Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky

Detekcia uviaznutí v diskrétnych udalostných systémoch so zdieľanými zdrojmi

Modelovanie a simulácia udalostných systémov: Časť prvá

2016

Obsah

Zoznam obrázkov

Úvod

Workflow procesy [1, 2, 3, 4] patria do triedy dynamických systémov nazývaných diskrétne udalostné systémy [16, 8]. Definíciu workflow procesu môžeme chápať ako definíciu pracovného postupu, ktorý pozostáva zo sledu predpísaných úloh, pričom môžeme definovať, že na vykonanie úlohy sú nutné nejaké zdroje. Ak sa workflow proces následne vykonáva, znamená to, že sa podľa predpísaného postupu vykonávajú jednotlivé úlohy. Vykonávané úlohy nazývame aktivity, sled vykonávaných úloh podľa definície nazývame inštancia workflow procesu. Definícia workflow proocesu určuje pre každú inštanciu jednoznačný začiatok a koniec. Rozlišujeme teda medzi definíciou pracovného postupu a samotným vykonávaním úloh podľa predpísaného pracovného postupu. V širšom zmysle slova môžeme medzi workflow procesy zahrnúť algoritmy, komunikačné protokoly [14], hardvér [17], pružné výrobné systémy [21, 82], procesy v oblasti služieb, ako napríklad proces spracovania poistnej udalosti a podobne [4].

V práci [32] autori pracujú s triedou workflow procesov, v ktorých jednotlivé inštancie zdieľajú spoločné zdroje. Okrem zdieľaných zdrojov sú však inštancie navzájom nezávislé. Zdieľané zdroje uvažované v práci [32] sú trvácne, teda pri vykonávaní predpísaných úloh pri spracovaní inštancie sa zdieľané zdroje nevytvárajú ani nespotrebujú. Zdroje v informačných systémoch, ako pamäť, procesory, vstupné a výstupné porty na hardvérových zariadeniach sú typickým príkladom trvácnych zdieľaných zdrojov. Podobne pracovníci v jednotlivých roliach uvažovaní pri podnikových procesoch predstavujú príklad trvácnych zdrojov.

Hlavným problémom riešeným v práci [32] je otázka korektného ukončenia vykonávaného workflow procesu, v ktorom nezávislé inštancie zdieľajú trvácne zdroje. Problém je vyriešený pre triedu workflow procesov s jedným typom zdieľaných zdrojov pre prípad, že existuje taký počet zdrojov, že pre tento počet zdrojov a každý vyšší počet zdrojov inštancie majú korektné ukončenie. To znamená, že algoritmus prezentovaný

v práci [32] rozhodne, či pre daný workflow proces s jedným typom zdieľaných zdrojov existuje taký počet zdrojov, že pre tento počet zdrojov a ľubovoľný vyšší počet zdrojov, každá inštancia môže byť korektne ukončená. Ak je odpoveď kladná, workflow proces so zdieľanými zdrojmi je podľa [32] korektný, inak je nekorektný. Ako prvý hlavný problém budúceho výskumu autori práce [32] uvádzajú rozšírenie problému na viac typov zdrojov.

V tejto práci budeme riešiť problém inšpirovaný prácou [32], pričom budeme uvažovať viac typov zdrojov. Ako ukážeme v Kapitole 3 na ilustratívnom prípade s tromi typmi zdrojov, existujú workflow procesy, v ktorých je možné ľubovoľný počet inštancií korektne ukončiť pre daný počet zdrojov, avšak pre každý vyšší počet zdrojov to už možné nie je. Vyžadovať, aby existoval pri viacerých typoch zdrojov počet zdrojov, pre ktorý bude môcť byť korektne ukončených ľubovoľný počet inštancií a zároveň aby takáto vlastnosť platila pre každý vyšší počet zdrojov, môže byť teda veľmi reštriktívne. Predstavme si situáciu, že algoritmus obdobný algoritmu v práci [32] by rozhodol, že neexistuje taký počet zdrojov, že pre tento počet zdrojov a každý vyšší počet zdrojov je možné ukončiť ľubovoľný počet inštancií. Napriek tomuto rozhodnutiu môže pre daný proces existovať konkrétny počet zdrojov, pre ktorý je možné ukončiť ľubovoľný počet inštancií. Algoritmus obdobný algoritmu [32] by tento proces vyhodnotil ako nekorektný napriek tomu, že pre daný počet zdrojov je možné každú z ľubovoľného počtu inštancií korektne ukončiť. Aby sme pre takýto proces a pre daný počet zdrojov vedeli túto situáciu rozlíšiť, potrebujeme nájsť metódu a zostaviť algoritmus, ktorý zistí, či v prípade viacerých typov zdieľaných zdrojov pre vopred daný konkrétny počet zdrojov je možné korektne ukončiť ľubovoľný počet inštancií.

Stav, z ktorého nie je možné dosiahnuť korektné ukončenie inštancií budeme nazývať uviaznutím. Ako ilustruje analýza referenčných procesov SAP prezentovaná v [60], uviaznutia predstavujú vážny problém pri aplikácii workflow procesov v praxi. Rozlišujeme dva typy uviaznutí - zamrznutie a zacyklenie (anglicky deadlock a livelock). Zamrznutie predstavuje také uviaznutie, v ktorom nie je možné vykonať žiadnu aktivitu. Zacyklenie je uviaznutie, v ktorom je možné vykonávať aktivity, avšak nie je možné aspoň jednu inštanciu korektne ukončiť. V predchádzajúcich prácach [41, 42, 43] sme zostavili metódu, ktorá pre workflow proces s viacerými typmi trvácnych zdieľaných zdrojov pre konkrétny vopred daný počet zdrojov rozhodne, či workflow proces

pre nejaký počet inštancií obsahuje zamrznutie. Táto metóda však neidentifikovala zacyklenia a teda neriešila problém, či v procese je možné korektne ukončiť ľubovoľný počet inštancií. Problém, ktorý potrebujeme vyriešiť, môžeme teda pomocou uviaznutí formulovať nasledovne. Potrebujeme nájsť metódu a algoritmus, ktorý zistí, či v prípade viacerých typov zdieľaných trvácnych zdrojov pre vopred daný konkrétny počet zdrojov existuje pre nejaký počet inštancií v procese uviaznutie. Takýto algoritmus doposiaľ nie je k dispozícii.

Hlavným cieľom tejto práce je zostaviť metódu a algoritmus na detekciu uviaznutí workflow procesov s nezávislými inštanciami a viacerými typmi zdieľaných trvácnych zdrojov pri danom počte zdrojov jednotlivých typov pre ľubovoľný počet inštancií.

Práca je organizovaná nasledovne. V Kapitole 1 sú popísané definície formalizmov pre modelovanie diskrétnych udalostných systémov s inštanciami a zdieľanými zdrojmi, ktoré budú použité v práci.

V Časti 1.1 sú popísané prechodové systémy podľa [45, 80] ako základný modelovací formalizmus diskrétnych udalostných systémov.

V Časti 1.2 sú popísané Petriho siete [44, 62, 63, 67, 23, 24], ktoré predstavujú rozšírený formalizmus na modelovanie diskrétnych udalostných systémov. Sú pomenované podľa Carla Adama Petriho [64]. V literatúre sa využívajú mnohé podtriedy Petriho sietí, ich základnú klasifikáciu je možné nájsť napríklad v práci [11]. Okrem už spomenutých oblastí ako komunikačné protokoly [14], hardvér [17], či pružné výrobné systémy [21, 82] sú Petriho siete využívané pri riadení diskrétnych udalostných systémov [34, 35], ale napríklad aj pri modelovaní biologických systémov a v oblasti genetiky [61, 26]. Existujú rozšírenia Petriho sietí pre modelovanie hybridných systémov, teda systémov kombinujúcich spojité premenné s udalosťami [49, 50, 19, 20, 22].

Kapitola 1 pokračuje Časťou 1.3, v ktorej sú popísané workflow siete podľa [1, 2, 3, 4].

V Časti 1.4 sa venujeme popisu sietí zo statickými miestami, ktoré modelujú zdroje. Táto trieda sietí bola definovaná v prácach [6, 32, 33] pod názvom workflow siete s obmedzenými zdrojmi (anglicky resource constrained workflow nets). V porovnaní s prácami [6, 32, 33] formalizujeme skutočnosť, že zdroje sú trvácne, pomocou komplementárnych miest [25] pre statické miesta. Tieto komplementárne miesta nazývame

určujúce miesta statických miest a takéto siete nazývame určené workflow siete. Analogicky s prácou [4] definujeme korektné správanie (anglicky soundness) ako schopnosť ukončiť korektne jednu inštanciu.

V Časti 1.5 popisujeme úpravou definície z článkov [41, 42, 43] model vykonávanej siete pre workflow sieť so statickými miestami pre ľubovoľný počet inštancií. Vykonávané siete sú ekvivalentné sieťam s identifikačnými číslami používaným v práci [32]. Siete s identifikačnými číslami pre značky jednotlivých inštancií sú špeciálnou podtriedou farebných Petriho sietí, definovaných napríklad v prácach [37, 38, 39, 40]. V Časti 1.6 formalizujeme pojem uviaznutia vykonávanej siete.

Kapitola 2 je venovaná hlavnej časti vlastného výskumu - detekcii uviaznutí vo vykonávanej sieti pre určenú workflow sieť so statickými miestami, ktorá má korektné správanie.

V prvej časti tejto kapitoly, teda v Časti 2.1 pridávame k workflow sieťam so statickými miestami konštruktor a ukazujeme, že pridanie konštruktora nestačí na detekciu uviaznutí. Ilustrujeme situáciu, kde sieť s konštruktorom neodhalí existujúce uviaznutie vykonávanej siete a naopak, situáciu, kde sieť s konštruktorom odhalí uviaznutie neexistujúce vo vykonávanej sieti.

V Časti 2.2 definujeme siete dosiahnuteľnosti. Následne konštruujeme algoritmus, ktorý v prípade, keď vstupná určená workflow sieť so statickými miestami má korektné správanie, vytvorí sieť dosiahnuteľnosti, inak vráti informáciu, že vstupná určená workflow sieť nemá korektné správanie. Definujeme taktiež uviaznutia v sieti dosiahnuteľnosti.

V Časti 2.3 ukazujeme, že značkovania siete dosiahnuteľnosti zachytávajú pre všetky dosiahnuteľné značkovania dynamických miest originálnej určenej workflow siete informáciu o počte inštancií, ktoré sú v danom značkovaní originálnej určenej workflow siete. Značkovania vykonávanej siete teda zodpovedajú značkovaniam siete dosiahnuteľnosti. V Časti 2.3 dokazujeme prvý hlavný výsledok tejto práce - dokazujeme, že značkovanie vykonávanej siete je uviaznutím vykonávanej siete práve vtedy, keď jemu zodpovedajúce značkovanie siete dosiahnuteľnosti je uviaznutím v sieti dosiahnuteľnosti.

V Časti 2.4 definujeme základné uviaznutia siete dosiahnuteľnosti ako uviaznutia, v ktorých sú označkované spomedzi nestatických miest iba takzvané kritické miesta. V Časti 2.4 dokazujeme druhý hlavný výsledok práce - dokazujeme, že ak má sieť

dosiahnuteľnosti uviaznutie, má zodpovedajúce základné uviaznutie.

V Časti 2.5 dokazujeme, že kritické miesta sú ohraničené a teda počet základných uviaznutí musí byť konečný. Pre ľubovoľné kladné celé číslo definujeme n-obmedzenú sieť dosiahnuteľnosti ako ohraničenú sieť, ktorá simuluje správanie siete dosiahnuteľnosti pre n inštancií. Ukazujeme dva spôsoby, ako vypočítať horné ohraničenie kritických miest. V Časti 2.5 dokazujeme tretí hlavný výsledok tejto práce - dokazujeme, že sieť dosiahnuteľnosti má základné uviaznutie práve vtedy, ak má zodpovedajúce základné uviaznutie ohraničená n-obmedzená sieť dosiahnuteľnosti, pričom n je ohraničenie počtu značiek v kritických miestach.

V Časti 2.6 zostavíme finálny algoritmus na detekciu základných uviaznutí siete dosiahnuteľnosti prostredníctvom detekcie základných uviaznutí ohraničenej Petriho siete - teda n-obmedzenej siete dosiahnuteľnosti.

Kapitola 3 sa venuje vzťahu prezentovaných vlastných výsledkov s príbuznými prácami a zároveň diskusii možných smerov budúceho výskumu.

Práca je uzatvorená Záverom, ktorý sumarizuje dosiahnuté výsledky.

Kapitola 1

Diskrétne udalostné systémy s inštanciami a zdieľanými zdrojmi

1.1 Prechodové systémy

Dynamické systémy sú charakterizované zmenou stavových veličín v čase. Klasickým príkladom sú dynamické systémy spojitej premennej (spojitých premenných), v angličtine známe pod označením Continuous variable dynamic systems. Diskrétne udalostné systémy (ďalej aj DUS) predstavujú triedu dynamických systémov, ktorých stavy sa menia na základe uskutočnenia udalostí v diskrétnych časových okamihoch. V tejto práci sa budeme venovať podtriede DUS nazývanej v literatúre logické diskrétne udalostné systémy. Logické diskrétne udalostné systémy (ďalej aj LDUS), abstrahujú od času ako miery a uvažujú iba kauzálne vzťahy medzi jednotlivými udalosťami. Najjednoduchším spôsobom, ako modelovať LDUS, je použiť orientovaný graf, ktorého vrcholy predstavujú stavy a ktorého hrany sú označené udalosťami. Hrana zo stavu s do stavu s' označená udalosťou e v takomto grafe predstavuje uskutočnenie udalosti e v stave s, ktoré zmení stav s na stav s'. Takýto model LDUS je možné matematicky formalizovať niekoľkými spôsobmi, najznámejšie sú označené prechodové systémy a automaty [16]. V tejto práci budeme používať ako základný model LDUS označené prechodové systémy [45, 80]. Nech S a E sú ľubovoľné množiny. Potom $S \times E \times S$ označuje množinu všetkých trojíc, v ktorých prvý prvok patrí do množiny S, druhý prvok patrí do množiny E a tretí prvok patrí opäť do množiny S, teda $S \times E \times S$ označuje kartézsky súčin množín S, E a S. Označený prechodový systém je formálne definovaný množinou

stavov S, množinou udalostí E a prechodovou reláciou $\longrightarrow$. Prechodová relácia $\longrightarrow$ je podmnožinou $S \times E \times S$. Jej prvkami sú teda trojice (s, e, s').

Definícia 1 (Označený prechodový systém)
Označený prechodový systém je usporiadaná trojica $(S, E, \longrightarrow)$, *kde*

- S *je množina stavov,*
- E *je množina udalostí,*
- $\longrightarrow \subseteq S \times E \times S$ *je prechodová relácia.*

Skutočnosť, že (s, e, s') *patrí do* $\longrightarrow$ *označujeme ako* $s \xrightarrow{e} s'$.

Budeme predpokladať, že označený prechodový systém má určený počiatočný stav a každý stav označeného prechodového systému je z tohto stavu dosiahnuteľný. Pre takýto označený prechodový systém budeme používať pomenovanie označený prechodový systém s počiatočným stavom.

Na formálnu definíciu dosiahnuteľnosti budeme využívať pojem postupnosť. Intuitívne je postupnosť prvkov z danej množiny reťazec prvkov tejto množiny, často sa formalizuje tak, že prvkom postupnosti sa priradí celé číslo označujúce poradie daného prvku v reťazci. Pre účely tejto práce budeme formalizovať postupnosť prvkov množiny tak, že poradovým číslam jednotlivých prvkov priradíme tieto prvky. Keďže chceme definíciu postupnosti formalizovať tak, aby sme vedeli vyjadriť prázdnu postupnosť, konečné neprázdne postupnosti a nekonečné postupnosti, definujeme najskôr prípustné množiny poradových čísel, ktoré budeme nazývať množiny indexov. Množiny indexov budú teda konečné množiny kladných celých čísel od 1 do n, prázdna množina a množina všetkých celých čísel. Formálne ich definujeme nasledovne:

Definícia 2 (Množina indexov)
Množina indexov $\mathbb{I}$ *je podmnožina kladných celých čísel taká, ktorá spĺňa: ak kladné celé číslo* i *patrí do množiny indexov* $\mathbb{I}$, *potom každé kladné celé číslo menšie ako* i *patrí taktiež do množiny indexov* $\mathbb{I}$. *Ak* $\mathbb{I}$ *je konečná neprázdna množina s najväčším prvkom* n *potom číslo* n *označujeme* $max_{\mathbb{I}}$. *Ak* $\mathbb{I}$ *je prázdna množina, potom* $max_{\mathbb{I}} = 0$.

Definícia 3 (Postupnosť)
Nech $\mathbb{I}$ *je množina indexov a* S *je množina. Potom funkciu* $\alpha : \mathbb{I} \to S$, *ktorá priradí*

každému indexu i z množiny indexov $\mathbb{I}$ prvok $\alpha(i)$ z množiny S, sa nazýva postupnosť prvkov z množiny S. Ak $\mathbb{I}$ je konečná množina, potom hovoríme, že α je konečná postupnosť dĺžky $max_{\mathbb{I}}$. Špeciálne, ak $\mathbb{I}$ je prázdna množina, potom hovoríme, že α je prázdna postupnosť. Ak $\mathbb{I}$ je rovná množine všetkých kladných celých čísel, potom hovoríme, že α je nekonečná postupnosť.

Pre ilustráciu, uvažujme postupnosť znakov $\alpha = abba$ z množiny znakov $S = \{a, b\}$. Intuitívne rozumieme, že je to konečná postupnosť zo 4 prvkami. V súlade s definíciou 2 použijeme indexovú množinu $\mathbb{I} = \{1, 2, 3, 4\}$. Postupnosť α je potom formalizovaná nasledovne: $\alpha(1) = a$, $\alpha(2) = b$, $\alpha(3) = b$ a $\alpha(4) = a$, teda prvý prvok postupnosti α je znak a, druhým prvkom znak b, tretím prvkom opäť znak b a štvrtým prvkom znak a.

Definícia 4 (Spustiteľná postupnosť, dosiahnuteľnosť)
Nech $(S, E, \longrightarrow)$ je označený prechodový systém. Nech $s \in S$ je stav a nech $\epsilon : \mathbb{I} \to E$ je konečná postupnosť udalostí. Postupnosť ϵ je spustiteľná v stave s práve vtedy, keď existuje funkcia $\sigma : \mathbb{I} \cup \{0\} \to S$ taká, že $\sigma(0) = s$ a pre každé kladné celé číslo $i \in \mathbb{I}$: $\sigma(i-1) \xrightarrow{\epsilon(i)} \sigma(i)$. Spustenie postupnosti ϵ v stave s vedie do stavu $\sigma(max_{\mathbb{I}})$.

Stav $s' \in S$ je dosiahnuteľný zo stavu s práve vtedy keď existuje postupnosť ϵ spustiteľná v stave s, ktorej spustenie vedie do stavu s'.

Skutočnosť, že konečná postupnosť ϵ je spustiteľná v stave s a jej spustenie vedie do stavu s' označujeme ako $s \xrightarrow{\epsilon} s'$.

Pripomeňme, že v zmysle Definície 4 pre ľubovoľný stav s platí: prázdna postupnosť je spustiteľná v stave s a jej spustenie vedie do stavu s, teda stav s je dosiahnuteľný sám zo seba spustením prázdnej postupnosti.

Definícia 5 (Označený prechodový systém s počiatočným stavom)
Označený prechodový systém s počiatočným stavom je usporiadaná štvorica $(S, E, \longrightarrow, q)$, kde $(S, E, \longrightarrow)$ je označený prechodový systém a $q \in S$ je počiatočný stav, pričom každý stav $s \in S$ je dosiahnuteľný z q.

Logické diskrétne udalostné systémy majú veľmi široké uplatnenie. Prakticky všetky informačné systémy, softvérové systémy, ľubovoľné počítačové programy predstavujú príklady LDUS. Podobne komunikačné protokoly, pružné výrobné systémy, vnorené systémy sú príkladom použitia LDUS. Opisy bežných situácií, návody a predpisy, popis

správania v ľudskej reči taktiež využívajú črty udalostných systémov. Keďže LDUS sa využívajú v rôznych aplikačných oblastiach, ich rozvoj a nástroje na ich analýzu sú predmetom skúmania v rôznych vedných odboroch. Okrem automatizácie a riadenia a kybernetiky, kde sa skúmajú pod názvom LDUS, sa rozvíjajú najmä v informatike ako výpočtové modely, a zároveň v oblasti manažmentu a riadenia ako workflow procesy.

1.2 Petriho siete

V tejto práci budeme uvažovať LDUS, ktorých stavy sú štrukturované. Budeme uvažovať množinu stavových premenných, pričom stavové premenné môžu nadobúdať rôzne hodnoty. Stav LDUS bude daný ako funkcia, ktorá priraďuje každej stavovej premennej aktuálnu hodnotu. Budeme uvažovať situáciu, keď počet hodnôt, ktoré môže nadobúdať stavová premenná, je spočitateľný. Pre zjednodušenie budeme uvažovať ako hodnoty stavovej premennej nezáporné celé čísla. Stav stavovej premennej bude teda daný nezáporným celým číslom. Toto číslo môže reprezentovať počet zdrojov určitého typu, splnenie (nenulová hodnota), respektíve nesplnenie (nulová hodnota) podmienky. Zmena stavu spôsobená uskutočnením udalosti bude definovaná pomocou vstupnej a výstupnej funkcie. Vstupná funkcia priradí každej stavovej premennej a každej udalosti nezáporné celé číslo. Zároveň výstupná funkcia priradí každej stavovej premennej a každej udalosti nezáporné celé číslo. Vstupná funkcia udáva minimálnu hodnotu jednotlivých stavových premenných nevyhnutnú na to, aby sa daná udalosť mohla uskutočniť. Uskutočnenie udalosti zmení hodnoty každej stavovej premennej tak, že odpočíta od aktuálneho stavu hodnotu danú vstupnou funkciou a pripočíta hodnotu danú výstupnou funkciou.

Takýto modelovací formalizmus je v literatúre známy ako Petriho sieť [44, 62, 63, 67, 23, 24]. Petriho siete sú pomenované podľa Carla Adama Petriho [64]. V Petriho sieti nazývame stavové premenné miesta a udalosti nazývame prechody. Formálne je Petriho sieť definovaná nasledovne.

Definícia 6 (Petriho sieť)
Petriho sieť je usporiadaná štvorica (P, T, I, O)*, kde:*

- P *je množina miest*
- T *je množina prechodov*
- *prienik množiny* P *a množiny* T *je prázdny, t.j.* $P \cap T = \emptyset$
- I *je vstupná funkcia, ktorá každej dvojici, v ktorej prvý prvok je miesto a druhý prvok je prechod (t.j. každej dvojici, ktorá patrí do kartézskeho súčinu* $P \times T$*) priradí nezáporné celé číslo, formálne* $I : P \times T \to \mathbb{N}$*, kde* $\mathbb{N}$ *označuje množinu nezáporných celých čísel.*

- *O je výstupná funkcia, ktorá každej dvojici, v ktorej prvý prvok je miesto a druhý prvok je prechod, priradí nezáporné celé číslo, formálne $O : P \times T \to \mathbb{N}$.*

Stav siete bude daný značkovaním, teda funkciou $m : P \to \mathbb{N}$, ktorá priradí každému miestu nezáporné celé číslo určujúce počet značiek, ktoré sa v danom mieste nachádzajú.

Definícia 7 (Značkovanie)
Nech $PN = (P, T, I, O)$ je Petriho sieť. Funkciu $m : P \to \mathbb{N}$, ktorá priradí ku každému miestu nezáporné celé číslo, nazývame značkovanie Petriho siete PN. Hodnota $m(p)$ definuje počet značiek v mieste $p \in P$. Značkovanie budeme označovať výrazom v tvare sumy značiek v miestach, teda $\sum_{p \in P} m(p)p$. Množinu miest, pre ktoré $m(p)$ je nenulové, nazývame nosič značkovania m a označujeme $sup(m)$, formálne teda pre každé $p \in P$ platí, že p patrí do $sup(m)$ práve vtedy keď $m(p) > 0$.

Miesta môžu byť teda chápané ako typy značiek. Značkovanie budeme označovať výrazom v tvare sumy značiek daných typov.

Ak napríklad množina miest je daná ako $P = \{a, b, c\}$, máme 3 typy značiek, značky typu a, značky typu b a značky typu c. Uvažujme ďalej napríklad značkovanie $m : P \to \mathbb{N}$, ktoré priradí miestu a počet značiek $m(a) = 2$, miestu b počet značiek $m(b) = 0$ a miestu c počet značiek $m(c) = 1$. To znamená, že značkovanie vyjadruje stav s dvoma značkami typu a, žiadnou značkou typu b a jednou značkou typu c, čo môžeme zapísať výrazom $2a + 0b + 1c$, alebo výrazom $2a + c$ (teda 2 značky typu a a značka typu c), všeobecne výrazom $\sum_{p \in P} m(p)p$.

Skutočnosť, že pre značkovanie m a značkovanie m' platí $m(p) \leq m'(p)$ pre každé $p \in P$, označujeme $m \leq m'$. Skutočnosť, že $m \leq m'$ a zároveň existuje $p \in P$ také, že $m(p) < m'(p)$, označujeme $m < m'$, resp. $m' > m$ a hovoríme, že m je menšie ako m', resp. m' je väčšie ako m.

Súčet dvoch značkovaní m a m' je značkovanie $m + m'$ také, že $(m + m')(p) = m(p) + m'(p)$ pre každé $p \in P$.

Ak $m \leq m'$, potom rozdiel značkovaní m' a m je značkovanie $m' - m$ také, že $(m' - m)(p) = m'(p) - m(p)$ pre každé $p \in P$.

Dynamika Petriho siete je daná spúšťaním prechodov.

Definícia 8 (Spustiteľnosť prechodov)
Nech $PN = (P, T, I, O)$ je Petriho sieť. Nech $m : P \to \mathbb{N}$ je značkovanie a $t \in T$ je prechod siete PN. Prechod t je spustiteľný v značkovaní m práve vtedy, keď pre každé miesto $p \in P$ platí, že počet značiek $m(p)$ v mieste $p \in P$ nie je menší ako hodnota vstupnej funkcie $I(p, t)$, teda ak platí: $m(p) \geq I(p, t)$.

Ak je prechod t spustiteľný v značkovaní m, potom jeho spustenie v značkovaní m vedie do značkovania m', pričom pre každé miesto $p \in P$ platí:

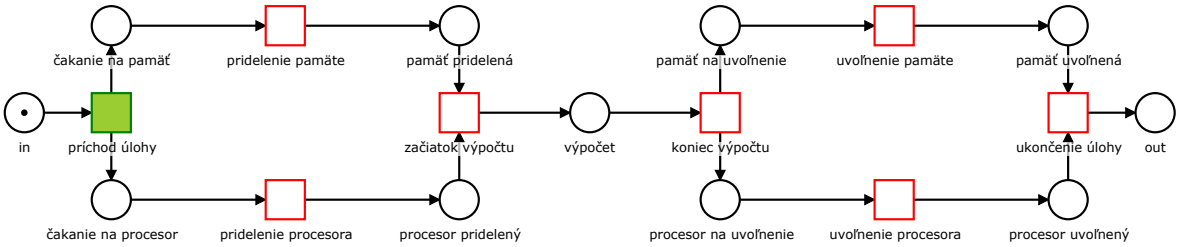
$$m'(p) = m(p) - I(p, t) + O(p, t)$$

Podobne ako v prípade označených prechodových systémov budeme predpokladať, že Petriho sieť má určené počiatočné značkovanie. Pre takúto Petriho sieť budeme používať pomenovanie označkovaná Petriho sieť.

Definícia 9 (Označkovaná Petriho sieť)
Označkovaná Petriho sieť je usporiadaná pätica $MPN = (P, T, I, O, m_0)$, kde $PN = (P, T, I, O)$ je Petriho sieť a m_0 je jej značkovanie nazývané počiatočné značkovanie.

V práci budeme využívať jednoduchý ilustratívny príklad, ktorý čitateľovi pomôže rýchlejšie priblížiť jednotlivé definované pojmy. Ilustratívny príklad popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy: výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí. Systém môžeme modelovať označkovanou Petriho sieťou na Obr. 1.1.

V grafickom znázornení budú miesta modelované formou kruhov, značkovanie v rozmedzí 0 až 9 počtom značiek vo vnútri kruhu, väčšie značkovania príslušným číslom vo vnútri kruhu. Prechody budú modelované štvorcami, pričom spustiteľné prechody budú zvýraznené zelenou farbou, resp. budú vyplnené, zatiaľ čo prechody, ktoré nebudú spustiteľné v danom značkovaní, budú zvýraznené červenou farbou, resp. budú bez výplne. Nenulovým hodnotám vstupnej funkcie budú zodpovedať orientované hrany ukončené šípkou smerujúce z príslušného miesta do prechodu, pričom hodnota väčšia ako 1 bude vyjadrená číslom vedľa hrany. Obdobne, nenulovým hodnotám výstupnej funkcie budú zodpovedať orientované hrany ukončené šípkou smerujúce z príslušného prechodu do miesta, pričom hodnota väčšia ako 1 bude vyjadrená číslom vedľa hrany.



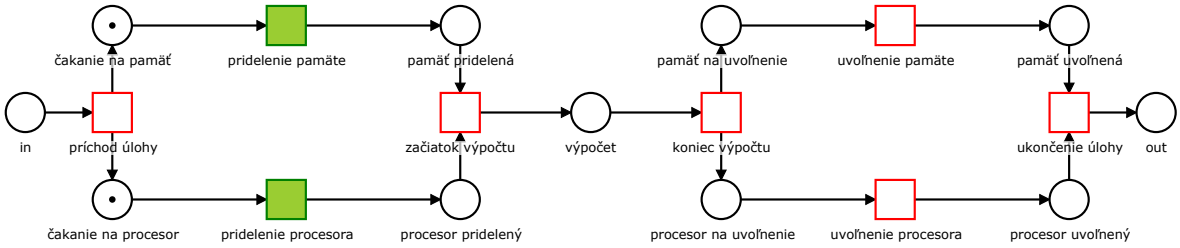
Obr. 1.1: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy

Nenulovú hodnotu vstupnej, resp. výstupnej funkcie nazývame váhou hrany. Všetky Petriho siete použité v práci boli modelované vo vlastnom online editore Petriho sietí PetriFlow, ktorý je prístupný na stránke `www.petriflow.com`. Editor je implementovaný v skriptovacom jazyku JavaScript a značkovacom jazyku HTML5, pričom grafické znázornenie je realizované prostredníctvom SVG (Scalar Vector Graphics) komponentov. Editor umožňuje modelovať Petriho siete, ukladať a načítavať ich vo formáte XML. Editor zároveň umožňuje exportovať súbory s grafickým znázornením sietí vo formáte SVG.

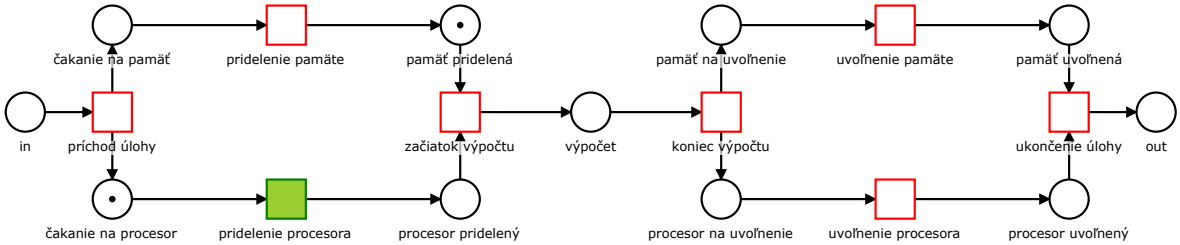
V značkovaní Petriho siete in na Obr. 1.1 je jediným spustiteľným prechodom prechod *príchod úlohy*. Jeho spustenie vedie do značkovania *čakanie na pamäť* + *čakanie na procesor* znázornenom na Obr. 1.2, v ktorom sú spustiteľné prechody zvýraznené zelenou farbou *pridelenie pamäte* a *pridelenie procesora*. Ak zvolíme spustenie prechodu *pridelenie pamäte*, dostaneme sa do značkovania na Obr. 1.3, v ktorom je spustiteľný prechod *pridelenie procesora*. Po jeho spustení dosiahneme stav na Obr. 1.4 so spustiteľným prechodom *začiatok výpočtu*. Spustenie prechodu *začiatok výpočtu* vedie do značkovania na Obr. 1.5 prezentujúcom výpočet. Spustením prechodu *koniec výpočtu* sa dostaneme do značkovania na Obr. 1.6. V tomto značkovaní je možné spustiť prechody *uvoľnenie pamäte* a *uvoľnenie procesora*. Po uvoľnení pamäte sa dostaneme do značkovania na Obr. 1.7. Následne po uvoľnení procesora dosiahneme značkovanie na Obr. 1.8, v ktorom spustením úlohy ukončíme inštanciu a dostaneme sa do finálneho značkovania na Obr. 1.9.

Petriho sieť definuje prirodzeným spôsobom označený prechodový systém, nazývaný graf dosiahnuteľnosti.

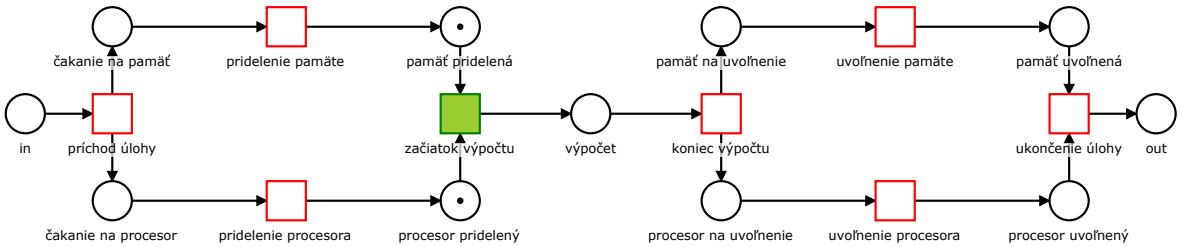
Definícia 10 (Graf dosiahnuteľnosti Petriho siete)
Nech $PN = (P, T, I, O)$ *je Petriho sieť. Označený prechodový systém* $(S, E, \longrightarrow)$, *kde*



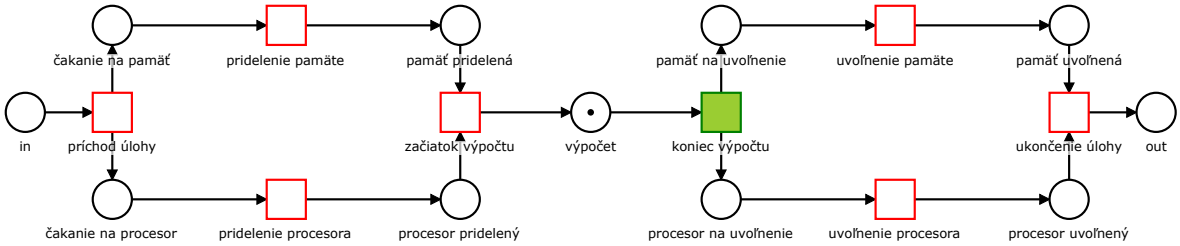
Obr. 1.2: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po spustení prechodu *príchod úlohy*



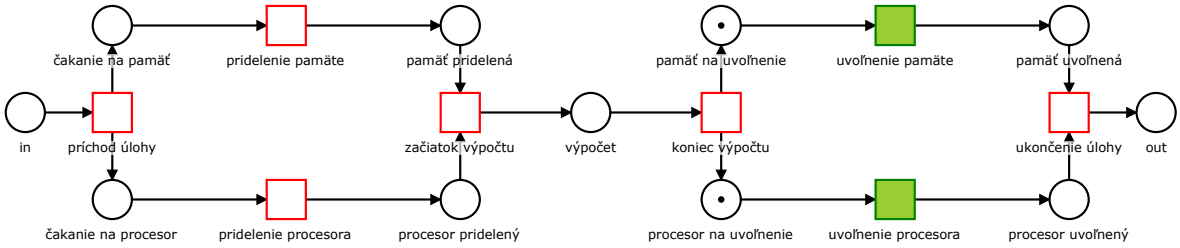
Obr. 1.3: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po pridelení pamäte



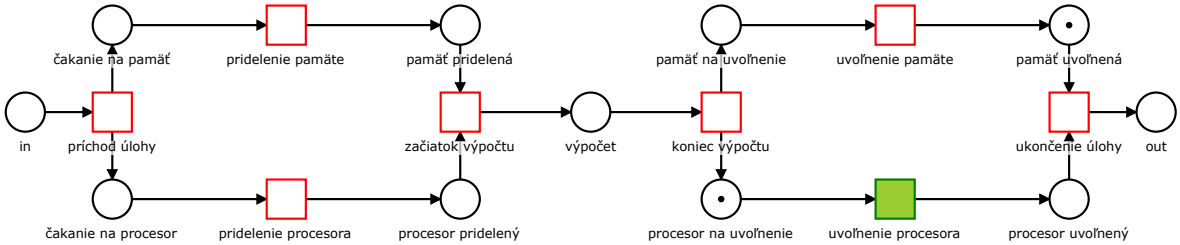
Obr. 1.4: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po pridelení pamäte a procesora



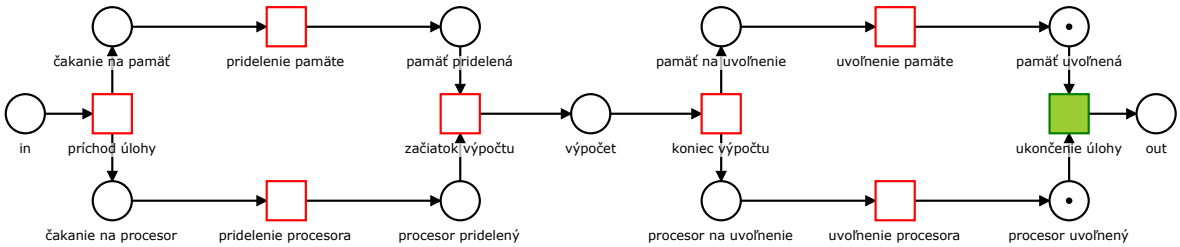
Obr. 1.5: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po začatí výpočtu



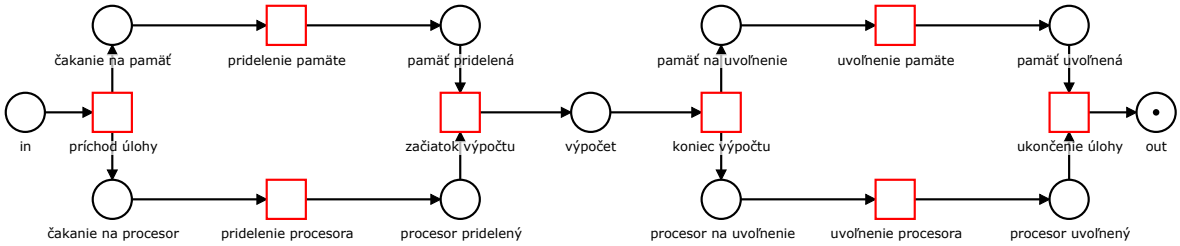
Obr. 1.6: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po ukončení výpočtu



Obr. 1.7: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po uvoľnení pamäte



Obr. 1.8: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po uvoľnení procesora



Obr. 1.9: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy po ukončení úlohy

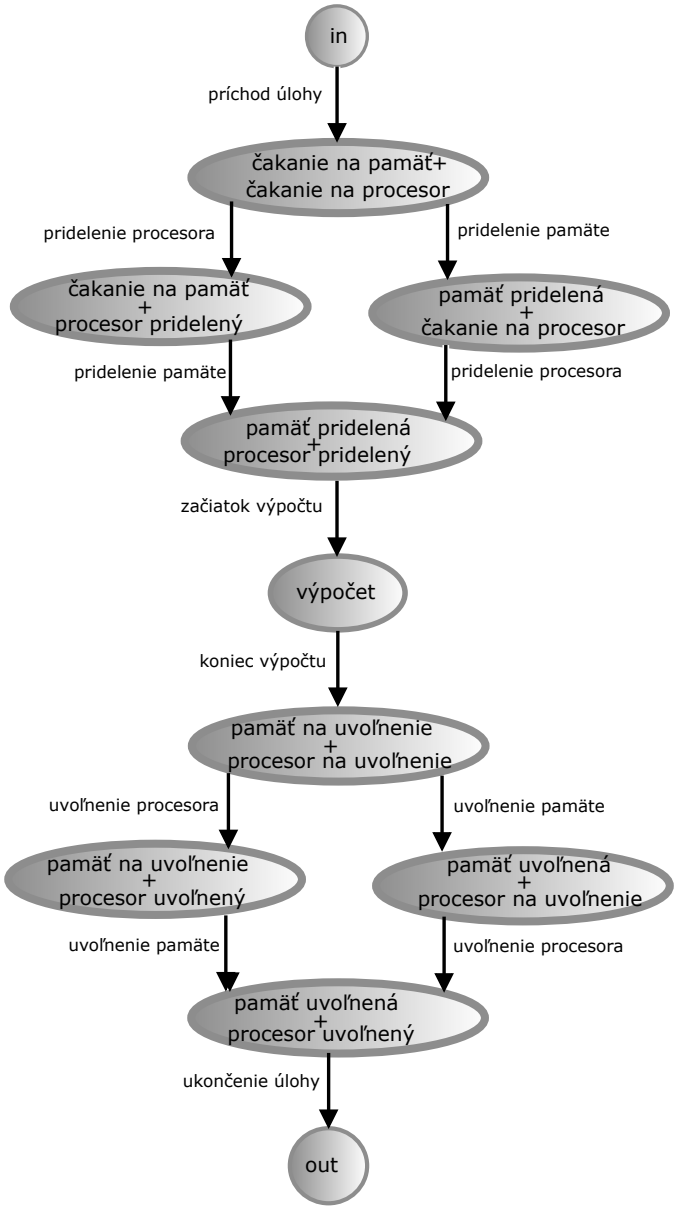
- *S je množina všetkých značkovaní, teda množina všetkých funkcií z P do množiny nezáporných celých čísel $\mathbb{N}$,*
- *$E = T$,*
- *$m \xrightarrow{t} m'$ práve vtedy, keď prechod t je spustiteľný v značkovaní m a jeho spustenie vedie do značkovania m',*

nazývame graf dosiahnuteľnosti Petriho siete PN.

Grafom dosiahnuteľnosti označkovanej Petriho siete bude označený prechodový systém s počiatočným stavom rovným počiatočnému značkovaniu m_0. Graf dosiahnuteľnosti označkovanej Petriho siete $MPN = (P, T, I, O, m_0)$ je možné definovať pomocou grafu dosiahnuteľnosti $(S, E, \longrightarrow)$ Petriho siete bez počiatočného značkovania $PN = (P, T, I, O)$ tak, že budeme uvažovať ako stavy grafu dosiahnuteľnosti označkovanej Petriho siete iba značkovania dosiahnuteľné z počiatočného značkovania (označíme ich $[m_0\rangle$) a v prechodovej relácii budeme uvažovať iba tie trojice (stav, udalosť, stav), v ktorých sú prvý a posledný prvok dosiahnuteľné značkovania, teda budeme uvažovať ako prechodovú reláciu prienik pôvodnej prechodovej relácie $\longrightarrow$ a všetkých trojíc z $[m_0\rangle \times E \times [m_0\rangle$. Prechodová relácia grafu dosiahnuteľnosti označkovanej Petriho siete bude teda daná ako $\longrightarrow \cap([m_0\rangle \times E \times [m_0\rangle)$.

Definícia 11 (Graf dosiahnuteľnosti označkovanej Petriho siete)
Nech $MPN = (P, T, I, O, m_0)$ je označkovaná Petriho sieť. Nech $(S, E, \longrightarrow)$ je graf dosiahnuteľnosti Petriho siete $PN = (P, T, I, O)$. Nech $[m_0\rangle$ označuje množinu všetkých značkovaní dosiahnuteľných zo značkovania m_0 v grafe dosiahnuteľnosti Petriho siete PN. Potom označený prechodový systém s počiatočným stavom $([m_0\rangle, E, \longrightarrow \cap([m_0\rangle \times E \times [m_0\rangle), m_0)$ nazývame graf dosiahnuteľnosti označkovanej Petriho siete MPN. Ak je postupnosť ϵ spustiteľná v značkovaní m a jej spustenie vedie do značkovania m' v jej grafe dosiahnuteľnosti, hovoríme, že postupnosť ϵ je spustiteľná z m v sieti MPN, jej spustenie vedie do značkovania m' v sieti MPN a m' je dosiahnuteľné z m v sieti MPN.

Na Obr. 1.10 je znázornený graf dosiahnuteľnosti označkovanej Petriho siete z Obr. 1.1.



Obr. 1.10: Graf dosiahnuteľnosti označkovanej Petriho sieť modelujúcej spracovanie výpočtovej úlohy

Definícia 12 (Ohraničenosť)
Označkovaná Petriho sieť $MPN = (P, T, I, O, m_0)$ *je ohraničená práve vtedy, ak existuje také značkovanie* $b : P \to \mathbb{N}$*, že pre každé značkovanie* m *dosiahnuteľné z* m_0 *platí* $m \leq b$*. Značkovanie* b *nazývame ohraničenie siete* MPN*.*

V prípade, že Petriho sieť má konečný počet miest a prechodov a počet dosiahnuteľných značkovaní grafu dosiahnuteľnosti označkovanej Petriho siete je konečný, môžeme graf dosiahnuteľnosti skonštruovať. Pre Petriho sieť s konečným počtom miest a prechodov je konečný počet dosiahnuteľných značkovaní rovnocenný s ohraničenosťou Petriho siete.

Dôsledok 1 *Označkovaná Petriho sieť* $MPN = (P, T, I, O, m_0)$ *s konečným počtom miest a prechodov je ohraničená práve vtedy, ak počet značkovaní dosiahnuteľných z* m_0 *v jej grafe dosiahnuteľnosti je konečný.*

Z Dicksonovej lemy [29] vyplýva nasledujúci výsledok.

Lema 1 *Označkovaná Petriho sieť* $MPN = (P, T, I, O, m_0)$ *s konečným počtom miest a prechodov má nekonečný počet stavov práve vtedy, ak existujú značkovania* m *a* m' *také, že* m *je dosiahnuteľné z* m_0*,* m' *je dosiahnuteľné z* m *a* $m < m'$*.*

Výstupom nasledujúceho algoritmu v prípade ohraničenej označkovanej Petriho siete s konečným počtom miest a prechodov je jej graf dosiahnuteľnosti, v prípade, ak sieť nie je ohraničená, výstupom je informácia o neohraničenosti. Skutočnosť, že algoritmus terminuje je dôsledkom lemy 1.

Algoritmus 1 (Overenie ohraničenosti, graf dosiahnuteľnosti MPN)
Pre označkovanú Petriho sieť $MPN = (P, T, I, O, m_0)$ s konečným počtom miest a prechodov

1. vytvor prázdny zoznam M nájdených značkovaní
2. vytvor prázdny zoznam H označených hrán
3. vlož do zoznamu M počiatočné značkovanie m_0 a nastav množinu $Pre(m_0)$ jeho predchodcov prázdnu

4. pokiaľ je v zozname M značkovanie m, ktoré nie je označené ako preskúmané, vezmi ho a rob nasledovné:

   (a) pre každý prechod t spustiteľný z m

      i. počítaj značkovanie m' dosiahnuté spustením prechodu t zo značkovania m

      ii. ak m' ešte nie je v zozname M, vlož ho do zoznamu M a nastav množinu $Pre(m')$ jeho predchodcov prázdnu

      iii. nastav množinu $Pre(m')$ predchodcov značkovania m' rovnú zjednoteniu $Pre(m') \cup Pre(m) \cup \{m\}$

      iv. ak existuje predchodca m'' z $Pre(m')$ taký, že pre všetky miesta $p \in P$ platí $m''(p) \leq m'(p)$ a zároveň existuje miesto $p \in P$ také, že platí $m''(p) < m'(p)$, potom algoritmus zastav a vráť informáciu "Sieť je neohraničená"

      v. vlož do zoznamu H hranu z m do m' označenú prechodom t, teda trojicu (m, t, m')

   (b) označ m ako preskúmané

5. vráť graf dosiahnuteľnosti (M, T, H, m_0) a informáciu "Sieť je ohraničená"

Označkovaná Petriho sieť na Obr. 1.1 je ohraničená.

1.3 Workflow siete

Veľmi častým typom LDUS sú systémy, pri ktorých systém spracováva viacero inštancií rovnakého typu. Typickým príkladom takýchto LDUS sú napríklad workflow procesy, pri ktorých sa inštancie označujú ako prípady, napr. systém spravujúci poisťovacie prípady. Iným príkladom sú webové aplikácie, kde inštanciami sú „sessions“, objektovo-orientované programy, kde inštanciami sú objekty danej triedy. Objektovo orientovaný princíp dnes predstavuje základný kameň životného cyklu produktu v koncepte Industry 4.0, kde výrobok - produkt sa stáva inštanciou udalostného systému – výrobného procesu. Takéto LDUS budeme nazývať LDUS s inštanciami. Zameriame sa na systémy, kde inštancie majú jednoznačný začiatok a korektné ukončenie. V literatúre sú takéto systémy modelované pomocou workflow sietí, ktoré predstavujú špeciálnu podtriedu označkovaných Petriho sietí [1, 2, 3, 4]. Jednoznačný začiatok vo workflow sieťach je vyjadrený pomocou špeciálneho miesta, pre ktoré je hodnota výstupnej funkcie nulová pre ľubovoľný prechod a hodnota vstupnej funkcie nenulová aspoň pre jeden prechod. Podobne ukončenie inštancie je vyjadrené pomocou špeciálneho miesta, pre ktoré je hodnota vstupnej funkcie nulová pre ľubovoľný prechod a hodnota výstupnej funkcie nenulová aspoň pre jeden prechod.

Definícia 13 (Workflow sieť)
Workflow sieť je Petriho sieť $PN = (P, T, I, O)$ *s konečným počtom miest a prechodov v ktorej existuje práve jedno miesto* $in \in P$ *také a práve jedno miesto* $out \in P$ *také, že*

- *pre všetky* $t \in T$ *platí* $O(in, t) = 0$ *a zároveň existuje také* $t \in T$ *že* $I(in, t) \neq 0$,
- *pre všetky* $t \in T$ *platí* $I(out, t) = 0$ *a zároveň existuje také* $t \in T$ *že* $O(out, t) \neq 0$.

Miesto in je nazývané vstupné miesto workflow siete, miesto out je nazývané výstupné miesto workflow siete.

Označkovaná workflow sieť je označkovaná workflow sieť so špeciálnym počiatočným značkovaním, v ktorom je vstupné miesto označené práve jednou značkou a všetky ostatné miesta sú prázdne.

Definícia 14 (Označkovaná workflow sieť)
Označkovaná workflow sieť je taká označkovaná Petriho sieť $MPN = (P, T, I, O, m_0)$,

že $PN = (P, T, I, O)$ *je workflow sieť a* $m_0 = in$*, teda* $m_0(in) = 1$ *a* $m_0(p) = 0$ *pre každé* $p \in P$*, ktoré je rôzne od in.*

Príkladom označkovanej workflow siete je sieť na Obr. 1.1.

Možné správanie sa jednotlivých inštancií workflow procesu je reprezentované spustiteľnými postupnosťami v grafe dosiahnuteľnosti označkovanej workflow siete. Aby mali inštancie korektné ukončenie, je potrebné, aby graf dosiahnuteľnosti označkovanej workflow siete mal práve jeden konečný stav, teda stav, v ktorom nie je spustiteľný žiaden prechod. Týmto stavom musí byť značkovanie *out*, teda značkovanie, pri ktorom miesto *out* obsahuje práve jednu značku a žiadne iné miesto značky neobsahuje. Toto je zároveň jediné dosiahnuteľné značkovanie, v ktorom miesto out obsahuje značku. Inými slovami, korektné ukončenie inštancie označkovanej workflow siete je vyjadrené presunom značky zo vstupného do výstupného miesta. Takúto vlastnosť budeme nazývať korektné správanie (v literatúre sa táto vlastnosť označuje soundness) [4].

Definícia 15 (Korektné správanie označkovanej workflow siete)
Nech $MPN = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť a nech* $in \in P$ *označuje vstupné miesto a* $out \in P$ *výstupné miesto. Označkovaná workflow sieť* MPN *má korektné správanie práve vtedy, keď pre každé značkovanie* m *dosiahnuteľné z* m_0 *platí:*

- *značkovanie out je dosiahnuteľné zo značkovania* m*,*
- *ak* $m(out) \geq 1$ *potom* $m = out$*, teda* $m(out) = 1$ *a* $m(p) = 0$ *pre každé* $p \in P$*, ktoré je rôzne od out.*

Pre označkované workflow siete s korektným správaním platí nasledujúci výsledok.

Lema 2 *Ak označkovaná workflow sieť má korektné správanie, potom počet stavov dosiahnuteľných z počiatočného značkovania je konečný.*

Dôkaz. Nech počet značkovaní dosiahnuteľný z počiatočného značkovania m_0 nie je konečný. Pripomeňme, že značkovania sú vlastne n-tice nezáporných celých čísel, kde n je počet miest workflow siete. Podľa Dicksonovej lemy [29], v každej nekonečnej množine n-tíc nezáporných celých čísel musí existovať dvojica m a m', taká, že m je menšie ako m'. Nech m a m' sú značkovania dosiahnuteľné z m_0 také, že m je menšie ako m'. Z definície korektného správania vyplýva, že $m' = out$ alebo $m'(out) = 0$ a že $m = out$ alebo $m(out) = 0$.

- Kombinácia $m' = out$ a $m = out$ je v rozpore s predpokladom, že m' je väčšie ako m.

- Kombinácia $m'(out) = 0$ a $m = out$ je taktiež v rozpore s predpokladom, že m' je väčšie ako m.

- Kombinácia $m' = out$ a $m(out) = 0$ znamená pri predpoklade m' je väčšie ako m, že $m = 0$ ($m(p) = 0$ pre každé $p \in P$). Zároveň z prvej odrážky definície korektného správania dostávame, že značkovanie out je dosiahnuteľné zo značkovania m spustením nejakej postupnosti prechodov. Z definície spustiteľnosti prechodov je zrejmé, že ak je postupnosť prechodov spustiteľná z nejakého značkovania, potom je spustiteľná z každého väčšieho značkovania, čiže aj zo značkovania m'. Ak spustenie tejto postupnosti vedie zo značkovania m do značkovania out, potom spustenie tej istej postupnosti prechodov zo značkovania m' vedie do značkovania $m'' = out + (m' - m) = out + out = 2out$, čo je v rozpore s druhou odrážkou definície korektného správania.

- Kombinácia $m'(out) = 0$ a $m(out) = 0$ znamená pri predpoklade m' je väčšie ako m, že existuje $p \in P$ rôzne od out také, že $m'(p) > m(p)$. Zároveň z prvej odrážky definície korektného správania dostávame, že značkovanie out je dosiahnuteľné zo značkovania m spustením nejakej postupnosti prechodov. Z definície spustiteľnosti prechodov je zrejmé, že ak je postupnosť prechodov spustiteľná z nejakého značkovania, potom je spustiteľná z každého väčšieho značkovania, čiže aj zo značkovania m'. Ak spustenie tejto postupnosti vedie zo značkovania m do značkovania out, potom spustenie tej istej postupnosti prechodov zo značkovania m' vedie do značkovania $m'' = m' + (out - m)$, a teda $m''(out) = 0 + (1 - 0) = 1$. Keďže zároveň $m'(p) > m(p)$ dostávame $m''(p) > 0$, čo je v rozpore s druhou odrážkou definície korektného správania.

To znamená, že ak označkovaná workflow sieť má korektné správanie, potom počet stavov dosiahnuteľných z počiatočného značkovania je konečný. □

Nutnou podmienkou korektného správania je teda konečný počet stavov dosiahnuteľných z počiatočného značkovania siete. Overiť, či označkovaná workflow sieť má korektné správanie je možné jednoduchou modifikáciou Algoritmu 1.

Algoritmus 2 (Overenie korektného správania workflow siete)

Pre označkovanú workflow sieť $MPN = (P, T, I, O, m_0)$

1. vytvor prázdny zoznam M nájdených značkovaní
2. vytvor premennú *out je dosiahnuteľné* a vlož do nej hodnotu *nie*
3. vytvor prázdny zoznam H označených hrán
4. vlož do zoznamu M počiatočné značkovanie m_0 a nastav množinu $Pre(m_0)$ jeho predchodcov prázdnu
5. pokiaľ je v zozname M značkovanie m, ktoré nie je označené ako preskúmané, vezmi ho a rob nasledovné:
   (a) ak $m(out) \geq 1$ a zároveň $m \neq out$ potom algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"
   (b) ak $m = out$ vlož do *out je dosiahnuteľné* hodnotu *áno*
   (c) pre každý prechod t spustiteľný z m
      i. počítaj značkovanie m' dosiahnuté spustením prechodu t zo značkovania m
      ii. ak m' ešte nie je v zozname M, vlož ho do zoznamu M a nastav množinu $Pre(m')$ jeho predchodcov prázdnu
      iii. nastav množinu $Pre(m')$ predchodcov značkovania m' rovnú zjednoteniu $Pre(m') \cup Pre(m) \cup \{m\}$
      iv. ak existuje predchodca m'' z $Pre(m')$ taký, že pre všetky miesta $p \in P$ platí $m''(p) \leq m'(p)$ a zároveň existuje miesto $p \in P$ také, že platí $m''(p) < m'(p)$, potom algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"
      v. vlož do zoznamu H hranu z m do m' označenú prechodom t, teda trojicu (m, t, m')
   (d) označ m ako preskúmané
6. ak *out je dosiahnuteľné* = *áno* a $Pre(out) \cup \{out\} = M$, vráť informáciu "Sieť má korektné správanie", inak vráť "Sieť nemá korektné správanie"

Z grafu označkovanej workflow siete na Obr. 1.10 vyplýva, že označkovaná workflow sieť na Obr. 1.1 má korektné správanie. Na Obr. 1.9 je sieť znázornená v značkovaní dosiahnutom spustením každého s prechodov práve raz. V tomto značkovaní je označené značkou iba výstupné miesto *out*. Značkovanie reprezentuje ukončenú výpočtovú úlohu.

1.4 Petriho siete a workflow siete so statickými miestami

Budeme uvažovať, že inštancie môžu zdieľať zdroje, napr. operačnú pamäť či procesor v prípade programov. Takéto LDUS s inštanciami budeme nazývať LDUS s inštanciami a zdieľanými zdrojmi. Predpokladáme, že zdieľané zdroje sú inštanciami používané, nie sú však inštanciami ani vytvárané ani spotrebované. To znamená, že počet zdieľaných zdrojov pred a po ukončení inštancie sa nemení. Takéto LDUS s inštanciami budeme nazývať LDUS s inštanciami a trvácnymi zdieľanými zdrojmi. Typickým príkladom LDUS s inštanciami a trvácnymi zdieľanými zdrojmi sú workflow procesy, kde každý prípad predstavuje vytvorenie inštancie, respektíve webové aplikácie, kde každý prípad predstavuje vytvorenie „session“, inštancie zdieľajú zdroje ako procesor, operačná pamäť, port, ktoré sú používané pri uskutočnení jednotlivých aktivít prípadu.

V závislosti od aplikačnej oblasti, LDUS s inštanciami a trvácnymi zdieľanými zdrojmi je možné ilustrovať na rôznych príkladoch. Pritom je možné použiť rozličnú terminológiu, ktorá je blízka expertom z danej oblasti a umožní im lepšie pochopenie problému. Ako príklad uveďme popis uvažovanej triedy LDUS v terminológii objektovo-orientovaného a udalosťami riadeného programovania (object oriented and event driven programming). Podobne ako trieda v objektovo-orientovanom programovaní predstavuje definíciu, podľa ktorej sa budú jednotlivé inštancie správať, budeme uvažovať definíciu LDUS ako Petriho sieť. Atribútom objektov v definícii triedy budú zodpovedať miesta v Petriho sieti. Koncept zdieľania premenných je v objektovo-orientovaných jazykoch ako napríklad Java, realizovaný prostredníctvom statických premenných. V Petriho sieťach im budú zodpovedať statické miesta. Zmeny v udalosťami riadených programoch sú realizované prostredníctvom ovládača udalostí (event handler). Ak je program v stave, že príslušná udalosť sa môže uskutočniť, tento ovládač pri uskutočnení udalosti zavolá príslušnú funkciu, ktorá zmení stav atribútov. V prípade Petriho sietí, ak aktuálny stav umožňuje uskutočnenie udalosti, teda prechod je spustiteľný, uskutočnenie udalosti zavolá vstupnú a výstupnú funkciu, ktorá odpočíta, resp. pripočíta príslušnú hodnotu.

Formálne budeme LDUS s inštanciami a zdieľanými zdrojmi modelovať pomocou Petriho sietí so statickými miestami [41, 42, 43]. Sú to Petriho siete s pridanými sta-

tickými miestami, ktoré modelujú zdieľané zdroje. Táto trieda sietí bola definovaná v prácach [6, 32, 33] pod názvom workflow siete s obmedzenými zdrojmi (anglicky resource constrained workflow nets).

Definícia 16 (Petriho sieť so statickými miestami)
Petriho sieť so statickými miestami je pätica $PNS = (D, S, T, I, O)$, *kde*

- D *je množina dynamických miest*
- S *je množina statických miest*
- *prienik množiny dynamických miest* D *a množiny statických miest* S *je prázdny, t.j.* $D \cap S = \emptyset$
- $PN = (P = D \cup S, T, I, O)$ *je Petriho sieť.*

Definícia 17 (Označkovaná Petriho sieť so statickými miestami)
Označkovaná Petriho sieť so statickými miestami je šestica $MPNS = (D, S, T, I, O, m_0)$, *kde*

- $PNS = (D, S, T, I, O)$ *je Petriho sieť so statickými miestami a*
- $MPN = (P = D \cup S, T, I, O, m_0)$ *je označkovaná Petriho sieť.*

Hovoríme, že prechod je spustiteľný v označkovanej Petriho sieti so statickými miestami $MPNS$ *ak je spustiteľný v označkovanej Petriho sieti* MPN. *Grafom dosiahnuteľnosti označkovanej Petriho siete so statickými miestami* $MPNS$ *je graf dosiahnuteľnosti označkovanej Petriho siete* MPN. *Všetky pojmy definované pre označkovanú Petriho sieť* MPN *budeme analogicky používať pre označkovanú Petriho sieť so statickými miestami* $MPNS$.

Definícia 18 (Workflow sieť so statickými miestami)
Workflow sieť so statickými miestami je Petriho sieť so statickými miestami $W = (D, S, T, I, O)$, *kde* $PN = (P = D \cup S, T, I, O)$ *je workflow sieť, pričom vstupné miesto a výstupné miesto patria do množiny dynamických miest, t.j.* $in \in D$ *a* $out \in D$.

V označkovanej workflow sieti so statickými miestami je v počiatočnom značkovaní práve jedna značka vo vstupnom mieste a zároveň statické miesta obsahujú v počiatočnom značkovaní počet značiek reprezentujúci počet príslušných zdrojov.

Definícia 19 (Označkovaná workflow sieť so statickými miestami) *Označkovaná workflow sieť so statickými miestami je šestica* $MW = (D, S, T, I, O, m_0)$, *kde*

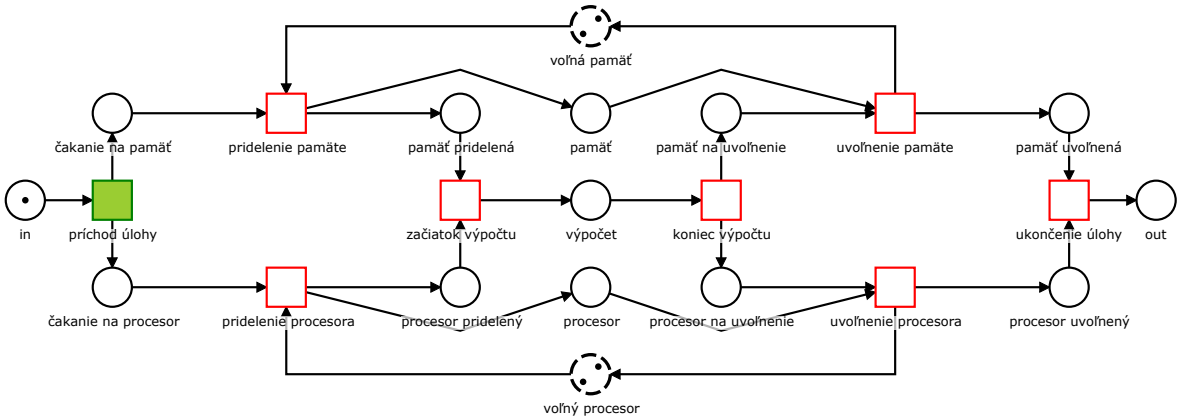
- $W = (D, S, T, I, O)$ *je workflow sieť so statickými miestami*
- m_0 *je počiatočné značkovanie také, že vstupné miesto obsahuje jednu značku, t.j.* $m(in) = 1$ *a zároveň* $m(d) = 0$ *pre každé dynamické miesto* $d \in D$, *ktoré je rôzne od in.*

V porovnaní s prácami [6, 32, 33] formalizujeme skutočnosť, že zdroje sú trvácne, pomocou komplementárnych miest [25] pre statické miesta. Tieto komplementárne miesta nazývame určujúce miesta statických miest. Pre určenú workflow sieť so statickými miestami budeme teda požadovať, aby pre každé miesto $s \in S$ existovalo miesto $d_s \in D$ určujúce počet používaných značiek zo statického miesta $s \in S$.

Definícia 20 (Určená workflow sieť so statickými miestami) *Nech* $W = (D, S, T, I, O)$ *je taká workflow sieť so statickými miestami, že pre každé miesto* $s \in S$ *existuje práve jedno miesto* $d_s \in D$ *rôzne od in a out, ktoré pre každé* $t \in T$ *spĺňa* $I(d_s, t) - O(d_s, t) = O(s, t) - I(s, t)$. *Potom sa takéto miesto* d_s *nazýva určujúce miesto statického miesta s. Workflow sieť W sa nazýva určená workflow sieť so statickými miestami. Množinu všetkých určujúcich miest určenej workflow siete označujeme* D_S.

Pripomeňme, že rovnosť $I(d_s, t) - O(d_s, t) = O(s, t) - I(s, t)$ v určenej workflow sieti so statickými miestami zabezpečuje, že v prípade, ak $O(s, t) - I(s, t) = 0$, platí taktiež $I(d_s, t) - O(d_s, t) = 0$. To znamená, že spustenie ľubovoľného prechodu môže vytvoriť značku v určujúcom mieste iba vtedy, ak zoberie značku z príslušného statického miesta. Keďže v počiatočnom značkovaní je určujúce miesto prázdne, pre označkované určené workflow siete teda vyplýva priamo z definície určujúcich miest, že súčet značiek v statickom mieste s a v jeho určujúcom mieste d_s zostáva v každom dosiahnuteľnom značkovaní rovný hodnote $m_0(s)$.

Dôsledok 2 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami. Nech* $s \in S$ *je ľubovoľné statické miesto a nech* $d_s \in D_S$ *je jeho*



Obr. 1.11: Označkovaná určená workflow sieť so statickými miestami, ktorých označkovanie modeluje počet pamäťových jednotiek a procesorov k dispozícii

určujúce miesto. Potom pre každé značkovanie m *dosiahnuteľné z* m_0 *platí, že* $m(s) + m(d_s) = m_0(s)$ *a teda* $m(s) \leq m_0(s)$.

Statické miesta budú graficky znázornené kruhmi vykreslenými prerušovanou čiarou. V ilustratívnom príklade môžeme statickými miestami *voľná pamäť* a *voľný procesor* a ich počiatočným značkovaním modelovať voľné jednotky pamäte a voľné procesory, ako je to to znázornené na Obr. 1.11. Miesto *pamäť* je určujúcim miestom pre statické miesto *voľná pamäť*, podobne miesto *procesor* je určujúce miesto pre statické miesto *voľný procesor*.

Korektnosť správania označkovanej workflow siete so statickými miestami definujeme analogicky ku korektnosti správania označkovanej workflow siete [4] ako schopnosť ukončiť korektne jednu inštanciu.

Definícia 21 (Finálne značkovania označkovanej workflow siete so statickými miestami siete)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami. Značkovanie* m_f *siete* MW, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0, *nazývame finálne značkovanie, ak platí* $m_f(out) = 1$, $m_f(d) = 0$ *pre každé* $d \in D$, *ktoré je rôzne od out.*

Definícia 22 (Korektné správanie označkovanej workflow siete so statickými miestami)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami*

a nech $out \in P$ označuje výstupné miesto. Označkovaná workflow sieť so statickými miestami MW má korektné správanie práve vtedy, keď pre každé značkovanie m dosiahnuteľné z m_0 platí:

- *existuje finálne značkovanie m_f siete MW, ktoré je dosiahnuteľné zo značkovania m,*
- *ak $m(out) \geq 1$ potom m je finálne značkovanie siete MW.*

Existencia určujúcich miest v určenej workflow sieti navyše zabezpečí, že nové zdieľané zdroje sa nevytvárajú ani nespotrebujú, teda po ukončení inštancie sa počet zdrojov rovná počtu zdrojov danému značkovaním statických miest v počiatočnom značkovaní. Takéto systémy sa často vyskytujú v praxi, v ilustračnom príklade táto vlastnosť znamená, že počas spusteného procesu sa počet procesorov a pamäťových jednotiek nemení. To zároveň znamená, že finálne značkovanie označkovanej určenej workflow siete so statickými miestami a korektným správaním je jednoznačne definované.

Dôsledok 3 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami s korektným správaním a nech m_f je finálne značkovanie MW. Potom $m_f(s) = m_0(s)$ pre každé $s \in S$.*

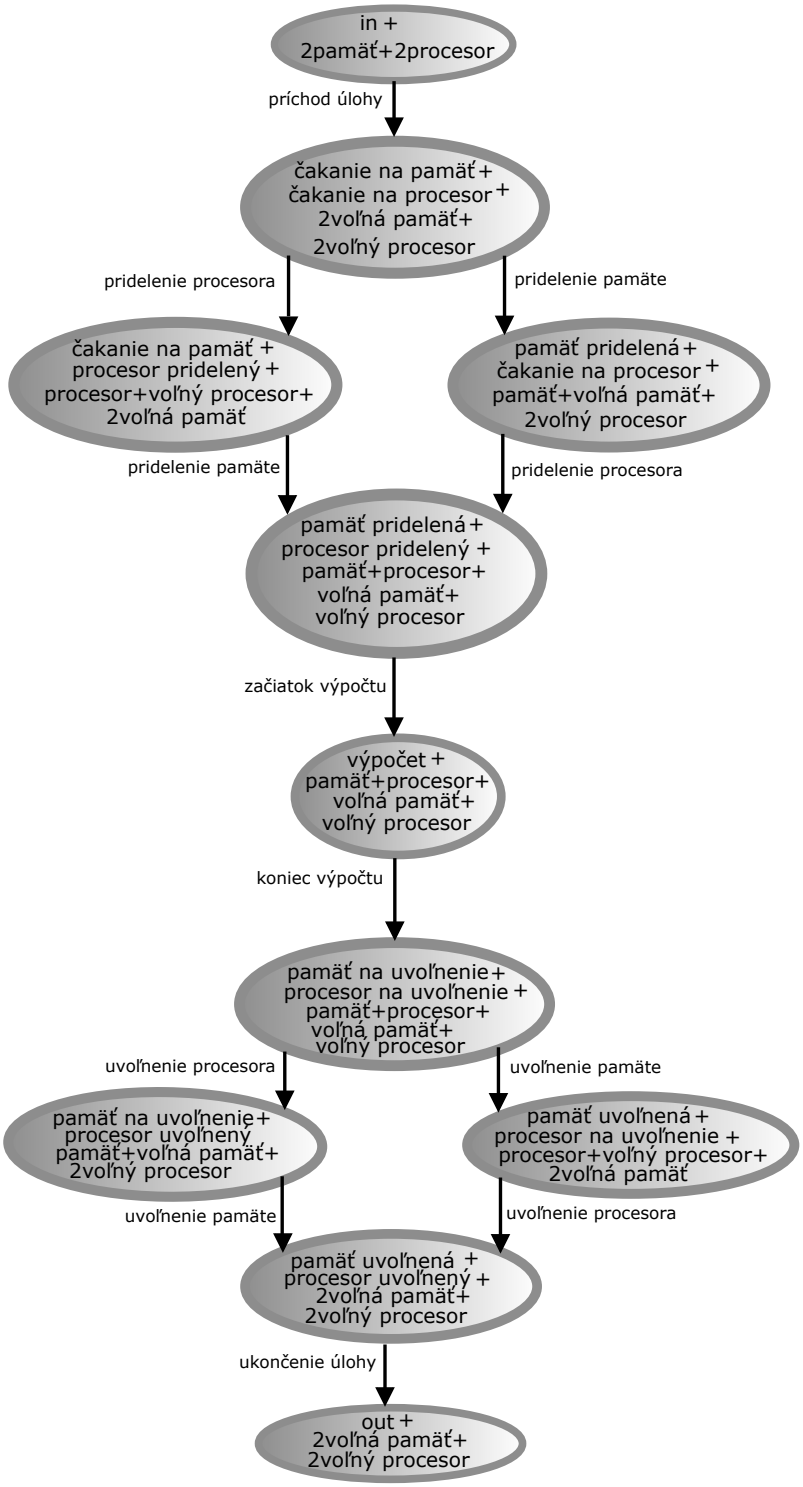
Na Obr. 1.12 je znázornený graf dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami z Obr. 1.11.

Porovnaním s grafom dosiahnuteľnosti označkovanej workflow siete z Obr. 1.1, ktorý je znázornený Na Obr. 1.10 je zrejmé, že statické miesta neovplyvnili spustiteľnosť jednotlivých prechodov. Inými slovami, grafy dosiahnuteľnosti oboch sietí sú izomorfné.

Pre označkované určené workflow siete so statickými miestami a korektným správaním platí zároveň obdobný výsledok ako lemma 2.

Lema 3 *Ak označkovaná určená workflow sieť so statickými miestami má korektné správanie, potom počet stavov dosiahnuteľných z počiatočného značkovania je konečný.*

Dôkaz. Nech počet značkovaní dosiahnuteľný z počiatočného značkovania m_0 nie je konečný, a teda podľa Dicksonovej lemy [29], nech m a m' sú značkovania dosiahnuteľné z m_0 také, že m je menšie ako m'. Z definície korektného správania vyplýva, že $m'(out) = 1$ alebo $m'(out) = 0$ a že $m(out) = 1$ alebo $m(out) = 0$. Keďže $m < m'$, platí zároveň $m(s) \leq m'(s)$ pre každé $s \in S$.



Obr. 1.12: Graf dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami modelujúcej spracovanie výpočtovej úlohy

- Kombinácia $m'(out) = 1$ a $m(out) = 1$ znamená, že $m' = m_f = m$, čo je v rozpore s predpokladom, že m' je väčšie ako m.

- Kombinácia $m'(out) = 0$ a $m(out) = 1$ je taktiež v rozpore s predpokladom, že m' je väčšie ako m.

- Kombinácia $m'(out) = 1$ a $m(out) = 0$ znamená, že $m' = m_f$. Pri predpoklade, že m_f je väčšie ako m, potom $m(d) = 0$ pre každé $d \in D$). Zároveň z prvej odrážky definície korektného správania dostávame, že značkovanie m_f je dosiahnuteľné zo značkovania m spustením nejakej postupnosti prechodov. Z definície spustiteľnosti prechodov je zrejmé, že ak je postupnosť prechodov spustiteľná z nejakého značkovania, potom je spustiteľná z každého väčšieho značkovania, čiže aj zo značkovania m_f. Ak spustenie tejto postupnosti vedie zo značkovania m do značkovania m_f, potom spustenie tej istej postupnosti prechodov zo značkovania m_f vedie do značkovania $m''(out) = m_f(out) + (m_f(out) - m(out))$, a teda $m''(out) = 1 + (1 - 0)) = 2$, čo je v rozpore s druhou odrážkou definície korektného správania, z ktorej vyplýva, že každé dosiahnuteľné značkovanie m'' platí, že počet značiek v mieste *out* je maximálne 1.

- Kombinácia $m'(out) = 0$ a $m(out) = 0$ znamená pri predpoklade m' je väčšie ako m, že existuje $p \in P$ rôzne od *out* také, že $m'(p) > m(p)$. Zároveň z prvej odrážky definície korektného správania dostávame, že značkovanie m_f je dosiahnuteľné zo značkovania m spustením nejakej postupnosti prechodov. Z definície spustiteľnosti prechodov je zrejmé, že ak je postupnosť prechodov spustiteľná z nejakého značkovania, potom je spustiteľná z každého väčšieho značkovania, čiže aj zo značkovania m'. Ak spustenie tejto postupnosti vedie zo značkovania m do značkovania m_f, potom spustenie tej istej postupnosti prechodov zo značkovania m' vedie do značkovania $m'' = m' + (m_f - m)$. Keďže $m'(out) = 0$ a $m(out) = 0$, dostávame $m''(out) = 1$. Zároveň $m'(p) > m(p)$. Ak $p \in D$, dostávame $m''(p) > 0$, čo je v rozpore s druhou odrážkou definície korektného správania. Ak $p \in S$, dostávame $m''(p) > m_f(p)$, čo je v rozpore s tvrdením Dôsledkov 2 a 3.

To znamená, že ak označkovaná určená workflow sieť so statickými miestami má korektné správanie, potom počet stavov dosiahnuteľných z počiatočného značkovania je konečný, teda sieť je ohraničená. □

Nutnou podmienkou korektného správania určenej siete je teda konečný počet stavov dosiahnuteľných z počiatočného značkovania siete. Overiť, či označkovaná určená workflow sieť má korektné správanie je možné jednoduchou modifikáciou Algoritmu 2.

Algoritmus 3 (Overenie korektného správania určenej workflow siete so statickými miestami)

Pre označkovanú určenú workflow sieť so statickými miestami $MW = (D, S, T, I, O, m_0)$

1. vytvor prázdny zoznam M nájdených značkovaní
2. vytvor premennú m_f *je dosiahnuteľné* a vlož do nej hodnotu *nie*
3. vytvor prázdny zoznam H označených hrán
4. vlož do zoznamu M počiatočné značkovanie m_0 a nastav množinu $Pre(m_0)$ jeho predchodcov prázdnu
5. pokiaľ je v zozname M značkovanie m, ktoré nie je označené ako preskúmané, vezmi ho a rob nasledovné:
   (a) ak $m(out) \geq 1$ a zároveň $m \neq m_f$ potom algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"
   (b) ak $m = m_f$ vlož do m_f *je dosiahnuteľné* hodnotu *áno*
   (c) pre každý prechod t spustiteľný z m
      i. počítaj značkovanie m' dosiahnuté spustením prechodu t zo značkovania m
      ii. ak m' ešte nie je v zozname M, vlož ho do zoznamu M a nastav množinu $Pre(m')$ jeho predchodcov prázdnu
      iii. nastav množinu $Pre(m')$ predchodcov značkovania m' rovnú zjednoteniu $Pre(m') \cup Pre(m) \cup \{m\}$
      iv. ak existuje predchodca m'' z $Pre(m')$ taký, že pre všetky miesta $p \in P$ platí $m''(p) \leq m'(p)$ a zároveň existuje miesto $p \in P$ také, že platí $m''(p) < m'(p)$, potom algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"
      v. vlož do zoznamu H hranu z m do m' označenú prechodom t, teda trojicu (m, t, m')

(d) označ m ako preskúmané

6. ak m_f *je dosiahnuteľné* = *áno* a $Pre(m_f) \cup \{m_f\} = M$, vráť informáciu "Sieť má korektné správanie", inak vráť "Sieť nemá korektné správanie"

Z grafu dosiahnuteľnosti je zrejmé, že označkovaná určená workflow sieť so statickými miestami z Obr. 1.11 má korektné správanie.

1.5 Vykonávané siete so statickými miestami a inštanciami

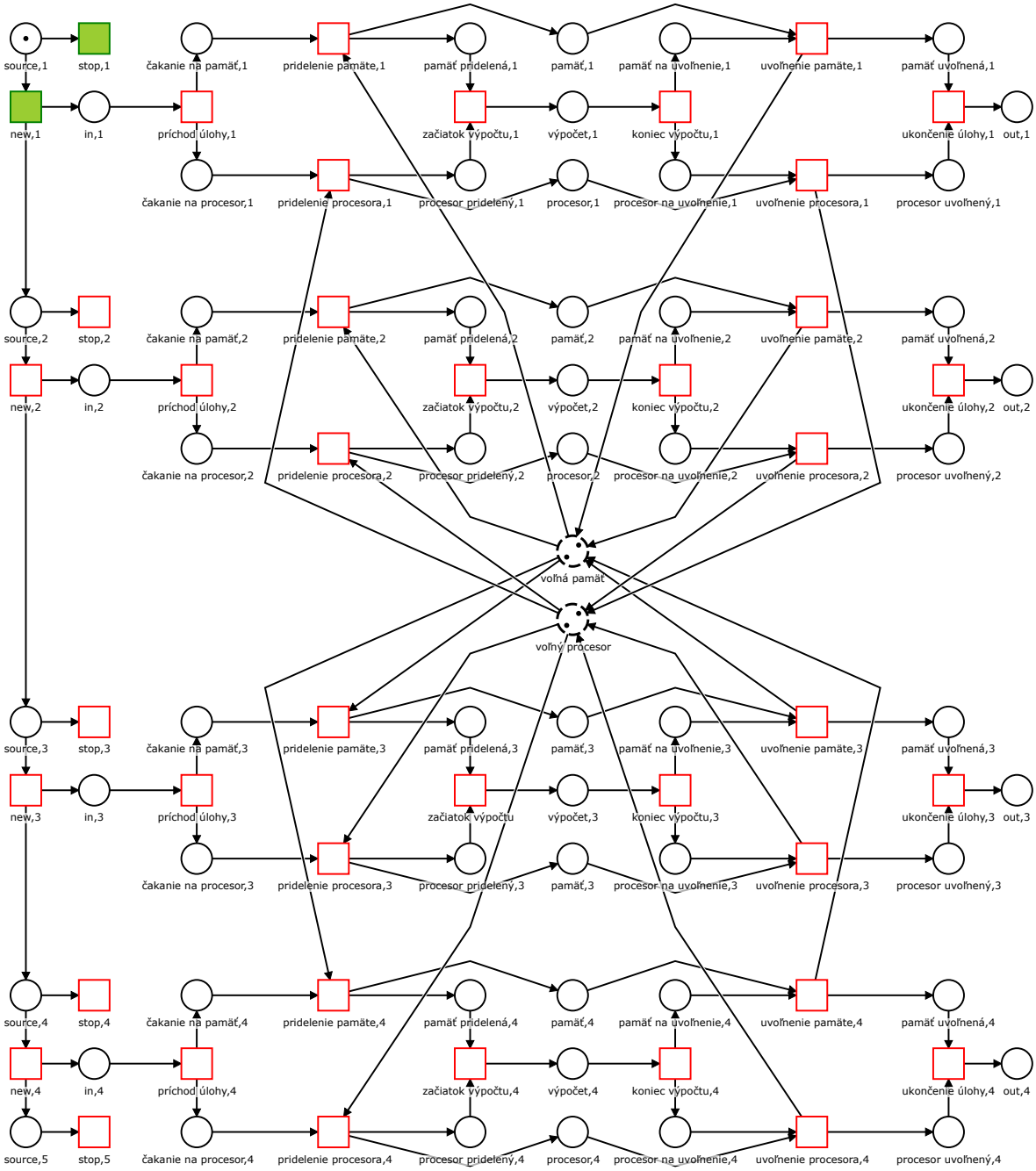
Vykonávaná workflow sieť so statickými miestami bude reprezentovať bežiaci workflow proces s inštanciami. Inštancie budú reprezentované kópiami pôvodnej workflow siete, ktoré budú indexované kladnými celými číslami. Pre každú inštanciu navyše pridáme špeciálnu konštrukciu pozostávajúcu z miesta $source, i$, prechodu $stop, i$ a prechodu new, i s príslušným indexom i, hrany zo $source, i$ do $stop, i$, hrany zo $source, i$ do new, i a hrany z new, i do $source, i + 1$. Táto konštrukcia reprezentuje konštruktor inštancie s indexom i. V počiatočnom značkovaní je označené iba miesto $source, 1$. Spustenie prechodu new, i vytvorí značku vo vstupnom mieste inštancie workflow siete s poradovým číslom i. Vytvorenie značky vo vstupnom mieste indexovanom kladným celým číslom $i \in Z$ reprezentuje teda vytvorenie inštancie s poradovým číslom i. Zároveň spustenie prechodu new, i vytvorí značku v mieste $source, i + 1$, čím sa stane spustiteľný prechod $new, i + 1$, teda konštruktor pre ďalšiu inštanciu. Prechod $stop, i$ umožňuje pri označkovanom mieste $source, i$ zastaviť vytváranie ďalších inštancií.

Definícia 23 (Vykonávaná workflow sieť so statickými miestami)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech new, stop a source označujú prvky, ktoré nepatria do množín* D, S *a* T, *teda* $\{new, stop, source\} \cap (D \cup S \cup T) = \emptyset$. *Potom vykonávaná workflow sieť so statickými miestami siete* MW *je označkovaná Petriho sieť* $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$, *kde*

- $P^\infty = S \cup (D \times \mathbb{Z}) \cup (\{source\} \times \mathbb{Z})$, *teda miesta siete* MPN *tvoria statické miesta siete* MW, *kópie dynamických miest označené kladnými celými číslami a kópie source označené kladnými celými číslami*

- $T^\infty = (T \times \mathbb{Z}) \cup (\{new, stop\} \times \mathbb{Z})$, *teda prechody siete* MPN *tvoria kópie prechodov siete* MW *označené kladnými celými číslami a kópie new a stop označené kladnými celými číslami*

- $I^\infty(s, (t, i)) = I(s, t)$ *pre všetky* $s \in S$, $t \in T$ *a* $i \in \mathbb{Z}$, *teda hrany zo statických miest do kópií prechodov sa kopírujú*



Obr. 1.13: Časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie

- $I^\infty((d,i),(t,i)) = I(d,t)$ *pre všetky* $d \in D$, $t \in T$ *a* $i \in \mathbb{Z}$, *teda hrany z kópií dynamických miest do kópií prechodov sa kopírujú*

- $I^\infty((source,i),(new,i)) = 1$ *a* $I^\infty((source,i),(stop,i)) = 1$ *pre všetky* $i \in \mathbb{Z}$, *teda spustenie konštruktora* (new,i) *pre inštanciu s indexom* i *je možné iba v značkovaní s označeným miestom* $(source,i)$, *podobne zastavenie* $(stop,i)$

- $O^\infty(s,(t,i)) = O(s,t)$ *pre všetky* $s \in S$, $t \in T$ *a* $i \in \mathbb{Z}$, *teda hrany z kópií prechodov do statických miest sa kopírujú*

- $O^\infty((d,i),(t,i)) = O(d,t)$ *pre všetky* $d \in D$, $t \in T$ *a* $i \in \mathbb{Z}$, *teda hrany z kópií prechodov do kópií dynamických miest sa kopírujú*

- $O^\infty((in,i),(new,i)) = 1$ *a* $O^\infty((source,i+1),(new,i)) = 1$ *pre všetky* $i \in \mathbb{Z}$, *teda spustenie konštruktora* (new,i) *pre inštanciu s indexom* i *vytvorí značku v mieste* (in,i), *čím vytvorí inštanciu s indexom* i, *a zároveň vytvorí značku v mieste* $(source,i+1)$, *čím sa stane spustiteľným konštruktor* $(new,i+1)$

- $I^\infty(p,t) = 0$ *a* $O^\infty(p,t) = 0$ *pre všetky ostatné dvojice* $(p,t) \in (P^\infty \times T^\infty)$

- $m_0^\infty(source,1) = 1$, $m_0^\infty(s) = m_0(s)$ *pre každé* $s \in S$, $m_0^\infty(d,i) = 0$ *pre každé* $(d,i) \in (D \times \mathbb{Z})$ *a* $m_0^\infty(source,i) = 0$ *pre každé celé číslo* i *väčšie ako* 1, *teda v počiatočnom značkovaní je jedna značka v mieste* $(source,1)$, *značky v statických miestach sú skopírované a ostatné miesta sú prázdne.*

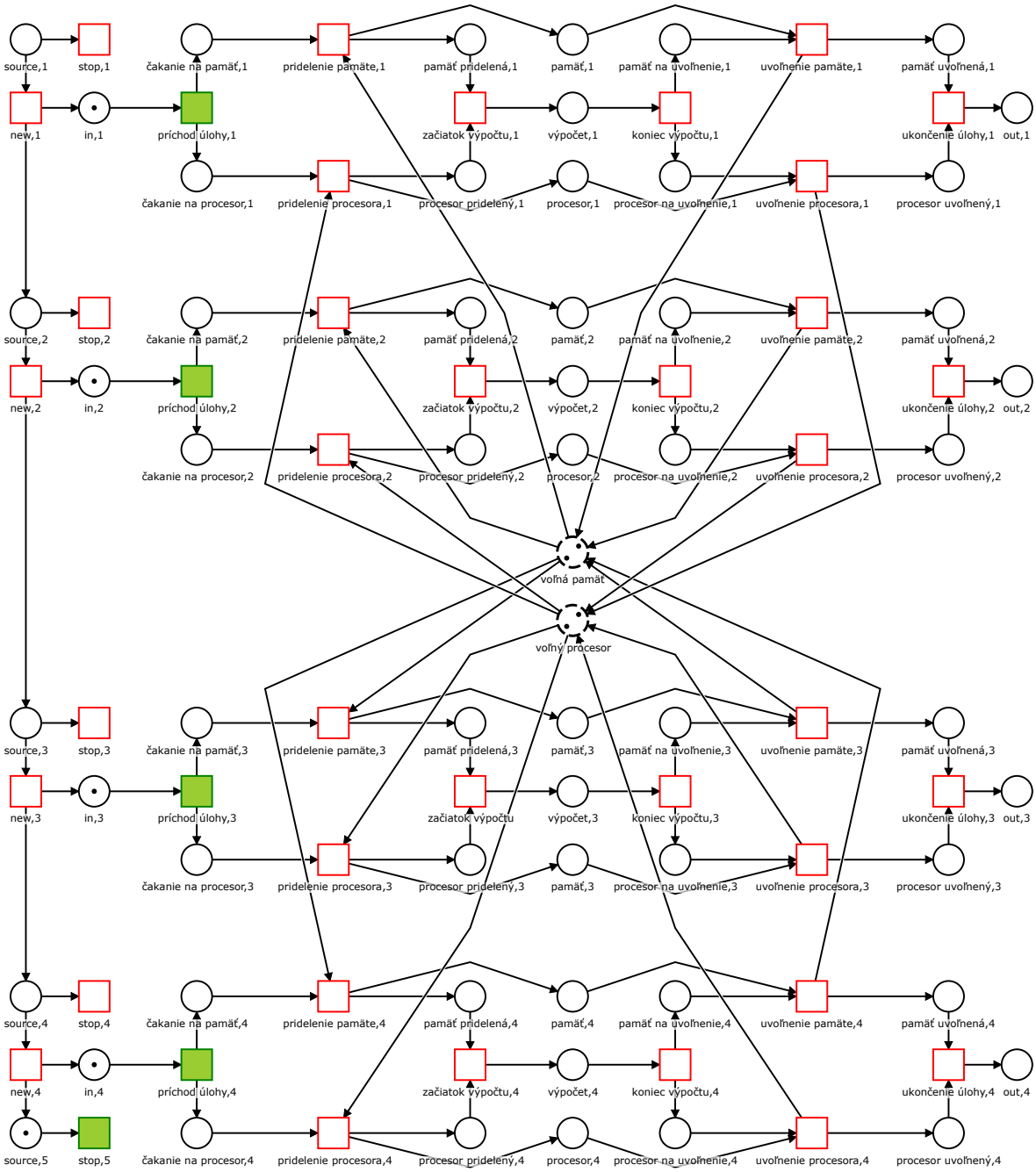
Miesta i-tej inštancie budeme označovať P_i^∞, *teda* $P_i^\infty = (D \cup \{source\}) \times \{i\}$ *pre kladné celé číslo* i.
Prechody i-tej inštancie budeme označovať T_i^∞, *teda* $T_i^\infty = (T \cup \{new, stop\}) \times \{i\}$ *pre kladné celé číslo* i.

Vykonávaná workflow sieť je teda Petriho sieť, ktorá má formálne nekonečne veľa prechodov a nekonečne veľa miest, keďže modeluje systémy bez ohraničeného počtu inštancií.

Pre vykonávané workflow siete určených workflow sietí s korektným správaním priamo z definície vyplýva, že súčet značiek v statickom mieste s a v kópiách jeho určujúcich miest (d_s,i) zostáva v každom dosiahnuteľnom značkovaní rovný hodnote $m_0(s)$.

Dôsledok 4 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW. *Nech* $s \in S$ *je ľubovoľné statické miesto a nech* $d_s \in D_S$ *je jeho určujúce miesto. Potom pre každé značkovanie* m^∞ *dosiahnuteľné z* m_0^∞ *platí, že* $m^\infty(s) + \sum_{i \in \mathbb{Z}} m^\infty(d_s, i) = m_0^\infty(s) = m_0(s)$, *a teda* $m^\infty(s) \leq m_0^\infty(s) = m_0(s)$.

Na Obr. 1.13 je znázornená časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie workflow procesu z Obr. 1.11, konkrétne sú znázornené iba statické miesta, miesta s indexom 1 až 4 a prechody s indexom 1 až 4, miesto *source,5*, prechod *stop,5* a hrany medzi príslušnými elementami.



Obr. 1.14: Časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie po vytvorení 4 inštancií konštruktorom

1.6 Uviaznutia vo vykonávaných sieťach so statickými miestami a inštanciami

1.6.1 Zamrznutie

Počiatočné značkovanie vykonávanej siete znázornené na Obr. 1.13 vyjadruje stav systému s dvoma jednotkami pamäte a dvoma procesormi. Spustiteľné sú prechody *new,1* a *stop,1*.

Spustenie *stop,1* reprezentuje ukončenie programu pred spracovaním prvej výpočtovej úlohy.

Spustenie prechodu *new,1* reprezentuje volanie konštruktora, ktorý vytvorí prvú inštanciu výpočtovej úlohy. Jeho spustenie spotrebuje značku v mieste *source,1* a vytvorí značku v miestach *in,1* a *source,2*, čím sa stanú spustiteľnými prechody *príchod úlohy,1*, *new,2* a *stop,2*.

Postupným spúšťaním prechodov *new,2*, *new,3* a *new,4* dosiahneme značkovanie *in,1 + in,2 + in,3 + in,4 + source,5 + 2voľná pamäť + 2voľný procesor*, ktoré je znázornené na obrázku Obr. 1.14.

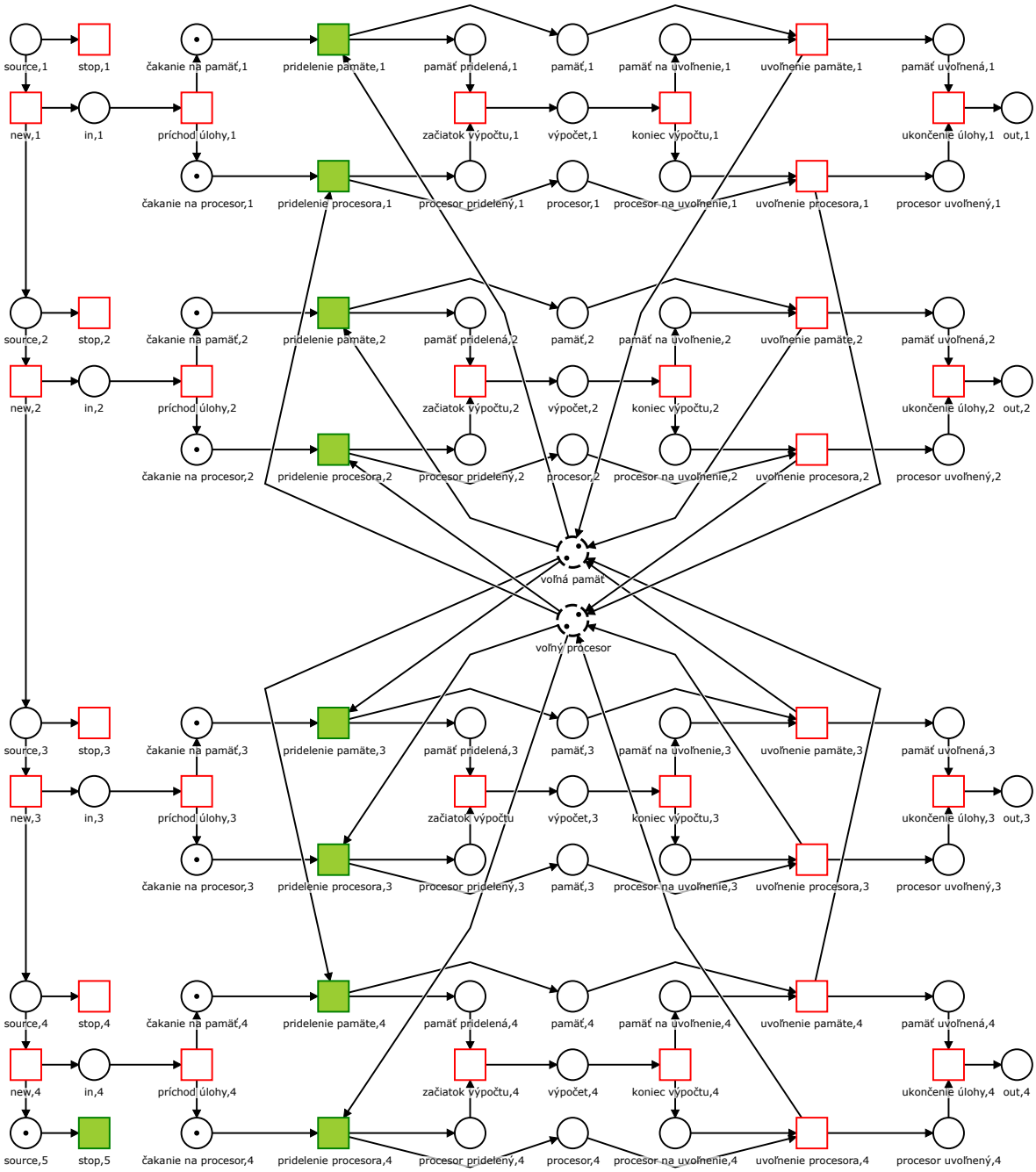
V tomto značkovaní sú spustiteľné prechody *príchod úlohy,1*, *príchod úlohy,2*, *príchod úlohy,3*, *príchod úlohy,4* a *stop,5*, ktoré sú zvýraznené zelenou farbou na Obr. 1.14 a zároveň prechod *new,5*, ktorý na Obr. 1.14 nie je znázornený.

Keďže prechody *príchod úlohy,1 príchod úlohy,2*, *príchod úlohy,3* a *príchod úlohy,4* sú spustiteľné nezávisle v značkovaní *in,1 + in,2 + in,3 + in,4 + source,5 + 2voľná pamäť + 2voľný procesor*, môžeme ich spustiť v ľubovoľnom poradí.

Spustenie prechodu *príchod úlohy,1* reprezentuje príchod prvej konkrétnej výpočtovej úlohy s konkrétnymi parametrami. Spustenie tohto prechodu spotrebuje značku z miesta *in,1* a vytvorí značku v mieste *čakanie na pamäť,1* a značku v mieste *čakanie na procesor,1*.

Podobne spustenie prechodov *príchod úlohy,2*, *príchod úlohy,3* a *príchod úlohy,4* spotrebuje značky z miest *in,2*, *in,3* a *in,4* a vytvorí značky v miestach *čakanie na pamäť,2*, *čakanie na pamäť,3* a *čakanie na pamäť,4*, respektíve v miestach *čakanie na procesor,2*, *čakanie na procesor,3* a *čakanie na procesor,4*.

To znamená, že po spustení prechodov *príchod úlohy,1 príchod úlohy,2*, *príchod úlohy,3* a *príchod úlohy,4* v značkovaní *in,1 + in,2 + in,3 + in,4 + source,5* na Obr. 1.14



Obr. 1.15: Časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie po načítaní parametrov úloh

sa sieť dostane do značkovania *čakanie na pamäť,1 + čakanie na procesor,1 + čakanie na pamäť,2 + čakanie na procesor,2 + čakanie na pamäť,3 + čakanie na procesor,3 + čakanie na pamäť,4 + čakanie na procesor,4 + source,5 + 2voľná pamäť + 2voľný procesor*, ktoré je znázornené na Obr. 1.15.

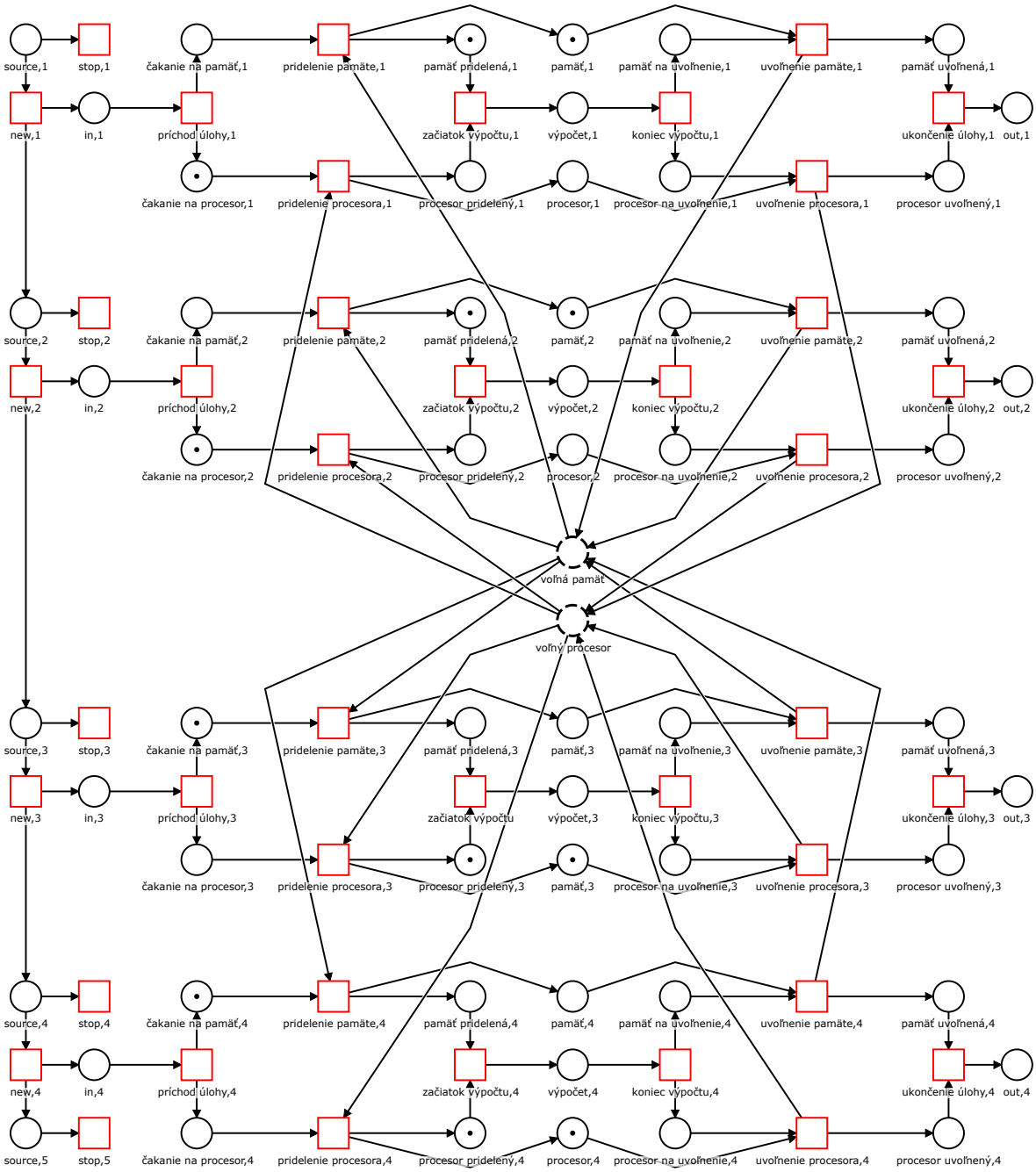
Toto značkovanie zodpovedá stavu, v ktorom 4 výpočtové úlohy čakajú na pridelenie pamäťovej jednotky a pridelenie procesora. Spustiteľné sú dvojica prechodov *pridelenie pamäte,1*, *pridelenie procesora,1*, dvojica prechodov *pridelenie pamäte,2*, *pridelenie procesora,2*, dvojica prechodov *pridelenie pamäte,3*, *pridelenie procesora,3* a dvojica prechodov *pridelenie pamäte,4*, *pridelenie procesora,4*. Okrem týchto prechodov je spustiteľný aj prechod *stop,5* a prechod *new,5*, ktorý nie je znázornený na obrázkoch.

Predstavme si situáciu, že v prvých dvoch inštanciách sa spustia prechody *pridelenie pamäte,1* a *pridelenie pamäte,2*, zatiaľ čo v ďalších dvoch inštanciách sa spustia prechody *pridelenie procesora,3* a *pridelenie procesora,4*. Pre zjednodušenie si predstavme, že sa spustí aj prechod *stop,5*.

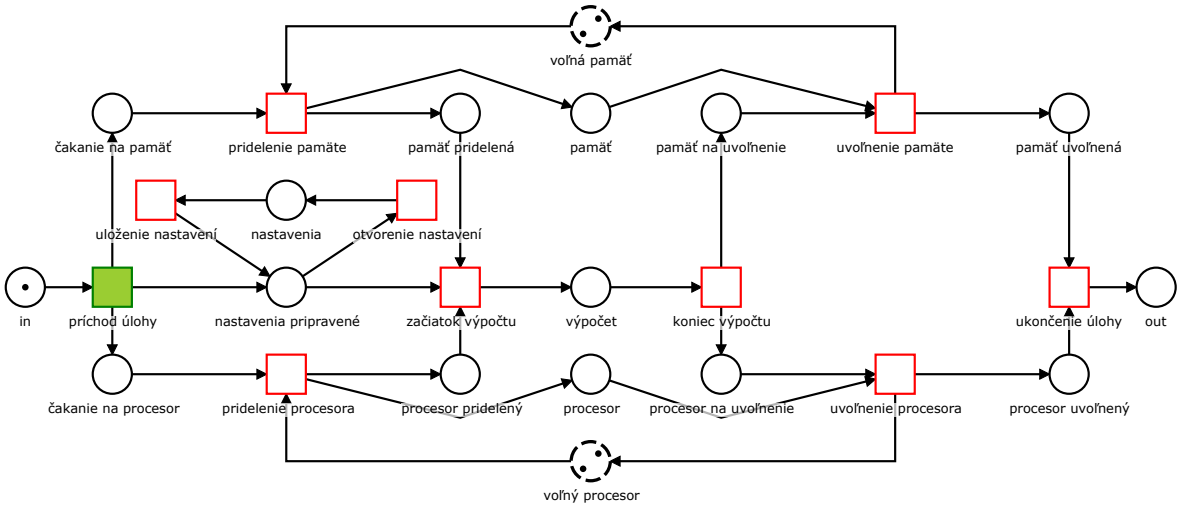
Systém sa dostane do značkovania *pamäť pridelená,1 + čakanie na procesor,1 + pamäť,1 + pamäť pridelená,2 + čakanie na procesor,2 +pamäť,2 + čakanie na pamäť,3 + procesor pridelený,3 + procesor,3 + čakanie na pamäť,4 + procesor pridelený,4 + procesor,4*, znázorneného na Obr. 1.16.

V tomto značkovaní nie sú spustiteľné žiadne prechody a systém zamrzne. Nie je možné dokončiť žiadnu zo 4 výpočtových úloh. Ak by sme nespustili prechod *stop,5*, systém síce nezamrzne, bude prijímať ďalšie úlohy, avšak v žiadnej z inštancií nebude možné prideliť ani pamäť ani procesor. Žiadnu inštanciu nebude možné korektne ukončiť. Značkovanie vykonávanej siete, z ktorého jednotlivé inštancie workflow siete so statickými miestami nie je možné korektne ukončiť, nazývame uviaznutie. Uviaznutie je spôsobené tým, že jednotlivé inštancie súťažia o zdieľané zdroje, v tomto prípade pamäť a procesory.

Ak je teda vo vykonávanej sieti pred dokončením bežiacich inštancií dosiahnuteľné značkovanie, v ktorom nie je možné spustiť žiadny prechod, tak ako je to na Obr. 1.16, nazývame takéto uviaznutie zamrznutím (v anglickom jazyku deadlock). Ako však ilustruje nasledovný príklad, možu existovať pred dokončením situácie, pri ktorých sa systém dostane zo množiny značkovaní, v ktorých je vždy nejaký prechod spustiteľný, ale napriek tomu nie je možné bežiace inštancie dokončiť. Takéto uviaznutie nazývame



Obr. 1.16: Časť uviaznutej vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie v stave zamrznutia



Obr. 1.17: Výpočtová úloha s možnosťou zmeny nastavenia parametrov výpočtu

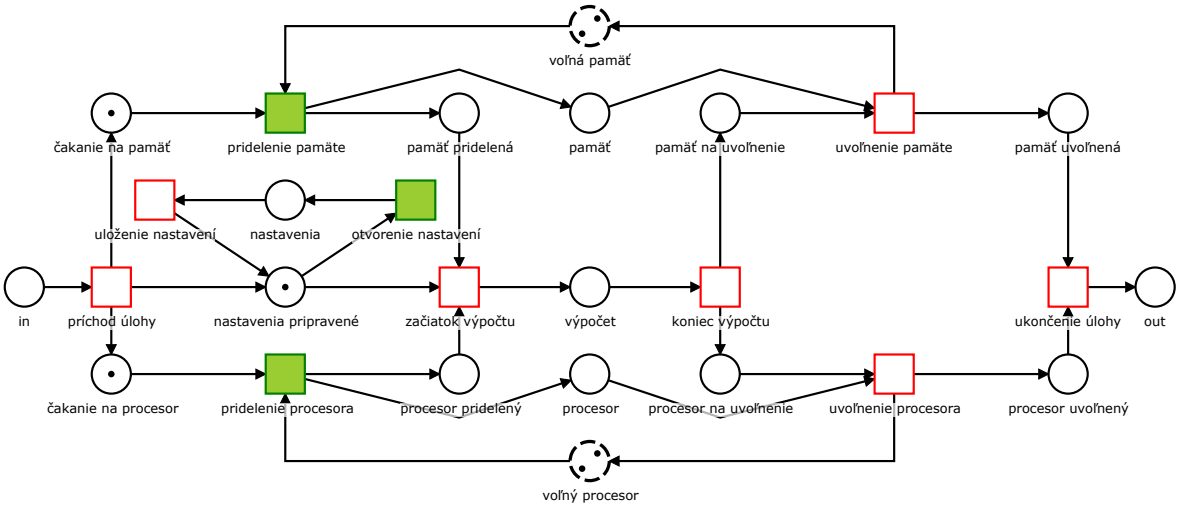
zacyklením (anglicky livelock).

1.6.2 Zacyklenie

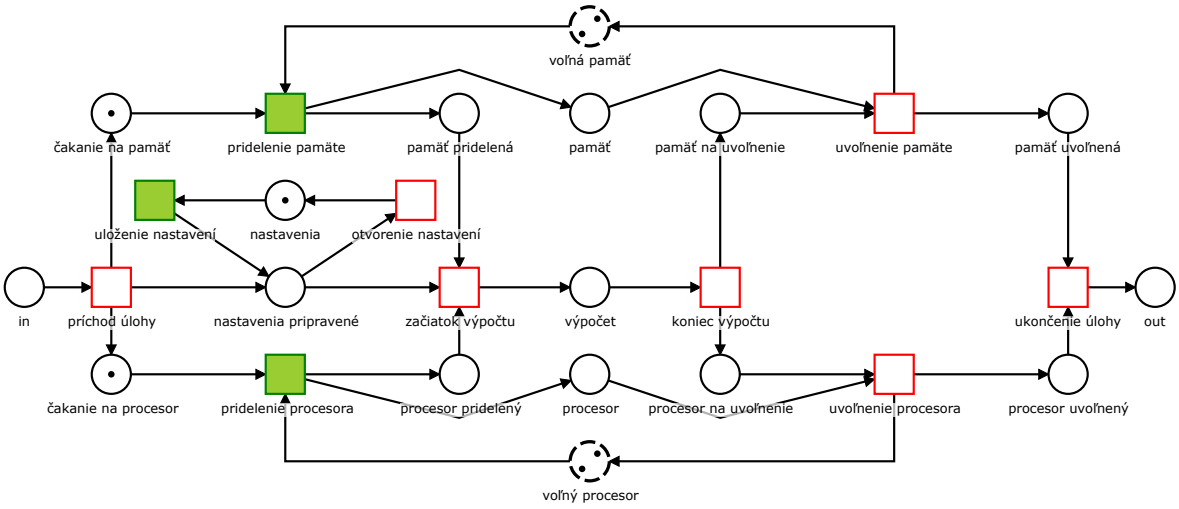
Predstavme si situáciu, modelovanú určenou workflow sieťou so statickými miestami a s korektným správaním na Obr. 1.17: po príchode výpočtovej úlohy sa načítajú prednastavené parametre výpočtu, čo je vyjadrené značkou v mieste *nastavenia pripravené* (Obr. 1.18). V tomto stave je možné spustiť prechod *otvorenie nastavení*, pričom sa otvorí dialógové okno s nastaveniami, teda vytvorí sa značka v mieste *nastavenia* (Obr. 1.19). V tomto stave je možné vykonať zmenu nastavenia prednastavených parametrov výpočtu, napr. zmenu maximálneho času výpočtu. Následne je možné spustením prechodu *uloženie nastavení* nastavenia uložiť a vytvoriť značku v mieste *nastavenia pripravené*. Nastaviť parametre výpočtu je možné teda opakovane pred spustením samotného výpočtu. Ak má úloha pridelenú pamäť, procesor a nastavenia sú pripravené (Obr. 1.20), môže sa spustiť samotný výpočet, čo je modelované prechodom *začiatok výpočtu*.

Opätovne, označkovaná určená workflow sieť so statickými miestami na Obr. 1.17 má korektné správanie. Graf dosiahnuteľnosti tejto siete, znázornený na Obr. 1.21, obsahuje cykly.

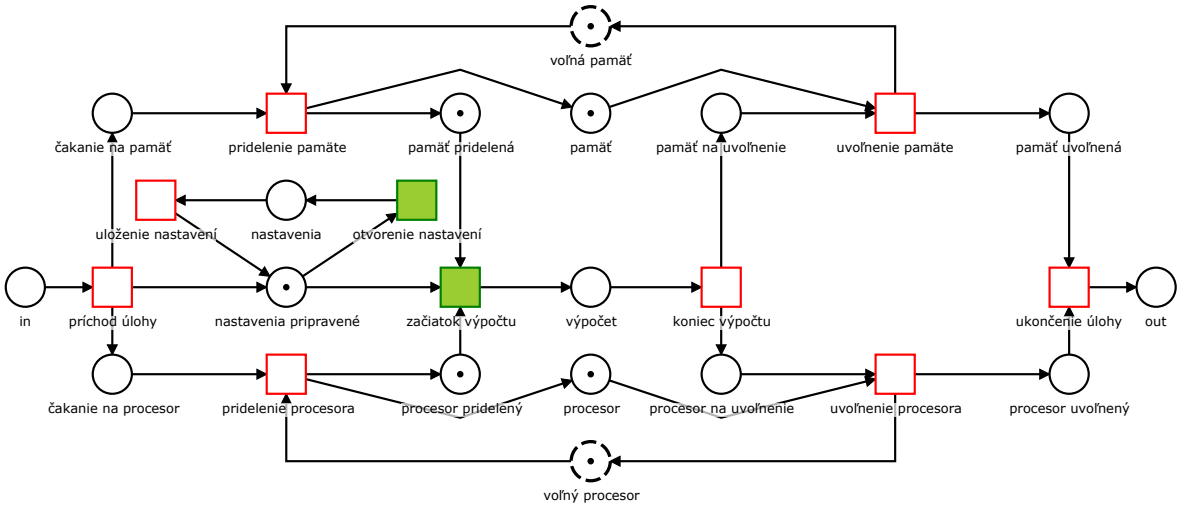
Na Obr. 1.22 je znázornená časť vykonávanej workflow siete so statickými miestami pre proces na Obr. 1.17 pre prvé 4 inštancie, konkrétne sú znázornené iba statické miesta, miesta s indexom 1 až 4 a prechody s indexom 1 až 4, miesto *source,5*, prechod



Obr. 1.18: Výpočtová úloha s možnosťou zmeny nastavenia parametrov výpočtu po príchode výpočtovej úlohy



Obr. 1.19: Výpočtová úloha s možnosťou zmeny nastavenia parametrov výpočtu po otvorení dialógového okna s nastaveniami



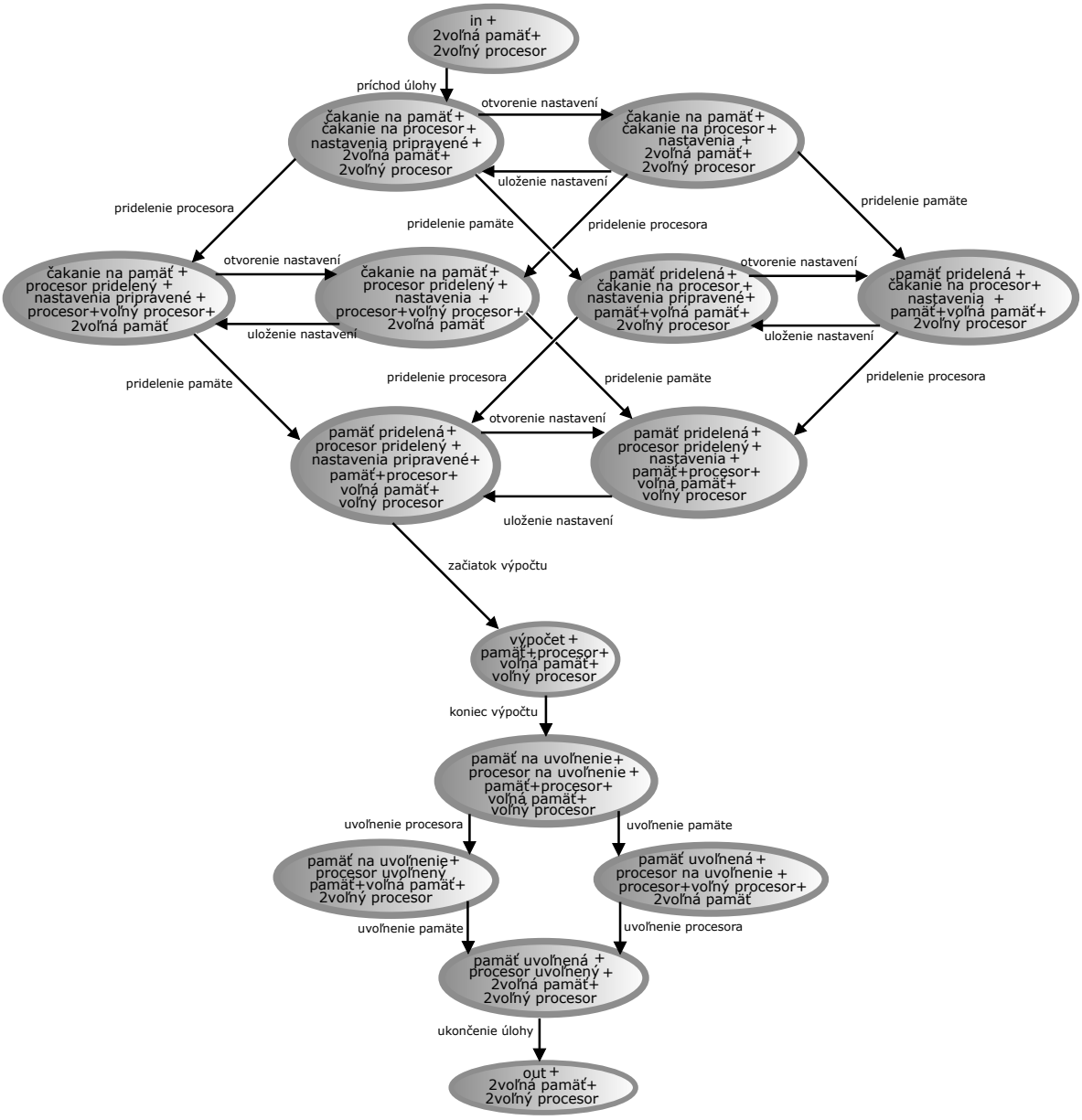
Obr. 1.20: Výpočtová úloha po pridelení pamäti a procesora s pripravenými nastaveniami a pustiteľným začiatkom výpočtu

stop,5 a hrany medzi príslušnými elementami. Počiatočné značkovanie znázorné na Obr. 1.22 vyjadruje stav systému s dvoma jednotkami pamäte a dvoma procesormi.

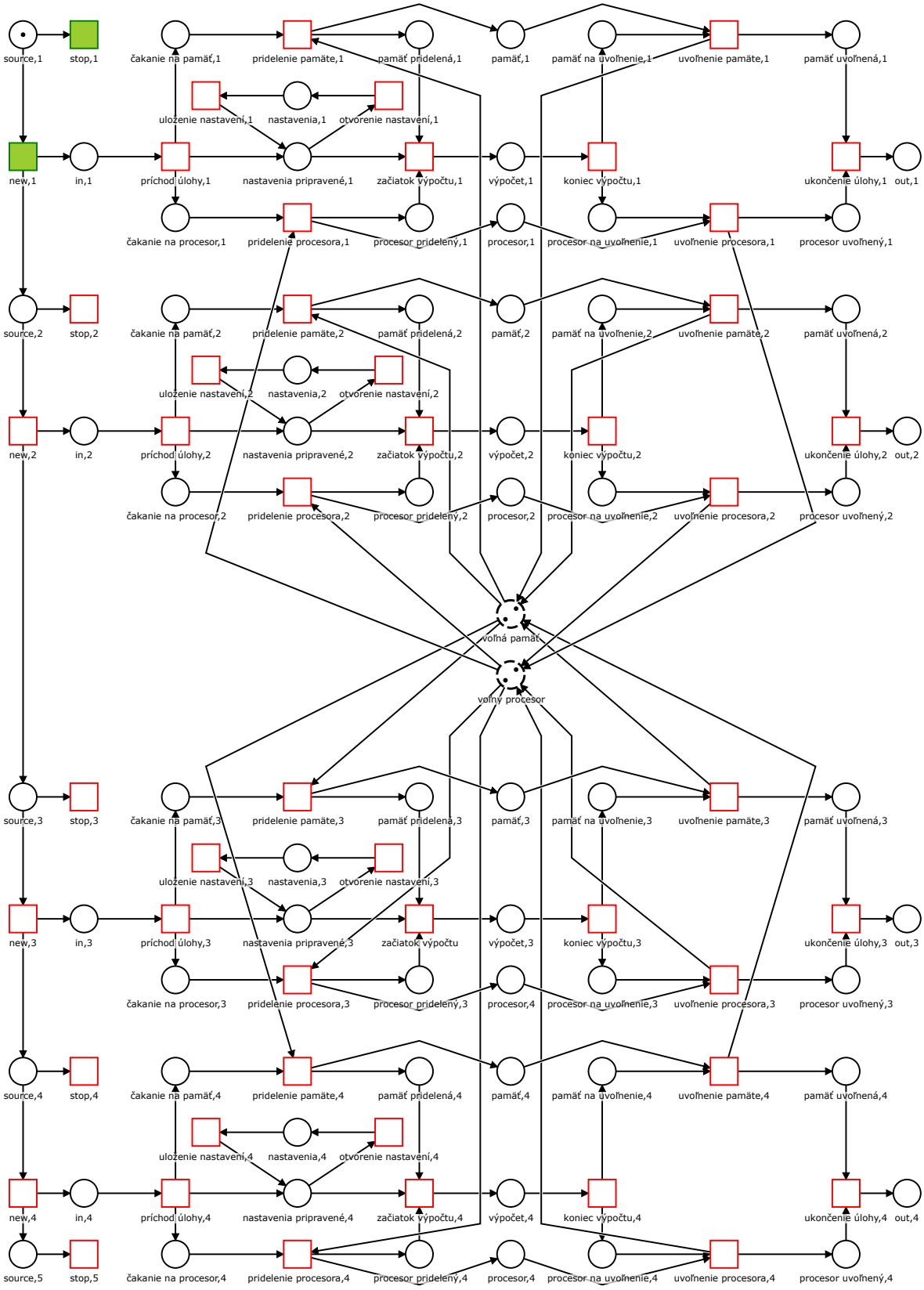
Postupným spúšťaním prechodov *new,1*, *new,2*, *new,3* a *new,4* a spustením prechodov *príchod úlohy,1*, *príchod úlohy,2*, *príchod úlohy,3* a *príchod úlohy,4* sa vytvoria značky v miestach *čakanie na pamäť,1*, *čakanie na pamäť,2*, *čakanie na pamäť,3* a *čakanie na pamäť,4*, značky v miestach *čakanie na procesor,1*, *čakanie na procesor,2*, *čakanie na procesor,3* a *čakanie na procesor,4* a značky v miestach *nastavenia pripravené,1*, *nastavenia pripravené,2*, *nastavenia pripravené,3* a *nastavenia pripravené,4*, ako je znázornené na Obr. 1.23.

Predstavme si situáciu, že v tomto značkovaní sa v inštancii *1* a v *inštancii 3* otvoria prechody *otvorenie nastavení,1* a *otvorenie nastavení,3*, čím sa sieť dostane do značkovania *čakanie na pamäť,1 + nastavenia,1 + čakanie na procesor,1 + čakanie na pamäť,2 + nastavenia pripravené,2 + čakanie na procesor,2 + čakanie na pamäť,3 + nastavenia,3 + čakanie na procesor,3 + čakanie na pamäť,4 + nastavenia pripravené,4 + čakanie na procesor,4 + source,5 + 2voľná pamäť + 2voľný procesor*, ktoré je znázornené na Obr. 1.24.

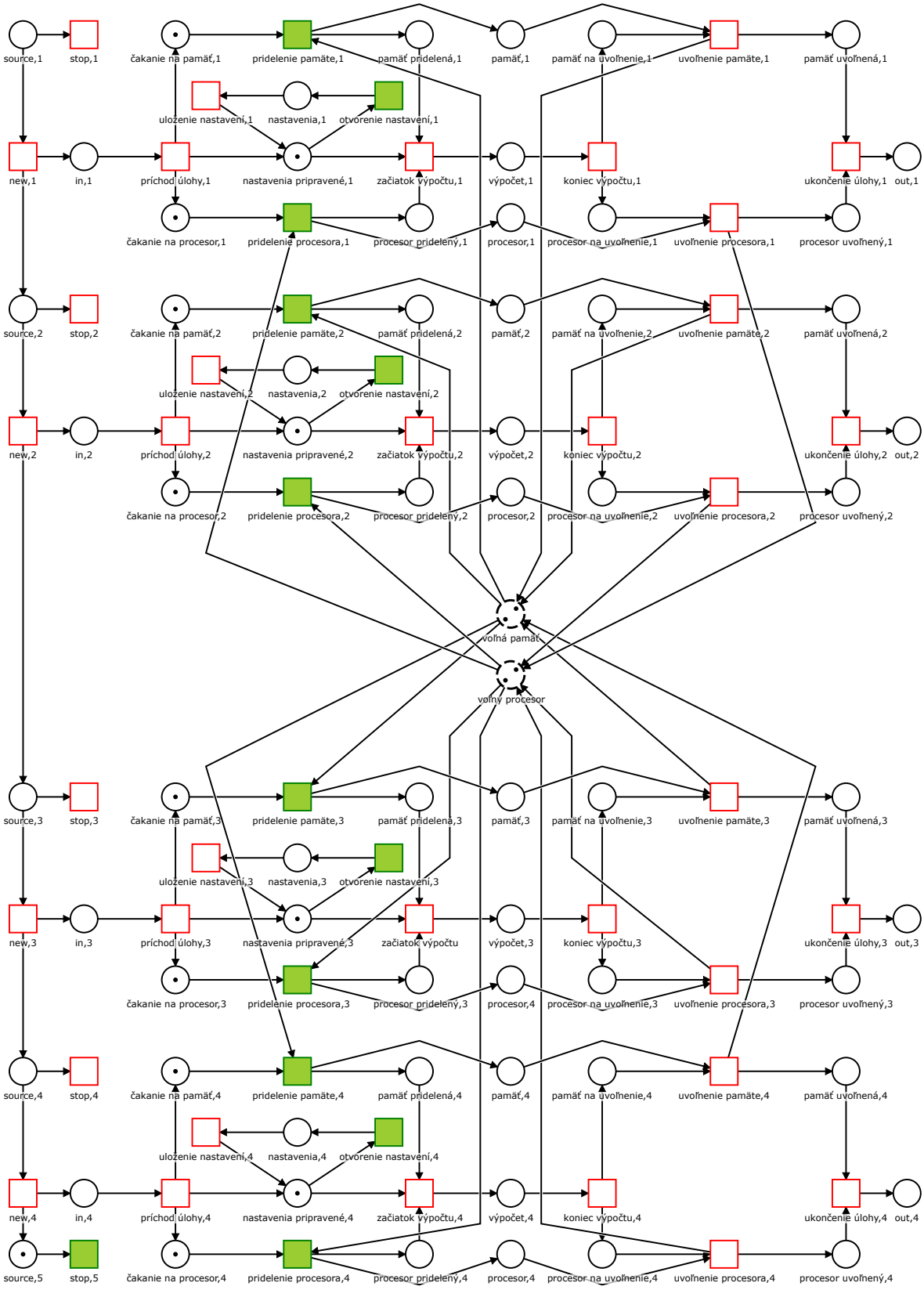
Toto značkovanie zodpovedá stavu, v ktorom 4 výpočtové úlohy čakajú na pridelenie pamäťovej jednotky a pridelenie procesora a v prvej a tretej úlohe prebieha zmena nastavení parametrov. Spustiteľné sú trojica prechodov *pridelenie pamäti,1*, *uloženie nastavení,1* a *pridelenie procesora,1*, trojica prechodov *pridelenie pamäti,2*, *otvorenie*



Obr. 1.21: Graf dosiahnuteľnosti určenej workflow siete so statickými miestami na Obr. 1.17 pre výpočtovú úlohu s možnosťou zmeny nastavenia parametrov výpočtu



Obr. 1.22: Časť vykonávanej workflow siete so statickými miestami pre proces na Obr. 1.17 pre prvé 4 inštancie



Obr. 1.23: Časť vykonávanej workflow siete so statickými miestami pre proces na Obr. 1.17 pre prvé 4 inštancie po načítaní úloh

nastavení,2 a *pridelenie procesora,2*, trojica prechodov *pridelenie pamäti,3*, *uloženie nastavení,3* a *pridelenie procesora,3* a trojica prechodov *pridelenie pamäti,4*, *otvorenie nastavení,2* a *pridelenie procesora,4*. Okrem týchto prechodov je spustiteľný aj prechod *stop,5* a prechod *new,5*, ktorý nie je znázornený na obrázkoch.

Predstavme si situáciu, že v prvých dvoch inštanciách sa spustia prechody *pridelenie pamäti,1* a *pridelenie pamäť,2*, zatiaľ čo v ďalších dvoch inštanciách sa spustia prechody *pridelenie procesora,3* a *pridelenie procesora,4*. Pre zjednodušenie si predstavme, že sa spustí aj prechod *stop,5*.

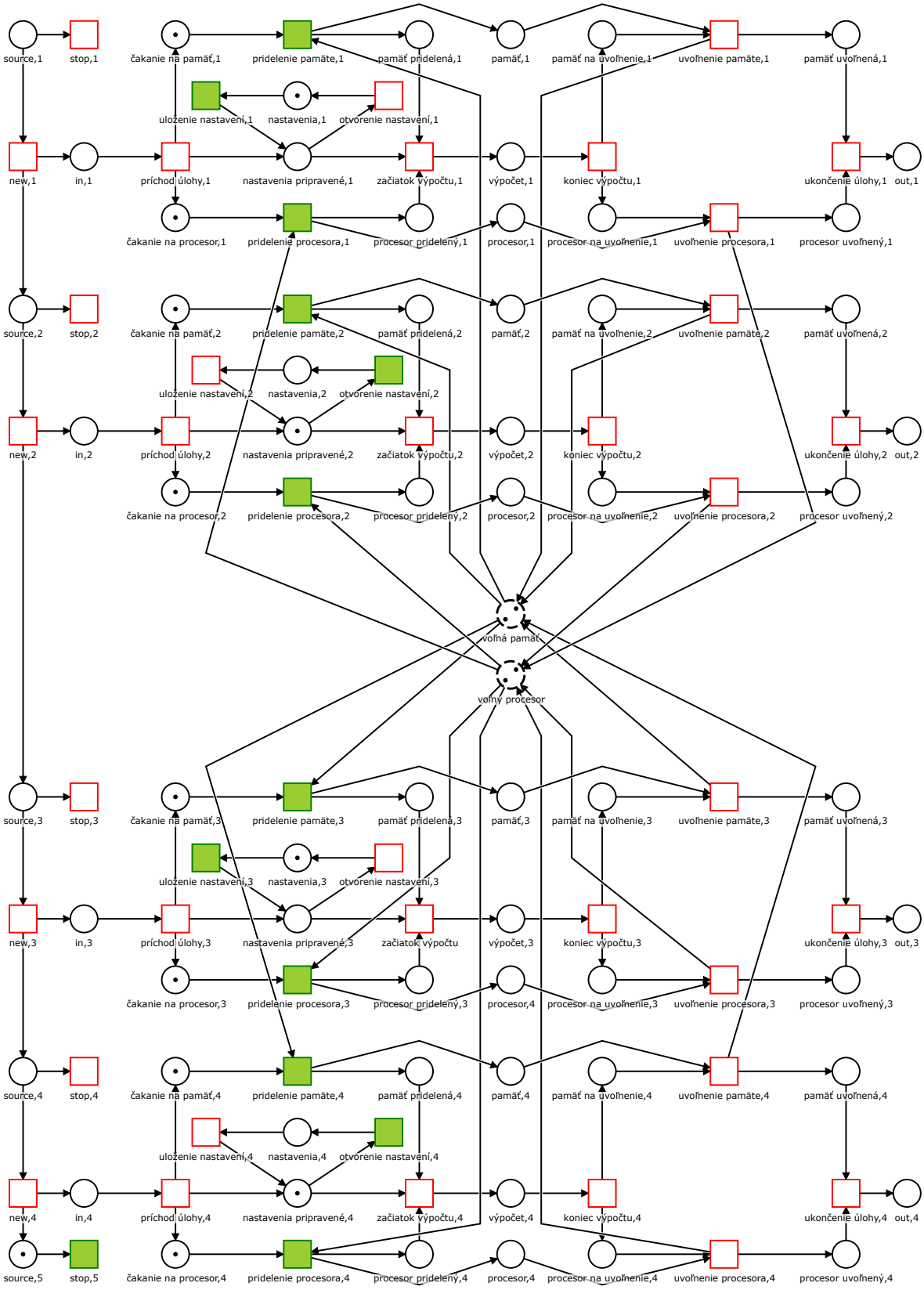
Systém sa dostane do značkovania *pamäť pridelená,1 + nastavenia,1 + čakanie na procesor,1 + pamäť,1 + pamäť pridelená,2 + nastavenia pripravené + čakanie na procesor,2 + pamäť,2 čakanie na pamäť,3 + nastavenia + procesor pridelený,3 + procesor,3 + čakanie na pamäť, 4 + nastavenia pripravené + procesor pridelený,4 + procesor,4*, znázorného na Obr. 1.25.

V tomto značkovaní sú cyklicky spustiteľné iba dvojica prechodov *uloženie nastavení,1* a *otvorenie nastavení,1*, dvojica prechodov *otvorenie nastavení,2* a *uloženie nastavení,2*, dvojica prechodov *uloženie nastavení,3* a *otvorenie nastavení,3*, respektíve dvojica prechodov *otvorenie nastavení,4* a *uloženie nastavení,4*. Napriek tomu, že systém nezamrzol, nie je možné dokončiť žiadnu zo 4 výpočtových úloh. Keďže sme spustili prechod *stop,5*, systém nebude prijímať ďalšie úlohy, napriek tomu nie je možné dosiahnuť stav, v ktorom sa nedá spustiť žiadny prechod. Avšak v žiadnej z inštancií nebude možné prideliť zároveň pamäť a procesor. Žiadnu inštanciu teda nebude možné korektne ukončiť. Toto uviaznutie nazývame zacyklením, respektíve hovoríme, že systém sa zacyklil.

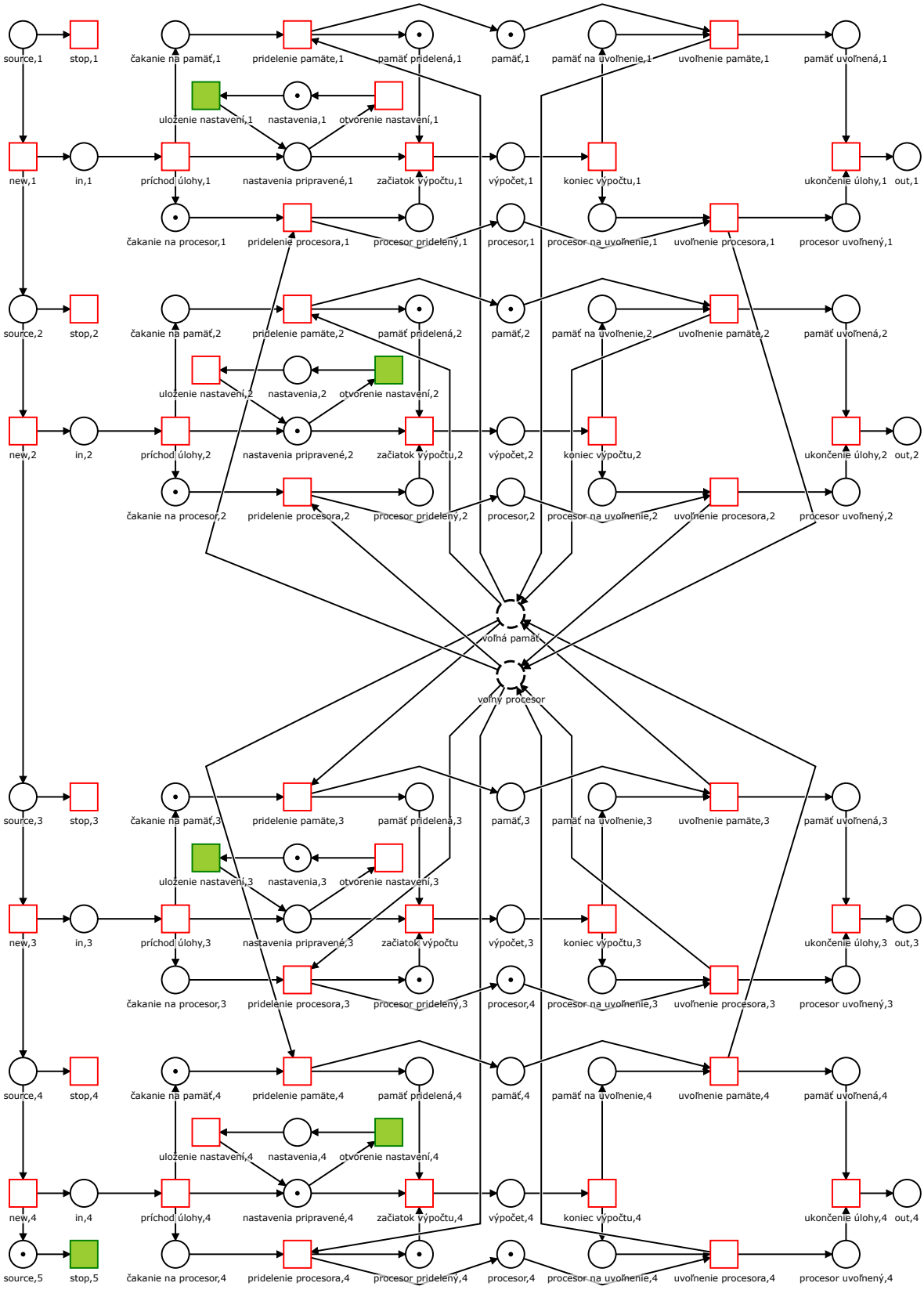
1.6.3 Uviaznutie

Skôr ako si definujeme uviaznutie vykonávanej siete formálne, rozšírime definíciu finálneho stavu tak, aby sme vedeli rozoznať korektne ukončené inštancie. Finálne značkovania sú také značkovania, ktoré majú označené iba niektoré kópie výstupného miesta *out*, pričom statické miesta majú hodnoty rovnaké, ako v počiatočnom značkovaní.

Definícia 24 (Finálne značkovania vykonávanej siete)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a*



Obr. 1.24: Časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie po načítaní úloh, v prvej a tretej úlohe sa menia nastavenia



Obr. 1.25: Časť uviaznutej vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie v stave zacyklenia

nech označkovaná Petriho sieť $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW. *Značkovanie* m_f^∞ *vykonávanej siete* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^∞, *nazývame finálne značkovanie, ak platí, že* $m_f^\infty(p) = 0$ *každé* $p \in P^\infty$, *ktoré nepatrí do množiny* $(\{out\} \times \mathbb{Z}) \cup S$.

Priamo z Definície 24 a z Definície 23 dostávame, že ak vo finálnom značkovaní vykonávanej siete je výstupné miesto označkované pre inštanciu s indexom i, potom je označkované aj výstupné miesto každej inštancie s indexom j menším ako i.

Dôsledok 5 *Nech* m_f^∞ *je ľubovoľné finálne značkovanie vykonávanej siete* MPN *siete určenej* MW *s korektným správaním.*
Ak $m_f^\infty(out, i) = 1$ *pre nejaké* $i \in \mathbb{Z}$, *potom* $m_f^\infty(out, j) = 1$ *pre každé* $j \in \mathbb{Z}$ *také, že* $j < i$.

Formálne budeme uviaznutie definovať ako také značkovanie dosiahnuteľné z počiatočného značkovania m_0^∞, z ktorého nie je možné dosiahnuť žiadne z finálnych značkovaní.

Definícia 25 (Uviaznutie vykonávanej siete)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW. *Značkovanie* m^∞ *vykonávanej siete* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^∞, *nazývame uviaznutie, ak z* m^∞ *nie je dosiahnuteľné žiadne finálne značkovanie vykonávanej siete* MPN.

Hlavným cieľom tejto práce je navrhnúť pre označkované určené workflow siete s korektným správaním metódu na detekciu uviaznutí ich vykonávaných workflow sietí so statickými miestami vrátane algoritmu, ktorý určí, či sieť má uviaznutia. Diskusia problému detekcie uviaznutí, návrh vlastnej metódy a vlastný algoritmus na detekciu uviaznutí tvoria obsah nasledujúcej kapitoly.

Kapitola 2

Detekcia uviaznutí v diskrétnych udalostných systémoch so zdieľanými zdrojmi

Prvý problém pri riešení detekcie uviaznutí je daný faktom, že vykonávaná sieť má nekonečne veľa miest a nekonečne veľa prechodov, preto ju nie je možné efektívne zostrojiť. Počet dosiahnuteľných značkovaní vo vykonávanej sieti je nekonečný. V počiatočnom mieste však má vykonávaná sieť označkovaný konečný počet miest, konkrétne iba miesto *source,1*. To znamená, že ako dôsledok Definície 23 dostávame, že pre každé dosiahnuteľné značkovanie platí, že počet miest, v ktorých sú v danom značkovaní nejaké značky, je konečný. Navyše, keďže v prípade označkovanej určenej workflow siete s korektným správaním vykonávaná sieť pozostáva s kópií určenej workflow siete s korektným správaním, všetky jej miesta sú ohraničené.

Dôsledok 6 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW.

- *Počet dosiahnuteľných značkovaní vo vykonávanej sieti je nekonečný, teda* $[m_0^\infty\rangle$ *je nekonečná množina.*
- *Pre každé značkovanie* m^∞ *vykonávanej siete* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^∞, *platí, že počet miest, pre ktoré je značkovanie* m_0^∞

nenulové, je konečný, teda pre každé $m^\infty \in [m_0^\infty\rangle$ *platí, že jeho nosič* $sup(m^\infty)$ *je konečná množina.*

- *Pre každé značkovanie* m^∞ *vykonávanej siete* MPN*, ktoré je dosiahnuteľné z počiatočného značkovania* m_0^∞*, platí pre ľubovoľné* $i \in \mathbb{Z}$*, že ak* $m^\infty(source, i) > 0$ *potom* $m^\infty(source, i) = 1$ *a* $m^\infty(d, i) = 0$ *pre každé* $d \in D$
- *Zároveň platí, že vykonávaná sieť* MPN *je ohraničená.*

Prvým krokom pri návrhu metódy bude nájsť pre určenú workflow sieť so statickými miestami a korektným správaním takú Petriho sieť s konečným počtom miest a konečným počtom prechodov a definovať jej finálne značkovania a uviaznutia tak, že táto sieť bude mať uviaznutia práve vtedy keď vykonávaná sieť. Druhým krokom bude nájsť algoritmus na detekciu uviaznutí v tejto sieti s konečným počtom miest a prechodov.

2.1 Siete s konštruktorom versus vykonávané siete

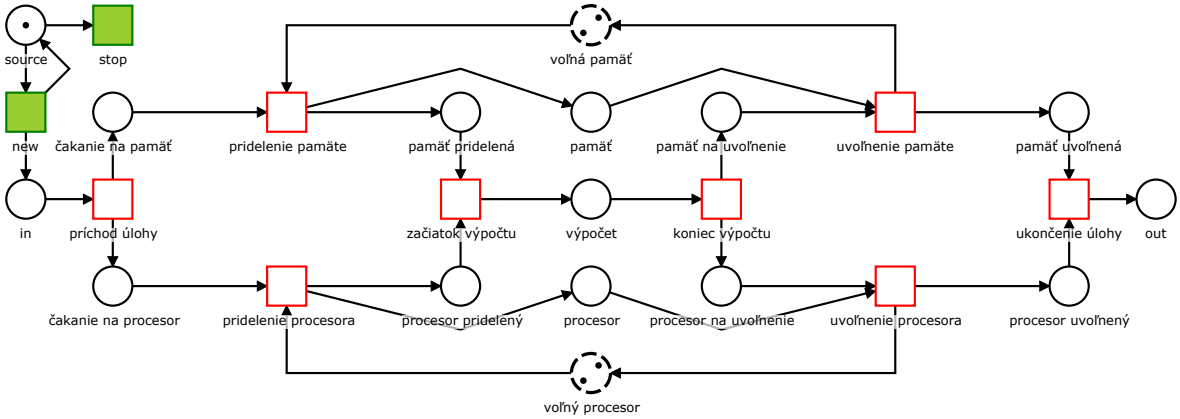
Ako prvý kandidát bude použitá pôvodná workflow sieť, obohatená o konštruktor.

Definícia 26 (Sieť s konštruktorom)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech new, stop a source označujú prvky, ktoré nepatria do množín* D, S *a* T, *teda* $\{new, stop, source\} \cap (D \cup S \cup T) = \emptyset$. *Potom sieť s konštruktorom siete* MW *je označkovaná Petriho sieť* $MPN = (D \cup S \cup \{source\}, T \cup \{new, stop\}, I^c, O^c, m_0^c)$, *kde*

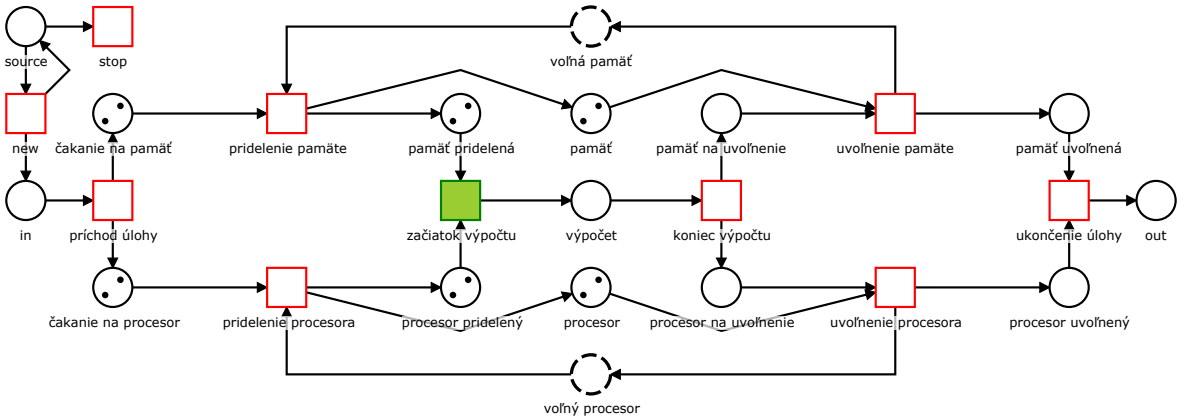
- $I^c(p,t) = I(p,t)$ *pre všetky* $p \in D \cup S$, $t \in T$
- $I^c(source, new) = 1$ *a* $I^c(source, stop) = 1$, *teda spustenie konštruktora new je možné iba v značkovaní s označeným miestom source, podobne zastavenie stop*
- $O^c(p,t) = O(p,t)$ *pre všetky* $p \in D \cup S$, $t \in T$
- $O^c(in, new) = 1$ *a* $O^c(source, new) = 1$ *teda spustenie konštruktora new vytvorí značku v mieste in a zároveň vráti značku do miesta source*
- $I^c(source, t) = 0$ *pre všetky* $t \in T$
- $O^c(source, stop) = 0$, $O^c(p, stop) = 0$ *pre všetky* $p \in P$ *a* $O^c(p, new) = 0$ *pre všetky* $p \in P$ *rôzne od in*
- $m_0^c(source) = 1$, $m_0^c(s) = m_0(s)$ *pre každé* $s \in S$, $m_0^c(d) = 0$ *pre každé* $d \in D$, *teda v počiatočnom značkovaní je jedna značka v mieste source, značky v statických miestach sú skopírované a ostatné miesta sú prázdne.*

Analogicky s vykonávanou sieťou, definujme finálne značkovania siete s konštruktorom.

Definícia 27 (Finálne značkovania siete s konstruktorom)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech označkovaná Petriho sieť* $MPN = (D \cup S \cup \{source\}, T \cup \{new, stop\}, I^c, O^c, m_0^c)$ *je sieť s konštruktorom siete* MW. *Značkovanie* m_f^c *siete* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^c, *nazývame finálne značkovanie, ak platí, že* $m_f^c(p) = 0$ *pre každé* $p \in D \cup \{source\}$ *rôzne od out.*



Obr. 2.1: Sieť s konštruktorom workflow siete z Obr. 1.11.



Obr. 2.2: Sieť s konštruktorom v značkovaní, ktoré zodpovedá uviaznutiu vykonávanej siete na Obr. 1.16

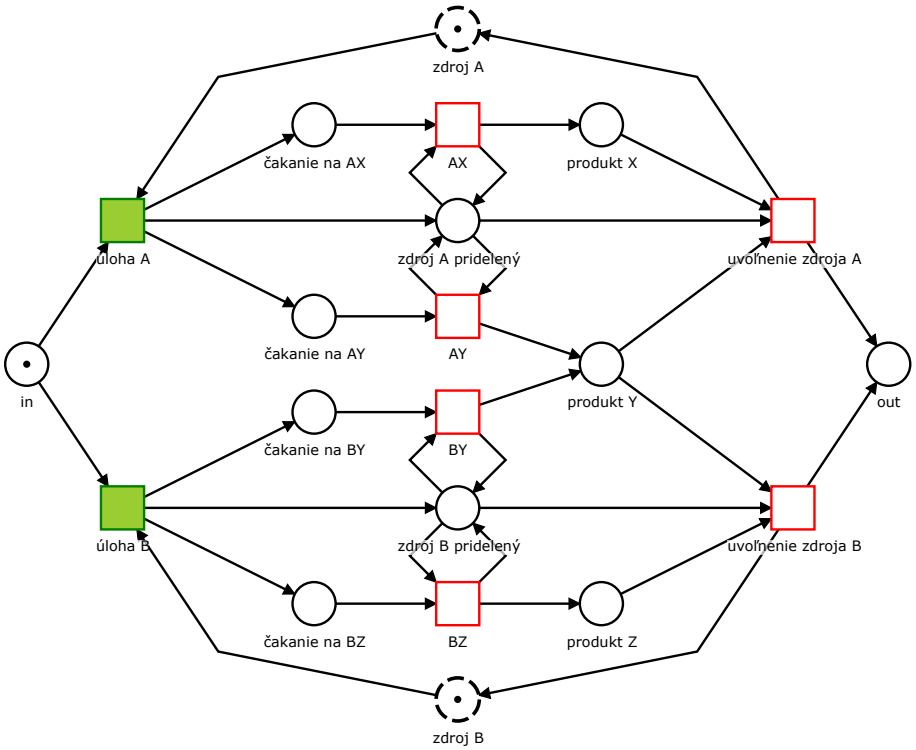
Nakoniec definujme uviaznutie siete s konštruktorom ako také značkovanie dosiahnuteľné z počiatočného značkovania m_0^c, z ktorého nie je možné dosiahnuť žiadne z finálnych značkovaní.

Definícia 28 (Uviaznutie siete s konstruktorom)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech označkovaná Petriho sieť* $MPN = (D \cup S \cup \{source\}, T \cup \{new, stop\}, I^c, O^c, m_0^c)$ *je sieť s konštruktorom siete* MW*. Značkovanie* m^c *siete s konštruktorom* MPN*, ktoré je dosiahnuteľné z počiatočného značkovania* m_0^c*, nazývame uviaznutie, ak z* m^c *nie je dosiahnuteľné žiadne finálne značkovanie siete s konštruktorom* MPN*.*

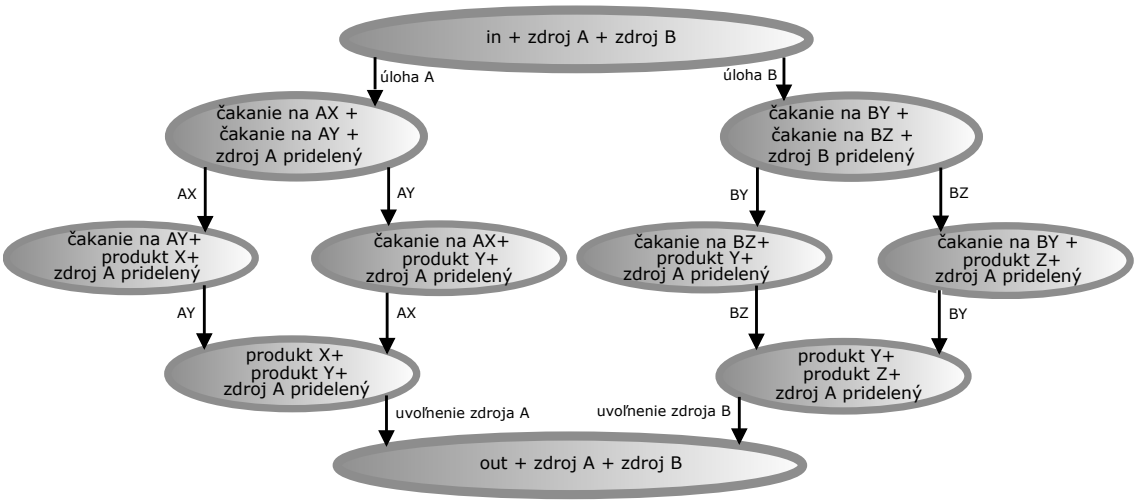
Na Obr. 2.1 je sieť s konštruktorom workflow siete z Obr. 1.11. Žiaľ, sieť s konštruktorom na Obr. 2.1 nemá žiadne uviaznutia, na rozdiel od vykonávanej siete na Obr. 1.13, ktorej uviaznutie je znázornené na Obr. 1.16. Tomuto uviaznutiu zodpovedá značkovanie siete s konštruktorom na Obr. 2.2. Značkovanie siete s konštruktorom na Obr. 2.2 však nie je uviaznutím, pretože je spustiteľný prechod *začiatok výpočtu*. Je to preto, že sieť s konštruktorom nevie rozoznať, ktorá značka patrí ktorej inštancii. To znamená, že pridať k originálnej určenej workflow sieti so statickými miestami a korektným správaním konštruktor nestačí. Predchádzajúci príklad ilustruje, že sieť s konštruktorom totiž nerozozná uviaznutia, ktoré vzniknú vo vykonávanej sieti, ktorá inštancie rozlišuje. Sieť s konštruktorom teda nerozozná existujúce uviaznutie vykonávanej siete.

Napriek tomu, že sieť s konštruktorom nevie rozoznať uviaznutia, analýza dôvodov, prečo je to tak, nám pomôže nájsť riešenie problému. Riešením bude konštrukcia siete s konečným počtom miest a prechodov, ktorá má rovnaké správanie ako pôvodná sieť, a zároveň rozlíši značky z rôznych inštancií. Skôr ako však pristúpime ku konštrukcii takejto siete, uzavrieme analýzu vzťahu medzi vykonávanou sieťou a sieťou s konštruktorom na nasledovnom príklade. Tento príklad ukazuje situáciu, keď sieť s konštruktorom rozozná neexistujúce uviaznutie vykonávanej siete. Uvedieme teda príklad workflow siete, ktorej vykonávaná sieť nemá žiadne uviaznutia avšak sieť s konštruktorom uviaznutie má. Dôvodom je opäť skutočnosť, že sieť s konštruktorom pomieša značky rôznych inštancií.

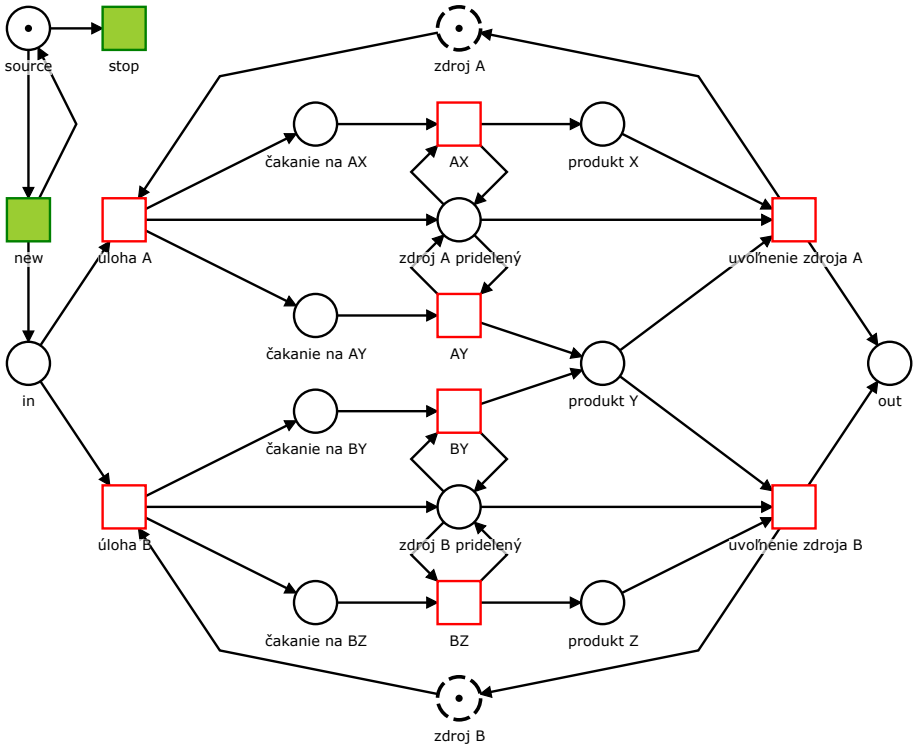
Uvažujme systém, ktorého model vo forme určenej workflow siete so statickými miestami a korektným správaním je na Obr. 2.3: Systém realizuje dva typy pracovných



Obr. 2.3: Systém z dvomi typmi úloh a tromi typmi produktov



Obr. 2.4: Graf dosiahnuuteľnosti siete z Obr. 2.3



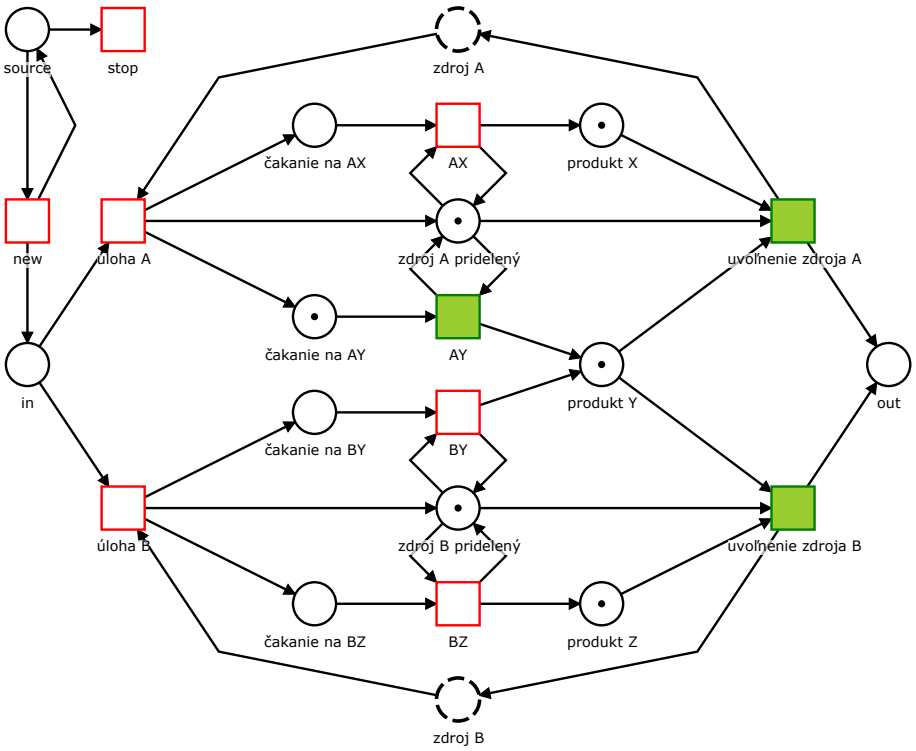
Obr. 2.5: Sieť s konštruktorom workflow siete z Obr. 2.3

úloh, úlohy typu A a úlohy typu B, pričom úlohy typu A realizuje zdroj v role A a úlohy typu B realizuje zdroj v role B. V rámci úlohy A vykonáva zdroj A aktivitu AX, ktorej výsledkom je produkt typu X a aktivitu AY, ktorej výsledkom je produkt typu Y. V rámci úlohy B vykonáva zdroj B aktivitu BY, ktorej výsledkom je produkt typu Y a aktivitu BZ, ktorej výsledkom je produkt typu Z.

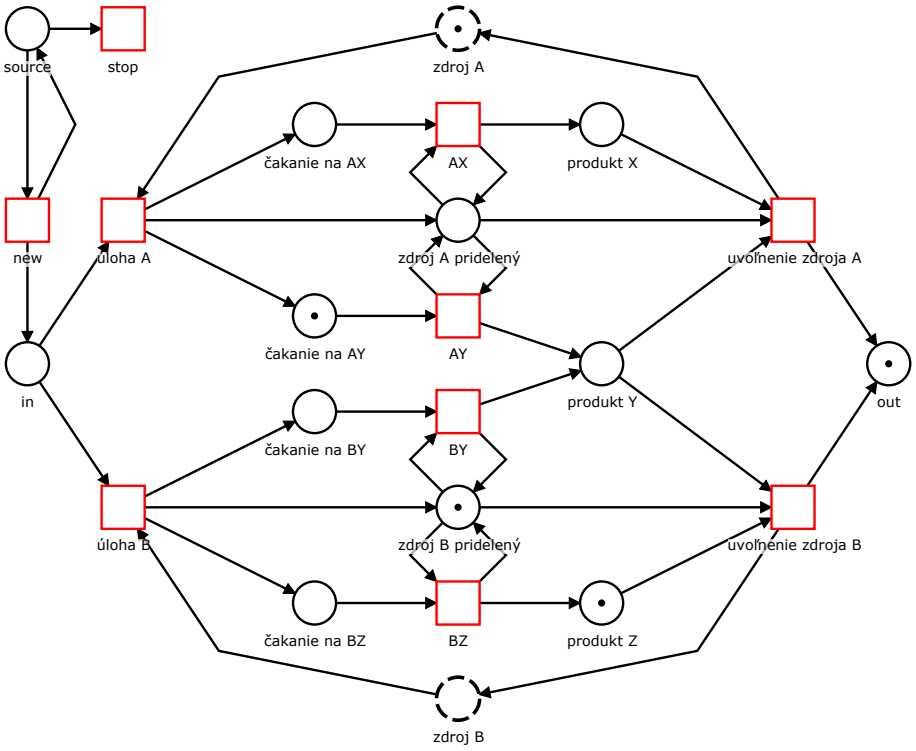
Označkovaná určená workflow sieť so statickými miestami na Obr. 2.3 má korektné správanie. Graf dosiahnuteľnosti workflow siete na Obr. 2.3 je znázornený na Obr. 2.4. Na Obr. 2.5 je znázornená sieť s konštruktorom workflow siete z Obr. 2.3. Ak dvakrát spustíme prechod *new*, ďalej spustíme prechod *stop*, potom prechod *úloha A* a prechod *úloha B* a následne spustíme prechod AX, prechod BY a prechod BZ, dostane sa sieť s konštruktorom do značkovania *čakanie na AY* + *zdroj A pridelený* + *zdroj B pridelený* + *produkt X* + *produkt Y* + *produkt Z*, ktoré je znázornené na Obr. 2.6.

Ak v tomto značkovaní spustíme spustiteľný prechod *uvoľnenie zdroja A*, sieť s konštruktorom uviazne v značkovaní *zdroj A* + *čakanie na AY* + *zdroj B pridelený* + *produkt Z* + *out* na Obr. 2.7.

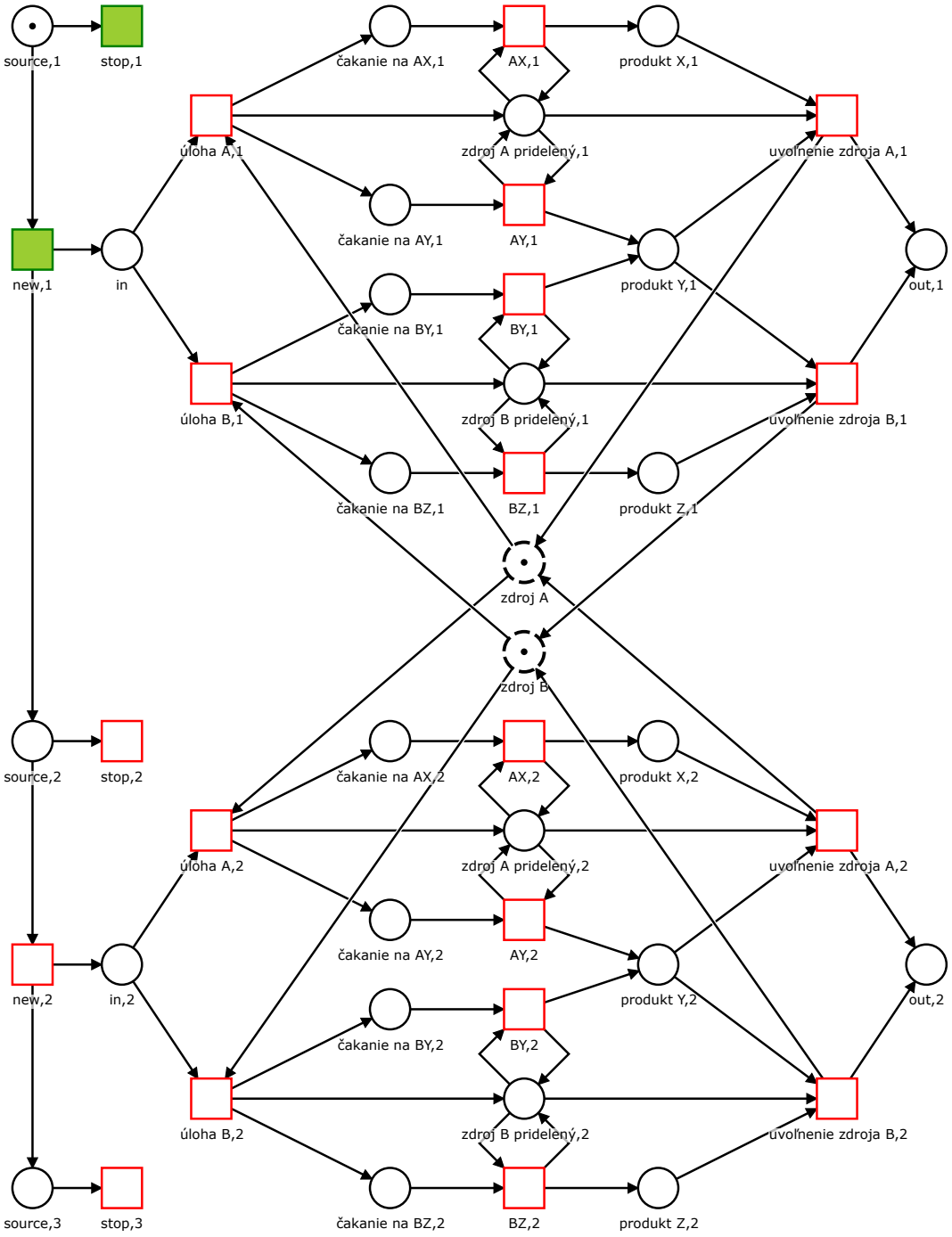
Toto uviaznutie však vo vykonávanej sieti, ktorej časť je zobrazená na Obr. 2.8, neexistuje. Vykonávaná sieť, ktorej časť je na Obr. 2.8, totiž žiadne uviaznutia nemá.



Obr. 2.6: Značkovanie siete z Obr. 2.5 po spustení prechodov *new, new, stop, úloha A, úloha B, AX, BY, BZ*



Obr. 2.7: Uviaznutie siete s konštruktorom z Obr. 2.5 po spustení prechodov *new, new, stop, úloha A, úloha B, AX, BY, BZ, uvoľnenie zdroja A*



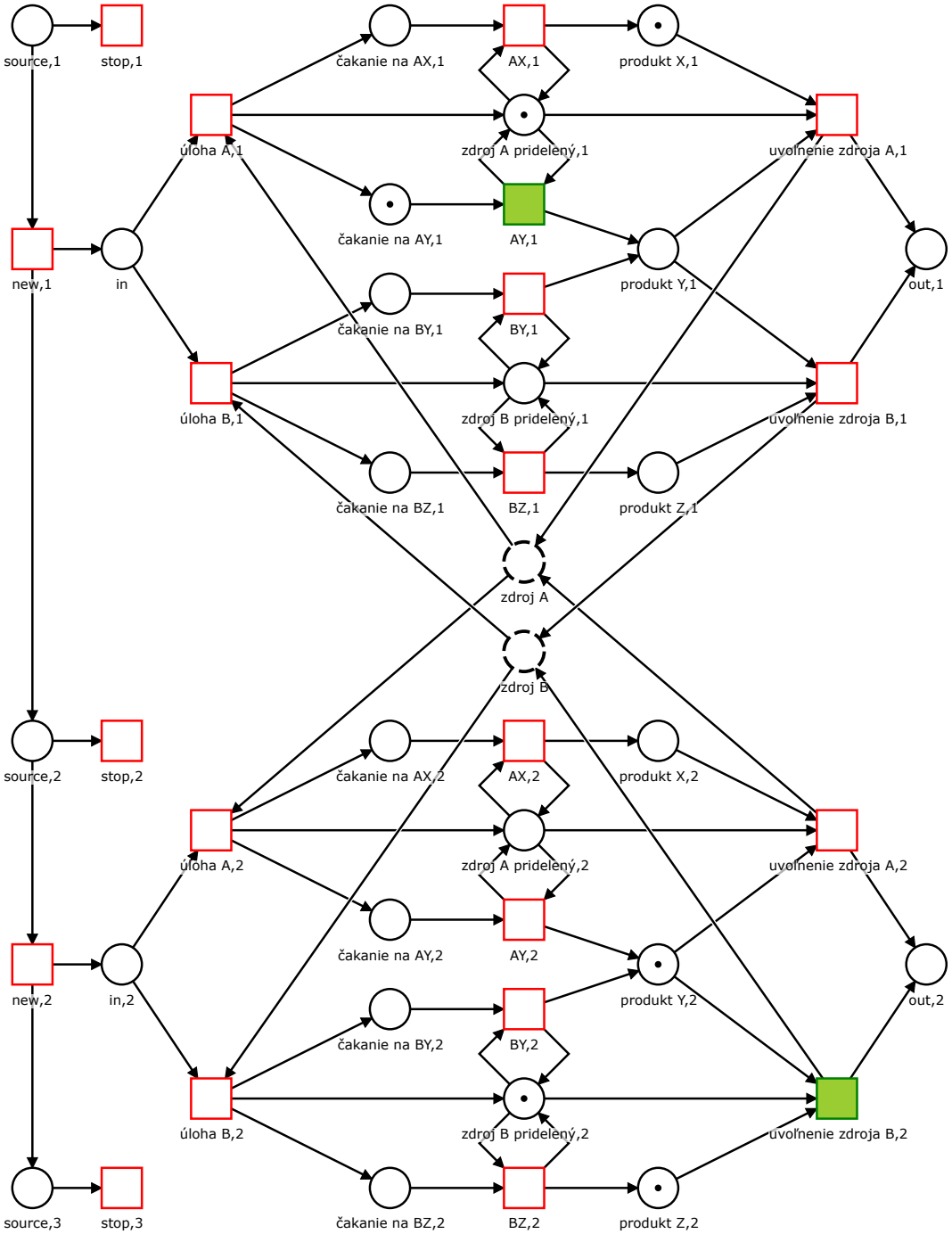
Obr. 2.8: Časť vykonávanej siete pre označkovanú workflow sieť z Obr. 2.3

Uviaznutie siete s konštruktorom je spôsobené pomiešaním značiek, konkrétne značka v mieste *produkt Y* vznikla spustením prechodu BY úlohy B ale bola použitá prechodom *uvoľnenie zdroja A* úlohy A v značkovaní *čakanie na AY + zdroj A pridelený + zdroj B pridelený + produkt X + produkt Y + produkt Z*, ktoré je znázornené na Obr. 2.6. Zodpovedajúce značkovanie vo vykonávanej sieti *čakanie na AY,1 + zdroj A pridelený,1 + zdroj B2 pridelený,2 + produkt X,1 + produkt Y,2 + produkt Z,2* dosiahnuté spustením postupnosti prechodov *new,1 new,2 stop,3 úloha A,1 úloha B,1 AX,1 BY,1 BZ,2*, ktoré je znázornené na Obr. 2.9, takéto miešanie značiek rôznych inštancií neumožňuje. Toto značkovanie zodpovedá dvom spusteným inštanciám, prvá inštancia realizuje úlohu A, zatiaľ čo druhá inštancia realizuje úlohu B.

2.2 Siete dosiahnuteľnosti

Aby sme zabránili miešaniu značiek, potrebujeme separovať dosiahnuteľné značkovania dynamických miest originálnej workflow siete. Jedným z možných spôsobov, ako to dosiahnuť je inšpirovať sa grafom dosiahnuteľnosti originálnej siete. S pomocou grafu dosiahnuteľnosti vytvoríme Petriho sieť so statickými miestami a doplníme ju konštruktorom. Takúto sieť nazveme sieť dosiahnuteľnosti. Najskôr však definujme matematickú notáciu, ktorú budeme ďalej používať, ako pojem zúženia funkcie, zjednotenia funkcií a ďalších pojmov.

Definícia 29 (Matematická notácia)
Nech P je nejaká množina, A je nejaká množina a nech D je nejaká podmnožina množiny P, teda $D \subseteq P$. Nech $m : P \to A$ je nejaká funkcia, ktorá priraďuje každému prvku množiny P nejaký prvok z množiny A. Nech $m|D$ označuje zúženie funkcie m na prvky množiny D, teda takú funkciu $m|D : D \to A$ z D do A, že $m|D(d) = m(d)$ pre každé $d \in D$. Nech S je nejaká podmnožina množiny P, teda $S \subseteq P$, a zároveň nech D a S nemajú spoločné prvky, teda $D \cap S = \emptyset$ (D a S sú disjunktné), potom označme $m|D \cup m|S = m|(D \cup S)$. Označme zároveň symbolom $P \setminus D$ rozdiel množín P a D, teda takú množinu, že $P \cap (P \setminus D) = \emptyset$ a zároveň $(P \setminus D) \cup D = P$ (definujeme teda rozdiel množin iba pre prípad, že množina D je podmnožinou množiny P). Nech $[P \to A]$ označuje množinu všetkých funkcií z P do A. Ak A je konečná množina, nech $|A|$ označuje počet jej prvkov.



Obr. 2.9: Časť vykonávanej siete pre označkovanú workflow sieť z Obr. 2.3 v značkovaní zodpovedajúcom značkovaniu predchádzajúcemu uviaznutiu siete s konštruktorom z Obr. 2.6

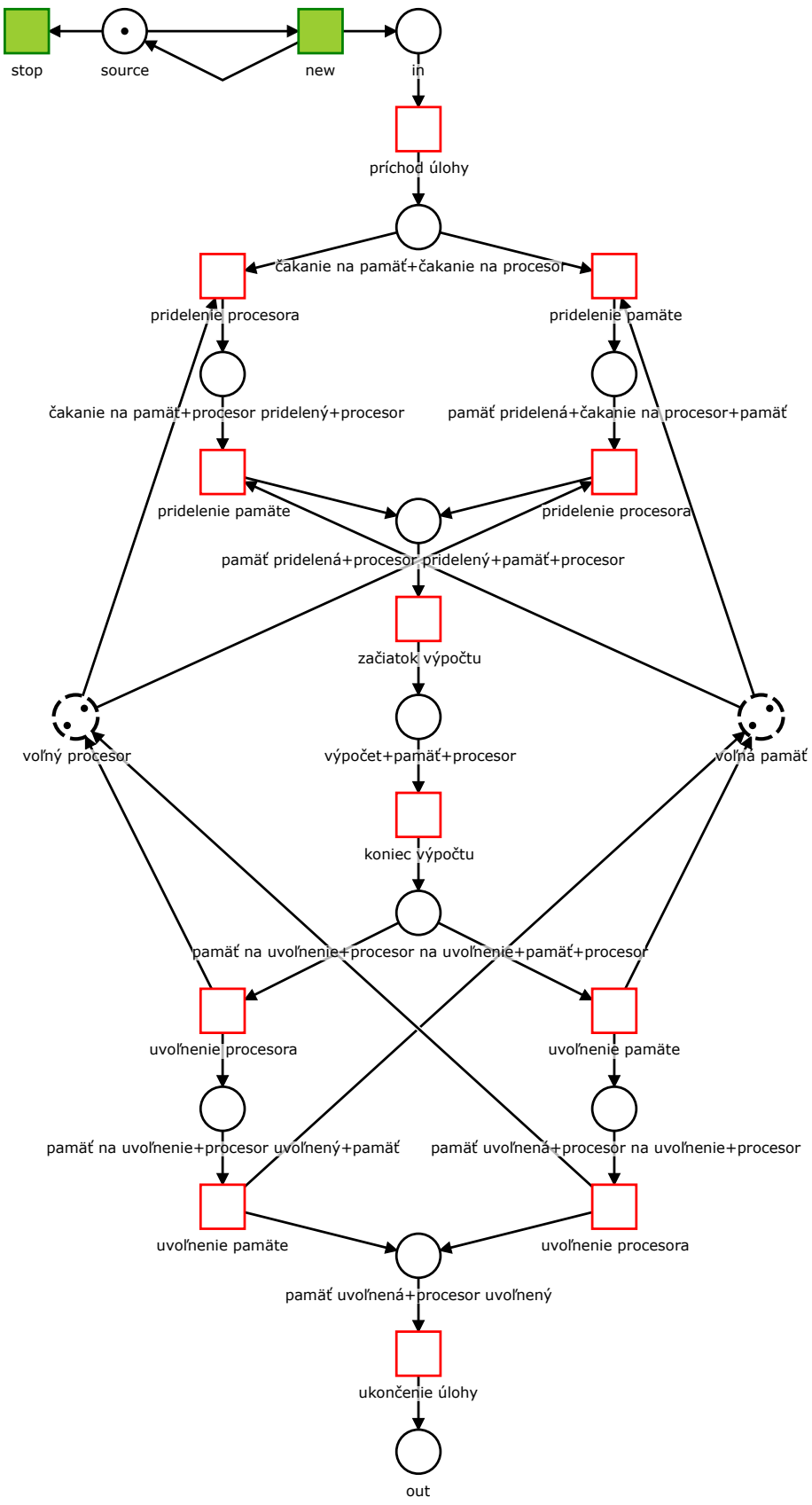
Definícia 30 (Množina dosiahnuteľných D-značkovaní)
Nech $MPN = (P, T, I, O, m_0)$ je označkovaná Petriho sieť. Nech D je nejaká podmnožina miest, teda $D \subseteq P$. Nech m je značkovanie MPN. Potom funkciu $m|D$ nazývame D-značkovanie. Symbolom $[m_0\rangle|D$ označme množinu všetkých takých D-značkovaní w, že platí: $w \in [m_0\rangle|D$ práve vtedy, keď existuje také $m \in [m_0\rangle$, že $w = m|D$. Symbol $[m_0\rangle|D$ teda označuje množinu všetkých takých D-značkovaní $m|D$, že m je dosiahnuteľné z m_0. Hovoríme tiež, že $[m_0\rangle|D$ označuje množinu všetkých D-značkovaní dosiahnuteľných z počiatočného D-značkovania $m_0|D$.

Miesta v sieti dosiahnuteľnosti budú vytvorené s pomocou dosiahnuteľných D-značkovaní, teda značkovaní dynamických miest originálnej označkovanej workflow siete. Pre každé dosiahnuteľné D-značkovanie $m|D$ z $[m_0\rangle|D$ originálnej siete vytvoríme jedno miesto v sieti dosiahnuteľnosti. K takto vytvoreným miestam pridáme ako miesta pôvodné statické miesta. Prvky prechodovej relácie (m, t, m') z $\longrightarrow$ využijeme pri vytváraní prechodov siete dosiahnuteľnosti. Trojica $(m|D, t, m'|D) \in ([m_0\rangle|D) \times T \times ([m_0\rangle|D)$ bude prechodom siete dosiahnuteľnosti vždy, keď $m \xrightarrow{t} m'$. Prechod $(m|D, t, m'|D)$ siete dosiahnuteľnosti spotrebuje jednu značku z miesta siete dosiahnuteľnosti $m|D$ a vytvorí jednu značku v mieste $m'|D$. Hrany medzi statickými miestami a prechodom $(m|D, t, m'|D)$ budú totožné s hranami medzi statickými miestami a prechodom t originálnej siete. Takto vzniknuté miesta a prechody obohatíme o konštruktor obdobným spôsobom ako v prípade siete s konštruktorom z Definície 26.

Keďže z miesta siete dosiahnuteľnosti $m|D$, prechodu t originálnej siete a miesta siete dosiahnuteľnosti $m'|D$ vieme jednoznačne určiť prechod $(m|D, t, m'|D)$ siete dosiahnuteľnosti, v grafickom znázornení označíme prechod $(m|D, t, m'|D)$ iba názvom prechodu t.

Pre lepšiu ilustráciu, na Obr. 2.10 zobrazujeme sieť dosiahnuteľnosti pre označkovanú určenú workflow sieť so statickými miestami a korektným správaním z Obr. 1.11. Porovnaním siete dosiahnuteľnosti na Obr. 2.10 s grafom dosiahnuteľnosti na Obr. 1.12 vidíme, že počiatočnému značkovaniu *in + 2pamäť + 2procesor* v grafe dosiahnuteľnosti zodpovedá značkovanie *in* v sieti dosiahnuteľnosti. Značkovaniu *čakanie na pamäť + procesor pridelený + procesor + voľný procesor + 2voľná pamäť* z grafu dosiahnuteľnosti zodpovedá miesto *čakanie na pamäť + procesor pridelený + procesor*.

Nasledujúca definícia formalizuje sieť dosiahnuteľnosti.



Obr. 2.10: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 1.11

Definícia 31 (Sieť dosiahnuteľnosti)

Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech new, stop a source ozančujú prvky, ktoré nepatria do množín* D, S *a* T, *teda* $\{new, stop, source\} \cap (D \cup S \cup T) = \emptyset$. *Potom sieť dosiahnuteľnosti siete* MW *je označkovaná Petriho sieť* $MPN = (P^r, T^r, I^r, O^r, m_0^r)$, *kde*

- $P^r = ([m_0\rangle|D) \cup S \cup \{source\}$, *teda miestami sú všetky D-značkovania dosiahnuteľné z počiatočného značkovania siete* MW, *statické miesta a miesto source.*
- $T^r = T^a \cup \{new, stop\}$, *kde* T^a *je množina všetkých trojíc* $(x, t, y) \in ([m_0\rangle|D) \times T \times ([m_0\rangle|D)$, *pre ktoré platí, že existuje taká trojica* $(m, t, m') \in [m_0\rangle \times T \times [m_0\rangle$, *že* $x = m|D$, $y = m'|D$ *a zároveň* $m \xrightarrow{t} m'$, *teda prechodmi siete dosiahnuteľnosti sú okrem new a stop také trojice* (x, t, y), *kde* x *a* y *sú D-značkovania a* t *je spustiteľný v značkovaní* m *a jeho spustenie vedie do značkovania* m', *pričom* $x = m|D$ *a* $x = m|D$.
- $I^r(x, (x, t, y)) = 1$ *pre všetky* $(x, t, y) \in T^a$
- $I^r(source, new) = 1$ *a* $I^r(source, stop) = 1$, *teda spustenie konštruktora new je možné iba v značkovaní s označeným miestom source, podobne zastavenie stop*
- $I^r(s, (x, t, y)) = I(s, t)$ *pre všetky* $s \in S$, $(x, t, y) \in T^a$, *teda hrany zo statických miest do kópií prechodov sa kopírujú*
- $O^r((x, t, y), y) = 1$ *pre všetky* $(x, t, y) \in T^a$
- $O^r(in, new) = 1$ *a* $O^r(source, new) = 1$ *teda spustenie konštruktora new vytvorí značku v mieste in a zároveň vráti značku do miesta source (pripomeňme, že z Definície 19 platí, že značkovanie* $in = m_0|D$*)*
- $O^r(s, (x, t, y)) = O(s, t)$ *pre všetky* $s \in S$, $(x, t, y) \in T^a$, *teda hrany z kópií prechodov do statických miest sa kopírujú*
- $I^r(p, t) = 0$ *a* $O^r(p, t) = 0$ *pre všetky ostatné dvojice* $(p, t) \in (P^r \times T^r)$
- $m_0^r(source) = 1$, $m_0^r|S = m_0|S$ *a* $m_0^r(x) = 0$ *pre každé* $x \in [m_0\rangle|D$, *teda v počiatočnom značkovaní je jedna značka v mieste source, značky v statických miestach sa zachovávajú a miesta zodpovedajúce dosiahnuteľným D-značkovaniam sú prázdne.*

Existencia určujúcich miest v určenej workflow sieti so statickými miestami zabezpečuje, že pre všetky dosiahnuteľné značkovania m a m' platí: ak zúženie značkovania $m|D$ sa rovná zúženiu značkovania $m'|D$ potom značkovanie m sa rovná značkovaniu m'.

Dôsledok 7 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami. Potom pre všetky značkovania m a m' z $[m_0\rangle$ platí: ak $m|D = m'|D$ potom $m = m'$.*

Pripomeňme zároveň, že v prípade označkovanej určenej workflow siete s korektným správaním je podľa Lemy 3 počet dosiahnuteľných značkovaní konečný a teda aj počet dosiahnuteľných D-značkovaní je konečný.

Dôsledok 8 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním. Potom jej sieť dosiahnuteľnosti $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ má konečný počet miest P^r a konečný počet prechodov T^r.*

Zostrojiť sieť dosiahnuteľnosti pre označkovanú určenú workflow sieť je teda možné jednoduchou modifikáciou Algoritmu 3. Vstupom do tohto modifikovaného algoritmu je označkovaná určená workflow sieť so statickými miestami. Výstupom je informácia, či označkovaná sieť je korektná a ak áno, potom je výstupom jej sieť dosiahnuteľnosti. Pripomeňme, že funckiu I^r, respektíve funkciu O^r môžeme chápať ako množinu (zoznam) trojíc $(p^r, t^r, n) \in P^r \times T^r \times \mathbb{N}$. Podobne m_0^r môžeme chápať ako množinu (zoznam) dvojíc $(p^r, n) \in P^r \times \mathbb{N}$.

Algoritmus 4 (Overenie korektného správania a vytvorenie siete dosiahnuteľnosti určenej workflow siete so statickými miestami)
Pre označkovanú určenú workflow sieť so statickými miestami $MW = (D, S, T, I, O, m_0)$

1. vytvor prázdny zoznam M nájdených značkovaní workflow siete MW
2. vytvor premennú m_f *je dosiahnuteľné* a vlož do nej hodnotu *nie*
3. vytvor prázdny zoznam miest P^r siete dosiahnuteľnosti
4. vytvor prázdny zoznam prechodov T^r siete dosiahnuteľnosti
5. vytvor prázdnu zoznam pre funkciu I^r

6. vytvor prázdnu zoznam pre funkciu O^r
7. vlož do zoznamu M počiatočné značkovanie m_0 a nastav množinu $Pre(m_0)$ jeho predchodcov prázdnu
8. vlož do zoznamu P^r miesto siete dosiahnuteľnosti $m_0|D$
9. pokiaľ je v zozname M značkovanie m, ktoré nie je označené ako preskúmané, vezmi ho a rob nasledovné:
   (a) ak $m(out) \geq 1$ a zároveň $m \neq m_f$ potom algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"
   (b) ak $m = m_f$ vlož do m_f *je dosiahnuteľné* hodnotu *áno*
   (c) pre každý prechod t spustiteľný z m
      i. počítaj značkovanie m' dosiahnuté spustením prechodu t zo značkovania m
      ii. ak m' ešte nie je v zozname M
         A. vlož m' do zoznamu M a nastav množinu $Pre(m')$ jeho predchodcov prázdnu
         B. do zoznamu P^r vlož $m'|D$
         C. vlož trojicu $(m'|D, y, 0)$ do zoznamu I^r aj do zoznamu O^r pre každé y zo zoznamu T^r
      iii. nastav množinu $Pre(m')$ predchodcov značkovania m' rovnú zjednoteniu $Pre(m') \cup Pre(m) \cup \{m\}$
      iv. ak existuje predchodca m'' z $Pre(m')$ taký, že pre všetky miesta $p \in P$ platí $m''(p) \leq m'(p)$ a zároveň existuje miesto $p \in P$ také, že platí $m''(p) < m'(p)$, potom algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"
      v. vlož do zoznamu T^r trojicu $(m|D, t, m'|D)$
      vi. vlož trojicu $(x, (m|D, t, m'|D), 0)$ do zoznamu I^r pre každé x zo zoznamu P^r rôzne od $m|D$
      vii. vlož trojicu $(m|D, (m|D, t, m'|D), 1)$ do zoznamu I^r

viii. vlož trojicu $(x, (m|D, t, m'|D), 0)$ do zoznamu O^r pre každé x zo zoznamu P^r rôzne od $m'|D$

ix. vlož trojicu $(m|D, (m|D, t, m'|D), 1)$ do zoznamu O^r

(d) označ m ako preskúmané

10. ak m_f *je dosiahnuteľné* = *nie* algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"

11. ak m_f *je dosiahnuteľné* = *áno* a $Pre(m_f) \cup \{m_f\} \neq M$, algoritmus zastav a vráť informáciu "Sieť nemá korektné správanie"

12. vlož *source* do zoznamu P^r

13. vlož trojicu $(source, y, 0)$ do zoznamu I^r aj do zoznamu O^r pre každé y zo zoznamu T^r

14. vlož *new* do zoznamu T^r

15. vlož trojicu $(x, new, 0)$ do zoznamu I^r pre každé x zo zoznamu P^r rôzne od *source*

16. vlož trojicu $(source, new, 1)$ do zoznamu I^r

17. vlož trojicu $(x, new, 0)$ do zoznamu O^r pre každé x zo zoznamu P^r rôzne od *source* a *in*

18. vlož trojicu $(source, new, 1)$ do zoznamu O^r

19. vlož trojicu $(in, new, 1)$ do zoznamu O^r

20. vlož *stop* do zoznamu T^r

21. vlož trojicu $(x, stop, 0)$ do zoznamu I^r pre každé x zo zoznamu P^r rôzne od *source*

22. vlož trojicu $(source, stop, 1)$ do zoznamu I^r

23. vlož trojicu $(x, stop, 0)$ do zoznamu O^r pre každé x zo zoznamu P^r

24. vytvor prázdny zoznam m_0^r

25. vlož dvojicu $(x, 0)$ do zoznamu m_0^r pre každé x zo zoznamu P^r rôzne od *source*

26. vlož dvojicu $(source, 1)$ do zoznamu m_0^r

27. vráť informáciu "Sieť má korektné správanie" a zároveň vráť sieť dosiahnuteľnosti $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$

Miesta v sieti dosiahnuteľnosti reprezentujú značkovania dynamických miest, teda stavy jednotlivých inštancií, vrátane informácie, koľko značiek zo statických premenných inštancia práve používa. Značky v miestach siete dosiahnuteľnosti reprezentujú jednotlivé inštancie. Počet značiek v danom mieste siete dosiahnuteľnosti reprezentuje teda počet inštancií vo vykonávanej sieti, ktoré sú v danom stave.

Definícia 32 (Finálne značkovania siete dosiahnuteľnosti)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech označkovaná Petriho sieť* $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Značkovanie* m_f^r *siete dosiahnuteľnosti* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^r, *nazývame finálne značkovanie, ak platí, že* $m_f^r(x) = 0$ *každé* $x \in P^r \setminus (S \cup \{out\})$ *(kde out označuje v zmysle zavedenej notácie funkciu* $1 \cdot out$ *z* D *do* $\mathbb{Z}$*).*

Uviaznutie siete dosiahnuteľnosti je definované ako jej značkovanie, z ktorého nie je možné dosiahnuť žiadne jej finálne značkovanie.

Definícia 33 (Uviaznutie siete dosiahnuteľnosti)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami a nech označkovaná Petriho sieť* $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Značkovanie* m^r *siete dosiahnuteľnosti* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^r, *nazývame uviaznutie, ak z* m^r *nie je dosiahnuteľné žiadne finálne značkovanie siete dosiahnuteľnosti* MPN.

2.3 Siete dosiahnuteľnosti versus vykonávané siete

Pripomeňme z Dôsledku 6, že ak miesto $source, i$ vo vykonávanej sieti označkovanej určenej workflow siete s korektným správaním obsahuje značku, potom všetky ostatné miesta i-tej inštancie vo vykonávanej sieti sú prázdne.

Ak vo vykonávanej sieti označkovanej určenej workflow siete s korektným správaním spúšťame iba prechody $new, 1$ až new, i a následne prechody iba pre inštanciu i, potom sú dosiahnuteľné pre i-tu inštanciu všetky dosiahnuteľné D-značkovania originálnej siete a inštanciu je možné korektne ukončiť, keďže inštancia nesúťaží o značky v statických miestach so žiadnou inou inštanciou. Z Dôsledku 4 vyplýva, že ak sa vykonávajú prechody viacerých inštancií, počet značiek v statických miestach sa nezvyšuje a teda v žiadnej inštancii nemôže byť dosiahnuté D-značkovanie, ktoré nie je dosiahnuteľné v originálnej sieti. Navyše platí, že označených môže byť iba konečný počet za sebou idúcich inštancií počnúc prvou inštanciou.

Dôsledok 9 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW. *Pre každé značkovanie* m^∞ *vykonávanej siete* MPN, *ktoré je dosiahnuteľné z počiatočného značkovania* m_0^∞, *platí pre ľubovoľné* $i \in \mathbb{Z}$ *práve jedna z nasledovných možností:*

1. $m^\infty|(P_i^\infty) = (source, i)$ *(kde* $(source, i)$ *označuje v zmysle zavedenej notácie funkciu* $1 \cdot (source, i)$ *z* (P_i^∞) *do* $\mathbb{Z}$*), teda označené je iba miesto* $(source, i)$, *ostatné miesta z* (P_i^∞) *nemajú žiadne značky.*

2. $m^\infty|(P_i^\infty) \in [m_0\rangle|D \times i$, *teda i-ta inštancia je v niektorom z dosiahnuteľných stavov siete* MW, *pričom existuje miesto* $(d, i) \in D \times \{i\}$ *také, že* $m^\infty(d, i) > 0$, *teda* (d, i) *obsahuje aspoň jednu značku.*

3. $m^\infty|(P_i^\infty) = 0$, *t.j.* $m^\infty(x) = 0$ *pre každé* $x \in P_i^\infty$, *teda miesta i-tej inštancie sú neoznačené, čo zodpovedá skutočnosti, že inštancia nebola vytvorená.*

Navyše pre každé $m^\infty \in [m_0^\infty\rangle$ *existuje* $i \in \mathbb{Z}$ *také, že* $m^\infty|(P_i^\infty) = 0$. *Zárovň platí, že ak* $m^\infty|(P_i^\infty) = 0$ *potom* $m^\infty|(P_j^\infty) = 0$ *pre všetky* $i, j \in \mathbb{Z}$ *také, že* $j > i$.

Z predchádzajúcich výsledkov vyplýva, že pre každé dosiahnuteľné značkovanie vykonávanej siete označkovanej určenej workflow siete s korektným správaním platí, že indexy nenulových, teda vytvorených inštancií tvoria množinu indexov. Každému značkovaniu miest vykonávanej siete, v ktorom indexy nenulových inštancií tvoria množinu indexov v zmysle definície 2, môžeme priradiť funkciu udávajúcu koľko inštancií sa nachádza v jednotlivých stavoch modelovaných nestatickými miestami siete dosiahnuteľnosti.

Definícia 34 (Počet inštancií v rovnakom stave)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním, nech označkovaná Petriho sieť* $MPN^{\infty} = (P^{\infty}, T^{\infty}, I^{\infty}, O^{\infty}, m_0^{\infty})$ *je vykonávaná workflow sieť so statickými miestami siete* MW *a nech označkovaná Petriho sieť* $MPN^{r} = (P^{r}, T^{r}, I^{r}, O^{r}, m_0^{r})$ *je sieť dosiahnuteľnosti siete* MW. *Uvažujme ľubovoľné značkovanie* $m^{\infty} : P^{\infty} \to \mathbb{N}$, *pre ktoré existuje taká konečná množina indexov* $\mathbb{I}(m^{\infty})$, *že*

- *pre každé* $i \leq max_{\mathbb{I}(m^{\infty})}$ *platí buď možnosť 1 alebo možnosť 2 z Dôsledku 9 a*
- *pre každé* $j > max_{\mathbb{I}(m^{\infty})}$ *platí možnosť 3 z Dôsledku 9.*

Takéto značkovanie budeme nazývať potencionálne značkovanie vykonávanej siete. Potencionálne značkovanie m^{∞} *má teda v miestach prvých* $max_{\mathbb{I}(m^{\infty})}$ *inštanciách aspoň jednu značku v aspoň jednom mieste, zatiaľ čo všetky miesta inštancií s indexom vyšším ako* $max_{\mathbb{I}(m^{\infty})}$ *sú neoznačkované.*

- *Označme symbolom* $f(m^{\infty}) : \mathbb{I}(m^{\infty}) \to (P^{r} \setminus S)$ *takú funkciu, ktorá priradí číslu* i:
  - *miesto siete dosiahnuteľnosti* $f(m^{\infty})(i) = source$, *ak* $m^{\infty}(source, i) = 1$.
  - *miesto siete dosiahnuteľnosti* $f(m^{\infty})(i) \in [m_0\rangle | D$ *také, že* $f(m^{\infty})(i)(d) = m^{\infty}(d, i)$ *pre každé* $d \in D$, *ak* $m^{\infty}(source, i) = 0$.
- *Nech* $x \in P^{r} \setminus S$, *potom* $\mathbb{I}(m^{\infty}, x) \subseteq \mathbb{I}(m^{\infty})$ *je taká konečná množina kladných celých čísel* i, *pre ktoré platí* $f(m^{\infty})(i) = x$, *teda* $\mathbb{I}(m^{\infty}, x)$ *je množina indexov takých inštancií, ktoré sú v rovnakom stave* x.

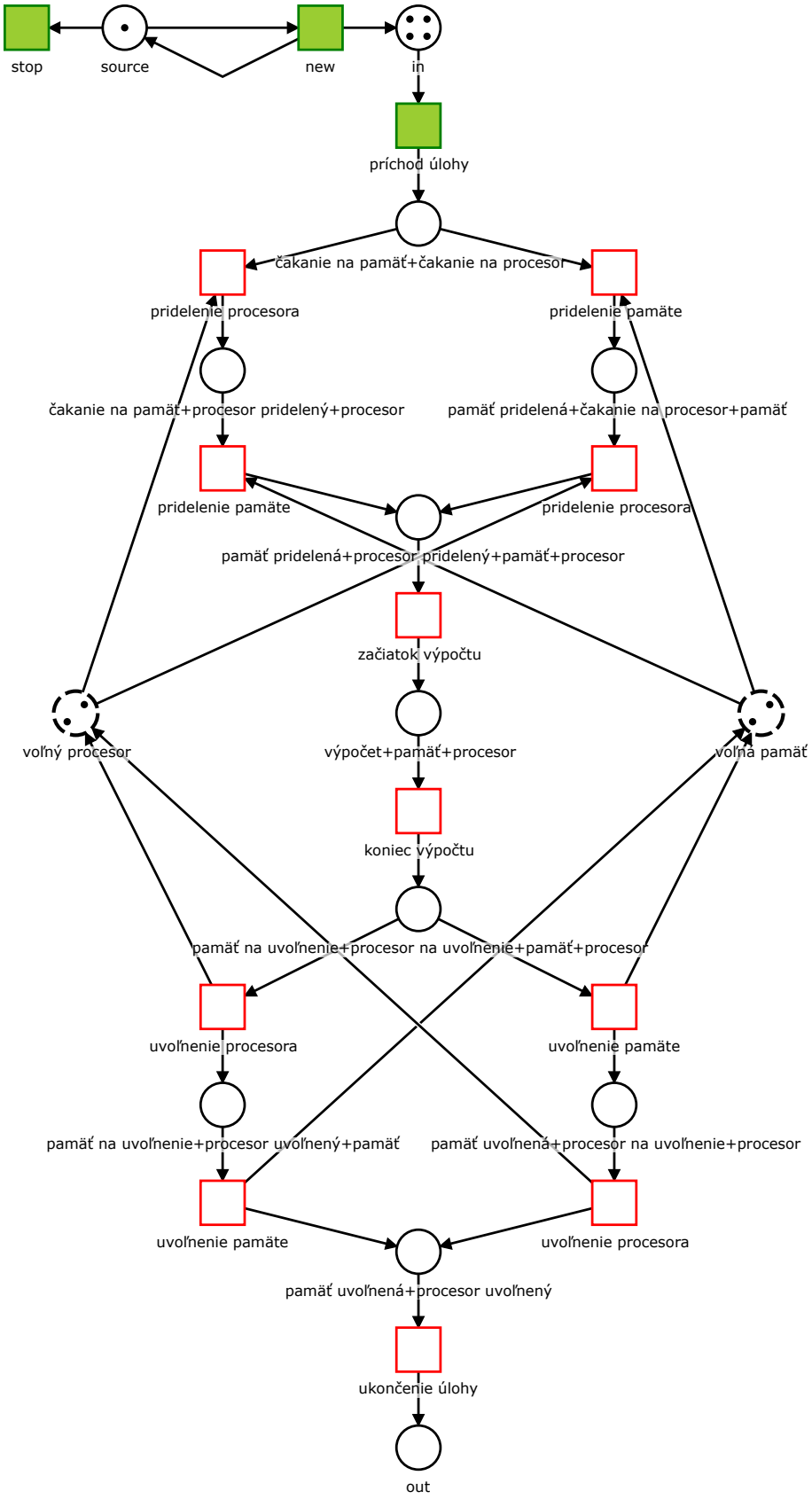
- *Nakoniec, nech* $n(m^\infty) : (P^r \setminus S) \to \mathbb{N}$ *je funkcia udávajúca pre každé miesto siete dosiahnuteľnosti počet inštancií, ktoré sú v danom stave, teda* $n(m^\infty)(x) = |\mathbb{I}(m^\infty, x)|$.

Dôsledok 9 môžeme pomocou pojmu potencionálne dosiahnuteľných značkovaní preformulovať nasledovne:

Dôsledok 10 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW. *Ak značkovanie* m^∞ *je dosiahnuteľné z počiatočného značkovania* m_0^∞, *potom* m^∞ *je potencionálne dosiahnuteľným značkovaním vykonávanje siete* MPN.

Na Obr. 2.11 je znázornená sieť dosiahnuteľnosti workflow siete Obr. 1.11 v značkovaní zodpovedajúcom značkovaniu vykonávanej siete z Obr. 1.14. Značkovanie vykonávanej siete zobrazené na Obr. 1.14 je $m^\infty = in{,}1 + in{,}2 + in{,}3 + in{,}4 + source{,}5 + 2voľná\ pamäť + 2voľný\ procesor$. Dostávame $\mathbb{I}(m^\infty) = 5$, $f(m^\infty, 1) = f(m^\infty, 2) = f(m^\infty, 3) = f(m^\infty, 4) = in$ a $f(m^\infty, 5) = source$. Ďalej dostávame $\mathbb{I}(m^\infty, in) = \{1, 2, 3, 4\}$ a $\mathbb{I}(m^\infty, source) = \{5\}$, pre ostatné nestatické miesta x siete dosiahnuteľnosti je $\mathbb{I}(m^\infty, x)$ prázdna množina. To znamená, že počet prvkov $|\mathbb{I}(m^\infty, in)| = 4$, $|\mathbb{I}(m^\infty, source)| = 1$, pre ostatné nestatické miesta x sú hodnoty $|\mathbb{I}(m^\infty, source)| = 0$. To znamená, že dostávame značkovanie nestatických miest siete dosiahnuteľnosti $n(m^\infty)$ s hodnotami $n(m^\infty)(in) = 4$, $n(m^\infty)(source) = 1$, pričom pre ostatné nestatické miesta x platí $n(m^\infty)(x) = 0$. V zápise pomocou sumy dostávame $n(m^\infty) = 4in + source$. Ak toto značkovanie zjednotíme so značkovaním statických miest z Obr. 1.14, čo je značkovanie $m^\infty|S = 2voľná\ pamäť + 2voľný\ procesor$, dostaneme značkovanie siete dosiahnuteľnosti $4in + source + 2voľná\ pamäť + 2voľný\ procesor$ zobrazené na Obr. 2.11.

Lema 4 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním, nech označkovaná Petriho sieť* $MPN^\infty = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ *je vykonávaná workflow sieť so statickými miestami siete* MW *a nech označkovaná Petriho sieť* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Nech* m_1^∞ *je značkovanie dosiahnuteľné zo značkovania* m_0^∞ *a*



Obr. 2.11: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním v značkovaní zodpovedajúcom značkovaniu vykonávanej siete z Obr. 1.14

nech m_2^∞ je značkovanie dosiahnuteľné zo značkovania m_1^∞ vo vykonávanej sieti. Potom $(n(m_1^\infty) \cup m_1^\infty|S)$ je značkovanie dosiahnuteľné zo značkovania m_0^r a $(n(m_2^\infty) \cup m_2^\infty|S)$ je značkovanie dosiahnuteľné zo značkovania $(n(m_1^\infty) \cup m_1^\infty|S)$ v sieti dosiahnuteľnosti.

Dôkaz. Z definície vykonávanej siete a siete dosiahnuteľnosti dostávame, že $(n(m_0^\infty) \cup m_1^\infty|S) = m_0^r = source \cup m_0|S$. Zároveň platí, že m_0^∞ je jediným značkovaním vykonávanej siete, pre ktoré platí predchádzajúca rovnosť.

Dokážeme, že ak je nejaký prechod vykonávanej siete spustiteľný v nejakom značkovaní m_1^∞ a jeho spustenie vedie do m_2^∞, potom zodpovedajúci prechod je spustiteľný zo značkovania siete dosiahnuteľnosti s rovnakým počtom inštancií v jednotlivých miestach zodpovedajúcich D-značkovaniam, resp. v mieste *source* ako značkovanie m_1^∞ a s rovnakým počtom značiek v statických miestach ako značkovanie m_1^∞. Zároveň ukážeme, že jeho spustenie vedie do značkovania siete dosiahnuteľnosti s rovnakým počtom inštancií v jednotlivých miestach zodpovedajúcich D-značkovaniam, resp. v mieste *source* ako značkovanie m_2^∞ a s rovnakým počtom značiek v statických miestach ako značkovanie m_2^∞.

- Ak pre ľubovoľné $i \in \mathbb{Z}$ je prechod (new, i) spustiteľný v $m_1^\infty \in [m_0^\infty\rangle$, potom $m_1^\infty(source, i) = 1$ a $m_1^\infty(in, i) = 0$. Spustenie (new, i) v m_1^∞ vedie do m_2^∞, pričom $m_2^\infty(source, i) = 0$ a $m_2^\infty(in, i) = 1$. Pre ostatné miesta sa značkovanie nezmení, teda $m_1^\infty|(P^\infty \setminus \{(source, i), (in, i)\}) = m_2^\infty|(P^\infty \setminus \{(source, i), (in, i)\})$. Potom pre $m_1^r = (n(m_1^\infty) \cup m_1^\infty|S)$ platí, že $m_1^r(source) \geq 1$, a teda *new* je spustiteľný v značkovaní m_1^r a jeho spustenie vedie v sieti dosiahnuteľnosti do značkovania m_2^r, pre ktoré $m_2^r(source) = m_1(source) - 1$, $m_2^r(in) = m_1(in) + 1$ a $m_2^r|(P^r \setminus \{source, in\}) = m_2^r|P^r \setminus \{source, in\})$, teda $m_2^r = (n(m_2^\infty) \cup m_2^\infty|S)$.

- Podobne, ak pre ľubovoľné $i \in \mathbb{Z}$ je prechod $(stop, i)$ spustiteľný v $m_1^\infty \in [m_0^\infty\rangle$ potom $m_1^\infty(source, i) = 1$ a $m_1^\infty(in, i) = 0$. Spustenie $(stop, i)$ v m_1^∞ vedie do m_2^∞, pričom $m_2^\infty(source, i) = 0$. Pre ostatné miesta sa značkovanie nezmení, teda $m_1^\infty|(P^\infty \setminus \{(source, i)\}) = m_2^\infty|(P^\infty \setminus \{(source, i)\})$. Potom pre $m_1^r = (n(m_1^\infty) \cup m_1^\infty|S)$ platí, že $m_1^r(source) \geq 1$, a teda *stop* je spustiteľný v značkovaní m_1^r a jeho spustenie vedie v sieti dosiahnuteľnosti do značkovania m_2^r, pre ktoré $m_2^r(source) = m_1(source) - 1$ a $m_2^r|(P^r \setminus \{source\}) = m_2^r|P^r \setminus \{source\})$, teda $m_2^r = (n(m_2^\infty) \cup m_2^\infty|S)$.

- Ak $(t,i) \in T \times \mathbb{Z}$ je spustiteľný v $m_1^\infty \in [m_0^\infty\rangle$ a vedie do m_2^∞, potom $m_1^r = (n(m_1^\infty) \cup m_1^\infty|S)(f(m_1^\infty)(i)) > 0$, a teda prechod $(f(m_1^\infty)(i), t, f(m_1^\infty)(i))$ je spustiteľný v značkovaní m_1^r a jeho spustenie vedie v sieti dosiahnuteľnosti do značkovania m_2^r, pre ktoré
  $m_2^r(f(m_1^\infty)(i)) = m_1(f(m_1^\infty)(i)) - 1$, $m_2^r(f(m_2^\infty)(i)) = m_1(f(m_2^\infty)(i)) + 1$ a
  $m_2^r|(P^r \setminus (\{f(m_1^\infty)(i), f(m_2^\infty)(i)\} \cup S)) = m_2^r|P^r \setminus (\{f(m_1^\infty)(i), f(m_2^\infty)(i)\} \cup S))$ a
  $m_2^r|S = m_2^\infty|S$, a teda $m_2^r = (n(m_2^\infty) \cup m_2^\infty|S)$.

Z definície dosiahnuteľnsti dostávame, že platí tvrdenie lemy 4. □

Analogický vzťah ako v pripade Lemy 4 platí i obrátene.

Lema 5 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním, nech označkovaná Petriho sieť $MPN^\infty = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ je vykonávaná workflow sieť so statickými miestami siete MW a nech označkovaná Petriho sieť $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW. Nech m_1^r je značkovanie dosiahnuteľné zo značkovania m_0^r a nech m_2^r je značkovanie dosiahnuteľné zo značkovania m_1^r v sieti dosiahnuteľnosti. Potom ľubovoľné potencionálne značkovanie m_1^∞ vykonávanej siete, pre ktoré $(n(m_1^\infty) \cup m_1^\infty|S) = m_1^r$, je dosiahnuteľné zo značkovania m_0^∞ a zároveň existuje značkovanie m_2^∞ dosiahnuteľné vo vykonávanej sieti zo značkovania m_1^∞, pričom $(n(m_2^\infty) \cup m_2^\infty|S) = m_2^r$.*

Dôkaz. Ako sme spomenuli v dôkaze Lemy 4 platí, že $(n(m_0^\infty) \cup m_1^\infty|S) = m_0^r = source \cup m_0|S$. Zároveň platí, že m_0^∞ je jediným značkovaním vykonávanej siete, pre ktoré platí predchádzajúca rovnosť.

Opačne ako v dôkaze Lemy 4 ukážeme, že ak je nejaký prechod siete dosiahnuteľnosti spustiteľný v nejakom značkovaní m_1^r a jeho spustenie vedie do m_2^r, potom z ľubovoľného značkovania vykonávanej siete m_1^∞ s rovnakým počtom inštancií v danom stave a s rovnakým počtom značiek v statických miestach ako značkovanie m_1^r je spustiteľný zodpovedajúci prechod. Zároveň ukážeme, že jeho spustenie vedie do značkovania vykonávanej siete m_2^∞ s rovnakým počtom inštancií v rovnakom stave a s rovnakým počtom značiek v statických miestach ako značkovanie m_2^r.

- Z definície vykonávanej siete, siete dosiahnuteľnosti a z Definície 34 dostávame, že ak $v \in \{new, stop\}$ je prechod spustiteľný v sieti dosiahnuteľnosti v značkovaní

$m_1^r \in [m_0^r\rangle$ a jeho spustenie vedie do značkovania m_2^r, potom pre ľubovoľné potenciálne značkovanie m_1^∞ spĺňajúce $n(m_1^\infty) = m_1^r|(P^r \setminus S)$ a $m_1^\infty|S = m_1^r|S$ platí, že $|\mathbb{I}(m_1^\infty, source)| > 0$ a zároveň pre ľubovoľné $i \in \mathbb{I}(m_1^\infty, source)$ je prechod (v, i) spustiteľný, pričom jeho spustenie vedie do m_2^∞ spĺňajúceho $n(m_2^\infty) = m_2^r|(P^r \setminus S)$ a $m_2^\infty|S = m_2^r|S$.

- Analogicky dostávame, že ak $(x, t, y) \in T^a$ je prechod spustiteľný v sieti dosiahnuteľnosti v značkovaní $m_1^r \in [m_0^r\rangle$ a jeho spustenie vedie do značkovania m_2^r, potom pre ľubovoľné potenciálne značkovanie m_1^∞ spĺňajúce $n(m_1^\infty) = m_1^r|(P^r \setminus S)$ a $m_1^\infty|S = m_1^r|S$ platí, že $|\mathbb{I}(m_1^\infty, x)| > 0$ a zároveň pre ľubovoľné $i \in \mathbb{I}(m_1^\infty, x)$ je prechod (t, i) spustiteľný, pričom jeho spustenie vedie do m_2^∞ spĺňajúceho $n(m_2^\infty) = m_2^r|(P^r \setminus S)$ a $m_2^\infty|S = m_2^r|S$.

Z definície dosiahnuteľnosti dostávame, že platí tvrdenie lemy 5. □

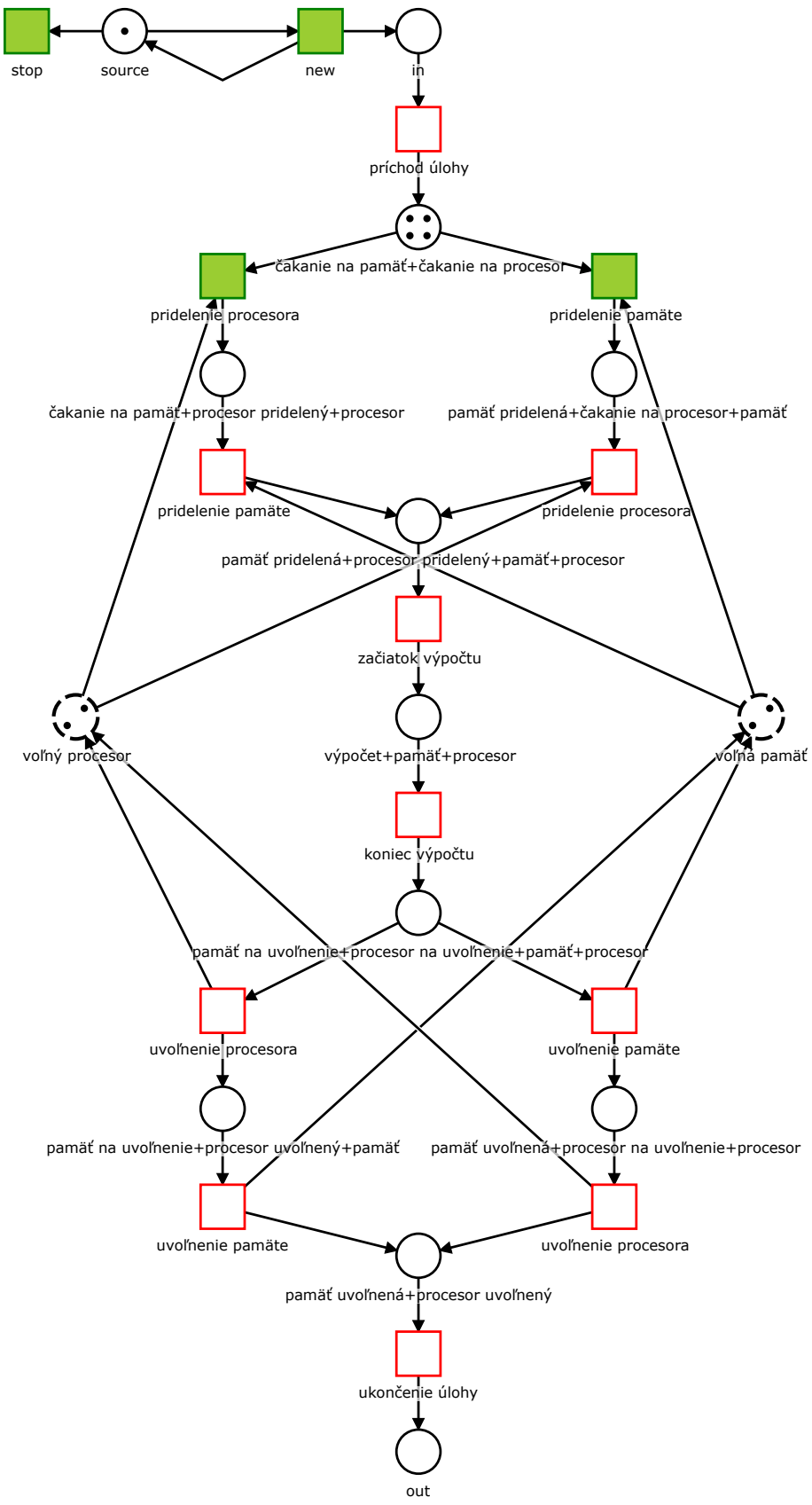
Na Obr. 2.12 je znázornené značkovanie siete dosiahnuteľnosti dosiahnuté z dosiahnuteľného značkovania na Obr. 2.11 štvornásobným spustením prechodu *príchod úlohy*. Toto značkovanie zodpovedá značkovaniu vykonávanej siete z Obr. 1.15, ktoré bolo dosiahnuté spustením prechodov *príchod úlohy,1 príchod úlohy,2 príchod úlohy,3 príchod úlohy,4* v značkovaní vykonávanej siete z Obr. 1.14.

Prvým hlavným výsledkom tejto práce je nasledovná veta, ktorá je priamym dôsledkom Lemy 4 a Lemy 5 a definície uviaznutia vo vykonávanej sieti a v sieti dosiahnuteľnosti.

Veta 1 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním, nech označkovaná Petriho sieť $MPN^\infty = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ je vykonávaná workflow sieť so statickými miestami siete MW a nech označkovaná Petriho sieť $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW. Značkovanie siete dosiahnuteľnosti $m^r \in [m_0^r\rangle$ je uviaznutím siete dosiahnuteľnosti práve vtedy, keď ľubovoľné značkovanie vykonávanej siete $m^\infty \in [m_0^\infty\rangle$, pre ktoré platí $(n(m^\infty) \cup m^\infty|S) = m^r$, je uviaznutím vykonávanej siete.*

Dôkaz.

⇒: Nech m^r je uviaznutie v sieti dosiahnuteľnosti. Nech m^∞ je také potencionálne značkovanie, že $(n(m_1^\infty) \cup m_1^\infty|S) = m^r$. Podľa Lemy 5 je m^∞ dosiahnuteľné z



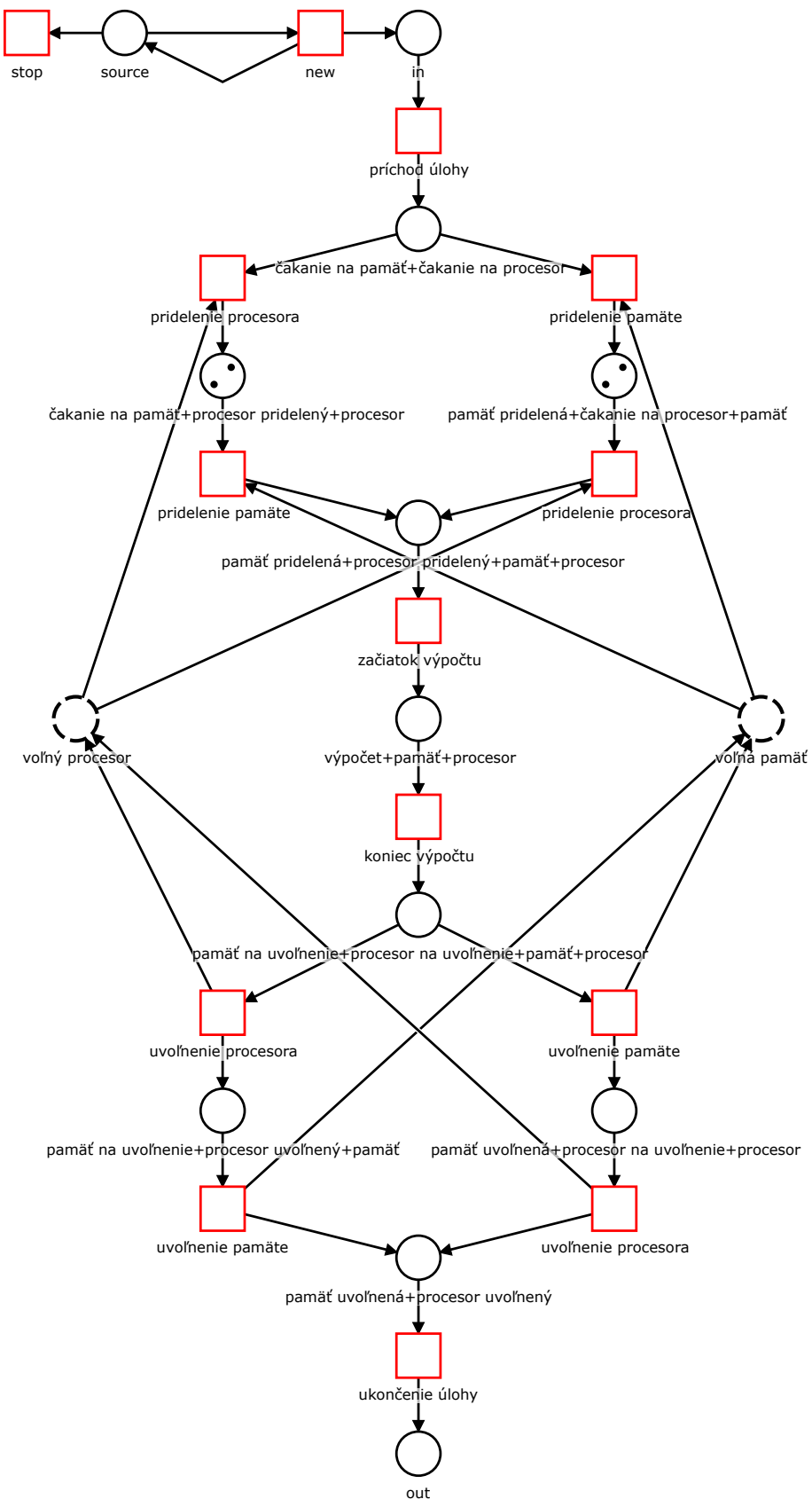
Obr. 2.12: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami v značkovaní zodpovedajúcom značkovaniu vykonávanej siete z Obr. 1.15

m_0^∞. Nech toto značkovanie nie je uviaznutím vo vykonávanej sieti, teda zo značkovania m^∞ je dosiahnuteľné nejaké finálne značkovanie m_f^∞ vykonávanej siete. Z Definície 24 finálneho značkovania vykonávanej siete a z Definície 32 finálneho značkovania v sieti dosiahnuteľnosti, dostávame, že značkovanie $(n(m_f^\infty) \cup m^\infty | S)$ je finálnym značkovaním siete dosiahnuteľnosti. Podľa Lemy 4 je toto značkovanie dosiahnuteľné zo značkovania m^r, čo je v spore s predpokladom, že m^r je uviaznutie siete dosiahnuteľnosti.

$\Leftarrow$: Nech m^∞ je uviaznutie vykonávanej siete. Podľa Lemy 4 je $m^r = (n(m_1^\infty) \cup m_1^\infty | S)$ dosiahnuteľné z m_0^r. Nech toto značkovanie nie je uviaznutím v sieti dosiahnuteľnosti, teda zo značkovania m^r je dosiahnuteľné nejaké finálne značkovanie m_f^r siete dosiahnuteľnosti. Z Definície 24 finálneho značkovania vykonávanej siete a z Definície 32 finálneho značkovania v sieti dosiahnuteľnosti, dostávame, že potencionálne značkovanie m_f^∞, pre ktoré $(n(m_f^\infty) \cup m^\infty | S) = m_f^r$ (existuje práve jedno takéto potencionálne značkovanie), je finálnym značkovaním vykonávanej siete. Podľa Lemy 5 je toto značkovanie dosiahnuteľné zo značkovania m^∞, čo je v spore s predpokladom, že m^∞ je uviaznutie vykonávanej siete.

□

Na Obr. 2.13 je znázornené uviaznutie siete dosiahnuteľnosti z Obr. 2.10. Toto uviaznutie zodpovedá uviaznutiu vykonávanej siete zobrazenému na Obr. 1.16.



Obr. 2.13: Uviaznutie siete dosiahnuteľnosti z Obr. 2.10, ktoré zodpovedá uviaznutiu vykonávanej siete zobrazenému na Obr. 1.16

2.4 Základné uviaznutia siete dosiahnuteľnosti

Definujme pojem zúženie postupnosti, ktorý využijeme v dôkaze ďalšieho výsledku uvedeného v Leme 6.

Definícia 35 (Zúženie postupnosti)
Nech A je množina a $\alpha : \mathbb{I} \to A$ je konečná postupnosť prvkov z A. Nech B je podmnožina množiny A, teda $B \subseteq A$. Nech $\mathbb{I}(\alpha, B) \subseteq I$ je taká konečná množina všetkých kladných celých čísel i z množiny indexov $\mathbb{I}$, pre ktoré platí $\alpha(i) \in B$, teda $\mathbb{I}(\alpha, B)$ je množina indexov takých prvkov, ktoré patria do množiny B. Nech $\mathbb{J}$ je indexová množina taká, že $max_{\mathbb{J}} = |\mathbb{I}(\alpha, B)|$. Nech ďalej $\beta : \mathbb{J} \to \mathbb{I}(\alpha, B)$ je injektívna a rastúca funkcia, teda platí, že ak $\beta(i) = \beta(j)$ potom $i = j$ a zároveň ak $i < j$ potom $\beta(i) < \beta(j)$ pre všetky $i, j \in \mathbb{J}$ (keďže počet prvkov množín $\mathbb{J}$ a $\mathbb{I}(\alpha, B)$ je rovnaký, β je zároveň surjektívna, teda β je rastúca bijekcia, ktorá zoradí prvky $\mathbb{I}(\alpha, B)$ podľa veľkosti od najmenšieho). Nech $\alpha|B : \mathbb{J} \to B$ je taká postupnosť, že $\alpha|B(i) = \alpha(\beta(i))$ pre každé $i \in \mathbb{J}$. Potom postupnosť $\alpha|B$ nazývame zúženie postupnosti α na množinu B.

Keďže zúženie postupnosti je formálne pomerne komplikované, na nasledovnom príklade budeme tento pojem ilustrovať. Uvažujme množinu $A = \{a, b, c, d, e\}$ a postupnosť prvkov $\alpha = abeceda$. Formálne teda máme $\mathbb{I} = \{1, \ldots, 7\}$, $\alpha(1) = \alpha(7) = a$, $\alpha(2) = b$, $\alpha(3) = \alpha(5) = e$, $\alpha(4) = c$ a $\alpha(6) = d$. Uvažujme ďalej množinu $B = \{a, e\}$, teda podmnožinu množiny A tvorenú samohláskami. $\mathbb{I}(\alpha, B)$ je množina indexov od jedna do sedem takých prvkov, ktoré patria do množiny $B = \{a, e\}$, teda množina indexov samohlások (poradie samohlások a a e), teda $\mathbb{I}(\alpha, B) = \{1, 3, 5, 7\}$. Keďže $\mathbb{I}(\alpha, B)$ má 4 prvky, teda $|\mathbb{I}(\alpha, B)| = 4$, množina $\mathbb{J} = \{1, 2, 3, 4\}$. Funkcia β zoradí prvky množiny $\mathbb{I}(\alpha, B) = \{1, 3, 5, 7\}$ podľa poradia, teda $\beta(1) = 1$ $\beta(2) = 3$, $\beta(3) = 5$ a $\beta(4) = 7$. Dosadením dostaneme $\alpha|B(1) = \alpha(\beta(1)) = \alpha(1) = a$, $\alpha|B(2) = \alpha(\beta(2) = \alpha(3) = e$, $\alpha|B(3) = \alpha(\beta(3) = \alpha(5) = e$ a $\alpha|B(4) = \alpha(\beta(4) = \alpha(7) = a$, teda $\alpha|B = aeea$. Zúžením postupnosti $\alpha = abeceda$ na samohlásky $B = \{a, e\}$ je teda postupnosť $\alpha|B = aeea$.

Pre metódu vyšetrenia uviaznutí je kľúčový nasledovný výsledok, ktorý vraví, že ak vieme dostať do nestatických miest v sieti dosiahnuteľnosti nejaký počet značiek, potom vieme dostať do nestatických miest v sieti dosiahnuteľnosti ľubovoľný menší počet značiek.

Lema 6 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW*. Nech značkovanie* m_1^r *je dosiahnuteľné z* m_0^r *v sieti dosiahnuteľnosti. Potom pre ľubovoľné značkovanie* $w : (P^r \setminus S) \to \mathbb{N}$ *spĺňajúce* $m_1^r|(P^r \setminus S) > w$ *existuje také značkovanie* m_2^r *dosiahnuteľné z* m_0^r*, že* $m_2^r|(P^r \setminus S) = w$*.*

Dôkaz. Keďže m_2^r vieme dostať zo značkovania m_1^r opakovaným uberaním jednej značky z nestatických miest, stačí ukázať, že pre ľubovoľné nestatické miesto obsahujúce aspoň jednu značku je dosiahnuteľné značkovanie, ktoré sa od m_1^r v nestatických miestach líši iba tým, že v tomto nestatickom mieste je o jednu značku menej. Ukážeme teda, že ak m_1^r je dosiahnuteľné z m_0^r a $x \in P^r$ je ľubovoľné miesto pre ktoré $m_1^r(x) > 0$, potom je dosiahnuteľné také značkovanie m_2^r, že $m_2^r(x) = m_1^r(x) - 1$ a zároveň $m_2^r(y) = m_1^r(y)$ pre všetky $y \in (P^r \setminus S)$ rôzne od x.

Nech m_1^r je dosiahnuteľné z m_0^r a nech $x \in P^r$ je také, že $m_1^r(x) > 0$. Pomôžeme si vykonávanou sieťou. Nech označkovaná Petriho sieť $MPN^\infty = (P^\infty, T^\infty, I^\infty, O^\infty, m_0^\infty)$ je vykonávaná workflow sieť so statickými miestami siete MW. Podľa Lemy 5 ľubovoľné potencionálne značkovanie m_1^∞ vykonávanej siete, pre ktoré $(n(m_1^\infty) \cup m_1^\infty|S) = m_1^r$, je dosiahnuteľné zo značkovania m_0^∞. Zvoľme teda $m_1^\infty \in [m_0^\infty\rangle$ také, že $(n(m_1^\infty) \cup m_1^\infty|S) = m_1^r$ a zároveň pre $i = max_{\mathbb{I}(m_1^\infty)}$ platí, že $f(m_1^\infty)(i) = x$. Ďalej nech

- α je konečná postupnosť prechodov siete MPN spustiteľná zo značkovania m_0^∞, ktorej spustenie vedie do m_1^∞
- $B = T^\infty \setminus T_i^\infty$ je množina všetkých prechodov okrem prechodov i-tej inštancie
- $\alpha|B$ je zúženie postupnosti α na prechody okrem i-tej inštancie pre ľubovoľné $k \in \mathbb{Z}$

Ak $\alpha|B$ je spustiteľná v m_0^∞ potom jej spustenie vedie do značkovania m_2^∞, pričom platí pre $m_2^r = (n(m_2^\infty) \cup m_2^\infty|S)$, že $m_2(x) = m_1(x) - 1$ a zároveň $m_2(y) = m_1(y)$ pre všetky $y \in (P^r \setminus S)$ rôzne od x. Podľa Lemy 4 je však m_2^r dosiahnuteľné z m_0^r, čím sa ukončí dôkaz.

Ukážeme teda, že $\alpha|B$ je spustiteľná v m_0^∞. Nech $\mathbb{J}$ a β sú definované ako v Definícii 35. Nech $\alpha|B$ nie je spustiteľná v m_0^∞. Potom existuje $j \in \mathbb{J}$ také, že v značkovaní m_3^∞ dosiahnutom po spustení postupnosti prechodov $\alpha|B(1) \ldots \alpha|B(j-1)$ prechod

$\alpha|B(j)$ nie je spustiteľný. Keďže prechody z T_i^∞ (kde $i = max_{\mathbb{I}(m_1^\infty)}$) sme vynechali, $\alpha|B(j) \in T_k^\infty$, pričom $k < i$. Nech $\alpha|B(j) = (t,k)$. Musí existovať $(p,k) \in P_k^\infty$ pre ktoré $m_3^\infty(p,k) < I^\infty((p,k)(t,k))$ alebo $s \in S$ pre ktoré $m_3^\infty(s) < I^\infty(s,(t,k))$. Z Definície 35 dostávame, že prechod $\alpha|B(j) = \alpha(\beta(j))$. Keďže celá postupnosť α je spustiteľná, je spustiteľná aj postupnosť $\alpha(1)\ldots\alpha(\beta(j)-1)$, ktorá vedie do značkovania m_4^∞. V tomto značkovaní je však prechod $\alpha|B(j) = \alpha(\beta(j)) = (t,k)$ spustiteľný. To znamená, že pre každé $(p,k) \in P_k^\infty$ platí $m_4^\infty(p,k) \geq I^\infty((p,k)(t,k))$ a zároveň pre každé $s \in S$ platí $m_4^\infty(s) \geq I^\infty(s,(t,k))$. Keďže postupnosti $\alpha|B(1)\ldots\alpha|B(j-1)$ a $\alpha(1)\ldots\alpha(\beta(j)-1)$ sa líšia iba vynechaním prechodov s indexom i, teda značkovania m_3^∞ a m_4^∞ sa môžu líšiť iba v miestach $P_i^\infty \cup S$. Pre každé $(p,k) \in P_k^\infty$ teda platí $m_3^\infty(p,k) = m_4^\infty(p,k)$. To znamená, že musí existovať $s \in S$ pre ktoré $m_3^\infty(s) < I^\infty(s,(t,k))$. Keďže značkovania kópií určujúcich miest sa môžu líšiť iba pre i-tu inštanciu, pričom v značkovaní m_3^∞ sú i-te kópie určujúcich miest bez značiek, podľa Dôsledku 4 $m_4^\infty(s) \leq m_3(s)$. To je spor so skutočnosťou, že $m_3^\infty(s) < I^\infty(s,(t,k))$ a zároveň $m_4^\infty(s) \geq I^\infty(s,(t,k))$. Postupnosť $\alpha|B$ je teda spustiteľná v m_0^∞. □

V nasledujúcej definícii rozdelíme miesta siete dosiahnuteľnosti podľa toto, či reprezentujú stavy, v ktorých sú použité zdroje zo statickým miest.

Definícia 36 (Miesta so zdrojmi)
Nech $MW = (D,S,T,I,O,m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW.

- *Miesto $x \in P^r \setminus S$ siete dosiahnuteľnosti je nazývané miesto bez zdroja s pre $s \in S$ ak buď $x = source$ alebo pre určujúce miesto $d_s \in D_S$ miesta s platí $x(d_s) = 0$.*
- *Množinu všetkých miest bez zdroja s označujeme B_s^r.*
- *Ak je miesto $x \in P^r \setminus S$ siete dosiahnuteľnosti miestom bez zdroja pre každé $s \in S$, nazývame ho jednoducho miestom bez zdrojov.*
- *Množinu všetkých miest bez zdrojov označujeme B^r.*
- *Miesto $x \in P^r \setminus S$ siete dosiahnuteľnosti je nazývané miesto so zdrojom s pre $s \in S$ ak pre určujúce miesto $d_s \in D_S$ miesta s platí $x(d_s) > 0$. Množinu všetkých miest so zdrojom s označujeme Z_s^r.*

- *Ak pre miesto $x \in P^r \setminus S$ siete dosiahnuteľnosti existuje $s \in S$ také, že x je miestom so zdrojom s, potom ho nazývame jednoducho miestom so zdrojmi.*
- *Množinu všetkých miest so zdrojmi označujeme Z^r.*
- *Pre miesto $x \in Z^r$ siete dosiahnuteľnosti označujeme symbolom $Z(x)$ množinu takých $s \in S$, pre ktoré platí $x \in Z_s^r$.*

Podobne ako miesta, rozdeľme aj prechody siete dosiahnuteľnosti na tie, ktoré potrebujú k spusteniu značky zo statických miest a tie, ktoré ich nepotrebujú. Zároveň rozdeľme miesta podľa toho, či existuje prechod, ktorý presúva značku z miesta siete dosiahnuteľnosti so zdrojmi a pritom vyžaduje ďalšie zdroje.

Definícia 37 (Prechody vyžadujúce zdroje, kritické miesta)
Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW.

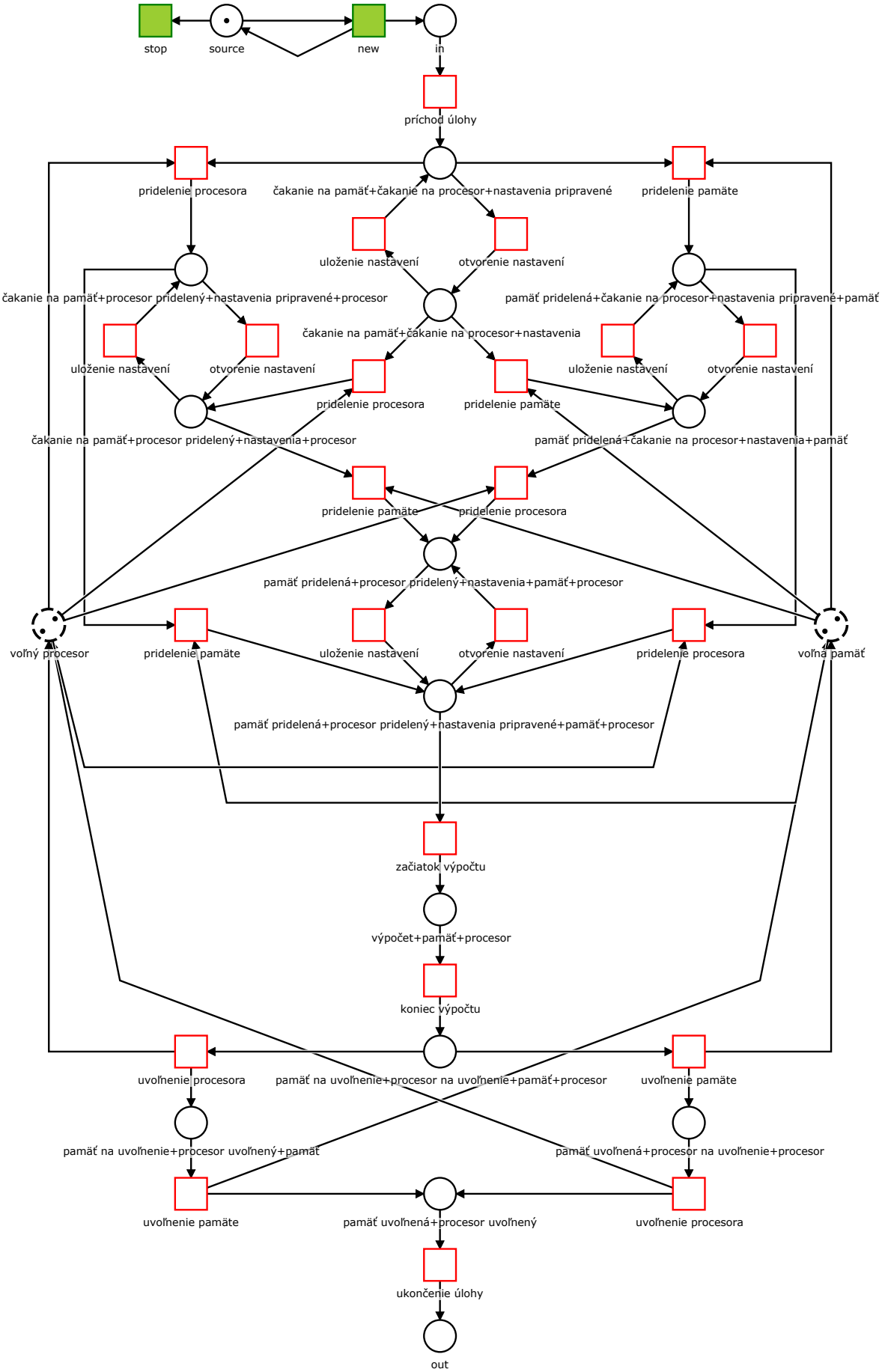
- *Prechod $(x, t, y) \in T^a$ je nazývaný prechod vyžadujúci zdroje ak $I(s, t) > 0$ pre nejaké $s \in S$.*
- *Ak pre miesto so zdrojmi $x \in Z^r$ platí, že existuje prechod $(x, t, y) \in T^a$ vyžadujúci zdroje, potom x nazývame kritické miesto.*
- *Množinu všetkých kritických miest označujeme K^r.*

Na Obr. 2.14 je znázornená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 1.17, na ktorej budeme ilustrovať jednotlive typy miest.

Množina miest bez zdrojov B^r siete dosiahnuteľnosti na Obr. 2.14 je tvorená šiestimi miestami: miestom *source*, miestom *in*, miestom *čakanie na pamäť + čakanie na procesor + nastavenia pripravené*, miestom *čakanie na pamäť + čakanie na procesor + nastavenia*, miestom *pamäť uvoľnená + procesor uvoľnený* a miestom *out*.

Množina miest so zdrojmi Z^r siete dosiahnuteľnosti na Obr. 2.14 je tvorená desiatimi miestami, z ktorých dve tvoria množinu kritických miest Z^r.

Prvým kritickým miestom siete dosiahnuteľnosti je jej miesto *čakanie na pamäť + procesor pridelený + nastavenia pripravené + procesor*, keďže predstavuje inštanciu s



Obr. 2.14: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 1.17

prideleným zdrojom zo statického miesta *voľný procesor*. Tento zdroj je reprezentovaný značkou v určujúcom mieste *procesor* originálnej siete. Z kritického miesta siete dosiahnuteľnosti *čakanie na pamäť + procesor pridelený + nastavenia pripravené + procesor* konzumuje prechod vyžadujúci zdroje (*čakanie na pamäť + procesor pridelený + nastavenia pripravené + procesor, pridelenie pamäte, pamäť pridelená + procesor pridelený + nastavenia pripravené + pamäť + procesor*). Tento prechod vyžaduje zdroj zo statického miesta *voľná pamäť*. Keďže nestatické miesto siete dosiahnuteľnosti *čakanie na pamäť + procesor pridelený + nastavenia pripravené + procesor*, z ktorého tento prechod spotrebuje značku, respektíve nestatické miesto *pamäť pridelená + procesor pridelený + nastavenia pripravené + pamäť + procesor*, v ktorom tento prechod značku vytvorí, sú jednoznačne dané hranami siete dosiahnuteľnosti, na obrázku tento prechod označujeme jednoducho iba *pridelenie pamäte*, teda prostrednou časťou danej trojice označujúcou prechod pôvodnej workflow siete. Upozornime, že prechod označený *pridelenie pamäte*, do ktorého smeruje hrana z miesta bez zdrojov *čakanie na pamäť + čakanie na procesor + nastavenia pripravené* reprezentuje iný prechod siete dosiahnuteľnosti, konkrétne prechod (*čakanie na pamäť + čakanie na procesor + nastavenia pripravené, pridelenie pamäti, pamäť pridelená + čakanie na procesor + nastavenia pripravené + pamäť*), ktorý je taktiež prechodom vyžadujúcim zdroje. Oba prechody reprezentujú spustenie prechodu *pridelenie pamäti* v originálnej workflow sieti, avšak v rôznom značkovaní originálnej workflow siete, preto sú z dôvodu nutnosti separovať tieto značkovania originálnej workflow siete modelované dvoma rôznymi prechodmi siete dosiahnuteľnosti.

Druhým kritickým miestom siete dosiahnuteľnosti je miesto *pamäť pridelená + čakanie na procesor + nastavenia pripravené + pamäť*, keďže reprezentuje inštancie používajúce značku zo statického miesta *voľná pamäť* umiestnenú v určujúcom mieste *procesor* originálnej workflow siete. Zároveň z kritického miesta siete dosiahnuteľnosti *pamäť pridelená + čakanie na procesor + nastavenia pripravené + pamäť* konzumuje prechod siete dosiahnuteľnosti vyžadujúci zdroje daný trojicou (*pamäť pridelená + čakanie na procesor + nastavenia pripravené + pamäť, pridelenie procesora, pamäť pridelená + procesor pridelený + nastavenia pripravené + pamäť + procesor*). Tento prechod siete dosiahnuteľnosti vyžaduje zdroj zo statického miesta *voľný procesor*.

Zvyšené miesta siete dosiahnuteľnosti sú miestami so zdrojmi, avšak nie sú kritic-

kými miestami.

Na Obr. 2.15 je znázornené uviaznutie siete dosiahnuteľnosti, ktoré je zacyklením. Značkovanie na Obr. 2.15 reprezentuje desať inštancií v systéme, pričom počet značiek v jednotlivých miestach siete dosiahnuteľnosti reprezentuje počet inštancií v daných stavoch.

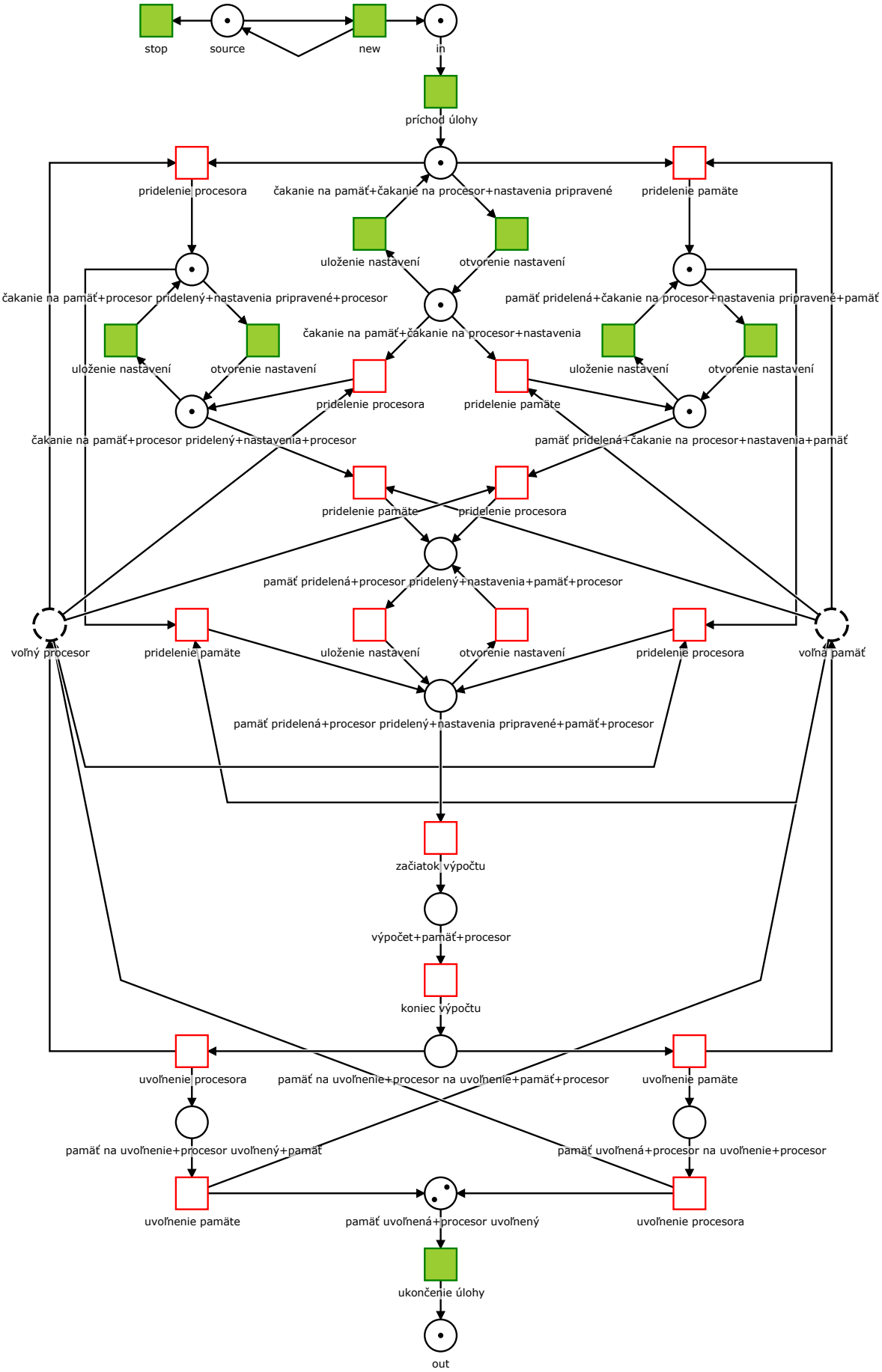
Dôsledok 11 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^r,\ T^r,\ I^r,\ O^r,\ m_0^r)$ *je sieť dosiahnuteľnosti siete* MW*. Nech* $s \in S$ *je ľubovoľné statické miesto a nech* $d_s \in D_S$ *je jeho určujúce miesto. Potom pre každé značkovanie* m^r *dosiahnuteľné z* m_0^r *platí, že* $m^r(s) + \sum_{x \in Z_s^r} m^r(x) \cdot x(d_s) = m_0(s)$*, a teda* $m^r(s) \leq m_0^r(s) = m_0(s)$*.*

Dôležitým výsledkom pre detekciu uviaznutí je nasledovná lema, ktorá hovorí, že z každého dosiahnuteľného značkovania siete dosiahnuteľnosti môžeme dosiahnuť značkovanie, v ktorom sú spomedzi všetkých miest so zdrojmi označkované iba kritické miesta.

Lema 7 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r,\ I^r,\ O^r,\ m_0^r)$ *je sieť dosiahnuteľnosti siete* MW*. Potom pre ľubovoľné značkovanie* m_1^r *dosiahnuteľné z* m_0^r *v sieti dosiahnuteľnosti existuje značkovanie* m_2^r *dosiahnuteľné z* m_1^r*, pričom platí* $m_2^r(x) = 0$ *pre každé* $x \in Z^r \setminus K^r$*.*

Dôkaz. Ukážeme, že ak je značka v mieste $x \in Z^r \setminus K^r$, môžeme ju presunúť buď do miesta bez zdrojov z B^r, alebo do kritického miesta z K^r. Opakovaním procedúry môžeme vynulovať miesta $x \in Z^r \setminus K^r$. Keďže pôvodná určená workflow sieť má korektné správanie, z každého $x \in Z^r \subseteq [m_0\rangle|D$ je v originálnej workflow sieti dosiahnuteľné finálne značkovanie $out \cup m_0|S$. To znamená, že pre každé miesto $x \in Z^r \setminus K^r$ existuje konečná postupnosť $\epsilon : \mathbb{I} \to T$ a $\sigma : \mathbb{I} \cup \{0\} \to [m_0\rangle|D$ taká, že $\sigma(0) = x$, $(\sigma(i-1), \epsilon(i), \sigma(i)) \in T^a$ pre každé kladné celé číslo $i \in \mathbb{I}$ a zároveň $\sigma(max_{\mathbb{I}}) = out$. Nech $\alpha : \mathbb{I} \to T^a$ je teda postupnosť prechodov z T^a taká, že $\alpha(i) = (\sigma(i-1), \epsilon(i), \sigma(i))$ pre $i \in \mathbb{I}$.

- Nech žiaden z prechodov $\alpha(i)$, kde $i \in \mathbb{I}$, je prechodom vyžadujúcim zdroje. Potom postupnosť α je spustiteľná v každom značkovaní siete dosiahnuteľnosti



Obr. 2.15: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami z Obr. 1.17 v uviaznutí pre desať inštancií

$m_1^r \in [m_0^r\rangle$ spĺňajúcom $m_1(x) > 0$ a jej spustenie vedie do značkovania m_2^r, pričom $m_2^r(x) = m_1^r(x) - 1$ a $m_2^r(out) = m_1^r(out) + 1$, teda spustenie postupnosti α presunie značku z miesta so zdrojmi x siete dosiahnuteľnosti do miesta bez zdrojov *out* siete dosiahnuteľnosti.

- Nech existuje $i \in \mathbb{I}$ také, že prechod $\alpha(i)$ je prechod vyžadujúci zdroje. Nech i je najmenší index, pre ktorý $\alpha(i)$ je prechod vyžadujúci zdroje. Keďže x nie je kritické miesto, $i \geq 2$.

  - Nech $\sigma(i-1) \in B^r$, teda $\sigma(i-1)$ je miesto bez zdrojov. Potom postupnosť $\alpha|\{1, \ldots, i-1\}$ je spustiteľná v každom značkovaní siete dosiahnuteľnosti $m_1 \in [m_0^r\rangle$ spĺňajúcom $m_1^r(x) > 0$ a jej spustenie vedie do značkovania m_2^r, pričom $m_2^r(x) = m_1^r(x) - 1$ a $m_2^r(\sigma(i-1)) = m_1^r(\sigma(i-1)) + 1$, teda spustenie postupnosti α presunie značku z miesta x siete dosiahnuteľnosti do miesta bez zdrojov $\sigma(i-1)$ siete dosiahnuteľnosti.
  - Nech $\sigma(i-1) \in Z^r$, teda $\sigma(i-1)$ je miesto so zdrojmi. Keďže $\alpha(i) = (\sigma(i-1), \epsilon(i), \sigma(i))$ je prechod vyžadujúci zdroje, $\sigma(i-1) \in K^r$, teda $\sigma(i-1)$ je kritické miesto. Potom postupnosť $\alpha|\{1, \ldots, i-1\}$ je spustiteľná v každom značkovaní siete dosiahnuteľnosti $m_1^r \in [m_0^r\rangle$ spĺňajúcom $m_1(x) > 0$ a jej spustenie vedie do značkovania m_2^r, pričom $m_2^r(x) = m_1^r(x) - 1$ a $m_2^r(\sigma(i-1)) = m_1^r(\sigma(i-1)) + 1$, teda spustenie postupnosti α presunie značku z miesta x siete dosiahnuteľnosti do kritického miesta $\sigma(i-1)$ siete dosiahnuteľnosti.

□

Ak výsledok z Lemy 7 aplikujeme na uviaznutia, dostaneme, že z každého uviaznutia siete dosiahnuteľnosti je dosiahnuteľné uviaznutie, v ktorom sú spomedzi všetkých miest so zdrojmi označkované iba kritické miesta. Takéto uviaznutia budeme nazývať kritické uviaznutia.

Definícia 38 (Kritické uviaznutie siete dosiahnuteľnosti)
Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním a $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW. Potom uviaznutie m^r dosiahnuteľné z m_0^r v sieti dosiahnuteľnosti,

pre ktoré platí $m^r(x) = 0$ *pre každé* $x \in Z^r \setminus K^r$, *nazývame kritické uviaznutie siete dosiahnuteľnosti.*

Dôsledok 12 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete MW. Potom platí, že ak ľubovoľné značkovanie* m_1^r *dosiahnuteľné z* m_0^r *v sieti dosiahnuteľnosti je uviaznutím, potom existuje kritické uviaznutie* m_2^r *dosiahnuteľné z* m_1^r.

Uviaznutie na Obr. 2.15 nie je kritickým uviaznutím, pretože miesto so zdrojmi *čakanie na pamäť + procesor pridelený + nastavenia + procesor* a miesto so zdrojmi *pamäť pridelená + čakanie na procesor + nastavenia + pamäť* nie sú prázdne. Tieto miesta však nie sú kritickými miestami siete dosiahnuteľnosti.

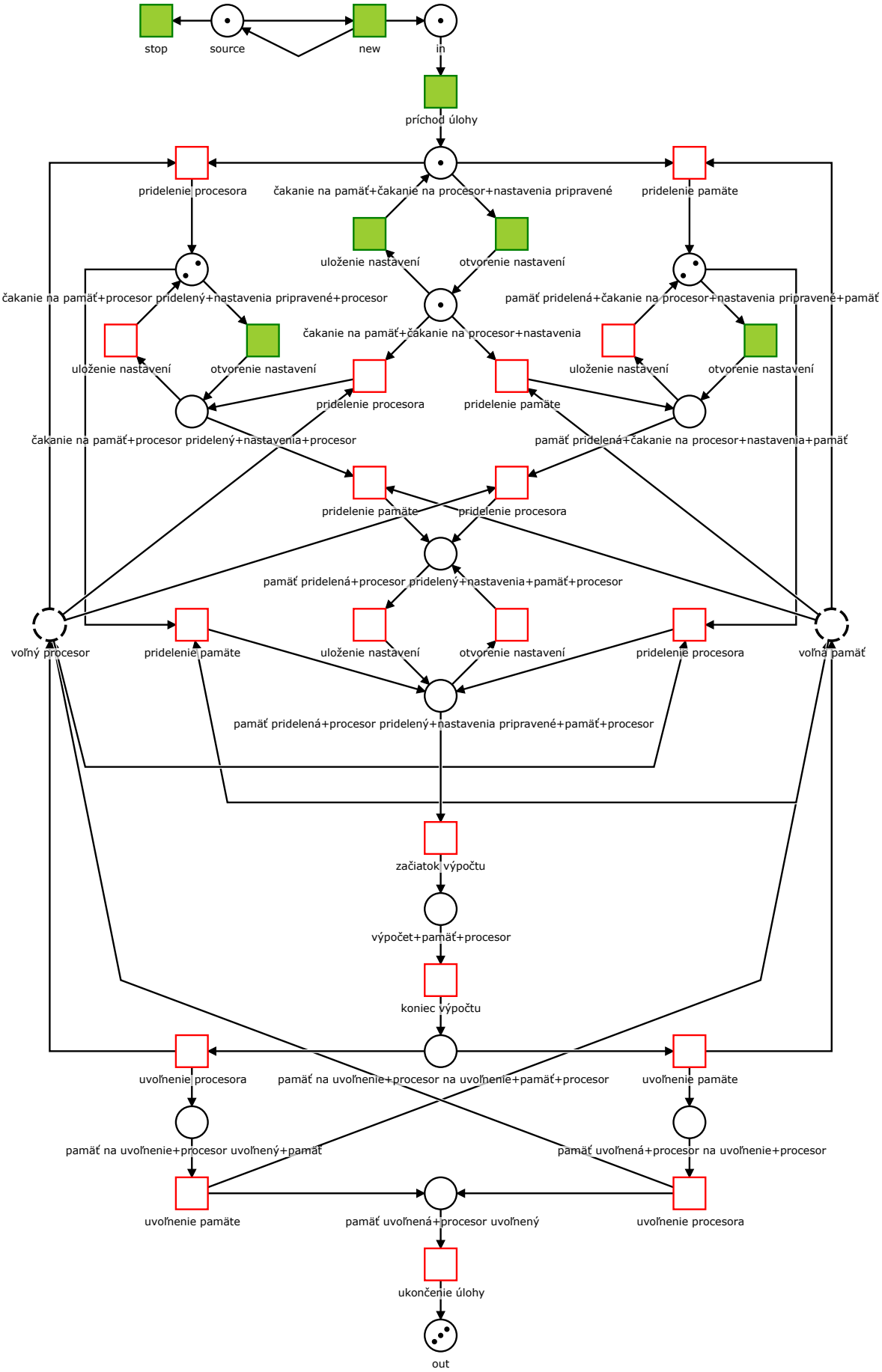
Zo značkovania na Obr. 2.15 je však dosiahnuteľné značkovanie zobrazené na Obr. 2.16, v ktorom sú spomedzi miest so zdrojmi označkované iba kritické miesta. Uviaznutie na Obr. 2.16 je teda kritickým uviaznutím siete dosiahnuteľnosti.

Špeciálnu množinu uviaznutí tvoria také kritické uviaznutia, pre ktoré všetky miesta okrem kritických miest a statických miest sú neoznačené. Takéto kritické uviaznutia nazveme základné uviaznutia.

Definícia 39 (Základné uviaznutie siete dosiahnuteľnosti)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete MW. Potom kritické uviaznutie* m^r *dosiahnuteľné z* m_0^r *v sieti dosiahnuteľnosti, pre ktoré platí* $m^r(x) = 0$ *pre každé miesto bez zdrojov* $x \in B^r$, *nazývame základné uviaznutie siete dosiahnuteľnosti.*

Nasledujúci výsledok hovorí, že ak m_1^r je kritické uviaznutie, potom základné uviaznutie m_2^r, ktoré vznikne z m_1^r odstránením všetkých značiek zo všetkých miest bez zdrojov, je tiež dosiahnuteľné z m_0^r v sieti dosiahnuteľnosti.

Lema 8 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete MW. Potom platí, že ak ľubovoľné značkovanie* m_1^r *dosiahnuteľné z* m_0^r *v sieti dosiahnuteľnosti je kritickým uviaznutím, potom* m_2^r, *ktoré spĺňa*



Obr. 2.16: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami z Obr. 1.17 v kritickom uviaznutí pre desať inštancií

$m_2^r(x) = m_1^r(x)$ pre každé miesto so zdrojmi $x \in Z^r$ a $m_2^r(x) = 0$ pre každé miesto bez zdrojov $x \in B^r$, je základné uviaznutie dosiahnuteľné z m_0^r.

Dôkaz. Podľa Lemy 6, m_2^r je dosiahnuteľné z m_0^r. Predpokladajme, že m_2^r nie je základným uviaznutím. Keďže, m_1^r je kritické uviaznutie, $m_2^r(x) = 0$ pre každé $x \in Z^r \setminus K^r$, teda m_2^r spĺňa podmienku kritickosti. Keďže $m_2^r(x) = 0$ pre každé miesto bez zdrojov $x \in B^r$, m_2^r spĺňa aj podmienku základnosti. To znamená, že m_2^r nie je uviaznutím. Potom existuje postupnosť prechodov β, ktorej spustenie vedie do nejakého finálneho značkovania m_f^r, pričom $m_f^r|S = m_0^r|S$. Keďže $m_2^r(x) = m_1^r(x)$ pre každé miesto so zdrojmi $x \in Z^r$, teda spomedzi nestatických miest siete dosiahnuteľnosti sa m_2^r a m_1^r líšia iba v značkovaní miest bez zdrojov, platí aj $m_2^r|S = m_1^r|S$, teda značkovanie statických miest m_2^r a m_1^r je rovnaké. Potom postupnosť β je spustiteľná aj v značkovaní m_1^r a jej spustenie vedie do stavu m_3, pre ktorý platí $m_3^r|Z^r = 0$ a zároveň $m_3^r|S = m_0^r|S$.

Keďže pôvodná sieť MW je korektná, potom (podobne ako bolo ukázané v dôkaze Lemy 7) z každého $x \in B^r \subseteq [m_0\rangle|D$ je v originálnej určenej workflow sieti dosiahnuteľné finálne značkovanie $out \cup m_0|S$. To znamená, že pre každé miesto $x \in B^r$ existuje konečná postupnosť $\epsilon : \mathbb{I} \to T$ a $\sigma : \mathbb{I} \cup \{0\} \to [m_0\rangle|D$ taká, že $\sigma(0) = x$, $(\sigma(i-1), \epsilon(i), \sigma(i)) \in T^a$ pre každé kladné celé číslo $i \in \mathbb{I}$ a zároveň $\sigma(max_{\mathbb{I}}) = out$. Nech $\alpha : \mathbb{I} \to T^a$ je teda postupnosť prechodov z T^a taká, že $\alpha(i) = (\sigma(i-1), \epsilon(i), \sigma(i))$ pre $i \in \mathbb{I}$. Postupnosť α je spustiteľná v každom značkovaní siete dosiahnuteľnosti $m_1^r \in [m_0^r\rangle$ spĺňajúcom $m_1(x) > 0$ a zároveň $m_1^r|S = m_0^r|S$ a jej spustenie vedie do značkovania m_4^r, pričom $m_4^r(x) = m_1^r(x) - 1$ a $m_4^r(out) = m_1^r(out) + 1$, teda spustenie postupnosti α presunie značku z miesta bez zdrojov x siete dosiahnuteľnosti do miesta bez zdrojov out siete dosiahnuteľnosti, a zároveň $m_4^r|Z^r = 0$, $m_4^r|B^r \setminus \{x, out\} = m_1^r|B^r \setminus \{x, out\}$ a $m_4^r|S = m_1^r|S = m_0^r|S$. Opakovaným spúšťaním postupnosti α dostaneme značkovanie m_5^r, pričom $m_5^r(x) = 0$ a $m_5^r(out) = m_1^r(out) + m_1^r(x)$, teda $m_1^r(x)$-krát opakované spustenie postupnosti α presunie všetkých $m_1^r(x)$ značiek z miesta bez zdrojov x siete dosiahnuteľnosti do miesta bez zdrojov out siete dosiahnuteľnosti, a zároveň $m_5^r|Z^r = 0$, $m_5^r|B^r \setminus \{x, out\} = m_1^r|B^r \setminus \{x, out\}$ a $m_5^r|S = m_1^r|S = m_0^r|S$. Opakovaným spúšťaním procedúry pre také $x \in B^r$, ktorých značkovanie je nenulové, dosiahneme značkovanie m_f^r také, že $m_f^r|((Br \setminus \{out\}) \cup Z^r) = 0$ a $m_f^r|S = m_0^r|S$, teda finálne značkovanie siete dosiahnuteľnosti. To je však spor s predpokladom, že m_1 je kritické uviaznutie. □

V kritickom uviaznutí zobrazenom na Obr. 2.16 sú okrem kritických miest označené

značkami aj miesta bez zdrojov. Aplikovaním Lemy 8 na kritické uviaznutie zobrazené na Obr. 2.16 dostaneme dosiahnuteľné základné uviaznutie na Obr. 2.17.

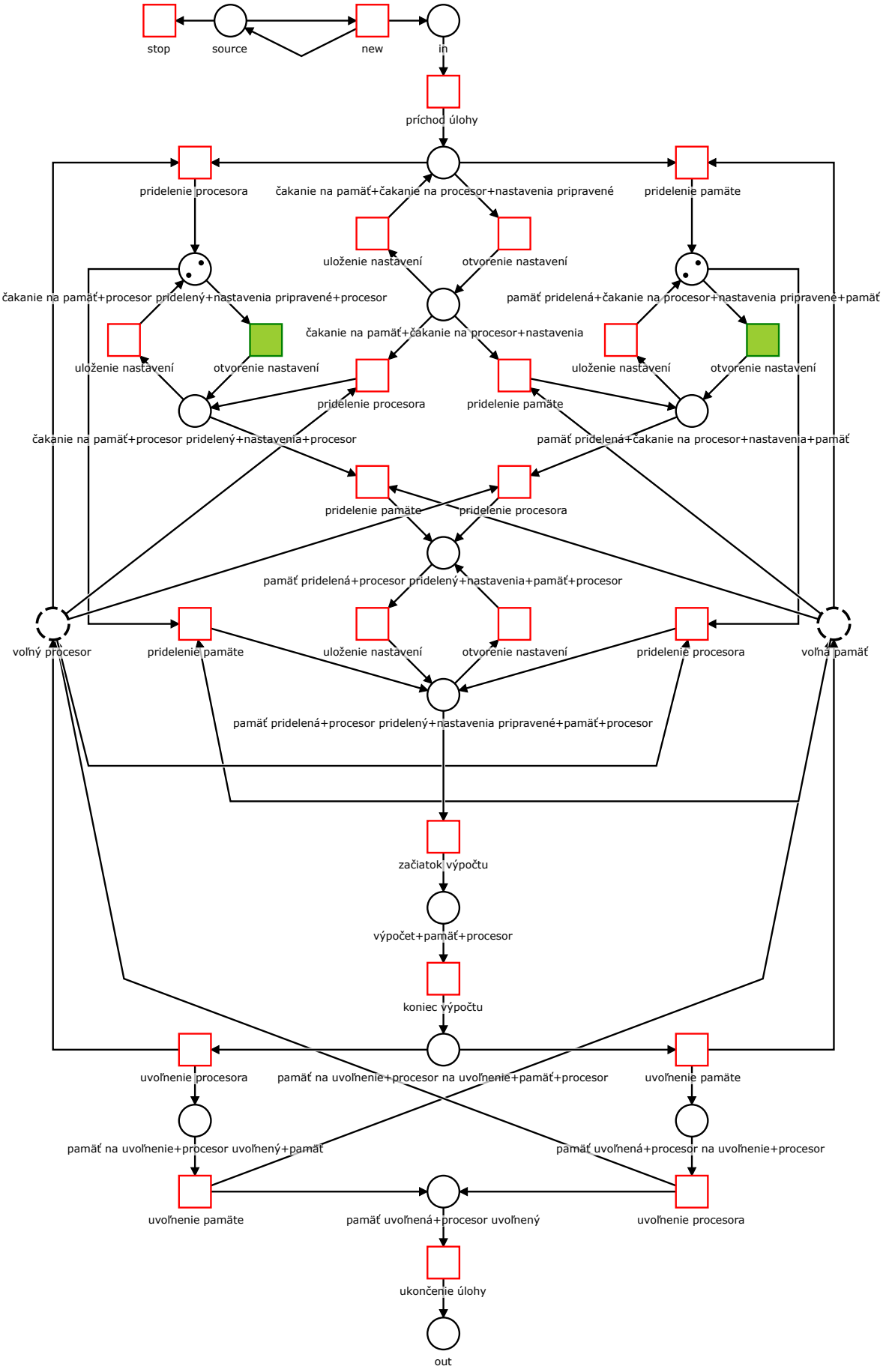
Spojením predchádzajúcich výsledkov, teda tvrdenia z Dôsledku 12 a Lemy 8 dostávame druhý hlavný výsledok práce.

Veta 2 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW*. Potom platí, že ak v sieti dosiahnuteľnosti existuje uviaznutie* m_1^r *dosiahnuteľné z* m_0^r*, potom v sieti dosiahnuteľnosti existuje základné uviaznutie* m_2^r *dosiahnuteľné z* m_0^r*.*

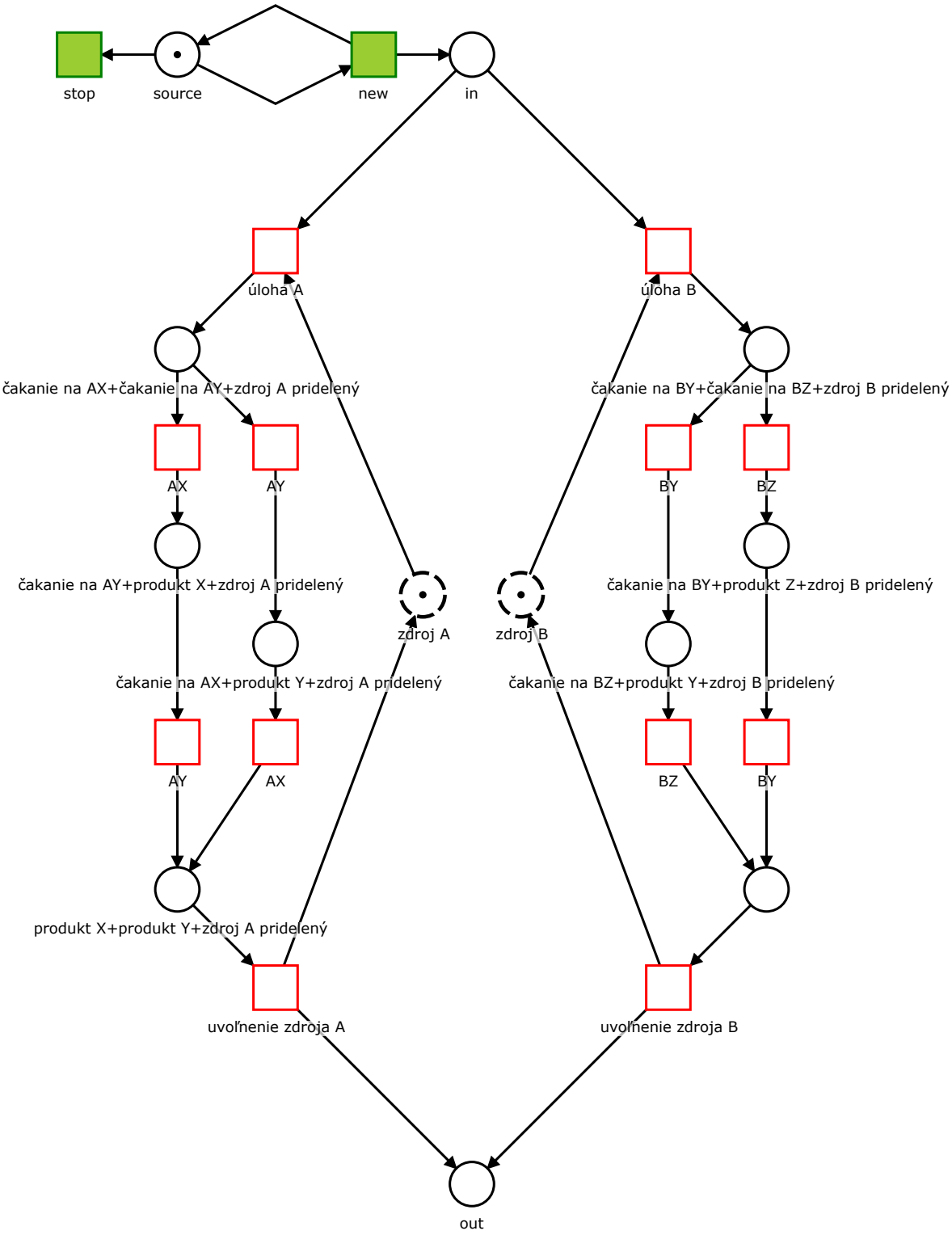
Jednoduchým dôsledkom Vety 2 je fakt, že ak sieť dosiahnuteľnosti nemá kritcké miesta, nemá ani uviaznutia.

Dôsledok 13 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW*. Ak sieť dosiahnuteľnosti nemá žiadne kritické miesto, potom nemá ani žiadne uviaznutia.*

Na Obr. 2.18 je znázornená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 2.3. Hoci v sieti dosiahnuteľnosti na Obr. 2.18 sú všetky miesta okrem miest *source, in, out* miestami so zdrojmi, sieť dosiahnuteľnosti nemá žiadne kritické miesta, a teda nemá ani žiadne uviaznutia.



Obr. 2.17: Sieť dosiahnuteľnosti workflow siete so statickými miestami z Obr. 1.17 v základnom uviaznutí pre štyri inštancie



Obr. 2.18: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 2.3

2.5 Obmedzené siete dosiahnuteľnosti

Výsledok vety 8 a Dôsledku 13 hovorí, že ak sieť nemá kritické miesta, nemá ani uviaznutia a ak má kritické miesta, potom na detekciu uviaznutí stačí zistiť, či sieť dosiahnuteľnosti má základné uviaznutia. Kritické miesta sú však v sieti dosiahnuteľnosti ohraničené. Značkovania, z ktorých sú dosiahnuteľné základné uviaznutia, sú taktiež ohraničené. Na vyšetrenie základných uviaznutí stačí obmedziť sieť dosiahnuteľnosti tak, aby boli dosiahnuteľné iba ohraničené značkovania, z ktorých sú potencionálne dosiahnuteľné základné uviaznutia. Pre takto obmedzenú sieť dosiahnuteľnosti, ktorá je ohraničená, vyšetríme graf dosiahnuteľnosti a zistíme, či v ňom existujú uviaznutia.

Definícia 40 (Jednoduché ohraničenie kritických miest)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Nech* $k \in K^r$ *je kritické miesto siete dosiahnuteľnosti. Nech* $s \in Z(k)$ *a nech* $d_s \in D_S$ *je určujúce miesto* s. *Označme* $pb(k, s) = m_0(s) \div k(d_s)$, *kde symbol* $\div$ *označuje celočíselné delenie. Následne označme* $sbound(k) = \min_{s \in Z(k)} pb(k, s)$ *minimum hodnôt* $pb(k, s)$ *pre* $s \in Z(k)$. *Hodnotu* $sbound(k)$ *nazývame jednoduché ohraničenie kritického miesta* k. *Súčet hodnôt*

$$sbound(K^r) = \sum_{k \in K^r} sbound(k)$$

nazývame jednoduché ohraničenie kritických miest K^r.

Dôsledok 14 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Nech* m^r *je ľubovoľné značkovanie siete dosiahnuteľnosti dosiahnuteľné z* m_0^r. *Pre každé kritické miesto* $k \in K^r$ *platí* $m^r(k) \leq sbound(k)$, *teda počet značiek v žiadnom dosiahnuteľnom značkovaní* m^r *siete dosiahnuteľnosti neprekročí jednoduché ohraničenie kritického miesta. Zároveň*

$$\sum_{k \in K^r} m^r(k) \leq sbound(K^r)$$

teda súčet značiek v kritických miestach v žiadnom dosiahnuteľnom značkovaní neprekročí jednoduché ohraničenie kritických miest K^r. *Taktiež platí, že* $sbound(K^r) \geq 1$.

Keďže značky v statických miestach môžu byť využívané vo viacerých miestach siete dosiahnuteľnosti a miesta dosiahnuteľnosti môžu využívať značky rôznych statických miest, presnejšie horné ohraničenie počtu značiek v kritických miestach môže byť vypočítane ako riešenie nasledovnej úlohy celočíselného lineárneho programovania.

Definícia 41 (Ohraničenie kritických miest)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Nech* $y : K^r \to \mathbb{N}$ *je celočíselným riešením sústavy nerovníc*

$$\sum_{k \in K^r} k(d_s) \cdot y(k) \leq m_0(s) \text{ pre } s \in S$$

$$y(k) \geq 0 \text{ pre } k \in K^r$$

ktoré maximalizuje účelovú funkciu

$$\sum_{k \in K^r} k(d_s) \cdot y(k)$$

Potom túto maximálnu hodnotu

$$bound(K^r) = \sum_{k \in K^r} k(d_s) \cdot y(k)$$

nazývame ohraničenie kritických miest K^r.

Z formulácie úlohy celočíselného lineárneho programovania je zrejmý nasledovný výsledok.

Dôsledok 15 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Nech* m^r *je ľubovoľné značkovanie siete dosiahnuteľnosti dosiahnuteľné z* m_0^r. *Potom*

$$\sum_{k \in K^r} m^r(k) \leq bound(K^r)$$

teda súčet značiek v kritických miestach v žiadnom dosiahnuteľnom značkovaní neprekročí ohraničenie kritických miest K^r. *Zároveň platí, že* $sbound(K^r) \geq bound(K^r)$.

Metódy pre riešenie príslušných oplimalizačných úloh celočíselného lineárneho programovania je možné nájsť napríklad v monografii [72].

Pre označkovanú určenú workflow sieť so statickými miestami a korektným správaním na Obr. 1.17 a jej sieť dosiahnuteľnosti na Obr. 2.15 dostávame

$$sbound(K^r) = bound(K^r) = 4$$

.

Zároveň z definície siete dosiahnuteľnosti priamo vyplýva, že zo značkovania, v ktorom súčet značiek v miestach siete dosiahnuteľnosti patriacich do $[m_0\rangle|D$ je rovný n, nie je dosiahnuteľné žiadne značkovanie, v ktorom súčet značiek v miestach siete dosiahnuteľnosti patriacich do $[m_0\rangle$ je menší ako n. Ak je miesto *source* prázdne, súčet značiek v miestach siete dosiahnuteľnosti patriacich do $[m_0\rangle|D$ zostáva rovný n.

Lema 9 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním a $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW. Nech m_1^r je ľubovoľné značkovanie siete dosiahnuteľnosti dosiahnuteľné z m_0^r. Potom pre každé značkovanie m_2^r dosiahnuteľné z m_1^r platí:*

$$\sum_{x\in[m_0\rangle|D} m_1^r(x) \leq \sum_{x\in[m_0\rangle|D} m_2^r(x)$$

Ak $m_1^r(source) = 0$ potom platí:

$$\sum_{x\in[m_0\rangle|D} m_1^r(x) = \sum_{x\in[m_0\rangle|D} m_2^r(x)$$

Dôkaz. Spustenie prechodu *stop* nespotrebuje žiadnu značku ani nevytvorí žiadnu značku v miestach patriacich do $[m_0\rangle|D$, prechod *new* nespotrebuje žiadnu značku z miest patriacich do $[m_0\rangle|D$ a vytvára novú značku v mieste *in*, čiže zvyšuje počet značiek v miestach patriacich do $[m_0\rangle|D$, prechody patriace do T^a spotrebujú práve jednu značku z nejakého miesta patriaceho do $[m_0\rangle|D$ a vytvoria práve jednu značku v nejakom mieste patriacom do $[m_0\rangle|D$, teda celkovo nezmenia súčet počtu značiek v miestach patriacich do $[m_0\rangle|D$. Ak *source* neobsahuje žiadnu značku, prechod *new* nie je spustiteľný. □

Ak teda obmedzíme súčet počtu značiek v miestach siete dosiahnuteľnosti patriacich do $[m_0\rangle|D$ na hodnotu rovnú jednoduchému ohraničeniu kritických miest $sbound(K^r)$,

respektíve na hodnotu rovnú ohraničeniu kritických miest $bound(K^r)$, a zároveň zachováme správanie pre značkovania obmedzenej siete dosiahnuteľnosti pre všetky značkovania s počtom neprevyšujúcom $sbound(K^r)$, respektíve $bound(K^r)$, budeme vedieť v konečnom grafe dosiahnuteľnosti takto obmedzenej ohraničenej sieti dosiahnuteľnosti nájsť všetky základné uviaznutia pôvodnej siete dosiahnuteľnosti.

Takúto obmedzenú sieť dosiahnuteľnosti definujeme pre dané kladné celé číslo $n \in \mathbb{Z}$ jednoducho pridaním miesta, z ktorého spustenie prechodu *new* spotrebuje práve jednu značku, pričom v počiatočnom značkovaní do tohto miesta umiestíme n značiek.

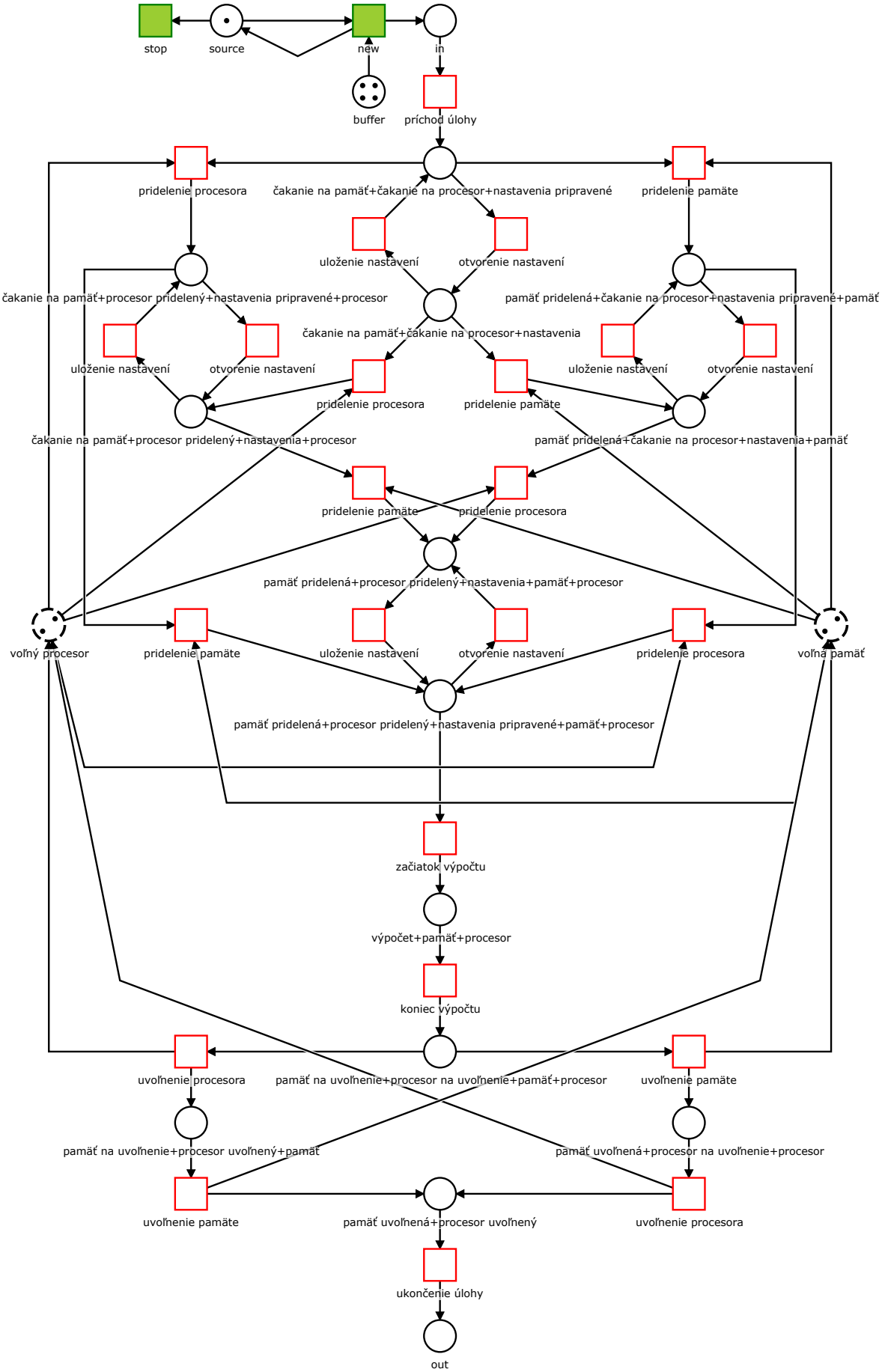
Definícia 42 (n-obmedzená sieť dosiahnuteľnosti)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Nech* $buffer \notin (P^r \cup T^r)$. *Potom* n*-obmedzená sieť dosiahnuteľnosti siete* MW *je označkovaná Petriho sieť* $MPN^n = (P^n = P^r \cup \{buffer\}, T^n = T^r, I^n, O^n = O^r, m^n = m_0^r \cup n{\cdot}buffer)$ *(kde* $n{\cdot}buffer$ *predstavuje v zmysle zavedenej notácie funkciu z jednoprvkovej množiny* $\{buffer\}$ *do* $\mathbb{Z}$ *priraďujúcu v počiatočnom značkovaní do miesta buffer* n *značiek), pričom* $I^n|(P^r \times T^r) = I^r$, $I^n(buffer, new) = 1$ *a* $I^n|(\{buffer\} \times (T^r \setminus \{new\})) = 0$.

Na Obr. 2.19 je znázornená n-obmedzená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 1.17 pre $n = sbound(K^r) = bound(K^r) = 4$, ktorá vznikla zo siete dosiahnuteľnosti na Obr. 2.14 jednoduchým pridaním miesta *buffer*, pridaním hrany z miesta *buffer* do prechodu *new* a pridaním 4 značiek do miesta *buffer*.

Je zrejmé, že n-obmedzená sieť dosiahnuteľnosti je ohraničená.

Dôsledok 16 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je* n*-obmedzená sieť dosiahnuteľnosti siete* MW. *Potom sieť* MPN^n *je ohraničená a značkovanie* $b^n : T^n \to \mathbb{N}$ *spĺňajúce* $b^n(source) = 1$ *a* $b^n(x) = n$ *pre všetky* $x \in P^n$ *rôzne od source je ohraničenie* n*-obmedzenej siete dosiahnuteľnosti* MPN^n.

Tvrdenie Lemy 9 platí aj pre n-obmedzenú sieť dosiahnuteľnosti.



Obr. 2.19: n-obmedzená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 1.17 pre $n = sbound(K^r) = bound(K^r) = 4$

Dôsledok 17 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je* n*-obmedzená sieť dosiahnuteľnosti siete* MW*. Nech* m_1^n *je ľubovoľné značkovanie siete dosiahnuteľnosti dosiahnuteľné z* m_0^r*. Potom pre každé značkovanie* m_2^r *dosiahnuteľné z* m_1^n *platí:*

$$\sum_{x \in [m_0\rangle|D} m_1^n(x) \leq \sum_{x \in [m_0\rangle|D} m_2^n(x)$$

Ak $m_1^r(source) = 0$ *potom platí:*

$$\sum_{x \in [m_0\rangle|D} m_1^n(x) = \sum_{x \in [m_0\rangle|D} m_2^n(x)$$

Pripomeňme, že počet možností, ako umiestniť v súčte i značiek do k rôznych miest je dané vzorcom (pre viac detailov pozri napr. [47]):

$$\binom{k+i-1}{k-1} = \frac{(k+i-1)!}{(k-1))! \cdot ((k+i-1)-(k-1))!} = \frac{(k+i-1)!}{(k-1))! \cdot i!}$$

Miesto *source* n-obmedzenej siete dosiahnuteľnosti môže nadobúdať dve rôzne značkovania - 0 a 1. Súčet značiek v miestach n-obmedzenej siete dosiahnuteľnosti patriacich do $[m_0\rangle|D$ môže byť 0 až n. Ak je súčet značiek v miestach n-obmedzenej siete dosiahnuteľnosti patriacich do $[m_0\rangle|D$ rovný i, potom značkovanie miesta *buffer* je rovné $n - i$. Dostávame teda, že počet dosiahnuteľných značkovaní n-obmedzenej siete dosiahnuteľnosti je ohraničený nasledovne:

Dôsledok 18 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je* n*-obmedzená sieť dosiahnuteľnosti siete* MW*. Nech* $k = |[m_0\rangle|D|$*, teda* k *označuje počet dosiahnuteľných* D*-značkovaní siete* MW*. Potom počet značkovaní dosiahnuteľných z počiatočného značkovania* m_0^n *v sieti* MPN^n *je ohraničený nasledovne:*

$$|[m_0^n\rangle| \leq \sum_{i=0}^{n} 2 \cdot \binom{k+i-1}{k-1} = \sum_{i=0}^{n} 2 \cdot \frac{(k+i-1)!}{(k-1))! \cdot i!}$$

Pre úplnosť definujme finálne značkovania, uviaznutia a základné uviaznutia n-obmedzenej siete dosiahnuteľnosti.

Definícia 43 (Finálne značkovania n-obmedzenej siete dosiahnuteľnosti) *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými*

miestami a korektným správaním a $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je n-obmedzená sieť dosiahnuteľnosti siete MW. Značkovanie* m_f^n *n-obmedzenej siete dosiahnuteľnosti* MPN^n*, ktoré je dosiahnuteľné z počiatočného značkovania* m_0^n*, nazývame finálne značkovanie, ak platí, že* $m_f^n(x) = 0$ *každé* $x \in P^n \setminus (S \cup \{out, buffer\}$.

Definícia 44 (Uviaznutie n-obmedzenej siete dosiahnuteľnosti) *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je n-obmedzená sieť dosiahnuteľnosti siete MW. Značkovanie* m^n *n-obmedzenej siete dosiahnuteľnosti* MPN^n*, ktoré je dosiahnuteľné z počiatočného značkovania* m_0^n*, nazývame uviaznutie, ak z* m^n *nie je dosiahnuteľné žiadne finálne značkovanie n-obmedzenej siete dosiahnuteľnosti* MPN^n.

Definícia 45 (Základné uviaznutie n-obmedzenej siete dosiahnuteľnosti) *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním, nech* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete MW a nech* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je n-obmedzená sieť dosiahnuteľnosti siete MW. Nech* K^r *sú kritické miesta siete dosiahnuteľnosti* MPN^r*. Potom základné uviaznutie n-obmedzenej siete dosiahnuteľnosti* MPN^n *je také uviaznutie* m^n *n-obmedzenej siete dosiahnuteľnosti* MPN^n*, pre ktoré platí* $m^n(x) = 0$ *pre každé* $x \in P^n \setminus (K^r \cup S \cup buffer)$*, teda v základnom uviaznutí n-obmedzenej siete dosiahnuteľnosti* MPN^n *môžu byť označené iba kritické miesta siete dosiahnuteľnosti, statické miesta a miesto buffer.*

Z Dôsledku 17 dostávame dôležitý výsledok, ktorý zefektívni detekciu základných uviaznutí.

Dôsledok 19 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je n-obmedzená sieť dosiahnuteľnosti siete MW. Značkovanie* m^n *dosiahnuteľné zo značkovania* m_0^n *v sieti* MPN^n*, pre ktoré platí* $m^n(x) = 0$ *pre každé* $x \in P^n \setminus (K^r \cup S \cup buffer)$*, je základným uviaznutím siete* MPN^n *práve vtedy, ak z neho nie je dosiahnuteľné finálne značkovanie* m_f^n *spĺňajúce* $\sum_{k \in K^r} m^n(k) = m_f^n(out)$.

Keďže novo pridané miesto *buffer* zabezpečí, že prechod *new* sa bude dať spustiť maximálne n-krát, avšak nijako neovplyvní spustiteľnosť prechodov ani značkovanie

zvyšných miest pred a po spustení prechodov pre ľubovoľnú dvojicu značkovaní pre ktoré platí, že suma v miestach patriacich do $[m_0\rangle|D$ nepresiahne n, platí nasledovný tretí hlavný výsledok tejto práce.

Veta 3 *Nech $MW = (D, S, T, I, O, m_0)$ je označkovaná určená workflow sieť so statickými miestami a korektným správaním, nech $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ je sieť dosiahnuteľnosti siete MW a nech $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ je n-obmedzená sieť dosiahnuteľnosti siete MW, kde $n \geq bound(K^r)$ (výsledok platí pre ľubovoľné n, ktoré nie je menšie ako $bound(K^r)$, teda aj pre $sbound(K^r)$). Značkovanie siete dosiahnuteľnosti $m^r \in [m_0^r\rangle$ je základným uviaznutím siete dosiahnuteľnosti MPN^r práve vtedy, keď značkovanie n-obmedzenej siete dosiahnuteľnosti $m^n \in [m_0^n\rangle$, pre ktoré platí $m^n|P^r = m^r$ a zároveň $m^n(buffer) = n - \sum_{k \in K^r} m^r(k)$, je základným uviaznutím n-obmedzenej siete dosiahnuteľnosti MPN^n.*

2.6 Detekcia základných uviaznutí

Základné uviaznutia siete dosiahnuteľnosti a n-obmedzenej siete dosiahnuteľnosti pre ľubovoľné n, ktoré nie je menšie ako $bound(K^r)$, sa teda líšia iba v počte značiek v mieste *buffer*, ktorého počet značiek zodpovedá nespotrebovaným značkám daným ako rozdiel medzi počiatočným obmedzením n a súčtom počtu značiek v kritických miestach.

Na záver môžeme zostaviť algoritmus na detekciu základných uviaznutí siete dosiahnuteľnosti MPN^r, Vstupom do algoritmu je sieť dosiahnuteľnosti. Ak sieť nemá uviaznutia, výstupom je informácia, že sieť dosiahnuteľnosti nemá uviaznutia. Ak sieť dosiahnuteľnosti uviaznutia má, potom je výstupom zoznam základných uviaznuti siete dosiahnuteľnosti MPN^r a informácia, že sieť dosiahnuteľnosti má uviaznutia.

Algoritmus 5 (Detekcia základných uviaznutí siete dosiahnuteľnosti)
Pre sieť dosiahnuteľnosti $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ získanú ako výstup Algoritmu 4 pre označkovanú určenú workflow sieť so statickými miestami $MW = (D, S, T, I, O, m_0)$

1. vytvor prázdny zoznam miest so zdrojom Z^r
2. pre každé miesto x z P^r okrem miesta *source*
   (a) ak $x|D_s > 0$ pridaj x do zoznamu Z^r
3. ak $Z^r = \emptyset$ algoritmus zastav a vráť informáciu "Sieť nemá uviaznutia"
4. vytvor prázdny zoznam kritických miest K^r siete dosiahnuteľnosti
5. pre každé miesto x z Z^r
   (a) pre každý prechod y z T^r okrem *new* a *stop*
      i. ak $I(x, y) > 0$ pridaj x do zoznamu K^r
6. ak $K^r = \emptyset$ algoritmus zastav a vráť informáciu "Sieť nemá uviaznutia"
7. vytvor premennnú n, vypočítaj $sbound(K^r)$ a vlož hodnotu $sbound(K^r)$ do premennej n alebo vypočítaj $bound(K^r)$ a vlož hodnotu $bound(K^r)$ do premennej n
8. premenuj zoznamy P^r, T^r, I^r, O^r, m_0^r na P^n, T^n, I^n, O^n, m_0^n

9. vlož *buffer* do zoznamu P^n
10. vlož trojicu $(buffer, y, 0)$ do zoznamu I^n pre každé y zo zoznamu T^n rôzne od new
11. vlož trojicu $(buffer, new, 1)$ do zoznamu I^n
12. vlož trojicu $(buffer, y, 0)$ do zoznamu O^n pre každé y zo zoznamu T^n
13. vlož dvojicu $(source, 1)$ do zoznamu m_0^n
14. vytvor prázdny zoznam M nájdených značkovaní
15. vytvor prázdny zoznam zoznam M_K^i značkovaní s i značkami v kritických miestach pre každé $i \in \{1 \dots n\}$
16. vytvor prázdny zoznam M_F finálnych značkovaní
17. vytvor prázdny zoznam H označených hrán
18. vlož do zoznamu M počiatočné značkovanie m_0^n a nastav množinu $Pre(m_0^n)$ jeho predchodcov prázdnu
19. pokiaľ je v zozname M značkovanie m_1^n, ktoré nie je označené ako preskúmané, vezmi ho a rob nasledovné:
    (a) pre každý prechod y z T^n spustiteľný z m_1^n
        i. počítaj značkovanie m_2^n dosiahnuté spustením prechodu t zo značkovania m_1^n
        ii. ak m_2^n ešte nie je v zozname M
            A. vlož m_2^n do zoznamu M a nastav množinu $Pre(m_2^n)$ jeho predchodcov prázdnu
            B. ak $m_2^n(x) = 0$ pre každé $x \in P^n \setminus (K^r \cup S \cup \text{buffer})$ potom pre každé $i \in \{1 \dots n\}$
               - ak $\sum_{k \in K^r} m_2^n(k) = i$ vlož m_2^n do zoznamu M_K^i
            C. ak $m_2^n(x) = 0$ pre každé $x \in P^n \setminus (S \cup \{out, \text{buffer}\}$ vlož m_2^n do zoznamu M_F
        iii. nastav množinu $Pre(m_2^n)$ predchodcov značkovania m_2^n rovnú zjednoteniu $Pre(m_2^n) \cup Pre(m_1^n) \cup \{m_1^n\}$

iv. vlož do zoznamu H hranu z m_1^n do m_2^n označenú prechodom y, teda trojicu (m_1^n, y, m_2^n)

(b) označ m_1^n ako preskúmané

20. vytvor premennú *uviaznutia* a vlož do nej hodnotu *nie* a vytvor premennú i

21. pre každé m_f^n zo zoznamu M_F

(a) vlož do premennej i hodnotu $i = m_f^n(out)$

(b) vytvor prázdny zoznam U^i základných uviaznutí s i značkami v kritických miestach

(c) vytvor premennú *i-uviaznutia* a vlož do nej hodnotu *nie*

(d) pre každé značkovanie m^n v zozname M_K^i

i. ak m^n nie je v množine $Pre(m_f^n)$

A. ak $m^n|(P^n \backslash buffer)$ nie je v zozname U^i, vlož $m^n|(P^n \backslash buffer)$ do zoznamu U^i

B. vlož do premennej *uviaznutia* hodnotu *áno*

C. vlož do premennej *i-uviaznutia* hodnotu *áno*

22. ak *uviaznutia* = *nie* zastav algoritmus a vráť informáciu "Sieť nemá uviaznutia"

23. ak *uviaznutia* = *áno*

(a) vráť informáciu "Sieť má uviaznutia"

(b) pre každé $i \in \{1 \dots n\}$

i. ak *i-uviaznutia* = *áno* vráť zoznam základných uviaznutí U^i

Kapitola 3

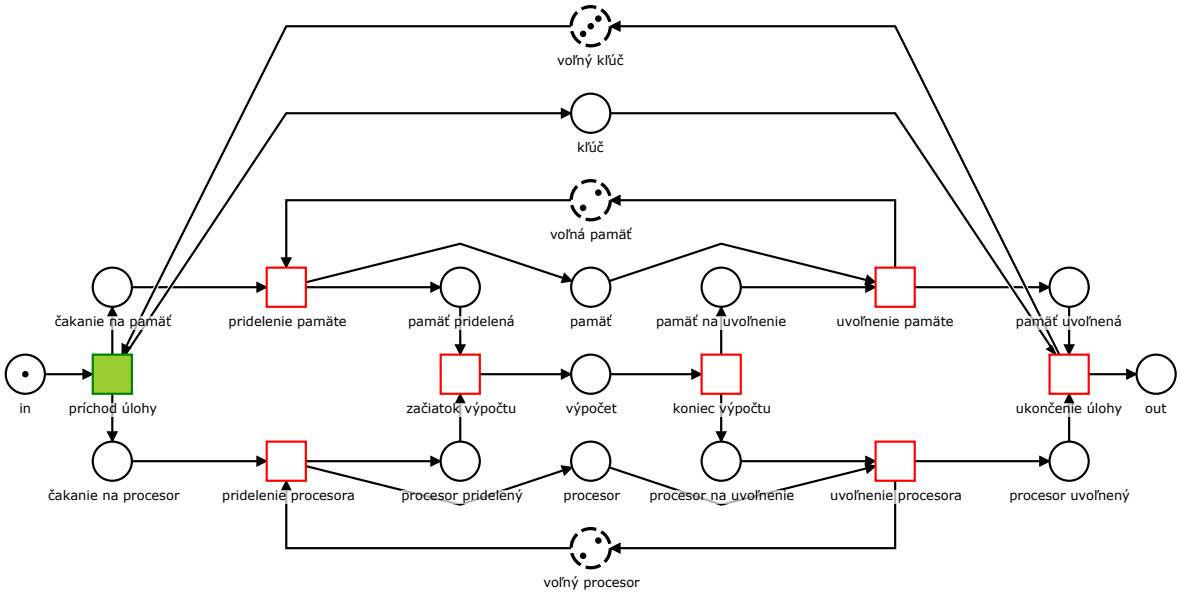
Prehľad príbuzných prác a námety na ďalší výskum

Workflow siete s obmedzenými zdrojmi pod originálnym anglickým názvom resource constrained workflow nets boli pôvodne definované v práci [6], vrátane požiadavky trvácnosti zdrojov. Z pohľadu tejto práce je však najdôležitejším článok [32], ktorý formuloval problém podobne, ako je prezentovaný v tejto práci. Metóda v práci [32] pracuje však s definíciou korektnosti, v anglickom originále soundness, v ktorej sa vyžaduje, aby existovalo také značkovanie statických miest, že sieť nemá žiadne uviaznutie pre toto značkovanie ani pre žiadne väčšie značkovanie statických miest.

Uvažujme napríklad sieť z Obr. 1.11. Vykonávaná sieť tohto procesu má nejaké uviaznutie pre ľubovoľné počiatočné značkovanie statických miest. Predstavme si, že v rámci budúceho výskumu by sme chceli nájsť doplnenie tejto siete o ďalšie statické miesta tak, aby existovalo značkovanie statických miest, pre ktoré jej vykonávaná sieť nebude mať uviaznutia. Výsledkom takéhoto doplnenia môže byť určená workflow sieť so statickými miestami a korektným správaním na Obr. 3.1. Sieť je doplnená o jedno statické miesto *voľný kľúč* a jeho komplementárne určujúce miesto *kľúč*. Úloha môže prísť do systému iba ak je v mieste *voľný kľúč* aspoň jedna značka. Spustenie prechodu *ukončenie úlohy* uvoľní používaný kľúč. Miesto *voľný kľúč* teda slúži ako ohraničenie kapacity paralelne spracovávaných úloh.

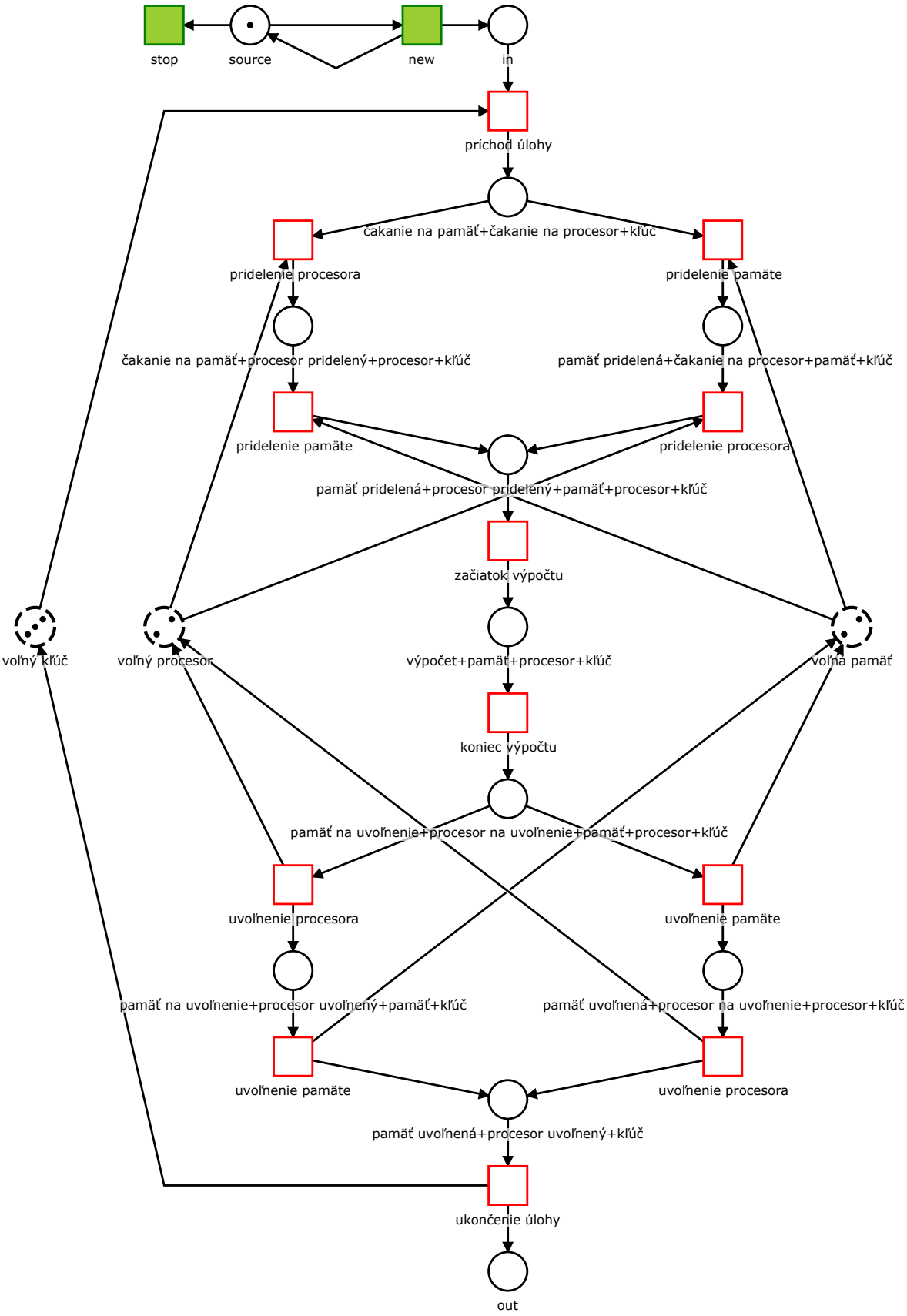
Na Obr. 3.2 je znázornená sieť dosiahnuteľnosti siete z Obr. 3.1.

Po vytvorení štyroch inštancií sa sieť dosiahnuteľnosti z Obr. 3.2 dostane do stavu na Obr. 3.3. Počet značiek v mieste *voľný kľúč* zabezpečí, že maximálny počet značiek

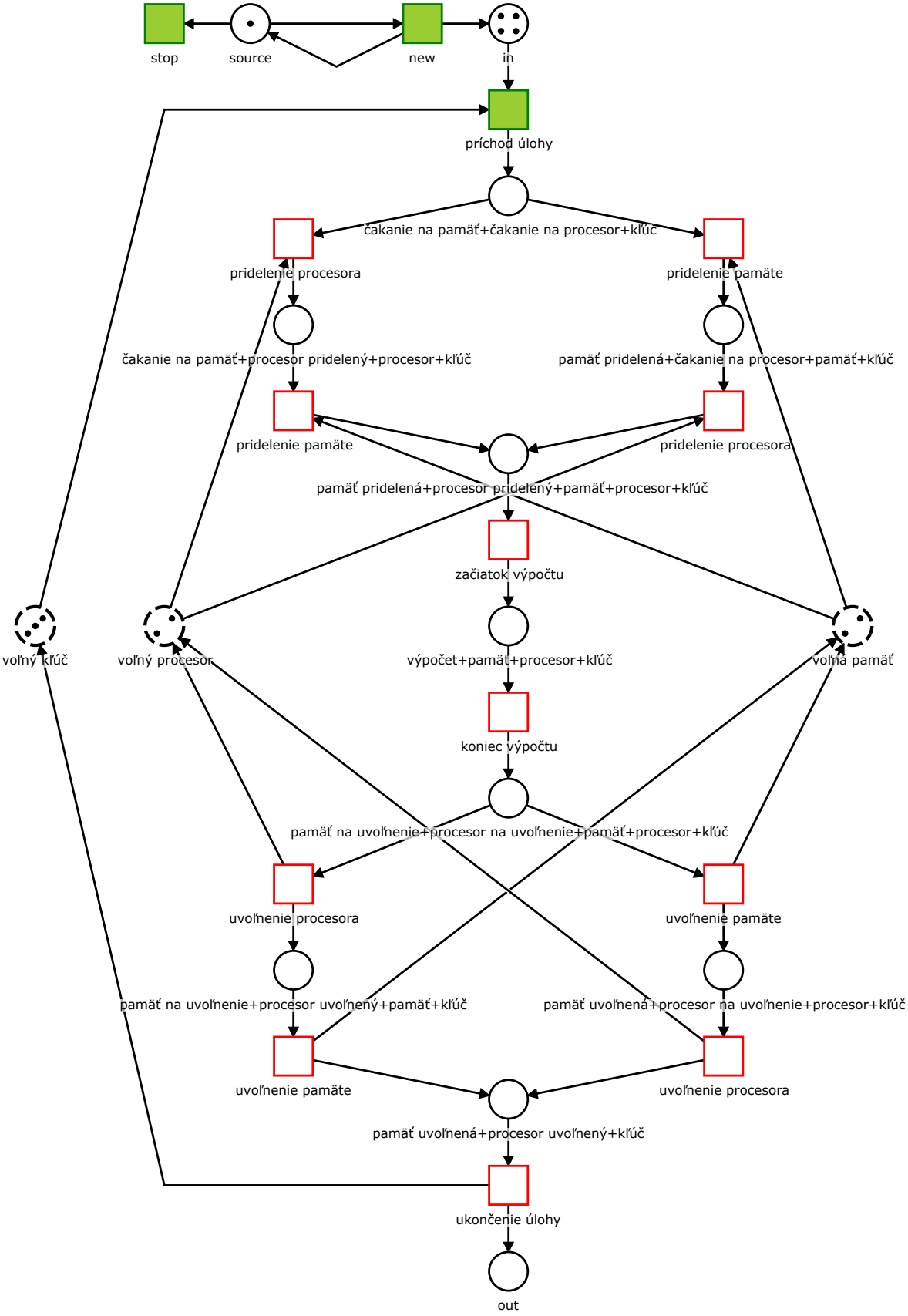


Obr. 3.1: Rozšírená označkovaná určená workflow sieť so statickými miestami, ktorých označkovanie modeluje počet pamäťových jednotiek a procesorov k dispozícii a kapacitu pre paralelné spracovanie úloh danú počtom značiek v mieste *voľný kľúč*

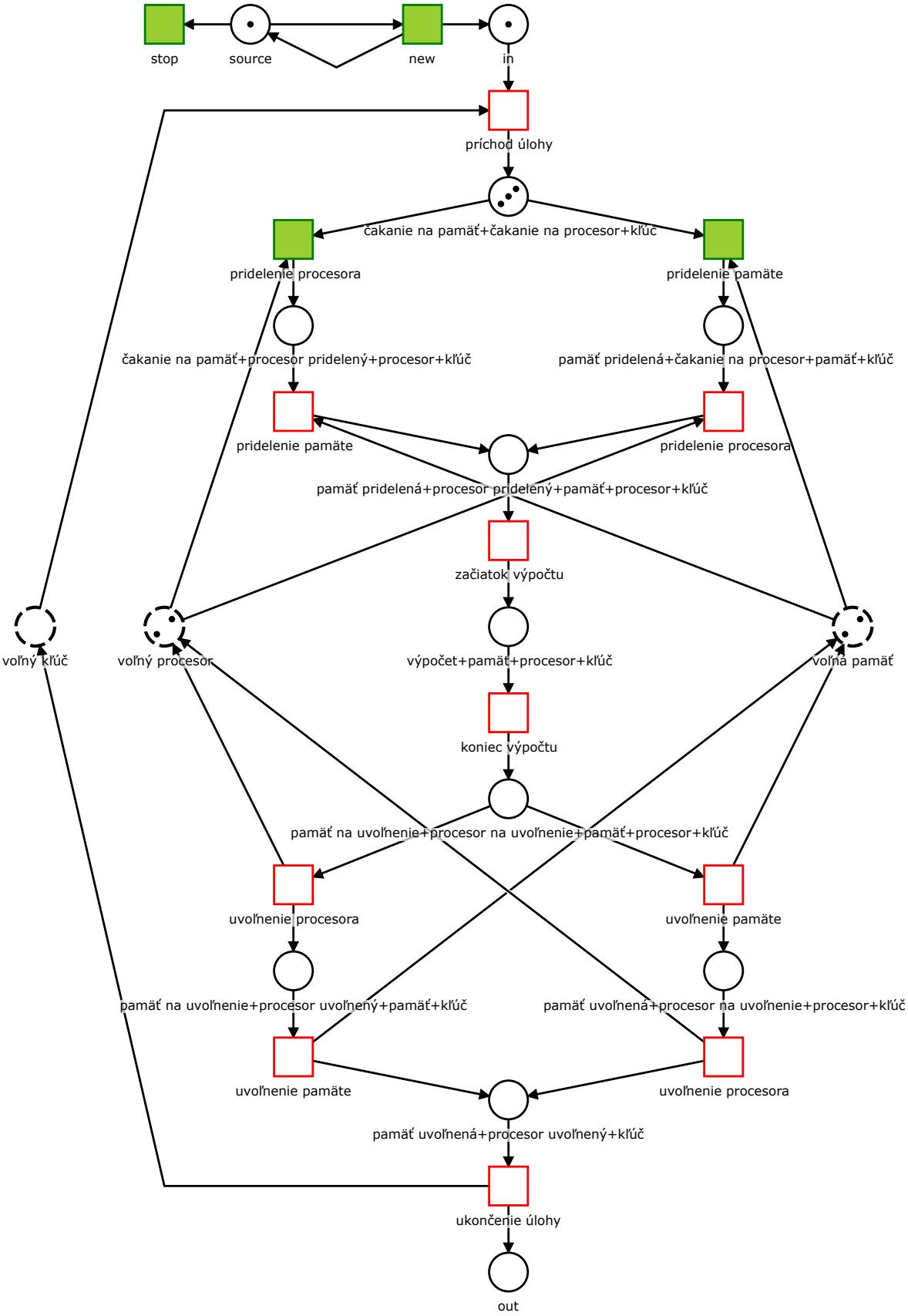
v mieste *čakanie na procesor* + *čakanie na pamäť* + *kľúč* bude rovný počiatočnému značkovaniu v mieste *voľný kľúč*, v tomto konkrétnom prípade 3, ako je znázornené na Obr. 3.4. Počiatočné značkovanie pridaného miesta *voľný kľúč* teda zabezpečí, že označkovaná určená workflow sieť so statickými miestami a korektným správaním na Obr. 3.1 nemá žiadne uviaznutia. Ak by sme zmenili značkovanie statických miest určenej workflow siete na Obr. 3.1 ľubovoľným spôsobom tak, že správanie siete zostane korektné (pre jednu inštanciu) a zároveň počet značiek v statickom mieste *voľný kľúč* bude menší ako súčet značiek v statických miestach *voľný procesor* a *voľná pamäť*, potom sieť nebude mať žiadne uviaznutia. V prípade, ak bude mať sieť korektné správanie (pre jednu inštanciu), ale v počiatočnom značkovaní nebude počet značiek v statickom mieste *voľný kľúč* menší ako súčet značiek v statických miestach *voľný procesor* a *voľná pamäť*, potom sieť bude mať uviaznutia. V takomto prípade bude mať sieť základné uviaznutie pre počet inštancií rovný súčtu značiek v statických miestach *voľný procesor* a *voľná pamäť*. Sieť dosiahnuteľnosti pre takýto prípad, konkrétne pre 4 značky v statickom mieste *voľný kľúč*, 2 značky v statickom mieste *voľný procesor* a 2 značky v statickom mieste *voľná pamäť* je znázornený na Obr. 3.5. V tomto prípade počet značiek v mieste *voľný kľúč* nezabezpečí dostatočné obmedzenie počtu paralelne spracovaných úloh, ako je zrejmé z Obr. 3.6 a sieť sa môže dostať do uviaznutia na Obr. 3.7.



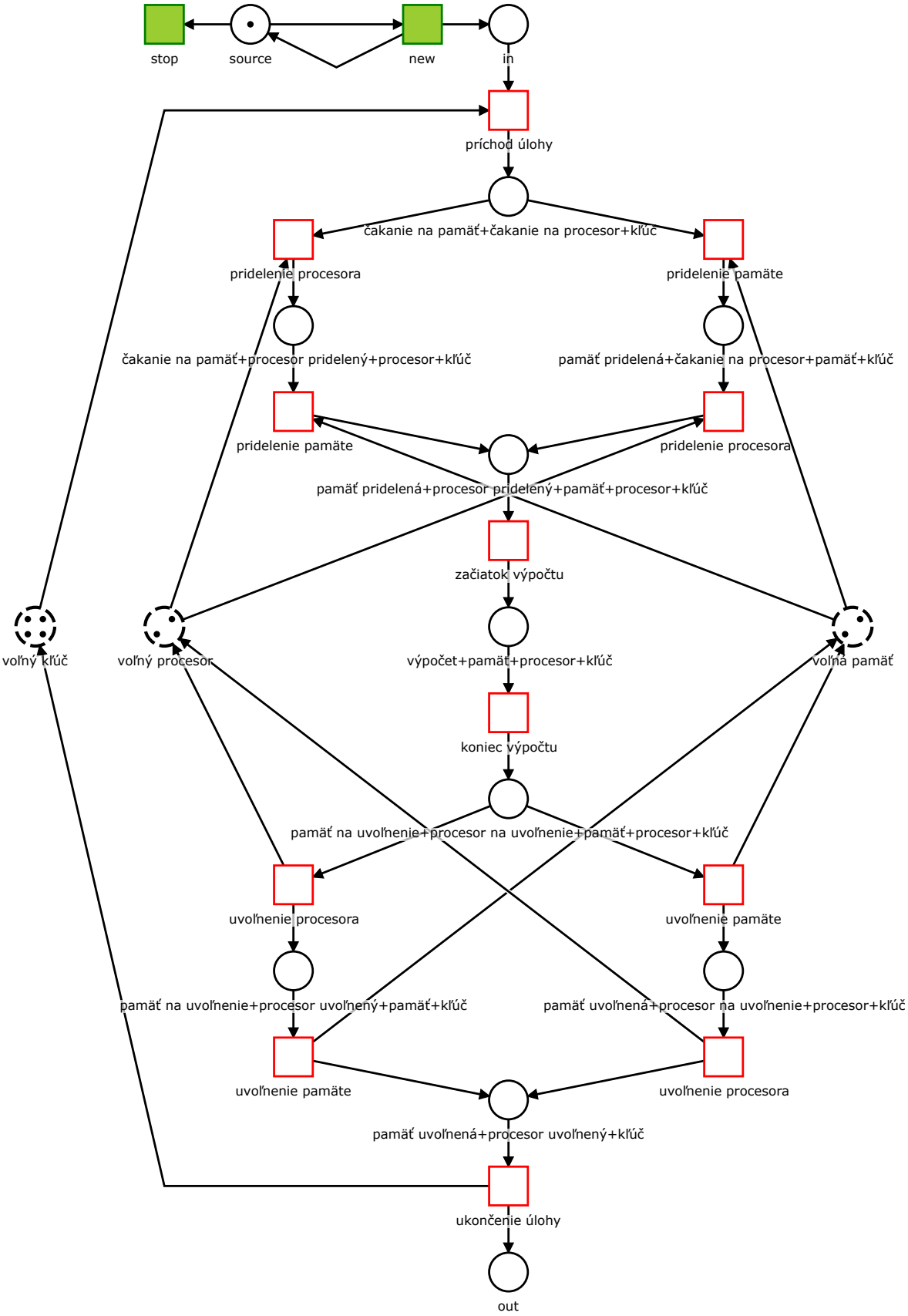
Obr. 3.2: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.1



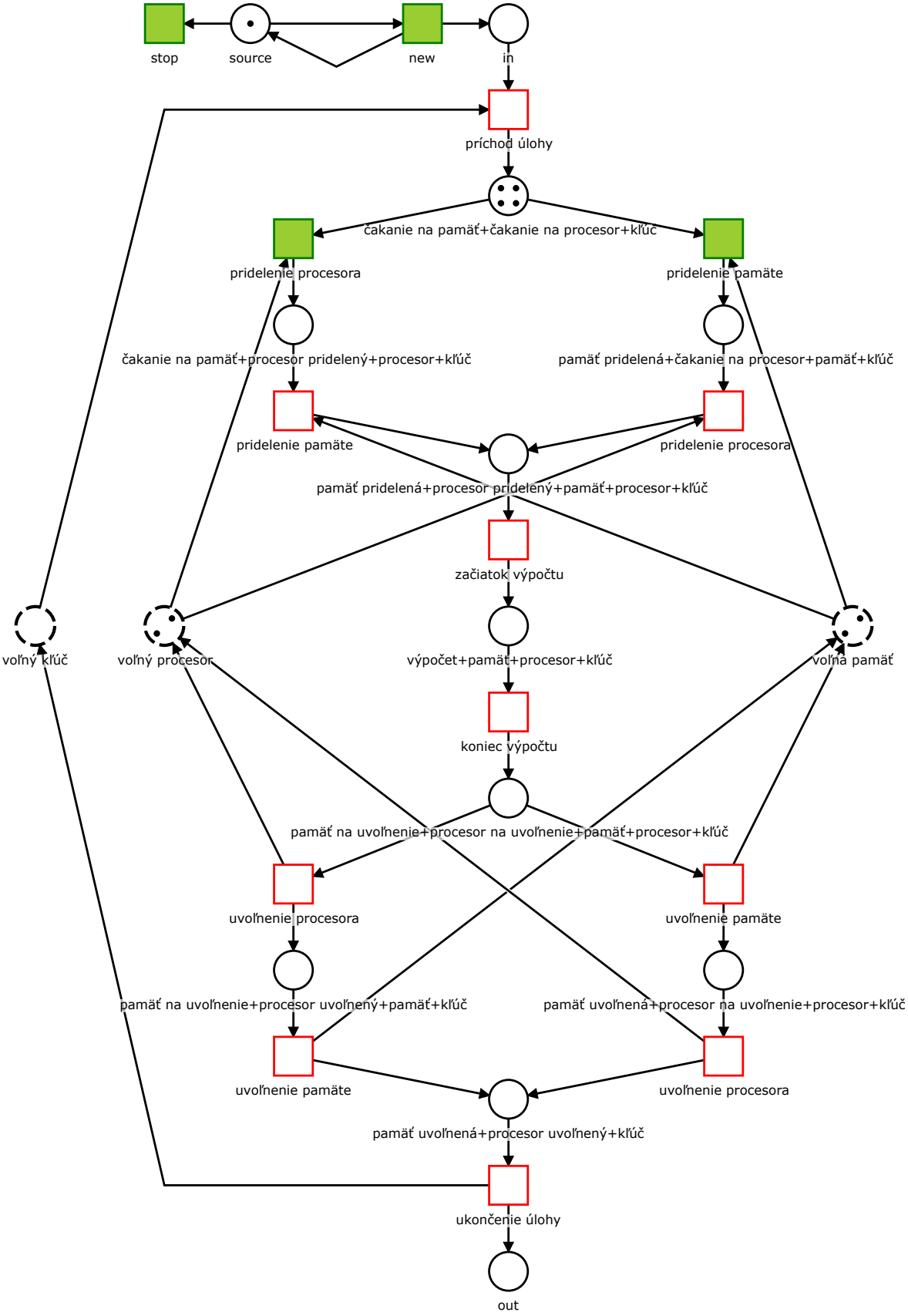
Obr. 3.3: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.1 po vytvorení 4 inšancií



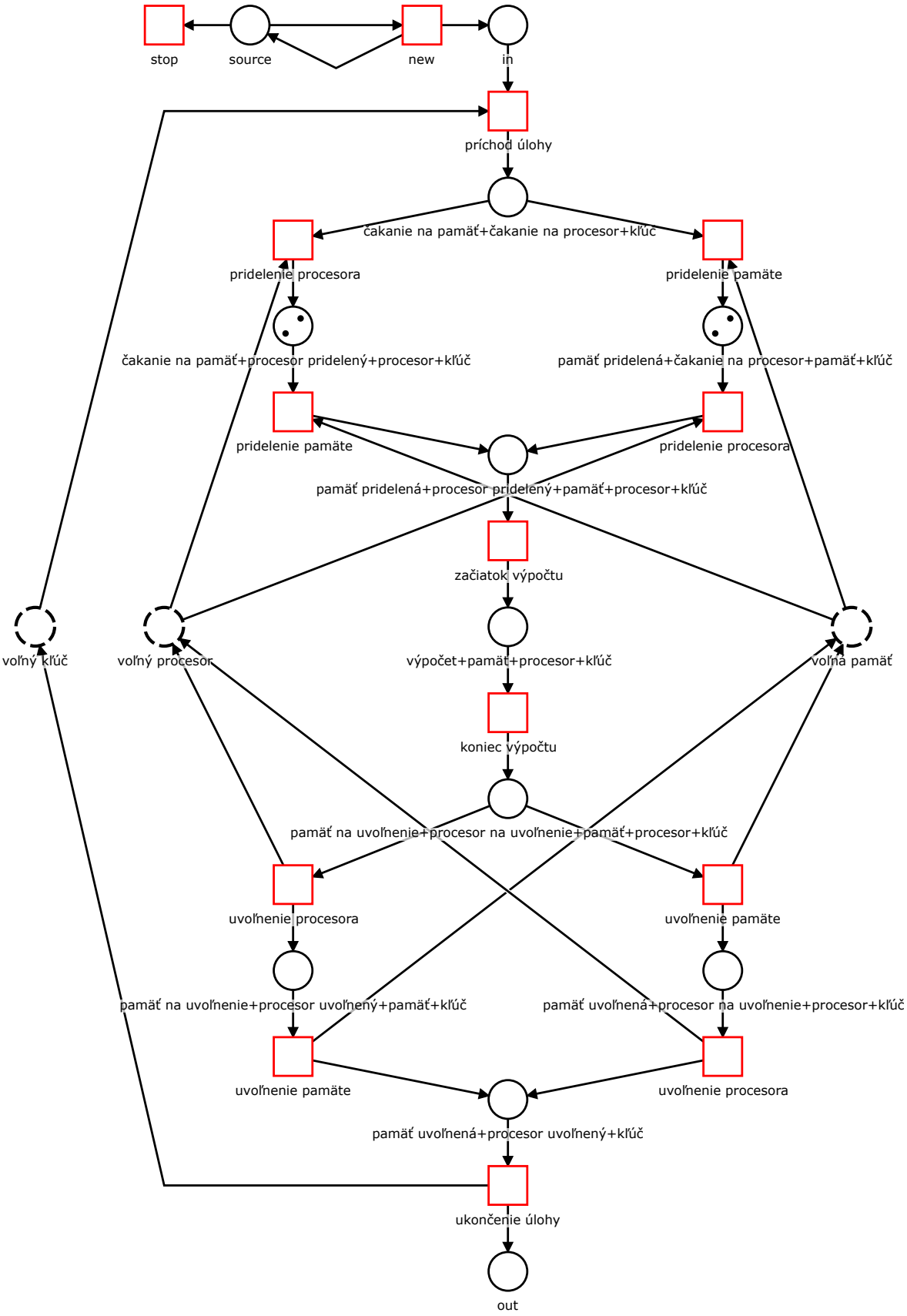
Obr. 3.4: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.1 s maximálnym počtom 3 paralelne spracovávaných úloh



Obr. 3.5: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.1 so zmeneným počiatočným značkovaním pridaním značky do miesta *voľný kľúč*



Obr. 3.6: Sieť dosiahnuteľnosti z Obr. 3.5 s maximálnym počtom 4 paralelne spracovávaných úloh



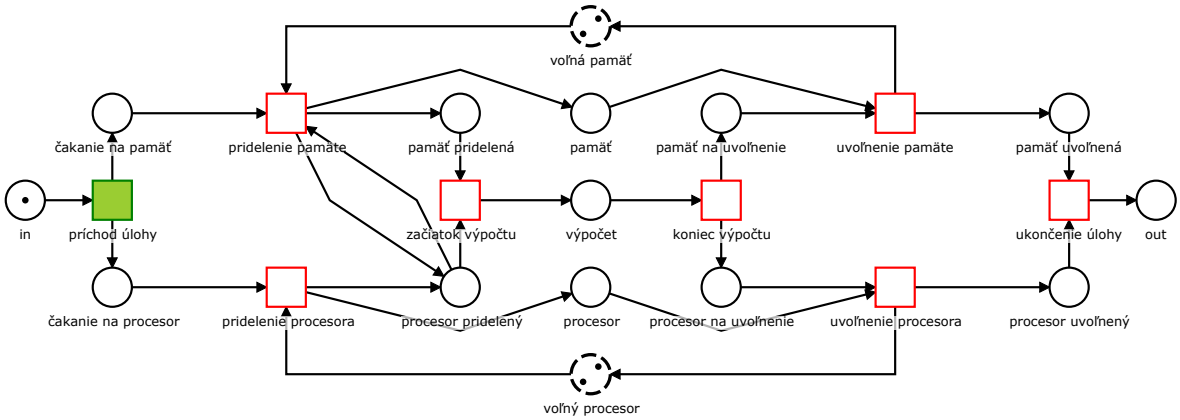
Obr. 3.7: Uviaznutie siete dosiahnuteľnosti z Obr. 3.5 s maximálnym počtom 4 paralelne spracovávaných úloh

Inou možnosťou ako doplniť sieť z Obr. 1.11 tak, aby jej vykonávaná sieť nemala uviaznutia, je doplniť hrany medzi prechodom *pridelenie pamäte* a miestom *procesor pridelený*, ako je to znázornené na Obr. 3.8. Doplnené hrany zabezpečia, že prechod *pridelenie pamäte* sa pre danú inštanciu nemôže uskutočniť skôr, ako bude pridelený procesor. Sieť dosiahnuteľnosti pre takto doplnenú označkovanú určenú workflow sieť so statickými miestami a s korektným správaním je znázornená na Obr. 3.9. Napriek tomu, že sieť dosiahnuteľnosti umožňuje príchod 4 úloh ako je znázornené na Obr. 3.10, sekvencializácia pridelenia procesora a pamäte zabezpečí, že viac ako 2 úlohy naraz nemôžu mať pridelený procesor alebo pamäť, ako je zrejmé z Obr. 3.11. Sieť dosiahnuteľnosti a teda aj vykonávaná sieť procesu z Obr. 3.8 nemá žiadne uviaznutia. Táto skutočnosť platí pre každé značkovanie statických miest, pre ktoré statické miesto *voľný procesor* obsahuje aspoň jednu značku a zároveň statické miesto *voľná pamäť* obsahuje aspoň jednu značku. Sieť na Obr. 3.8 teda predstavuje príklad procesu, v ktorom nie sú žiadne uviaznutia pre ľubovoľný počet zdrojov, ktoré postačujú na spracovanie aspoň jednej inštancie. Sieť na Obr. 3.8 teda spĺňa vlastnosť soundness ako bola definovaná v článku [32]. V porovnaní s Obr. 3.1 však za túto vlastnosť platíme znížením počtu paralelne spracovávaných inštancií. V prípade, ak je dôležité začať spracovanie maximálne možného počtu inštancií, takéto zníženie nemusí byť želané. Predstavme si pre ilustráciu, že v príklade namiesto procesora a pamäte budú vystupovať lekár a sestrička a namiesto spracovania výpočtovej úlohy bude proces modelovať ošetrenie pacienta. V takomto prípade môže byť želané a kľúčové, aby v prípade príchodu pacienta sa prvý voľný lekár alebo prvá voľná sestrička okamžite venoval resp. venovala ošetreniu pacienta.

Ak budeme považovať počiatočné značkovanie ako parameter, môžeme situáciu v príkladoch sietí z Obr. 1.11, Obr. 3.1 respektíve z Obr. 3.8 považovať za reprezentantov typu dynamickej korektnosti.

Definícia 46 (Typy dynamickej korektnosti)
Nech $W = (D, S, T, I, O)$ je workflow sieť so statickými miestami.

- *Ak existuje také značkovanie m_0 siete W, že vykonávaná sieť $MPN^{\infty} = (P^{\infty}, T^{\infty}, I^{\infty}, O^{\infty}, m^{\infty})$ označkovanej workflow siete $MW = (D, S, T, I, O, m)$ nemá uviaznutia pre žiadne počiatočné značkovanie $m \geq m_0$, potom sieť W nazývame dynamicky korektná.*



Obr. 3.8: Sekvencializovaná označkovaná určená workflow sieť so statickými miestami modelujúca spracovanie výpočtovej úlohy

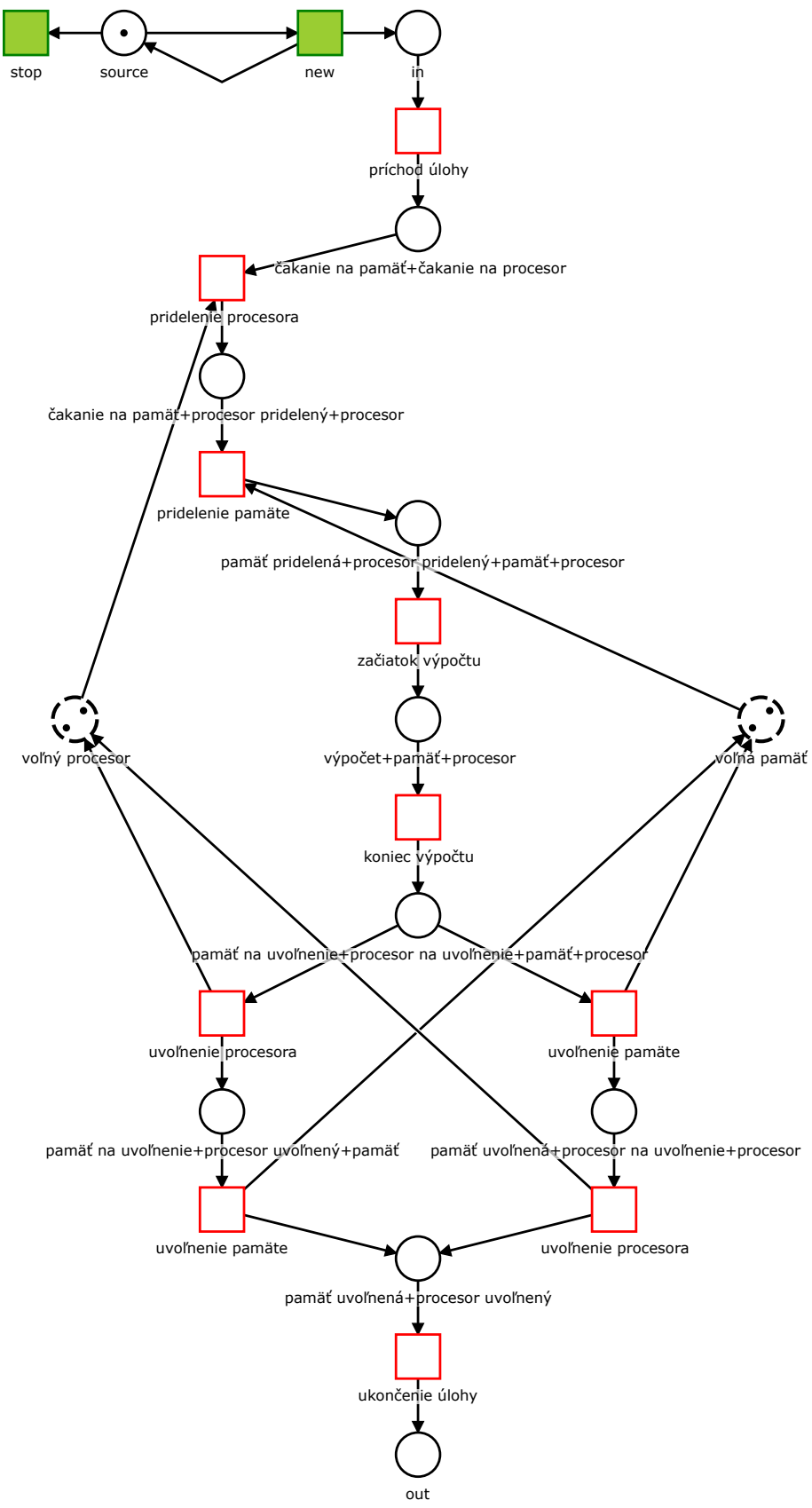
- *Ak sieť W nie je dynamicky korektná a zároveň existuje značkovanie m_0 siete W pre ktoré platí, že vykonávaná sieť $MPN^{\infty} = (P^{\infty}, T^{\infty}, I^{\infty}, O^{\infty}, m_0^{\infty})$ označkovanej workflow siete $MW = (D, S, T, I, O, m_0)$ nemá uviaznutia, potom sieť W nazývame dynamicky semi-korektná. Množina počiatočných značkovaní, pre ktoré dynamicky semi-korektná workflow sieť nemá uviaznutia sa nazýva korektný priestor značkovaní. Množina počiatočných značkovaní, pre ktoré dynamicky semi-korektná workflow sieť môže uviaznuť, sa nazýva nekorektný priestor značkovaní.*

- *Ak pre každé značkovanie m_0 siete W platí, že vykonávaná sieť $MPN^{\infty} = (P^{\infty}, T^{\infty}, I^{\infty}, O^{\infty}, m_0^{\infty})$ označkovanej workflow siete $MW = (D, S, T, I, O, m_0)$ má uviaznutia, potom sieť W nazývame dynamicky nekorektná.*

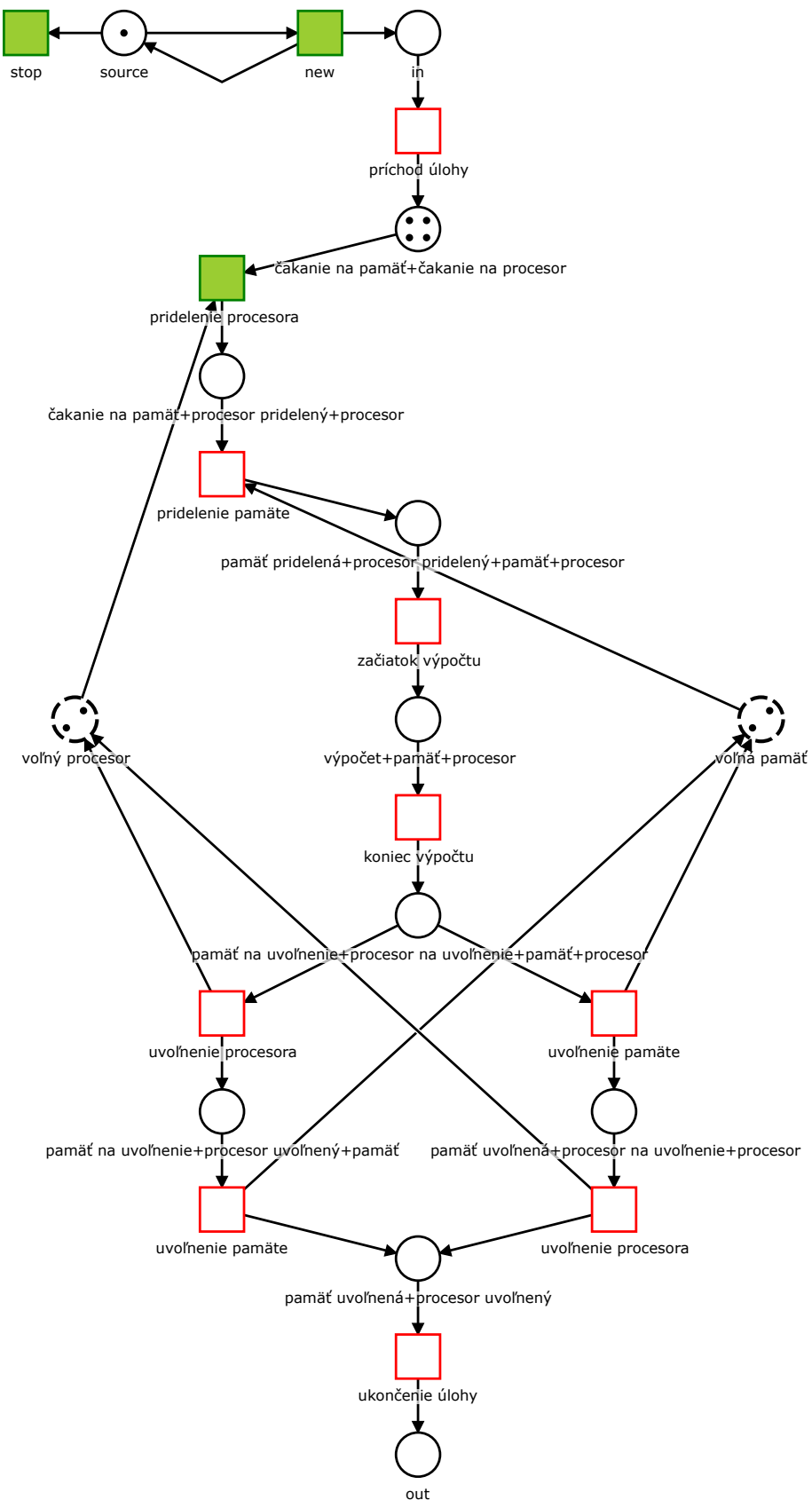
V zmysle predchádzajúcej definície teda sieť z Obr. 1.11 je dynamicky nekorektná, Obr. 3.1 je dynamicky semi-korektná a sieť z Obr. 3.8 je dynamicky korektná.

Metóda prezentovaná v práci v práci [32] teda vyšetruje dynamickú korektnosť pre jedno statické miesto. V práci [7] sa autori sústreďujú na stanovenie korektnosti pre podtriedy workflow sietí. V práci [33] sú prezentované 4 nutné podmienky dynamickej korektnosti založené na štrukturálnej analýze Petriho sietí. Dynamická korektnosť je inšpirovaná zovšeobecnenou korektnosťou workflow sietí podľa [31]. Korektnosť je skúmaná pre rôzne rozšírenia workflow sietí, napr. v článku [70] je definovaná korektnosť pre workflow siete s dátami.

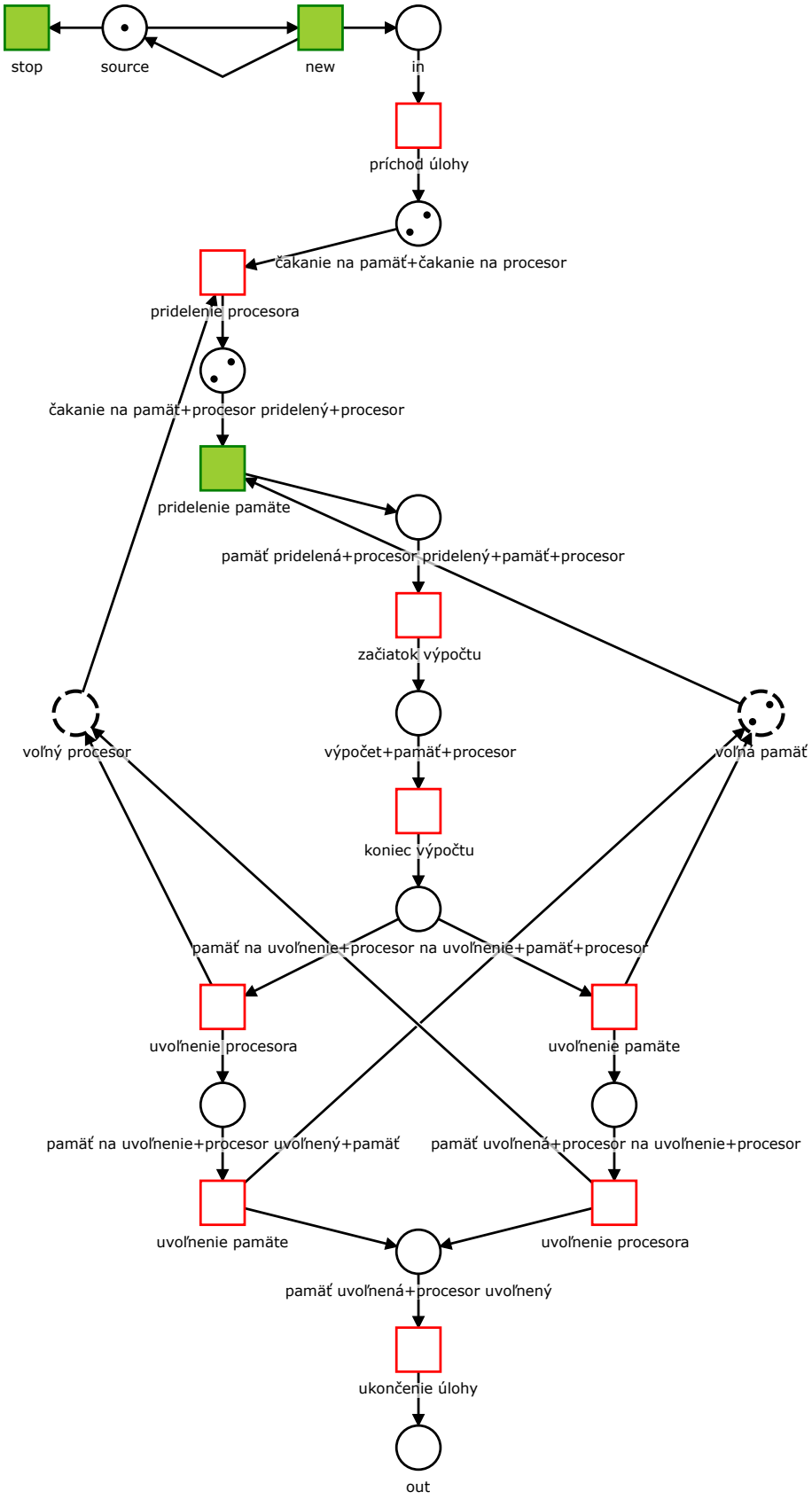
V prácach [41, 42, 43] sme postupne definovali techniku konštruktora a siete dosiahnuteľnosti a metódu detekcie uviaznutí typu zamrznutie pre workflow siete s pevne da-



Obr. 3.9: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.8



Obr. 3.10: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.8 po príchode 4 úloh



Obr. 3.11: Maximálny počet úloh s prideleným procesorom alebo pamäťou je v sieti dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 3.8 rovný 2

ným počiatočným značkovaním statických miest, trvácnymi zdrojmi, korektným správaním pre jednu inštanciu a nezávislými inštanciami pre ľubovoľný konečný počet statických miest a ľubovoľný počet inštancií. Metóda prezentovaná v [41] však nebola definovaná pre uviaznutia typu zacyklenie.

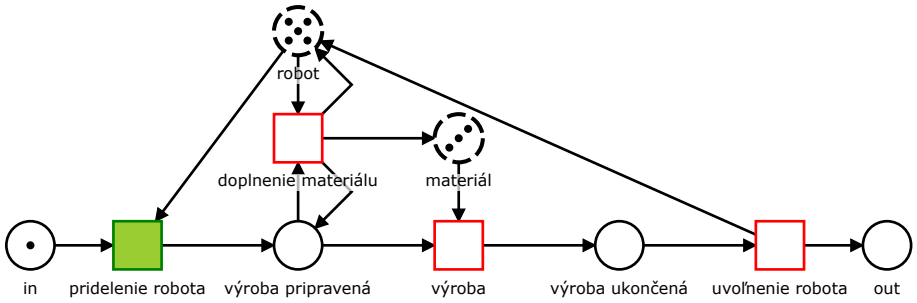
Práca [41] získala doposiaľ 20 ohlasov v databáze Google Scholar, pričom 16 ohlasov sa nachádza v databáze Web of Science (9 ohlasov) alebo v databáze Scopus (14 ohlasov).

Jedným z rozšírení výsledkov práce [41] je táto práca, ktorá definuje prvú metódu na detekciu ľubovoľných uviaznutí pre workflow siete s pevne daným počiatočným značkovaním statických miest, trvácnymi zdrojmi formalizovanými prostredníctvom určujúcich miest, korektným správaním pre jednu inštanciu a nezávislými inštanciami pre ľubovoľný konečný počet statických miest a ľubovoľný počet inštancií. Práca obsahuje dôkazy prezentovaných výsledkov a dvojstupňový algoritmus na detekciu uviaznutí. Prvý stupeň algoritmu vyšetrí korektnosť určenej workflow siete a v prípade, že je workflow sieť korektná skonštruuje jej sieť dosiahnuteľnosti. Druhý stupeň algoritmu realizuje samotnú detekciu základných uviaznutí siete dosiahnuteľnosti prostredníctvom vyšetrenia ohraničenej Petriho siete.

V prácach [54, 57] autori s využitím techniky konštruktora a siete dosiahnuteľnosti prezentovanej v článku [41] dokázali, že problém určenia dynamickej korektnosti pre workflow siete s korektných správaním je rozhodnuteľný (aj v prípade, ak zdroje nie sú trvácne). Dôkaz spočíva v prevode problému na problém domáceho priestoru Petriho sietí (anglicky home space). Viac informácii o probléme domáceho priestoru Petriho sietí je možné nájsť v práci [27]. Zároveň autori [54, 57] dokázali, že vo všeobecnosti, teda ak nie je vyžadovaná korektnosť správania pre jednu inštanciu a zdroje nemusia byť trvácne, problém dynamickej korektnosti je nerozhodnuteľný. V prácach [54, 57] autori používajú terminológiu statických miest zavedenú v práci [41]. V práci [71] autori s využitím techniky konštruktora z našej práce [41] dokázali, že problém určenia dynamickej korektnosti pre workflow siete s korektným správaním a trvácnymi zdrojmi je rozhodnuteľný i v prípade, že pre inštancie nie je požadované, aby boli izolované resp. separované, pričom namiesto siete dosiahnuteľnosti použili autori v [71] jednoducho sieť s konštruktorom. Pre viac informácii k téme separovateľnosti odkazujeme najmä na práce [30, 12, 13]. Práca [71] nadväzuje na neúspešný pokus dokázať rozhodnuteľnosť

dynamickej korektnosti prezentovaný v [74]. V dôkaze výsledkov [71] použili autori opäť redukciu na problém domáceho priestoru Petriho sietí. Ako sa konštatuje v závere práce [71], hoci dynamická korektnosť je rozhodnuteľná, doposiaľ nie je žiadny efektívny algoritmus, pretože algoritmus navrhnutý v [71] rozhoduje na základe domáceho priestoru v Petriho sieťach, ktorý vyžaduje overenie všeobecnej dosiahnuteľnosti v potencionálne neohraničených Petriho sieťach. Podobne v závere originálneho článku [27], ktorý rieši problém domáceho priestoru Petriho sietí je možné nájsť konštatovanie, že navrhovaná rozhodovacia procedúra je príliš komplexná, keďže používa veľmi komplexný algoritmus na vyšetrenie všeobecnej dosiahnuteľnosti v potencionálne neohraničených sieťach. Procedúry na vyšetrenie dynamickej korektnosti navyše pre semi-korektné siete, napr. pre sieť na Obr. 3.1 rozhodnú, že sieť nie je dynamicky korektná, napriek tomu, že pre počiatočné značkovanie na Obr. 3.1 sieť dosiahnuteľnosti a teda aj vykonávaná sieť nemá uviaznutia. Procedúry navrhnuté v článkoch [54, 57, 71] teda nevedia rozlíšiť dynamicky nekorektné a dynamicky semi-korektné siete. Viac výsledkov o problémoch ohraničenosti je možné nájsť v článkoch [52, 65, 69], problém všeobecnej dosiahnuteľnosti vrátane neohraničenýchh sietí bol vyriešený v prácach [58, 46, 59, 48, 51]. Prehľad výpočtovej zložitosti a rozhodnuteľnosti problémov v Petriho sieťach je možné nájsť v prácach [36, 28].

Nasledujúci odsek je venovaný stručnému prehľadu prác, ktoré citujú výsledky článku [41]. Okrem prác [54, 57] a [71], ktorým sme sa venovali podrobnejšie v predchádzajúcom odseku, sú to nasledovné práce: Články [9, 10] sú venované definovaniu rolí a personifikácii značiek v statických miestach. Práca [15] definuje workflow siete s obmedzenými zdrojmi, ktoré majú časované hrany. Práca [18] sa venuje verifikácii kompozície web-servisov. Článok [53] aplikuje Petriho siete v prípadovej štúdii. Práca [41] je citovaná taktiež v zdrojoch [68, 75]. Články [55, 56] a už spomenutá práca [57] sa venujú rozšíreniam workflow sietí s obmedzenými zdrojmi uvažovaním času a priradením ceny nákladov spojených so spustením prechodov a uchovaním značiek v miestach. Článok [66] sa venuje doplneniu dynamicky nekorektných workflow sietí tak, aby po doplnení boli dynamicky korektné. Doplnenie je realizované pomocou takzvaného prostredia, ktoré môže procesu požičať značky do statických miest. V práci [73] je článok [41] citovaný v kontexte dolovania procesov z počítačových logov. Séria článkov [76, 77, 78, 79, 81] sa venuje analýze minimálneho počtu zdrojov v acyklických



Obr. 3.12: Označkovaná neurčená workflow sieť so statickými miestami modelujúcimi spotrebný materiál a robotov v jednoduchom procese, v ktorého inštanciách robot buď doplní materiál alebo vyrobí výrobok, pričom spotrebuje materiál

a cyklických workflow procesoch a ich aplikáciam v prípadových štúdiách v medicínskych procesoch. Výsledky z práce [41] sú citované taktiež v prehľadovom článku [5] o korektnosti workflow sietí.

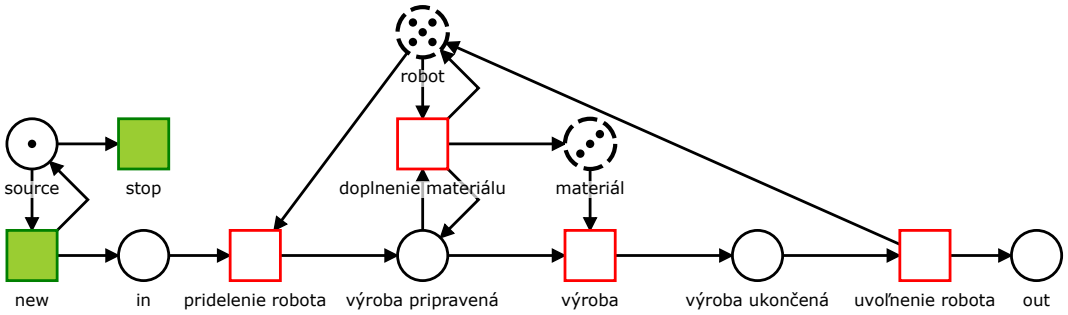
V nasledujúcich odsekoch sa budeme venovať možným rozšíreniam výsledkov tejto práce.

Predpokladajme, že pre danú podtriedu workflow sietí vieme určiť taký počet inštancií n, že platí výrok: ak vykonávaná sieť má uviaznutia, potom má uviaznutia pre n-inštancií. V takom prípade je možné adaptovať Algoritmus 5 na detekciu uviaznutí. Podtrieda workflow sietí, v ktorých je možné určiť takéto obmedzenie na počet inštancií nazvime siete s ohraničenými uviaznutiami. Pre takéto podtriedy je teda možné nájsť algoritmus, ktorý vyšetrí existenciu uviaznutí pomocou ohraničených Petriho sietí bez nutnosti použiť všeobecný algoritmus dosiahnuteľnosti, respektíve všeobecný algoritmus pre vyšetrenie domáceho priestoru Petriho sietí.

Ďalšia možnosť rozšírenia tejto práce je detekcia uviaznutí v sieťach, ktoré nie sú určené, teda v sieťach, v ktorých statické miesta nemajú komplementárne určujúce miesta. V takýchto sieťach môžu inštancie zdroje vytvárať aj spotrebovať. Neurčené workflow siete umožňujú teda modelovať procesy, v ktorých jedna inštancia spotrebuje zdroje vyprodukované inou inštanciou.

Posledný príklad ukazuje konkrétnu sieť, ktorá ilustruje podtriedu neurčených workflow sietí kombinovanú s ohraničenými uviaznutiami.

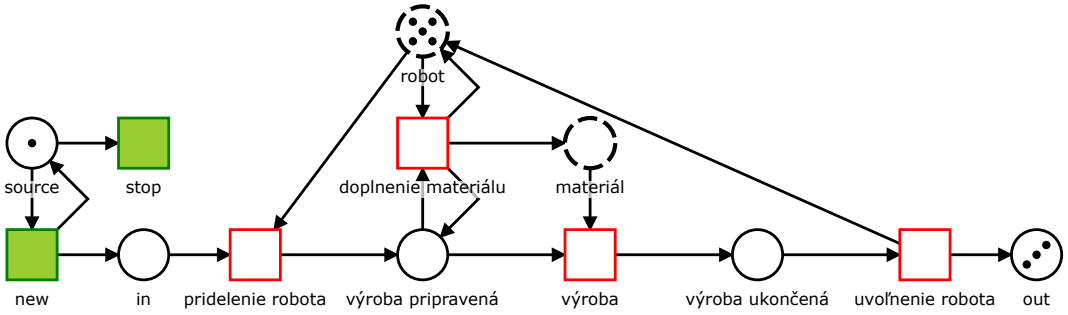
Uvažujme jednoduchú výrobnú linku, zobrazenú na Obr. 3.12. V procese vystupujú ako zdroje roboty a spotrebný materiál, modelované statickými miestami. V rámci jednej inštancie procesu robot buď dopĺňa spotrebný materiál alebo vyrobí výrobok,



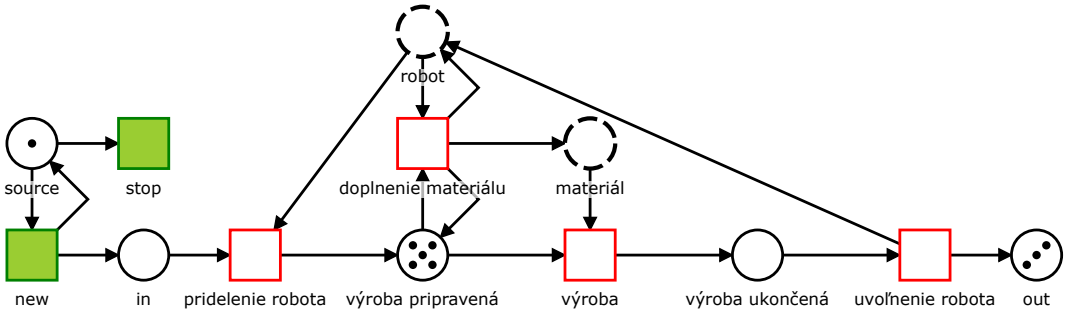
Obr. 3.13: Sieť dosiahnuteľnosti označkovanej neurčenej workflow sieť so statickými miestami a korektným správaním Obr. 3.12

pričom spotrebuje jednotku spotrebného materiálu. Na Obr. 3.13 je znázornená sieť dosiahnuteľnosti procesu na Obr. 3.12. Táto sieť, respektíve zodpovedajúca vykonávaná sieť má uviaznutia pre ľubovoľné značkovanie statických miest, teda je dynamicky nekorektná. V prípade, ak originálna workflow sieť nemá označkované statické miesto robot, potom má sieť dosiahnuteľnosti, respektíve zodpovedajúca vykonávaná sieť uviaznutie už pre jednu inštanciu. Inak má sieť dosiahnuteľnosti respektíve vykonávaná sieť procesu na Obr. 3.12 uviaznutia pre $m_0(robot)+m_0(materiál)$ inštancií, kde m_0 označuje počiatočné značkovanie siete z Obr. 3.12. Toto uviaznutie je možné dosiahnuť korektným spustením $m_0(materiál)$ inštancií, pričom pre každú inštanciu spustíme prechody *new, pridelenie robota, výroba, uvoľnenie robota*. Takýmto spôsobom spotrebujeme $m_0(materiál)$ jednotiek materiálu, čo je pre $m_0(materiál) = 3$ ilustrované na Obr. 3.14. Následne pre $m_0(robot)$ inštancií spustíme postupnosť *new, pridelenie robota*, čo spôsobí uviaznutie ilustrované pre $m_0(robot) = 5$ na Obr. 3.15. Workflow sieť znázornená na Obr. 3.12 teda reprezentuje proces, pre ktorý je možné určiť ohraničený počet inštancií ako funkciu počiatočného značkovania statických miest. Existenciu uviaznutí potom stačí overiť pre tento ohraničený počet inštancií. Cieľom budúceho výskumu môže byť charakterizácia takých podtried workflow sietí, pre ktoré je možné takéto ohraničenie inštancií určiť zo štruktúry siete a z počiatočného značkovania statických miest.

Predchádzajúce príklady priamo identifikujú viaceré témy pre budúci výskum. Problém určenia dynamickej semi-korektnosti a určenia korektného a nekorektného priestoru značkovaní doposiaľ nebol vyriešený ani pre určené ani pre neurčené workflow siete a predstavuje jeden z hlavných cieľov budúceho výskumu. Taktiež syntéza doplnenia



Obr. 3.14: Sieť dosiahnuteľnosti označkovanej neurčenej workflow sieť so statickými miestami a korektným správaním po spotrebovaní materiálu



Obr. 3.15: Uviazutie v sieti dosiahnuteľnosti označkovanej neurčenej workflow sieť so statickými miestami a korektným správaním pridelením všetkých robotov do výroby bez doplnenia spotrebovaného materiálu

workflow siete pre konkrétne značkovanie statických miest tak, aby neobsahovala uviaznutia, predstavuje tému budúceho výskumu. Všeobecnejšie, syntéza doplnenia dynamicky nekorektnej workflow siete na minimálne reštriktívnu dynamicky semi-korektnú workflow sieť s určením korektného a nekorektného priestoru značkovaní je ďalšou veľmi významnou témou budúceho výskumu. Ďalším možným cieľom budúceho výskumu je identifikovať podtriedy určených a neurčených workflow sietí s ohraničenými uviaznutiami. Problémom stále zostáva zostavenie efektívneho algoritmu na vyšetrenie dynamickej korektnosti workflow sietí.

Záver

Hlavným cieľom tejto práce bolo zostaviť metódu a algoritmus na detekciu uviaznutí workflow procesov s nezávislými inštanciami a viacerými typmi zdieľaných trvácnych zdrojov pri danom počte zdrojov jednotlivých typov pre ľubovoľný počet inštancií.

Tento cieľ bol naplnený nasledovne:

V Kapitole 1 sme popísali definície formalizmov pre modelovanie diskrétnych udalostných systémov s inštanciami a zdieľanými zdrojmi, ktoré boli použité v práci.

V Časti 1.1 sme popísali prechodové systémy podľa [45, 80] ako základný modelovací formalizmus diskrétnych udalostných systémov používaný v práci.

V Časti 1.2 sme uviedli základné definície Petriho sietí, ktoré sú pomenované podľa Carla Adama Petriho [64] a sú popísané napríklad v prácach [44, 62, 63, 67, 23, 24]. Petriho siete predstavujú hlavný formalizmus na modelovanie diskrétnych udalostných systémov a workflow procesov používaný v práci.

Kapitola 1 pokračuje Časťou 1.3, v ktorej sú popísané workflow siete podľa [1, 2, 3, 4].

V Časti 1.4 sme sa venovali popisu sietí zo statickými miestami, ktoré modelujú zdroje. Táto trieda sietí bola pôvodne definovaná v prácach [6, 32, 33] pod názvom workflow siete s obmedzenými zdrojmi (anglicky resource constrained workflow nets). V porovnaní s prácami [6, 32, 33] sme formalizovali skutočnosť, že zdroje sú trvácne, pomocou komplementárnych miest [25] pre statické miesta. Tieto komplementárne miesta sme nazvali určujúce miesta statických miest a takéto siete sme nazvali určené workflow siete. Analogicky s prácou [4] sme definovali korektné správanie (anglicky soundness) ako schopnosť ukončiť korektne jednu inštanciu.

V Časti 1.5 sme popísali úpravou definície z článkov [41, 42, 43] model vykonávanej siete pre workflow sieť so statickými miestami pre ľubovoľný počet inštancií. Vykoná-

vané siete sú ekvivalentné sieťam s identifikačnými číslami používaným v práci [32]. Siete s identifikačnými číslami pre značky jednotlivých inštancií sú špeciálnou podtriedou farebných Petriho sietí, definovaných napríklad v prácach [37, 38, 39, 40]. V Časti 1.6 sme formalizovali pojem uviaznutia vykonávanej siete.

Kapitola 2 je venovaná hlavnej časti vlastného výskumu - detekcii uviaznutí vo vykonávanej sieti pre určenú workflow sieť so statickými miestami, ktorá má korektné správanie.

V prvej časti tejto kapitoly, teda v Časti 2.1 sme pridaním konštruktora k workflow sieťam so statickými miestami definovali sieť s konštruktorom a ukázali sme, že pridanie konštruktora nestačí na detekciu uviaznutí. Ilustrovali sme situáciu, kde sieť s konštruktorom neodhalí existujúce uviaznutie vykonávanej siete a naopak, situáciu, kde sieť s konštruktorom odhalí uviaznutie neexistujúce vo vykonávanej sieti.

V Časti 2.2 sme definovali siete dosiahnuteľnosti. Následne sme skonštruovali algoritmus, ktorý v prípade, keď vstupná určená workflow sieť so statickými miestami má korektné správanie, vytvorí jej sieť dosiahnuteľnosti, inak vráti informáciu, že vstupná určená workflow sieť nemá korektné správanie. Definovali sme taktiež uviaznutia v sieti dosiahnuteľnosti.

V Časti 2.3 sme dokázali, že značkovania siete dosiahnuteľnosti zachytávajú pre všetky dosiahnuteľné značkovania dynamických miest originálnej určenej workflow siete informáciu o počte inštancií, ktoré sú v danom značkovaní originálnej určenej workflow siete. Značkovania vykonávanej siete teda zodpovedajú značkovaniam siete dosiahnuteľnosti. V Časti 2.3 sme dokázali prvý hlavný výsledok tejto práce - dokázali sme, že značkovanie vykonávanej siete je uviaznutím vykonávanej siete práve vtedy, keď jemu zodpovedajúce značkovanie siete dosiahnuteľnosti je uviaznutím v sieti dosiahnuteľnosti.

V Časti 2.4 sme definovali základné uviaznutia siete dosiahnuteľnosti ako uviaznutia, v ktorých sú označkované spomedzi nestatických miest siete dosiahnuteľnosti iba takzvané kritické miesta. V Časti 2.4 sme dokázali druhý hlavný výsledok práce - dokázali sme, že ak má sieť dosiahnuteľnosti uviaznutie, potom má sieť dosiahnuteľnosti zodpovedajúce základné uviaznutie.

V Časti 2.5 sme dokázali, že kritické miesta sú ohraničené a teda počet základných uviaznutí musí byť pre označkované určené workflow siete s korektným správaním ko-

nečný. Pre ľubovoľné kladné celé číslo sme definovali n-obmedzenú sieť dosiahnuteľnosti ako ohraničenú sieť, ktorá simuluje správanie siete dosiahnuteľnosti pre n inštancií. Ukázali sme dva spôsoby, ako vypočítať horné ohraničenie kritických miest. V Časti 2.5 sme pre označkované určené workflow siete s korektným správaním dokázali tretí hlavný výsledok tejto práce - dokázali sme, že sieť dosiahnuteľnosti má základné uviaznutie práve vtedy, ak má zodpovedajúce základné uviaznutie ohraničená n-obmedzená sieť dosiahnuteľnosti, pričom n je ohraničenie počtu značiek v kritických miestach.

V Časti 2.6 sme zostavili finálny algoritmus na detekciu základných uviaznutí siete dosiahnuteľnosti označkovanej určenej workflow siete s korektným správaním prostredníctvom detekcie základných uviaznutí ohraničenej Petriho siete - teda n-obmedzenej siete dosiahnuteľnosti.

Týmto sme naplnili stanovený cieľ práce. V Kapitole 3 sme sa následne venovali diskusii vzťahu prezentovaných vlastných výsledkov s príbuznými prácami, kategorizovali sme typy dynamickej korektnosti, a zároveň sme načrtli možné smery budúceho výskumu.

Literatúra

[1] Wil MP Van Der Aalst. Three good reasons for using a Petri-net-based workflow management system. In Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC96), pages 179–201. Cambridge, Massachusetts, 1996.

[2] W. M. P. v. d. Aalst. Verification of workflow nets. Lecture Notes in Computer Science, P. Azéma and G. Balbo, Eds., vol. 1248. Springer, 1997, pp. 407–426.

[3] W. M. P. van der Aalst. The application of Petri nets to workflow management. Journal of Circuits, Systems, and Computers, 8(1):21–66, 1998.

[4] W.M.P. van der Aalst and K. van Hee. *Workflow Management, Models Methods and Systems.* The MIT Press, Cambridge, Massachusetts, 2002.

[5] W. van der Aalst, K. van Hee, A. ter Hofstede, N. Sidorova, H. Verbeek, M. Voorhoeve, and M. Wynn. Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects of Computing, 23(3):333–363, 2011.

[6] K. Barkaoui and L. Petrucci, "Structural Analysis of Workflow Nets with Shared Resources," in WFM 1998, 1998, pp. 82–95.

[7] K. Barkaoui, R. Benayed, and Z. Sbai, Workflow Soundness Verification Based on Structure Theory of Petri Nets, International Journal of Computing & Information Sciences, vol. 5, no. 1, pp. 51–62, 2007.

[8] S. Balemi, P. Kozák, and R. Smedinga, editors. Discrete event systems: Modelling and Control. Birkhäuser, Basel, 1993.

[9] Bergenthum, Robin, Jörg Desel, and Sebastian Mauser. Workflow Nets with Roles. EMISA. 2011.

[10] Bergenthum, Robin, et al. Modeling and mining of learnflows. Transactions on Petri Nets and Other Models of Concurrency V. Springer Berlin Heidelberg, 2012. 22-50.

[11] L. Bernardinello and F. De Cindio. A survey of basic net models and modular net classes. In Advances in Petri Nets 1992, volume 609 of Lecture Notes in Computer Science, pages 304–351. Springer-Verlag, Berlin, 1992.

[12] E. Best, J. Esparza, H. Wimmel, and K. Wolf, Separability in conflict-free petri nets, in PSI 2006, ser. Lecture Notes in Computer Science, I. Virbitskaite and A. Voronkov, Eds., vol. 4378. Springer, 2007, pp. 1–18.

[13] E. Best and P. Darondeau, "Separability in persistent petri nets," Fundam. Inform., vol. 113, no. 3-4, pp. 179–203, 2011.

[14] Billington, Jonathan, Michel Diaz, and Grzegorz Rozenberg. Application of Petri nets to communication networks: advances in Petri nets. No. 1605. Springer Science & Business Media, 1999.

[15] Birch, Sine Viesmose, and Christoffer Moesgaard. Compositional Analysis of Timed-arc Resource Workflows with Communication. University of Aalborg, 2015.

[16] C. G. Cassandras and S. Lafortune. Introduction to Discrete Event Systems. Kluwer, 1999.

[17] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Hardware and Petri Nets: Application to Asynchronous Circuit Design. In: Nielsen, M., Simpson, D. (eds.) ICATPN 2000. LNCS, vol. 1825, pp. 1–15. Springer, Heidelberg (2000)

[18] Cortés, Mateo, Verification and Validation of Web Services Compositions Using Wormal Methods, PhD thesis, University of Castilla-La Mancha (2014).

[19] R. David and H. Alla. Continuous Petri nets. In Proceedings of 8th European Workshop on Application and Theory of Petri nets, pages 275–294, Zaragoza, 1987.

[20] R. David and H. Alla. Autonomous and timed continuous Petri nets. In Proceedings of 11th International Conference on Application and Theory of Petri nets, pages 367–386, Paris, 1990.

[21] R. David and H. Alla. Petri nets for modelling of dynamic systems-a survey. Automatica, 30(2):175–202, 1994.

[22] Isabel Demongodin and Nick T. Koussoulas. Differential Petri nets: Representing continuous systems in a discrete-event world. IEEE Tran. on Automatic Control, 43(4):573–579, 1998. Special issue on hybrid control systems.

[23] J. Desel and W.Reisig. Place/Transition Petri Nets. In Lectures on Petri nets I: Basic Models, LNCS 1491, pp. 123–174,1998.

[24] J. Desel and G. Juhás. What is a Petri Net? In H. Ehrig, G. Juhás, J. Padberg, G. Rozenberg (Eds.): *Unifying Petri Nets*, LNCS 2128, Springer, pp. 1–25, 2001.

[25] R. Devillers. The Semantics of Capacities in P/T Nets. In Advances in Petri Nets 1989, LNCS 424, pp. 128–150,1990.

[26] Egri-Nagy, Attila, and Chrystopher L. Nehaniv. Algebraic properties of automata associated to Petri nets and applications to computation in biological systems. BioSystems 94.1 (2008): 135-144.

[27] D. F. Escrig and C. Johnen, Decidability of home space property, Université Paris-Sud, LRI report 503, 1989.

[28] J. Esparza and M. Nielsen. Decidability issues for Petri nets–a survey. J. Inform. Process. Cybernet. 30 (3), pp. 143-160, 1994.

[29] Diego Figueira, Santiago Figueira, Sylvain Schmitz and Phillipe Schnoebelen, Ackermannian and primitive-recursive bounds for Dickson's lemma, 26th Annual IEEE Symposium on Logic in Computer Science LICS 2011, IEEE Computer Society, Los Alamitos, CA, 269-278.

[30] K. M. v. Hee, N. Sidorova, and M. Voorhoeve, Soundness and separability of workflow nets in the stepwise refinement approach, in ICATPN 2003, ser. Lecture

Notes in Computer Science, W. M. P. v. d. Aalst and E. Best, Eds., vol. 2679. Springer, 2003, pp. 337–356.

[31] K. van Hee, N. Sidorova, and M. Voorhoeve. Generalised soundness of workflow nets is decidable. In J. Cortadella and W. Reisig, editors, Applications and Theory of Petri Nets 2004, volume 3099 of Lecture Notes in Computer Science, pages 197–215. Springer Berlin Heidelberg, 2004.

[32] K. M. van Hee, A. Serebrenik, N. Sidorova and M. Voorhoeve. Soundness of Resource-Constrained Workflow Nets. Lecture Notes in Computer Science 3536, Springer, pp. 250-267, 2005.

[33] K. M. v. Hee, N. Sidorova, and M. Voorhoeve, Resource-constrained workflow nets, Fundam. Inform., vol. 71, no. 2-3, pp. 243–257, 2006.

[34] L. E. Holloway and B. H. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. IEEE Tran. on Automatic Control, 35(5):514–523, 1990.

[35] L. E. Holloway, B. H. Krogh, and A. Giua. A survey of net methods for controlled discrete event systems. Discrete Event Dynamic Systems: Theory and Applications, 7:151–190, 1997.

[36] Matthias Jantzen. Complexity of place/transition nets. In Proc. 2nd Advanced Course in Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, volume 254 of Lecture Notes in Computer Science, pages 60–94, Bad Honnef, 1987. Springer-Verlag, Berlin.

[37] K. Jensen. Coloured Petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986, volume 254 of Lecture Notes in Computer Science, pages 248–299. Springer, 1986.

[38] K. Jensen and G. Rozenberg, editors. High-Level Petri-Nets, Theory and Applications. Springer-Verlag, Berlin, 1991.

[39] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use I II III.* Springer, 1997.

[40] K. Jensen, L. M. Kristensen, and L. Wells. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. STTT, 9(3-4):213– 254, 2007.

[41] Gabriel Juhás, Igor Kazlov and Ana Juhásová: Instance deadlock: A mystery behind frozen programs. In Application and Theory of Petri Nets and Other Models of Concurrency. LNCS 6128, pp. 1-17, Springer-Verlag, 2010. ISSN 0302-9743.

[42] Juhásová, Ana - Janov, Vladimír - Juhás, Gabriel: Dynamický deadlock workflow procesov. Časopis pre elektrotechniku a energetiku. 15 (2009), pp. 205-208.

[43] Juhásová, Ana; Juhás, Gabriel: Soundess of resource constrained workflow nets is decidable. In: Petri Net Newsletter. - ISSN 0931-1084. - Vol. 76, November 2009, p. 3-6. ISSN 0931-1084.

[44] Richard M. Karp and Raymond E. Miller. Parallel program schemata. Journal of Computer and System Sciences, 3(2):147–195, May 1969.

[45] R. M. Keller. Formal verification of parallel programs. Communications of the ACM, 7(19):371–384, 1976.

[46] S. R. Kosaraju. Decidability of reachability in vector addition systems. In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC '82, pages 267–281, New York, NY, USA, 1982. ACM.

[47] D. Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching. Addisson-Wesley, Reading, MA, 1973.

[48] J. Lambert. A structure to decide reachability in Petri nets. Theoretical Computer Science, 99(1):79–104, 1992.

[49] J. Le Bail, H. Alla, and R. David. Hybrid Petri nets. In Proceedings of 1st European Control Conference, pages 1472–1477, Grenoble, 1991.

[50] J. Le Bail, H. Alla, and R. David. Asymptotic continuous Petri nets. Discrete Event Dynamic Systems: Theory and Applications, 2:235–263, 1993.

[51] J. Leroux. Vector addition system reachability problem: A short selfcontained proof. SIGPLAN Not., 46(1):307–316, Jan. 2011.

[52] R. Lipton. The reachability problem is exponential-space hard. Technical Report 62, Department of Computer Science, Yale University, January 1976.

[53] Martiník, Ivo. Automation of Presentation Record Production Based on Rich-Media Technology Using SNT Petri Nets Theory. The Scientific World Journal 2015 (2015).

[54] Martos-Salgado, María, and Fernando Rosa-Velardo. Dynamic soundness in resource-constrained workflow nets. Formal Techniques for Distributed Systems. Springer Berlin Heidelberg, 2011. 259-273.

[55] Martos-Salgado, María, and Fernando Rosa-Velardo. Cost soundness for priced resource-constrained workflow nets. International Conference on Application and Theory of Petri Nets and Concurrency. Springer Berlin Heidelberg, 2012.

[56] Martos-Salgado, María, and Fernando Rosa-Velardo. Safety and Soundness for Priced Resource-Constrained Workflow Nets. Fundamenta Informaticae 131.1 (2014): 55-80.

[57] Martos Salgado, María Rosa. Verification of priced and timed extensions of Petri Nets with multiple instances. PhD thesis, Complutense University of Madrid (2016).

[58] Ernst W. Mayr. Persistence of vector replacement systems is decidable. Acta Informatica, 15:309–318, 1981.

[59] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. SIAM Journal on Computing, 13:441–460, 1984.

[60] J. Medling and W.M.P. van der Aalst. Errors in the SAP Reference Models. BP-Trends. June 2006.

[61] Moore, Jason H., and Lance W. Hahn. Systems biology modeling in human genetics using Petri nets and grammatical evolution. Genetic and Evolutionary Computation Conference. Springer Berlin Heidelberg, 2004, 392-401.

[62] T. Murata. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4):541–588, 1989.

[63] J. L. Peterson. Petri nets. ACM Comput. Surv., 9(3):223–252, Sept. 1977.

[64] C. A. Petri. Kommunikation mit Automaten. PhD thesis, Institut fur Instrumentelle Mathematik, Bonn, 1962.

[65] C. Rackoff. The covering and boundedness problems for vector addition systems. Theoretical Computer Science, 6(2):223–231, 1978.

[66] Ramezani, Elham, Natalia Sidorova, and Christian Stahl. Interval soundness of resource-constrained workflow nets: decidability and repair. International Conference on Fundamentals of Software Engineering. Springer Berlin Heidelberg, 2013. 150-167.

[67] W. Reisig. Primer in Petri Net Design. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1992.

[68] Rosa-Velardo, Fernando, and David de Frutos-Escrig. Decidability and complexity of Petri nets with unordered data. Theoretical Computer Science 412.34 (2011): 4439-4451.

[69] L. E. Rosier and H.-C. Yen. A multiparameter analysis of the boundedness problem for vector addition systems. Journal of Computer and System Sciences, 32(1):105–135, 1986.

[70] Natalia Sidorova, Christian Stahl, and Nikola Trčka. Workflow soundness revisited: Checking correctness in the presence of data while staying conceptual. In Advanced Information Systems Engineering, pages 530–544. Springer, 2010.

[71] Sidorova, Natalia, and Christian Stahl. Soundness for resource-constrained workflow nets is decidable. Systems, Man, and Cybernetics: Systems, IEEE Transactions on 43.3 (2013): 724-729.

[72] Schrijver, A.: Theory of linear and integer programming. Wiley, Chichester (1986).

[73] Solé, Marc, and Josep Carmona. Light region-based techniques for process discovery. Fundamenta Informaticae 113.3-4 (2011): 343-376.

[74] F. L. Tiplea and C. Bocaneala, Decidability results for soundness criteria of resource-constrained workflow nets, IEEE Transactions on Systems, Man, and Cybernetics, Part A, vol. 42, no. 1, pp. 238–249, 2012.

[75] Velardo, Fernando Rosa. Liveness properties for Petri Net extensions with names AREA: Verification of Infinite State Systems. Fernando Rosa Velardo. Complutense University of Madrid (2011).

[76] Wang, Jiacun. Petri net based resource modeling and analysis of workflows with task failures. Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on. IEEE, 2013.

[77] Wang, Jiacun, and Demin Li. Resource oriented workflow nets and workflow resource requirement analysis. International Journal of Software Engineering and Knowledge Engineering 23.05 (2013): 677-693.

[78] Wang, Jiacun, Bill Tepfenhart, and Xiaoou Li. Analysis of Minimum Workflow Resource Requirement. International Workshop on Process-Aware Systems. Springer Singapore, 2015.

[79] Wang, Jiacun, Xiaoou Li, and Gaiyun Liu. Cyclic workflow resource requirement analysis and application in healthcare. 2016 13th International Workshop on Discrete Event Systems (WODES). IEEE, 2016. 291-297.

[80] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, Handbook of Logic in Computer Science, volume 4, pages 1–148. Oxford University Press, 1995. Also appeared in BRICS Report Series RS-94-12.

[81] Li, Xiaoou, et al. Resource requirement analysis for cyclic workflows. 2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC). IEEE, 2016.

[82] M.C. Zhou and F. Di Cesare. Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Kluwer, 1993.

Register