SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

Ing. Ana Juhásová

Autoreferát dizertačnej práce

DETEKCIA UVIAZNUTÍ V DISKRÉTNYCH UDALOSTNÝCH SYSTÉMOCH SO ZDIEĽANÝMI ZDROJMI

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe: **Automatizácia a riadenie**
v študijnom odbore 5.2.14 automatizácia

Miesto a dátum: Bratislava, 12.09.2016

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Ing. Ana Juhásová

Autoreferát dizertačnej práce

DETEKCIA UVIAZNUTÍ V DISKRÉTNYCH UDALOSTNÝCH SYSTÉMOCH SO ZDIEĽANÝMI ZDROJMI

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe:
Automatizácia a riadenie

Miesto a dátum: Bratislava, 12.09.2016

Dizertačná práca bola vypracovaná v externej forme doktorandského štúdia

Na Ústav robotiky a kybernetiky, Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19 Bratislava

Predkladateľ: Ing. Ana Juhásová, Ústav robotiky a kybernetiky, Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19 Bratislava

Školiteľ: prof. Ing. Ján Murgaš, PhD., Ústav robotiky a kybernetiky, Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19 Bratislava

Oponenti: doc. Ing. Zoltán Balogh, PhD., Katedra informatiky, Fakulta prírodných vied, Univerzita Konštantína Filozofa v Nitre
doc. Ing. Peter Trebuňa, PhD., Katedra priemyselného inžinierstva a manažmentu, Strojnícka fakulta, Technická univerzita Košice

Autoreferát bol rozoslaný: 13.09.2016

Obhajoba dizertačnej práce sa koná: _____ o _____h.

Na Ústave robotiky a kybernetiky, Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave, Ilkovičova 3, 812 19 Bratislava

prof. Dr. Ing. Miloš Oravec dekan FEI STU

Abstrakt

Práca "Detekcia uviaznutí v diskrétnych udalostných systémoch so zdieľanými zdrojmi" A. Juhásovej prezentuje metódu na detekciu uviaznutí (zamrznutí a zacyklení) v diskrétnych udalostných systémoch so zdieľanými zdrojmi a viacerými inštanciami. Ako modelovací formalizmus boli zvolené workflow siete, ktoré predstavujú rozšírený nástroj na modelovanie a analýzu workflow procesov. Vo workflow sieťach sa vykonávajú inštancie s určeným počiatočným stavom a korektným ukončením. Zaoberáme sa workflow procesmi, v ktorých inštancie môžu zdieľať zdroje viacerých typov, pričom inštancie zdieľané zdroje nevytvárajú ani nespotrebujú. To znamená, že inštancie môžu zdroje používať a najneskôr pri ukončení uvoľnia všetky používané zdroje. Takéto zdieľané zdroje sa nazývajú trvácne. Príkladom trvácnych zdrojov sú napríklad zdroje v informačných systémoch ako pamäť, procesory alebo pracovníci v rolách v organizačnej štruktúre. Zaoberáme sa procesmi, ktoré obsahujú dostatok zdrojov na korektné vykonanie jednej inštancie, hovoríme, že takéto procesy majú korektné správanie. Uviaznutím nazývame taký stav procesu, v ktorom sa vykonávajú viaceré inštancie a z ktorého nie je možné všetky vykonávané inštancie korektne ukončiť z dôvodu nedostatku zdieľaných zdrojov. Hlavným výsledkom práce je návrh metódy a algoritmu na detekciu uviaznutí workflow procesov s korektným správaním a viacerými typmi zdieľaných trvácnych zdrojov pre daný počiatočný počet zdrojov a ľubovoľný počet vykonávaných inštancií.

Kľúčové slová: Diskrétne udalostné systémy, workflow procesy, zdieľané zdroje, inštancie, Petriho siete, workflow siete, uviaznutia, zamrznutia, zacyklenia, korektnosť

Abstract

The thesis "Detection of livelocks and deadlocks in discrete event systems with shared resources" by A. Juhásová is presenting a method for detection of locks (both deadlocks and livelocks) in discrete event systems with shared resources and multiple instances. As a modelling framework we have chosen workflow nets, which represent a widely spread tool for modelling and analysis of workflow processes. We consider that multiple instances with determined initial state and a correct final state are running in workflow nets. We consider workflow processes, in which instances can share several types of resources, with instances neither creating nor destroying shared resources. It means, that instances can use resources but the used resources are returned at the latest by the correct finish of the instance. Such resources are said to be durable. Examples of durable resources include resources of information systems, such as memory and processors or employees in roles in an organizational structure. We consider processes, which have enough resources to execute a single instance, such processes are said to be sound. A lock is a state of the process, where several instances are running but because of the lack of shared resources not all running instances can finish properly. The main result of the thesis is the method and the algorithm for detection of locks in sound workflow processes with several types of shared durable resources for given initial number of resources and an arbitrary number of running instances.

Keywords: Discrete event systems, workflow processes, shared resources, instances, Petri nets, workflow nets, locks, deadlocks, livelocks, soundness

Obsah

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. 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. Ako prvý hlavný problém budúceho výskumu autori práce [32] uvádzajú rozšírenie problému na viac typov zdrojov.

V tejto práci riešime problém inšpirovaný prácou [32], pričom uvažujeme viac typov zdrojov. Ako ukazujeme 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. 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 nazývame uviaznutím. Uviaznutia predstavujú vážny problém pri aplikácii workflow procesov v praxi [60]. 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 tr-

vácnych zdieľaných zdrojov pre konkrétny vopred daný počet zdrojov rozhodne, či workflow proces pre nejaký počet inštancií obsahuje zamrznutie. Táto metóda však neidentifikovala zacyklenia.

Hlavným cieľom dizertačnej 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.

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 práci sa venujeme 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. Takýto model LDUS je možné matematicky formalizovať niekoľkými spôsobmi, najznámejšie sú označené prechodové systémy a automaty [16]. V práci používame ako základný model LDUS označené prechodové systémy [45, 80].

Definícia 1 (Označený prechodový systém)
Označený prechodový systém je usporiadaná trojica $(S, E, \longrightarrow)$*, kde* S *je množina stavov,* E *je množina udalostí a* $\longrightarrow \subseteq S \times E \times S$ *je prechodová relácia. Skutočnosť, že* (s, e, s') *patrí do* $\longrightarrow$ *označujeme ako* $s \xrightarrow{e} s'$*.*

Definícia 2 (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 práci uvažujeme LDUS, modelované Petriho sieťami [44, 62, 63, 67, 23, 24]. Petriho siete sú pomenované podľa Carla Adama Petriho [64].

Definícia 3 (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 je 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 4 (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 označujeme 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 $m(p) > 0$.

Dynamika Petriho siete je daná spúšťaním prechodov.

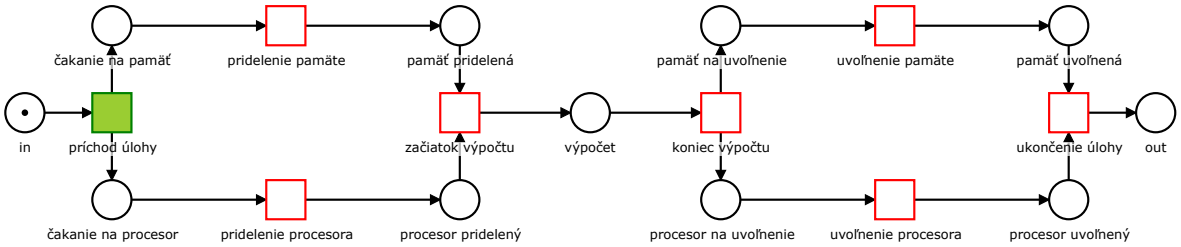
Definícia 5 (Spustiteľnosť prechodov)
Nech $PN = (P, T, I, O)$ je Petriho sieť. Nech $m : P \to \mathbb{N}$ je značkovanie a $t \in T$ je prechod siete PN. Prechod t je spustiteľný v značkovaní m práve vtedy, keď pre každé miesto $p \in P$ platí, že počet značiek $m(p)$ v mieste $p \in P$ nie je menší 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: $m'(p) = m(p) - I(p, t) + O(p, t)$.

Definícia 6 (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 využívame jednoduchý ilustratívny príklad popisujúci 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.

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 jazykoch JavaScript HTML5, grafické znázornenie je realizované prostredníctvom SVG komponentov. Editor umožňuje modelovať Petriho siete, ukladať a načítavať ich vo formáte XML a exportovať ich vo formáte SVG.

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



Obr. 1: Označkovaná Petriho sieť modelujúca spracovanie výpočtovej úlohy

Definícia 7 (Graf dosiahnuteľnosti Petriho siete)
Nech $PN = (P, T, I, O)$ je Petriho sieť. Označený prechodový systém $(S, E, \longrightarrow)$, kde 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.

Definícia 8 (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.

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 nazývame LDUS s inštanciami. Zameriame sa na systémy, kde inštancie majú jednoznačný začiatok a korektné ukončenie. V literatúre sú takéto systémy modelované pomocou podtriedy Petriho sietí - nazývanej workflow siete [1, 2, 3, 4]. Jednoznačný začiatok vo workflow sieťach je vyjadrený pomocou špeciálneho miesta. Podobne ukončenie inštancie je vyjadrené pomocou špeciálneho miesta.

Definícia 9 (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.

Definícia 10 (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.*

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. Týmto stavom musí byť značkovanie *out*, teda značkovanie, pri ktorom miesto *out* obsahuje práve jednu značku a žiadne iné miesto značky neobsahuje. Inými slovami, korektné ukončenie inštancie označkovanej workflow siete je vyjadrené presunom značky zo vstupného do výstupného miesta. Takúto vlastnosť nazývame korektné správanie (v literatúre sa táto vlastnosť označuje soundness) [4].

Definícia 11 (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: 1. značkovanie out je dosiahnuteľné zo značkovania* m*; 2. 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.*

Lema 1 *Ak označkovaná workflow sieť má korektné správanie, potom počet stavov dosiahnuteľných z počiatočného značkovania je konečný.*

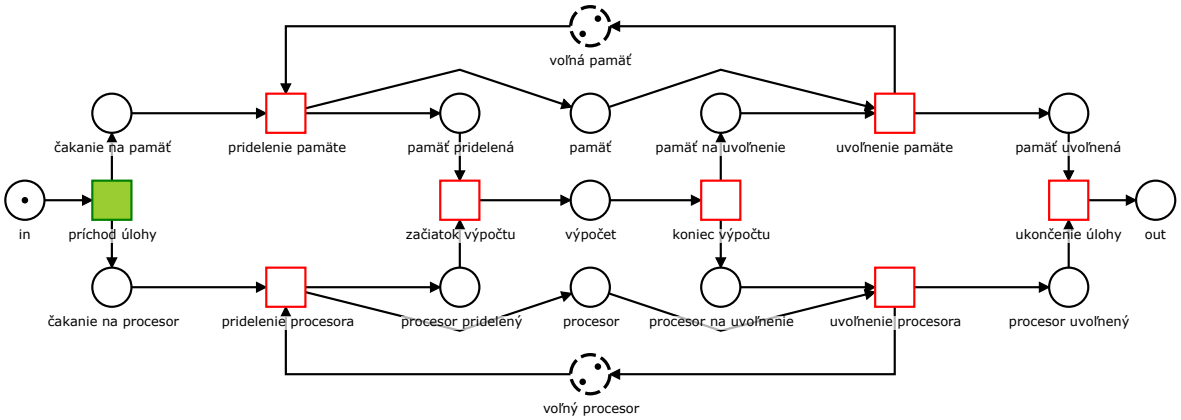
Príkladom označkovanej workflow siete s korektným správaním je sieť na Obr. 1.

1.4 Petriho siete a workflow siete so statickými miestami

Uvažujeme, že inštancie môžu zdieľať zdroje, napr. operačnú pamäť či procesor v prípade programov. Takéto LDUS s inštanciami nazývame 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 nazývame 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.

Formálne modelujeme LDUS s inštanciami a zdieľanými zdrojmi pomocou Petriho sietí so statickými miestami [41, 42, 43]. Sú to Petriho siete s pridanými statickými 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 12 (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.*



Obr. 2: Označkovaná určená workflow sieť so statickými miestami, ktorých označkovanie modeluje počet pamäťových jednotiek a procesorov k dispozícii

Definícia 13 (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 *analogicky používame pre označkovanú Petriho sieť so statickými miestami* $MPNS$.

Definícia 14 (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* $out \in D$.

Definícia 15 (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.

Definícia 16 (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.

Statické miesta sú 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. 2. Miesto *pamäť* je určujúcim miestom pre statické miesto *voľná pamäť*, podobne miesto *procesor* je určujúce miesto pre statické miesto *voľný procesor*.

Definícia 17 (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 18 (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: 1. existuje finálne značkovanie m_f siete MW, ktoré je dosiahnuteľné zo značkovania m; 2. ak $m(out) \geq 1$ potom m je finálne značkovanie siete MW.

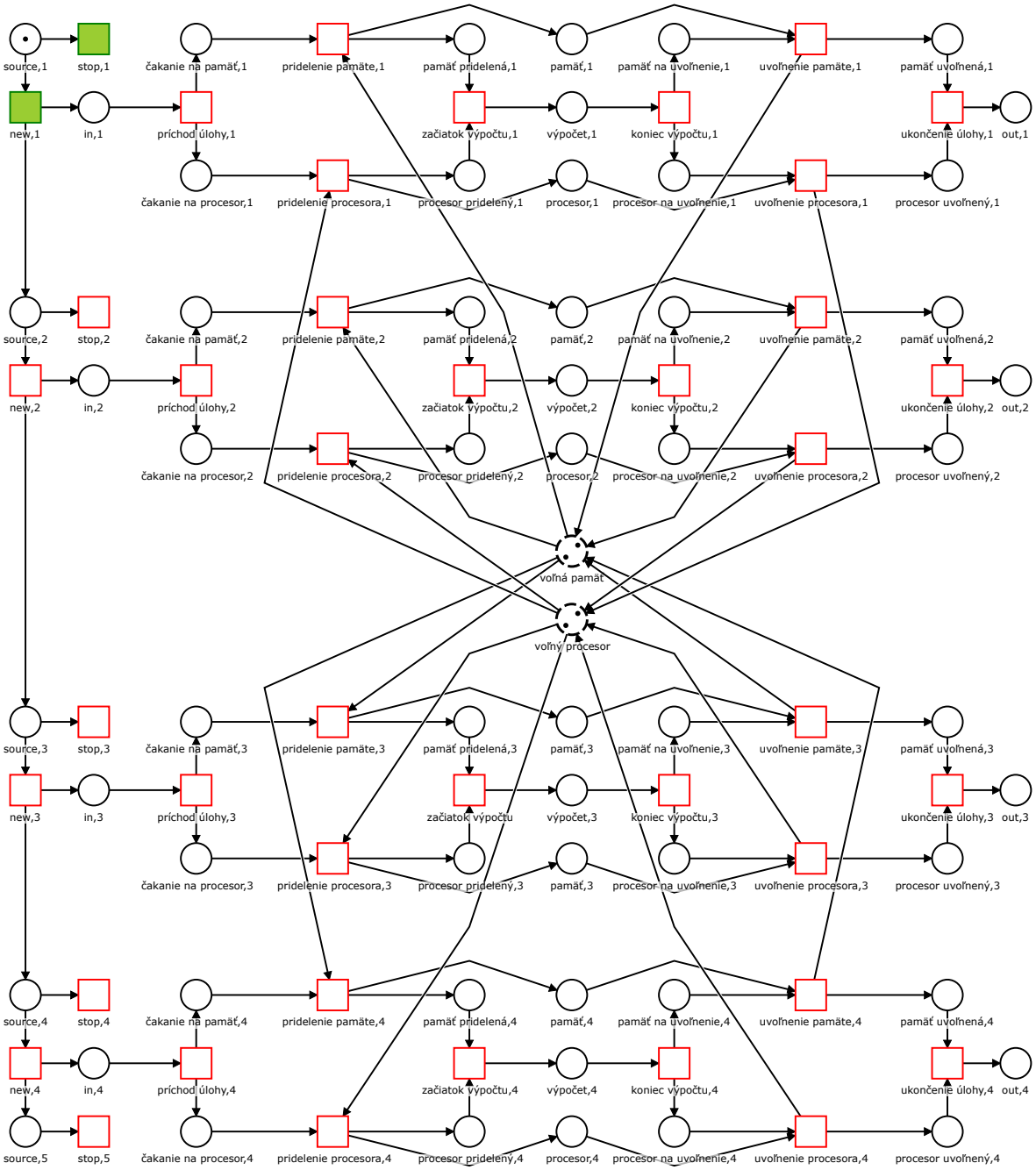
Existencia určujúcich miest v určenej workflow sieti navyše zabezpečí, že nové zdieľané zdroje sa nevytvárajú ani nespotrebujú, teda po ukončení inštancie sa počet zdrojov rovná počtu zdrojov danému značkovaním statických miest v počiatočnom značkovaní.

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

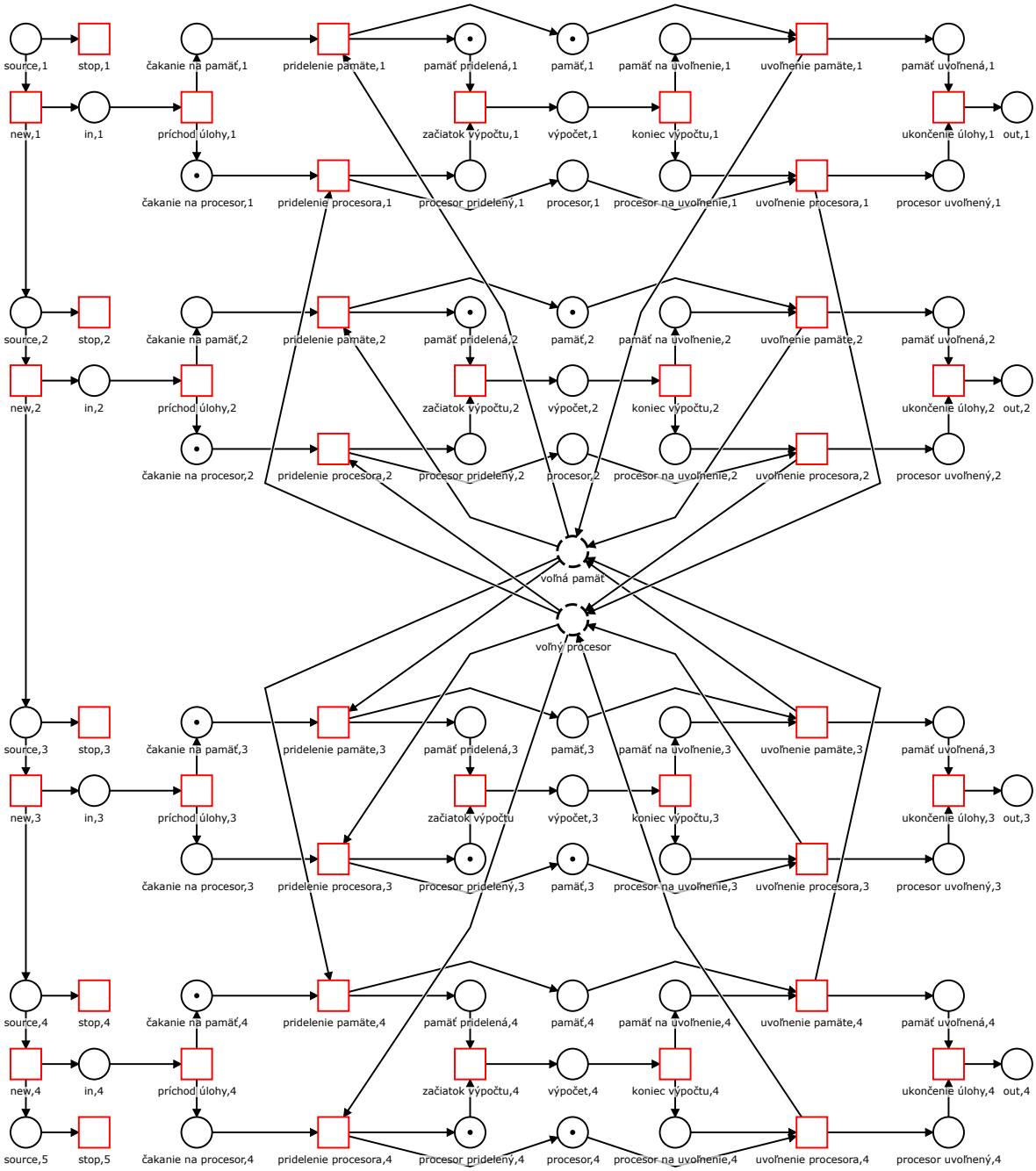
Je zrejmé, že označkovaná určená workflow sieť so statickými miestami z Obr. 2 má korektné správanie.

1.5 Vykonávané siete so statickými miestami a inštanciami

Vykonávaná workflow sieť so statickými miestami reprezentuje bežiaci workflow proces s inštanciami. Inštancie sú reprezentované kópiami pôvodnej workflow siete, ktoré sú 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.



Obr. 3: Časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie



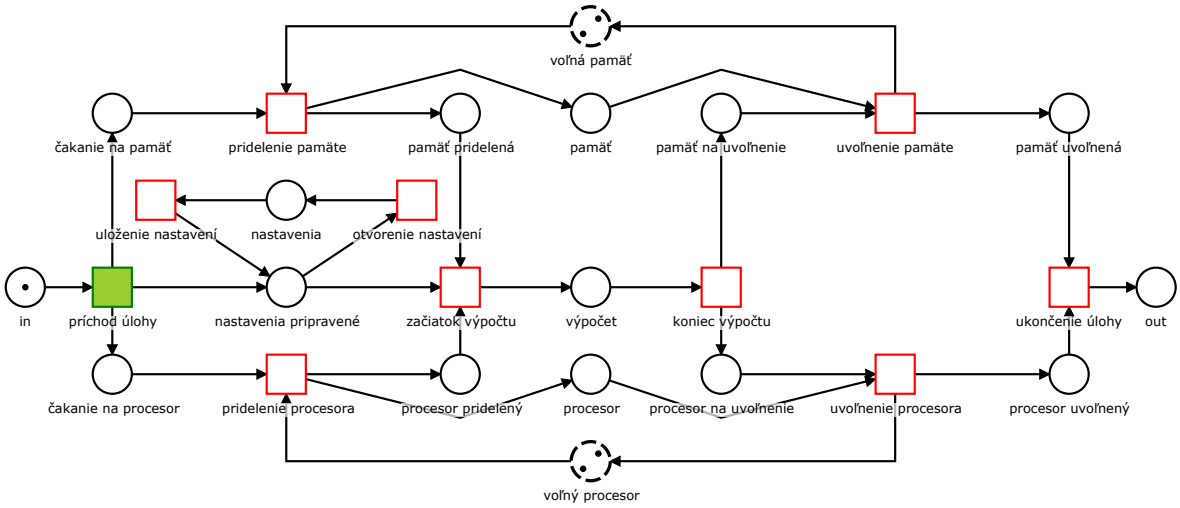
Obr. 4: Časť uviaznutej vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie v stave zamrznutia

Definícia 19 (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*
- $I^\infty((d, i), (t, i)) = I(d, t)$ *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 označujeme P_i^∞, *teda* $P_i^\infty = (D \cup \{source\}) \times \{i\}$ *pre kladné celé číslo i.*
Prechody i-tej inštancie označujeme T_i^∞, *teda* $T_i^\infty = (T \cup \{new, stop\}) \times \{i\}$ *pre kladné celé číslo i.*

Na Obr. 3 je znázornená časť vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie workflow procesu z Obr. 2, 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. 5: Výpočtová úloha s možnosťou zmeny nastavenia parametrov výpočtu

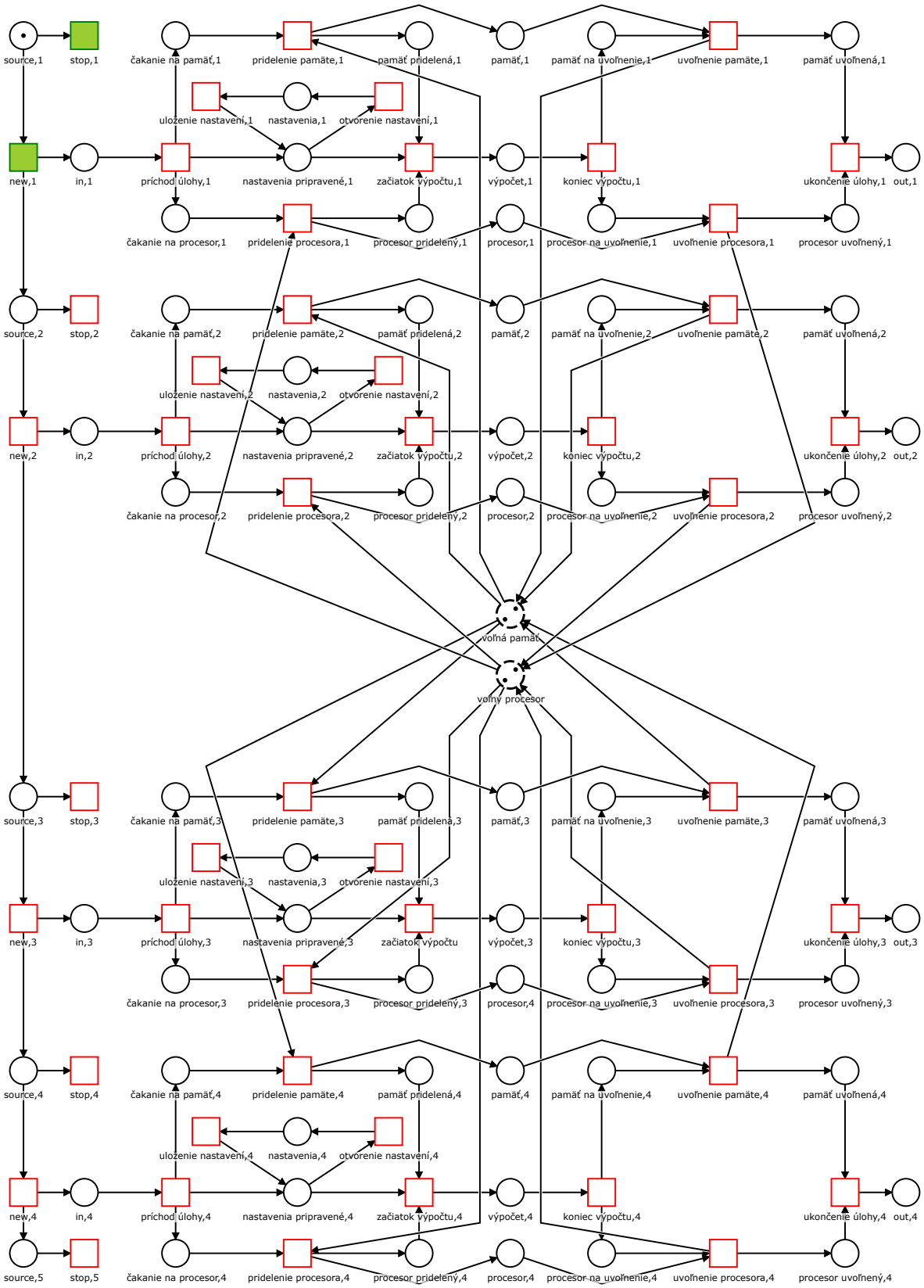
1.6 Uviaznutia vo vykonávaných sieťach so statickými miestami a inštanciami

Zamrznutie

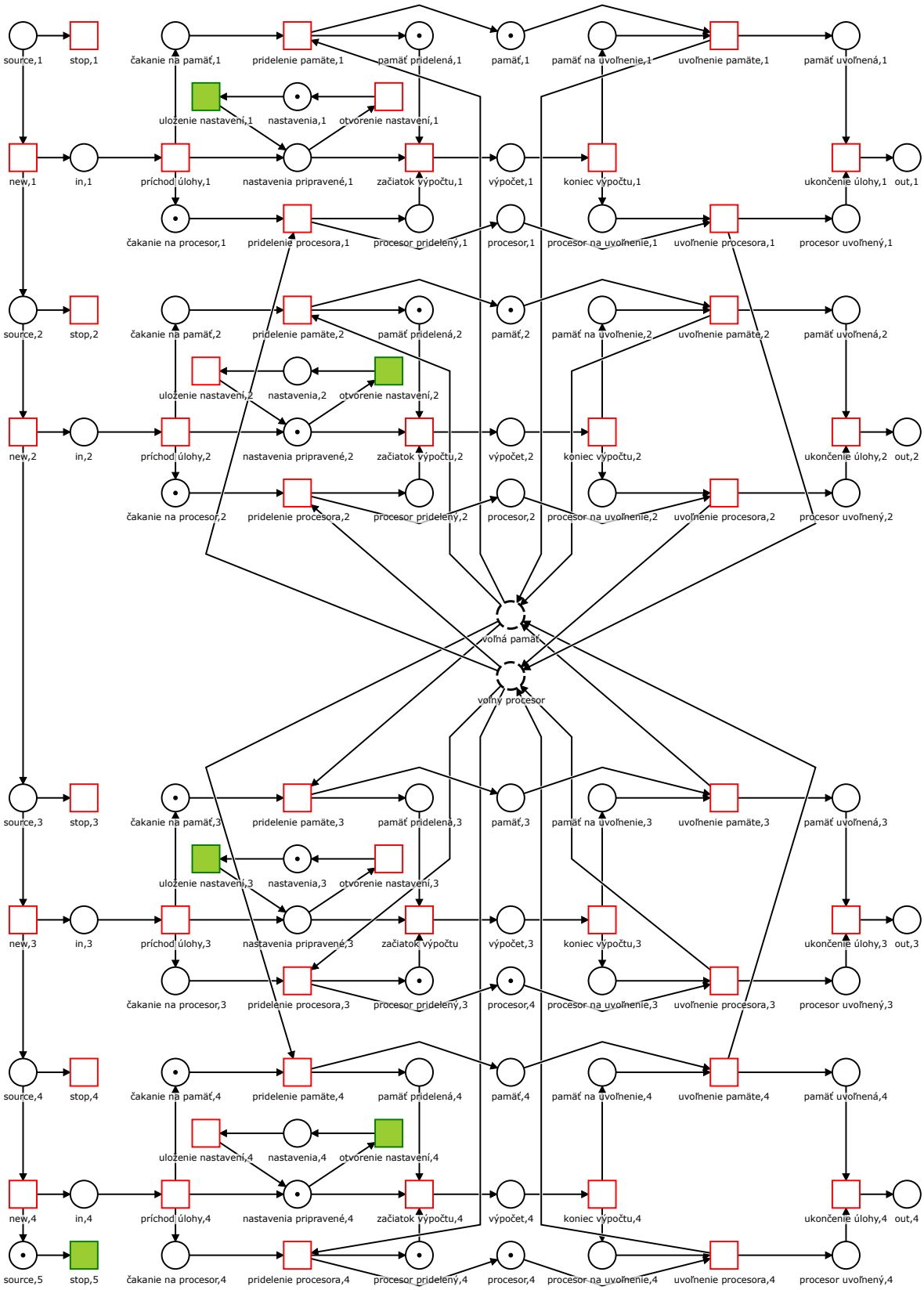
Počiatočné značkovanie vykonávanej siete znázornené na Obr. 3 vyjadruje stav systému s dvoma jednotkami pamäte a dvoma procesormi. V dosiahnuteľnom značkovaní na Obr. 4 nie sú spustiteľné žiadne prechody a systém zamrzne. Značkovanie vykonávanej siete, z ktorého jednotlivé inštancie workflow siete so 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. 4, 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 zacyklením (anglicky livelock).

Zacyklenie

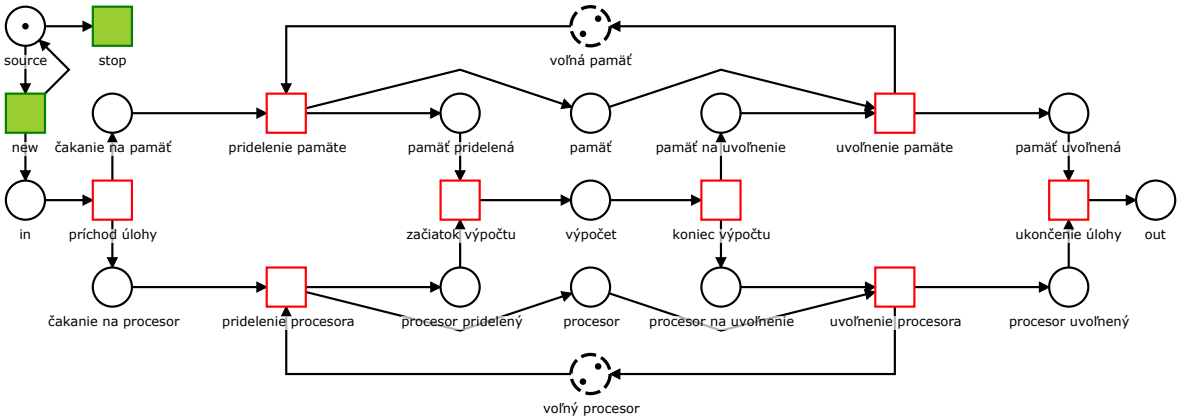
Predstavme si situáciu, modelovanú určenou workflow sieťou so statickými miestami a s korektným správaním na Obr. 5: po príchode výpočtovej úlohy sa načítajú prednastavené parametre výpočtu, čo je vyjadrené značkou v mieste *nastavenia pripravené* (Obr. **??**). V tomto stave je možné spustiť prechod *otvorenie nastavení*, pričom sa otvorí dialógové okno s nastaveniami, teda vytvorí sa značka v mieste *nastavenia* (Obr. **??**). 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. **??**), môže sa spustiť samotný výpočet, čo je modelované prechodom *začiatok výpočtu*.



Obr. 6: Časť vykonávanej workflow siete so statickými miestami pre proces na Obr. 5 pre prvé 4 inštancie



Obr. 7: Časť uviaznutej vykonávanej workflow siete so statickými miestami pre prvé 4 inštancie v stave zacyklenia



Obr. 8: Sieť s konštruktorom workflow siete z Obr. 2.

Opätovne, označkovaná určená workflow sieť so statickými miestami na Obr. 5 má korektné správanie.

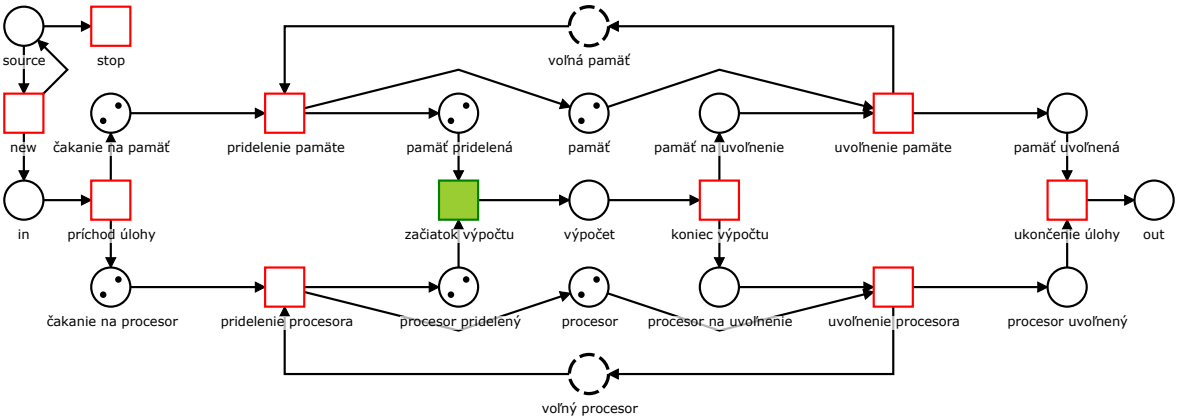
Na Obr. 6 je znázornená časť vykonávanej workflow siete so statickými miestami pre proces na Obr. 5 pre prvé 4 inštancie. Napriek tomu, že v dosiahnuteľnom značkovaní na značkovaní na Obr. 7 systém nezamrzol, nie je možné dokončiť žiadnu zo 4 výpočtových úloh. Toto uviaznutie nazývame zacyklením.

1.7 Uviaznutie

Definícia 20 (Finálne značkovania vykonávanej siete)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná workflow sieť so statickými miestami 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_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* $(\{out\} \times \mathbb{Z}) \cup S$.

Definícia 21 (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 Kapitoly 2 dizertačnej práce.



Obr. 9: Sieť s konštruktorom v značkovaní, ktoré zodpovedá uviaznutiu vykonávanej siete na Obr. 4

2 Detekcia uviaznutí v diskrétnych udalostných systémoch so zdieľanými zdrojmi

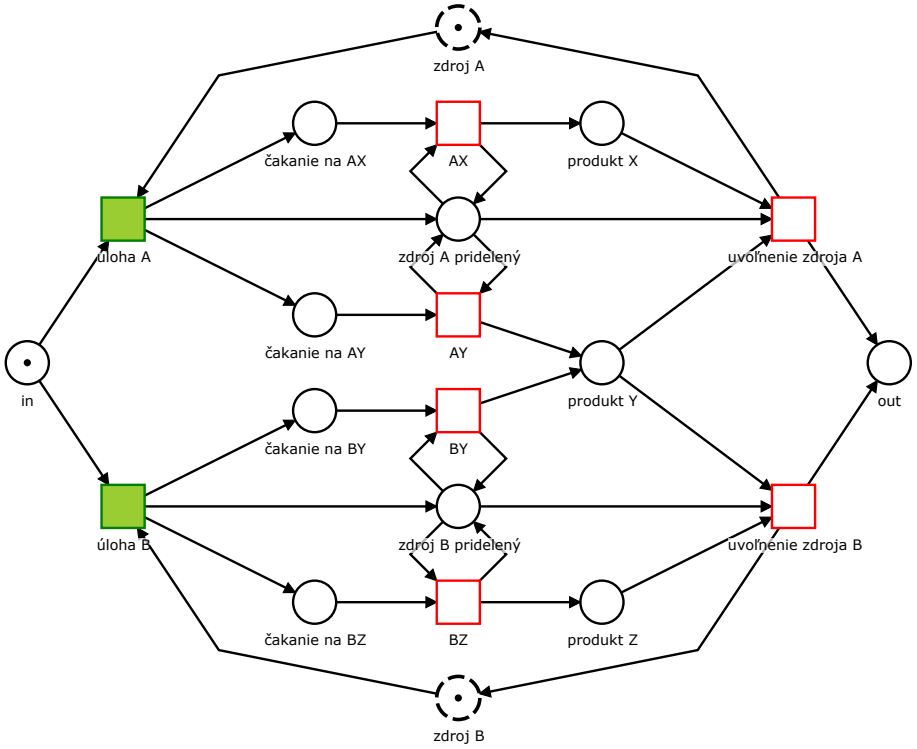
Prvým krokom pri návrhu metódy je nájsť pre určenú workflow sieť so statickými miestami a korektným správaním takú Petriho sieť s konečným počtom miest a konečným počtom prechodov a definovať jej finálne značkovania a uviaznutia tak, že táto sieť má uviaznutia práve vtedy keď vykonávaná sieť. Druhým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

2.1 Siete s konštruktorom versus vykonávané siete

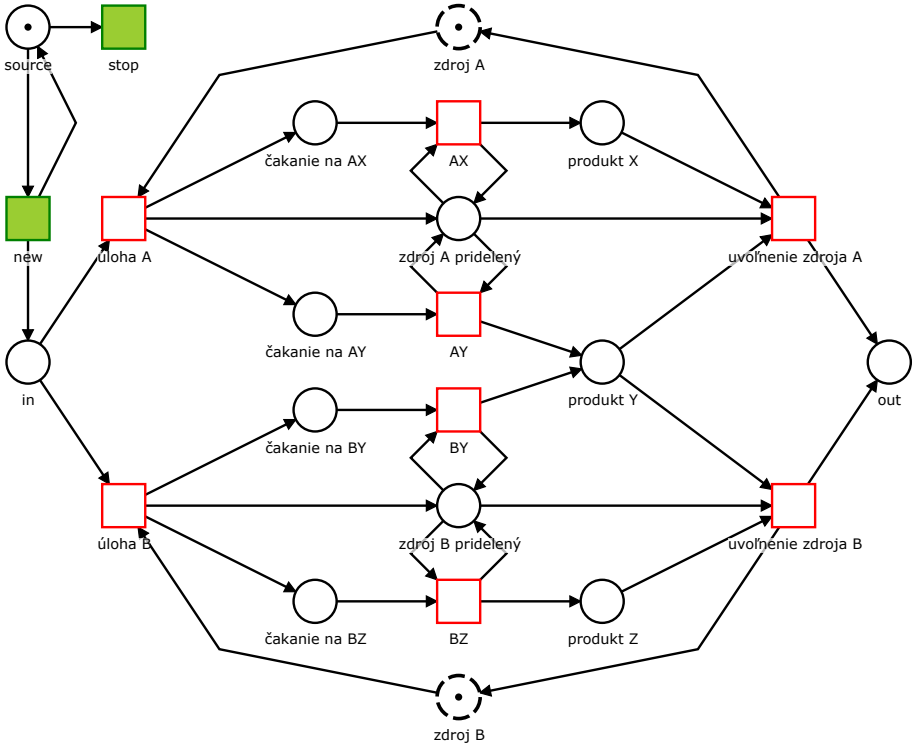
Ako prvý kandidát je v práci použitá pôvodná workflow sieť, obohatená o konštruktor.

Na Obr. 8 je sieť s konštruktorom workflow siete z Obr. 2. Žiaľ, sieť s konštruktorom na Obr. 8 nemá žiadne uviaznutia, na rozdiel od vykonávanej siete na Obr. 3, ktorej uviaznutie je znázornené na Obr. 4. Tomuto uviaznutiu zodpovedá značkovanie siete s konštruktorom na Obr. 9. Značkovanie siete s konštruktorom na Obr. 9 však nie je uviaznutím, pretože je spustiteľný prechod *začiatok výpočtu*. Je to preto, že sieť s konštruktorom nevie rozoznať, ktorá značka patrí ktorej inštancii. Predchádzajúci príklad ilustruje, že sieť s konštruktorom nerozozná existujúce uviaznutie, ktoré vznikne vo vykonávanej sieti, ktorá inštancie rozlišuje.

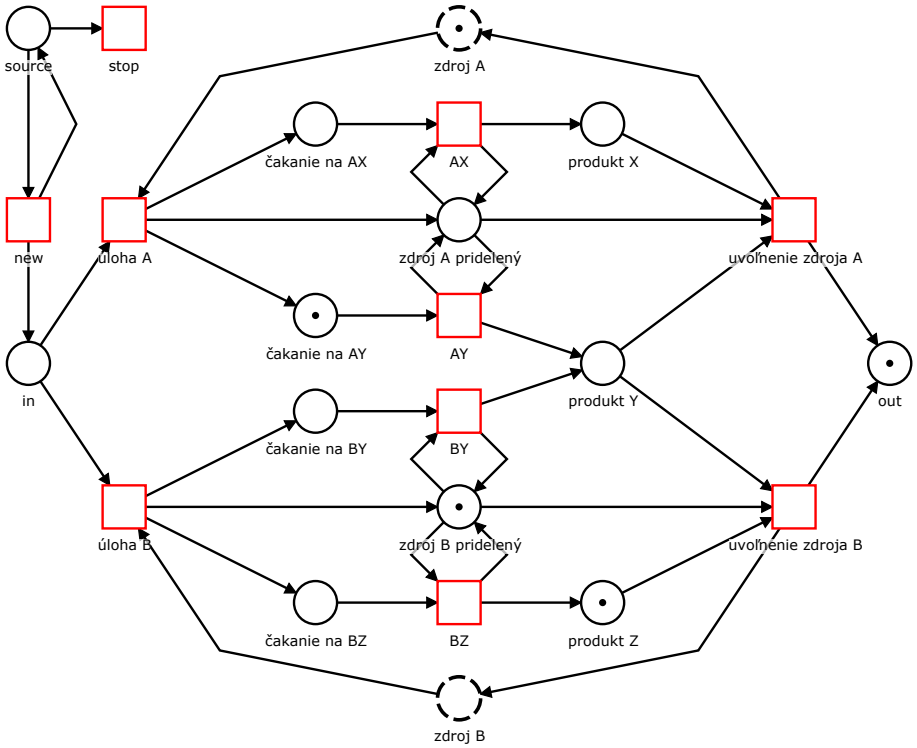
Uvažujme systém, ktorého model je na Obr. 10: Systém realizuje dva typy pracovných úloh, úlohy typu A a úlohy typu B, pričom úlohy typu A realizuje zdroj v role A a úlohy typu B realizuje zdroj v role B. V rámci úlohy A vykonáva zdroj A aktivitu AX, ktorej výsledkom je produkt typu X a aktivitu AY, ktorej 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. 10 má korektné správanie. Na Obr. 11 je znázornená sieť s konštruktorom workflow siete z Obr. 10. Ak dvakrát spustíme prechod *new*, spustíme prechod *stop*, potom prechod *úloha A* a prechod *úloha B* a následne spustíme prechod AX, prechod BY, prechod BZ a prechod *uvoľnenie zdroja A*, sieť s konštruktorom uviazne v značkovaní na Obr. 12. Toto uviaznutie však vo vykonávanej sieti neexistuje. Vykonávaná sieť totiž žiadne uviaznutia nemá. Sieť s konštruktorom teda rozozná uviaznutie neexistujúce vo vykonávanej sieti.



Obr. 10: Systém z dvomi typmi úloh a tromi typmi produktov



Obr. 11: Sieť s konštruktorom workflow siete z Obr. 10



Obr. 12: Uviaznutie siete s konštruktorom z Obr. 11 po spustení prechodov *new, new, stop, úloha A, úloha B, AX, BY, BZ, uvoľnenie zdroja A*

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.

Definícia 22 (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 množín 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.

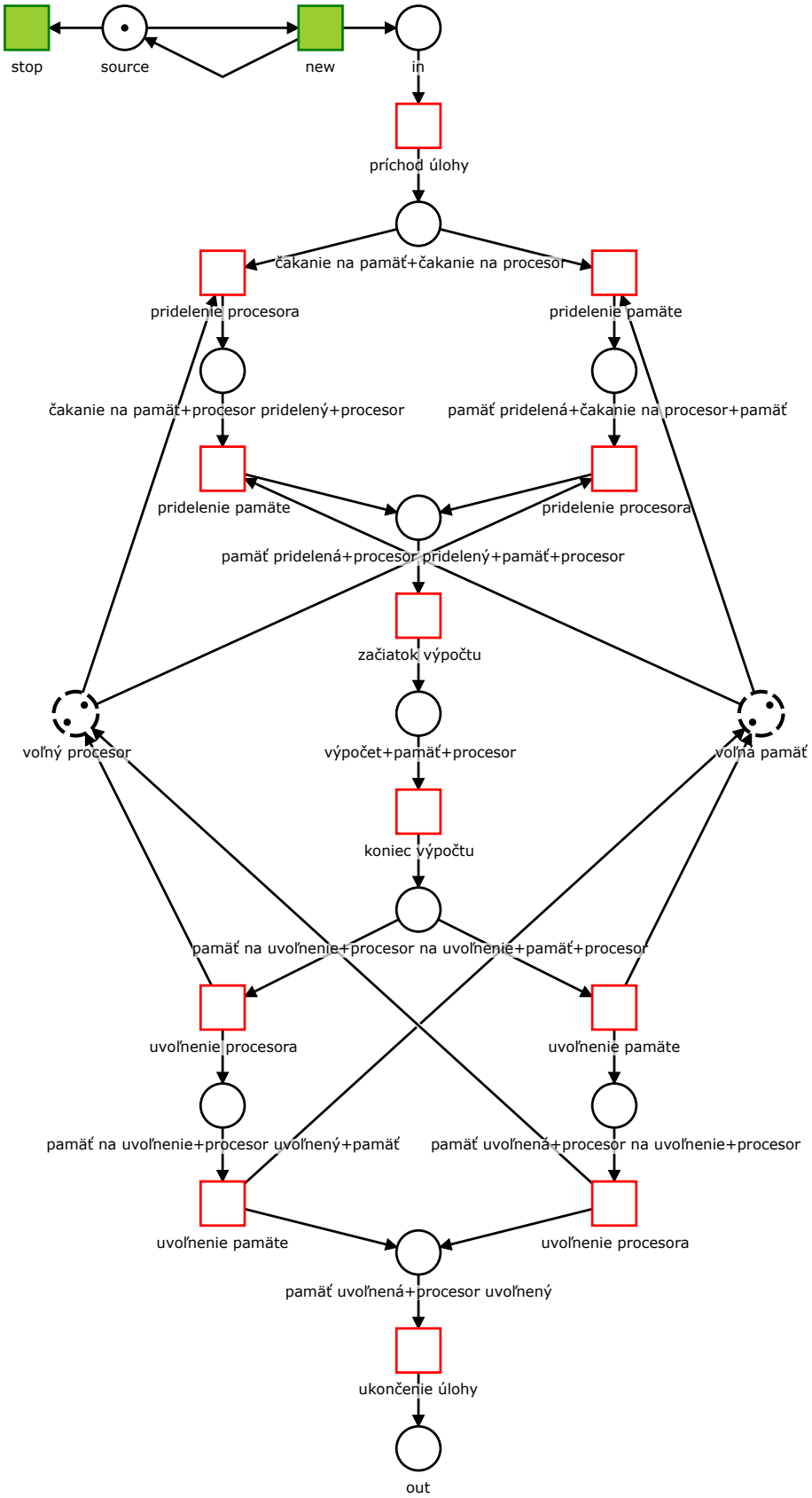
Definícia 23 (Množina dosiahnuteľných D-značkovaní)
Nech $MPN = (P, T, I, O, m_0)$ je označkovaná Petriho sieť. Nech D je nejaká podmnožina miest, teda $D \subseteq P$. Nech m je značkovanie MPN. Potom funkciu $m|D$ nazývame D-značkovanie. Symbolom $[m_0\rangle|D$ označme množinu všetkých takých D-značkovaní 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 sú vytvorené s pomocou dosiahnuteľných D-značkovaní. 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 $(m|D, t, m'|D)$ sú totožné s hranami medzi statickými miestami a prechodom t originálnej siete. Takto vzniknuté miesta a prechody obohatíme o konštruktor obdobným spôsobom ako v prípade sietí s konštruktorom. V grafickom znázornení označíme prechod $(m|D, t, m'|D)$ iba názvom prechodu t.

Definícia 24 (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 15 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 $(p, t) \in (P^r \times T^r)$*



Obr. 13: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 2

- $m_0^r(source) = 1$, $m_0^r|S = m_0|S$ *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.*

Dôsledok 1 *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 so statickými miestami je možné nasledovne.

Algoritmus 1 (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 25 (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* $\mathbb{Z}$*).*

Definícia 26 (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

Pre vykonávané siete platia nasledovné tvrdenia:

Dôsledok 2 *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* $j > i$.

Definícia 27 (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 2 a pre každé $j > max_{\mathbb{I}(m^\infty)}$ platí možnosť 3 z Dôsledku 2. Takéto značkovanie nazývame 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 $f(m^\infty) : \mathbb{I}(m^\infty) \to (P^r \setminus S)$ 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 $n(m^\infty)(x) = |\mathbb{I}(m^\infty, x)|$.*

Prvým hlavným výsledkom tejto práce je nasledovná veta.

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.*

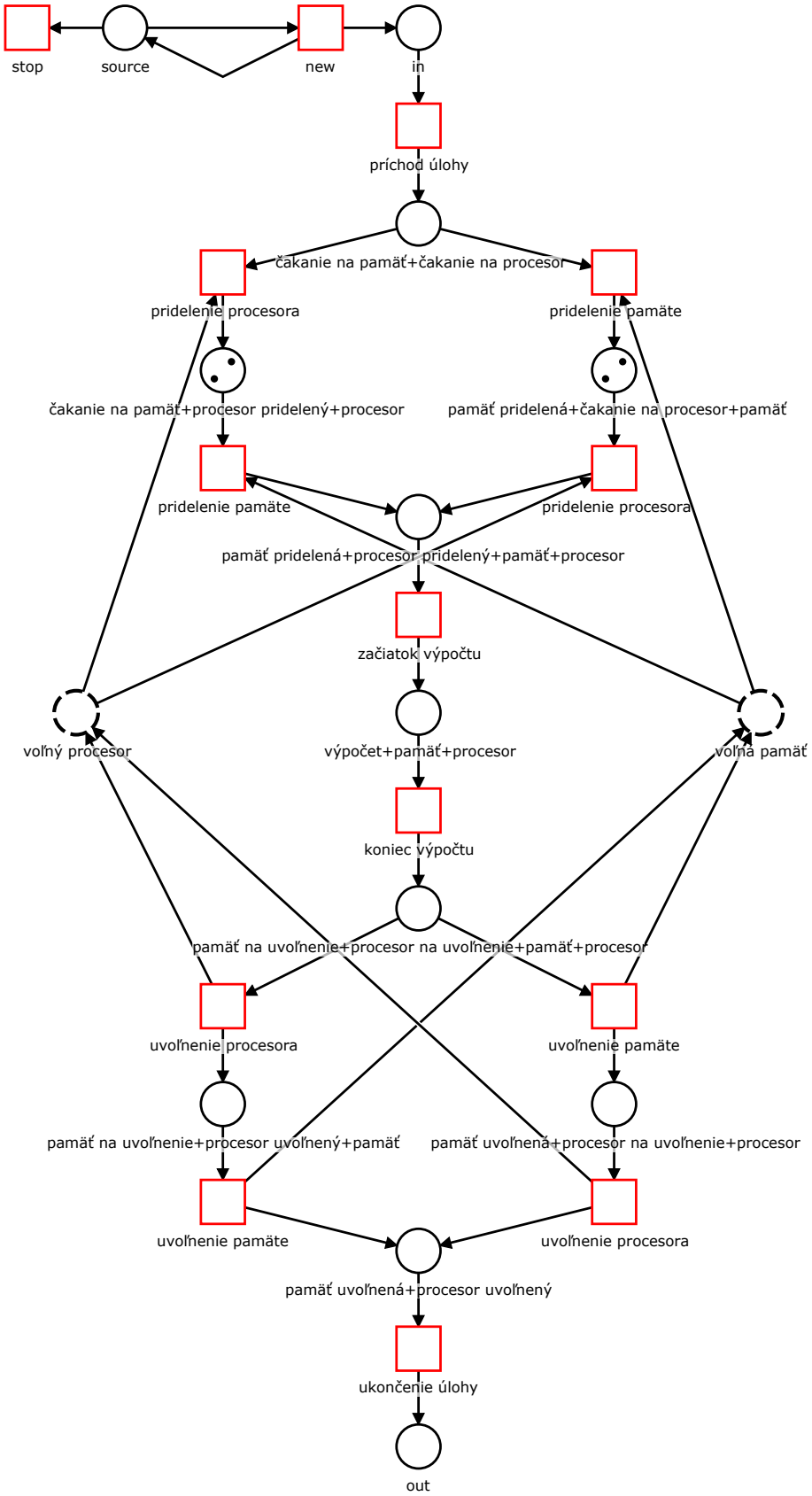
Na Obr. 14 je znázornené uviaznutie siete dosiahnuteľnosti z Obr. 13. Toto uviaznutie zodpovedá uviaznutiu vykonávanej siete zobrazenému na Obr. 4.

2.4 Základné uviaznutia siete dosiahnuteľnosti

Pre metódu vyšetrenia uviaznutí je kľúčový nasledovný výsledok.

Lema 3 *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 $m_2^r | (P^r \setminus S) = w$.*

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



Obr. 14: Uviaznutie siete dosiahnuteľnosti z Obr. 13, ktoré zodpovedá uviaznutiu vykonávanej siete zobrazenému na Obr. 4

Definícia 28 (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, rozdelíme aj prechody siete dosiahnuteľnosti podľa toho, či potrebujú k spusteniu značky zo statických miest. Zároveň rozdelíme miesta podľa toho, či existuje prechod, ktorý presúva značku z miesta so zdrojmi a pritom vyžaduje ďalšie zdroje.

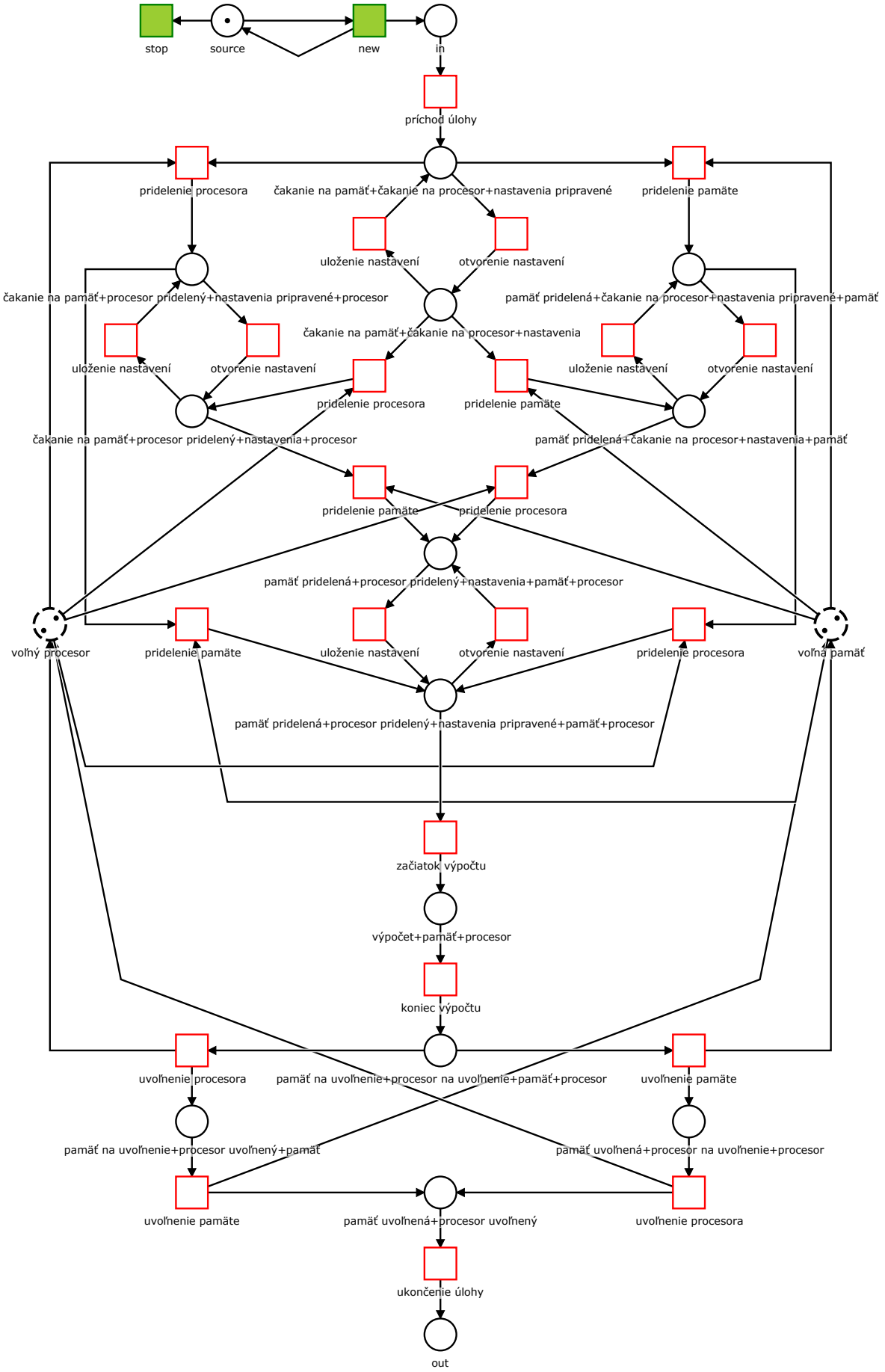
Definícia 29 (Prechody vyžadujúce zdroje, kritické miesta)
Nech $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a nech označkovaná Petriho sieť* $MPN = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Prechod* $(x, t, y) \in T^a$ *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. 15 je znázornená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 5.

Lema 4 *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* $x \in Z^r \setminus K^r$.

Definícia 30 (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 3 *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.



Obr. 15: Sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 5

Definícia 31 (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.*

Lema 5 *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* $m_2^r(x) = m_1^r(x)$ *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.

Na Obr. 16 je znázornené dosiahnuteľné základné uviaznutie siete dosiahnuteľnosti workflow siete so statickými miestami z Obr. 5.

Spojením predchádzajúcich výsledkov, teda tvrdenia z Dôsledku 3 a Lemy 5 dostávame druhý hlavný výsledok práce.

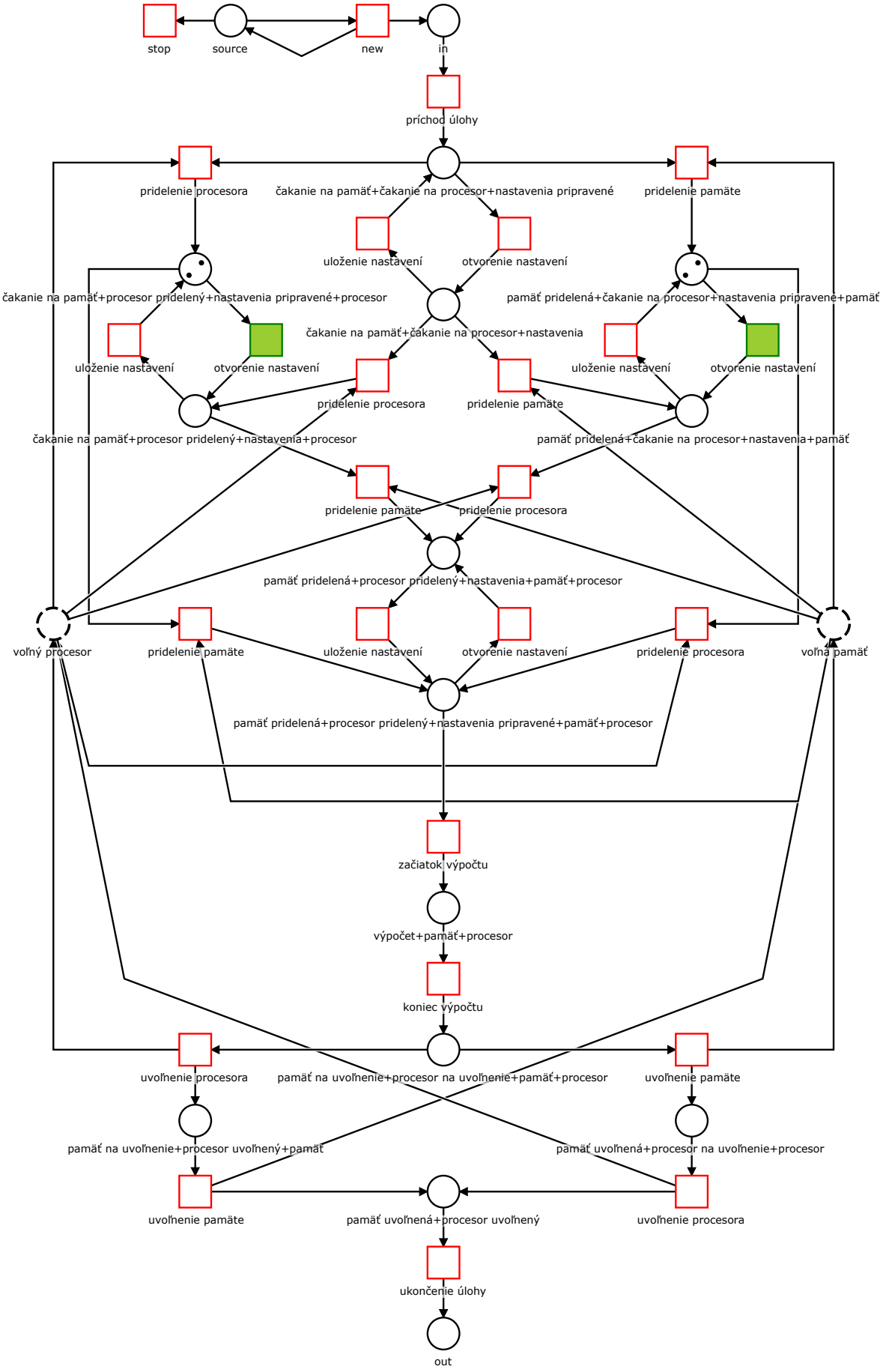
Veta 2 *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^r = (P^r, T^r, I^r, O^r, m_0^r)$ *je sieť dosiahnuteľnosti siete* MW. *Potom platí, že ak v sieti dosiahnuteľnosti existuje uviaznutie* m_1^r *dosiahnuteľné z* m_0^r, *potom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

Dôsledok 4 *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.*

2.5 Obmedzené siete dosiahnuteľnosti

Výsledok vety 5 a Dôsledku 4 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 32 (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.



Obr. 16: Sieť dosiahnuteľnosti workflow siete so statickými miestami z Obr. 5 v základnom uviaznutí pre štyri inštancie

Dôsledok 5 *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* $sbound(K^r) \geq 1$.

Presnejšie horné ohraničenie počtu značiek v kritických miestach môže byť vypočítane ako riešenie nasledovnej úlohy celočíselného lineárneho programovania.

Definícia 33 (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.

Dôsledok 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* 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].

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 $bound(K^r)$, vieme 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 34 (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* $I^n|(\{buffer\} \times (T^r \setminus \{new\})) = 0$.

Je zrejmé, že každá n-obmedzená sieť dosiahnuteľnosti je ohraničená. Na Obr. 17 je znázornená n-obmedzená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 5 pre $n = sbound(K^r) = bound(K^r) = 4$.

V práci taktiež uvádzame horné ohraničenie pre počet dosiahnuteľných značkovaní n-obmedzenej siete dosiahnuteľnosti vypočítané s pomocou výsledkov z [47].

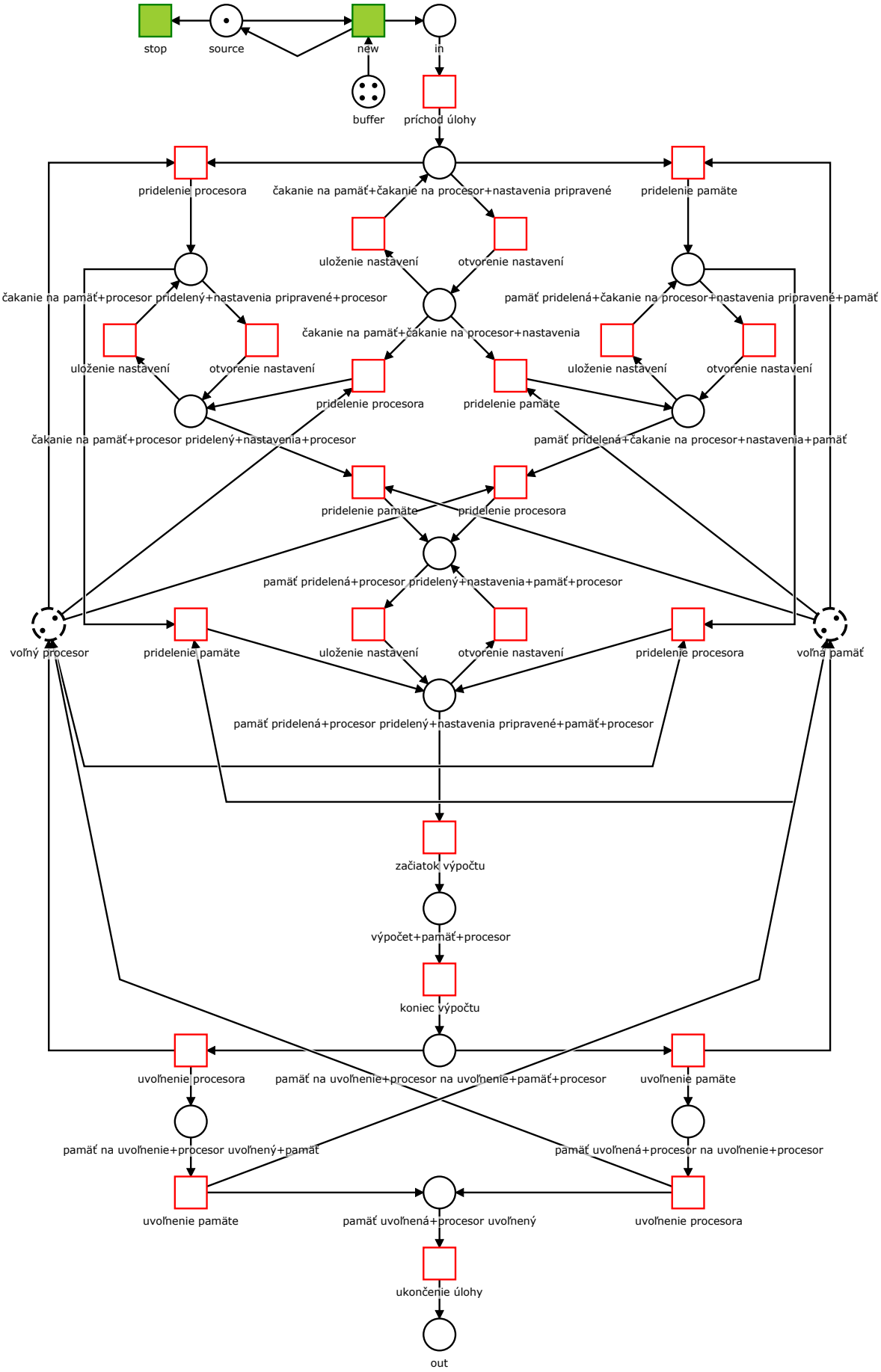
Definícia 35 (Finálne značkovania n-obmedzenej siete dosiahnuteľnosti) *Nech* $MW = (D, S, T, I, O, m_0)$ *je označkovaná určená workflow sieť so statickými miestami a korektným správaním a* $MPN^n = (P^n, T^n, I^n, O^n, m_0^n)$ *je n-obmedzená sieť dosiahnuteľnosti siete* MW. *Značkovanie* m_f^n *n-obmedzenej 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 36 (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 37 (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.*

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.



Obr. 17: n-obmedzená sieť dosiahnuteľnosti označkovanej určenej workflow siete so statickými miestami a korektným správaním z Obr. 5 pre $n = sbound(K^r) = bound(K^r) = 4$

2.6 Detekcia základných uviaznutí

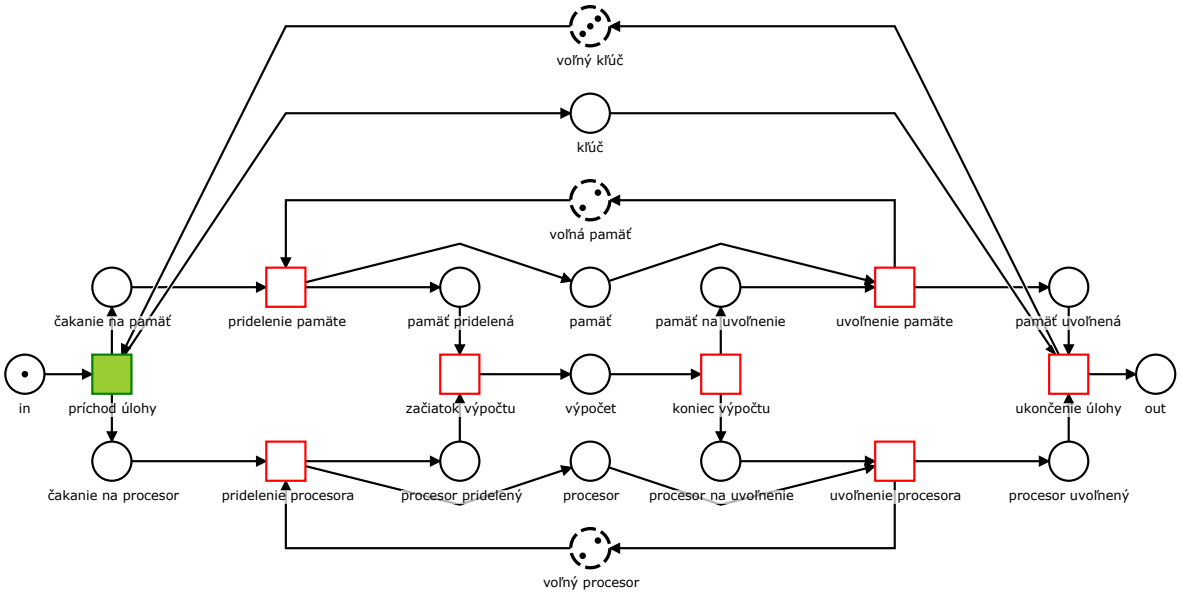
Základné uviaznutia siete dosiahnuteľnosti a n-obmedzenej siete dosiahnuteľnosti pre ľubovoľné n, ktoré nie je menšie ako $bound(K^r)$, sa teda líšia iba v počte značiek v mieste *buffer*, ktorého počet značiek zodpovedá nespotrebovaným značkám daným ako rozdiel medzi počiatočným obmedzením n a súčtom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

Na záver zostavíme 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 2 (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 1 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 $m^n|(P^n \setminus buffer)$ nie je v zozname U^i, vlož $m^n|(P^n \setminus buffer)$ 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



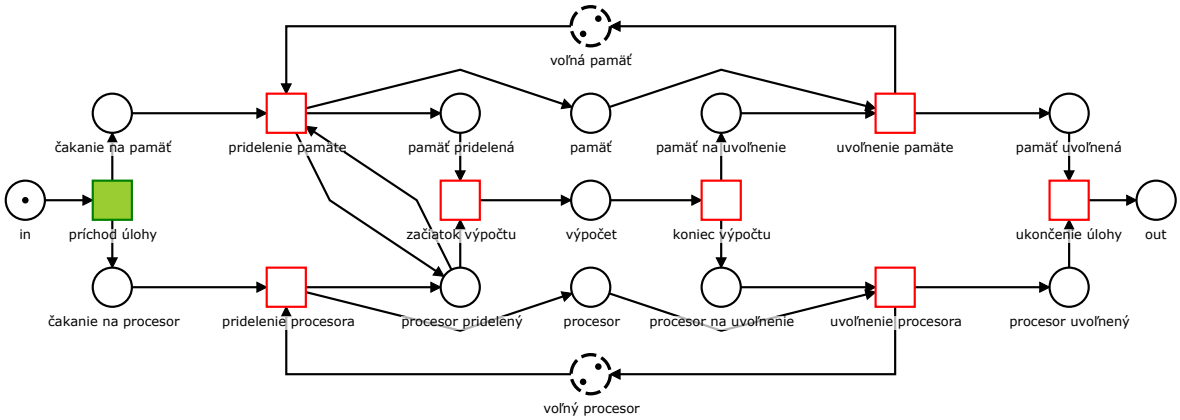
Obr. 18: Rozšírená označkovaná určená workflow sieť so statickými miestami, ktorých označkovanie modeluje počet pamäťových jednotiek a procesorov k dispozícii a kapacitu pre paralelné spracovanie úloh danú počtom značiek v mieste *voľný kľúč*

3 Prehľad príbuzných prác a námety na ďalší výskum

Workflow siete s obmedzenými zdrojmi a trvácnymi zdrojmi pod originálnym anglickým názvom resource constrained workflow nets boli definované v práci [6]. Z pohľadu tejto práce je však najdôležitejším článok [32]. Metóda v práci [32] pracuje 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. 2. Vykonávaná sieť tohto procesu má nejaké uviaznutie pre ľubovoľné počiatočné značkovanie statických miest. Doplňme do tejto siete ďalšie statické miesto tak, aby existovalo značkovanie statických miest, pre ktoré jej vykonávaná sieť nebude mať uviaznutia. Výsledkom je sieť na Obr. 18, doplnená o statické miesto *voľný kľúč* a jeho komplementárne určujúce miesto *kľúč*. Miesto *voľný kľúč* slúži ako ohraničenie kapacity paralelne spracovávaných úloh.

Počet značiek v mieste *voľný kľúč* zabezpečí, že vykonávaná sieť označkovanej určenej workflow siete so statickými miestami a korektným správaním na Obr. 18 nemá žiadne uviaznutia. Ak by sme zmenili značkovanie statických miest určenej workflow siete na Obr. 18 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.

Inou možnosťou ako doplniť sieť z Obr. 2 tak, aby jej vykonávaná sieť nemala uviaznutia, je doplniť hrany medzi prechodom *pridelenie pamäte* a miestom *procesor pridelený*, ako je to znázornené na Obr. 19. Sieť dosiahnuteľnosti a teda aj vykonávaná



Obr. 19: Sekvencializovaná označkovaná určená workflow sieť so statickými miestami modelujúca spracovanie výpočtovej úlohy

sieť procesu z Obr. 19 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. V porovnaní s Obr. 18 však za túto vlastnosť platíme znížením počtu paralelne spracovávaných inštancií.

Definícia 38 (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.*

- *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.*

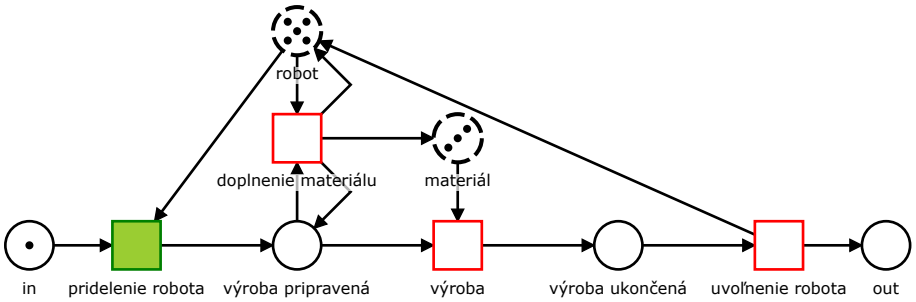
- *Ak pre každé značkovanie* m_0 *siete* W *platí, že vykonávaná sieť* $MPN^{\infty} = (P^{\infty}, T^{\infty}, I^{\infty}, O^{\infty}, m_0^{\infty})$ *označkovanej workflow siete* $MW = (D, S, T, I, O, m_0)$ *má uviaznutia, potom sieť* W *nazývame dynamicky nekorektná.*

V zmysle predchádzajúcej definície teda sieť z Obr. 2 je dynamicky nekorektná, Obr. 18 je dynamicky semi-korektná a sieť z Obr. 19 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. Dynamická korektnosť je inšpirovaná zovšeobecnenou korektnosťou workflow sietí podľa [31]. Korektnosť je skúmaná pre rôzne rozšírenia workflow sietí, napr. v článku [70]. V prácach [41, 42, 43] sme definovali techniku konštruktora a siete dosiahnuteľnosti a metódu detekcie uviaznutí typu zamrznutie pre workflow siete s

pevne daným 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, z toho 16 ohlasov je evidovaných 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] s využitím domáceho priestoru Petriho sietí (anglicky home space)[27] dokázali, že problém určenia dynamickej korektnosti pre workflow siete s korektných správaním je rozhodnuteľný (aj v prípade, ak zdroje nie sú trvácne). 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 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]. 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. Procedúry na vyšetrenie dynamickej korektnosti navyše pre semi-korektné siete, napr. pre sieť na Obr. 18 rozhodnú, že sieť nie je dynamicky korektná, napriek tomu, že pre počiatočné značkovanie na Obr. 18 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

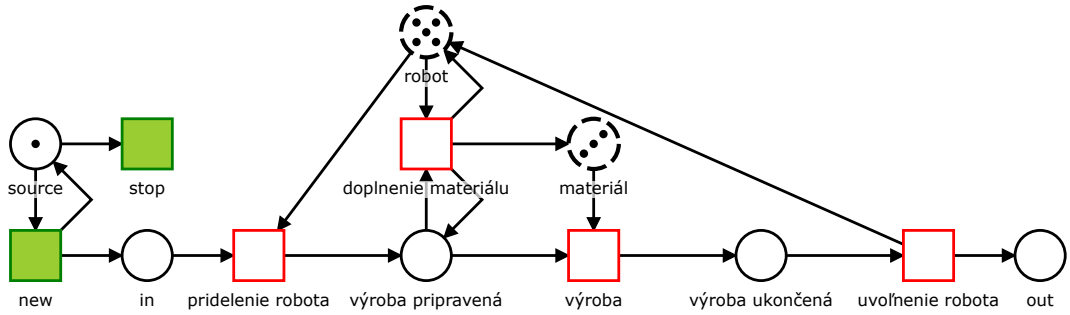


Obr. 20: Označkovaná neurčená workflow sieť so statickými miestami modelujúcimi spotrebný materiál a robotov v jednoduchom procese, v ktorého inštanciách robot buď doplní materiál alebo vyrobí výrobok, pričom spotrebuje materiál

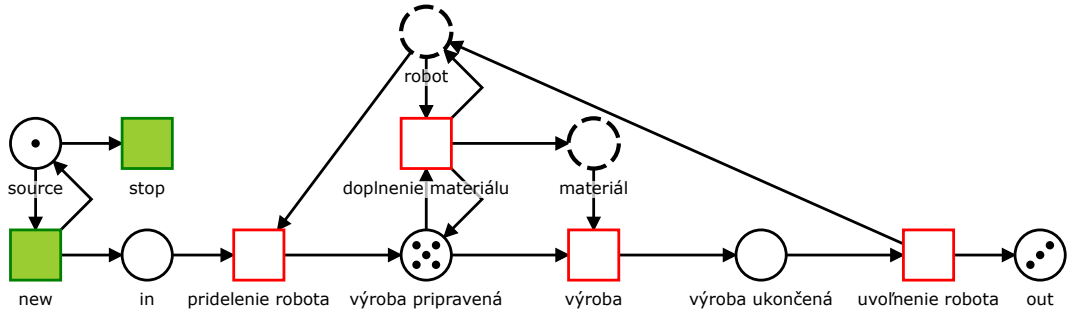
[57] sa venujú rozšíreniam workflow sietí s obmedzenými zdrojmi uvažovaním času a priradením ceny nákladov spojených so spustením prechodov a uchovaním značiek v miestach. Článok [66] sa venuje doplneniu dynamicky nekorektných workflow sietí tak, aby po doplnení boli dynamicky korektné. Doplnenie je 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 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 venujeme možným rozšíreniam výsledkov dizertačnej 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 2 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 work-flow sietí kombinovanú s ohraničenými uviaznutiami.

Uvažujme jednoduchú výrobnú linku, zobrazenú na Obr. 20. 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, pričom spotrebuje jednotku spotrebného materiálu. Na Obr. 21 je znázornená sieť dosiahnuteľnosti procesu na Obr. 20. 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. 20 uviaznutia pre $m_0(robot) + m_0(materiál)$ inštancií. Workflow sieť zná-



Obr. 21: Sieť dosiahnuteľnosti označkovanej neurčenej workflow sieť so statickými miestami a korektným správaním Obr. 20



Obr. 22: Uviazutie v sieti dosiahnuteľnosti označkovanej neurčenej workflow sieť so statickými miestami a korektným správaním pridelením všetkých robotov do výroby bez doplnenia spotrebovaného materiálu

zornená na Obr. 20 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. 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 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]. V Časti 1.2 sme uviedli základné definície Petriho sietí pomenované podľa Carla Adama Petriho [64] a popísané napríklad v prácach [44, 62, 63, 67, 23, 24]. Kapitola 1 pokra-

čuje 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, pôvodne definovaných 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. Vykonávané siete sú ekvivalentné sieťam s identifikačnými číslami používaným v práci [32]. Siete s identifikačnými číslami 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 work-flow 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 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 konečný. 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 venovali diskusii vzťahu prezentovaných vlastných výsledkov s príbuznými prácami, kategorizovali sme typy dynamickej korektnosti a načrtli sme možné smery budúceho výskumu.

Publikované práce autora

1. JUHÁS, G. – KAZLOV, I. – JUHÁSOVÁ, A. Instance Deadlock: A Mystery behind Frozen Programs. Lecture Notes in Computer Science, 6128. s. 1–17. Springer 2010. ISSN 0302-9743.

2. JUHÁSOVÁ, A. – JUHÁS, G. Soundess of resource constrained workflow nets is decidable. Petri Net Newsletter. Vol. 76, November 2009, p. 3-6. ISSN 0931-1084.

3. JUHÁSOVÁ, A. – JANOV, V. – JUHÁS, G. Dynamický deadlock WorkFlow procesov. EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie, 15. s. 205–208. ISSN 1335-2547. 2009.

4. JUHÁSOVÁ, A. The Application of Knowledge in Financial Environment. In Knowledge - New Challenges for National Economy Science: Medzinárodná vedecká konferencia Národohospodárskej fakulty Ekonomickej univerzity v Bratislave. Bratislava, Slovak Republic, 19.-20.10.2006. Bratislava : Ekonomická univerzita, ISBN 80-225-2249-X. 2006.

5. JUHÁSOVÁ, A. – JUHÁS, G. – LEHOCKI, F. – ŠEVČÍKOVÁ, Z. Petriho siete v modelovaní workflow procesov. EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie, 13. s. 303–305. ISSN 1335-2547. 2007.

6. ŠEVČÍKOVÁ, Z. – JUHÁS, G. – JUHÁSOVÁ, A. – LEHOCKI, F. Sémantika a aplikácie Petriho sietí. EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie, 13. s. 306–308. ISSN 1335-2547. 2007.

7. LEHOCKI, F. – JUHÁS, G. – ŠEVČÍKOVÁ, Z. – JUHÁSOVÁ, A. Aplikácie Petriho sietí v medicínskych diagnostických systémoch. EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie, 13. s. 299–302. ISSN 1335-2547. 2007.

8. ŠEVČÍKOVÁ, Z. – KAZLOV, I. – JUHÁSOVÁ, A. Možnosti využitia analýzy a syntézy udalostných systémov v praxi. EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie: ELOSYS, Trenčín, 11.-14.10.2011, 17. s. 227–229. ISSN 1335-2547. 2011.

9. LEHOCKI, F. – ORAVEC, M. – BALOGH, Š. – JUHÁSOVÁ, A. Vybrané modely diagnostických systémov. In SMART S2AI: Workshop SMART systémov a služieb v oblasti aplikovanej informatiky. Bratislava, s. 1–5. 2011.

10. JUHÁS, G. – FOLTIN, M. – ŠEVČÍKOVÁ, Z. – JUHÁSOVÁ, A. Zmráka sa. EE časopis pre elektrotechniku, elektroenergetiku, informačné a komunikačné technológie: ELOSYS, Trenčín, 11.-14.10.2011, 17. s. 204–207. ISSN 1335-2547. 2011.

Ohlasy na práce autora

Práca:
JUHÁS, G. – KAZLOV, I. – JUHÁSOVÁ, A. Instance Deadlock: A Mystery behind Frozen Programs. Lecture Notes in Computer Science, 6128. s. 1–17. Springer 2010. ISSN 0302-9743.
evidovaná v databázach Web of Science (WoS) a Scopus je citovaná v prácach:

1. 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. (publikácia evidovaná v databázach WoS a Scopus)

2. 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. (publikácia evidovaná v databázach WoS a Scopus)

3. Martos-Salgado, María, and Fernando Rosa-Velardo. Safety and Soundness for Priced Resource-Constrained Workflow Nets. Fundamenta Informaticae 131.1 (2014): 55-80. (publikácia evidovaná v databázach WoS a Scopus)

4. 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. (publikácia evidovaná v databázach WoS a Scopus)

5. Solé, Marc, and Josep Carmona. Light region-based techniques for process discovery. Fundamenta Informaticae 113.3-4 (2011): 343-376. (publikácia evidovaná v databázach WoS a Scopus)

6. 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. (publikácia evidovaná v databázach WoS a Scopus)

7. 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. (publikácia evidovaná v databázach WoS a Scopus)

8. 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. (publikácia evidovaná v databáze WoS)

9. Wang, Jiacun, Bill Tepfenhart, and Xiaoou Li. Analysis of Minimum Workflow Resource Requirement. International Workshop on Process-Aware Systems. Springer Singapore, 2015. (publikácia evidovaná v databáze WoS)

10. Bergenthum, Robin, Jörg Desel, and Sebastian Mauser. Workflow Nets with Roles. EMISA. 2011. (publikácia evidovaná v databáze Scopus)

11. Martiník, Ivo. Automation of Presentation Record Production Based on Rich-Media Technology Using SNT Petri Nets Theory. The Scientific World Journal 2015 (2015). (publikácia evidovaná v databáze Scopus)

12. 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. (publikácia evidovaná v databáze Scopus)

13. 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. (publikácia evidovaná v databáze Scopus)

14. 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. (publikácia evidovaná v databáze Scopus)

15. 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. (publikácia evidovaná v databáze Scopus)

16. Li, Xiaoou, et al. Resource requirement analysis for cyclic workflows. 2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC). IEEE, 2016. (publikácia evidovaná v databáze Scopus)

17. Birch, Sine Viesmose, and Christoffer Moesgaard. Compositional Analysis of Timed-arc Resource Workflows with Communication. University of Aalborg, 2015. (publikácia evidovaná v databáze Google Scholar)

18. Cortés, Mateo, Verification and Validation of Web Services Compositions Using Wormal Methods, PhD thesis, University of Castilla-La Mancha (2014). (publikácia evidovaná v databáze Google Scholar)

19. Martos Salgado, María Rosa. Verification of priced and timed extensions of Petri Nets with multiple instances. PhD thesis, Complutense University of Madrid (2016). (publikácia evidovaná v databáze Google Scholar)

20. 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). (publikácia evidovaná v databáze Google Scholar)

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. 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.