ;          // Získaj riadkový index P_km. //
   loop
                  // Ak možno spracovať P_km, prijmi stav. //
      if Potrebne[k][j] ≤ Pridelene[k][j] + Volne[j],  ∀j = 1..|R|
      then return Prijat
                  // Inak nájdi ľubovoľné p(i), ktoré možno spracovať. //
      else find i ∈ Π such that
         Potrebne[i][j] ≤ Pridelene[i][j] + Volne[j],  ∀j = 1..|R| ;
      if no such i exists, return Odmietnut;
                  // Inak spracuj p(i). //
      for j = 1 to |R| do begin
         Volne[j] = Volne[j] + Pridelene[i][j];
         Pridelene[i][j] = 0;
         Potrebne[i][j] = 0;
      end;
      Π = Π \ {i}
   end loop
end
```

■

SU-TO-RAS – algoritmus 3: Je $S_v \in V_1$, t. j. je daný stav V1-usporiadaný?

vstup: reprezentácia *Pridelene, Potrebne, Volne* pre stav $\mathbf{S}_v$.

výstup: *Prijat* / *Odmietnut*

```
begin
   for i = 1 to |S_v^+| do begin
                  // Okopíruj údajové štruktúry. //
      docas1 = Pridelene; docas2 = Potrebne; docas3 = Volne;
      loop     // Aktualizuj údajové štruktúry pokusom posunúť p(i) dopredu. //
         vysledok = posun_proces (docas1, docas2, docas3, i);
         if vysledok = Blokovany
                  // Proces je blokovaný a nemôže sa pohnúť. //
```

```
        then break
        else if vysledok = Posunuty  // Proces bol úspešne posunutý dopredu. //
                // Je výsledný stav usporiadaný? //
             then begin
                vysledok = PrvyAlgoritmus (docas1, docas2, docas3);
                if vysledok = Prijat
                then return Prijat                    //Sv ∈ V1//
             end
      end loop
   end
   return Odmietnut   //Sv ∉ V1//
end
```

■

SU-TO-RAS – algoritmus 4: Je $\mathbf{S}_v \in \mathbf{V}_0 \cup \mathbf{V}_1$?

predpoklad: π_{km} je posledný proces na posun dopredu a $\mathbf{S}_v$ je výsledný stav.

vstup: $\mathbf{S}_v$, π_{km} .

výstup: *Prijat* / *Odmietnut*

```
begin
                // Nastav Pridelene, Potrebne a Volne pre stav Sv. //
   Napln_udajove_struktury (Sv);
                // Prever, či stav Sv je čiastočne usporiadaný vzhľadom k Pkm. //
   vysledok = DruhyAlgoritmus (Pkm, Pridelene, Potrebne, Volne);
   if vysledok = Prijat,
   then return Prijat;  // Stav je čiastočne usporiadaný vzhľadom k Pkm. //
                // Prever, či stav Sv V1-usporiadaný. //
   vysledok = TretiAlgoritmus (Pridelene, Potrebne, Volne);
   if vysledok = Prijat
   then return Prijat          // Stav je V1-usporiadaný. //
   else return Odmietnut       // Sv ∉ V0 ∪ V1 . Nepovoľ navrhovaný krok. //
end
```

■

C.2 Algoritmy bankára pre AGV RAS

Tu zhrniem tie symboly, ktoré sa používajú v ďalej uvedených algoritmoch a nie sú v nich priamo vysvetlené:

n_1, n_2 a n_3	označujú tri miesta v grafe (sieti), ktoré má vozidlo v rámci jednej úlohy navštíviť, zvyčajne je to zdroj požiadavky na prepravu (n_1), cieľ požiadavky (n_2) a depo (n_3), kde sa majú vozidlá zdržiavať počas nečinnosti v systéme
$V(s)$	množina vozidiel, ktoré sú aktívne v stave s
res(p)	prostriedky alokované procesu p
$o(p)$	funkcia, ktorá parametru p priradí index v hľadanej usporiadanej postupnosti

Prvý algoritmus poskytuje všeobecnú logiku algoritmu bankára, ku ktorej sa viažu ďalšie algoritmy pre AGV RAS.

AGV RAS – algoritmus 1: Je daný stav usporiadaný?

1. Inicializuj:
   (a) $U = P(s)$, kde $P(s)$ je množina inštancií procesov bežiacich v stave s;
   (b) $R = F(s)$, kde $F(s)$ je množina voľných prostriedkov v stave s;
   (c) $i = 0$;
   (d) *Usporiadany* = true;
2. Pokiaľ ($Usporiadany \wedge (U \neq \emptyset)$) vykonaj:
   (a) $i = i + 1$;
   (b) pokús sa nájsť proces $p \in U$ taký, ktorý sa môže ukončiť s pomocou prostriedkov, ktoré má aktuálne pridelené, a prostriedkov v množine R;
   (c) ak nemožno nájsť nijaký taký proces, potom *Usporiadany* = false;
   (d) inak
      i. $o(p) = i$;
      ii. $U = U \setminus \{p\}$;
      iii. $R = R \cup$ res(p);
3. Vráť *Usporiadany*

■

AGV RAS – algoritmus 2: Môže vozidlo *v* dokončiť svoju úlohu?

Vozidlo $v \in V(s)$, ktoré je práve umiestnené na hrane grafu $e_v \equiv (k, l)$ a ktorému zostáva úloha definovaná usporiadaným zoznamom $L_v \equiv < n_1, (n_2 \vee \text{null}), (n_3 \vee \text{null}) >$, môže dokončiť svoju úlohu podľa kroku (2b) všeobecného algoritmu bankára práve vtedy,

(1) ak vrchol l spája s vrcholom n_1 cesta z hrán, ktoré sú práve voľné a
(2) ak vrcholy l alebo k spájajú s vrcholmi n_2 a n_3 cesty z hrán, ktoré sú práve voľné, za predpokladu, že $L_v(3)$ a/alebo $L_v(2) \neq$ null; inak je táto časť triviálne splnená.

■

AGV RAS – algoritmus 3: Výpočet vzťahu ekvivalentnej prepojenosti vrcholov.

Nech $F(s)$ označuje množinu voľných hrán grafu pre daný stav systému AGV RAS. Potom
I. vzťah ε definovaný na množine vrcholov grafu N ako:
$\varepsilon : N \times N \equiv \{(i, j)$: vrcholy i, j sú spojené aspoň jednou cestou z *voľných* hrán$\}$
je vzťah *ekvivalencie*.
II. Triedy ekvivalencie C_i vzťahu ε možno vypočítať nasledujúcim algoritmom:
(1) Na začiatku každý vrchol $n \in N$ tvorí oddelenú triedu ekvivalencie, t. j. $\varepsilon = \{C_n \equiv \{n\}, \forall n \in N\}$.
(2) Pre každú voľnú hranu $e \equiv (k, l) \in F(s)$, ak $C_k \neq C_l$, triedy ekvivalencie vrcholov k a l *sa zlúčia*, t. j. $\varepsilon = (\varepsilon \setminus \{C_k, C_l\}) \cup \{C_k \cup C_l\}$.

■

AGV RAS – algoritmus 4: Algoritmus bankára pre systém AGV.

1. Inicializuj:
   (a) $U = V(s)$;
   (b) použi algoritmus 3 na výpočet počiatočného vzťahu prepojenosti vrcholov ε množinu vrcholov grafu N;
   (c) $i = 0$;
   (d) *Usporiadany* = true;
2. Pokiaľ ($Usporiadany \wedge (U \neq \emptyset)$) vykonaj:
   (a) $i = i + 1$;
   (b) s použitím logiky algoritmu 2 a údajovej štruktúry ε sa pokús nájsť vozidlo $v \in U$, ktoré môže dokončiť svoju úlohu;
   (c) ak nemožno nájsť nijaké také vozidlo, potom *Usporiadany* = false;
   (d) inak
      i. $o(v) = i$;
      ii. $U = U \setminus \{v\}$;
      iii. použitím 2. kroku algoritmu 3 obnov údajovú štruktúru ε;
3. Vráť *Usporiadany*

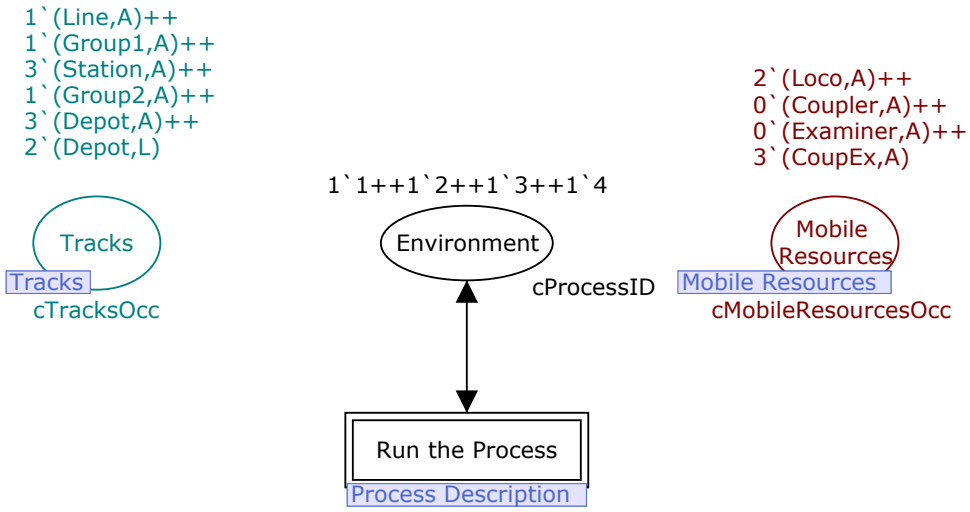
■

D Demonštračný model a jeho analýza

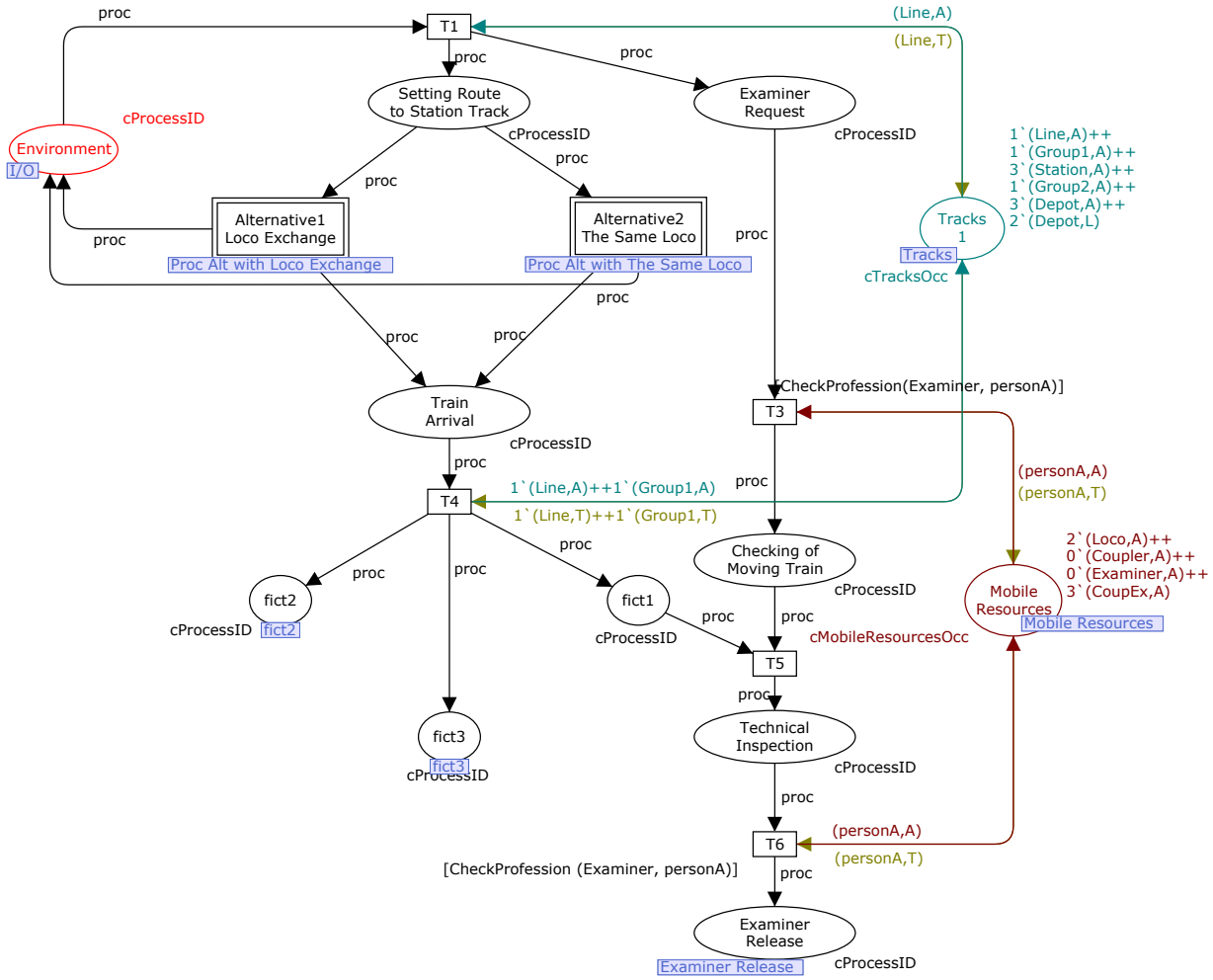
D.1 Model

Definícia použitých množín farieb:

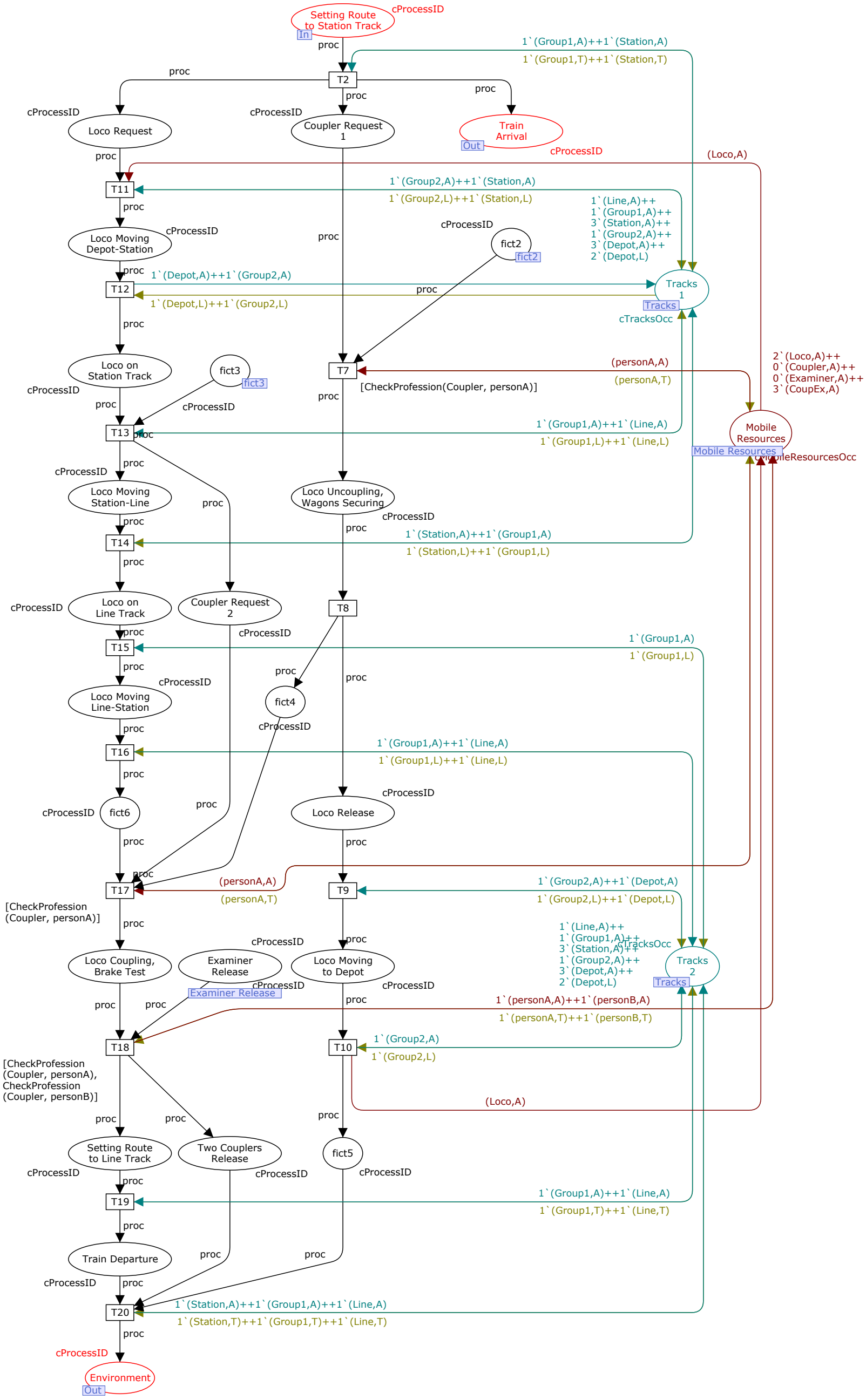
```
colset cTrains = INT;
colset cOccupation = with A | T | L; (* Available | Train | Loco *)
colset cTracks = with Line | Group1 | Station | Group2 | Depot;
colset cTracksOcc = product cTracks * cOccupation;
colset cMobileResources = with Loco | Coupler | Examiner | CoupEx;
colset cMobileResourcesOcc = product cMobileResources * cOccupation;
colset cResources = union RFR: cTracks + RMR: cMobileResources;
```



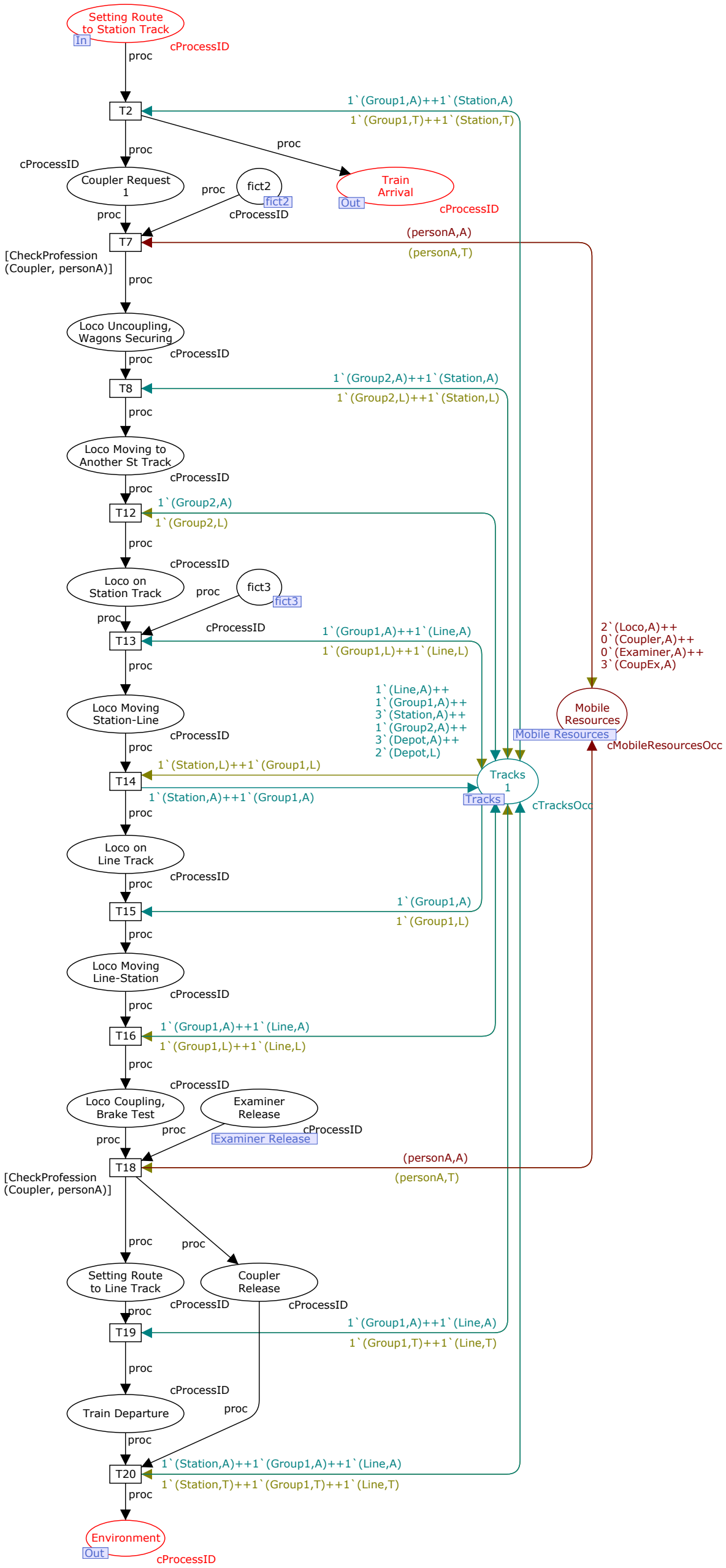
Obr. 9.11 CPN demonštračného modelu – hlavná stránka pre celý systém.



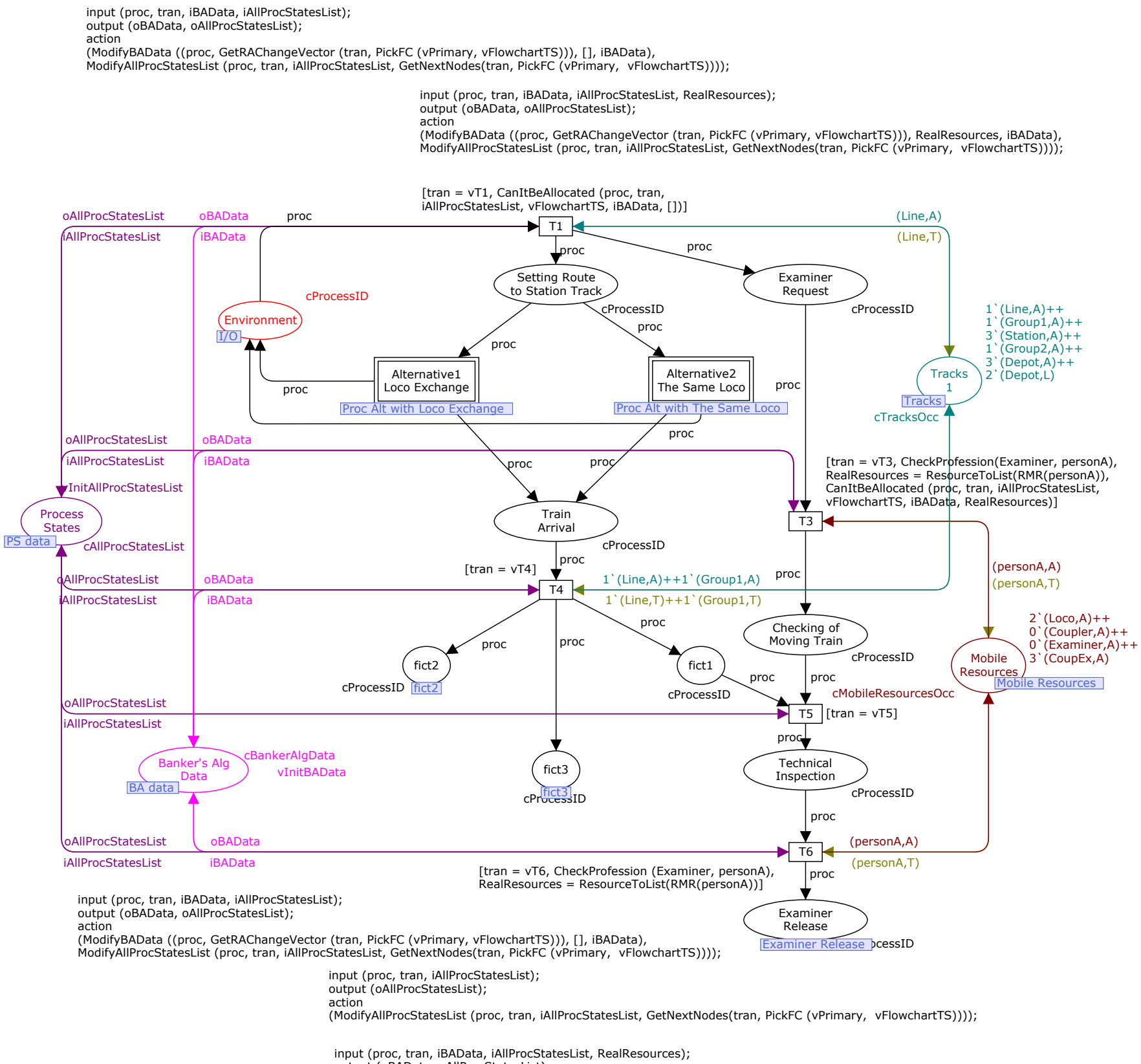
Obr. 9.12 CPN demonštračného modelu – spoločná časť technologického procesu.



Obr. 9.13 CPN demonštračného modelu – stránka variantu 1.



Obr. 9.14 CPN demonštračného modelu – stránka variantu 2.



Obr. 9.15 CPN demonštračného modelu – spoločná časť technolog. procesu s prvkami implementácie algoritmu bankára.

D.2 Ukážka protokolu

```
CPN Tools state space report for:
\\Genesis\zarnay\finalne testy\simple TS v10 no BA.cpn
Report generated: Fri Dec 15 11:59:43 2006


 Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  20798
     Arcs:   72008
     Secs:   230
     Status: Full

  Scc Graph
     Nodes:  329
     Arcs:   1014
     Secs:   2


 Boundedness Properties
------------------------------------------------------------------------


 Home Properties
------------------------------------------------------------------------

  Home Markings
     None


 Liveness Properties
------------------------------------------------------------------------

  Dead Markings
     6 [20361,20338,19362,19308,17943,...]

  Dead Transition Instances
     None

  Live Transition Instances
     None


 Fairness Properties
------------------------------------------------------------------------
```

Uvedená ukážka je vo forme štandardného protokolu poskytovaného v CPN Tools a vznikla po analýze stavového priestoru demonštračného modelu s druhou konfiguráciou, t. j. čatou troch pracovníkov schopných vykonávať obidve požadované

profesie, s dvoma vlakmi v systéme a bez zapojenia riadiaceho algoritmu na vyhýbanie sa uviaznutiam.

Z protokolu možno vyčítať, že stavový priestor tohto modelu obsahuje 20798 stavov a 72008 prechodov, je úplný a jeho výpočet trval počítaču 230 sekúnd, t. j. 3 minúty a 50 sekúnd. Graf SCC (strongly-connected components) obsahuje 329 vrcholov nahrádzajúcich silno-súvislé komponenty pôvodného grafu. Medzi nimi je 1014 hrán a jeho výpočet sa vykonal za 2 sekundy.

Z vlastností počítač zapísal iba návratnosť a živosť. V stavovom priestore sa nenašlo nijaké domovské značenie, no našlo sa šesť mŕtvych značení, ktoré zodpovedajú šiestim stavom, v ktorých môže systém modelovaný pôvodnou Petriho sieťou nežiaduco skončiť svoju činnosť. V zátvorke nasledujú čísla vrcholov, zodpovedajúcich týmto stavom, aby bolo možné preskúmať okolnosti ich vzniku v prípade potreby. Ďalej sa možno dozvedieť, že nijaký prechod nie je mŕtvy a nijaký nie je živý, čo znamená, že všetky prechody majú pri danom počiatočnom značení šancu sa vykonať aspoň raz.

Ostatné vlastnosti sú relevantné vo fáze vývoja modelu, avšak nepotrebné pre porovnávaciu analýzu v tejto práci. Keďže ich zápis do protokolu zaberá nepomerne dlhší čas, vyradil som ho.

Každý riadok v tabuľkách 5.2 a 5.3 bol vypísaný z jedného takéhoto protokolu.