Detekcia uviaznutí v diskrétnych udalostných systémoch so zdieľanými zdrojmi

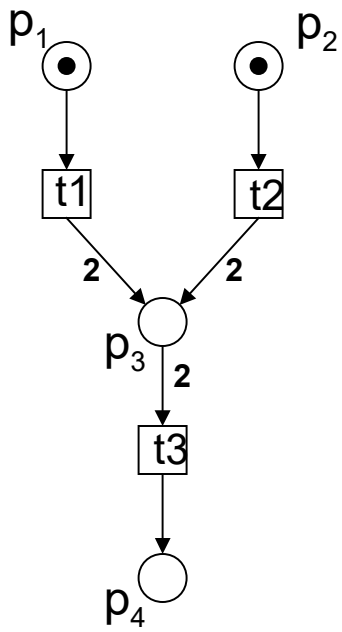
Gabriel Juhás

Modelovanie a simulácia udalostných systémov
(I-MSUS - ZS 2017/2018 – FEI, prvá časť)

Obsah

Petriho sieť - definícia

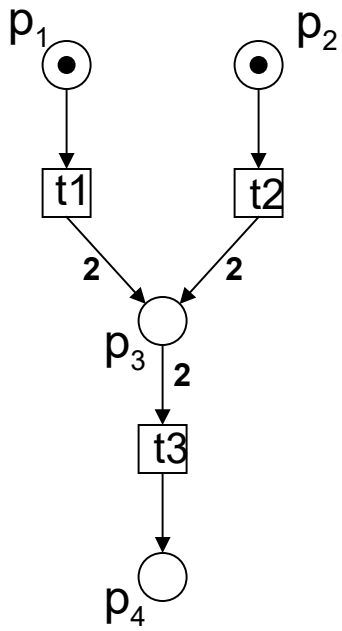
(P,T,I,O,m0)



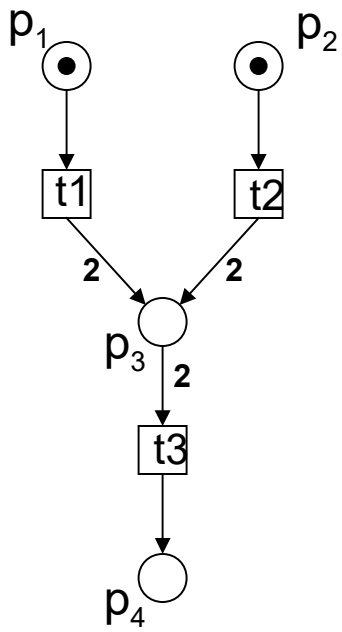
Petriho sieť - definícia

(P,T,I,O,m0)

P je konečná množina miest



Petriho sieť - definícia



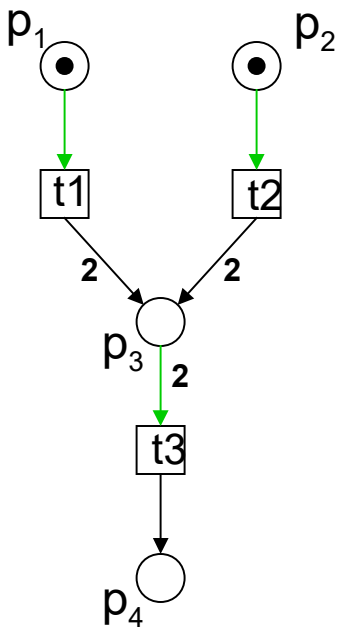
(P,T,I,O,m0)

P je konečná množina miest

T je konečná množina prechodov

Petriho sieť - definícia

(P,T,I,O,m0)



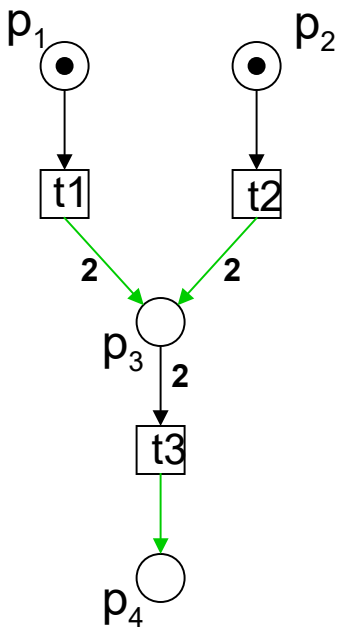
I je Vstupná funkcia (vstupná matica)

I: P × T → N priraďuje každému miestu p a každému prechodu t koľko značiek spotrebuje z miesta p spustenie prechodu t

I	t1	t2	t3
p1	1	0	0
p2	0	1	0
p3	0	0	2
p4	0	0	0

Petriho sieť - definícia

(P,T,I,O,m0)



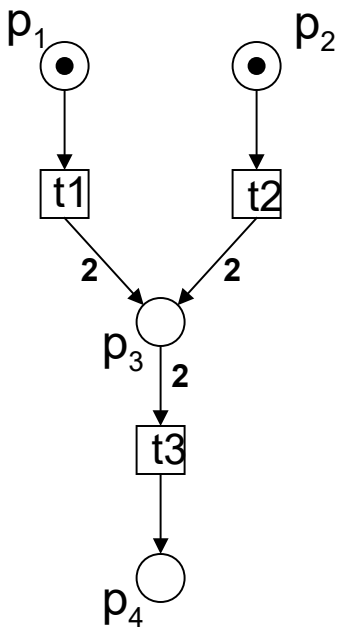
O je Výstupná funkcia (výstupná matica)

O: P × T → N priraďuje každému miestu p a každému prechodu t koľko značiek sa vyprodukuje v mieste p spustením prechodu t

O	t1	t2	t3
p1	0	0	0
p2	0	0	0
p3	2	2	0
p4	0	0	1

Petriho sieť - definícia

(P,T,I,O,m0)



m0 je počiatočné značkovanie, teda funkcia m0: P $\rightarrow$ N, ktorá priraďuje každému miestu p počet značiek v mieste p

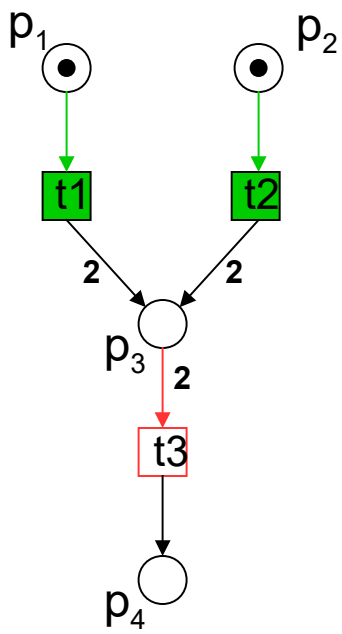
m0(p1) = 1

m0(p2) = 1

m0(p3) = 0

m0(p4) = 0

Petriho sieť – spustiteľnosť prechodu

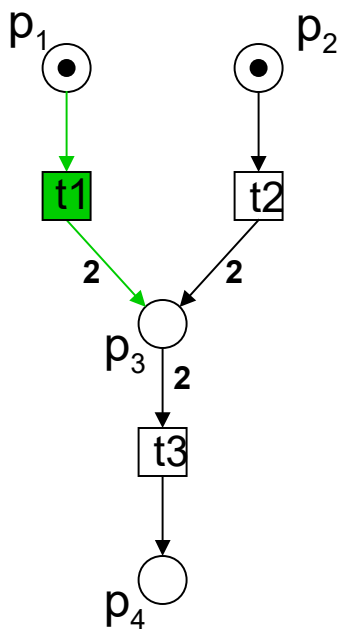


Prechod $t \in T$ je spustiteľný v značkovaní $m: P \rightarrow N$ práve vtedy keď platí

$\forall\ p \in P: m(p) \geq I(p,t)$

I	t1	t2	t3
p1	1	0	0
p2	0	1	0
p3	0	0	2
p4	0	0	0

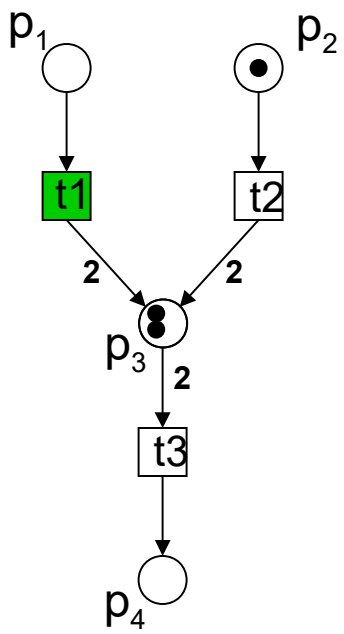
Petriho sieť – spustenie prechodu



Ak prechod $t \in T$ je spustiteľný v značkovaní m: $P \rightarrow N$ potom pustenie prechodu t v značkovaní m vedie k novému značkovaniu m′: $P \rightarrow N$ takému že $\forall\ p \in P$:

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

Petriho sieť – spustenie prechodu



Ak prechod $t \in T$ je spustiteľný v značkovaní m: $P \rightarrow N$ potom pustenie prechodu t v značkovaní m vedie k novému značkovaniu m′: $P \rightarrow N$ takému že $\forall\ p \in P$:

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

Workflow siete –

práve jedno vstupné miesto in

(O(in,t) = 0 pre každý prechod t a existuje prechod t taký, že I(in,t)>0)

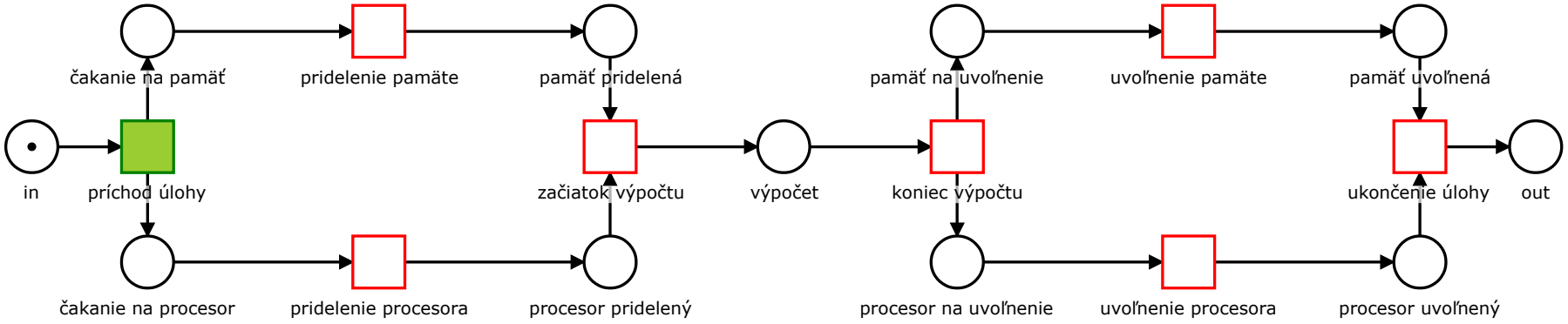
práve jedno výstupné miesto out

(I(out,t) = 0 pre každý prechod t a existuje prechod t taký, že O(out,t)>0)

počiatočné značkovanie – jedna značka vo vstupnom mieste in

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

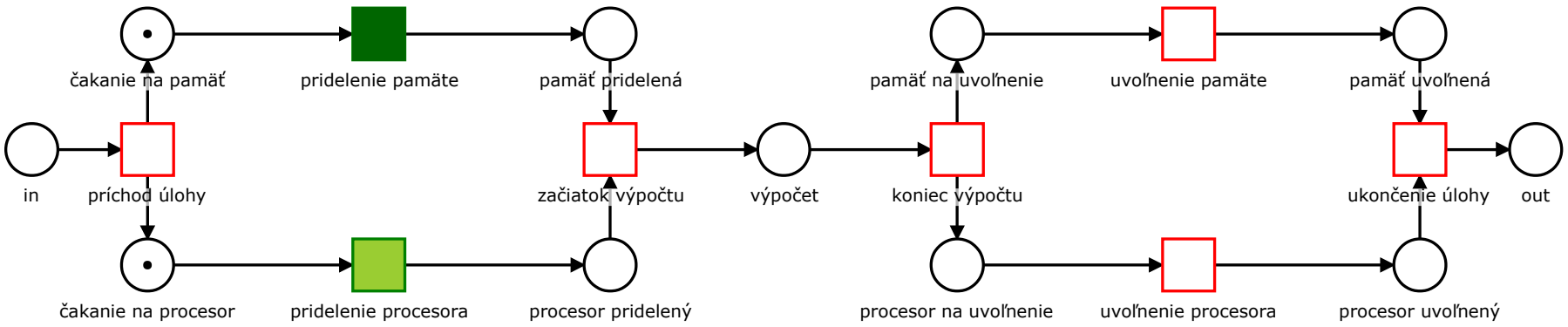
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

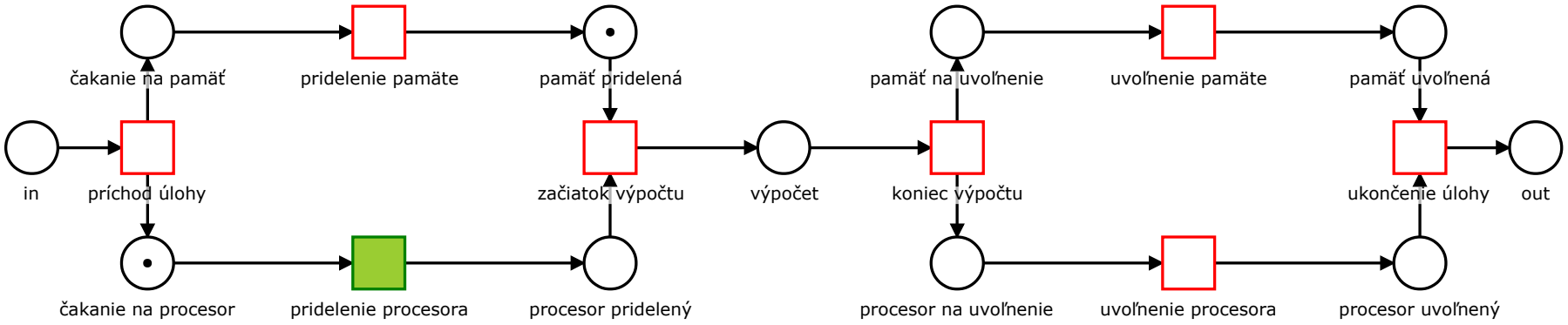
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

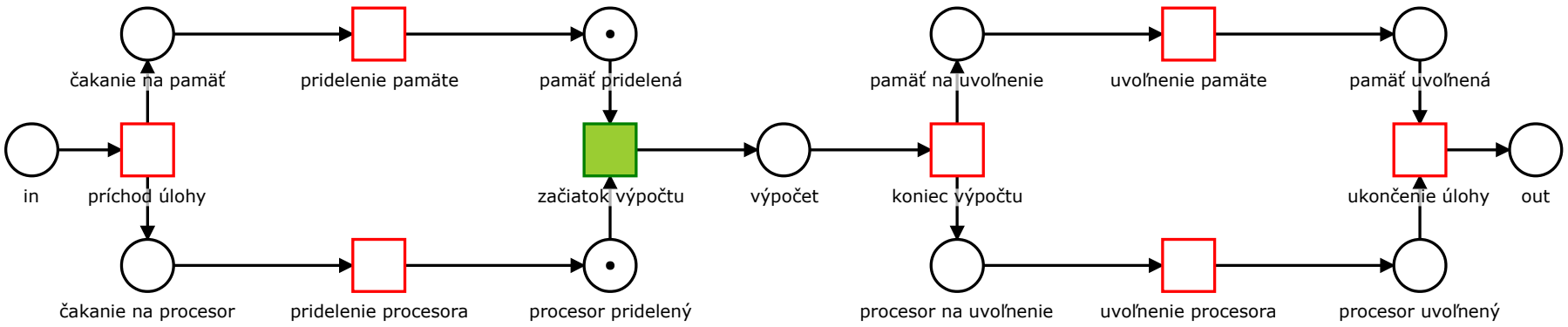
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

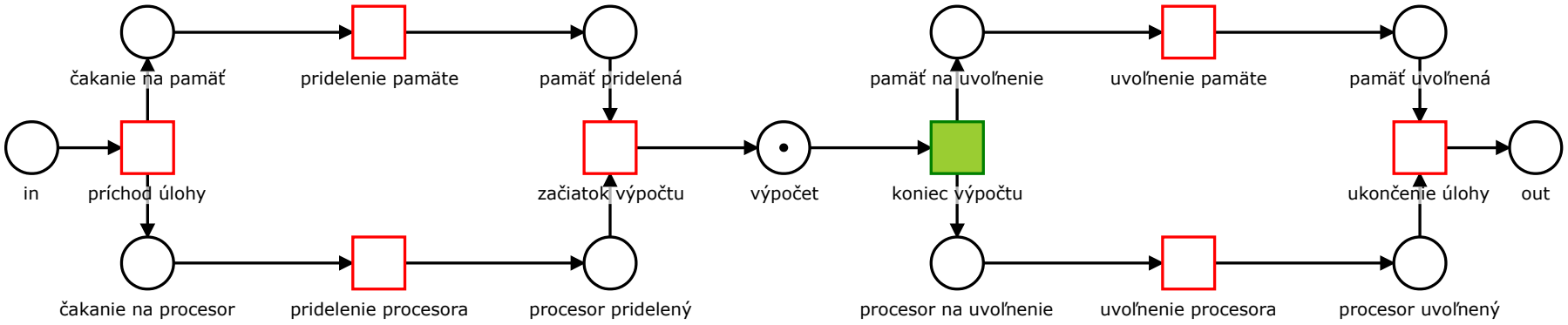
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

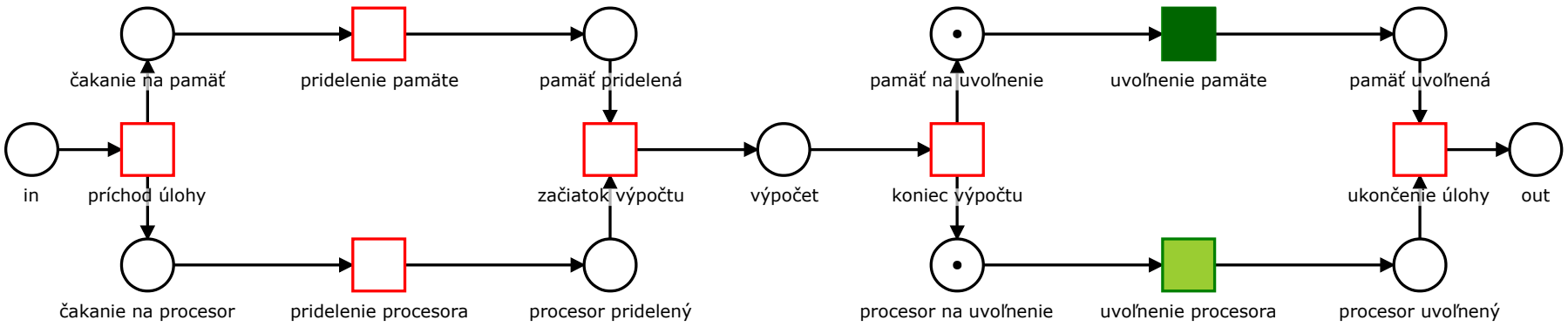
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

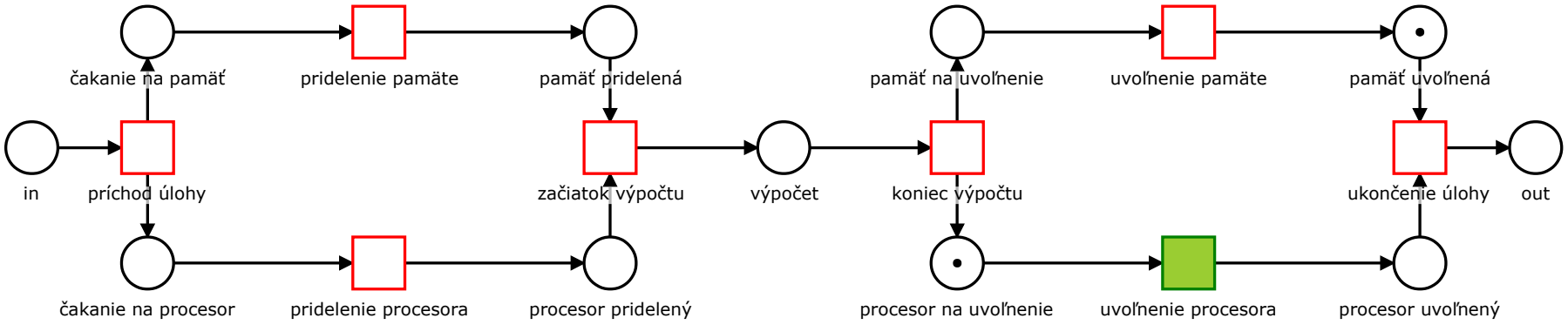
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

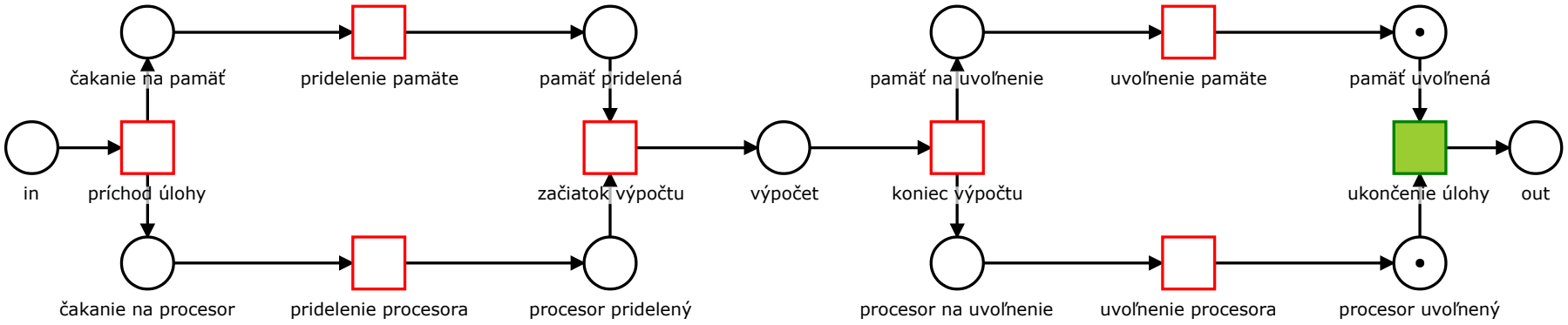
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

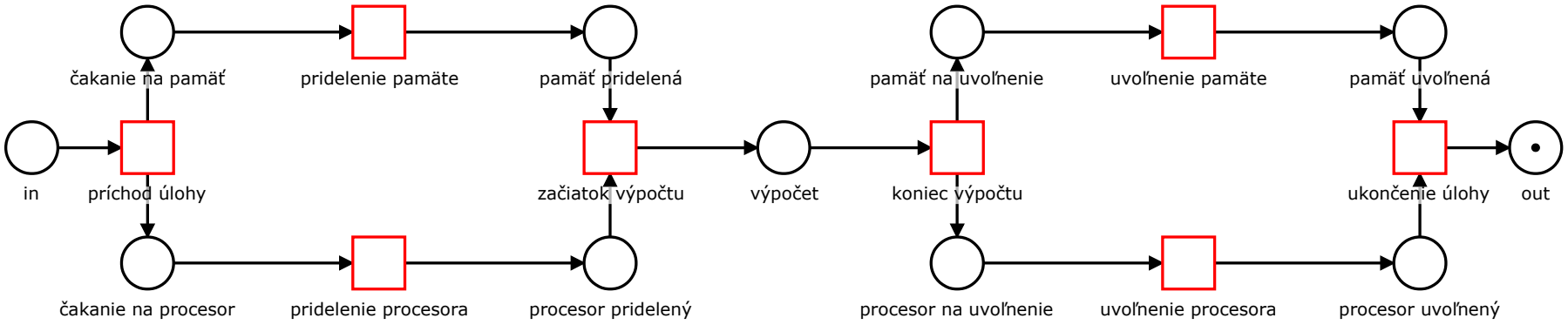
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

Ilustratívny príklad - popisuje systém - program, v ktorom budú vykonávané výpočtové úlohy:

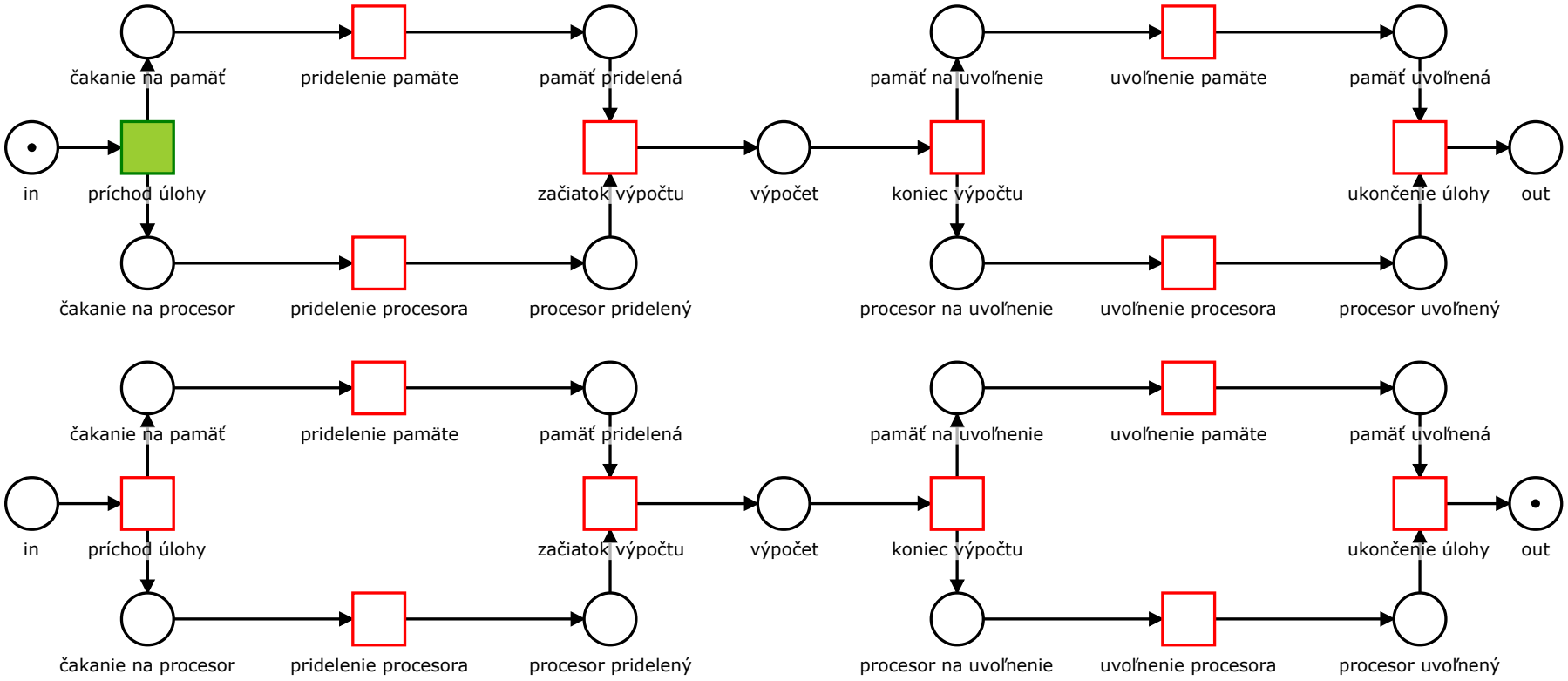
výpočtová úloha príde do systému, systém jej pridelí jednotku operačnej pamäte a procesor, následne sa vykoná výpočet, po jeho ukončení sa uvoľní jednotka pamäte a procesor a úloha sa ukončí.



Workflow siete –

workflow sieť je korektná, ak značkovanie s jednou značkou vo výstupnom mieste out

- **je dosiahnuteľné z každého dosiahnuteľného značkovania**
- **je jediným dosiahnuteľným značkovaním so značkou vo výstupnom mieste out**

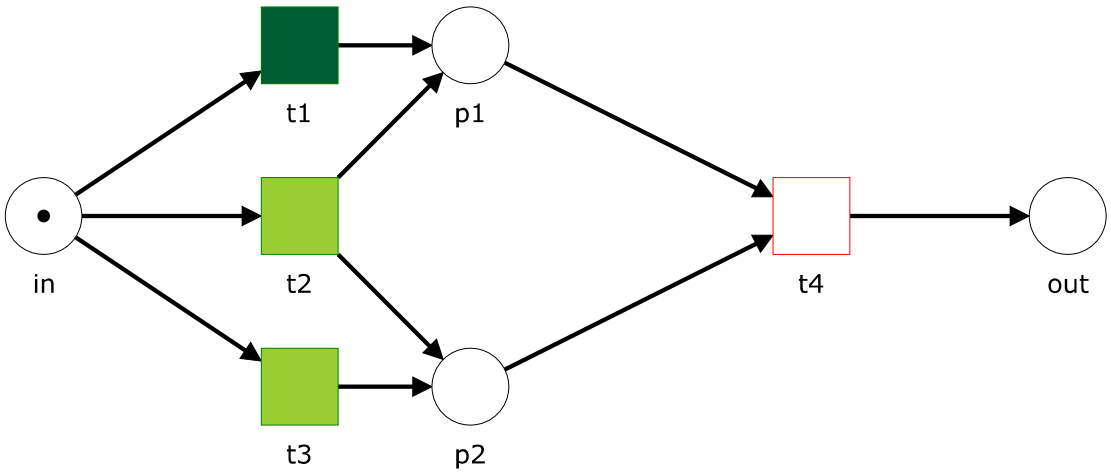


Workflow siete – príklad nekorektnej siete

workflow sieť je korektná, ak značkovanie s jednou značkou vo výstupnom mieste out

1) je dosiahnuteľné z každého dosiahnuteľného značkovania

2) je jediným dosiahnuteľným značkovaním so značkou vo výstupnom mieste out

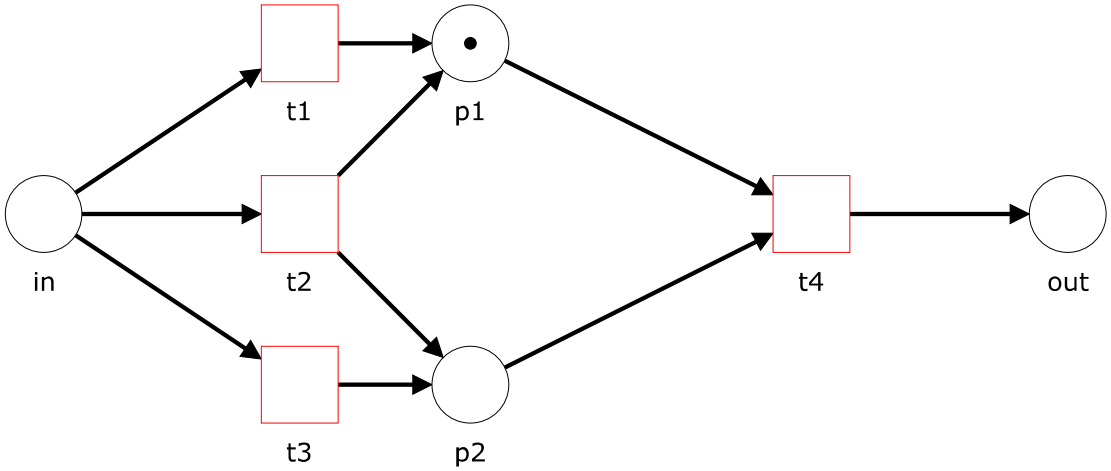


Workflow siete – príklad nekorektnej siete

workflow sieť je korektná, ak značkovanie s jednou značkou vo výstupnom mieste out

1) ~~je dosiahnuteľné z každého dosiahnuteľného značkovania~~

2) je jediným dosiahnuteľným značkovaním so značkou vo výstupnom mieste out

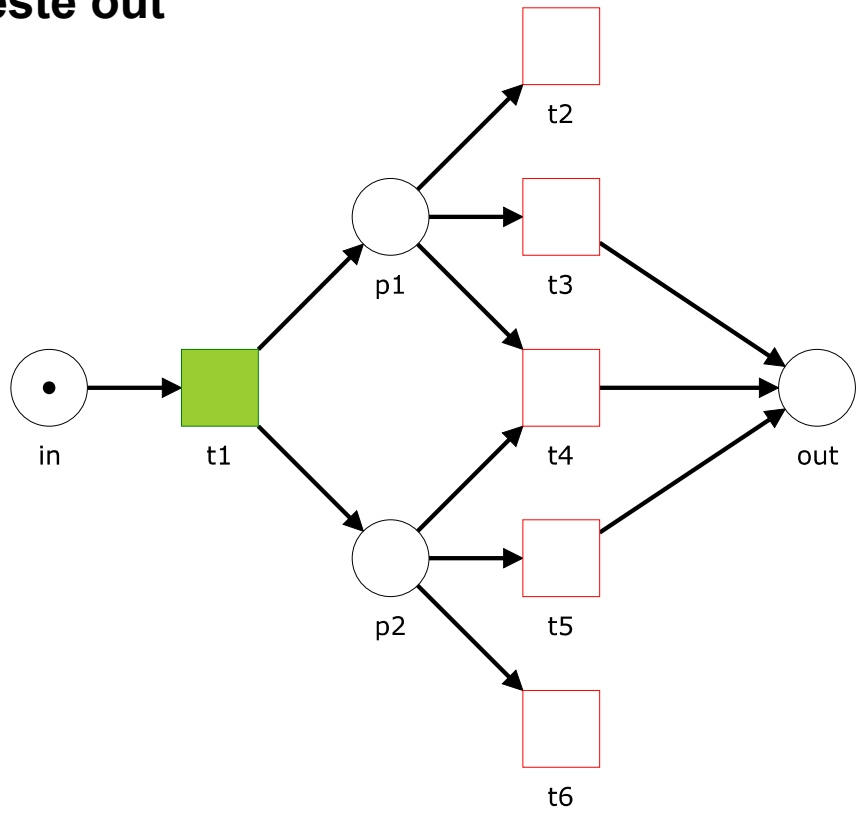


Workflow siete – príklad nekorektnej siete

workflow sieť je korektná, ak značkovanie s jednou značkou vo výstupnom mieste out

1) je dosiahnuteľné z každého dosiahnuteľného značkovania

2) je jediným dosiahnuteľným značkovaním so značkou vo výstupnom mieste out

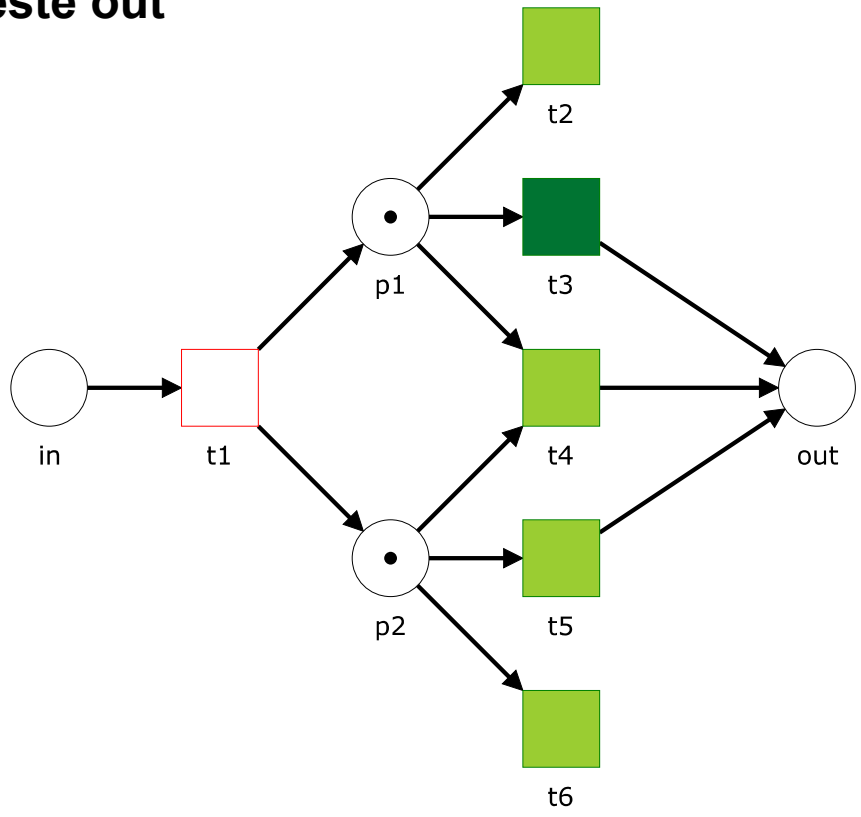


Workflow siete – príklad nekorektnej siete

workflow sieť je korektná, ak značkovanie s jednou značkou vo výstupnom mieste out

1) je dosiahnuteľné z každého dosiahnuteľného značkovania

2) je jediným dosiahnuteľným značkovaním so značkou vo výstupnom mieste out

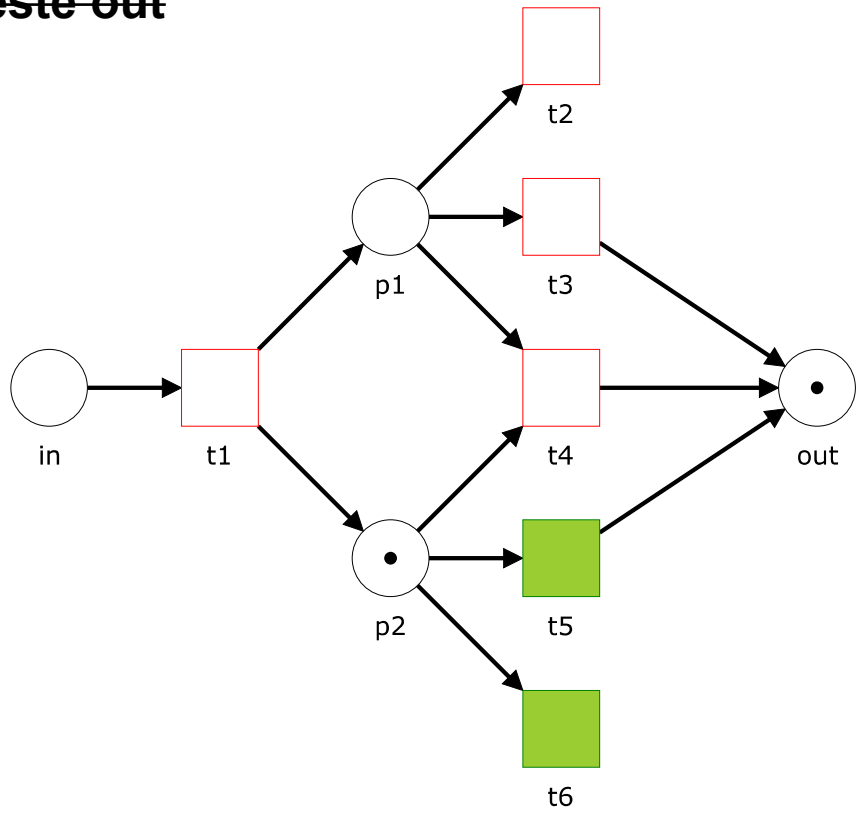


Workflow siete – príklad nekorektnej siete

workflow sieť je korektná, ak značkovanie s jednou značkou vo výstupnom mieste out

1) je dosiahnuteľné z každého dosiahnuteľného značkovania

2) ~~je jediným dosiahnuteľným značkovaním so značkou vo výstupnom mieste out~~



Workflow siete –

Ak je workflow sieť korektná, potom je ohraničená

(má konečný počet dosiahnuteľných značkovaní)

Workflow siete –

Ak je workflow sieť korektná, potom je ohraničená

(má konečný počet dosiahnuteľných značkovaní)

Graf dosiahnuteľnosti – vrcholy sú značkovania

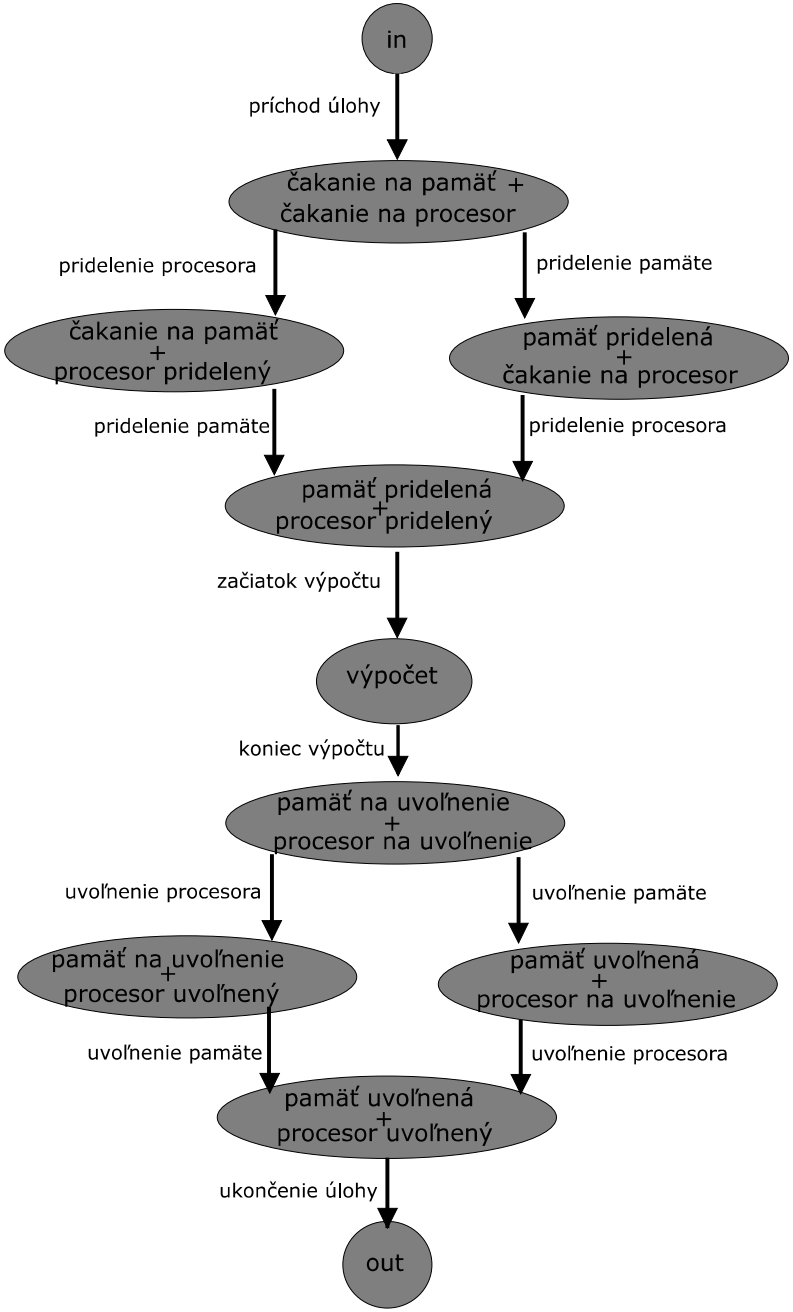
Hrana spájajúca vrcholy

m a m' označená vrcholom t

reprezentuje, že:

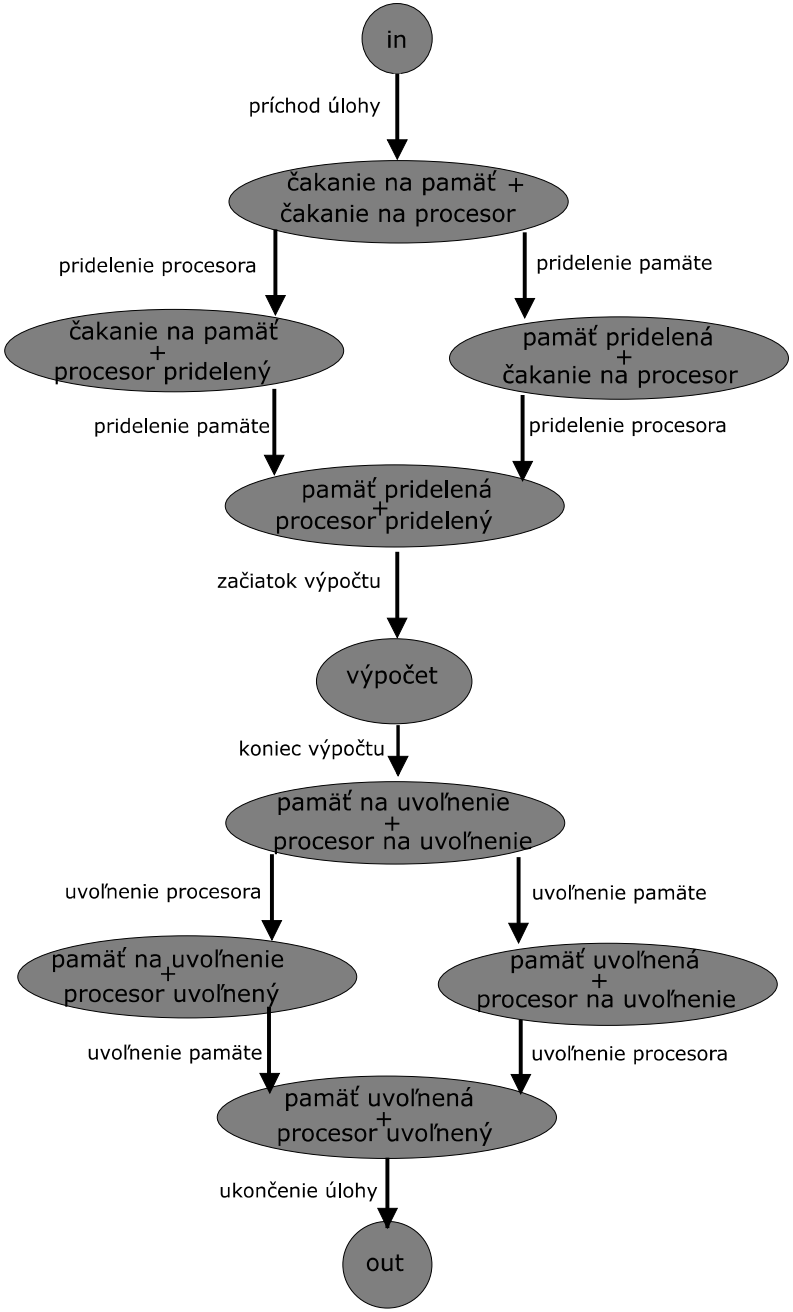
prechod t je spustiteľný v značkovaní m

a jeho spustenie v značkovaní m vedie do značkovania m'



Workflow siete –

Ak je Petriho sieť neohraničená, potom existuje v jej grafe dosiahnuteľnosti taký vrchol, že niektorý z jeho predchodcov je menšie značkovanie

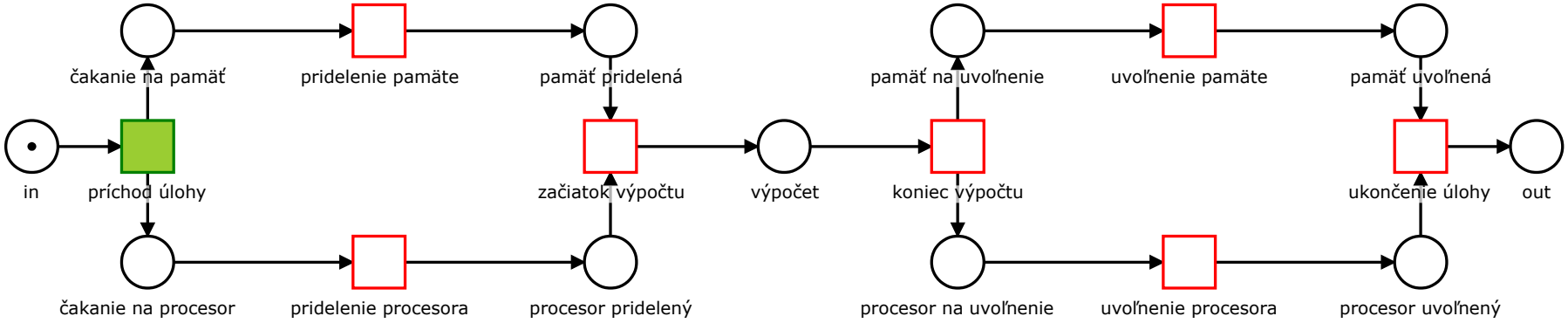


Workflow siete –

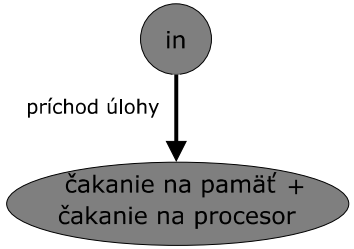
in

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

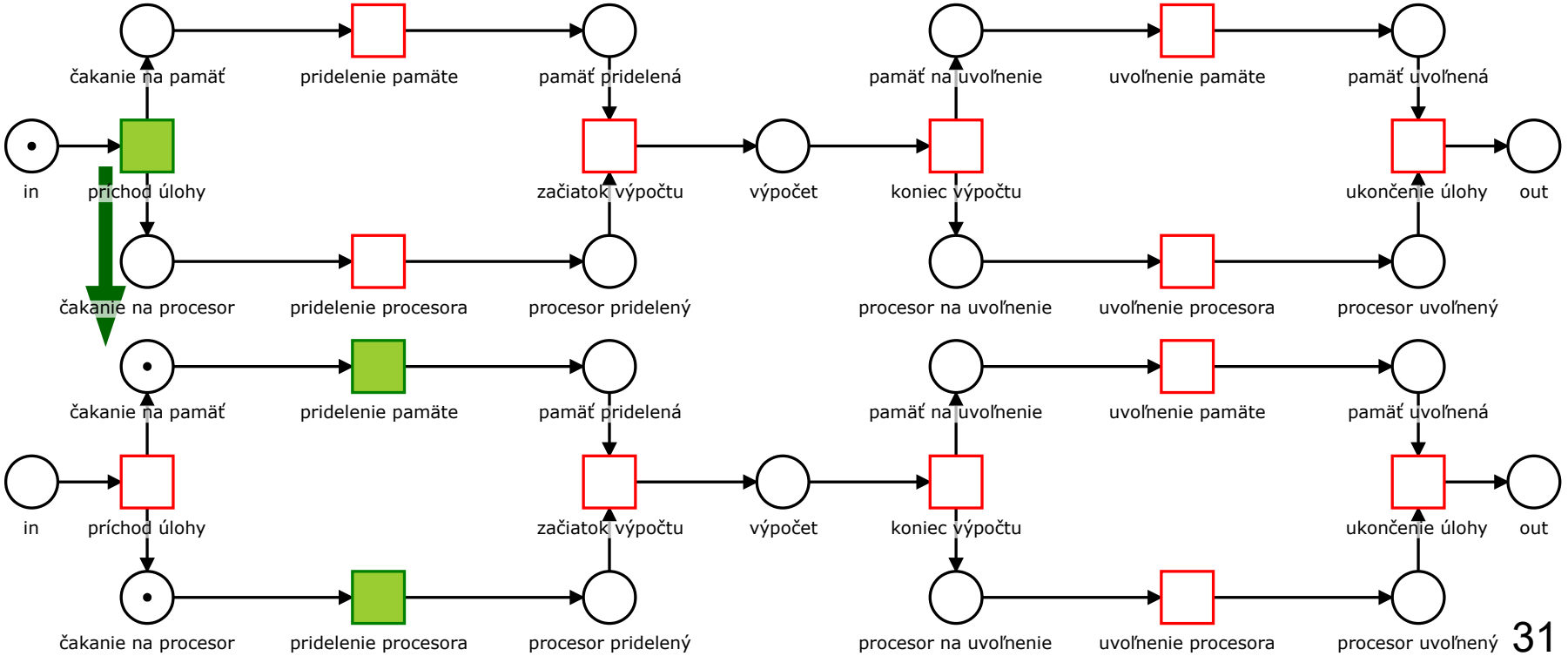


Workflow siete –



Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

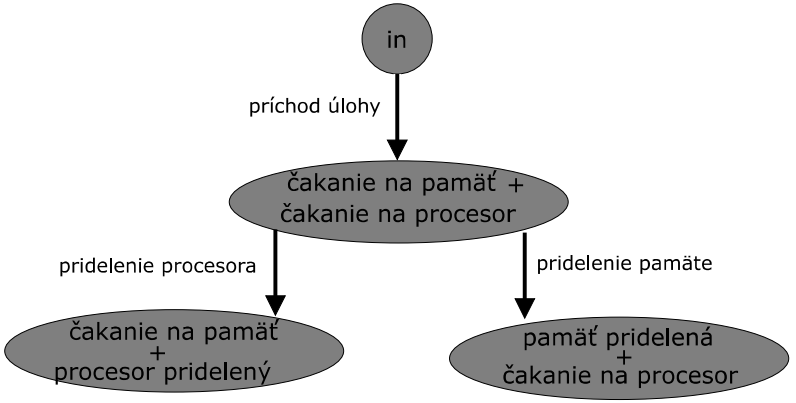
ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

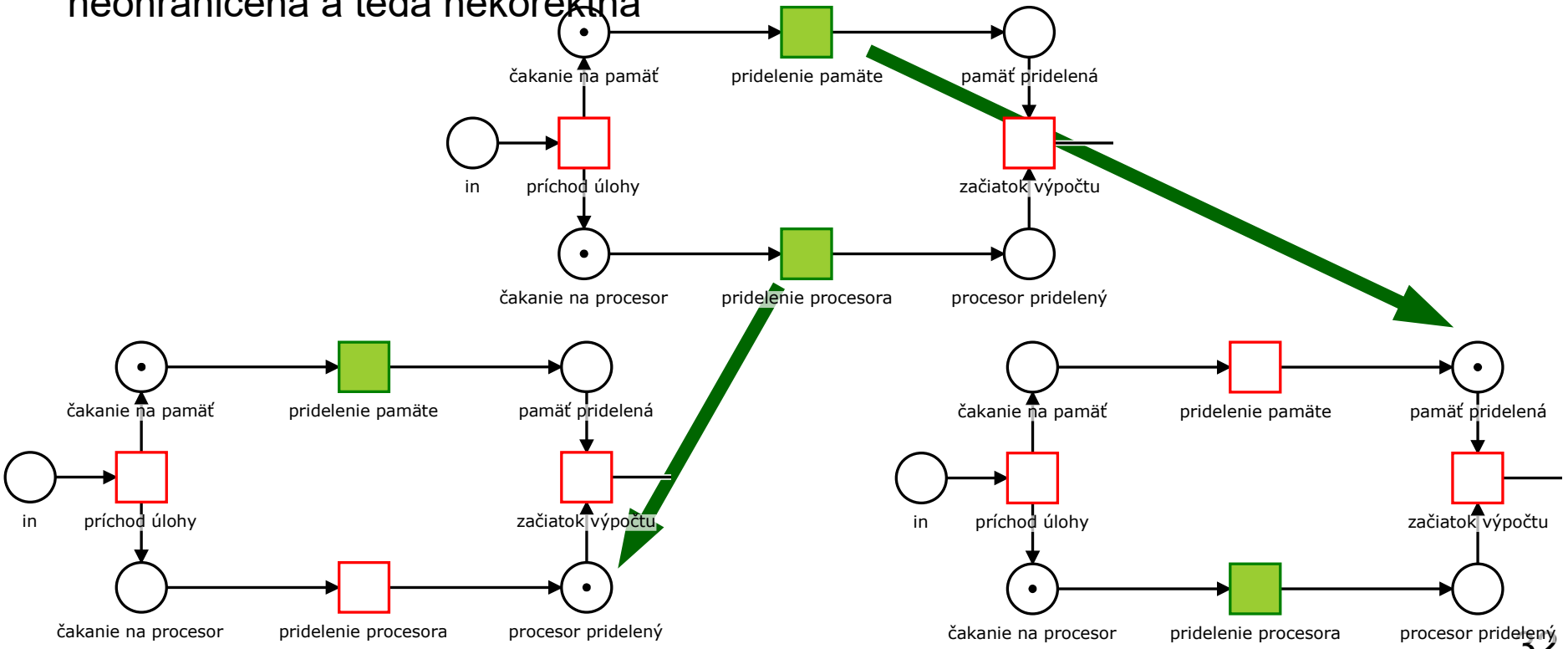


Workflow siete –

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

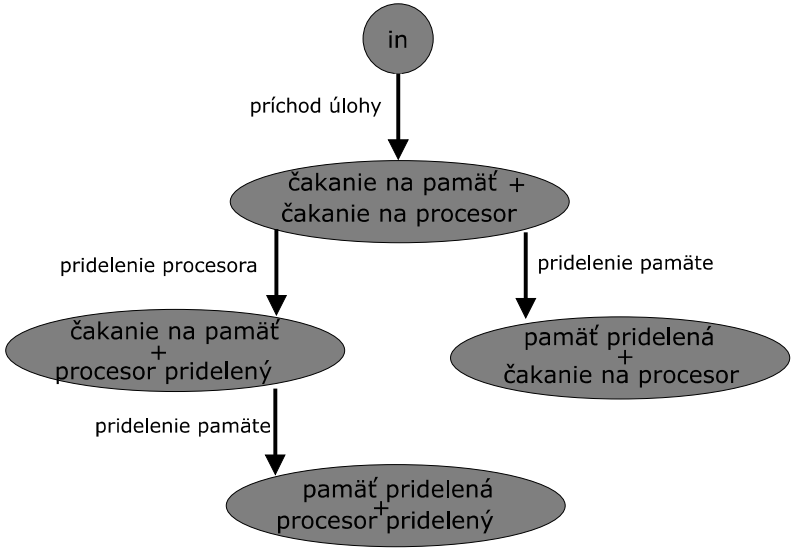


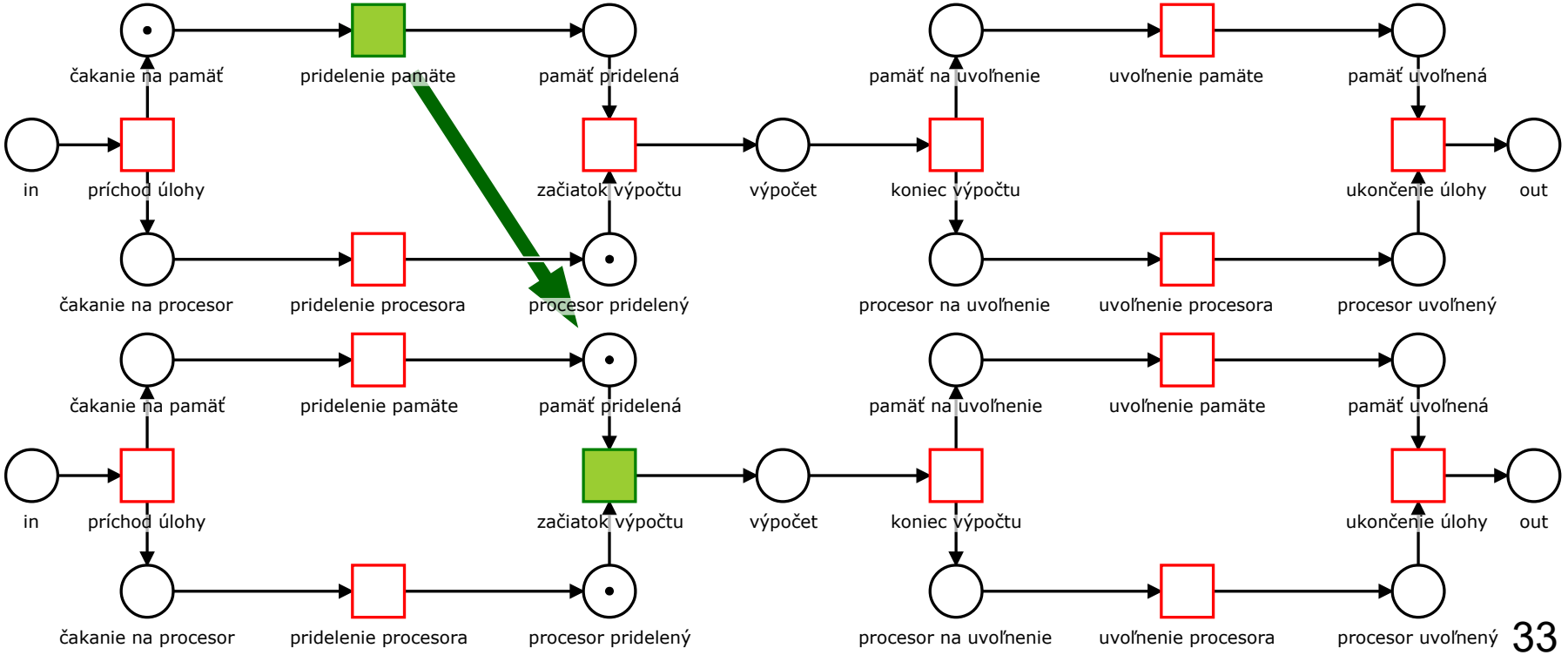


Workflow siete –

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

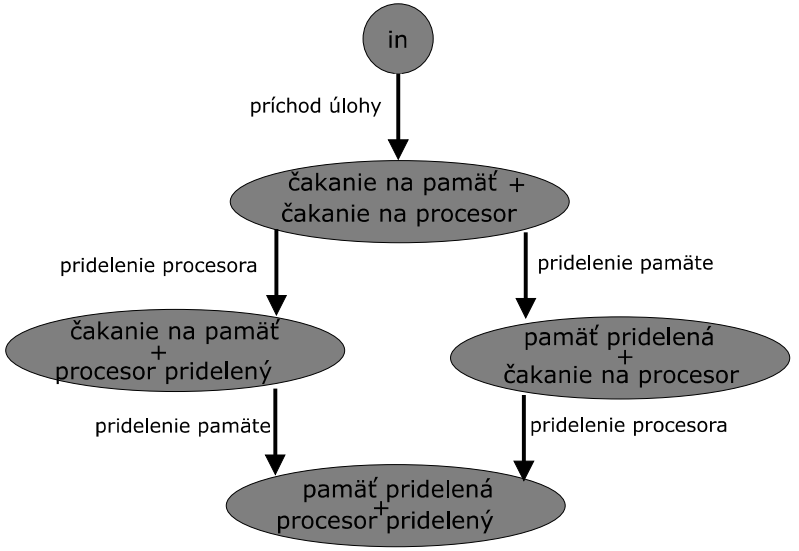


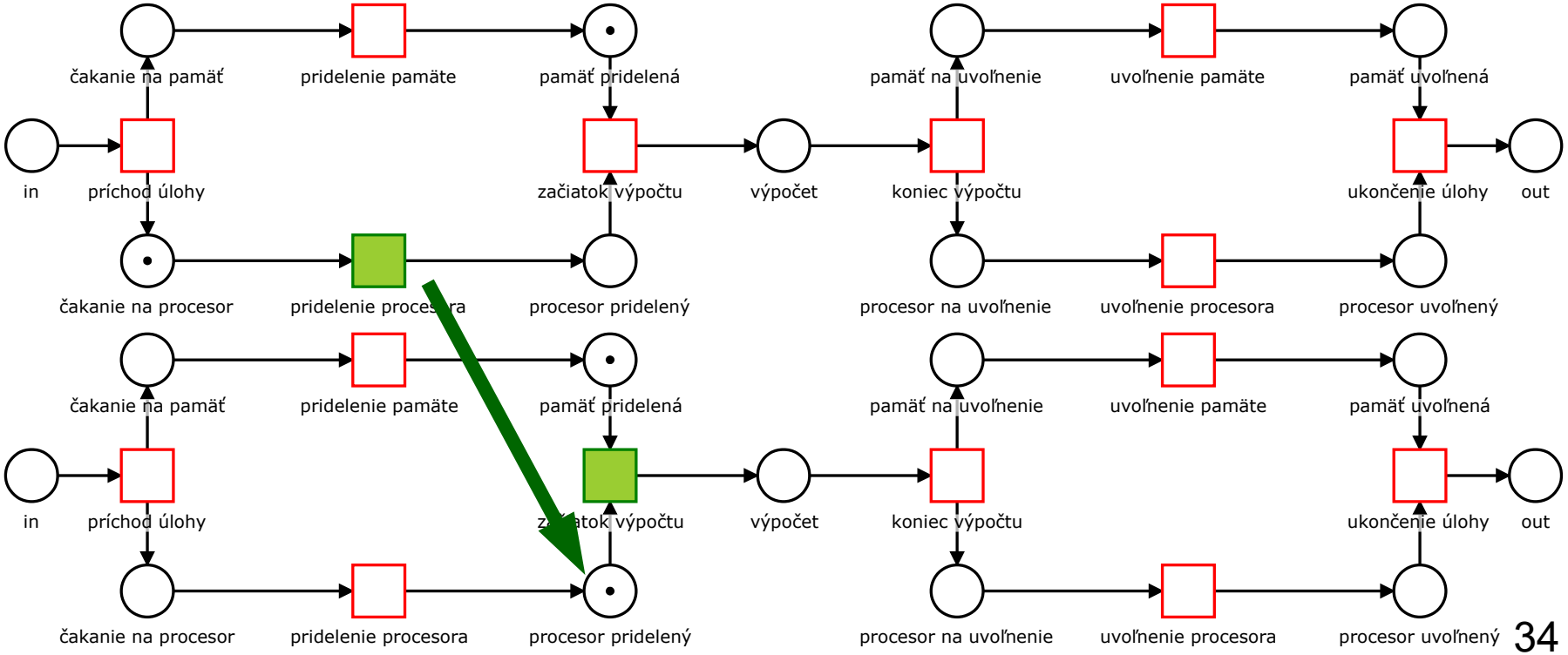


Workflow siete –

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

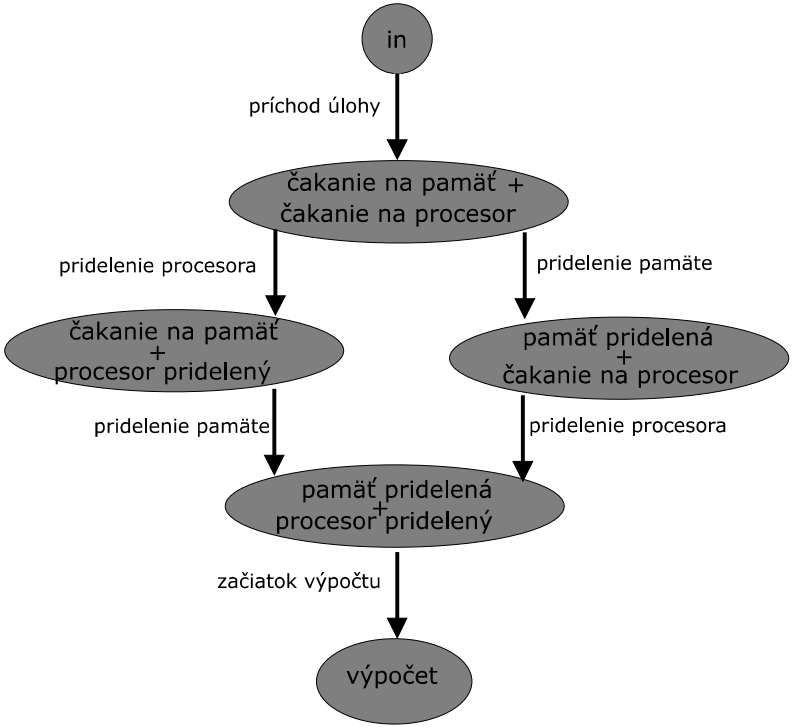




Workflow siete –

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

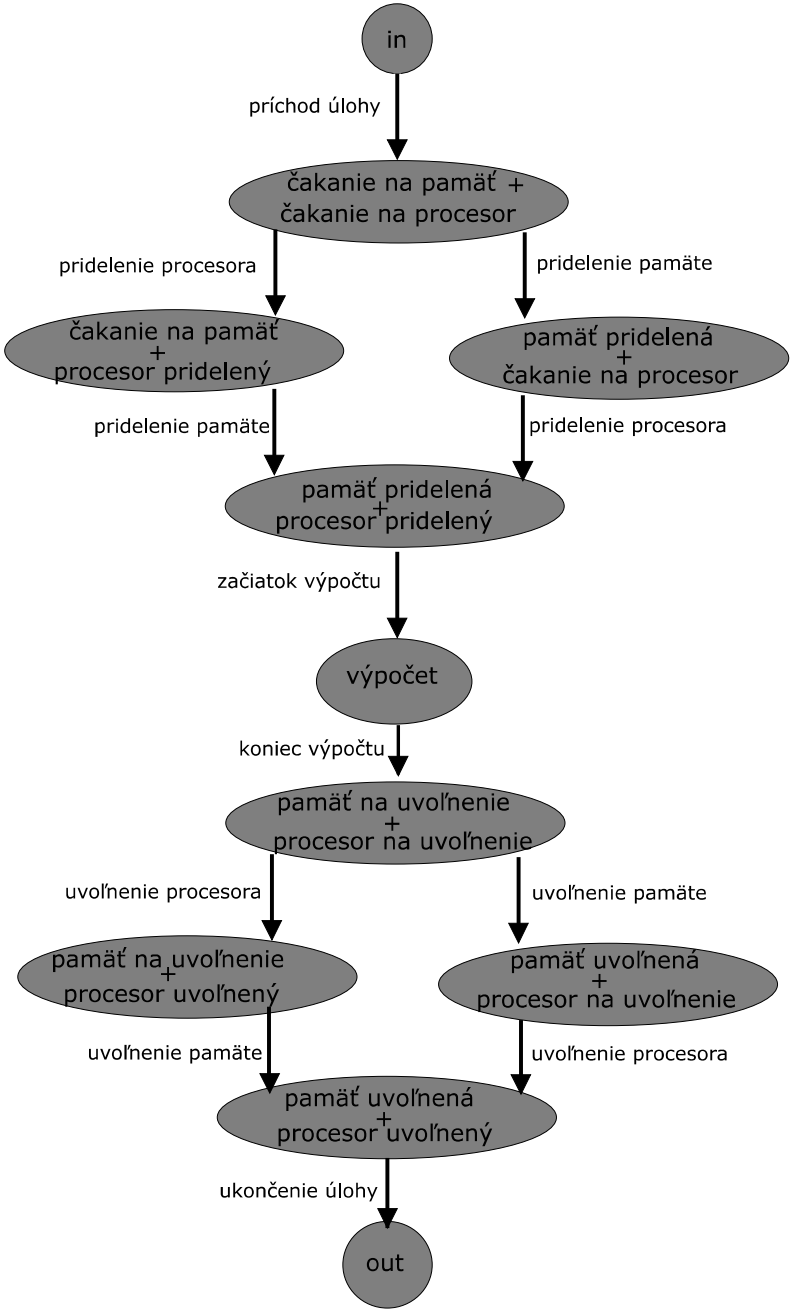
ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná



Workflow siete –

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

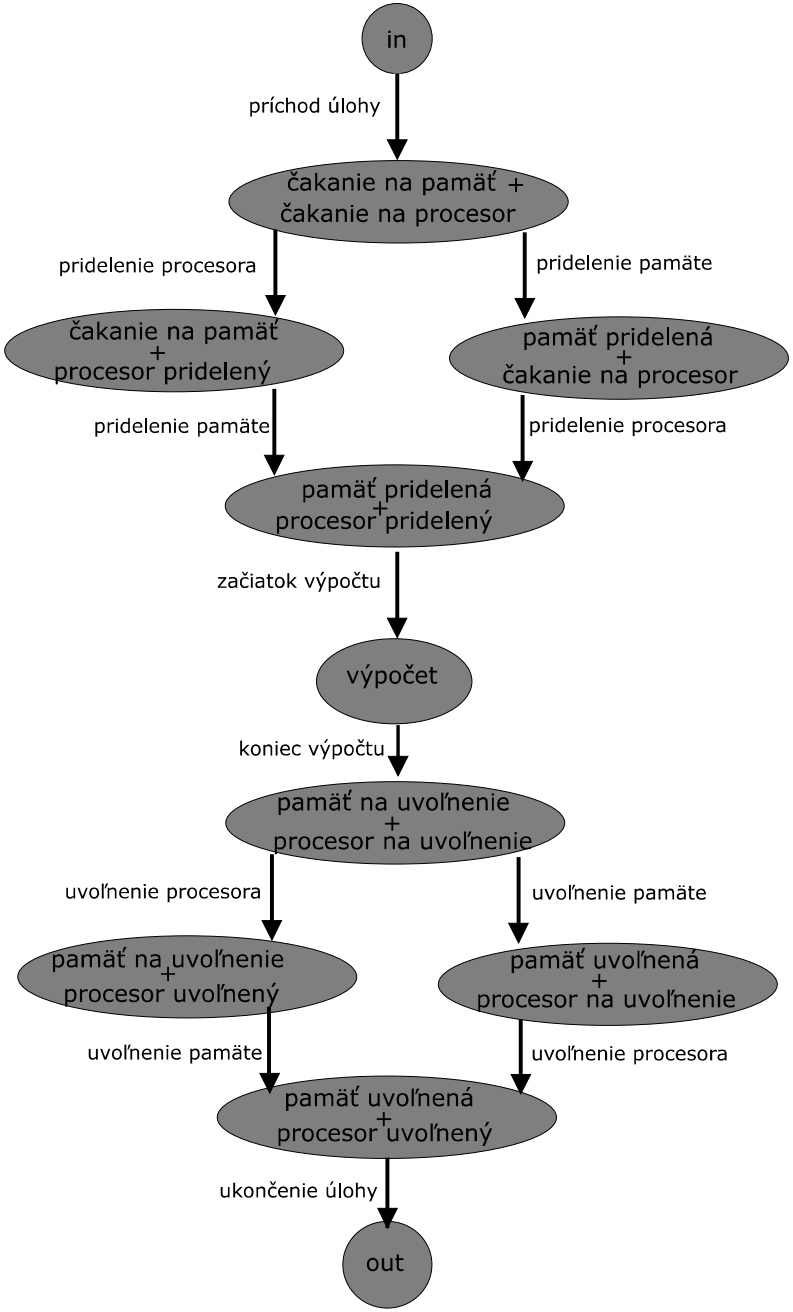


Workflow siete –

Algoritmus na overenie korektnosti – začni s počiatočným značkovaním a generuj graf dosiahnuteľnosti postupným spúšťaním prechodov

ak niektorý z vrcholov má menšieho prechodcu – zastav a vráť sieť je neohraničená a teda nekorektná

Ak značkovanie out je dosiahnuteľné a jeho predchodcovia sú všetky dosiahnuteľné značkovania a žiadne iné so značkou v out nie je dosiahnuteľné, sieť je korektná

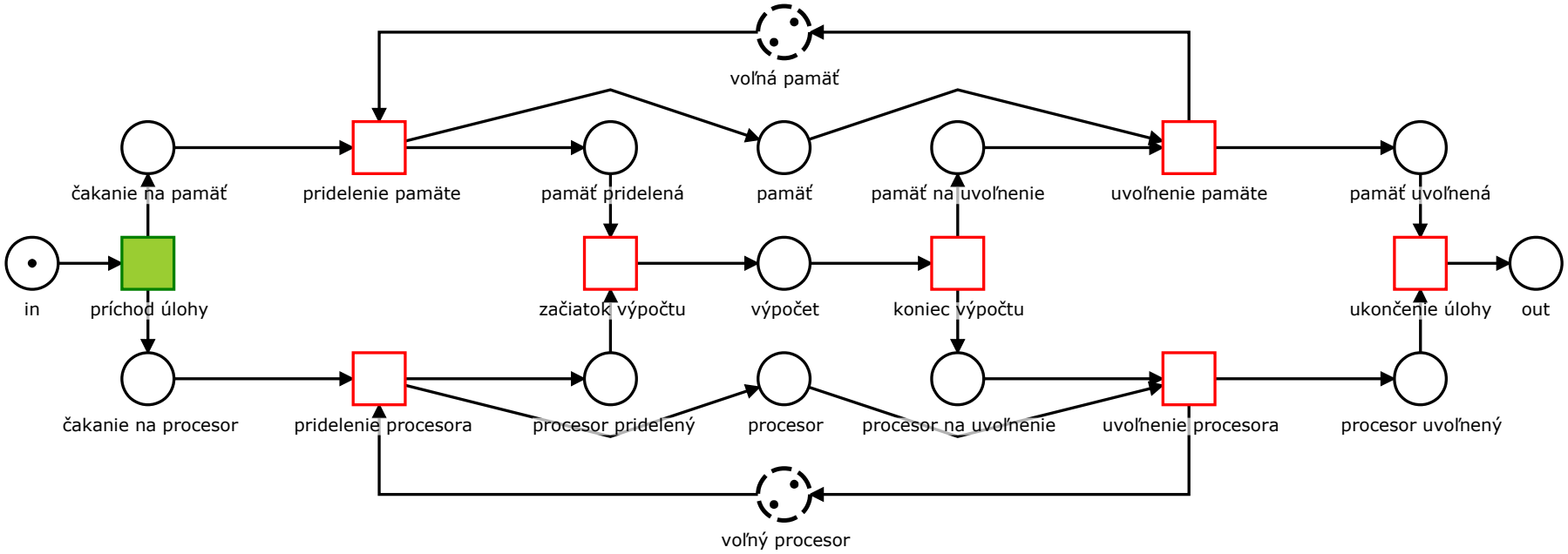


Petriho siete a workflow siete so statickými miestami

Statické miesta modelujú zdroje,

zvyšné miesta nazývame dynamické

voľná pamäť, voľný procesor



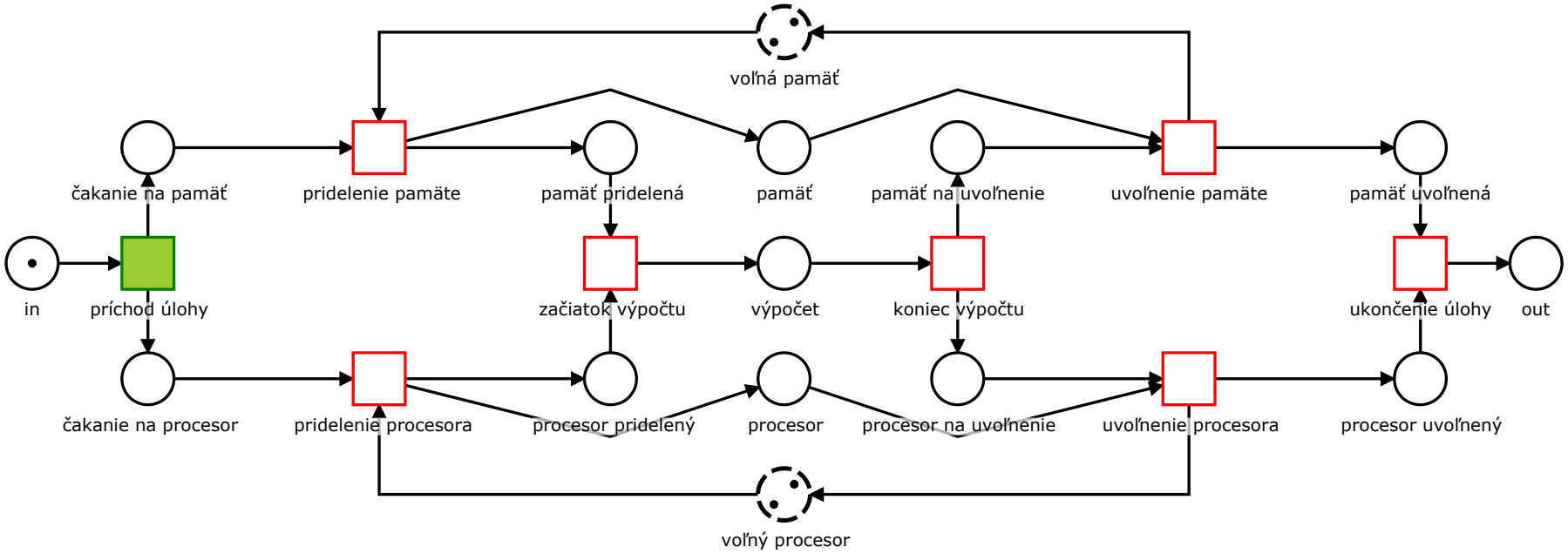
Petriho siete a workflow siete so statickými miestami

Určené siete – každé statické miesto s ma také komplementárne miesto ds, ktoré modeluje používané zdroje, že pre každý prechod t platí:

I(ds,t) – O(ds,t) = O(s,t) – I(s,t)

Určujúce miesto statického miesta voľná pamäť je miesto pamäť

Určujúce miesto statického miesta voľný procesor je miesto procesor



Petriho siete a workflow siete so statickými miestami

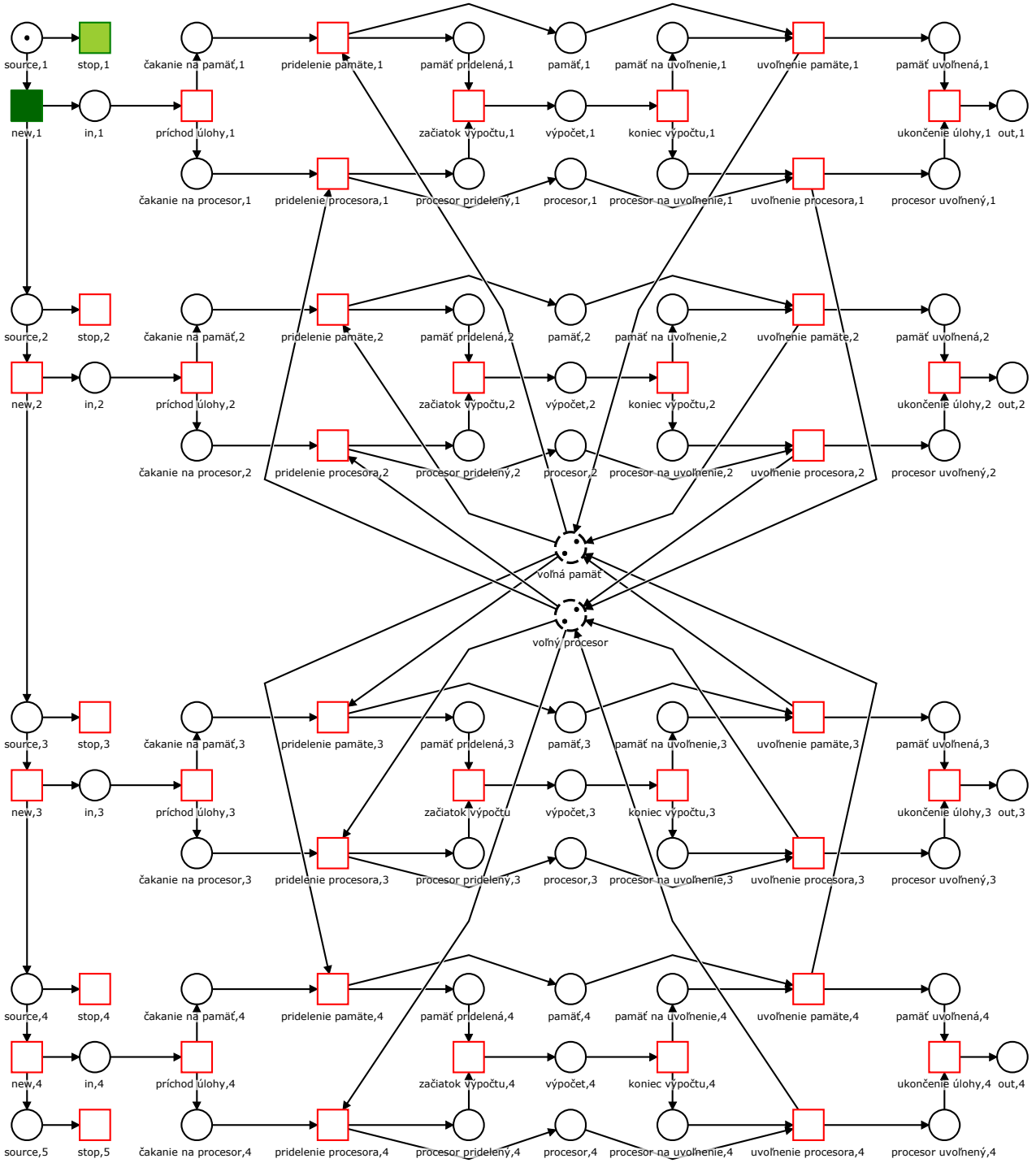
Určené siete – korektnosť je definovaná aj overovaná rovnako ako bez statických miest

Vykonávané siete workflow sietí so statickými miestami (runtime siete)

Modelujú proces počas behu, každá inštancia vytvára vlastnú kópiu workflow siete, statické miesta sa zdieľajú, inštancie vytvára konštruktor

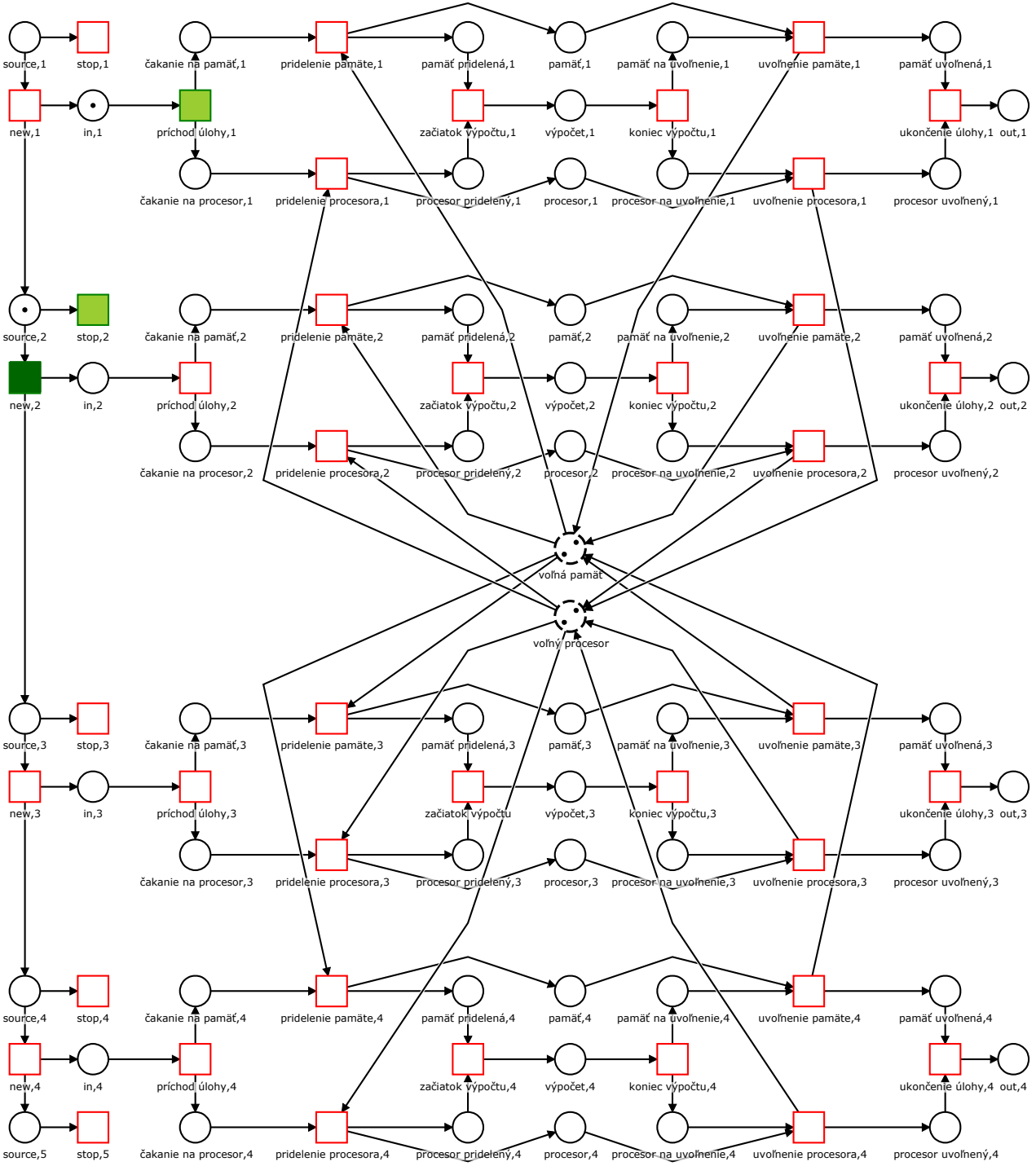
Vykonávané siete

new,1



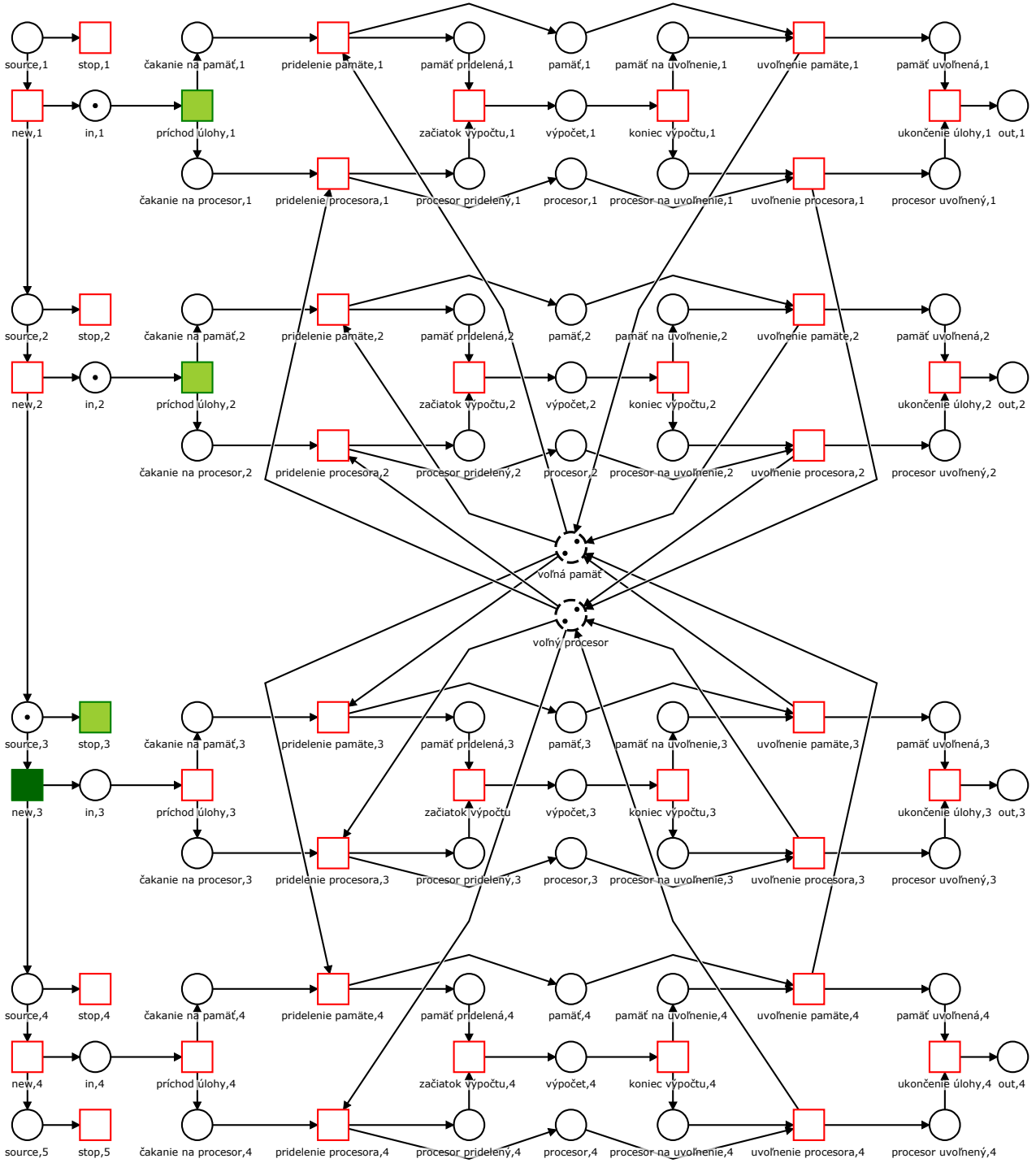
Vykonávané siete

new,2



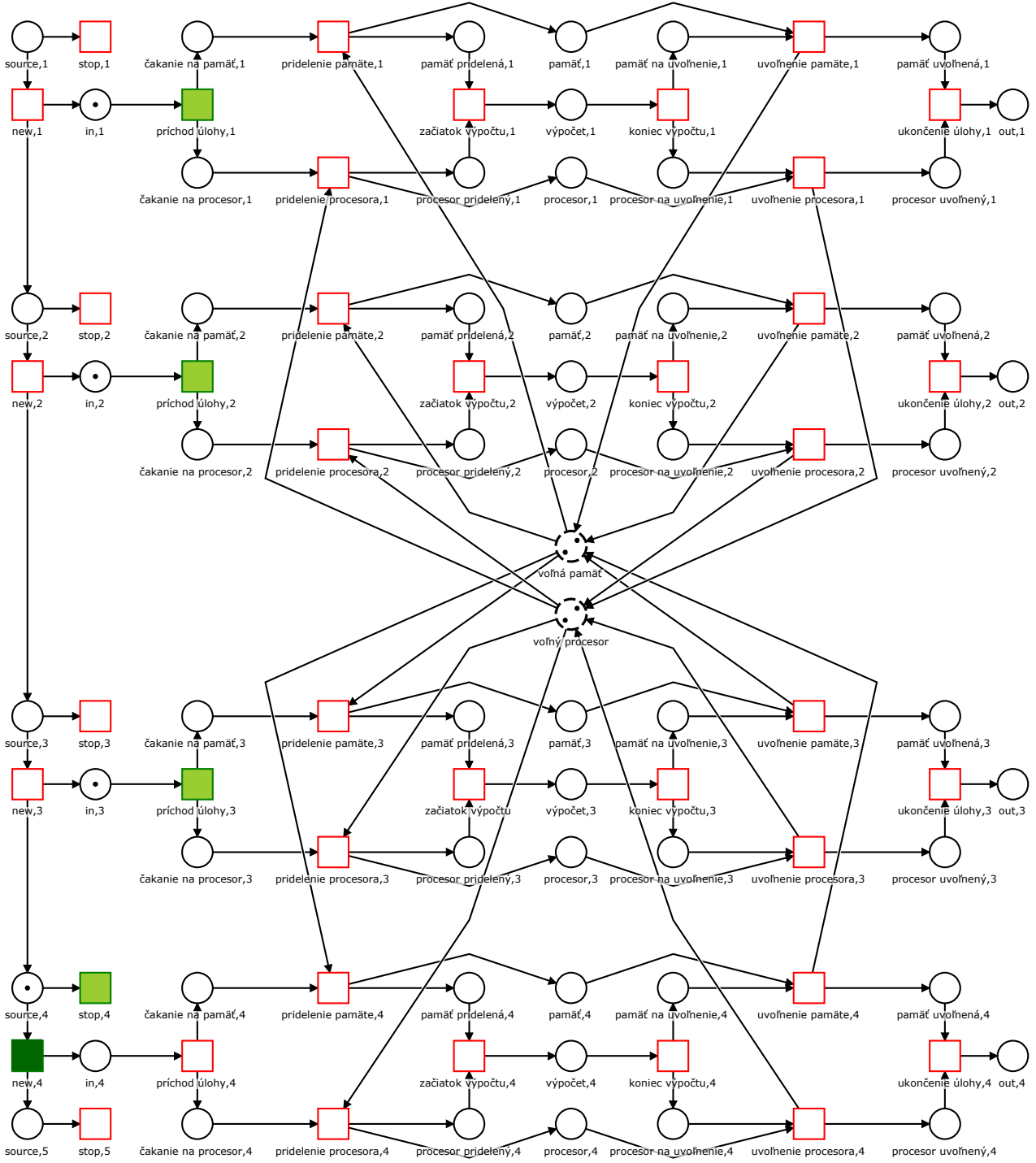
Vykonávané siete

new,3



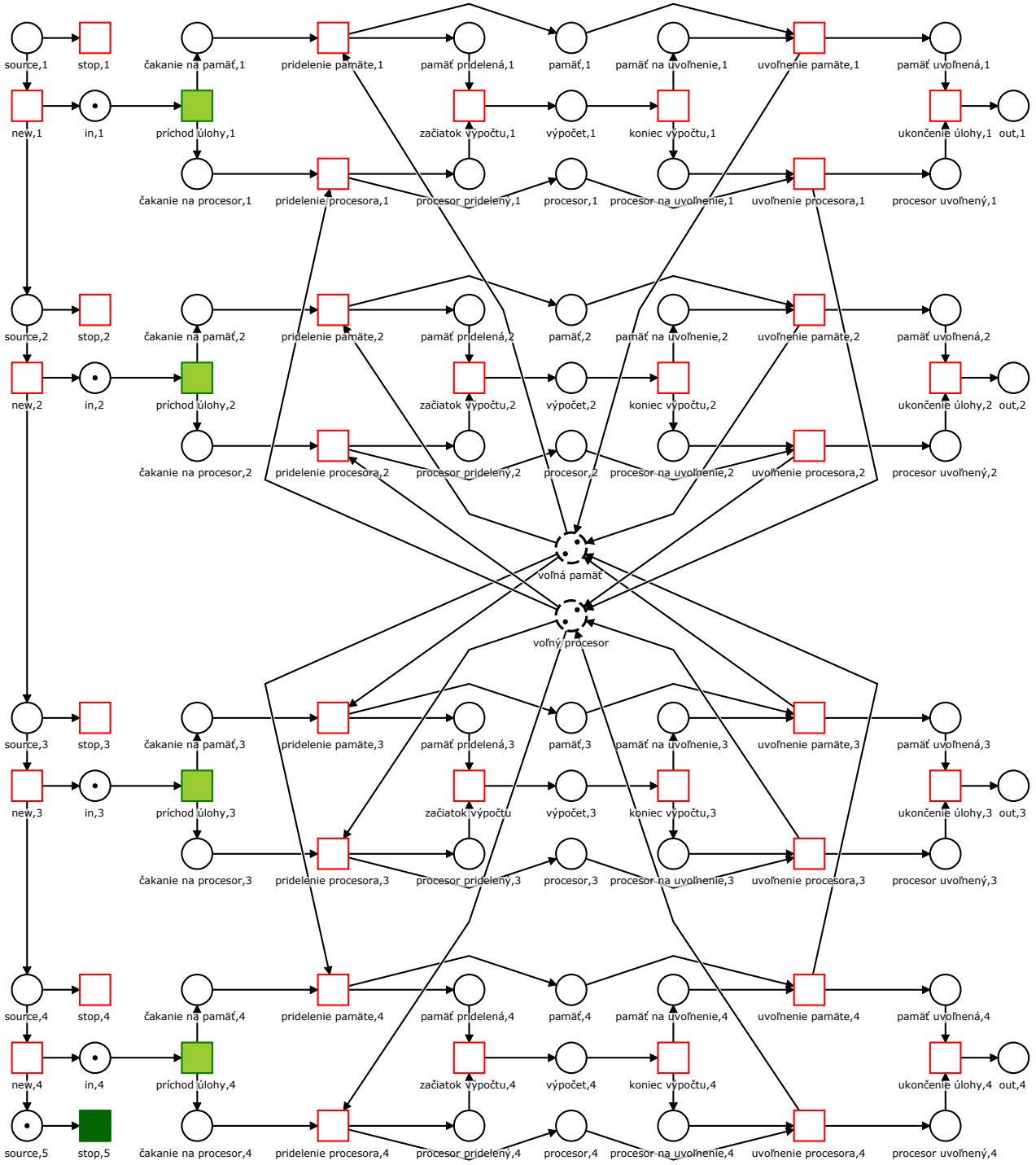
Vykonávané siete

new,4



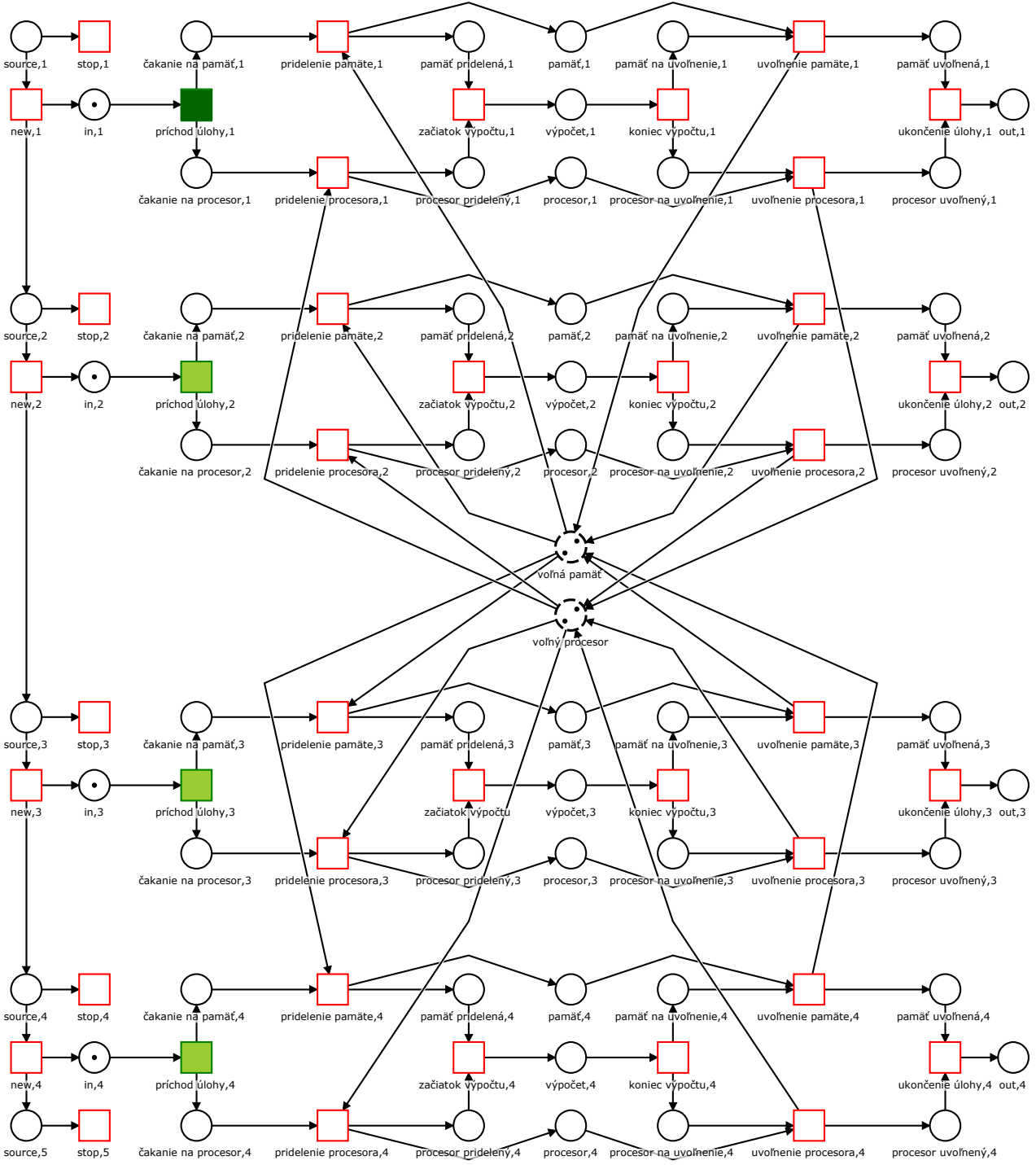
Vykonávané siete

stop,5



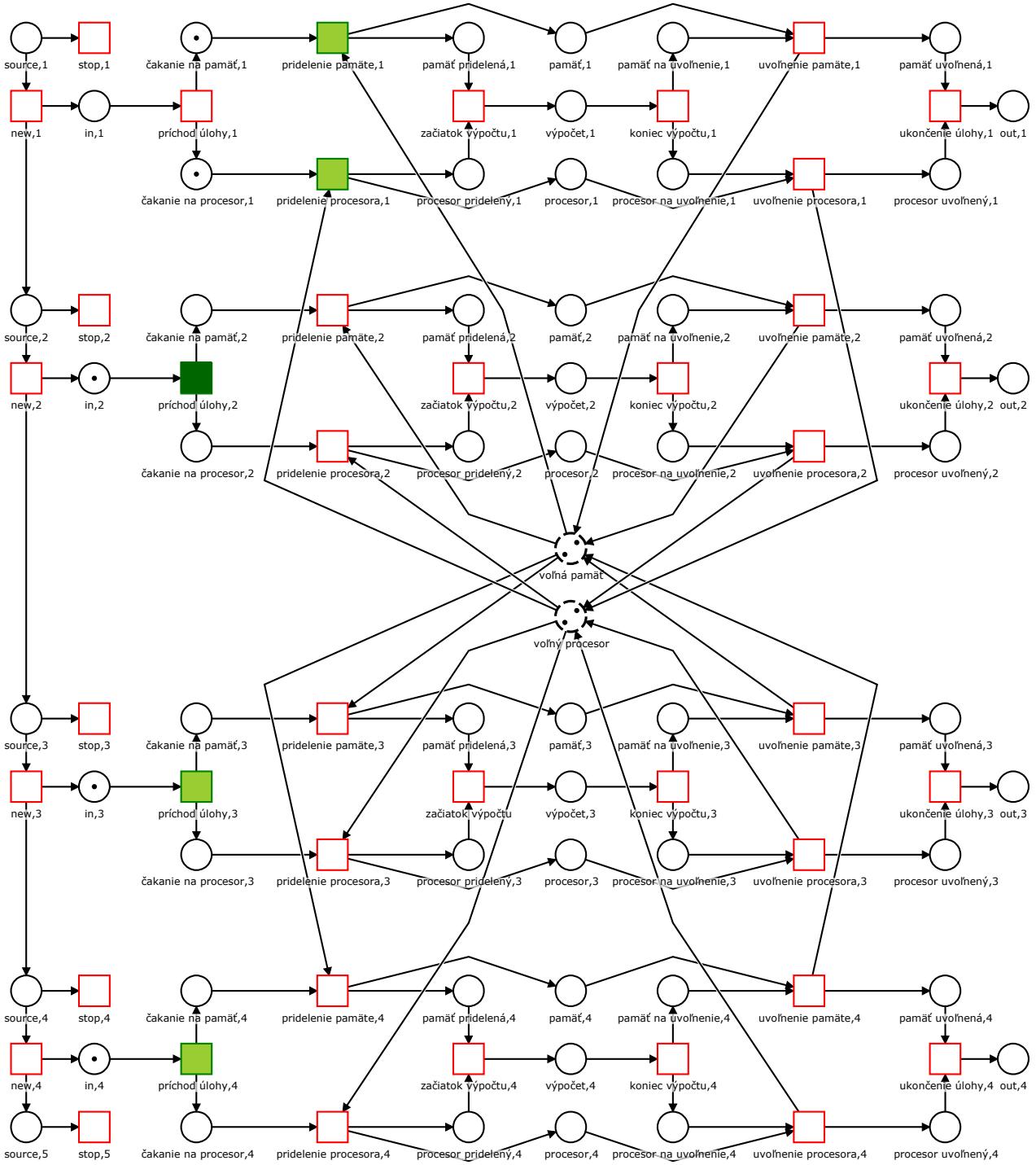
Vykonávané siete

príchod úlohy,1



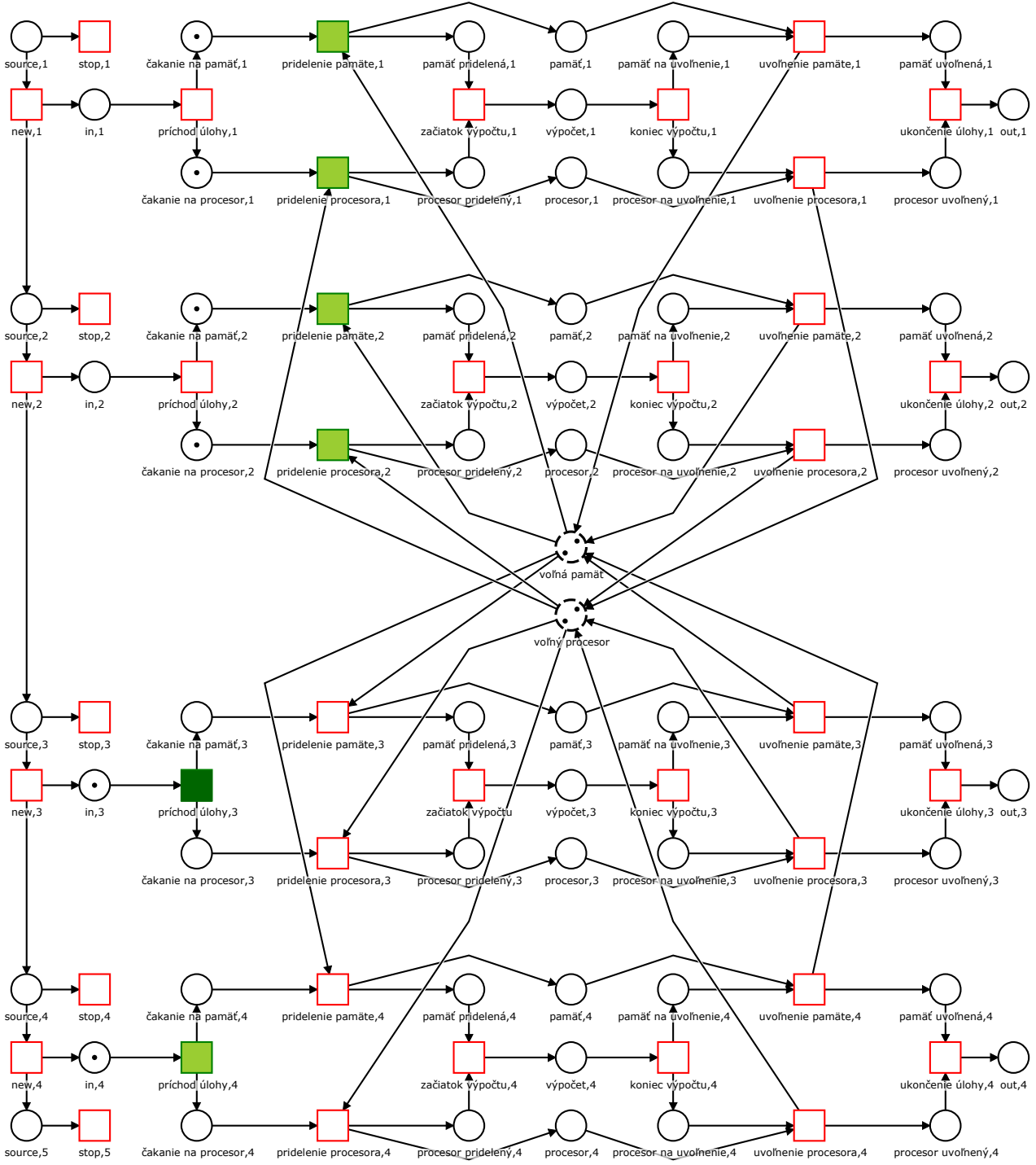
Vykonávané siete

príchod úlohy,2



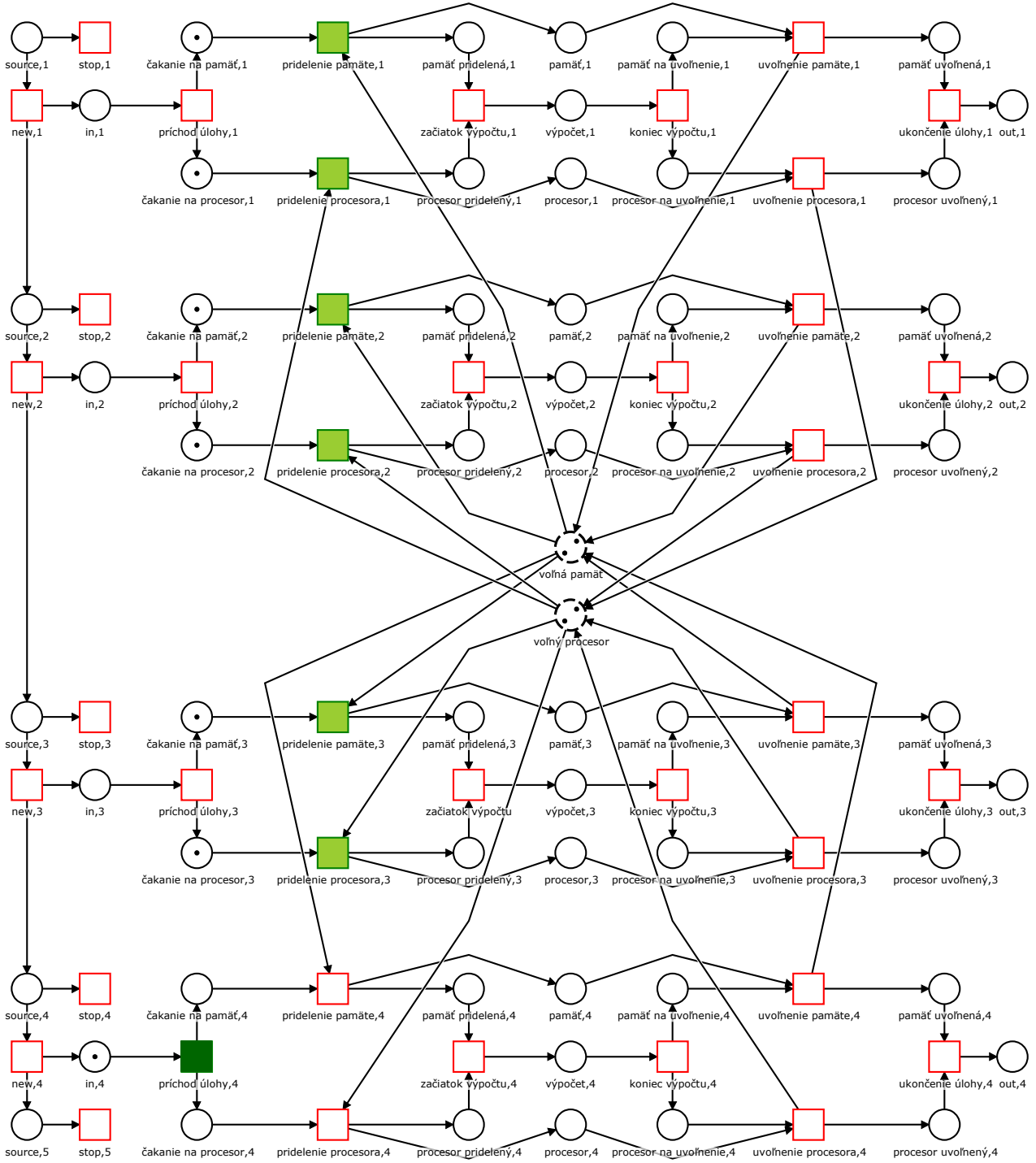
Vykonávané siete

príchod úlohy,3



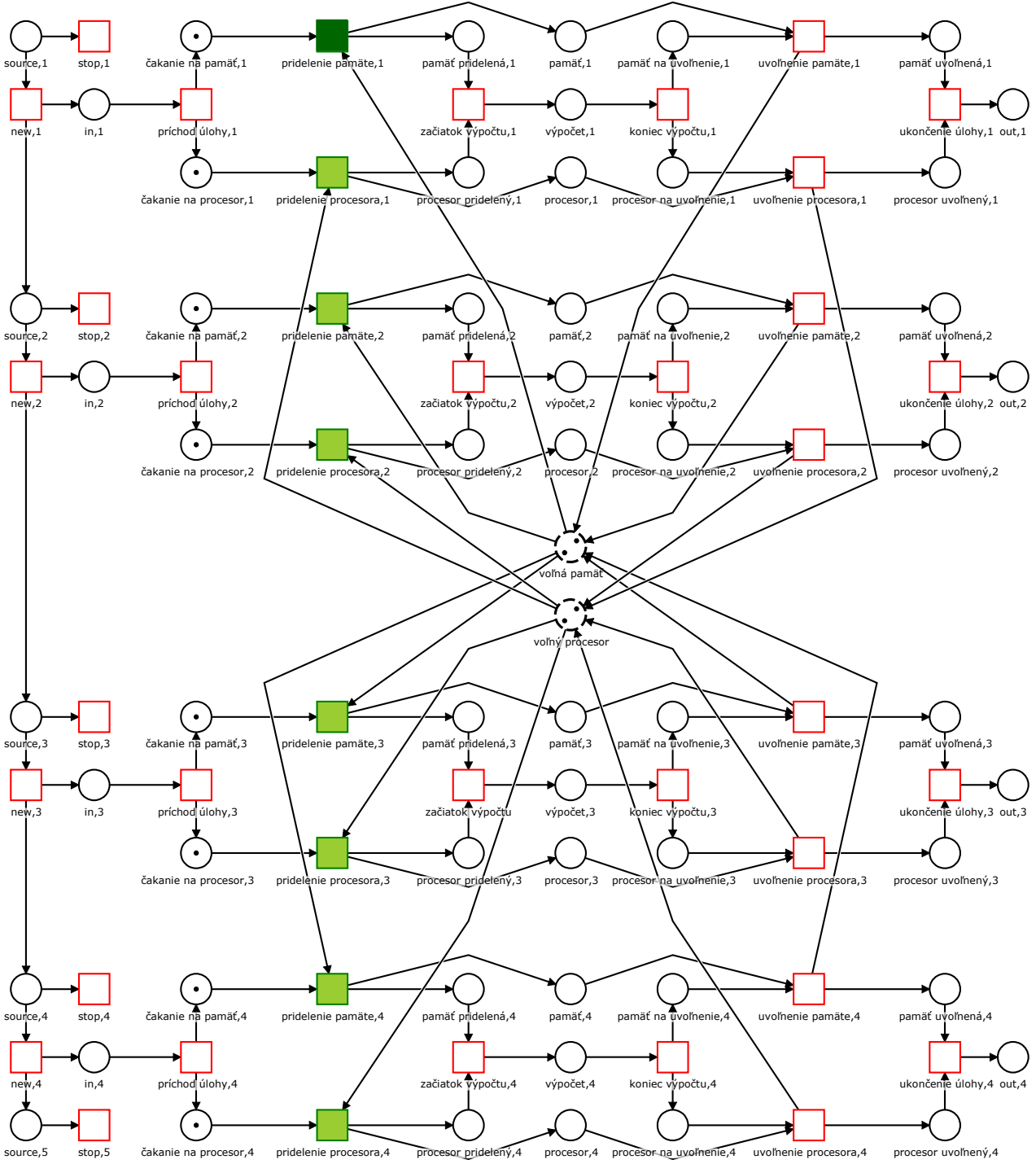
Vykonávané siete

príchod úlohy,4



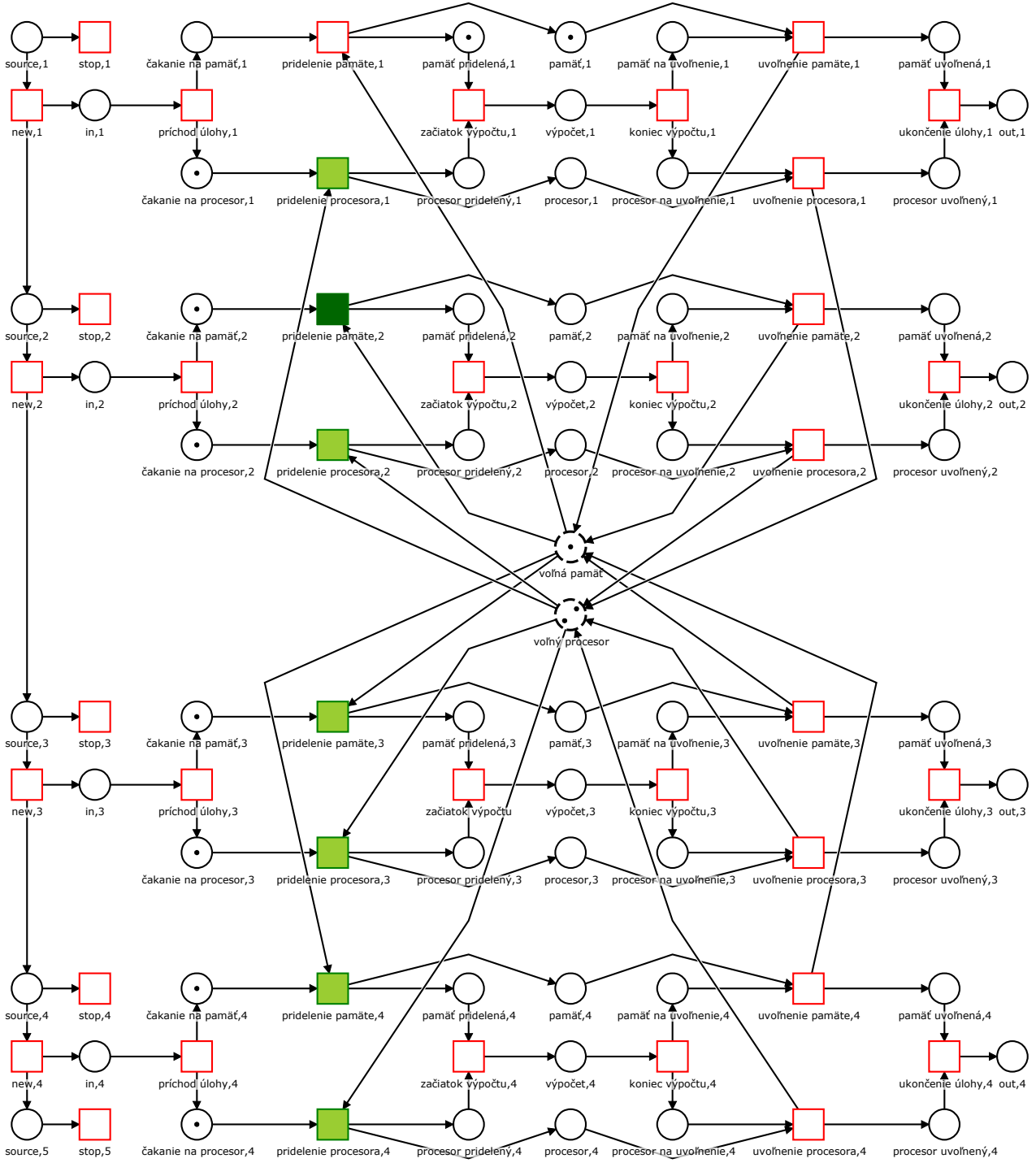
Vykonávané siete

pridelenie pamäte,1



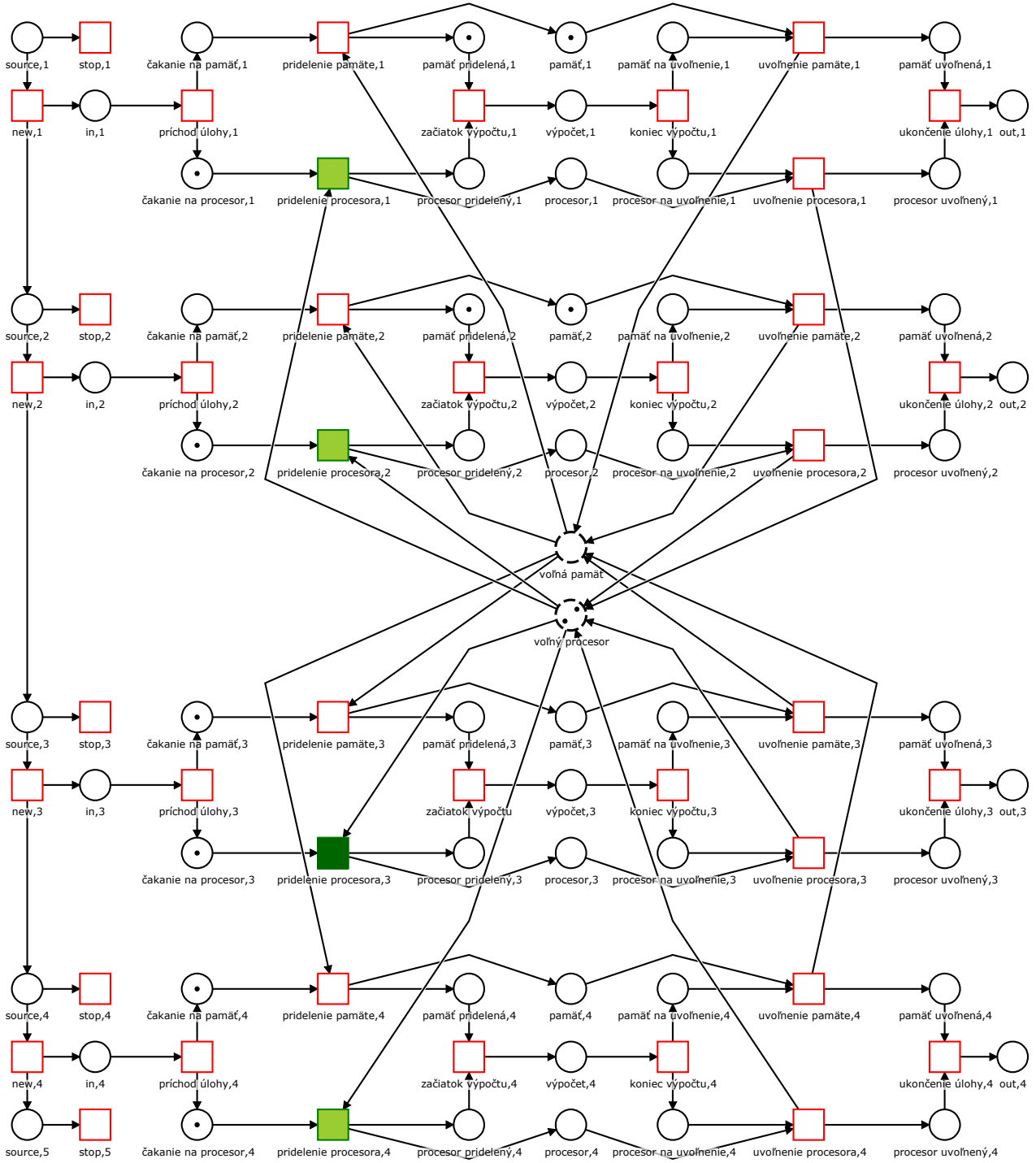
Vykonávané siete

pridelenie pamäte,2



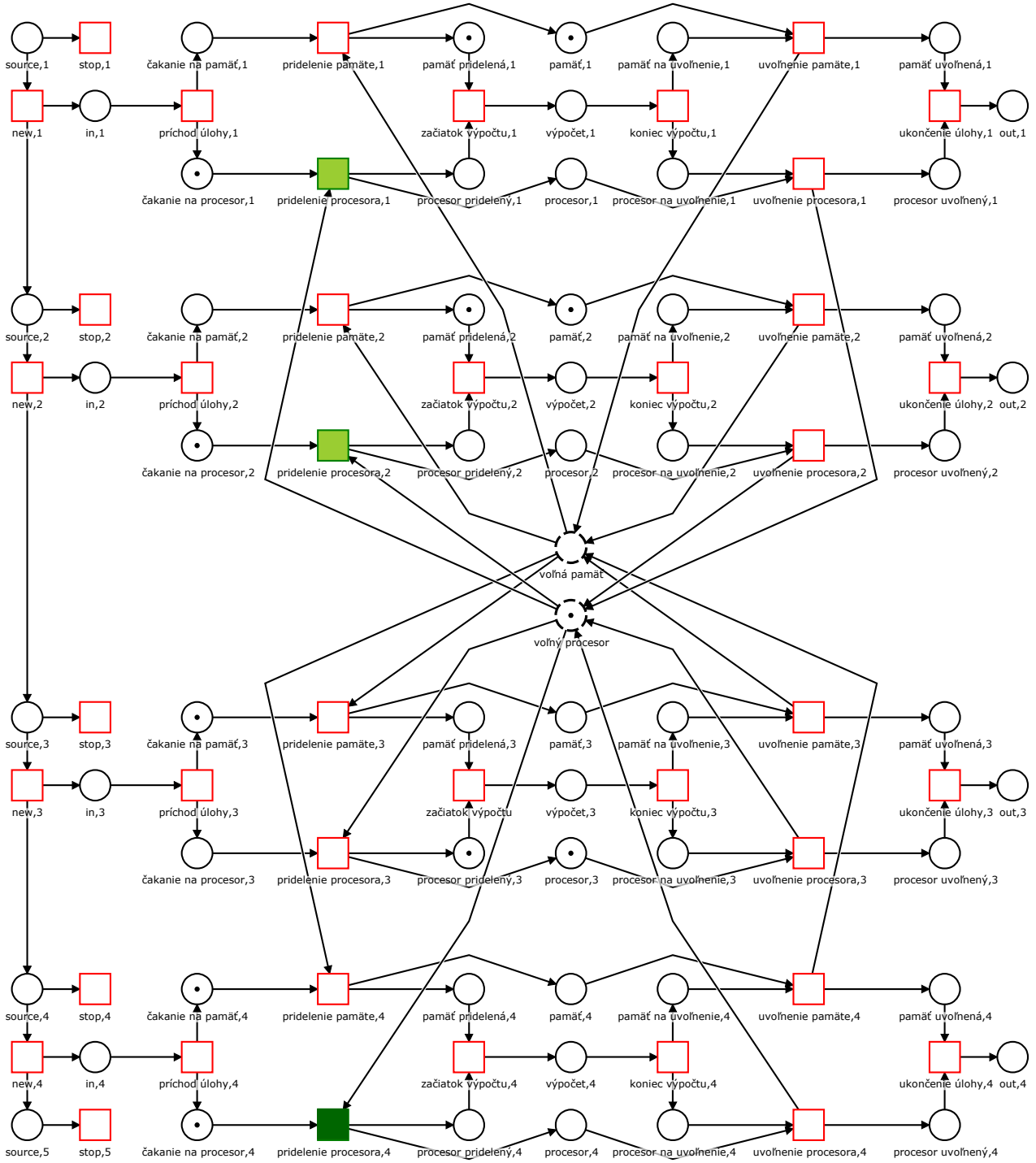
Vykonávané siete

pridelenie procesora,3



Vykonávané siete

pridelenie procesora,4

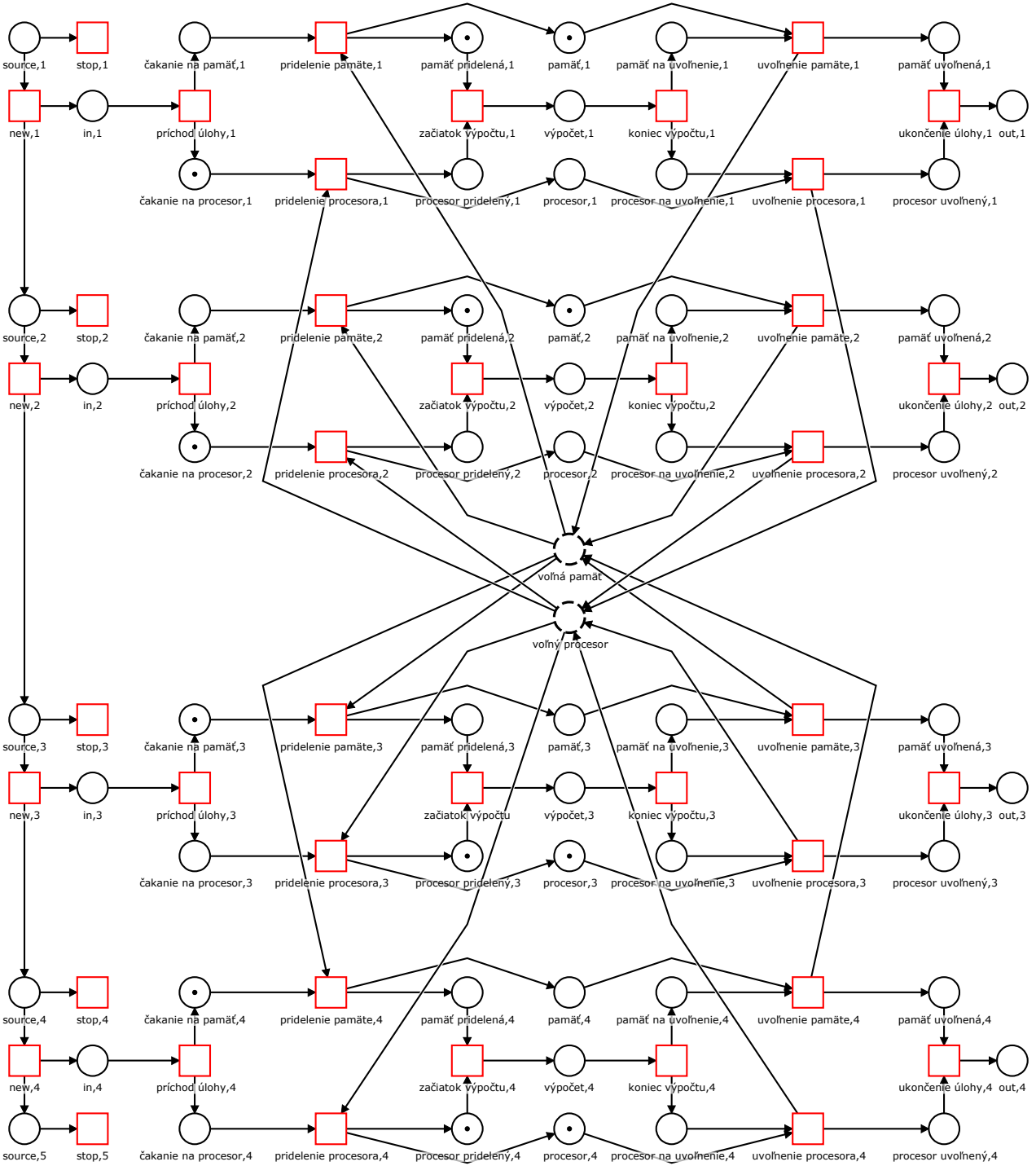


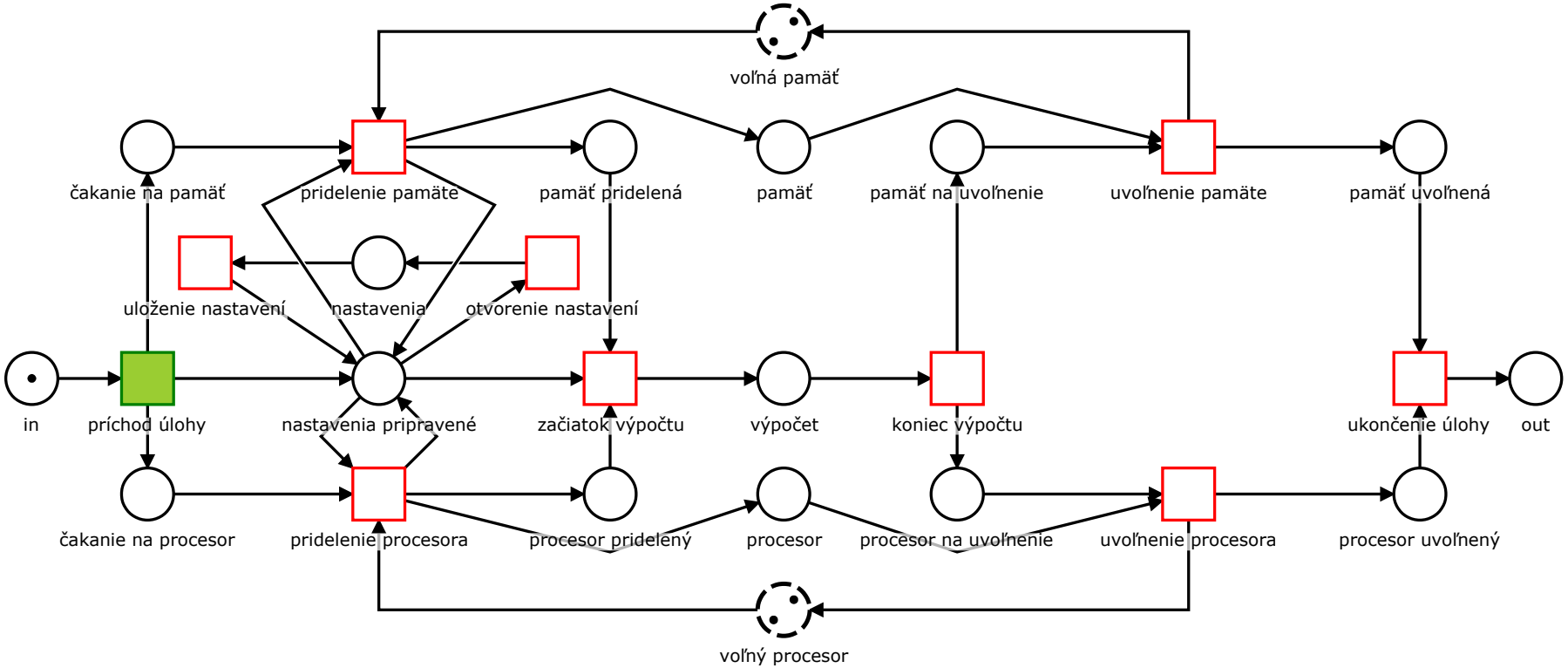
Vykonávané siete

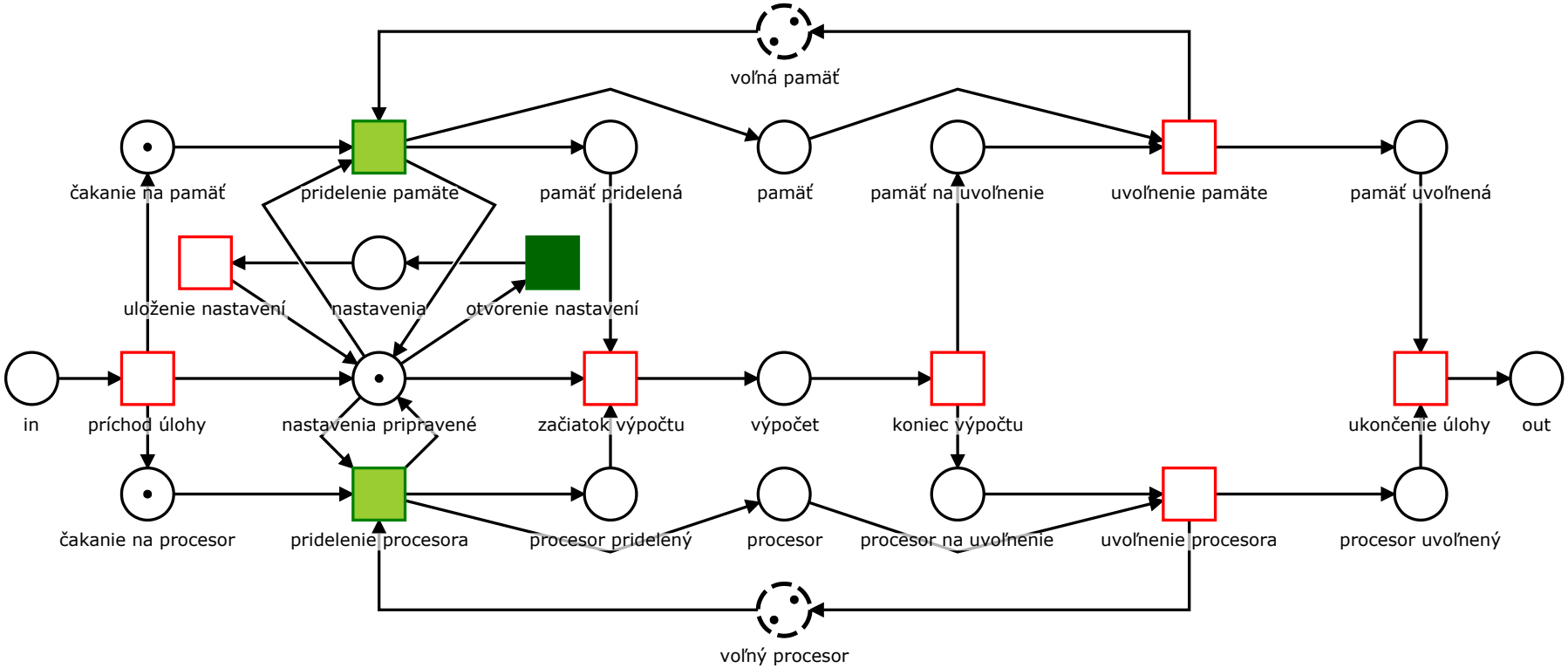
Uviaznutie -

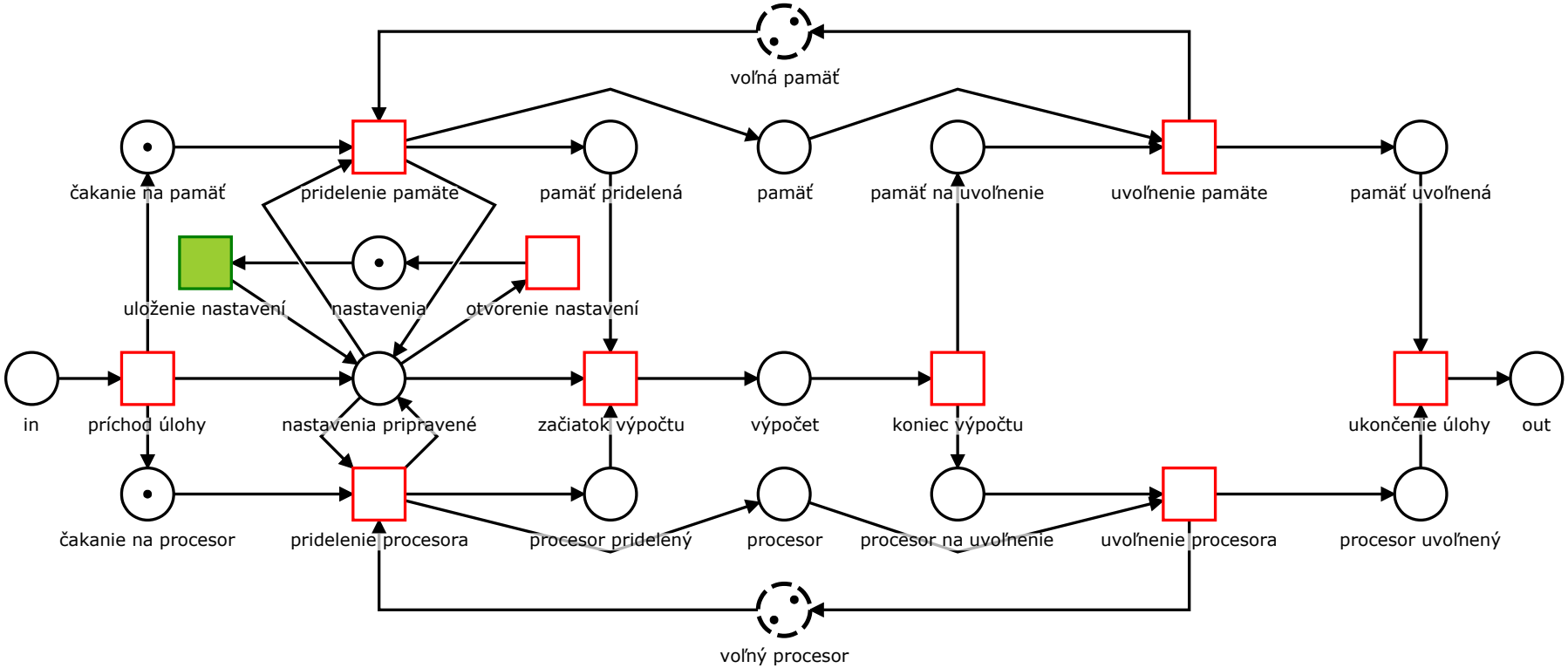
Nie je možné ukončiť inštancie

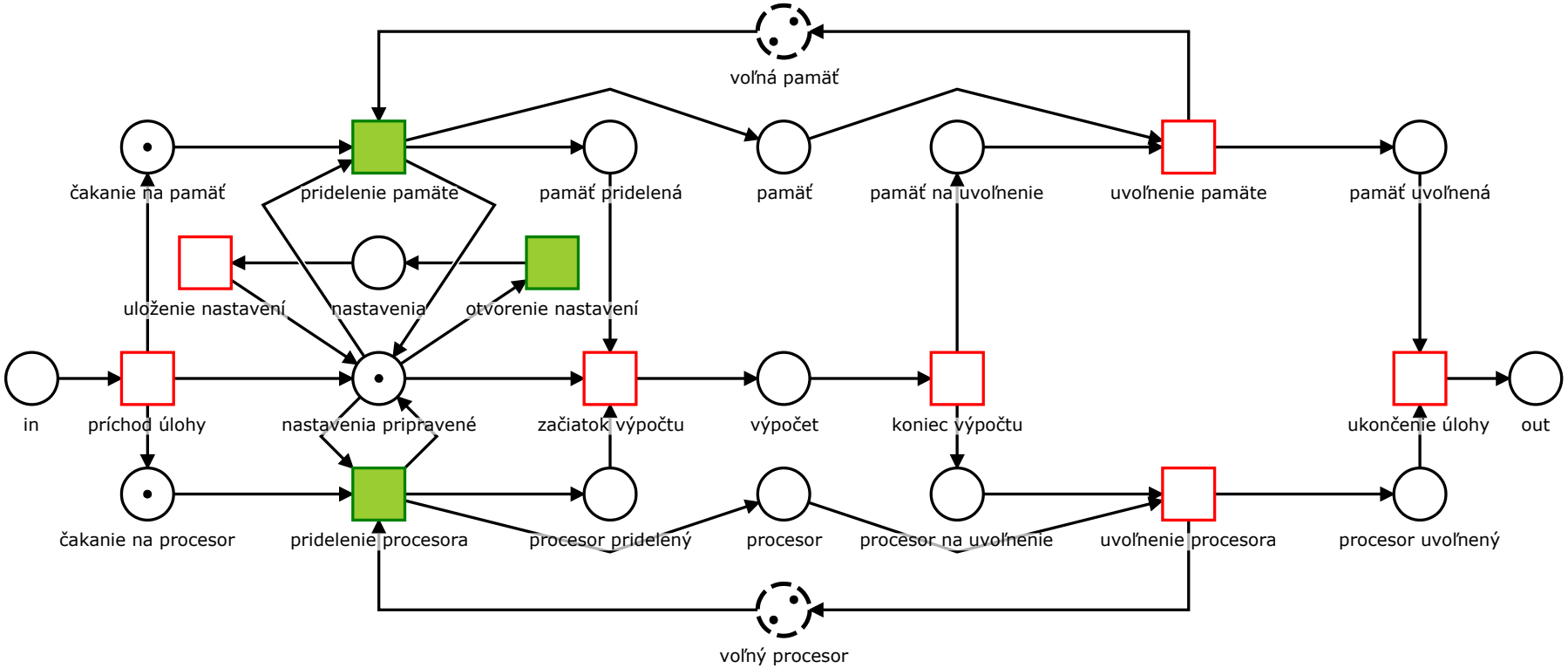
Zamrznutie je také uviaznutie, v ktorom nie je možné spustiť žiadny prechod




voľná pamäť
čakanie na pamäť
pridelenie pamäte
pamäť pridelená
pamäť
pamäť na uvoľnenie
uvoľnenie pamäte
pamäť uvoľnená
uloženie nastavení
nastavenia
otvorenie nastavení
in
príchod úlohy
nastavenia pripravené
začiatok výpočtu
výpočet
koniec výpočtu
ukončenie úlohy
out
čakanie na procesor
pridelenie procesora
procesor pridelený
procesor
procesor na uvoľnenie
uvoľnenie procesora
procesor uvoľnený
voľný procesor

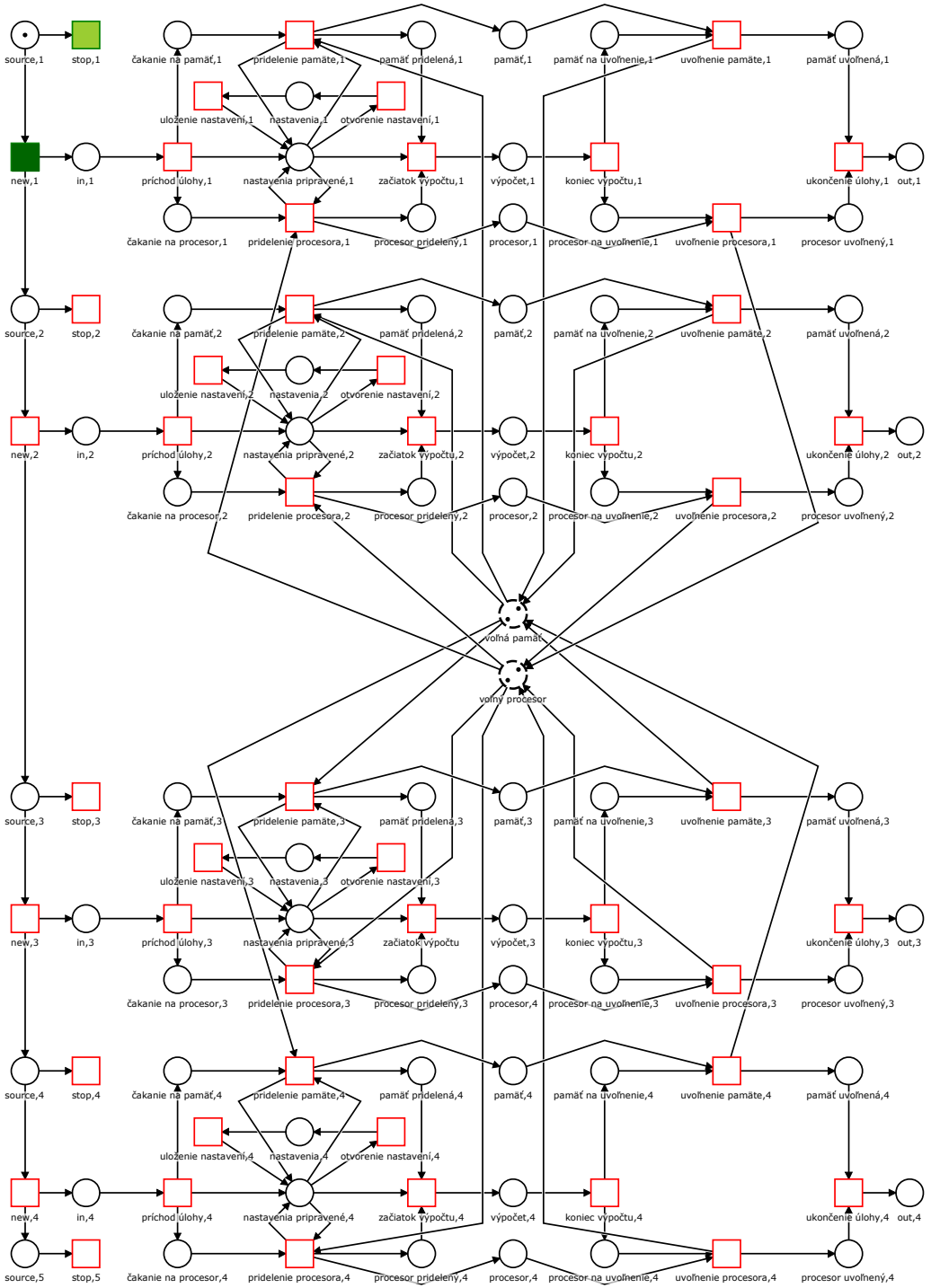

voľná pamäť
čakanie na pamäť
pridelenie pamäte
pamäť pridelená
pamäť
pamäť na uvoľnenie
uvoľnenie pamäte
pamäť uvoľnená
uloženie nastavení
nastavenia
otvorenie nastavení
in
príchod úlohy
nastavenia pripravené
začiatok výpočtu
výpočet
koniec výpočtu
ukončenie úlohy
out
čakanie na procesor
pridelenie procesora
procesor pridelený
procesor
procesor na uvoľnenie
uvoľnenie procesora
procesor uvoľnený
voľný procesor


voľná pamäť
čakanie na pamäť
pridelenie pamäte
pamäť pridelená
pamäť
pamäť na uvoľnenie
uvoľnenie pamäte
pamäť uvoľnená
uloženie nastavení
nastavenia
otvorenie nastavení
in
príchod úlohy
nastavenia pripravené
začiatok výpočtu
výpočet
koniec výpočtu
ukončenie úlohy
out
čakanie na procesor
pridelenie procesora
procesor pridelený
procesor
procesor na uvoľnenie
uvoľnenie procesora
procesor uvoľnený
voľný procesor


voľná pamäť
čakanie na pamäť
pridelenie pamäte
pamäť pridelená
pamäť
pamäť na uvoľnenie
uvoľnenie pamäte
pamäť uvoľnená
uloženie nastavení
nastavenia
otvorenie nastavení
in
príchod úlohy
nastavenia pripravené
začiatok výpočtu
výpočet
koniec výpočtu
ukončenie úlohy
out
čakanie na procesor
pridelenie procesora
procesor pridelený
procesor
procesor na uvoľnenie
uvoľnenie procesora
procesor uvoľnený
voľný procesor

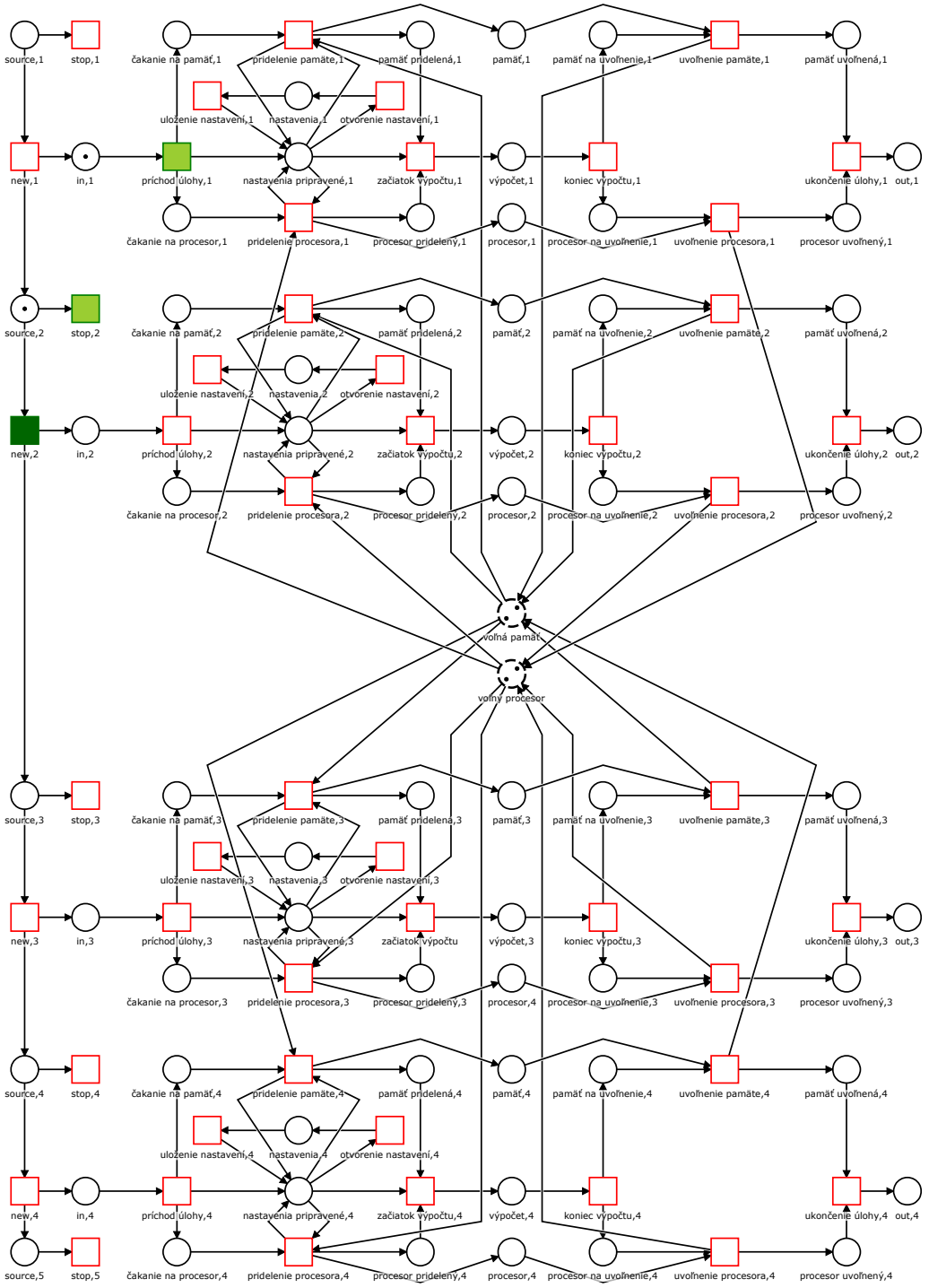
Vykonávaná sieť

new,1



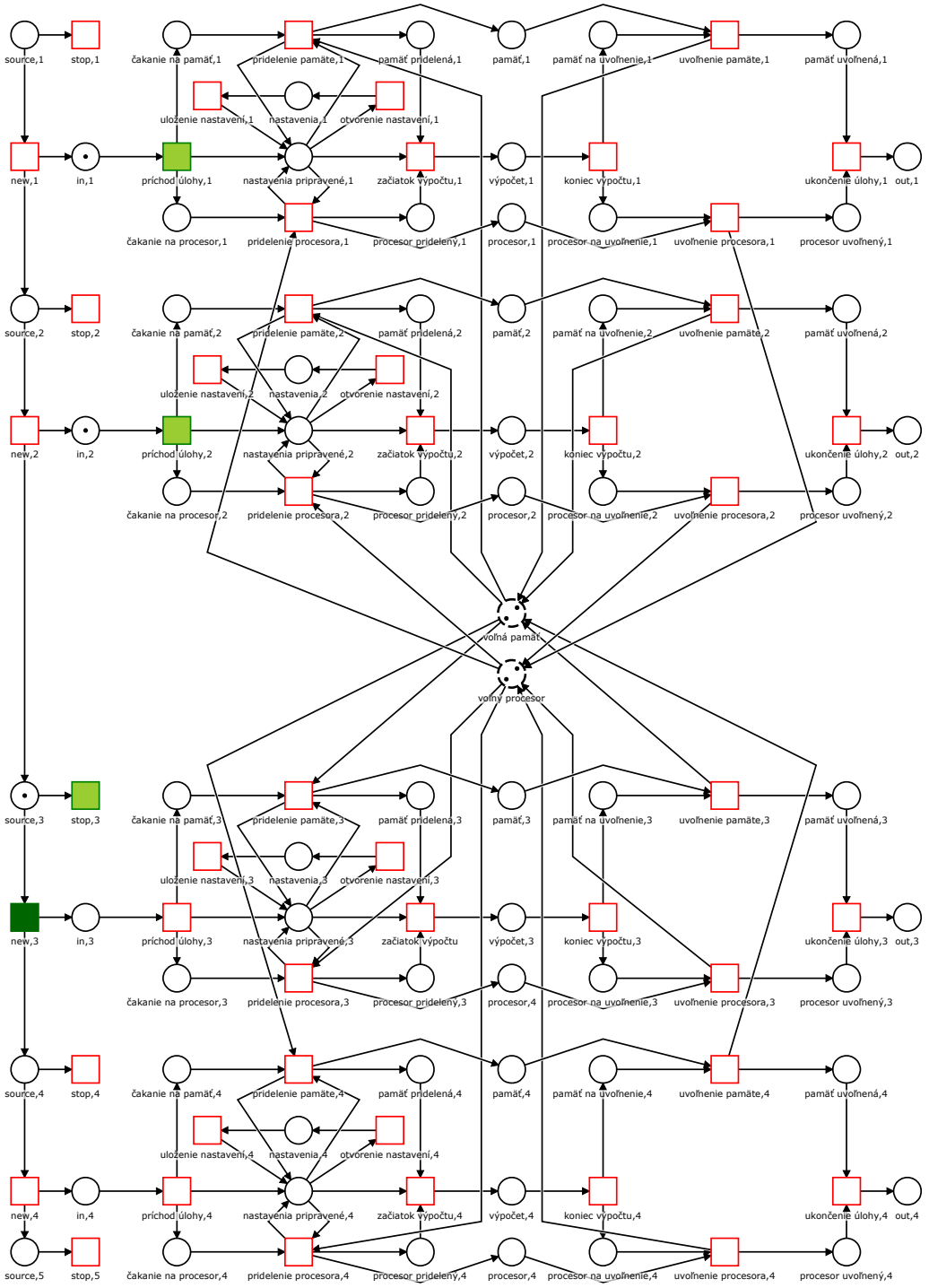
Vykonávaná sieť

new,2



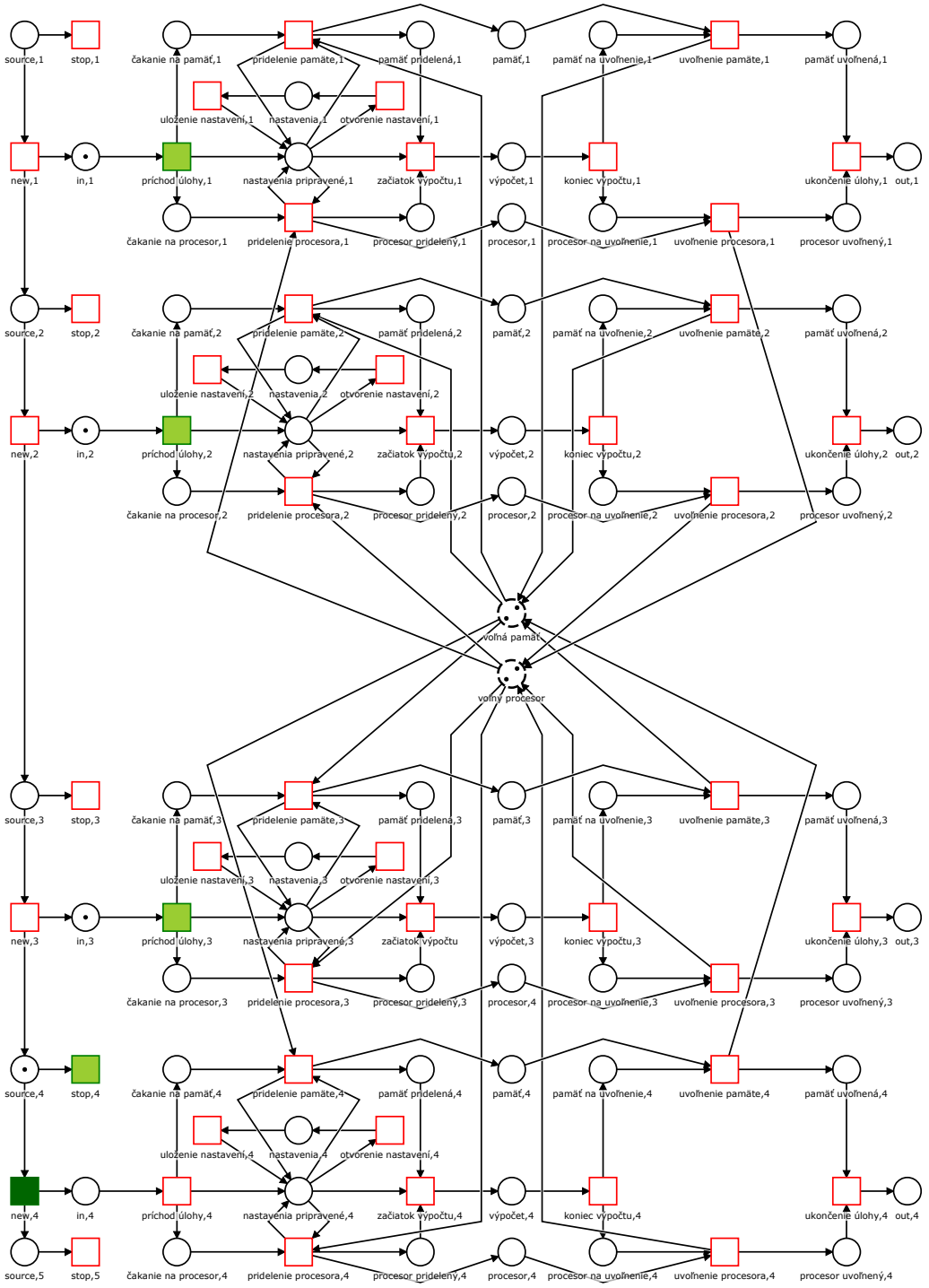
Vykonávaná sieť

new,3



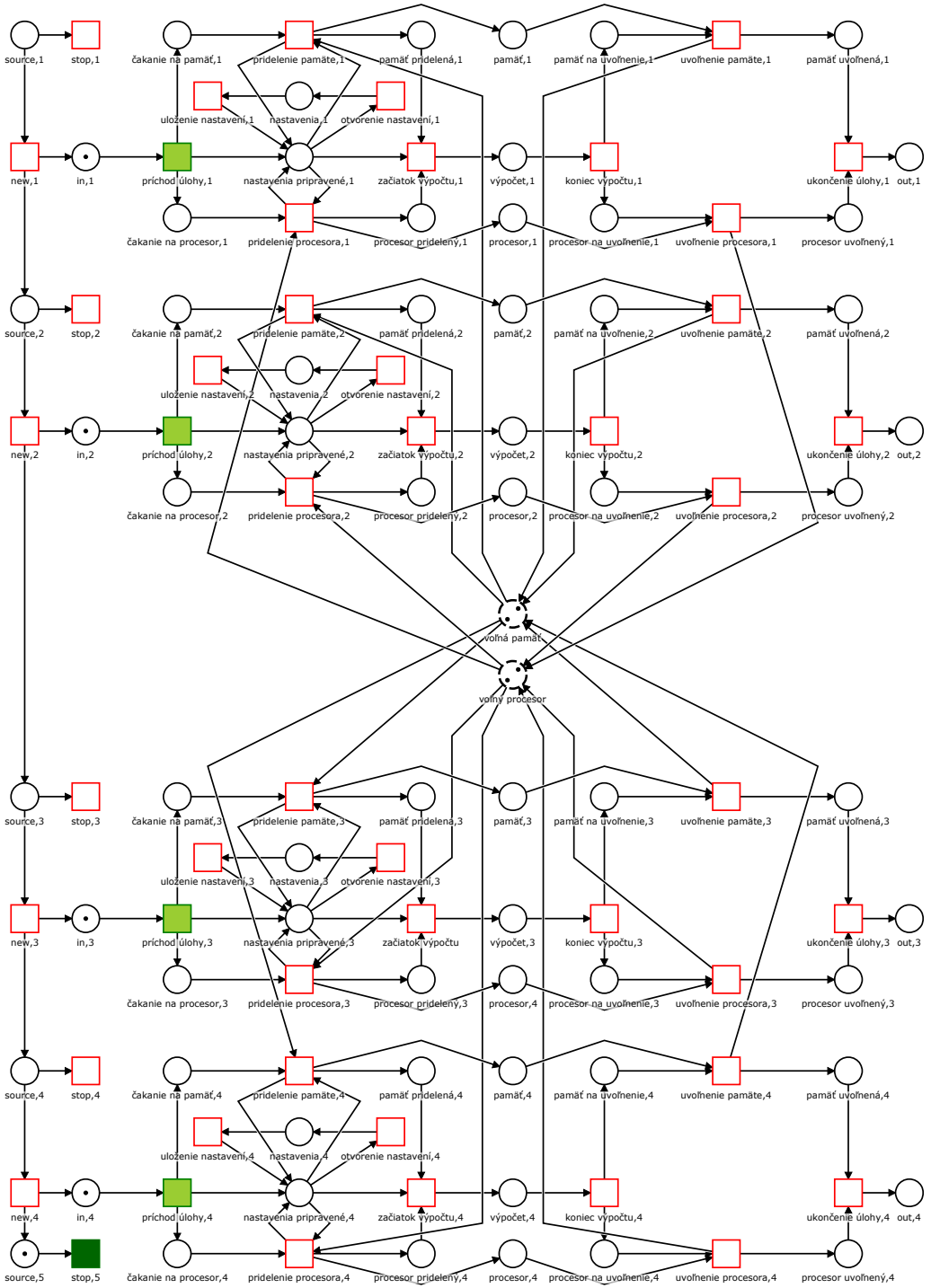
Vykonávaná sieť

new,4



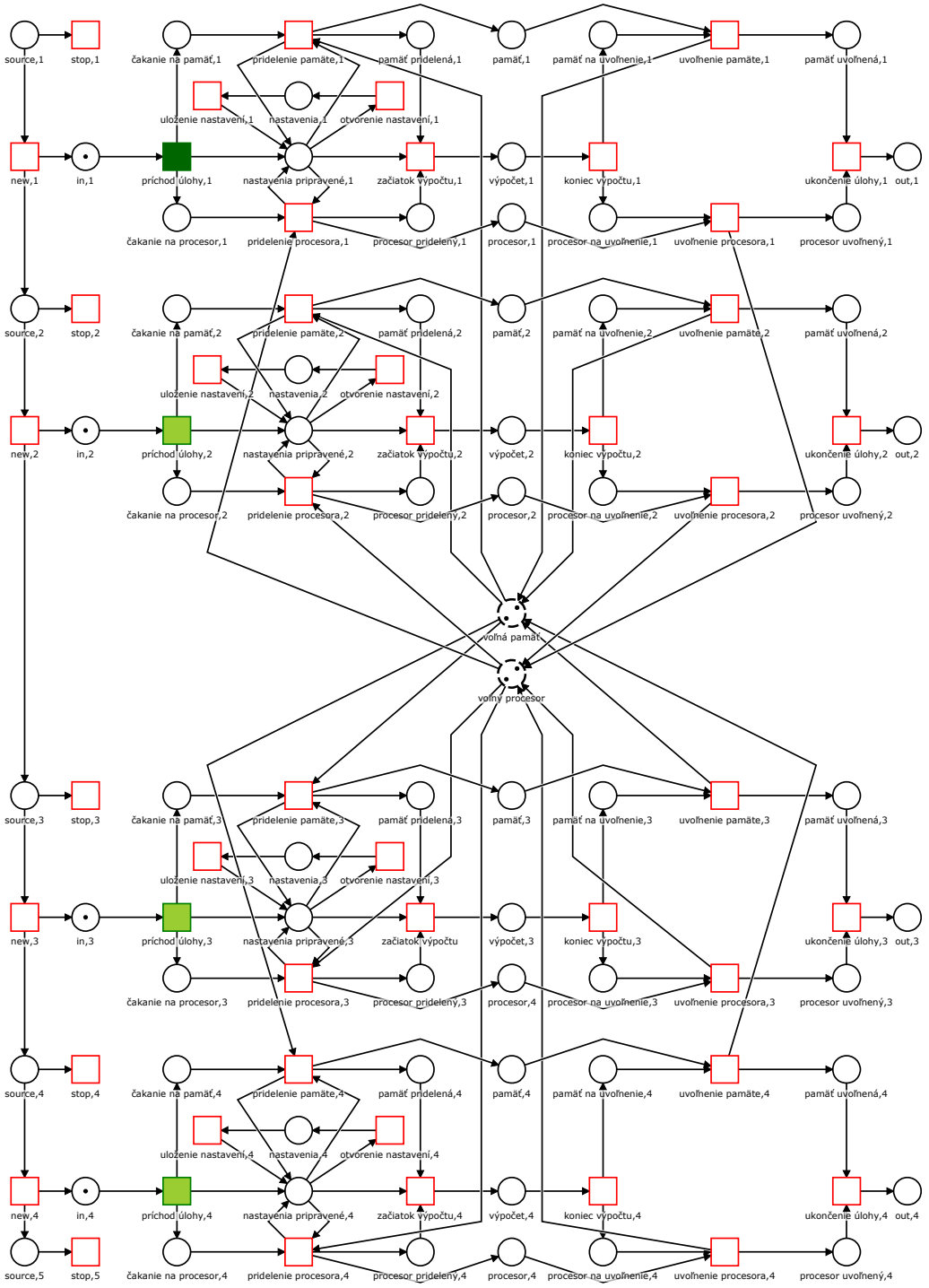
Vykonávaná sieť

stop,5



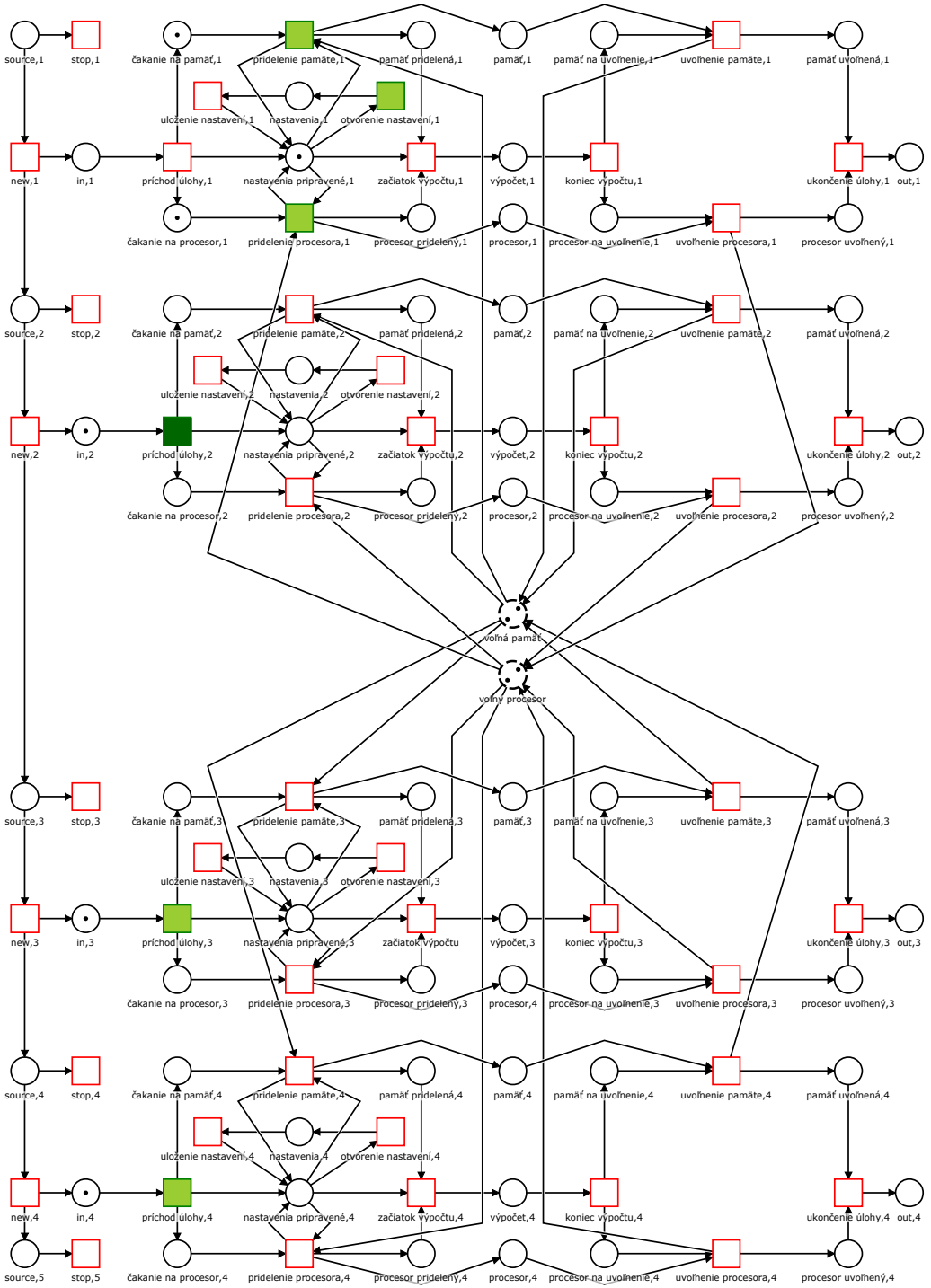
Vykonávaná sieť

príchod úlohy,1



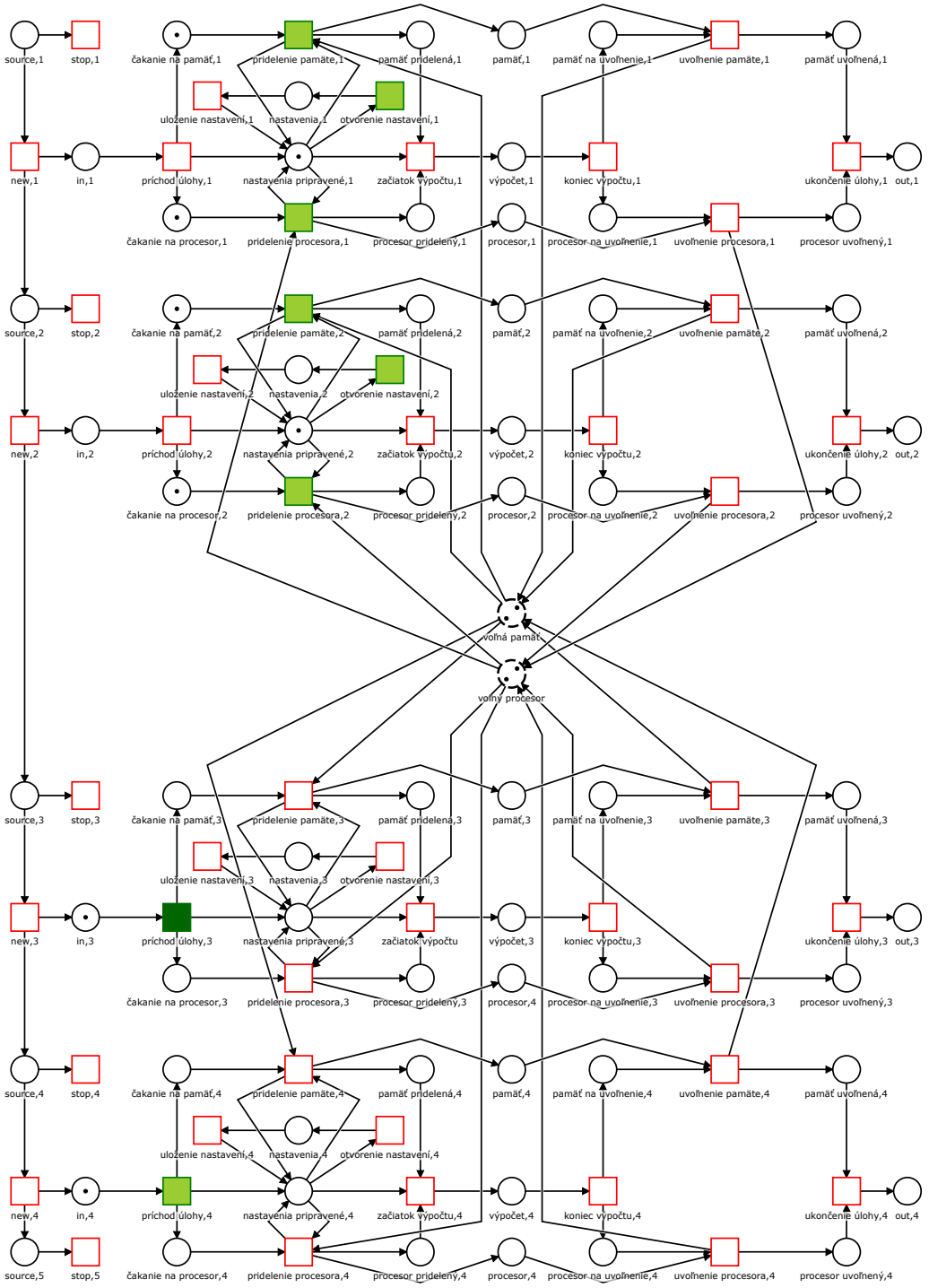
Vykonávaná sieť

príchod úlohy,2



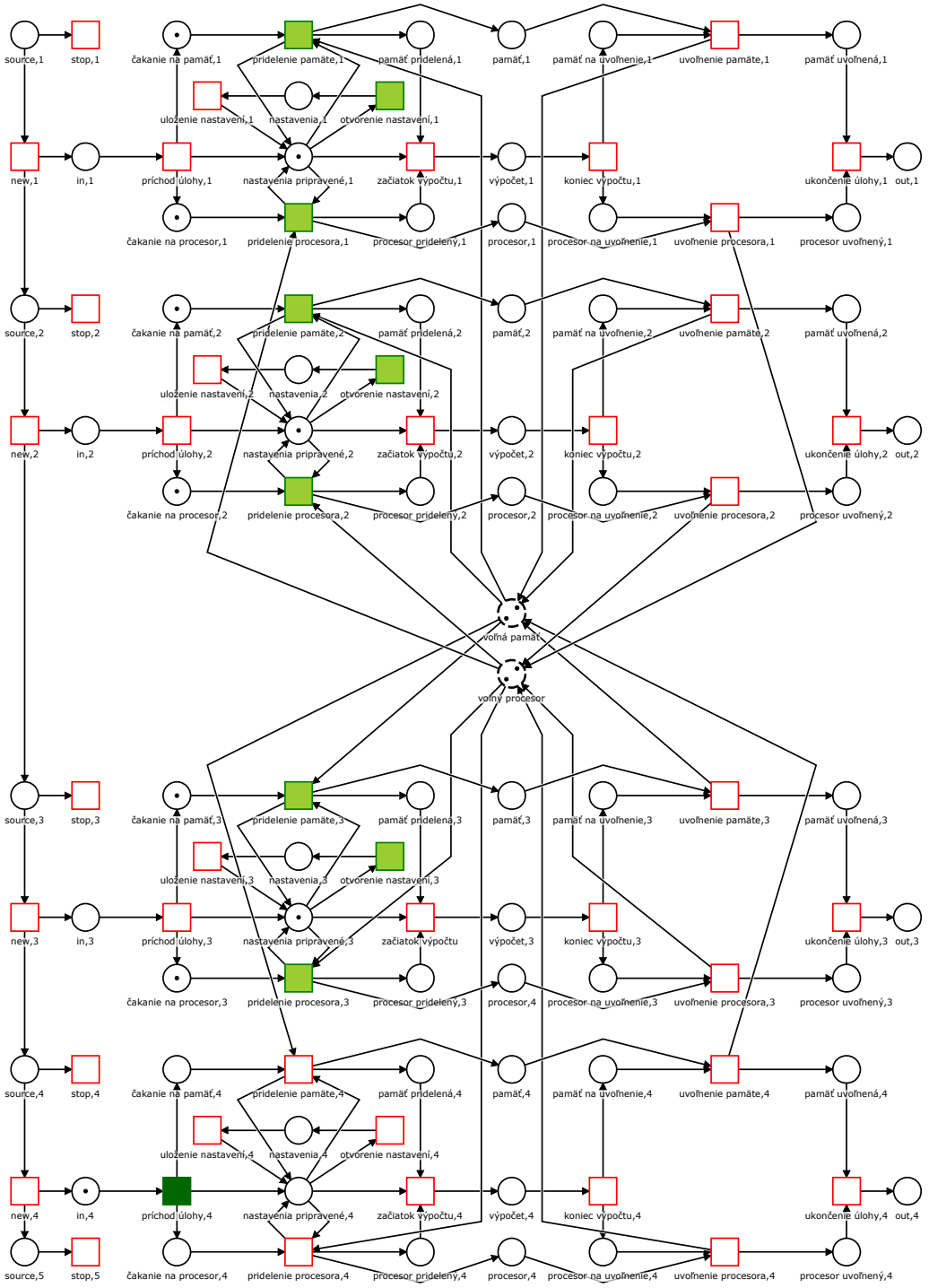
Vykonávaná sieť

príchod úlohy,3



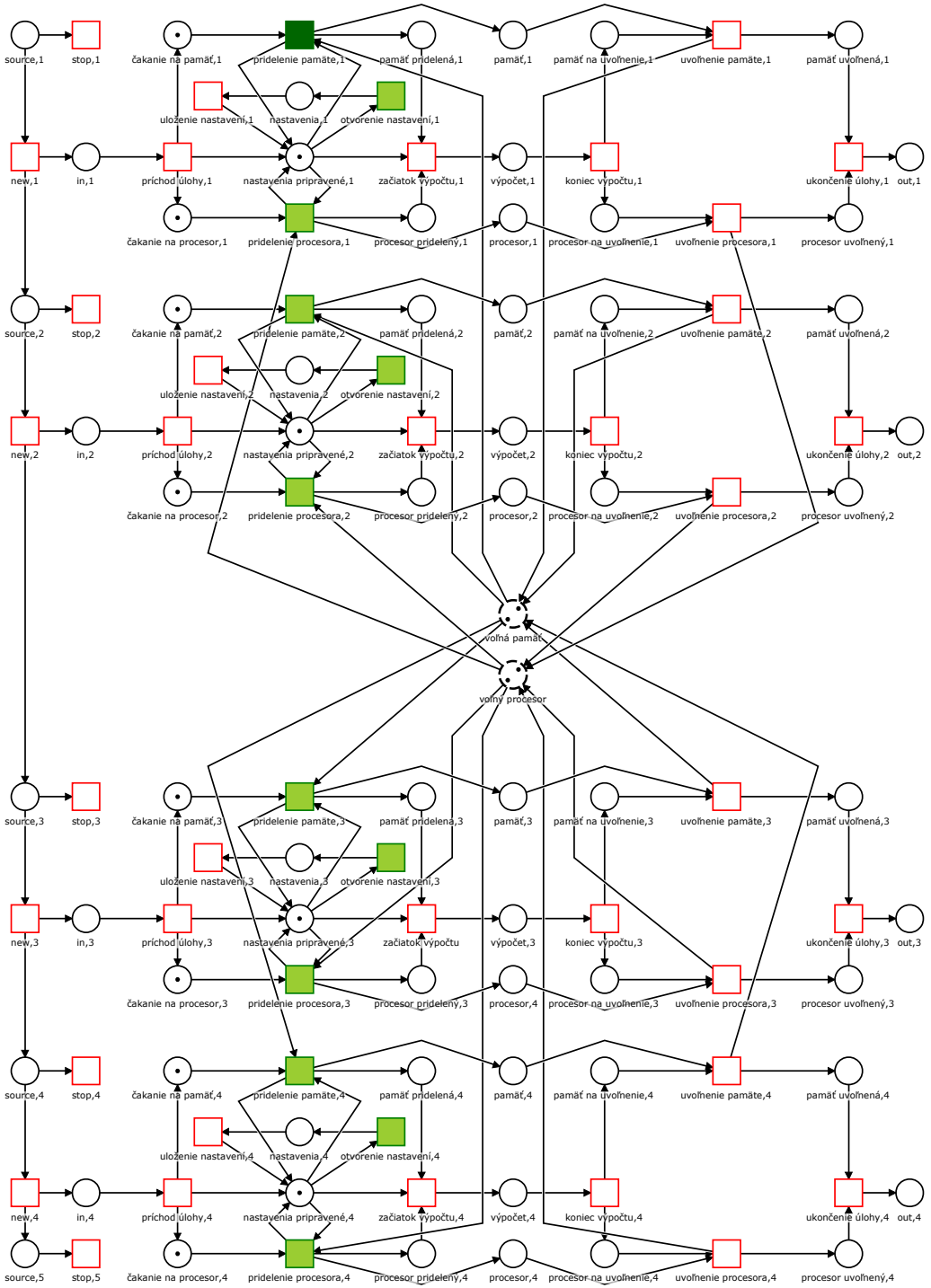
Vykonávaná sieť

príchod úlohy,4



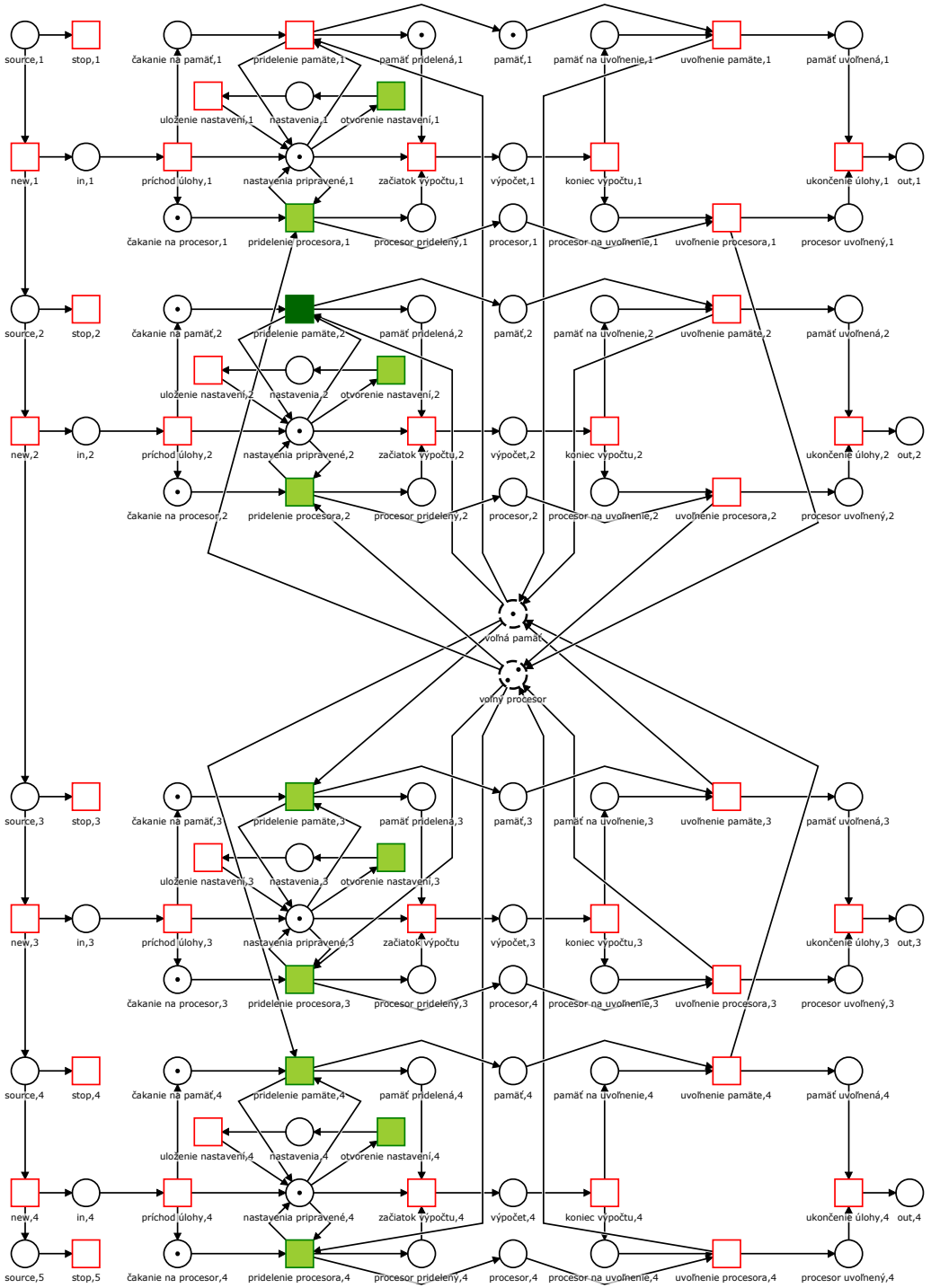
Vykonávaná sieť

pridelenie pamäte,1



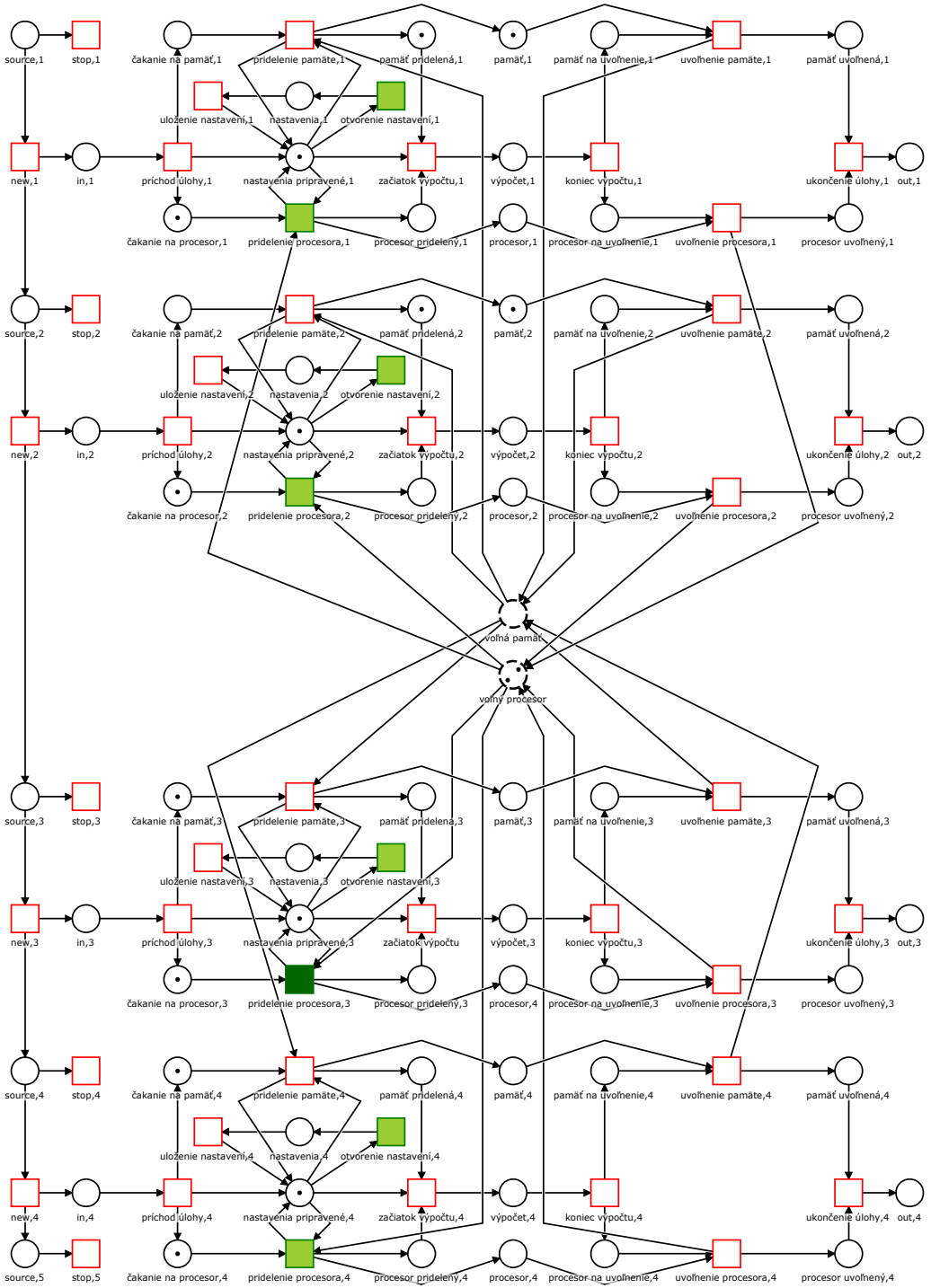
Vykonávaná sieť

pridelenie pamäte,2



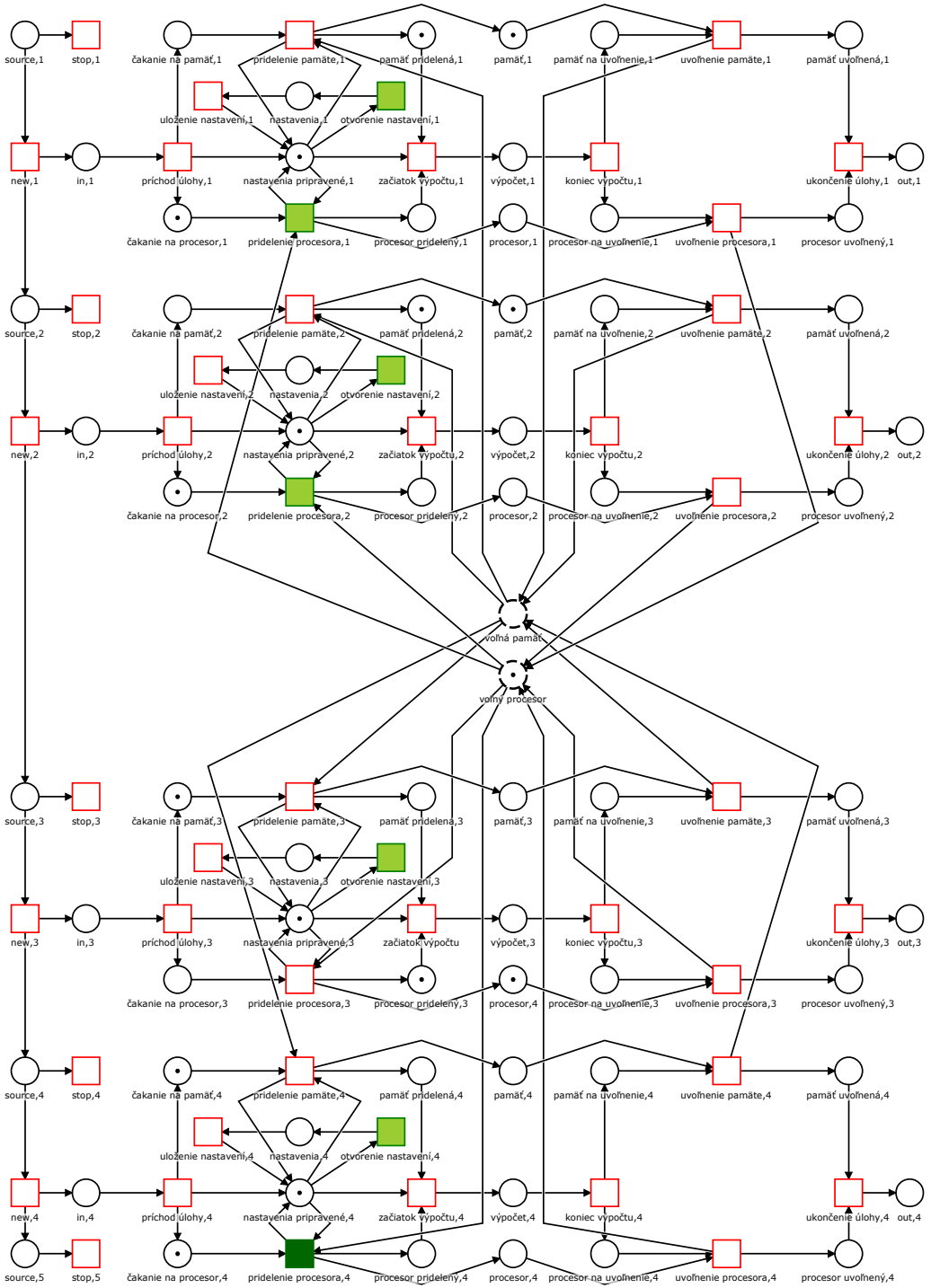
Vykonávaná sieť

pridelenie procesora,3



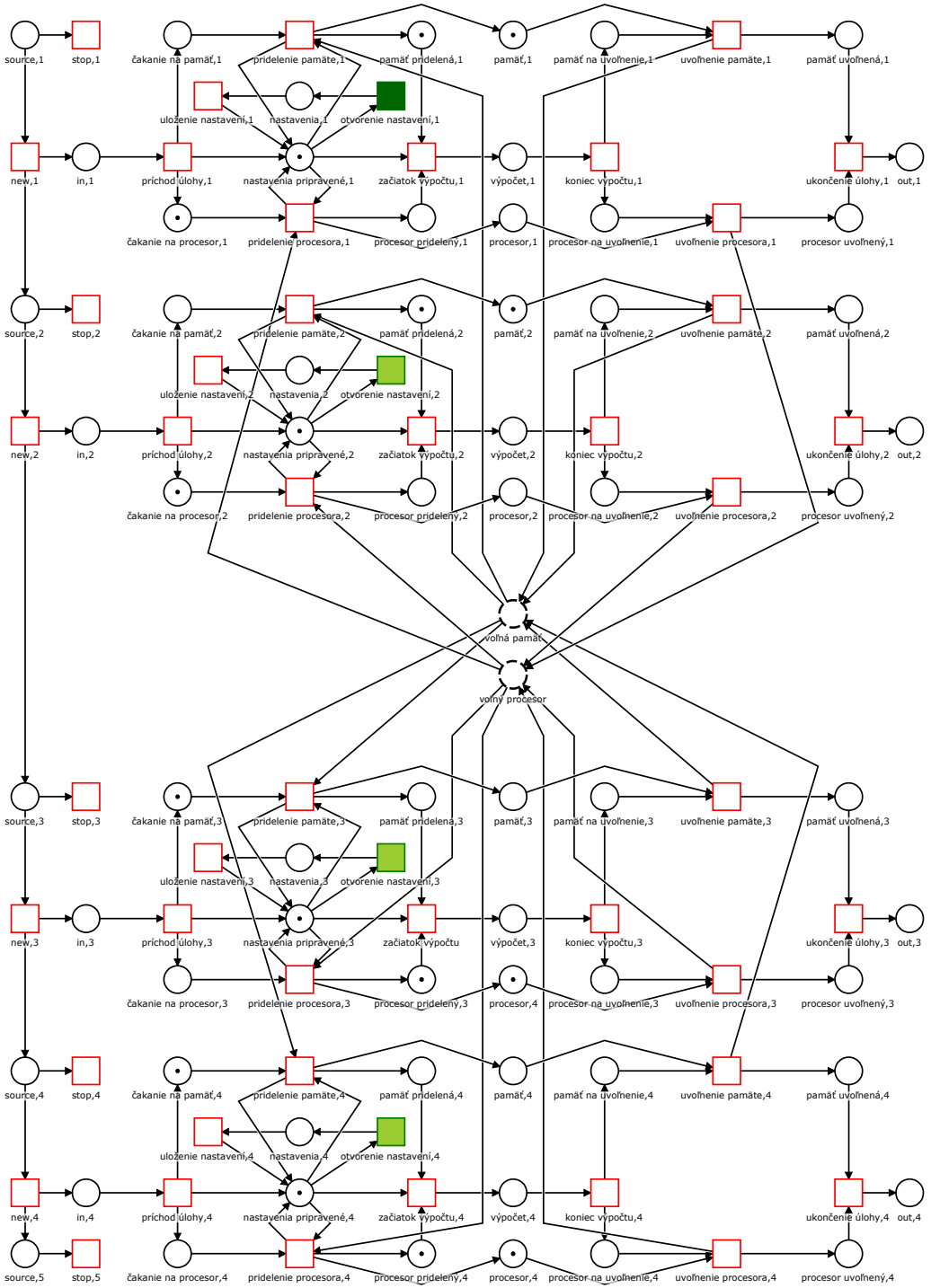
Vykonávaná sieť

pridelenie procesora,4



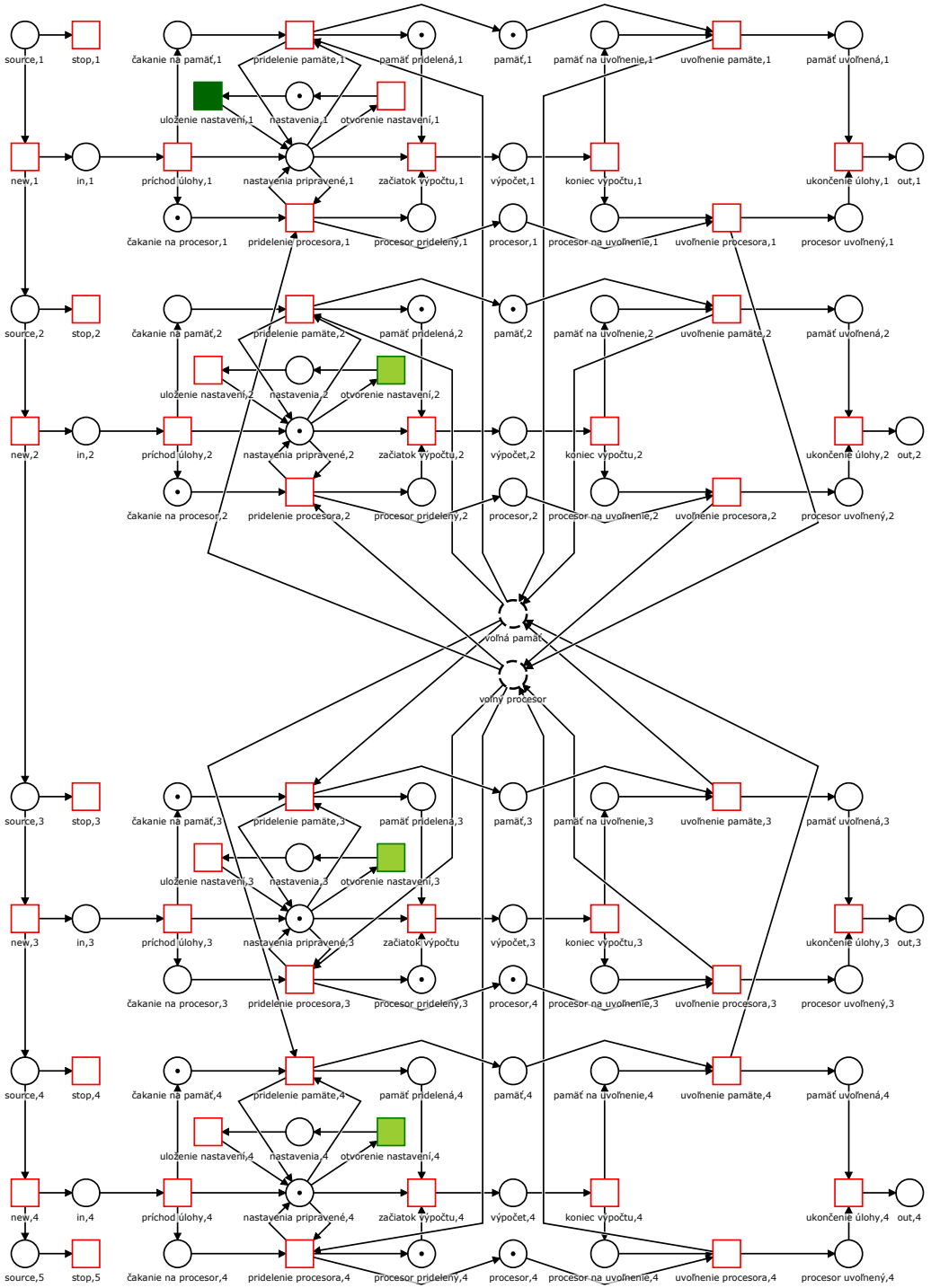
Vykonávaná sieť

otvorenie nastavení,1



Vykonávaná sieť

otvorenie nastavení,2

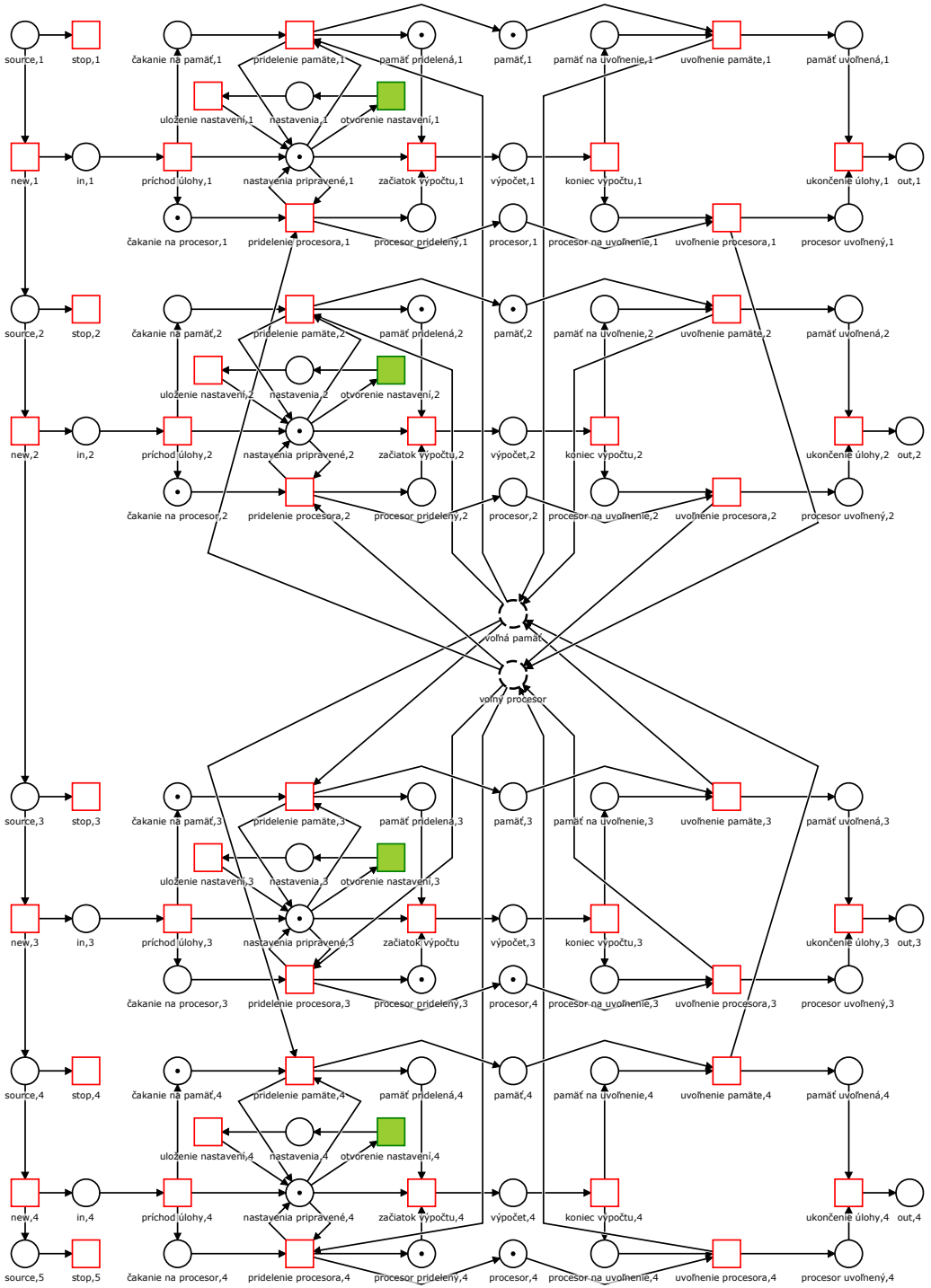


Vykonávaná sieť

Uviaznutie -

Nie je možné ukončiť inštancie

Uviaznutie, ktoré nie je zamrznutím, nazývame zacyklenie



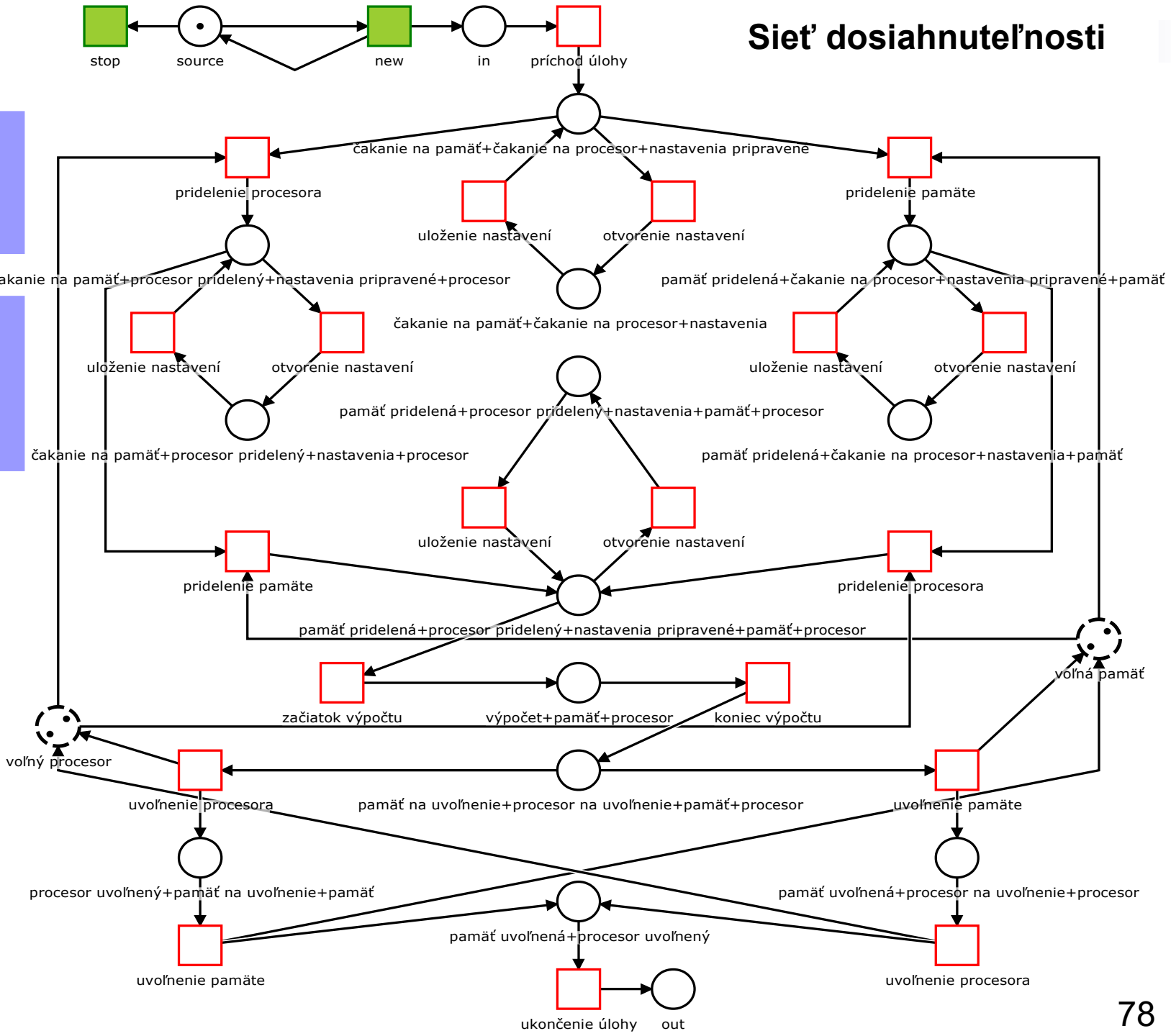
Vlastné výsledky

Keďže počet inštancií nie je obmedzený, vykonávané siete majú nekonečne veľa miest a prechodov

Sieť dosiahnuteľnosti

Bude simulovať správanie sa vykonávanej siete

Budeme si pamätať počet inštancií v jednotlivých stavoch

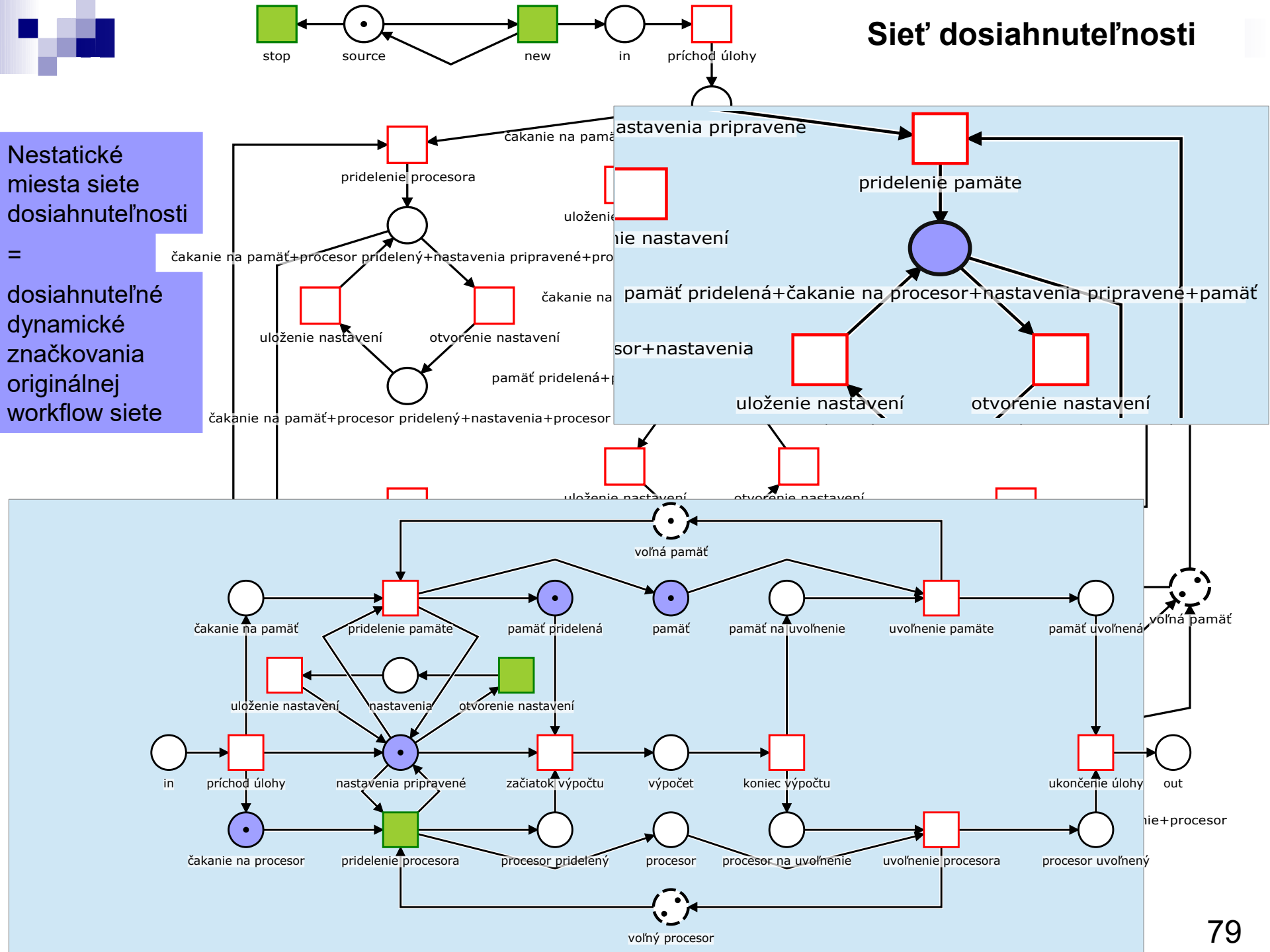


Sieť dosiahnuteľnosti

Nestatické miesta siete dosiahnuteľnosti

=

dosiahnuteľné dynamické značkovania originálnej workflow siete

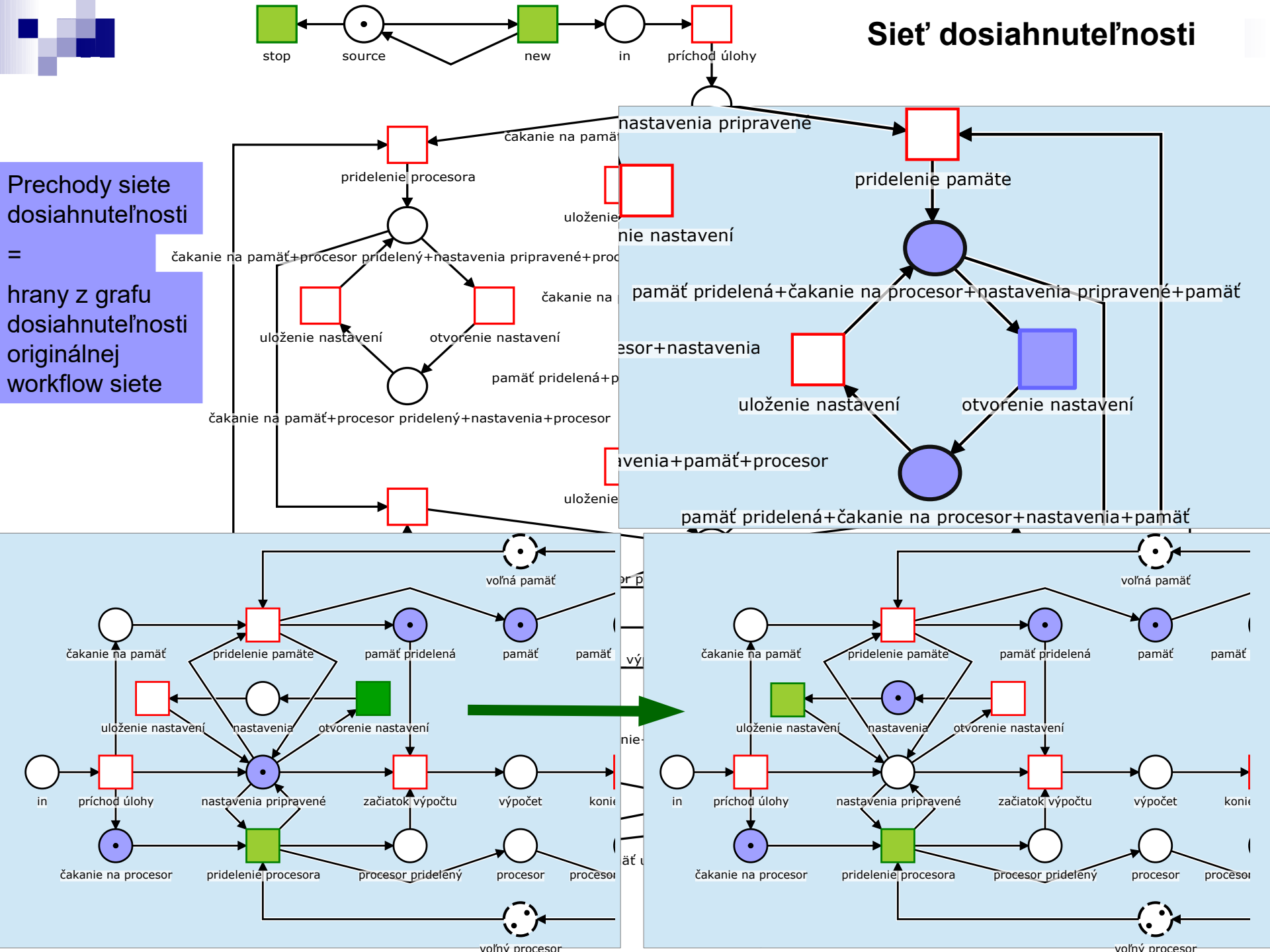


Sieť dosiahnuteľnosti

Prechody siete dosiahnuteľnosti

=

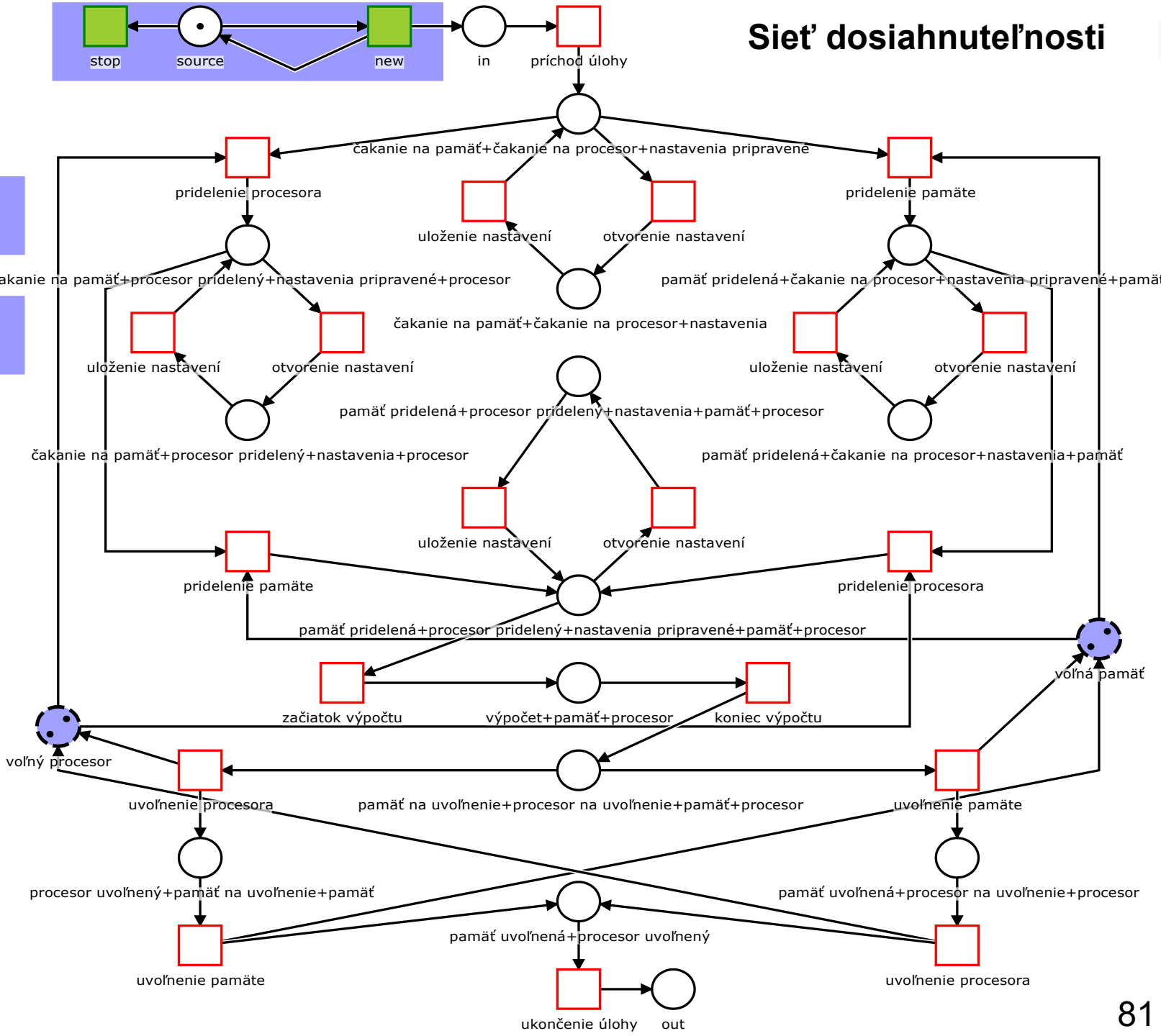
hrany z grafu dosiahnuteľnosti originálnej workflow siete



Sieť dosiahnuteľnosti

Statické miesta sa kopírujú

Pridáme konštruktor



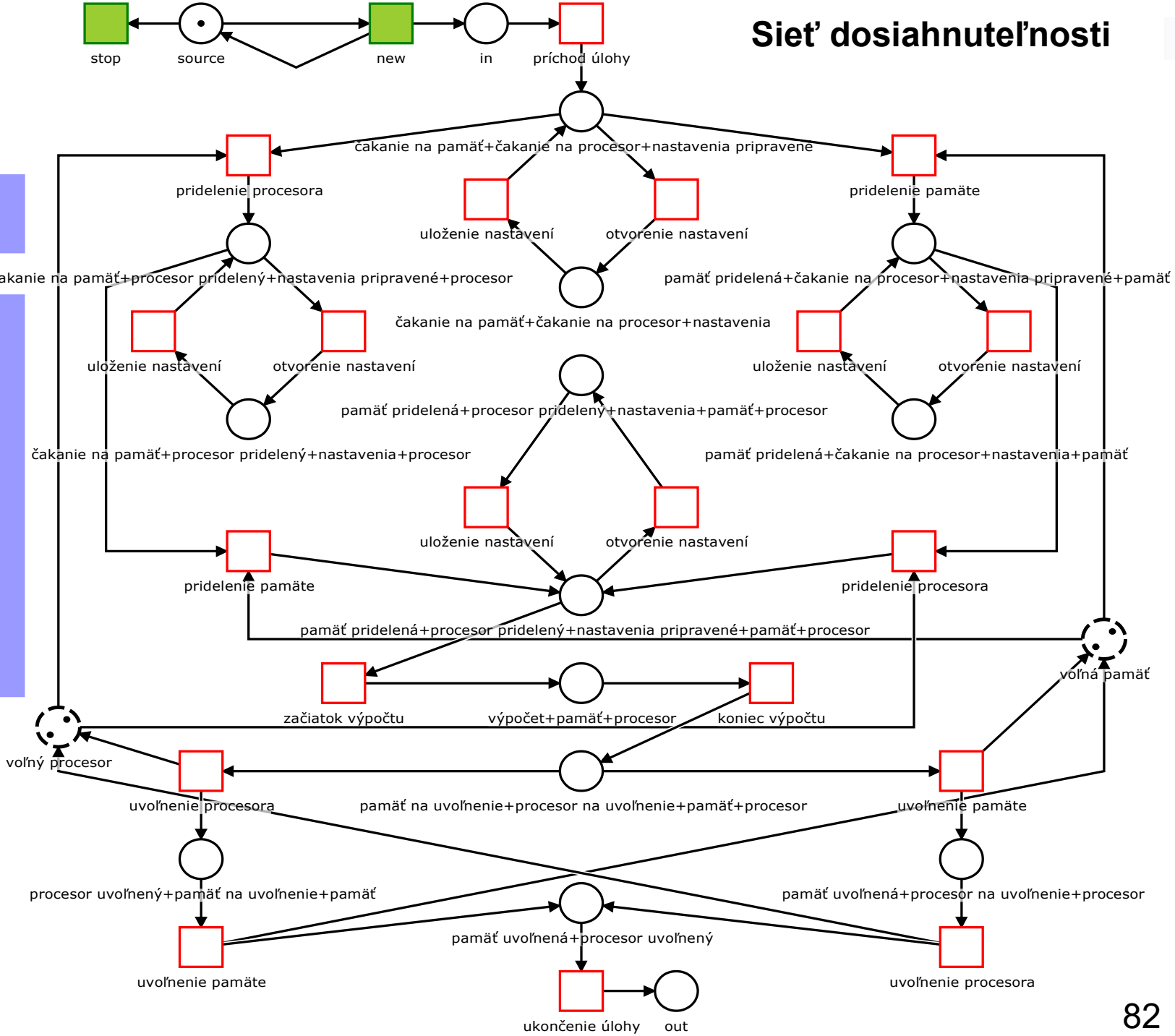
Sieť dosiahnuteľnosti

Finálne značkovanie

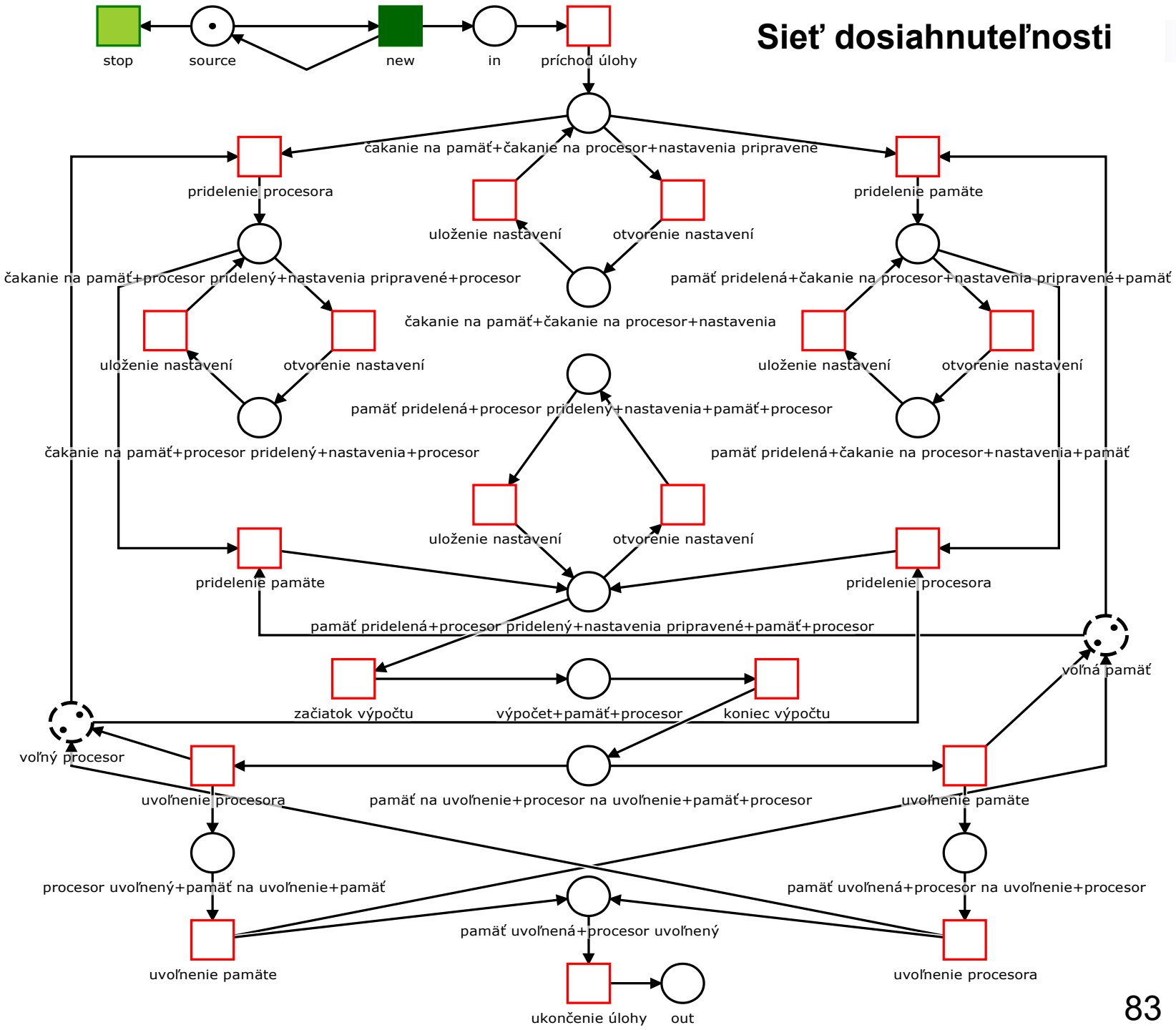
Značky sú len v mieste out a v statických miestach

Uviaznutie

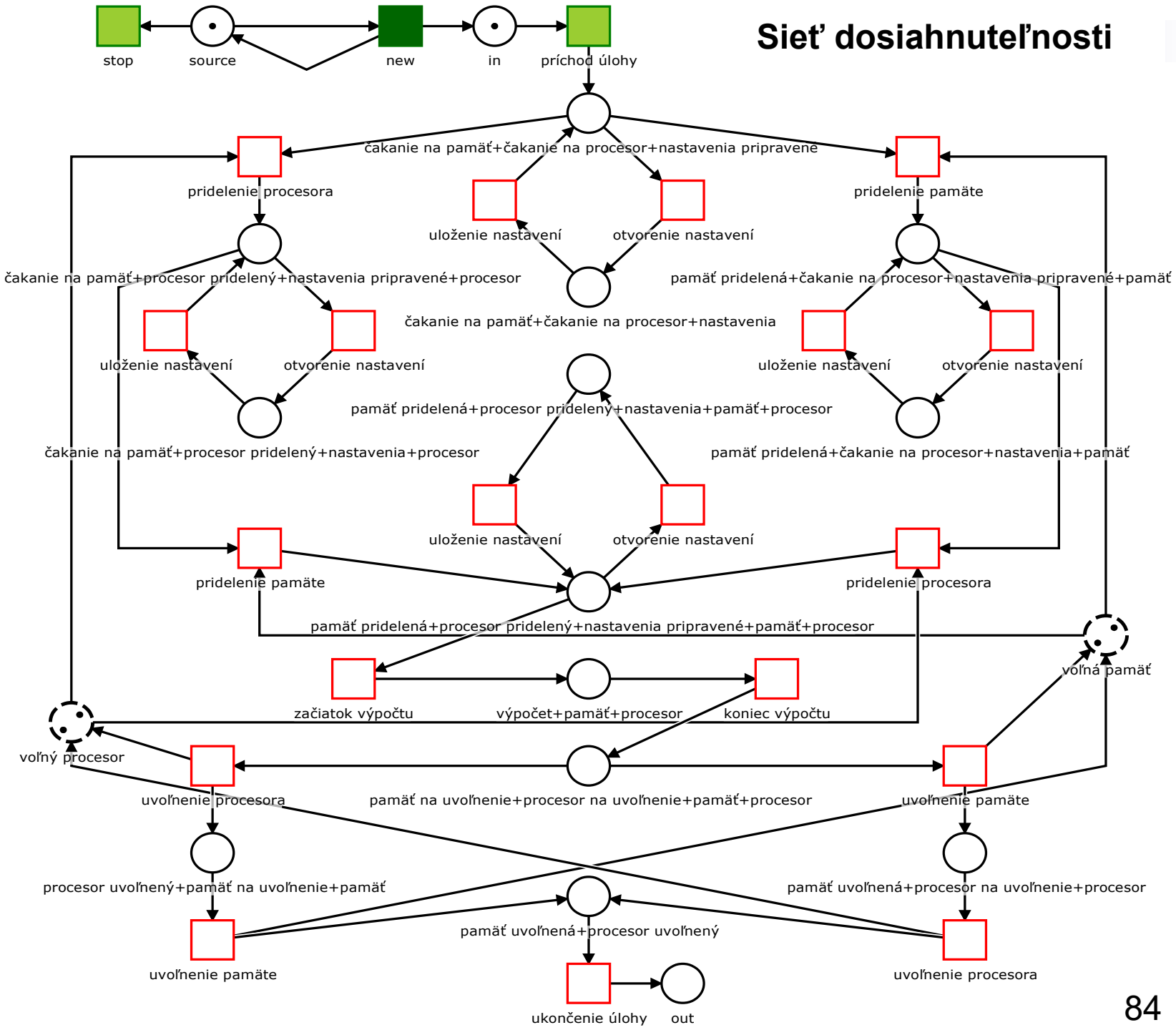
Nie je možné dosiahnuť žiadne finálne značkovanie



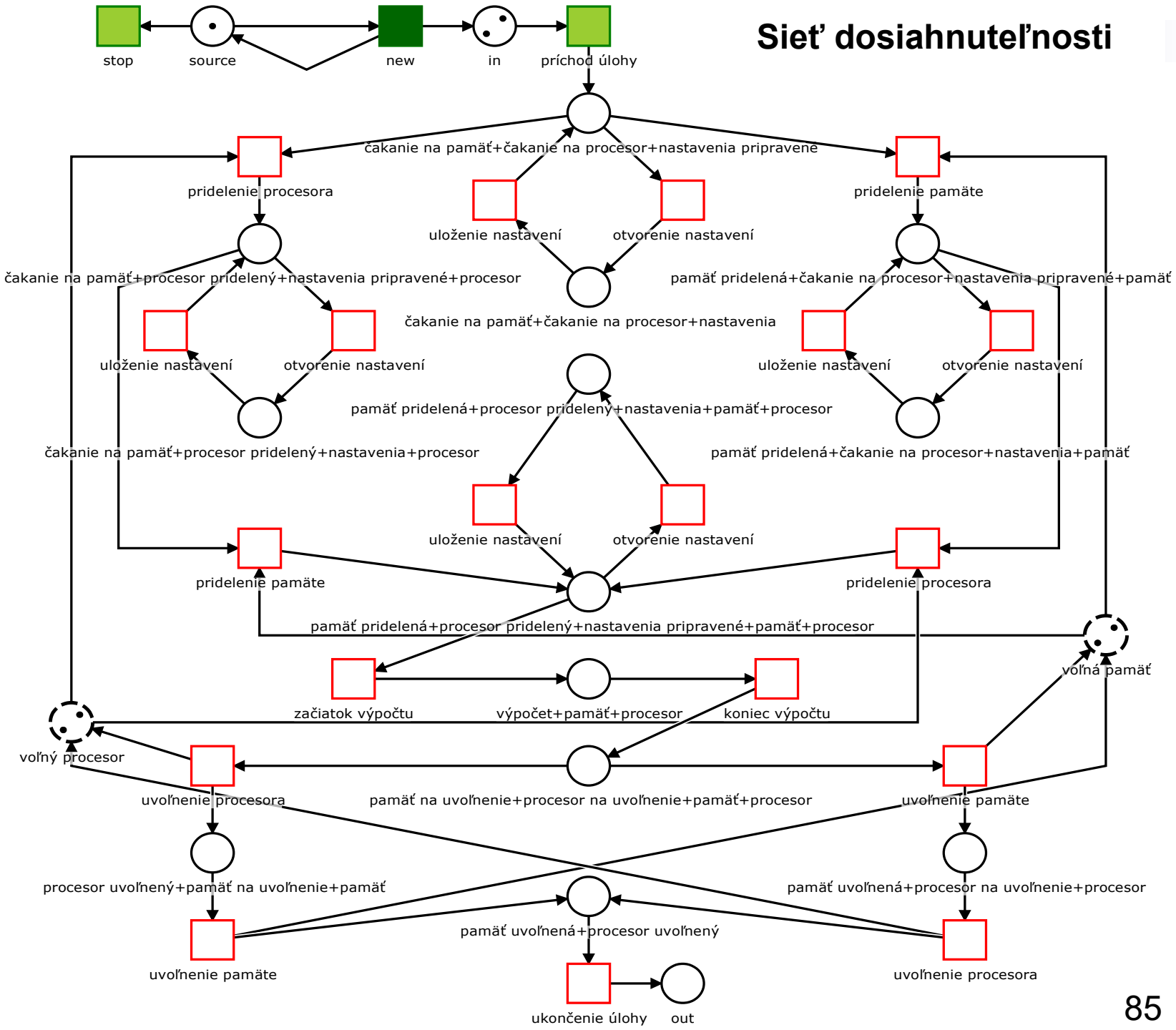
Sieť dosiahnuteľnosti



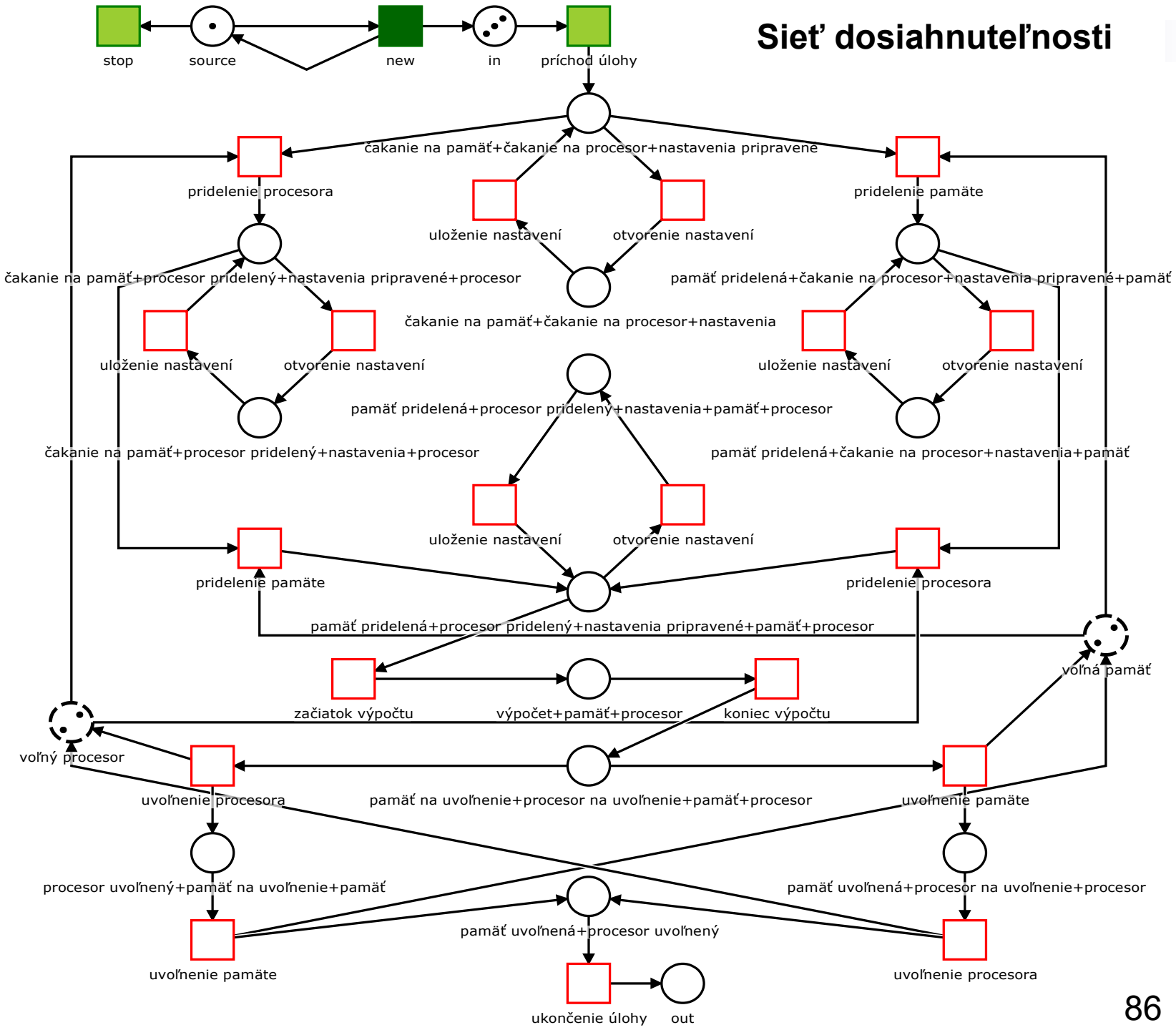
Sieť dosiahnuteľnosti



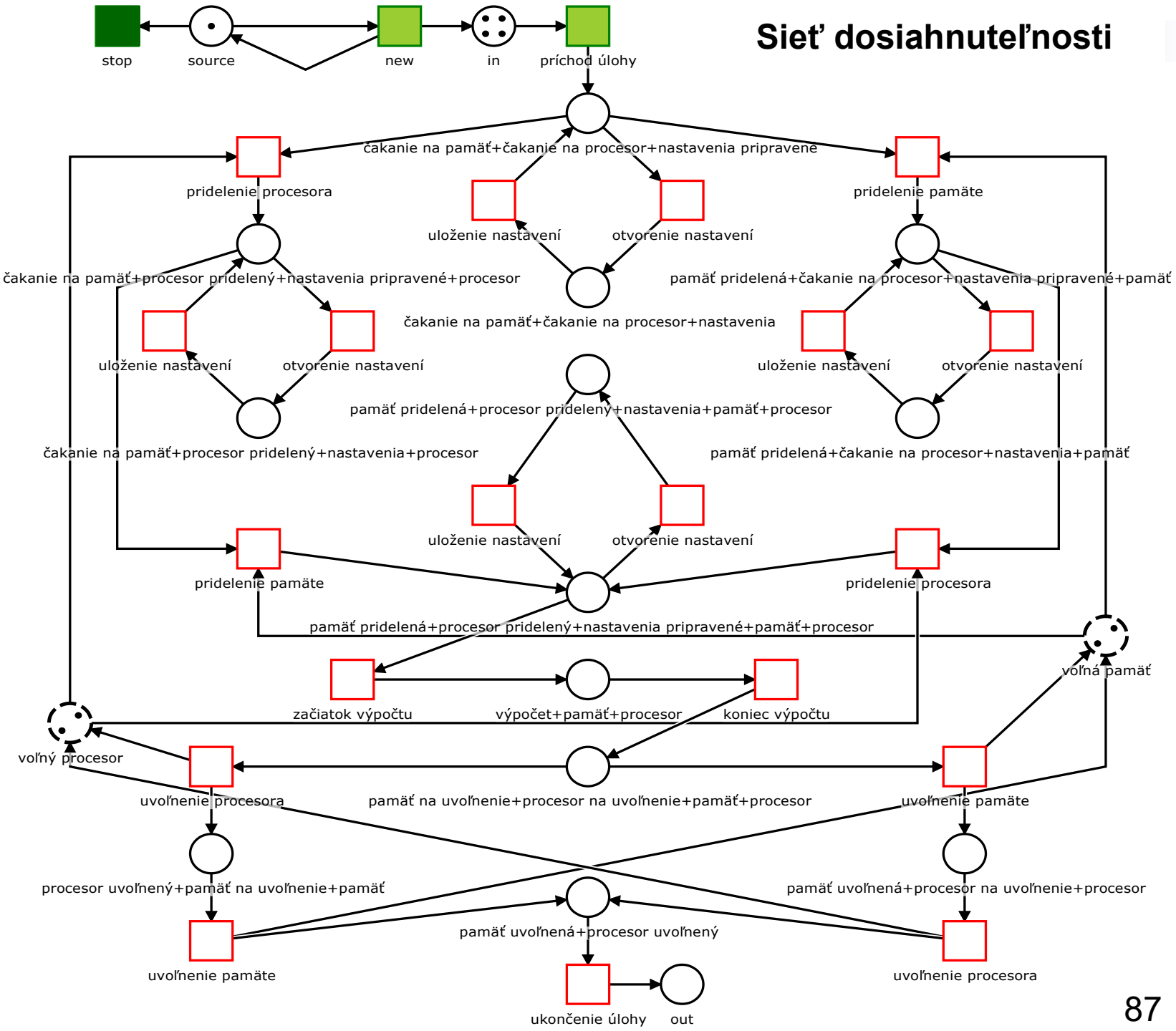
Sieť dosiahnuteľnosti



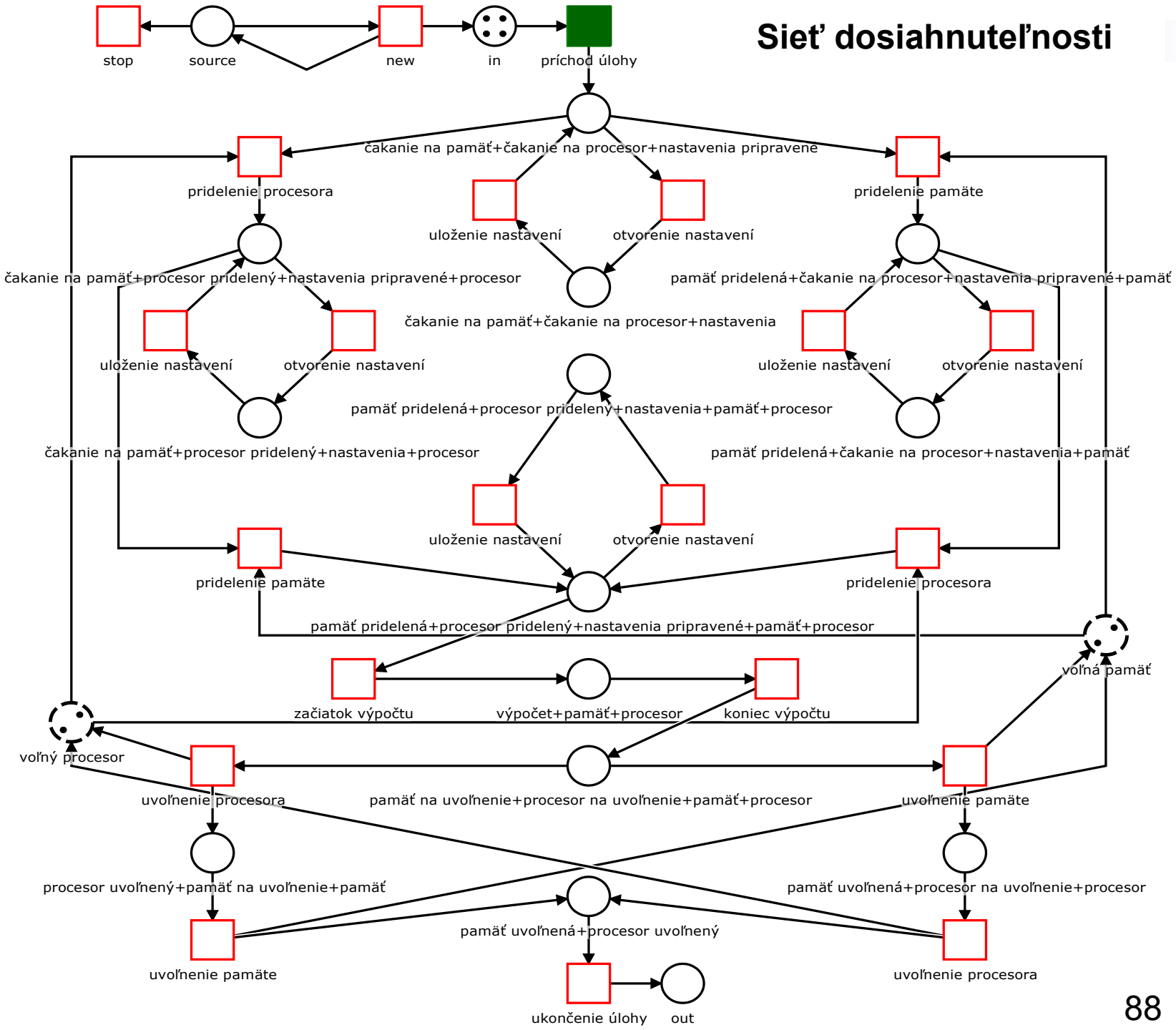
Sieť dosiahnuteľnosti



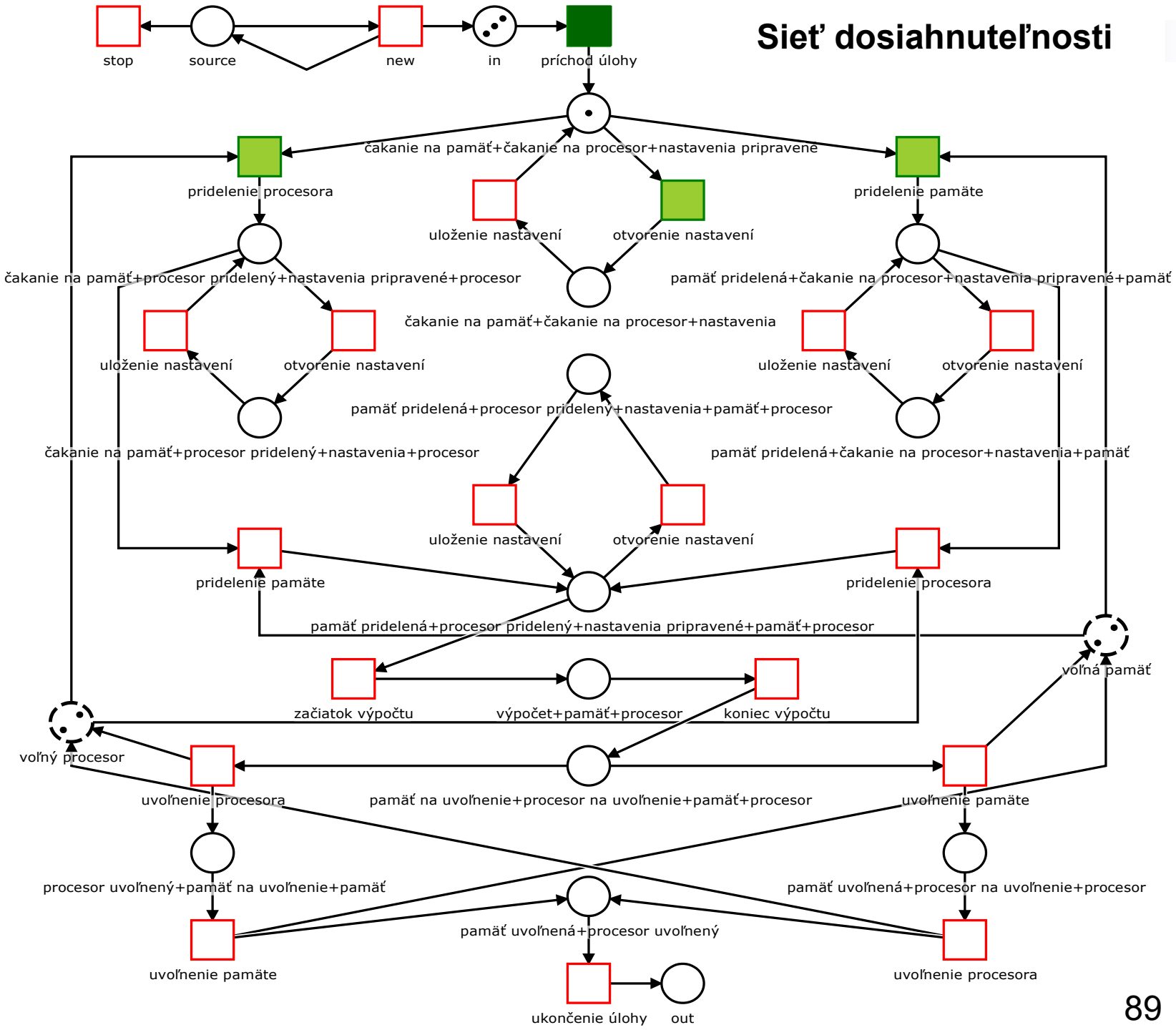
Sieť dosiahnuteľnosti



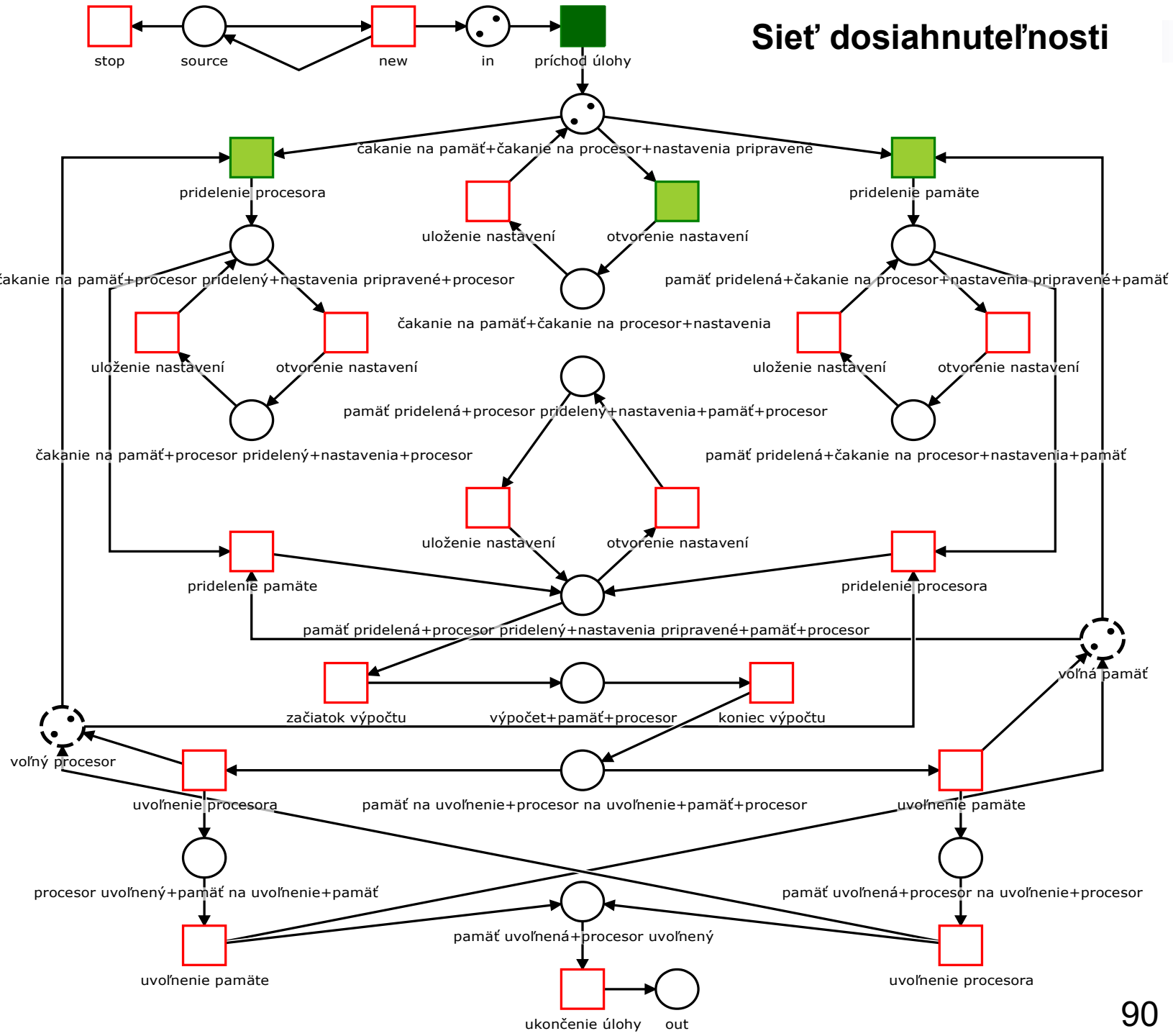
Sieť dosiahnuteľnosti



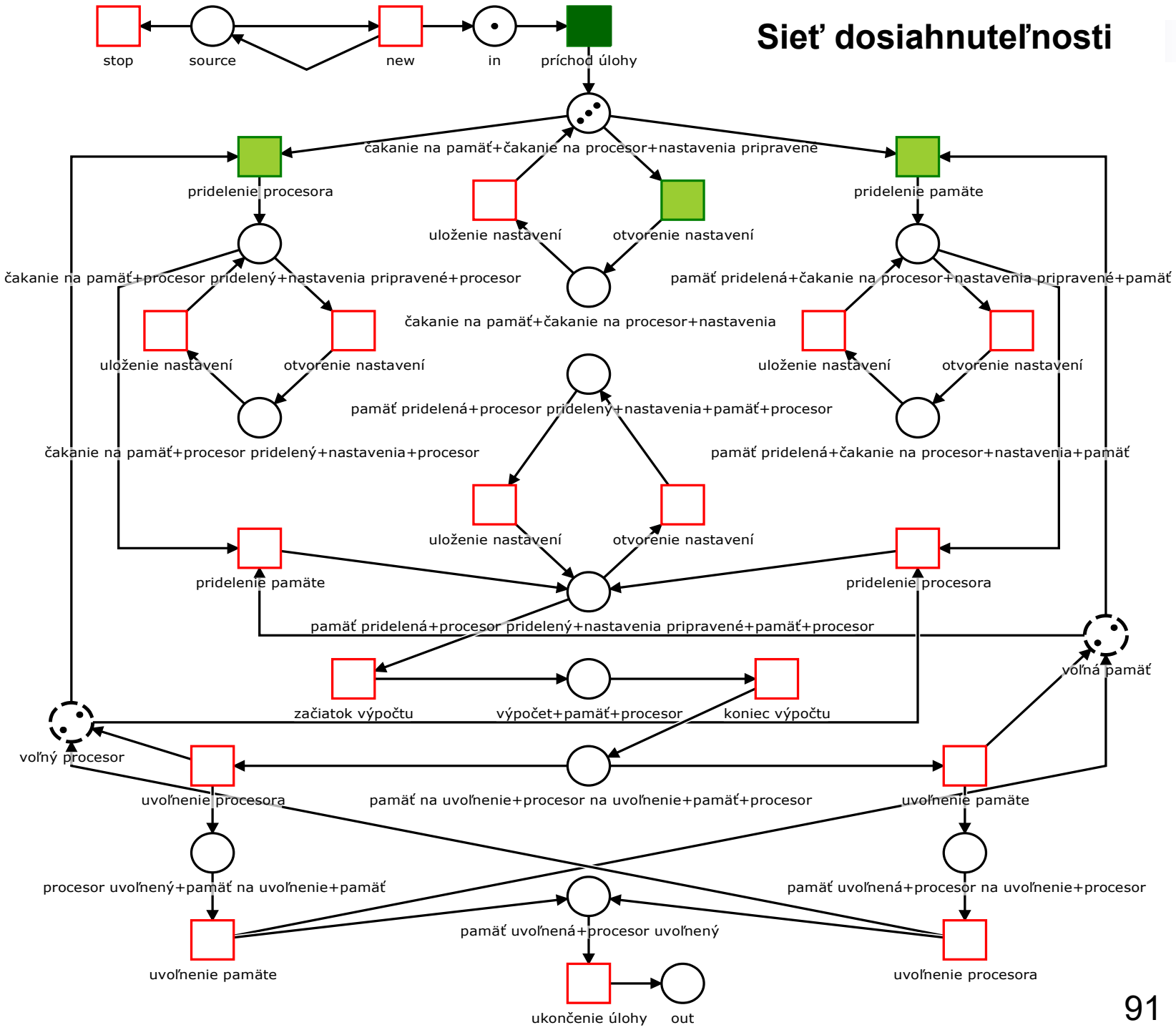
Sieť dosiahnuteľnosti



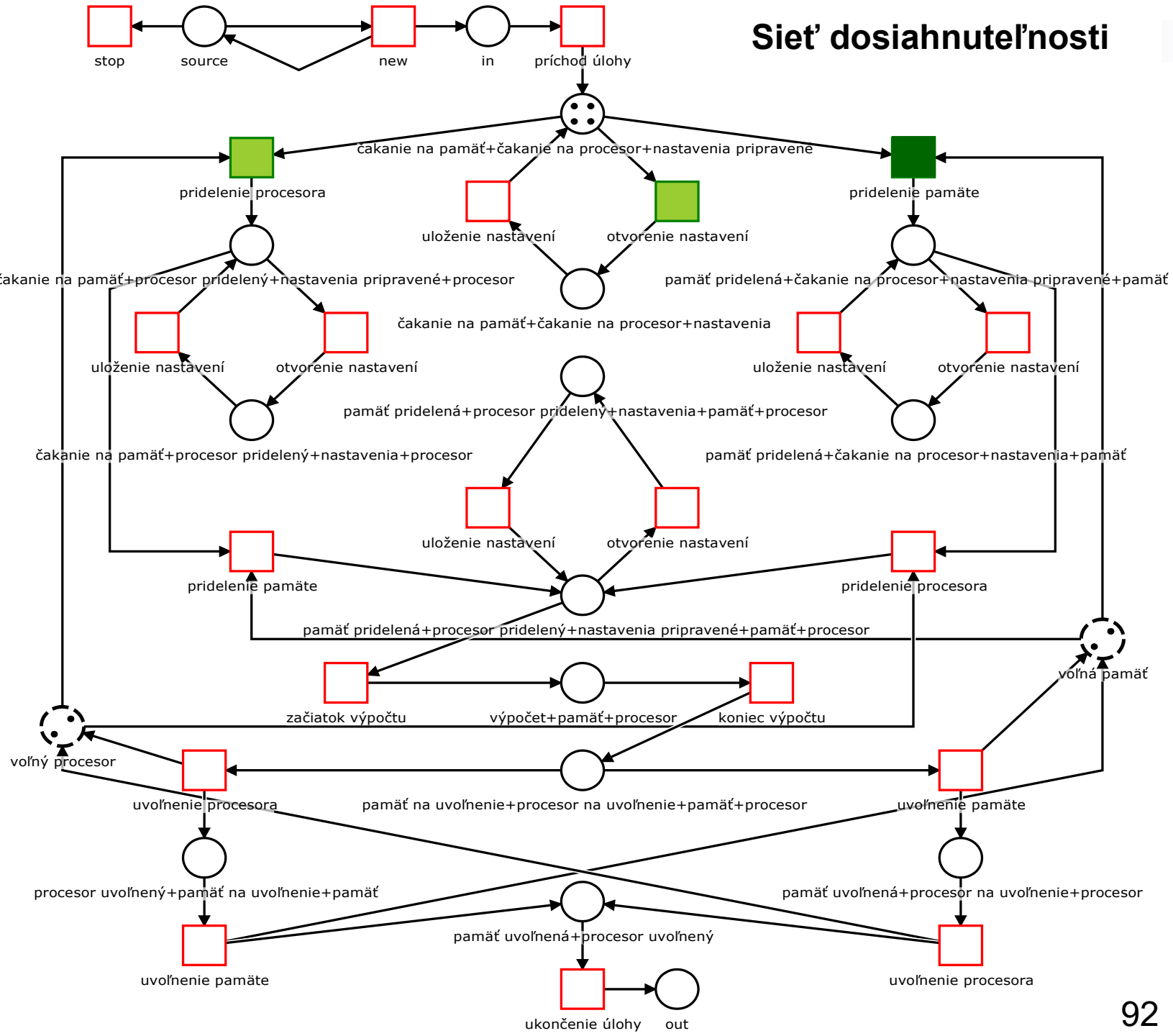
Sieť dosiahnuteľnosti



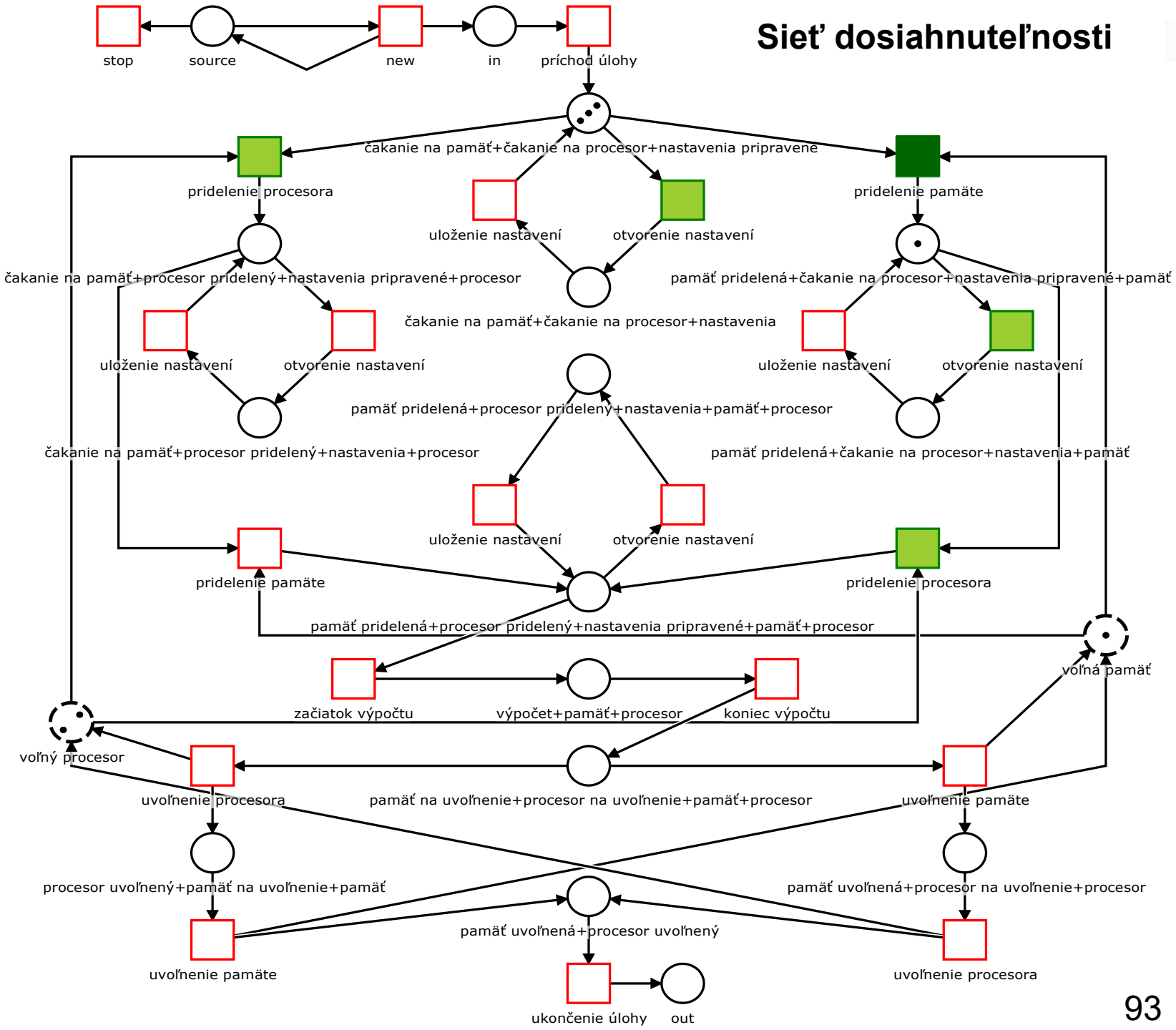
Sieť dosiahnuteľnosti



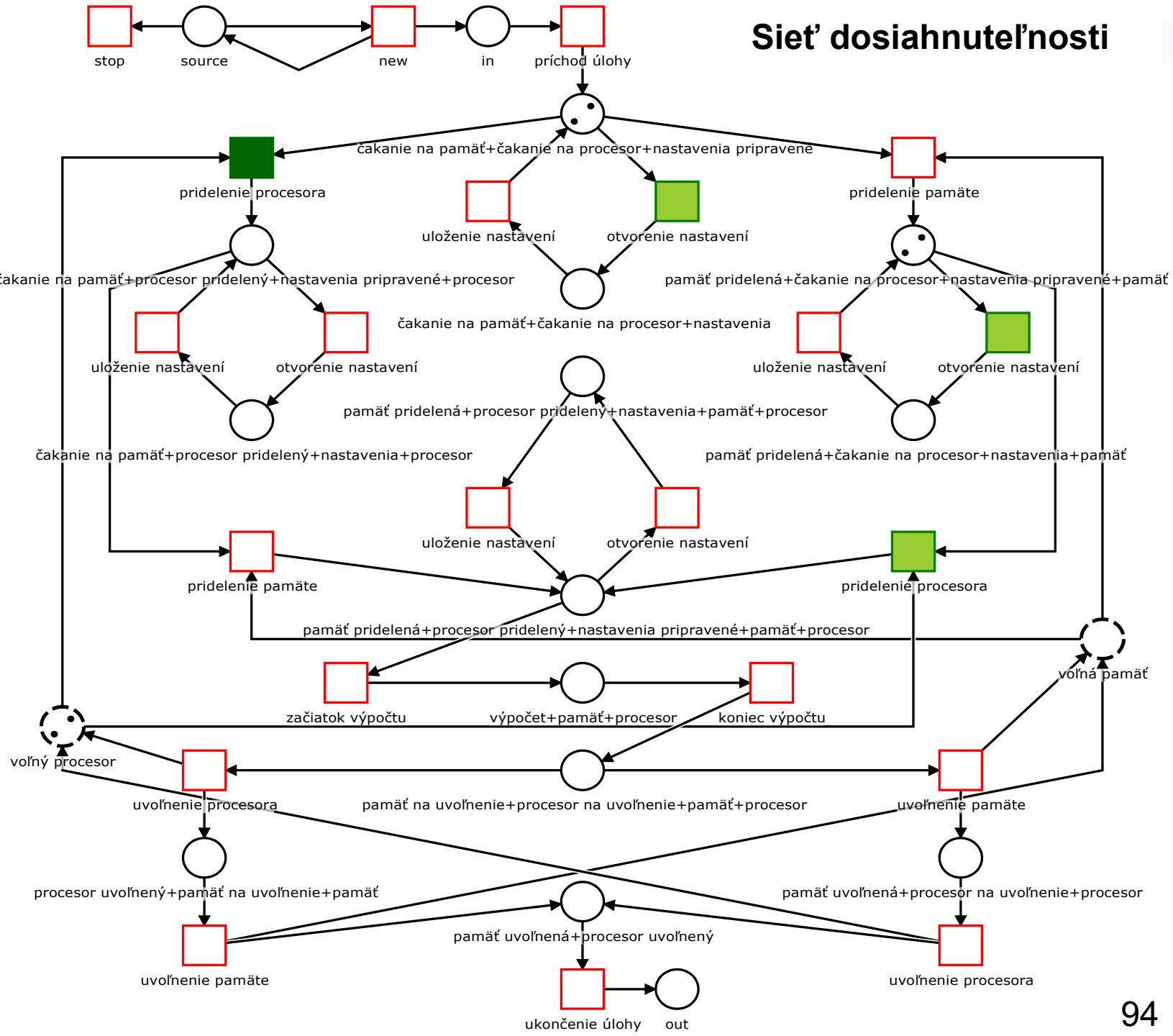
Sieť dosiahnuteľnosti



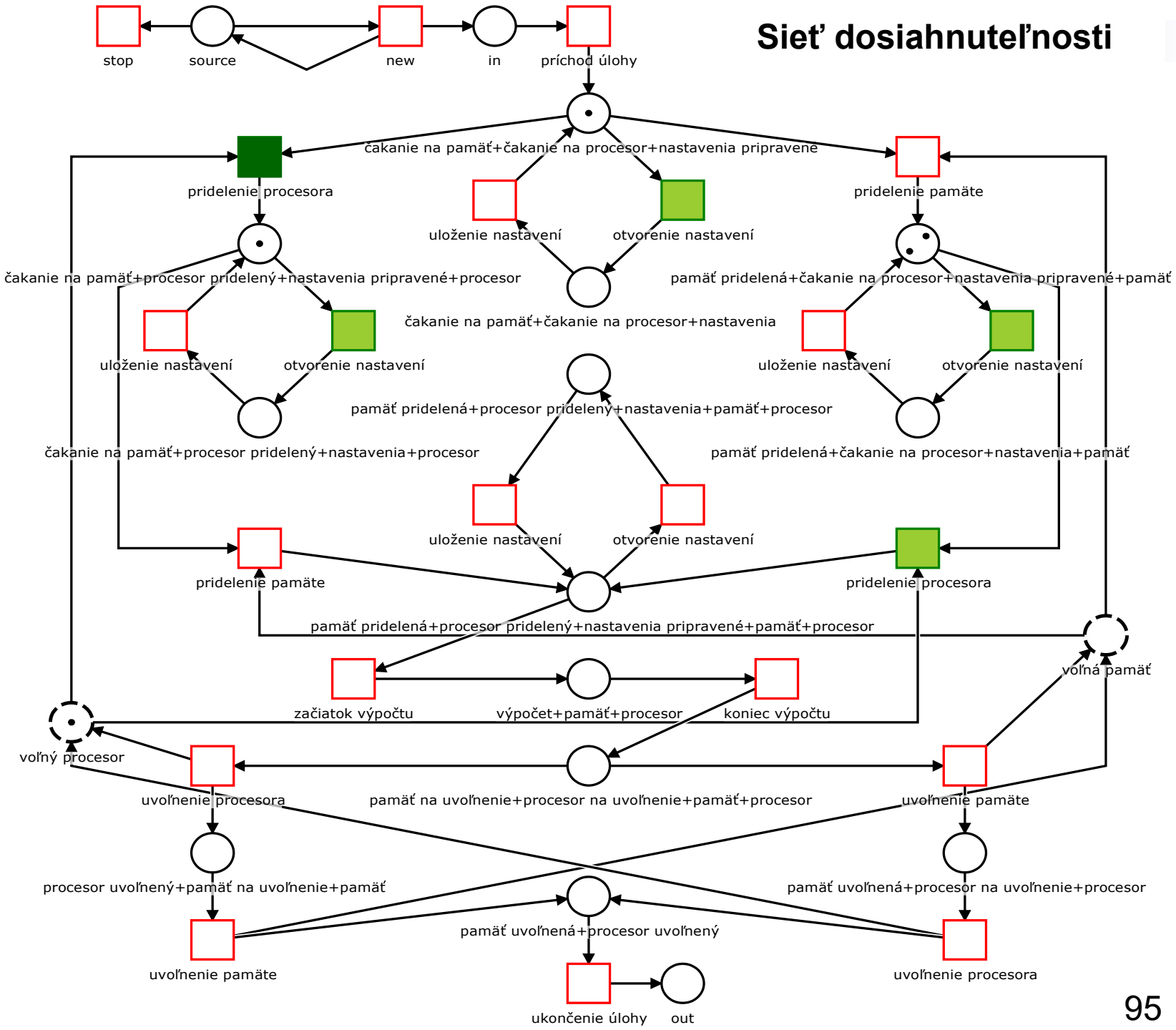
Sieť dosiahnuteľnosti



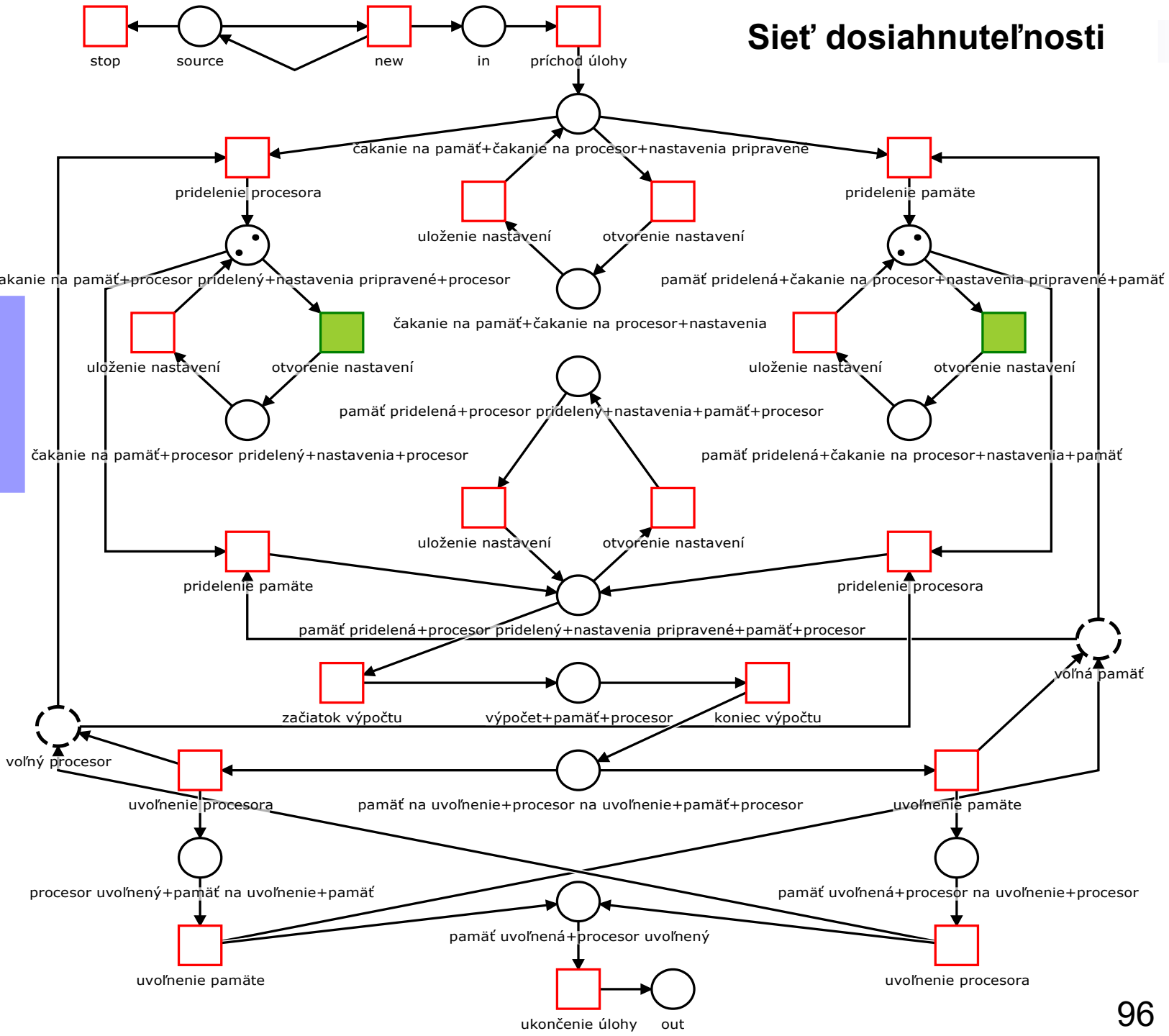
Sieť dosiahnuteľnosti



Sieť dosiahnuteľnosti



Sieť dosiahnuteľnosti



Uviaznutie

Nie je možné dosiahnuť žiadne finálne značkovanie

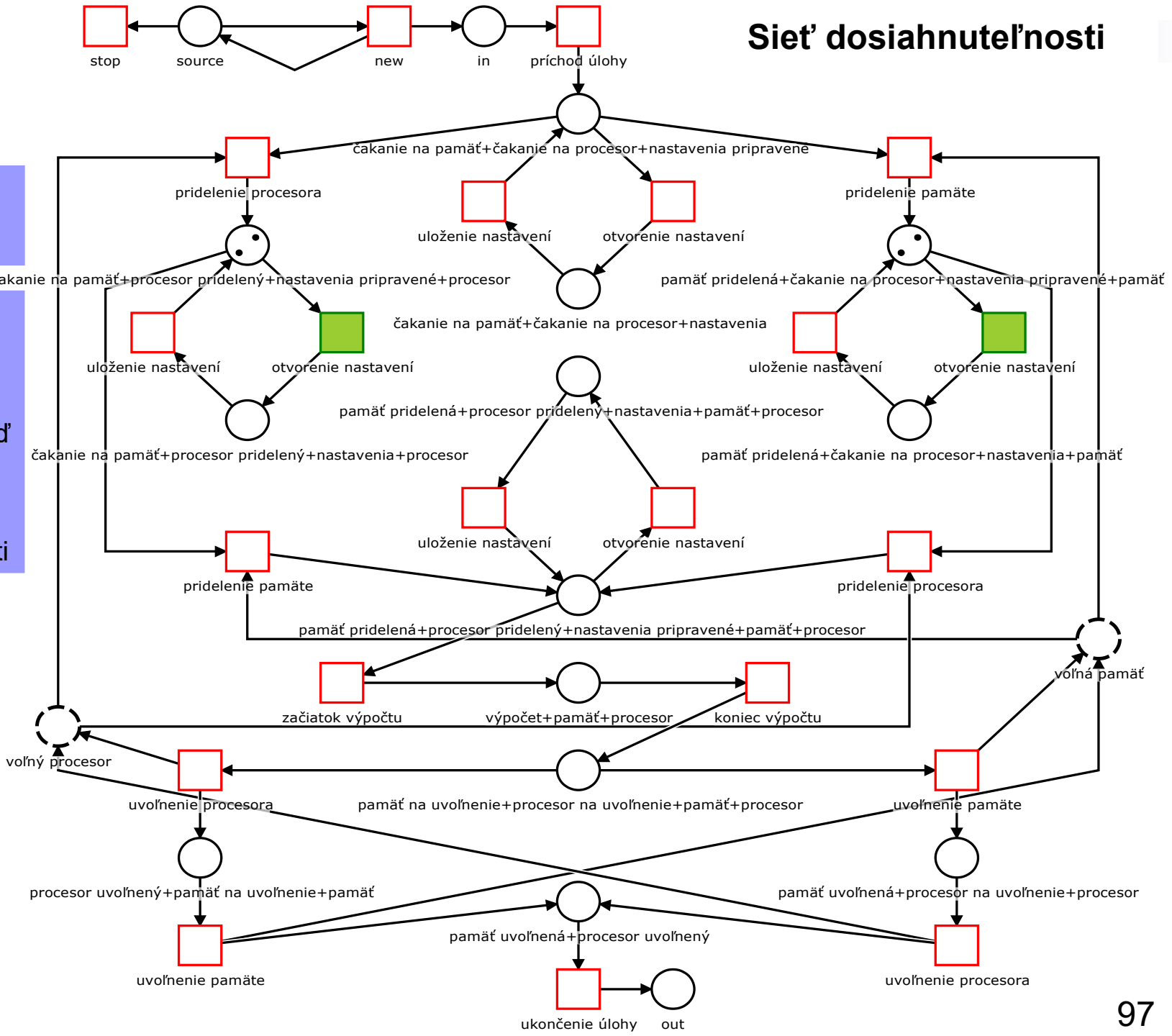
Sieť dosiahnuteľnosti

Výsledok 1:

Vykonávaná sieť má uviaznutie

práve vtedy, keď

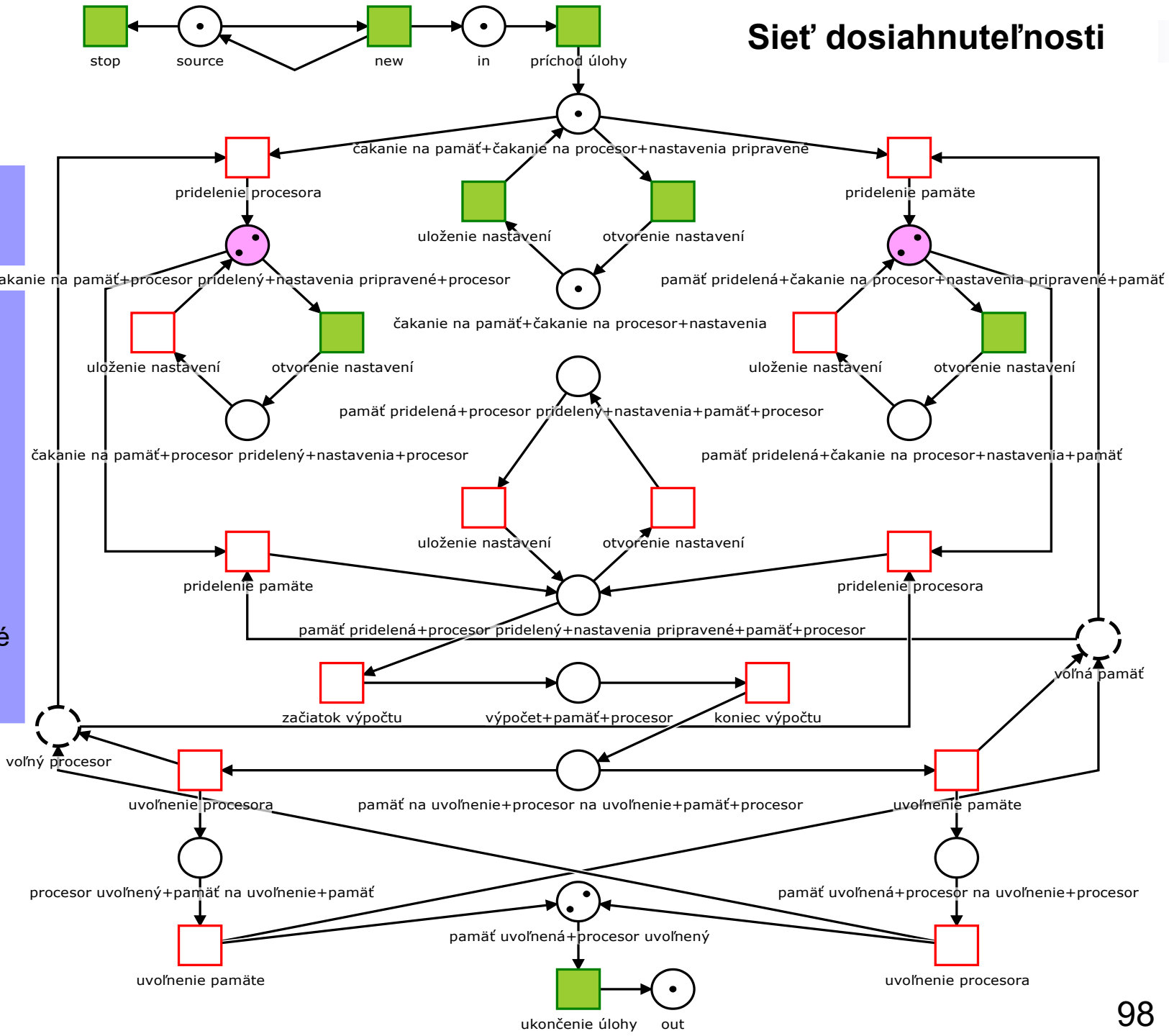
má uviaznutie sieť dosiahnuteľnosti



Sieť dosiahnuteľnosti

Výsledok 2:

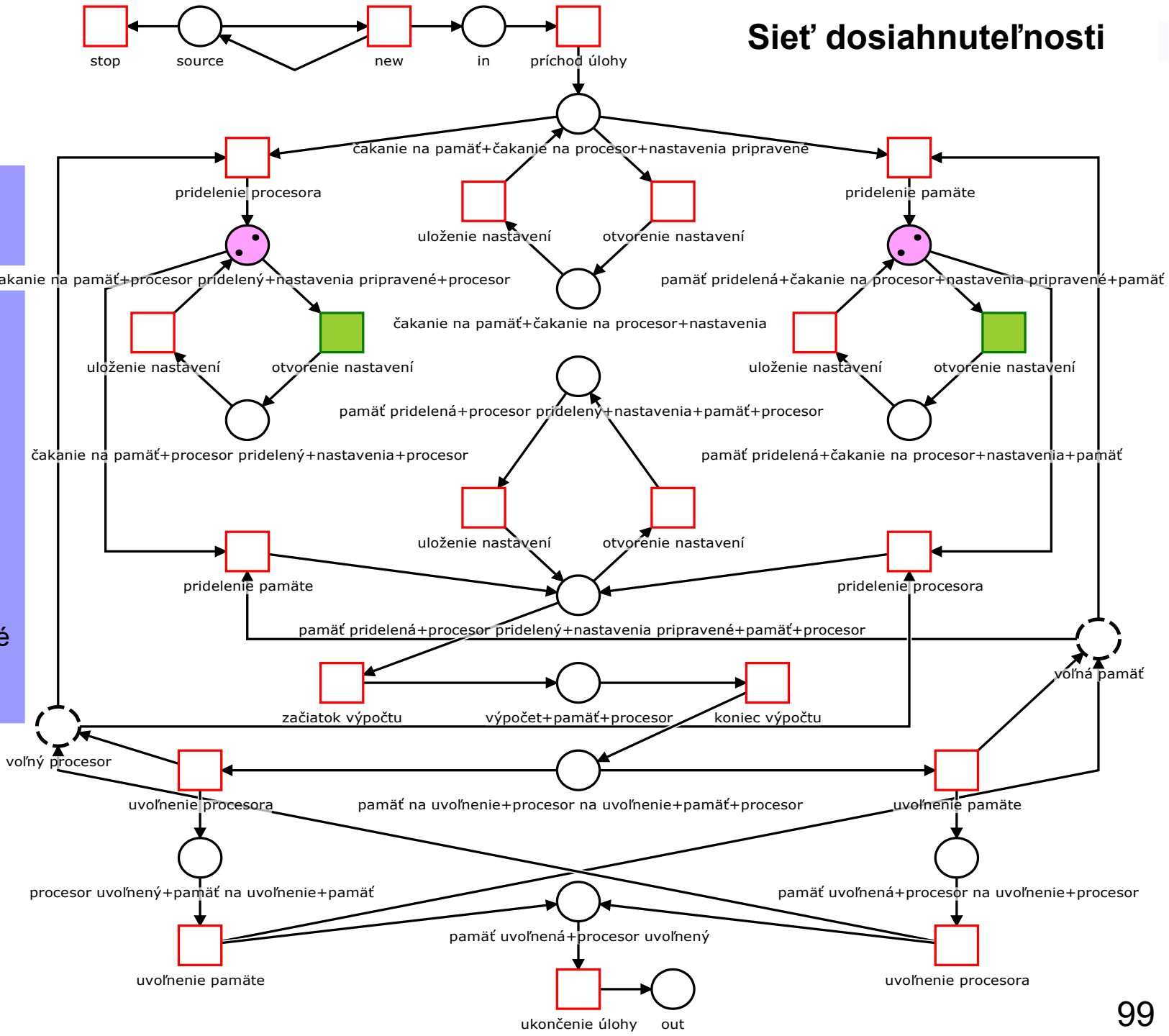
Ak m je dosiahnuteľné značkovanie nestatických miest, potom každé značkovanie nestatických miest menšie ako značkovanie m je dosiahnuteľné z počiatočného značkovania



Sieť dosiahnuteľnosti

Výsledok 2:

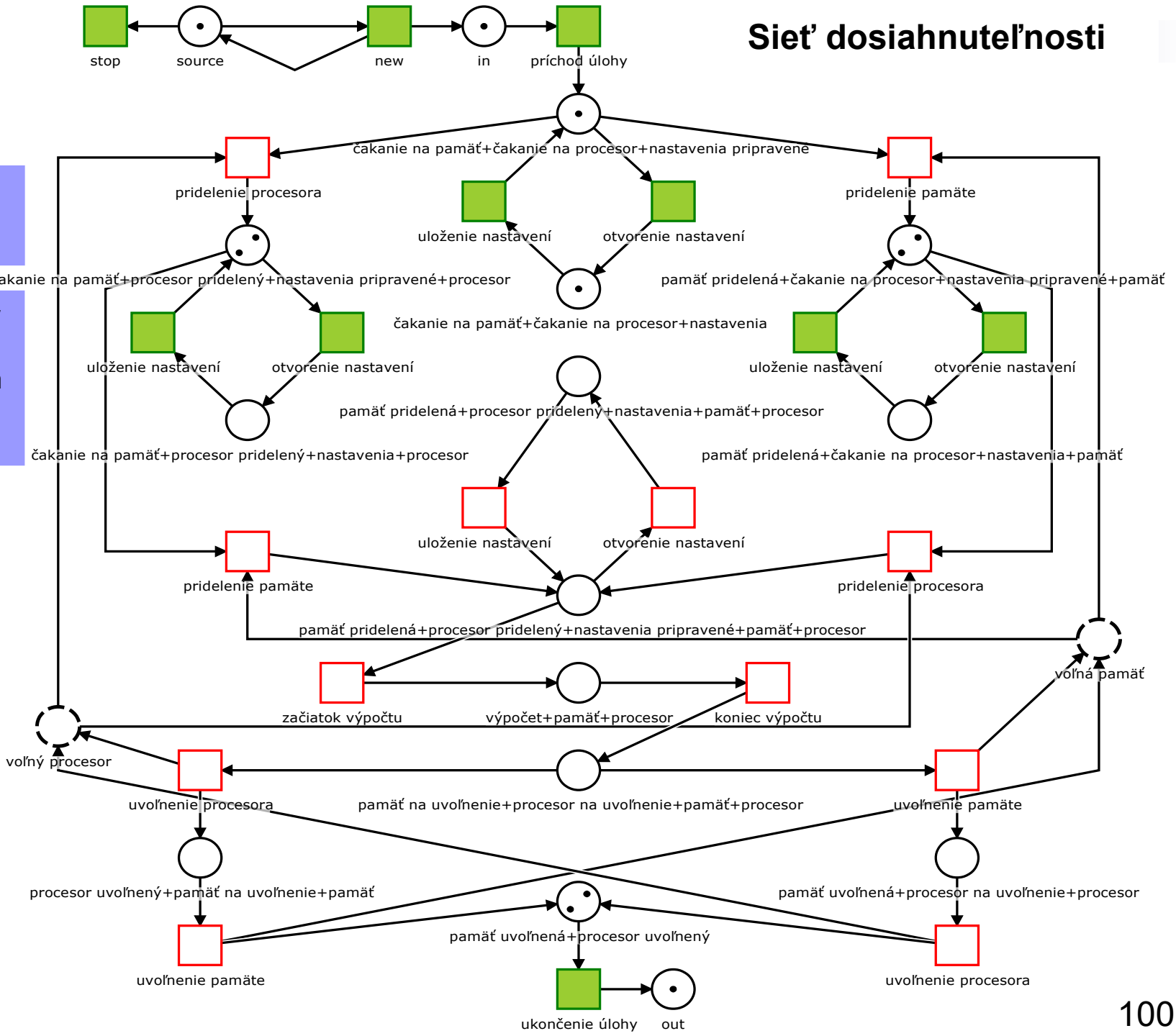
Ak m je dosiahnuteľné značkovanie nestatických miest, potom každé značkovanie nestatických miest menšie ako značkovanie m je dosiahnuteľné z počiatočného značkovania



Sieť dosiahnuteľnosti

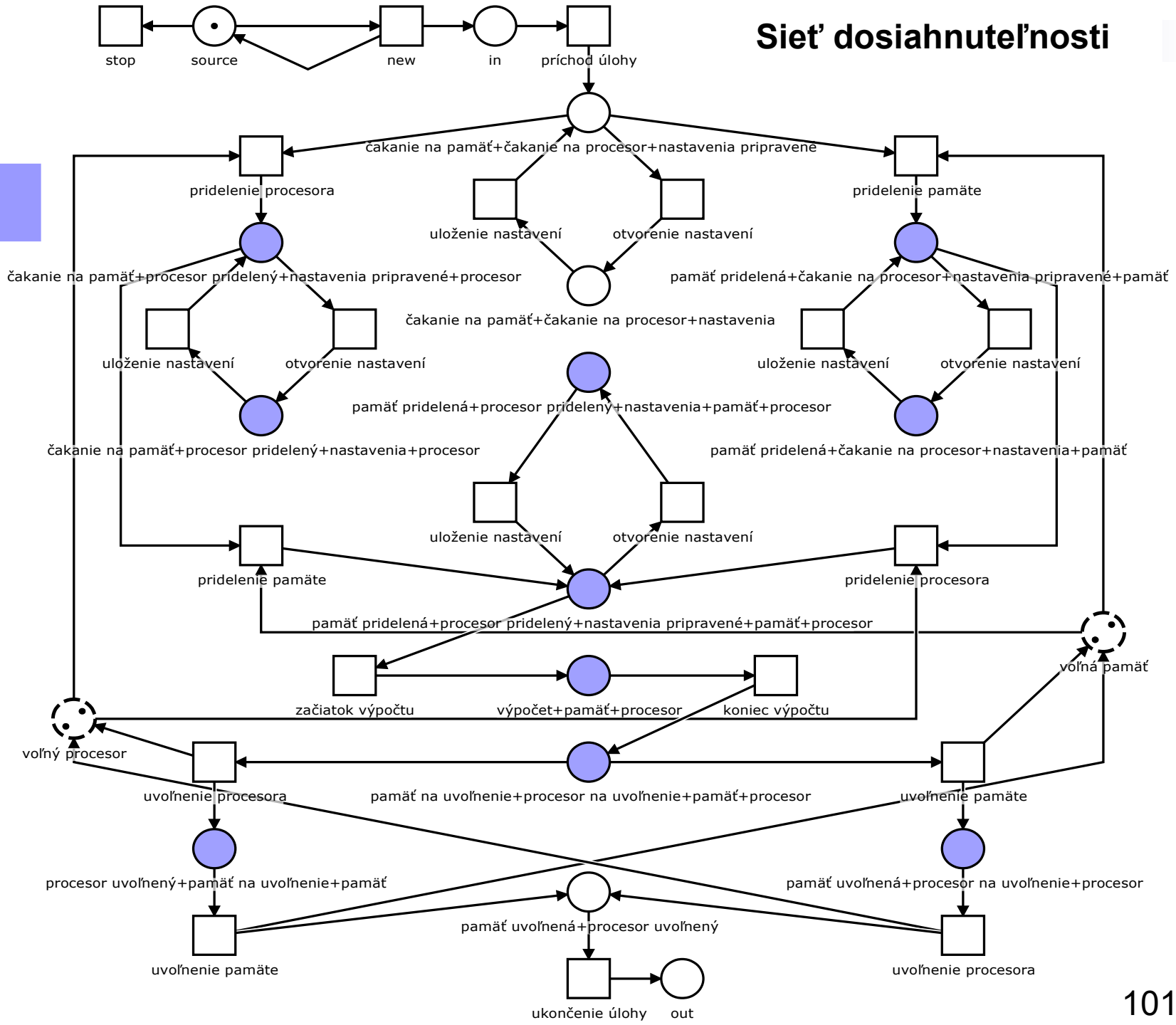
Výsledok 3:

Súčet značiek v nestatických miestach okrem miesta source neklesá.



Sieť dosiahnuteľnosti

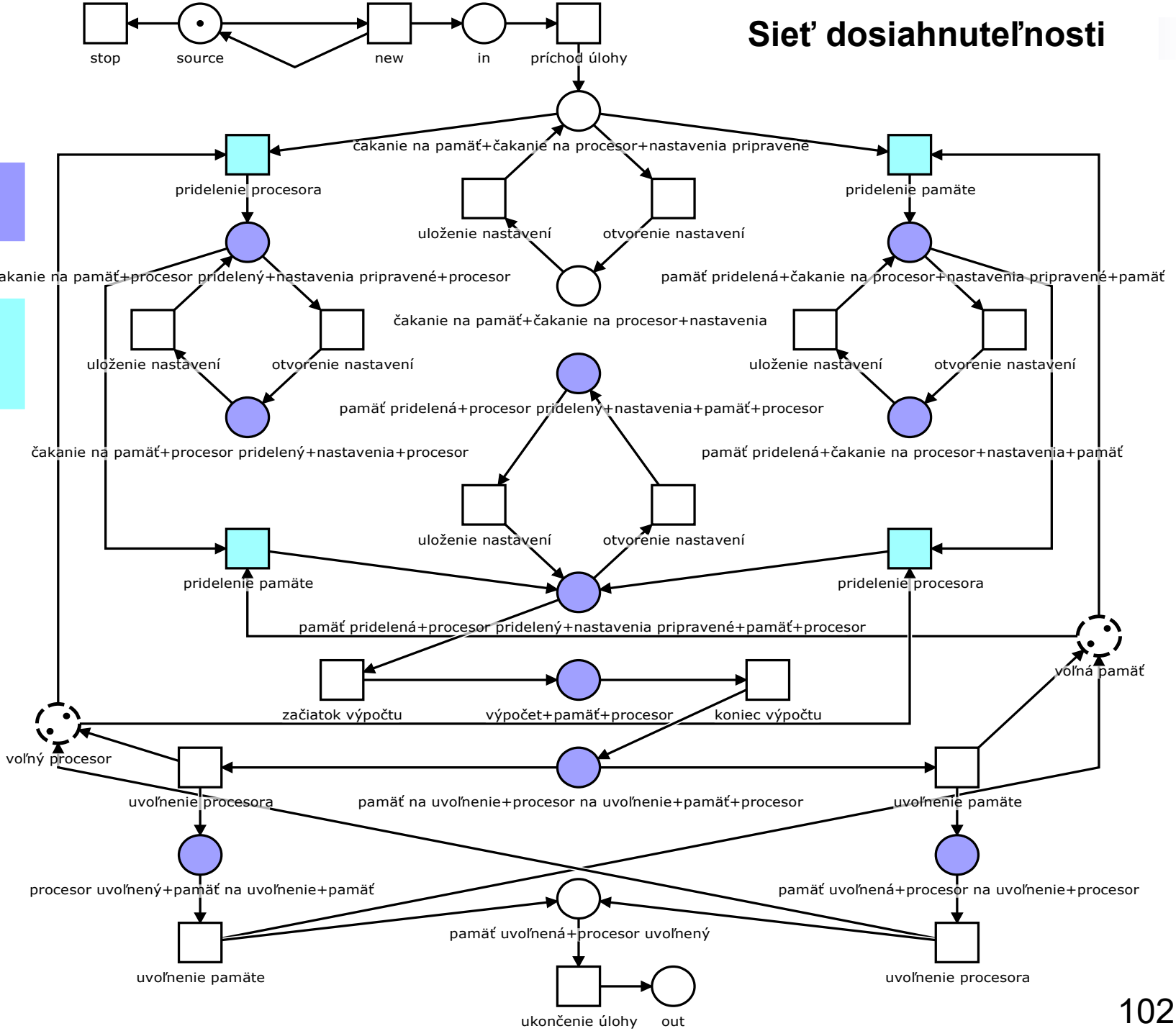
Miesta so zdrojmi



Sieť dosiahnuteľnosti

Miesta so zdrojmi

Prechody vyžadujúce zdroje

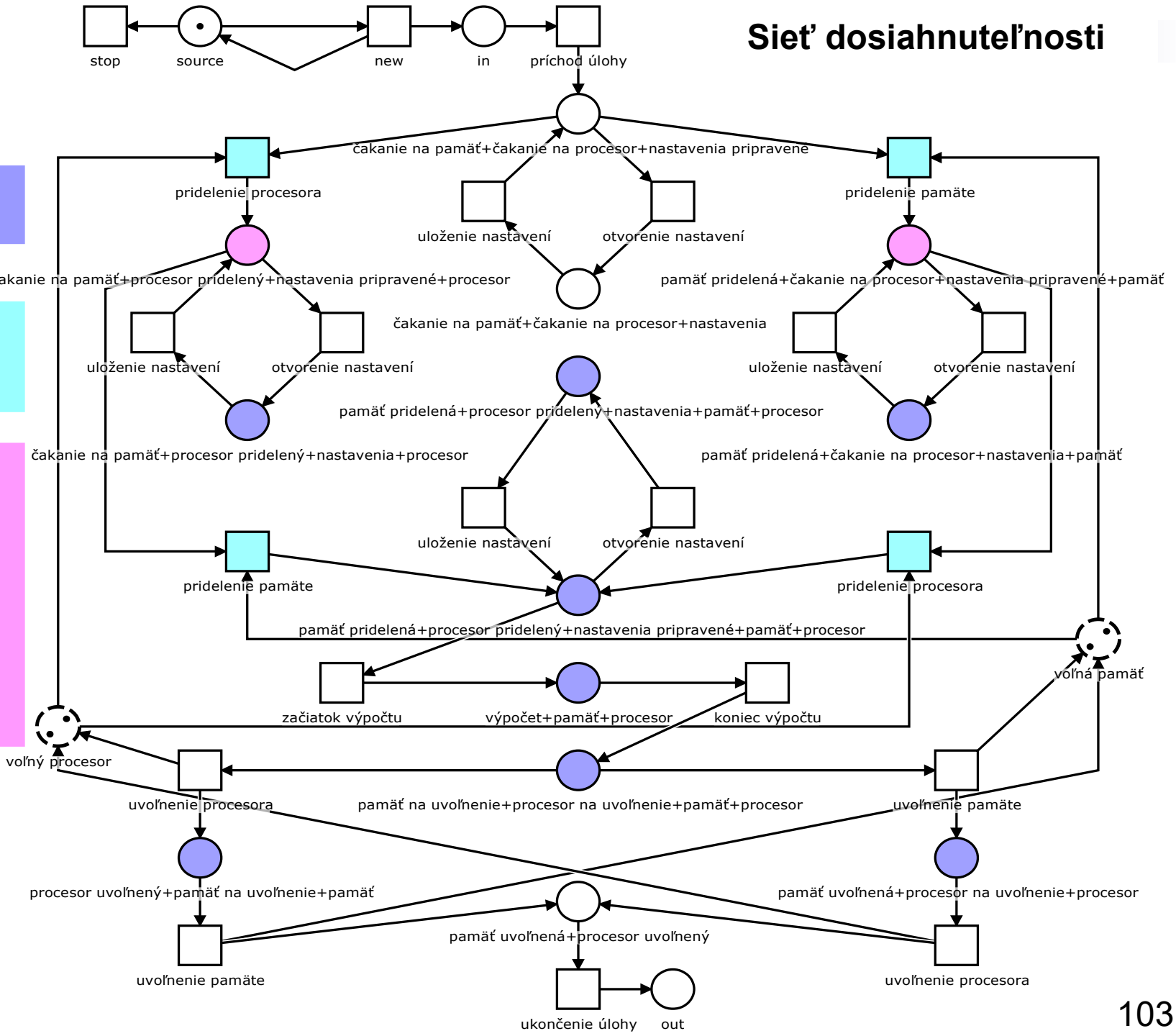


Sieť dosiahnuteľnosti

Miesta so zdrojmi

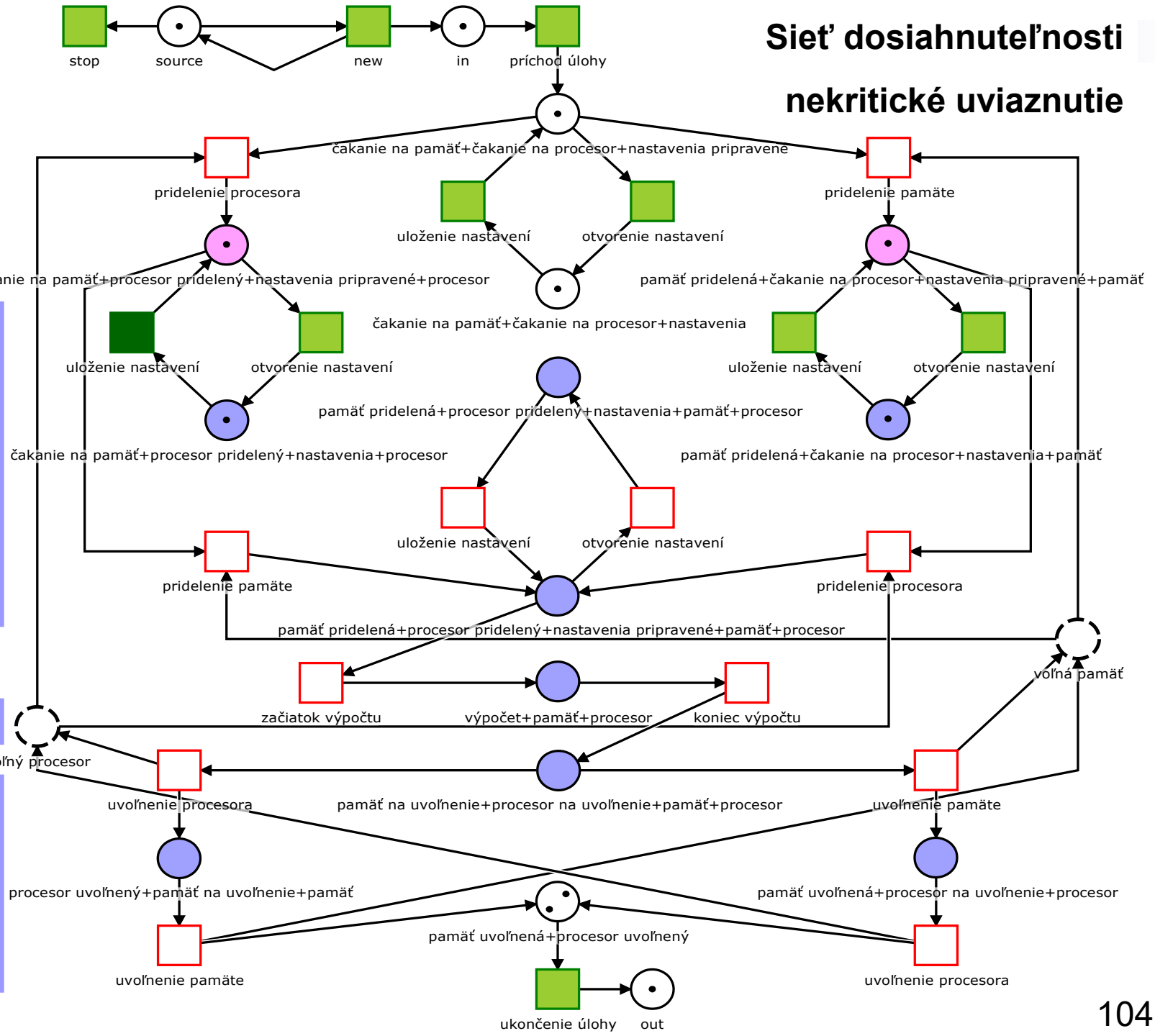
Prechody vyžadujúce zdroje

Kritické miesta = miesta so zdrojmi, z ktorých spotrebuje prechod vyžadujúci zdroje



Sieť dosiahnuteľnosti

nekritické uviaznutie



Kritické uviaznutia

– spomedzi miest so zdrojmi sú označkované

iba kritické miesta

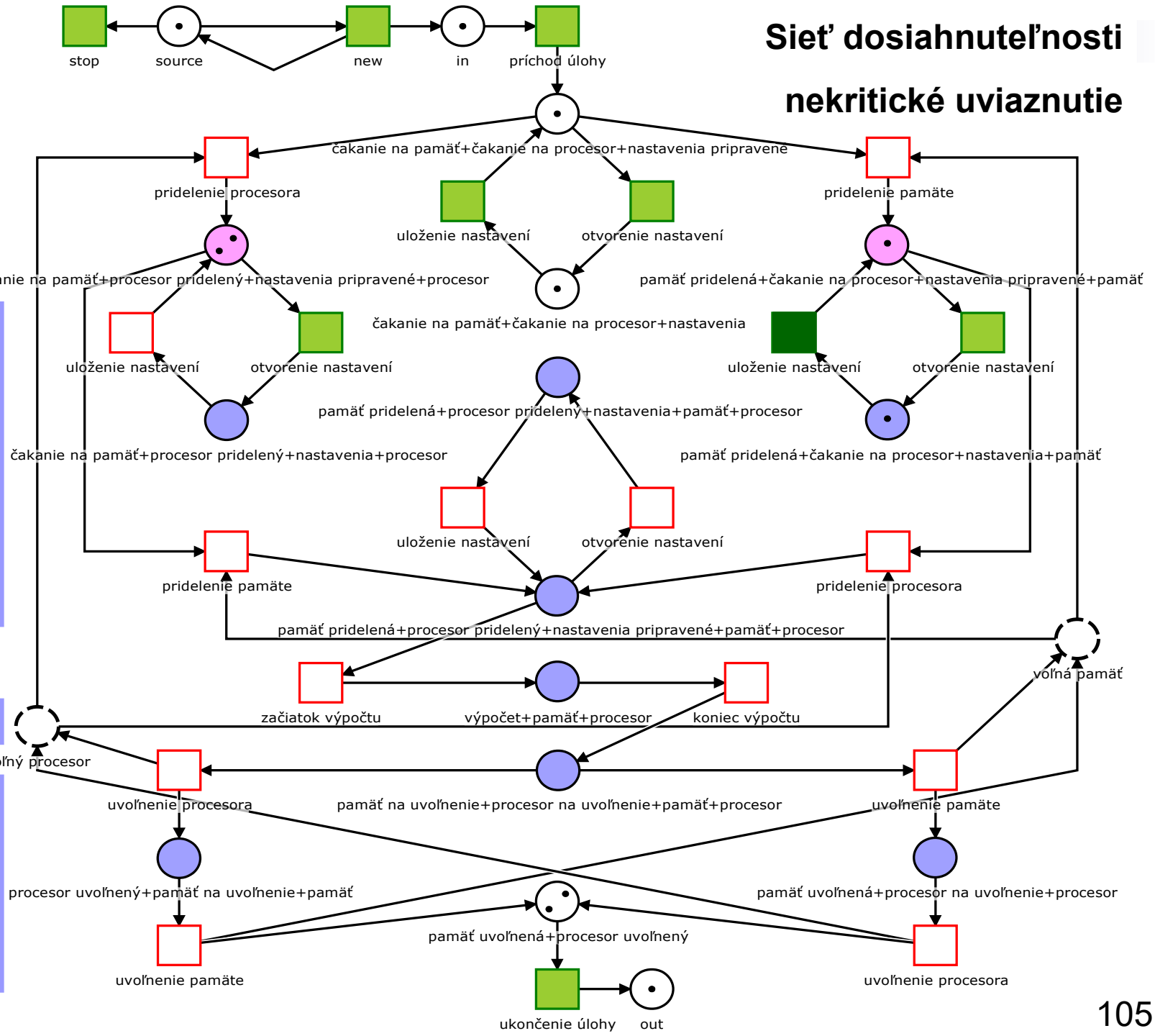
Výsledok 4:

Z každého uviaznutia je dosiahnuteľné

kritické uviaznutie

Sieť dosiahnuteľnosti

nekritické uviaznutie



Kritické uviaznutia

– spomedzi miest so zdrojmi sú označkované

iba kritické miesta

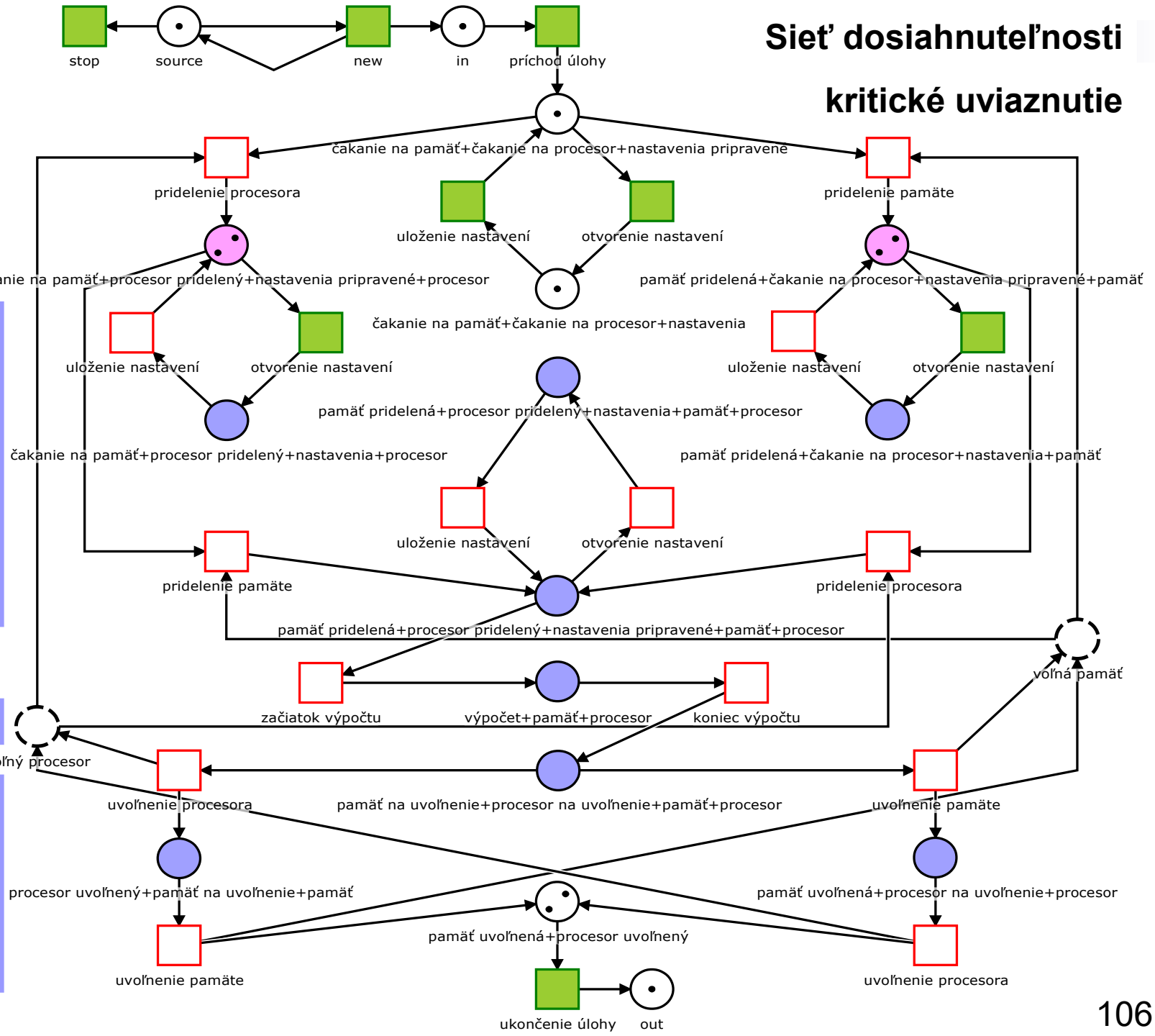
Výsledok 4:

Z každého uviaznutia je dosiahnuteľné

kritické uviaznutie

Sieť dosiahnuteľnosti

kritické uviaznutie



Kritické uviaznutia

– spomedzi miest so zdrojmi sú označkované

iba kritické miesta

Výsledok 4:

Z každého uviaznutia je dosiahnuteľné

kritické uviaznutie

Sieť dosiahnuteľnosti

kritické uviaznutie

Základné uviaznutia

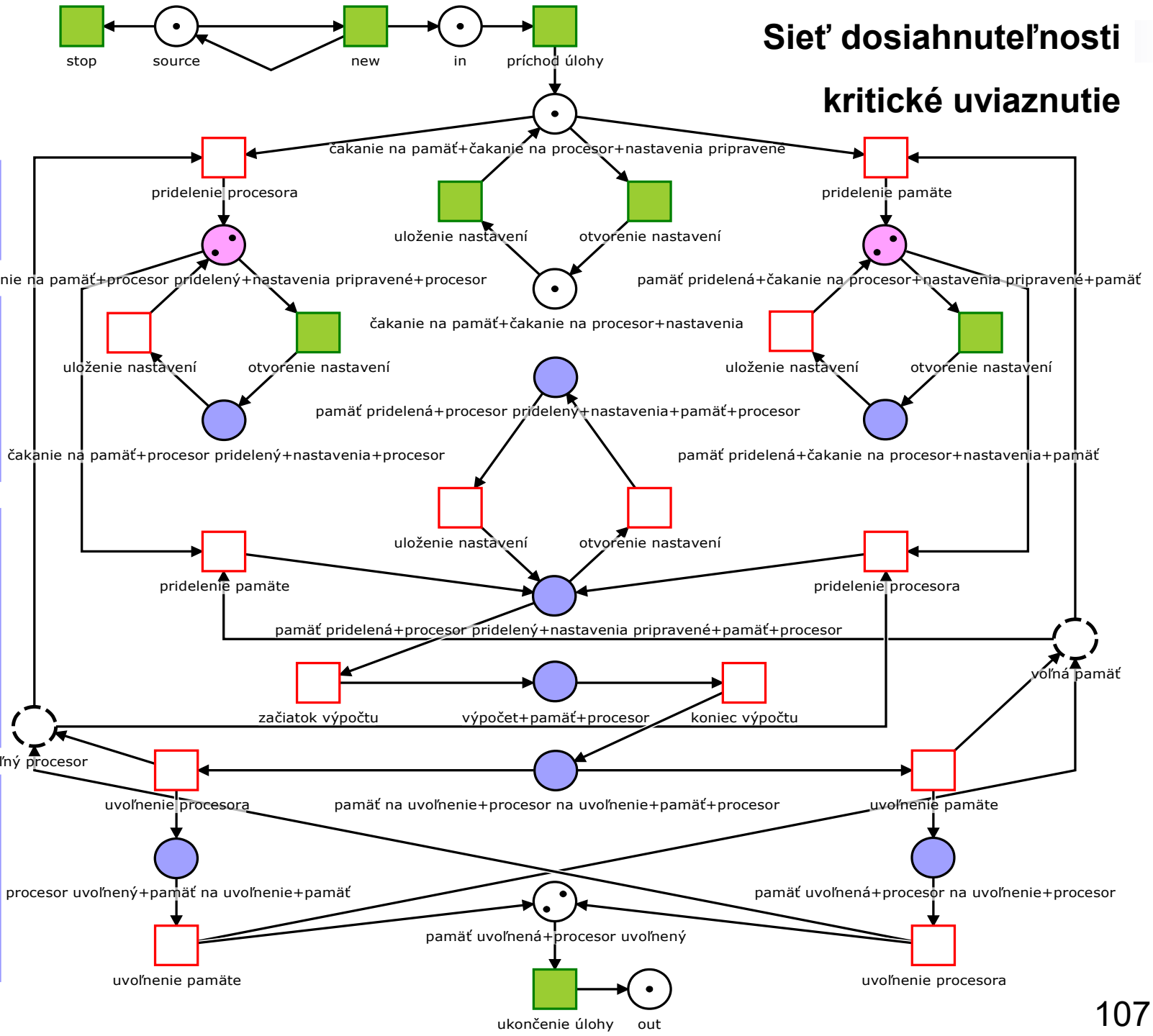
spomedzi všetkých miest sú označkované

iba kritické miesta

Výsledok 5:

Ak odstránime v kritickom uviaznutí všetky značky okrem značiek v kritických miestach

dostaneme základné uviaznutie



Sieť dosiahnuteľnosti

základné uviaznutie

Základné uviaznutia

spomedzi všetkých miest sú označkované

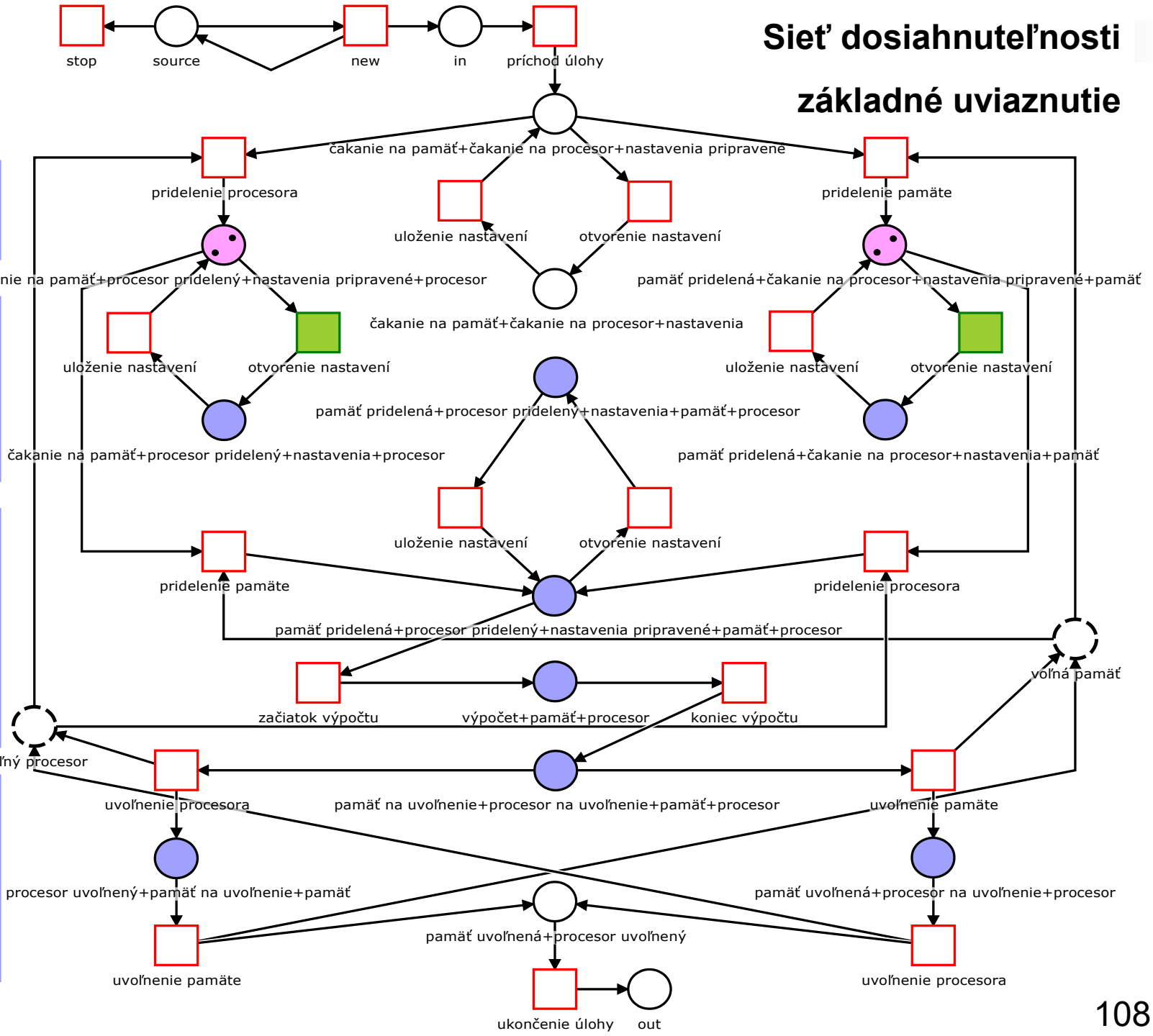
iba kritické miesta

Výsledok 5:

Ak odstránime v kritickom uviaznutí všetky značky okrem značiek v kritických miestach

dostaneme

základné uviaznutie



Sieť dosiahnuteľnosti

základné uviaznutie

Základné uviaznutia

spomedzi všetkých miest sú označkované

iba kritické miesta

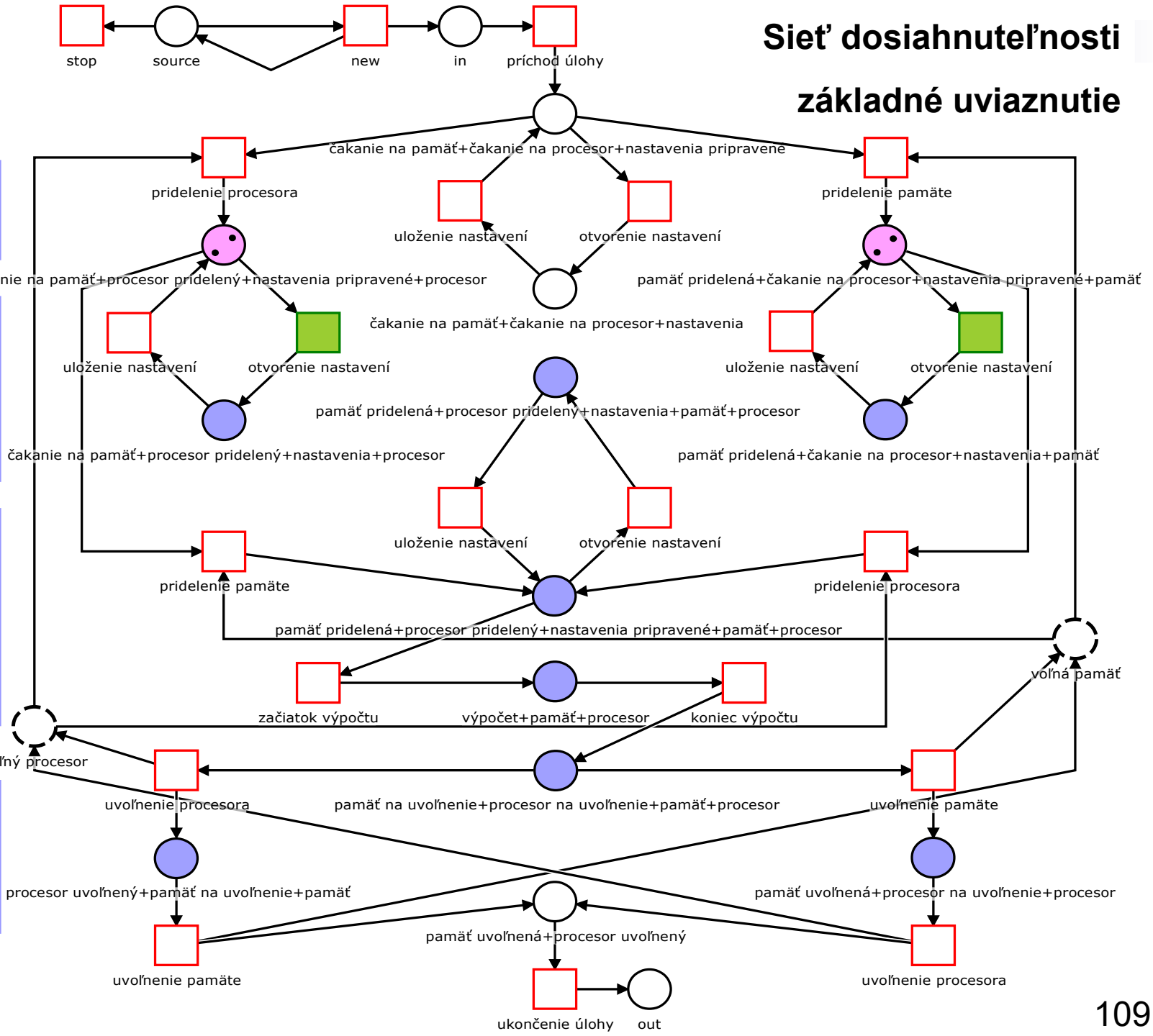
Výsledok 5:

Ak má sieť uviaznutia, potom má

základné uviaznutia

Stačí overovať

základné uviaznutia



Základné uviaznutia – spomedzi všetkých miest sú označkované iba kritické miesta

Výsledok 6:

- Kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný
- Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený
- Stačí vyšetriť základné uviaznutia pre takýto ohraničený a teda konečný počet inštancií

Základné uviaznutia – spomedzi všetkých miest sú označkované iba kritické miesta

Výsledok 6:

- Kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný
- Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený
- Stačí vyšetriť základné uviaznutia pre takýto ohraničený a teda konečný počet inštancií

Nech pb(k,s) = m0(s) ÷ k(ds), kde symbol ÷ označuje celočíselné delenie, k je kritické miesto siete dosiahnuteľnosti, s je statické miesto a ds je jeho komplementárne určujúce miesto v originálnej workflow sieti.

Základné uviaznutia – spomedzi všetkých miest sú označkované iba kritické miesta

Výsledok 6:

- Kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný
- Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený
- Stačí vyšetriť základné uviaznutia pre takýto ohraničený a teda konečný počet inštancií

Nech pb(k,s) = m0(s) ÷ k(ds), kde symbol ÷ označuje celočíselné delenie, k je kritické miesto siete dosiahnuteľnosti, s je statické miesto a ds je jeho komplementárne určujúce miesto v originálnej workflow sieti.

Nech sbound(k) je minimum hodnôt pb(k,s) pre všetky statické miesta.
Počet značiek v kritickom mieste k je zhora ohraničený hodnotou sbound(k).

Základné uviaznutia – spomedzi všetkých miest sú označkované iba kritické miesta

Výsledok 6:

- Kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný
- Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený
- Stačí vyšetriť základné uviaznutia pre takýto ohraničený a teda konečný počet inštancií

Nech pb(k,s) = m0(s) ÷ k(ds), kde symbol ÷ označuje celočíselné delenie, k je kritické miesto siete dosiahnuteľnosti, s je statické miesto a ds je jeho komplementárne určujúce miesto v originálnej workflow sieti.

Nech sbound(k) je minimum hodnôt pb(k,s) pre všetky statické miesta.
Počet značiek v kritickom mieste k je zhora ohraničený hodnotou sbound(k).

Nech sbound(K) je súčet hodnôt sbound(k) pre všetky kritické miesta k.
Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený hodnotou sbound(K).

Základné uviaznutia – spomedzi všetkých miest sú označkované iba kritické miesta

Výsledok 6:

- Kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný
- Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený
- Stačí vyšetriť základné uviaznutia pre takýto ohraničený a teda konečný počet inštancií

Nech pb(k,s) = m0(s) ÷ k(ds), kde symbol ÷ označuje celočíselné delenie, k je kritické miesto siete dosiahnuteľnosti, s je statické miesto a ds je jeho komplementárne určujúce miesto v originálnej workflow sieti.

Nech sbound(k) je minimum hodnôt pb(k,s) pre všetky statické miesta.
Počet značiek v kritickom mieste k je zhora ohraničený hodnotou sbound(k).

Nech sbound(K) je súčet hodnôt sbound(k) pre všetky kritické miesta k.
Počet inštancií, pre ktoré sieť môže mať základné uviaznutie je zhora ohraničený hodnotou sbound(K).

Lepší odhad (ale aj výpočtovo náročnejší) v práci pomocou celočíselného lineárneho programovania – kvôli výpočtovej zložitosti sme ho pri overení nepoužili.

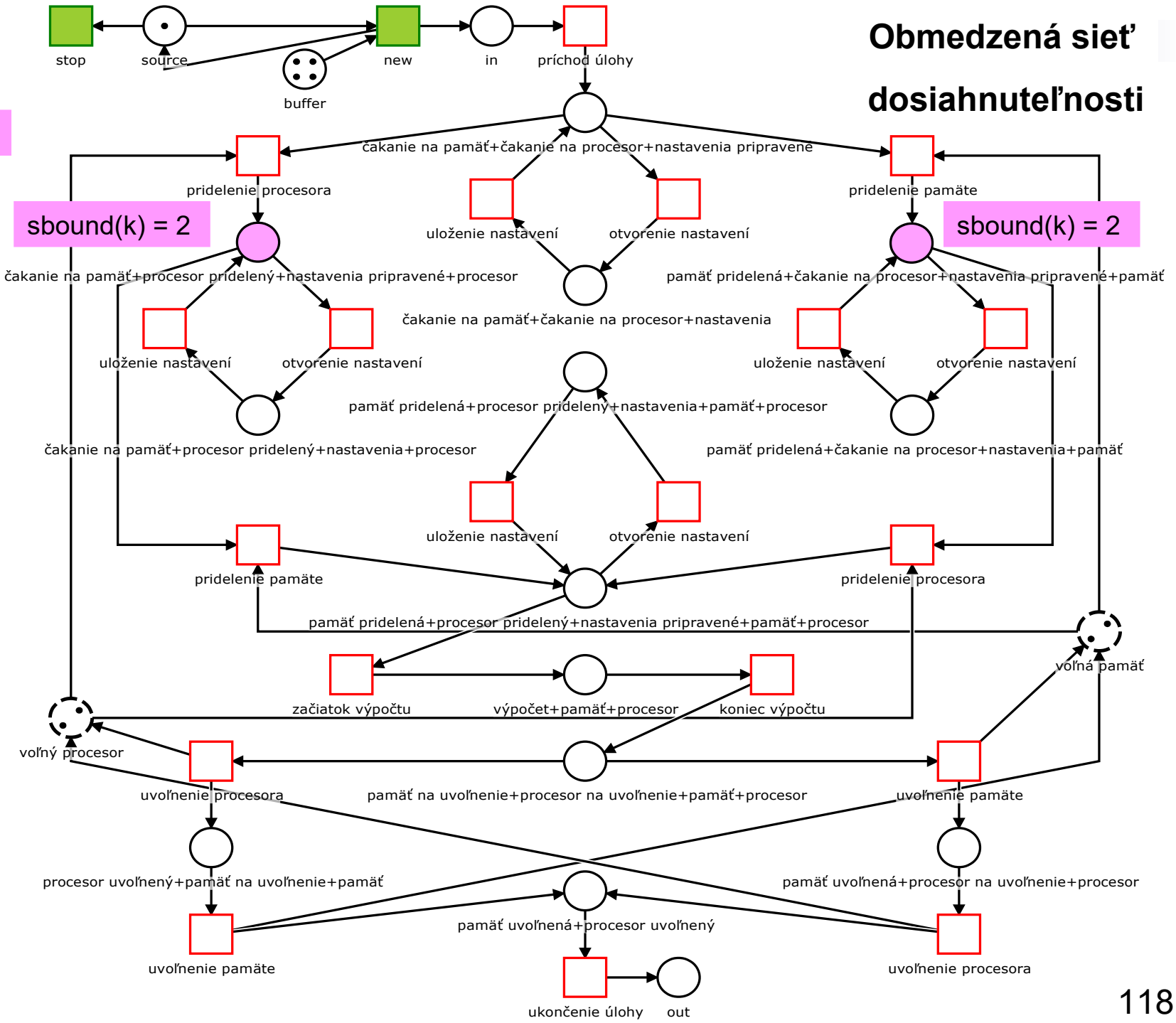
Algoritmus na detekciu základných uviaznutí:
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti

Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“

Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“
3) Vytvor zo siete dosiahnuteľnosti ohraničenú obmedzenú sieť dosiahnuteľnosti pre sbound(K)-inštancií, pridaním miesta buffer s sbound(K) značkami, z ktorého spotrebuje prechod new jednu značku

Obmedzená sieť dosiahnuteľnosti

sbound(K) = 4



Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“
3) Vytvor zo siete dosiahnuteľnosti ohraničenú obmedzenú sieť dosiahnuteľnosti pre sbound(K)-inštancií, pridaním miesta buffer s sbound(K) značkami, z ktorého spotrebuje prechod new jednu značku
4) Skonštruuj graf dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti

Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“
3) Vytvor zo siete dosiahnuteľnosti ohraničenú obmedzenú sieť dosiahnuteľnosti pre sbound(K)-inštancií, pridaním miesta buffer s sbound(K) značkami, z ktorého spotrebuje prechod new jednu značku
4) Skonštruuj graf dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti
5) Pri konštruovaní grafu dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti si pamätaj predchodcov každého dosiahnuteľného značkovania

Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“
3) Vytvor zo siete dosiahnuteľnosti ohraničenú obmedzenú sieť dosiahnuteľnosti pre sbound(K)-inštancií, pridaním miesta buffer s sbound(K) značkami, z ktorého spotrebuje prechod new jednu značku
4) Skonštruuj graf dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti
5) Pri konštruovaní grafu dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti si pamätaj predchodcov každého dosiahnuteľného značkovania
6) Zároveň skonštruuj zoznam značkovaní, v ktorých sú označkované iba kritické miesta

Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“
3) Vytvor zo siete dosiahnuteľnosti ohraničenú obmedzenú sieť dosiahnuteľnosti pre sbound(K)-inštancií, pridaním miesta buffer s sbound(K) značkami, z ktorého spotrebuje prechod new jednu značku
4) Skonštruuj graf dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti
5) Pri konštruovaní grafu dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti si pamätaj predchodcov každého dosiahnuteľného značkovania
6) Zároveň skonštruuj zoznam značkovaní, v ktorých sú označkované iba kritické miesta
7) Pre každé značkovanie, v ktorom sú označkované iba kritické miesta, over, či je predchodcom finálneho značkovania s rovnakým súčtom značiek v nestatických miestach, ak nie, označ ho za základné uviaznutie

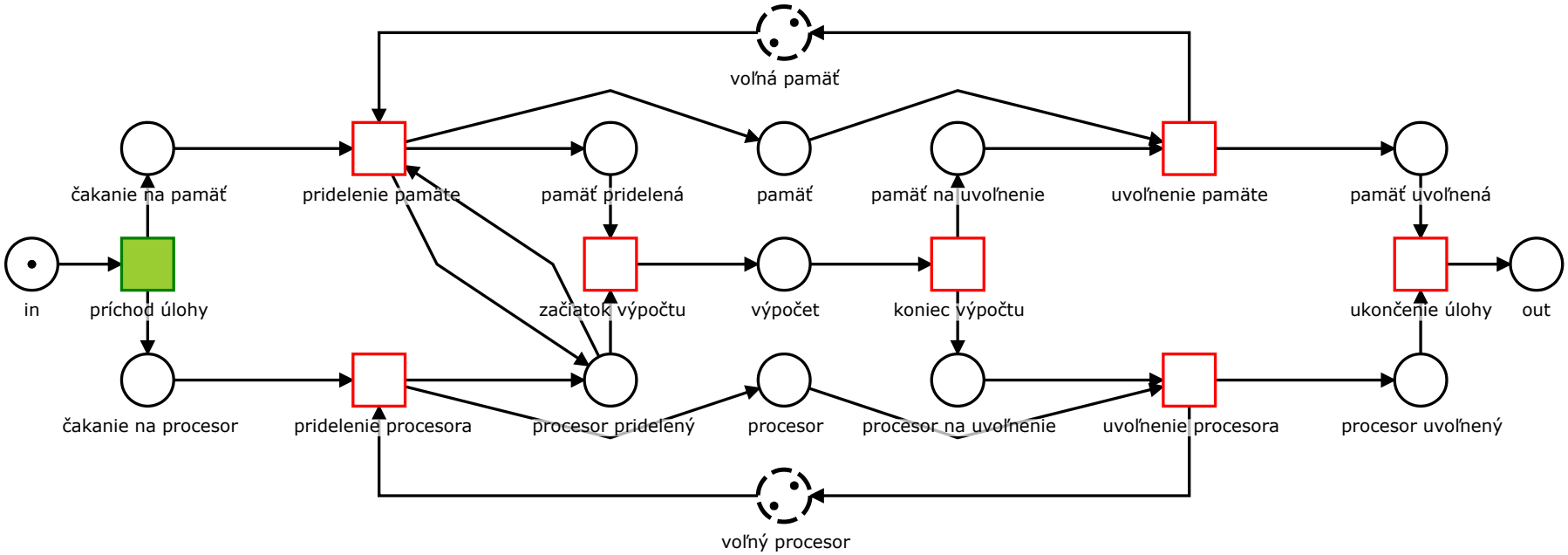
Algoritmus na detekciu základných uviaznutí::
1) Ak je workflow sieť korektná, zostroj sieť dosiahnuteľnosti
2) Ak sieť dosiahnuteľnosti nemá kritické miesta, potom zastav a vráť „sieť nemá uviaznutia“
3) Vytvor zo siete dosiahnuteľnosti ohraničenú obmedzenú sieť dosiahnuteľnosti pre sbound(K)-inštancií, pridaním miesta buffer s sbound(K) značkami, z ktorého spotrebuje prechod new jednu značku
4) Skonštruuj graf dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti
5) Pri konštruovaní grafu dosiahnuteľnosti obmedzenej siete dosiahnuteľnosti si pamätaj predchodcov každého dosiahnuteľného značkovania
6) Zároveň skonštruuj zoznam značkovaní, v ktorých sú označkované iba kritické miesta
7) Pre každé značkovanie, v ktorom sú označkované iba kritické miesta, over, či je predchodcom finálneho značkovania s rovnakým súčtom značiek v nestatických miestach, ak nie, označ ho za základné uviaznutie

Algoritmus nájde všetky základné uviaznutia

Budúci výskum:

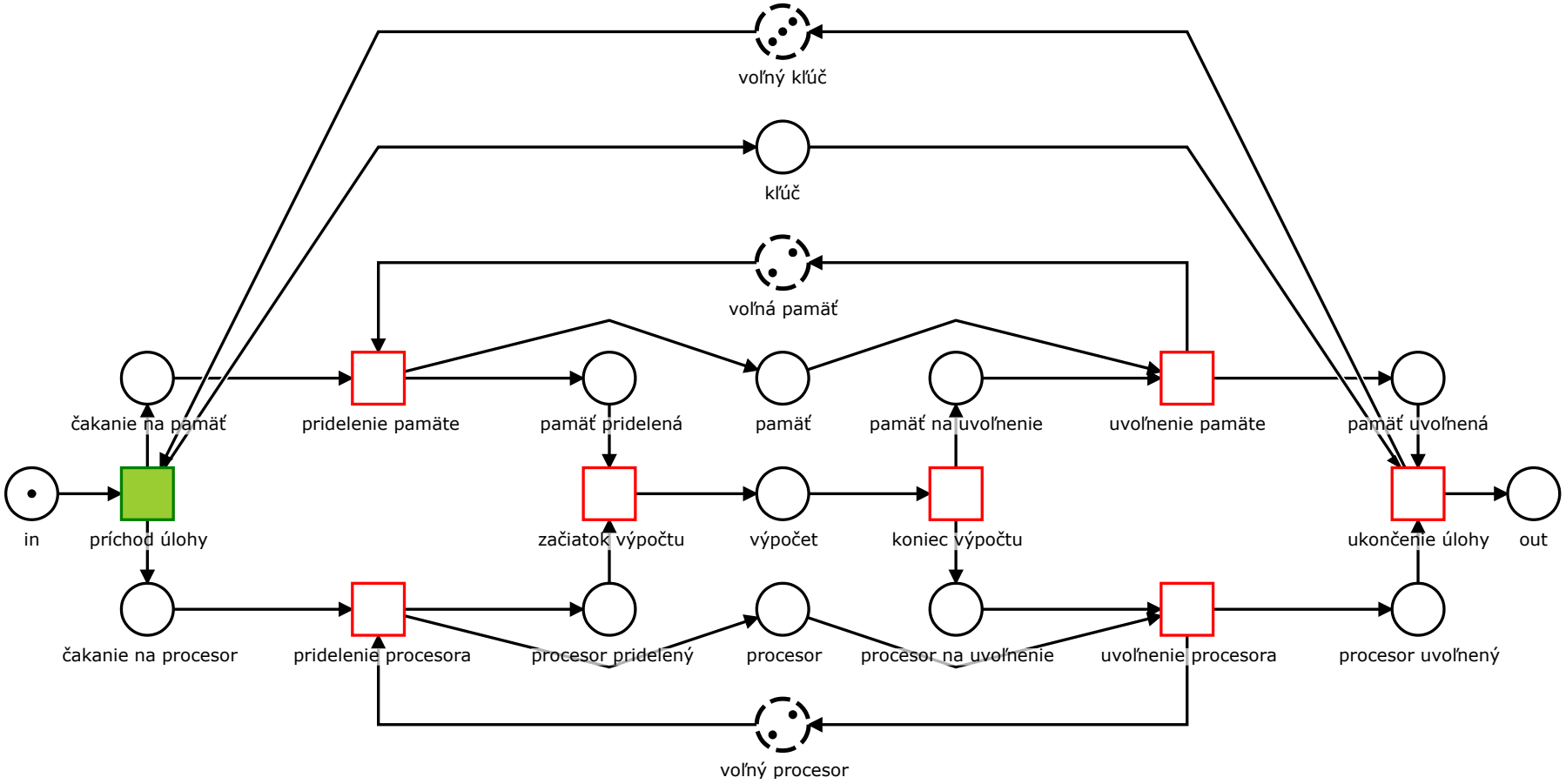
Ak má sieť uviaznutia, ako ju upraviť tak, aby uviaznutia boli odstránené

Budúci výskum:
Ak má sieť uviaznutia, ako ju upraviť tak, aby uviaznutia boli odstránené



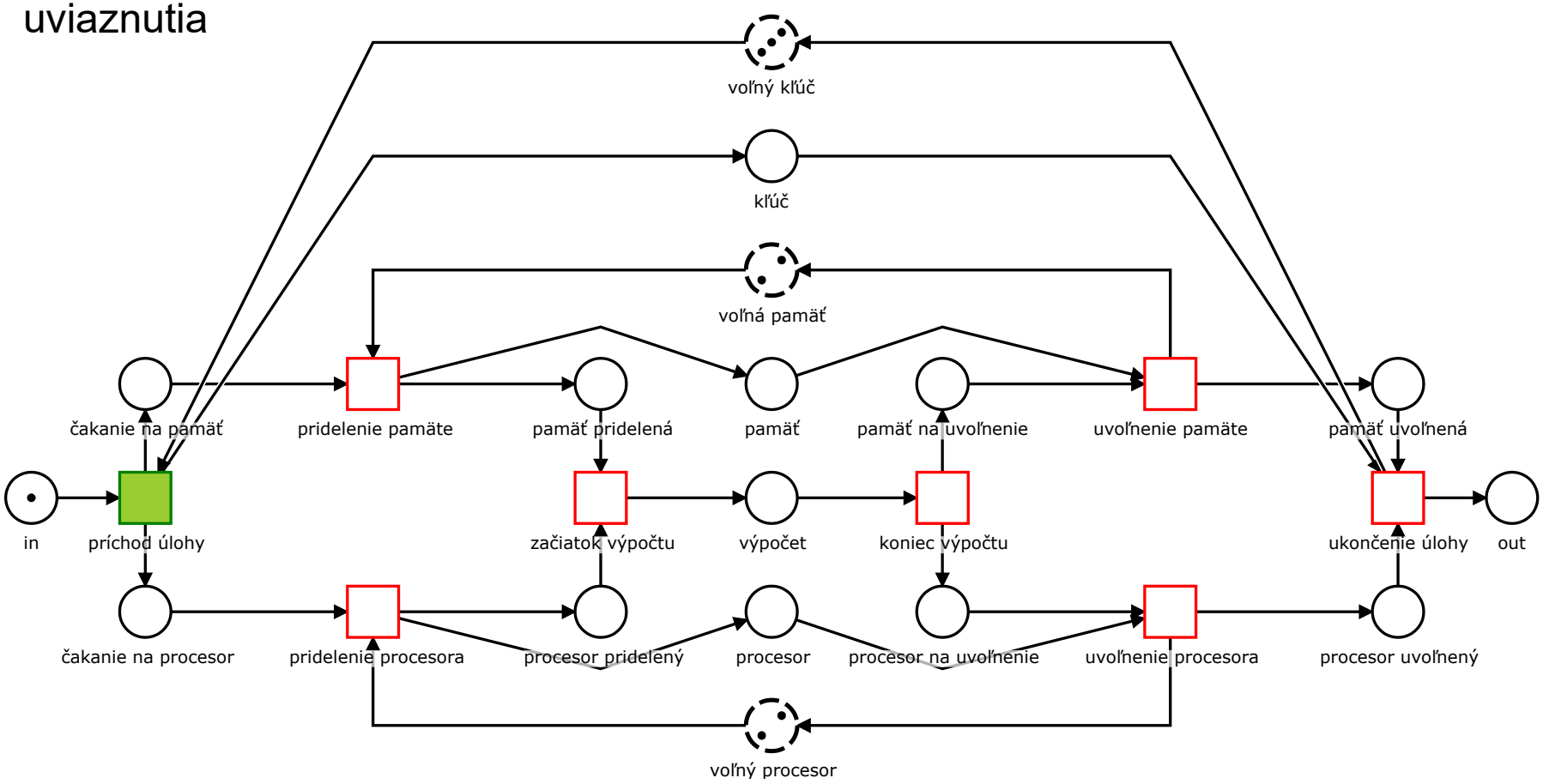
Upravená workflow sieť definuje poradie prideľovania zdrojov, najskôr procesor, potom pamäť
Upravená workflow sieť nemá žiadne uviaznutia pre ľubovoľný kladný počet značiek v statických miestach voľná pamäť a voľný procesor
Obmedzuje správanie sa jednej inštancie

Budúci výskum:
Ak má sieť uviaznutia, ako ju upraviť tak, aby uviaznutia boli odstránené



Upravená workflow sieť obmedzuje pridaním statického miesta voľný kľúč počet paralelne spracovaných inštancií
Upravená workflow sieť nemá uviaznutia pre dané počiatočné značkovanie.
Správanie sa jednej inštancie nie je obmedzené

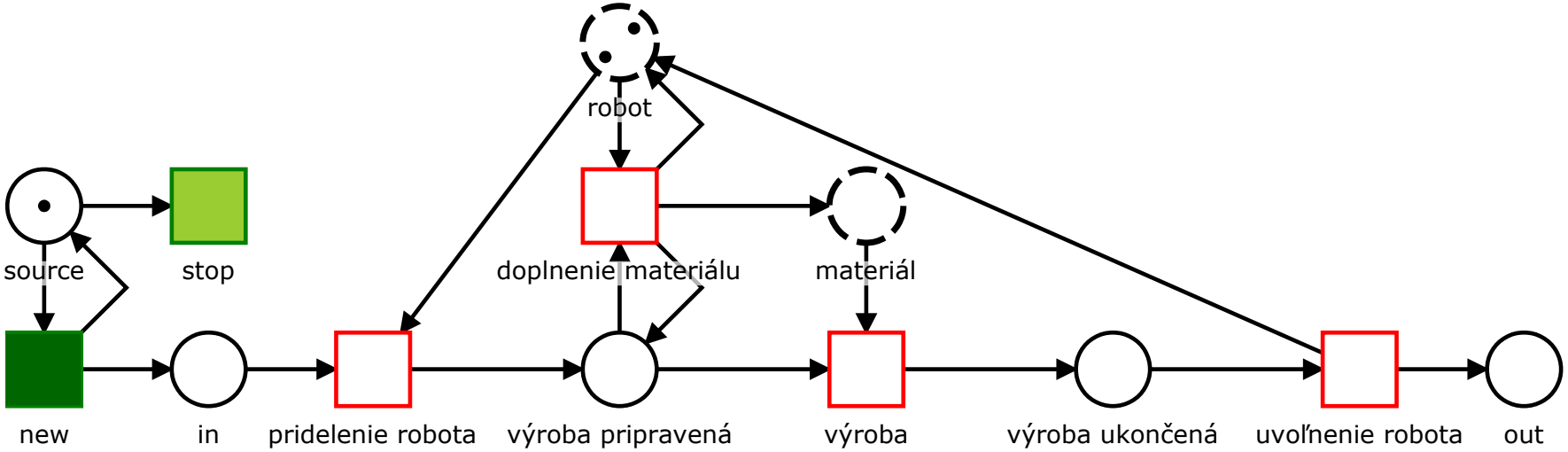
Budúci výskum: Ako určiť množinu značkovaní statických miest, pre ktorú má sieť uviaznutia, resp. množinu značkovaní statických miest, pre ktorú sieť nemá uviaznutia



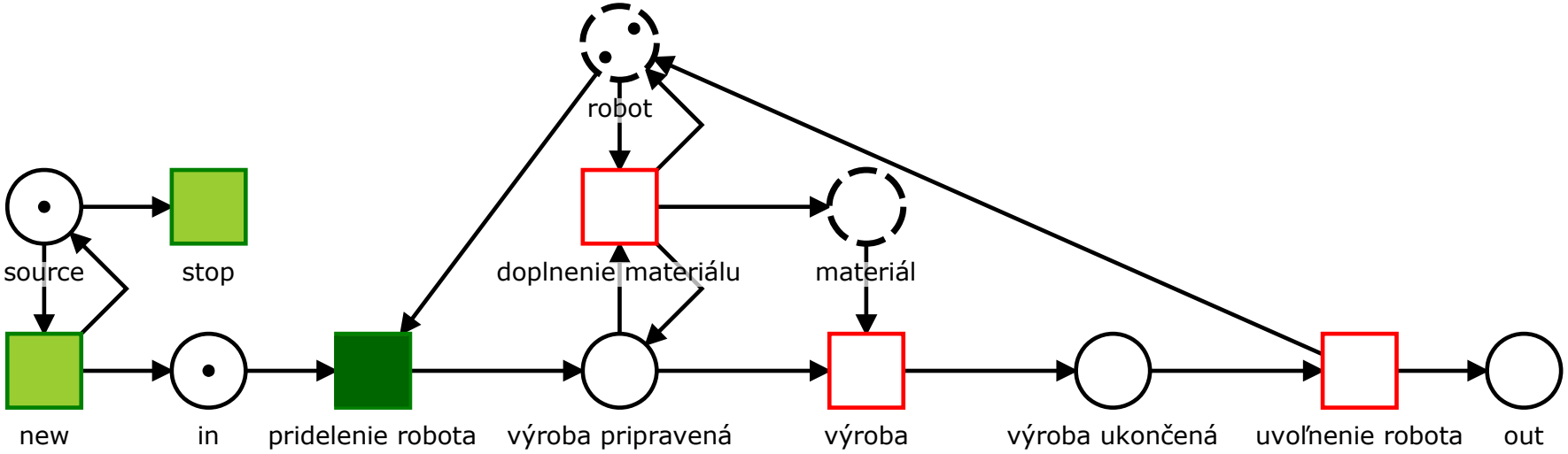
Upravená workflow sieť obmedzuje pridaním statického miesta voľný kľúč počet paralelne spracovaných inštancií

Ak počet značiek v statickom mieste voľný kľúč nie je menší ako súčet počtu značiek v statických miestach voľná pamäť a voľný procesor, sieť má uviaznutia, inak sieť nemá uviaznutia

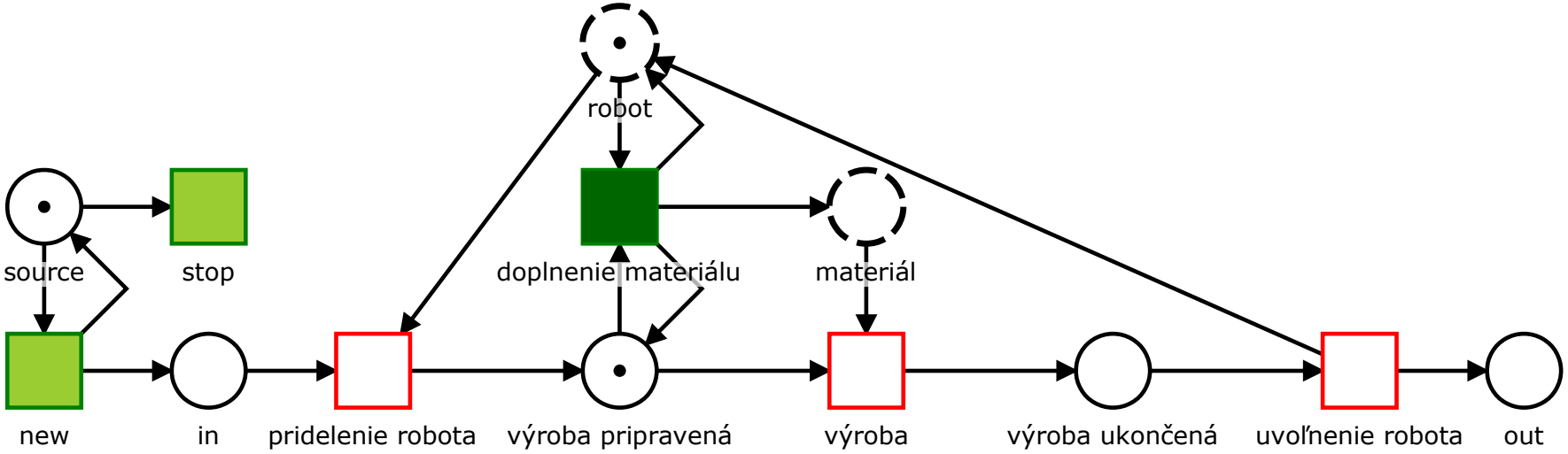
Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu

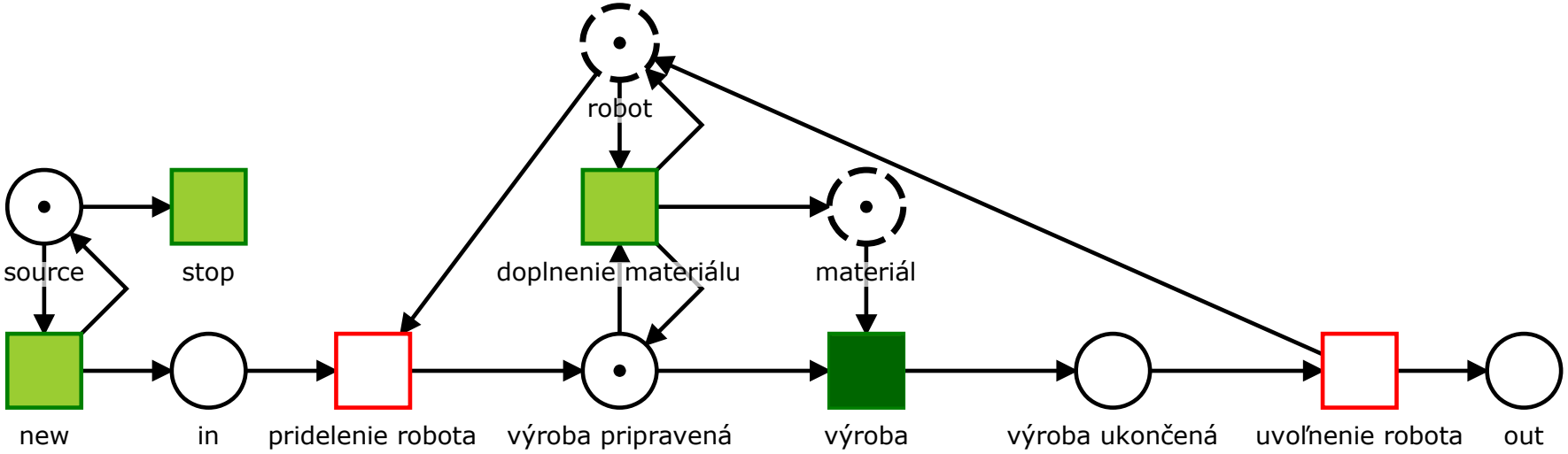


Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



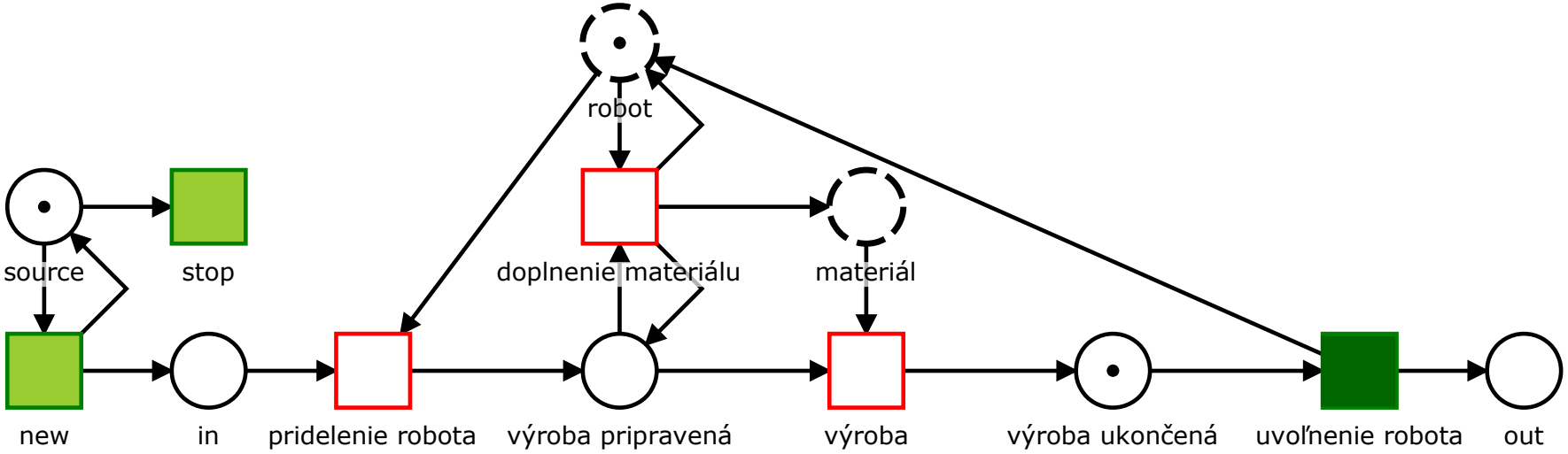
Možnosť ľubovoľne veľa krát doplniť tovar v rámci inštancie
(potrebný pridelený robot a voľný robot)

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



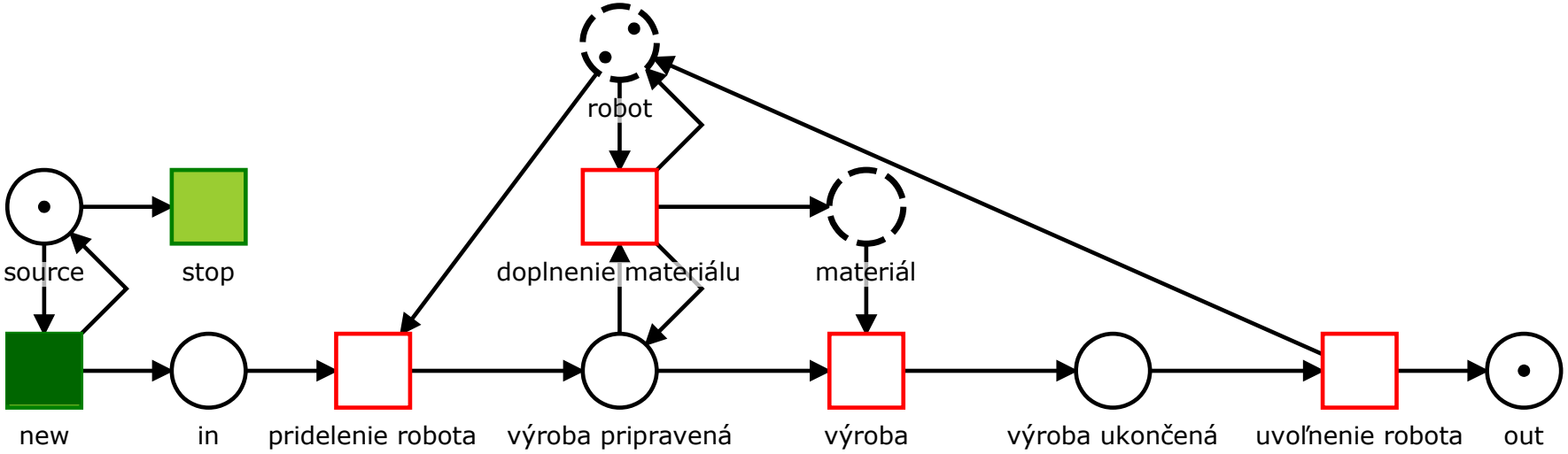
Možnosť ľubovoľne veľa krát doplniť tovar v rámci inštancie (potrebný pridelený robot a voľný robot)
Možnosť začať výrobu

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



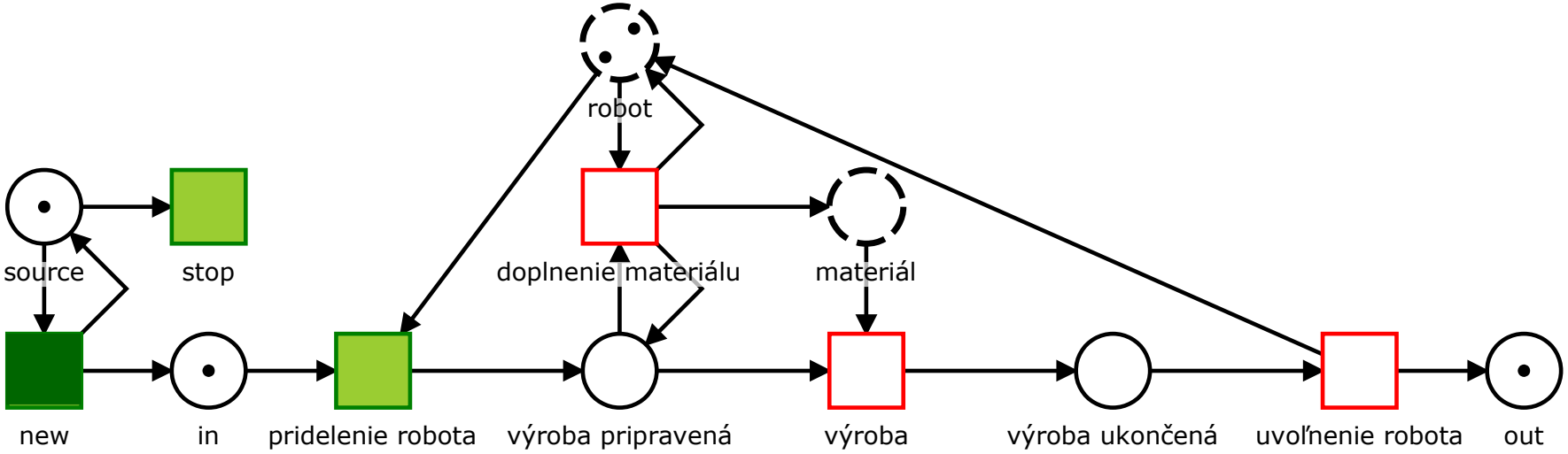
Nie je možnosť doplniť tovar v rámci inštancie

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



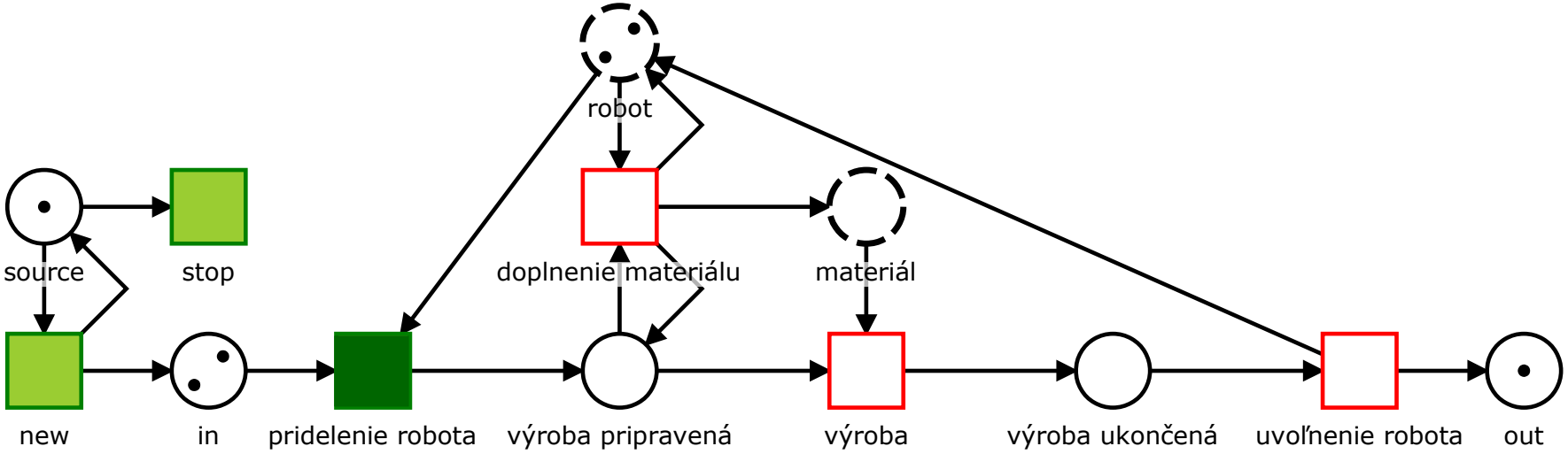
Ukončená inštancia

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



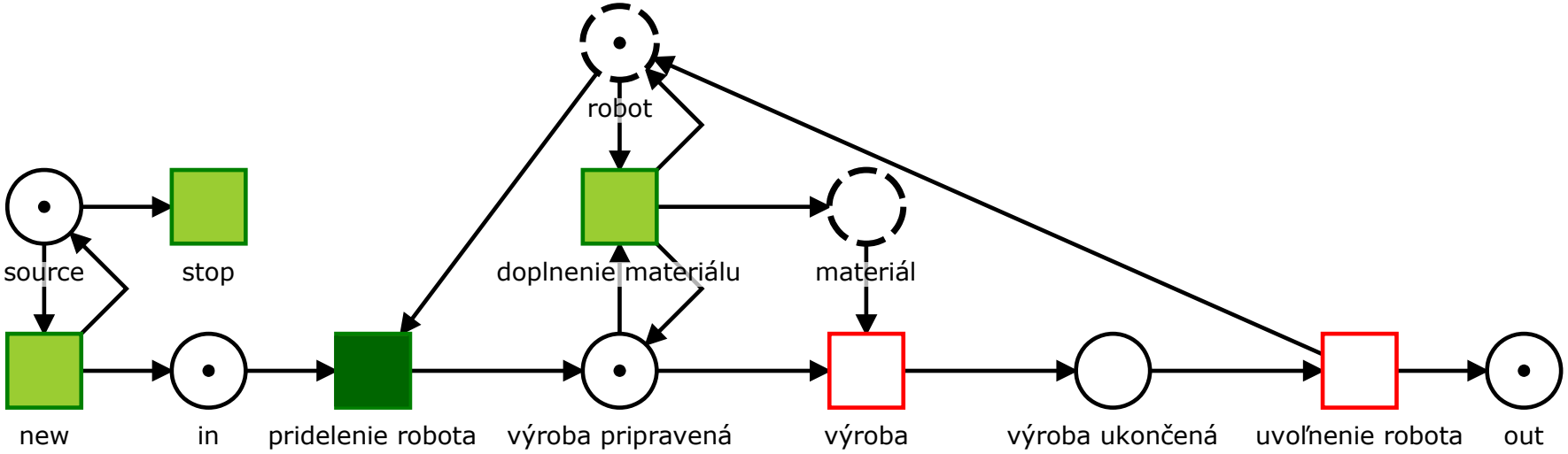
Nová inštancia

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



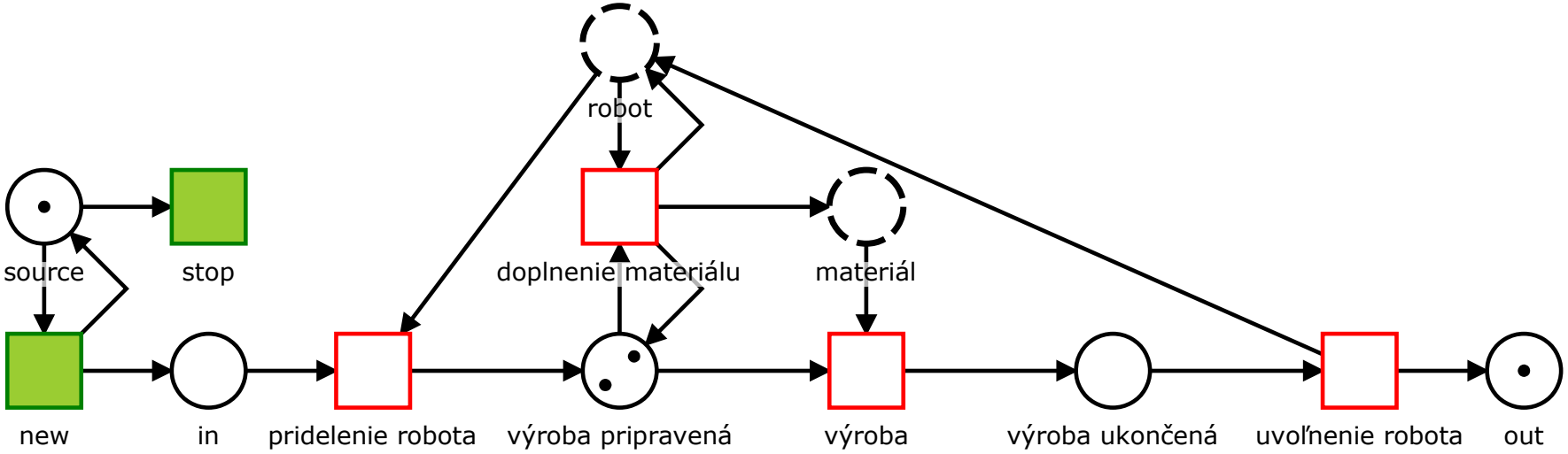
Ďalšia inštancia

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



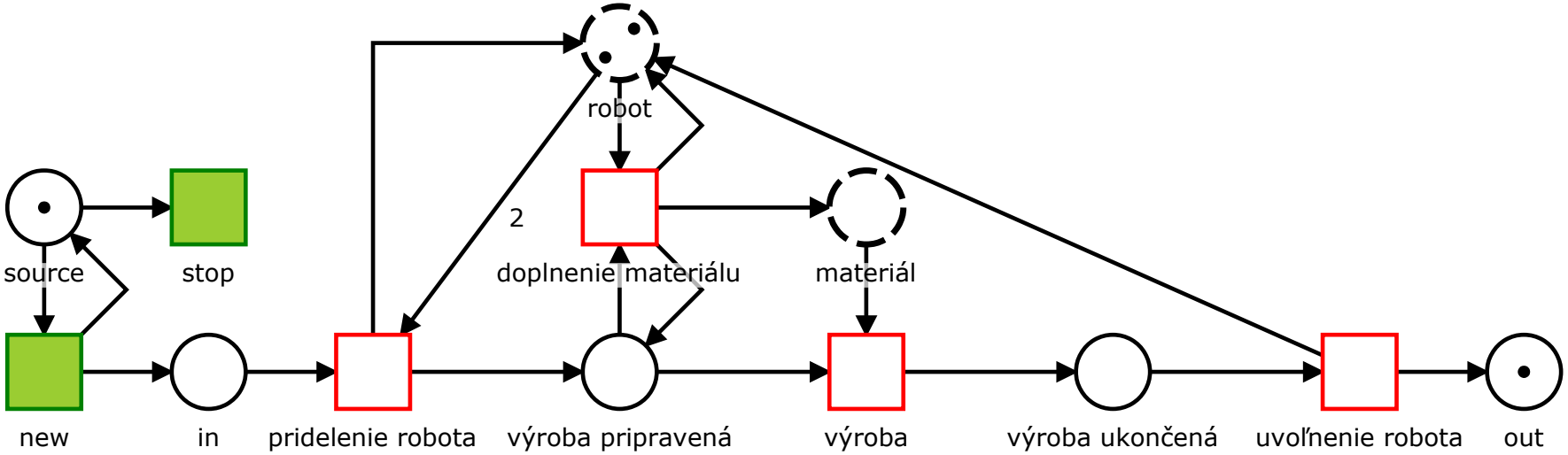
Pridelenie robota na výrobu

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



Paralelne pridelenie druhého robota na výrobu spôsobí uviaznutie

Budúci výskum: Detekcia uviaznutí pre netrvácne zdroje – statické miesta bez komplementárnych určujúcich miest
Komplikovanejšie procesy, kde výstupy jedného procesu sú vstupom iného procesu



Upravená sieť bez uviaznutí pre ľubovoľné značkovanie
s minimálne dvomi robotmi
Upravená sieť zachováva pôvodné správanie pre jednu inštanciu

Zhrnutie

1) Definovali sme siete dosiahnuteľnosti.

2) Skonštruovali sme algoritmus, ktorý overí, či vstupná workflow sieť je korektná a ak áno, vytvorí sieť dosiahnuteľnosti.

3) Definovali sme taktiež uviaznutia v sieti dosiahnuteľnosti.

Zhrnutie

1) Definovali sme siete dosiahnuteľnosti.

2) Skonštruovali sme algoritmus, ktorý overí, či vstupná workflow sieť je korektná a ak áno, vytvorí sieť dosiahnuteľnosti.

3) Definovali sme taktiež uviaznutia v sieti dosiahnuteľnosti.

4) Dokázali sme, že značkovania vykonávanej siete zodpovedajú značkovaniam siete dosiahnuteľnosti.

5) Dokázali sme, že značkovanie vykonávanej siete je uviaznutím vykonávanej siete práve vtedy, keď jemu zodpovedajúce značkovanie siete dosiahnuteľnosti je uviaznutím v sieti dosiahnuteľnosti.

Zhrnutie

1) Definovali sme siete dosiahnuteľnosti.

2) Skonštruovali sme algoritmus, ktorý overí, či vstupná workflow sieť je korektná a ak áno, vytvorí sieť dosiahnuteľnosti.

3) Definovali sme taktiež uviaznutia v sieti dosiahnuteľnosti.

4) Dokázali sme, že značkovania vykonávanej siete zodpovedajú značkovaniam siete dosiahnuteľnosti.

5) Dokázali sme, že značkovanie vykonávanej siete je uviaznutím vykonávanej siete práve vtedy, keď jemu zodpovedajúce značkovanie siete dosiahnuteľnosti je uviaznutím v sieti dosiahnuteľnosti.

6) Definovali sme základné uviaznutia siete dosiahnuteľnosti.

7) Dokázali sme, že ak má sieť dosiahnuteľnosti uviaznutie, potom má sieť dosiahnuteľnosti zodpovedajúce základné uviaznutie.

Zhrnutie

1) Definovali sme siete dosiahnuteľnosti.

2) Skonštruovali sme algoritmus, ktorý overí, či vstupná workflow sieť je korektná a ak áno, vytvorí sieť dosiahnuteľnosti.

3) Definovali sme taktiež uviaznutia v sieti dosiahnuteľnosti.

4) Dokázali sme, že značkovania vykonávanej siete zodpovedajú značkovaniam siete dosiahnuteľnosti.

5) Dokázali sme, že značkovanie vykonávanej siete je uviaznutím vykonávanej siete práve vtedy, keď jemu zodpovedajúce značkovanie siete dosiahnuteľnosti je uviaznutím v sieti dosiahnuteľnosti.

6) Definovali sme základné uviaznutia siete dosiahnuteľnosti.

7) Dokázali sme, že ak má sieť dosiahnuteľnosti uviaznutie, potom má sieť dosiahnuteľnosti zodpovedajúce základné uviaznutie.

8) Dokázali sme, že kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný.

9) Ukázali sme dva spôsoby, ako vypočítať horné ohraničenie kritických miest.

10) Definovali sme ohraničenú obmedzenú sieť dosiahnuteľnosti.

Zhrnutie

1) Definovali sme siete dosiahnuteľnosti.

2) Skonštruovali sme algoritmus, ktorý overí, či vstupná workflow sieť je korektná a ak áno, vytvorí sieť dosiahnuteľnosti.

3) Definovali sme taktiež uviaznutia v sieti dosiahnuteľnosti.

4) Dokázali sme, že značkovania vykonávanej siete zodpovedajú značkovaniam siete dosiahnuteľnosti.

5) Dokázali sme, že značkovanie vykonávanej siete je uviaznutím vykonávanej siete práve vtedy, keď jemu zodpovedajúce značkovanie siete dosiahnuteľnosti je uviaznutím v sieti dosiahnuteľnosti.

6) Definovali sme základné uviaznutia siete dosiahnuteľnosti.

7) Dokázali sme, že ak má sieť dosiahnuteľnosti uviaznutie, potom má sieť dosiahnuteľnosti zodpovedajúce základné uviaznutie.

8) Dokázali sme, že kritické miesta sú ohraničené a teda počet základných uviaznutí je konečný.

9) Ukázali sme dva spôsoby, ako vypočítať horné ohraničenie kritických miest.

10) Definovali sme ohraničenú obmedzenú sieť dosiahnuteľnosti.

11) 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.

12) Zostavili sme finálny algoritmus na detekciu základných uviaznutí siete dosiahnuteľnosti.

Budúci výskum – zhrnutie:

1. Súčasný algoritmus pre vopred daný počet jednotlivých zdrojov určí, či workflow sieť má uviaznutia. Problémom pre budúci výskum je v prípade, ak počet zdrojov nie je známy určiť, pre aký počet zdrojov sieť nebude mať uviaznutia a pre aký počet zdrojov bude mať uviaznutia.

Budúci výskum – zhrnutie:

1. Súčasný algoritmus pre vopred daný počet jednotlivých zdrojov určí, či workflow sieť má uviaznutia. Problémom pre budúci výskum je v prípade, ak počet zdrojov nie je známy určiť, pre aký počet zdrojov sieť nebude mať uviaznutia a pre aký počet zdrojov bude mať uviaznutia.

2. Rozšírenie algoritmu na detekciu uviaznutí tak, aby keď algoritmus odhalí uviaznutie, teda keď sieť má uviaznutia, doplnil (opravil) workflow sieť takým spôsobom, aby opravená doplnená sieť uviaznutia nemala (zmenou počtu zdrojov, pridaním hrán, pridaním nových statických miest atď.).

Budúci výskum – zhrnutie:

1. Súčasný algoritmus pre vopred daný počet jednotlivých zdrojov určí, či workflow sieť má uviaznutia. Problémom pre budúci výskum je v prípade, ak počet zdrojov nie je známy určiť, pre aký počet zdrojov sieť nebude mať uviaznutia a pre aký počet zdrojov bude mať uviaznutia.

2. Rozšírenie algoritmu na detekciu uviaznutí tak, aby keď algoritmus odhalí uviaznutie, teda keď sieť má uviaznutia, doplnil (opravil) workflow sieť takým spôsobom, aby opravil doplnená sieť uviaznutia nemala (zmenou počtu zdrojov, pridaním hrán, pridaním nových statických miest atď.).

3. Pri takomto opravení môžeme požadovať, aby opravenie zabránilo uviaznutiam, ale zároveň neobmedzilo možnosti výberu poradia prechodov a podobne.

Budúci výskum – zhrnutie:

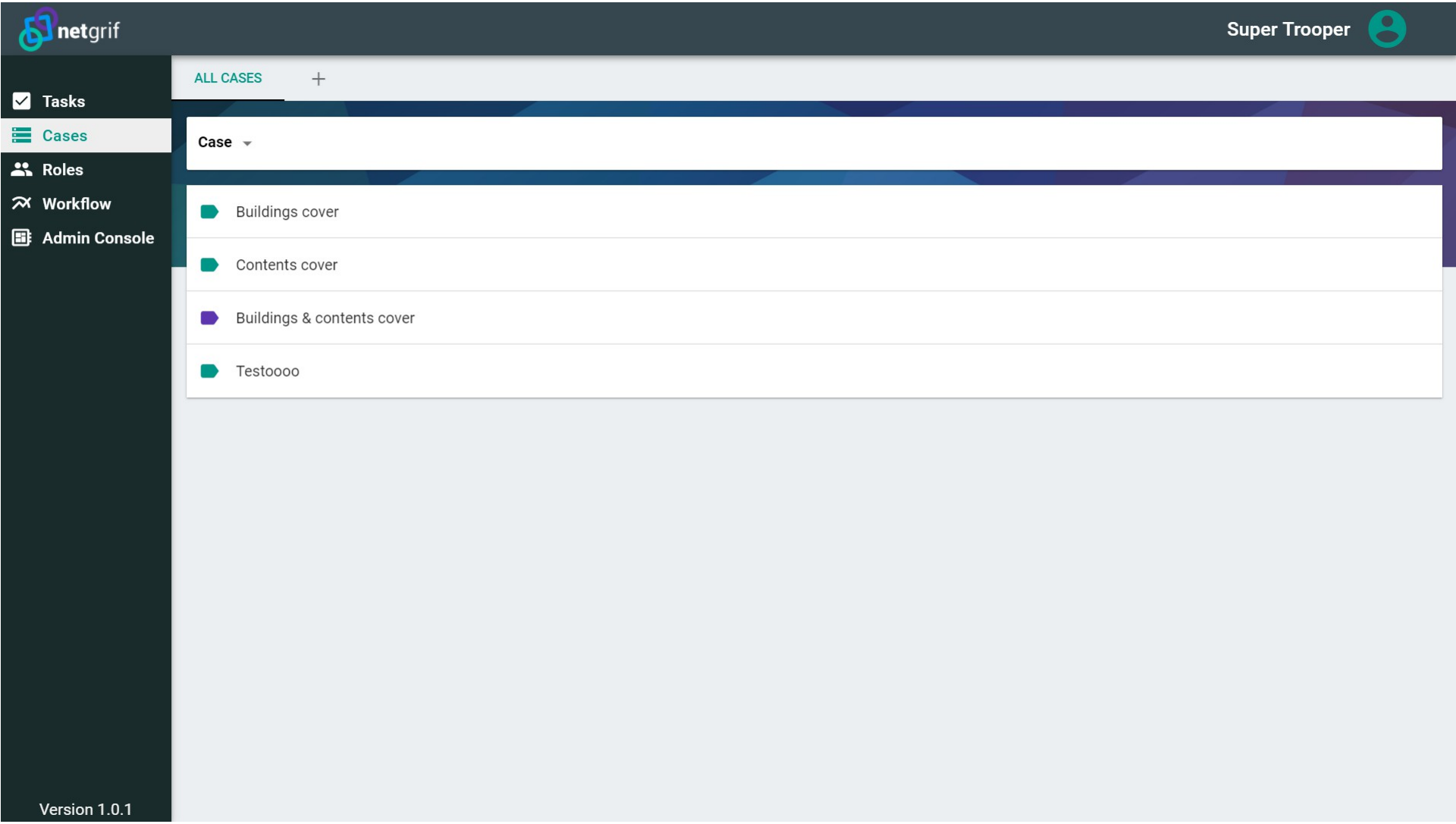
1. Súčasný algoritmus pre vopred daný počet jednotlivých zdrojov určí, či workflow sieť má uviaznutia. Problémom pre budúci výskum je v prípade, ak počet zdrojov nie je známy určiť, pre aký počet zdrojov sieť nebude mať uviaznutia a pre aký počet zdrojov bude mať uviaznutia.

2. Rozšírenie algoritmu na detekciu uviaznutí tak, aby keď algoritmus odhalí uviaznutie, teda keď sieť má uviaznutia, doplnil (opravil) workflow sieť takým spôsobom, aby opravil doplnená sieť uviaznutia nemala (zmenou počtu zdrojov, pridaním hrán, pridaním nových statických miest atď.).

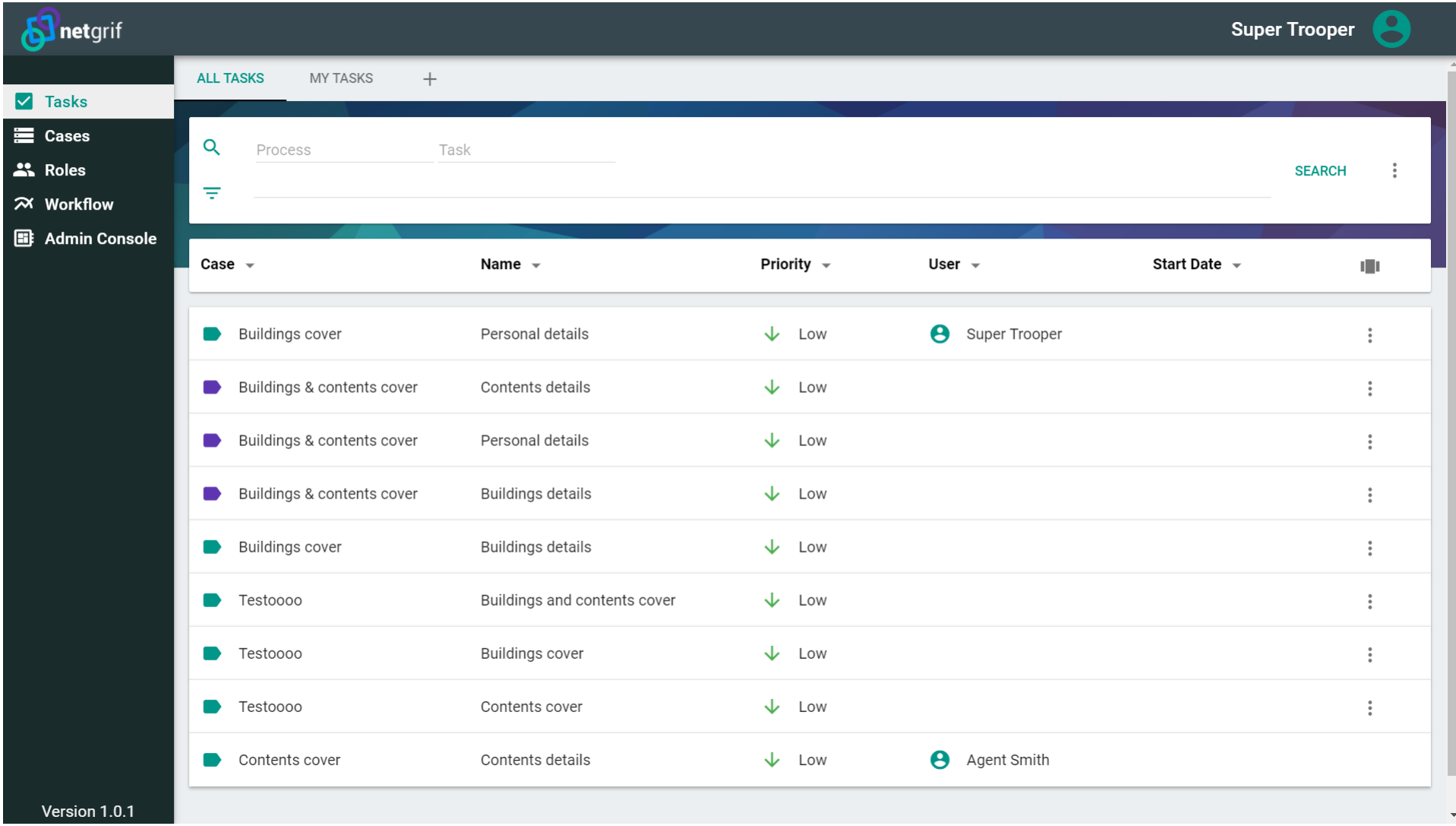
3. Pri takomto opravení môžeme požadovať, aby opravenie zabránilo uviaznutiam, ale zároveň neobmedzilo možnosti výberu poradia prechodov a podobne.

4. Prezentovaný algoritmus platí iba ak zdroje sú trvácne (inštancia si ich požičia a potom vráti – typický priklad je pamäť, procesor, robot, či nástroj). Témou budúceho výskumu je upraviť algoritmus tak, aby našiel uviaznutia aj procesy, v ktorých sú zdroje, ktoré sa spotrebujú a nevrátia a sú produktom iného procesu (kolesá, spotrebný materiál a pod.)

PrintScreen Workflow Enginu NETGRIF WMS, ktorý umožňuje spúšťať viaceré inštancie workflow siete a vykonávať prechody v nich užívateľom podľa priradenia k rolám, teda umožňuje vytvárať vykonávanú runtime sieť.



V rámci Workflow Enginu NETGRIF WMS má užívateľ k dispozícii task manažér, kde vidí aktuálne úlohy (spustiteľné prechody jednotlivých inštancií vykonávanej siete) a môže ich spúšťať (ak je priradený do role, ktorá je priradená prechodu).



Industry 4.0:

V súlade s dokumentom* agentúry Germany Trade & Invest (GTAI) Spolkovej republiky Nemecko, ktorý hovorí, že
„The product is able to control its own production process as it has the all necessary information available in its digital product memory“
za jeden z kľúčových aspektov Industry 4.0 a konceptu Smart Factory považujeme dôraz na jednotlivé inštancie produktu a teda produkčného procesu a individualizáciu produktov.

*https://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf

Industry 4.0:

V súlade s dokumentom* agentúry Germany Trade & Invest (GTAI) Spolkovej republiky Nemecko, ktorý hovorí, že
„The product is able to control its own production process as it has the all necessary information available in its digital product memory“
za jeden z kľúčových aspektov Industry 4.0 a konceptu Smart Factory považujeme dôraz na jednotlivé inštancie produktu a teda produkčného procesu a individualizáciu produktov.

To znamená orientáciu na workflow procesy s inštanciami, pričom inštancia je kľučová. Inštancie zdieľajú zdroje (nástroje, robotické stanoviská, klasická výrobná linka sa mení - produkty-inštancie samé putujú po Smart Factory a aktívne požadujú služby robotických stanovísk, ktoré sú práve voľné a ktoré inštancia potrebuje).

V takomto chápaní je potrebná práve detekcia uviaznutí spôsobených zdieľaním zdrojov (robotov a pod.) jednotlivými inštanciami, ako je načrtnuté v mojej práci

*https://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf