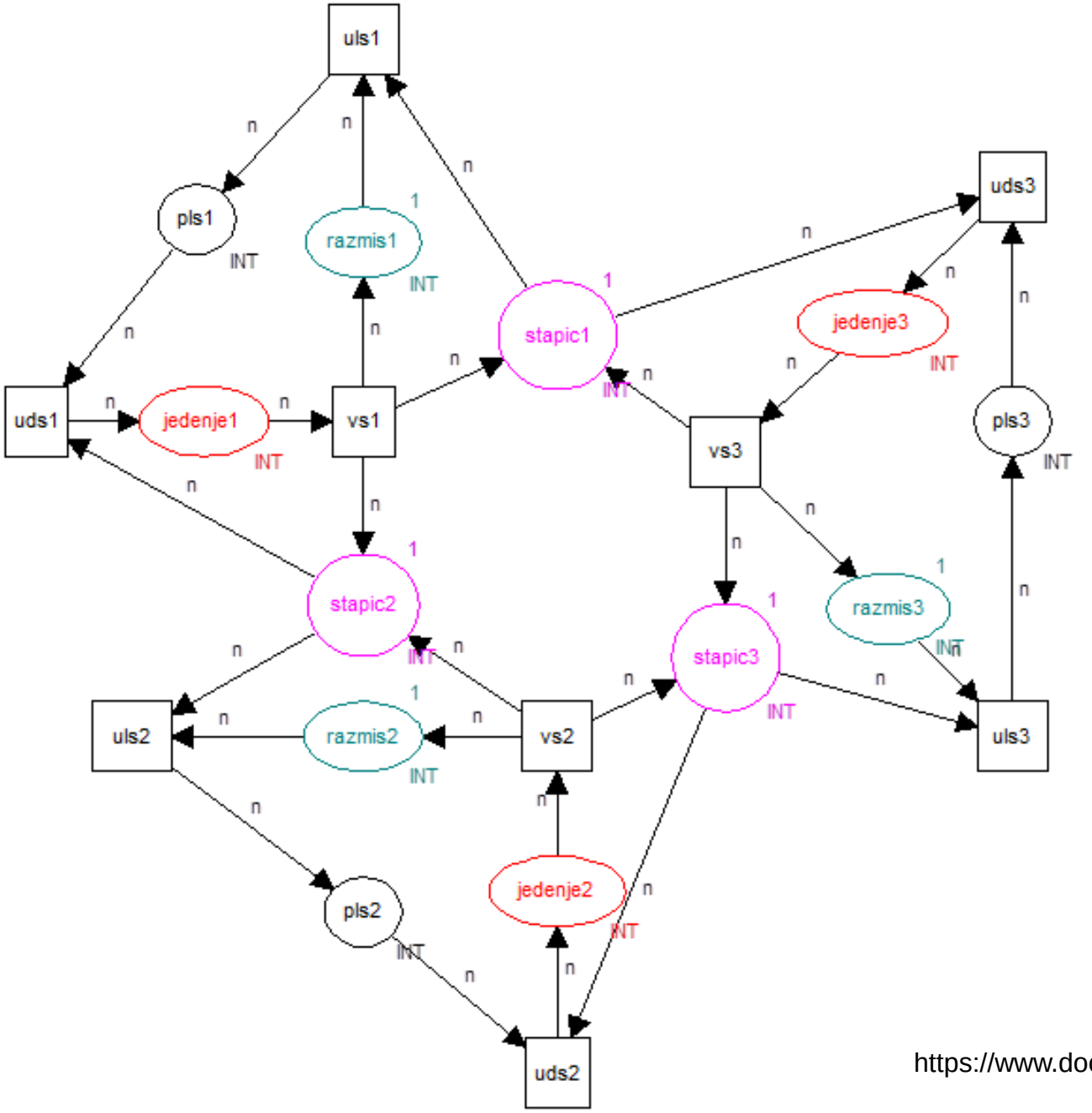


Анализа процеса и Петријеве мреже

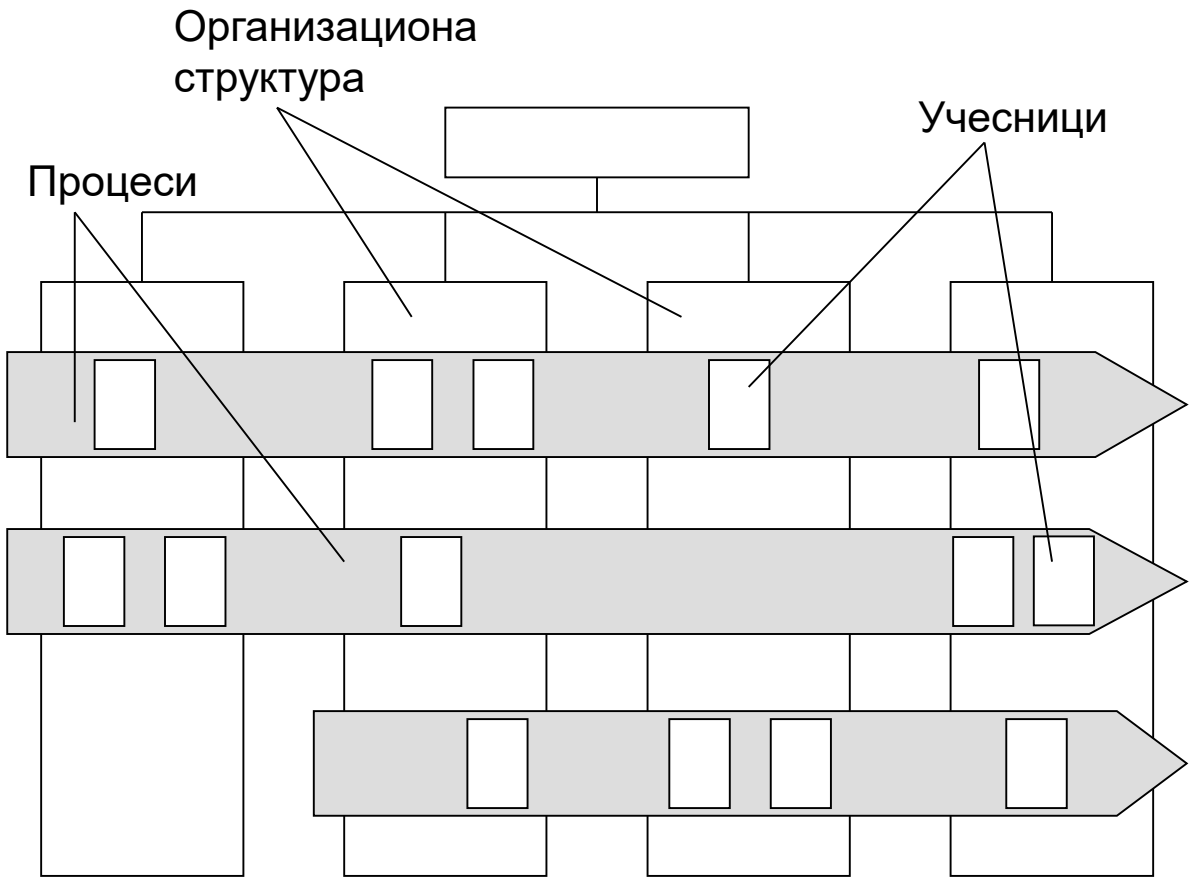
https://www.docsity.com/wordpress/wp-content/uploads/sites/2/2014/03/dinning-table.gif

Процес

- *Davenport* (1990): “… специфичан низ активности у времену и простору, који има свој почетак и крај и прецизно дефинисане улазе и излазе.”
- *Harrington* (1991): “… свака активност или група активности које узимају улаз, повећавају му вредност и тиме производе излаз за унутрашњег или спољашњег корисника.”
- *Hammer* и *Champy* (1993): “… колекција активности која има више улаза и ствара више излаза који имају неку вредност за корисника.”
- Standard ISO 8402 (1996) “… скуп међусобно повезаних ресурса и активности који претварају улазне елементе у излазне.“
- Standard ISO 9000 (2000) “… скуп међусобно повезаних или међусобно делујућих активности који претвара улазне у излазне елементе.“
- Вероватно најкраћа дефиниција процеса је

“… трансформација улазних величина (*input*) у излазне величине (*output*) .“

Пословни процеси



MIT Process Handbook Project

Сврха: развијање методологије и софтверског алата за представљање пословних процеса на различитим нивоима апстракције и сакупљању, организовању и анализирању примера о томе како различите организације изводе сличне пословне процесе

Основни концепти:

Декомпозиција - специјализација процеса.

Управљање зависностима.

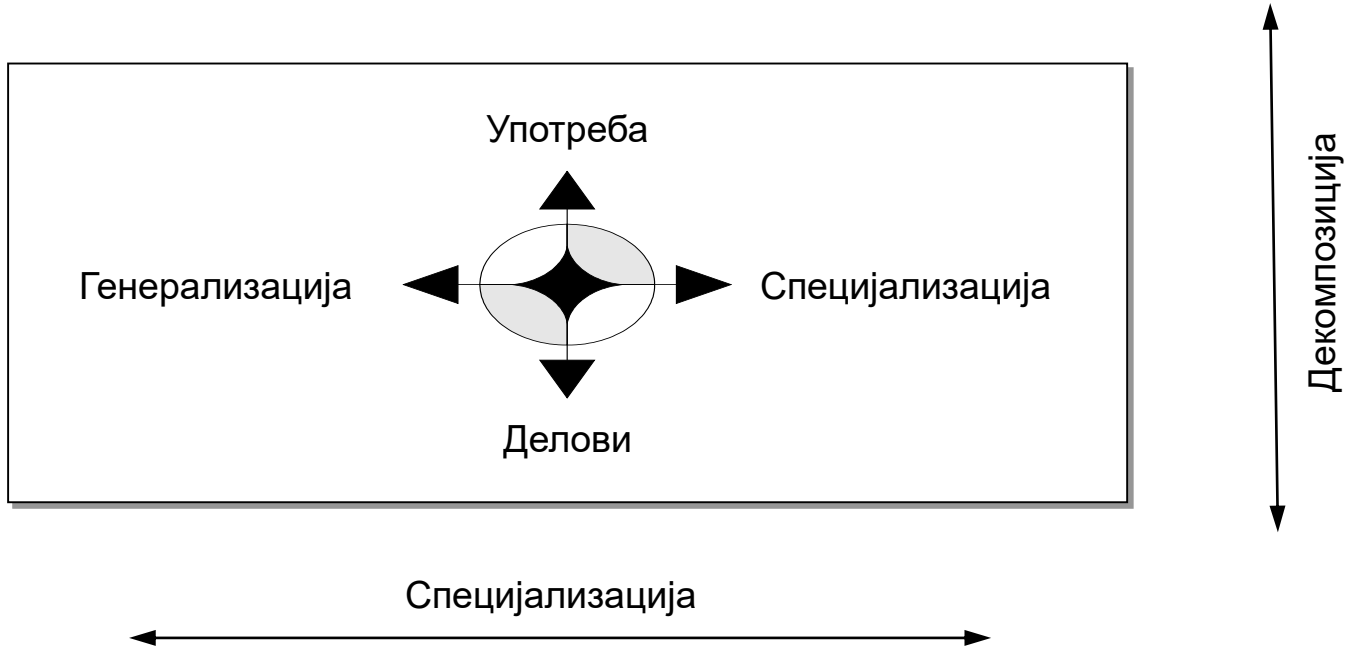
(Center for Coordination Science,
Massachusetts Institute of Technology, Cambridge, Massachusetts)

Декомпозиција и специјализација

- Две димензије у опису пословних процеса:
  - Различити *делови* (декомпозиција) пословних процеса Декомпозиција процеса - раздвајање процеса на кораке (активности) који треба да се изврше да би се процес комплетирао, односно остварио његов циљ.
  - Различити *типови* (специјализација) пословних процеса Специјализација процеса – нова димензија у опису процеса која анализира процесе по типу (класи) и даје одговоре на питања: како, шта, коме итд. Класе процеса се могу дефинисати помоћу:
    - Минималног скупа понашања система: сва понашања (активности) које сваки процес из посматране класе мора да садржи. Специјализација процеса се врши додавањем активности које посматрани процес разликује од осталих процеса у класи.
    - Максималног скупа понашања система: сва могућа понашања (активности) које процеси из посматране класе могу да садрже. Специјализација процеса се врши уклањањем активности, односно рестрикцијом овог скупа.

Компас процеса

- Који су различити делови овог процеса? - *Parts* (делови)
- У ком већем процесу се користи овај процес? - *Uses* (употреба)
- Који су други процеси попут овог? - *Generalization* (генерализација)
- На које друге начине може да се изврши овај процес? - *Specialization* (специјализација)



http://process.mit.edu/Info/CaseLinks.asp

Управљање зависностима

- Три основне врсте зависности:
  1. Ток (произвођач - корисник)
  2. Дељење улаза
  3. Слагање, дељење излаза

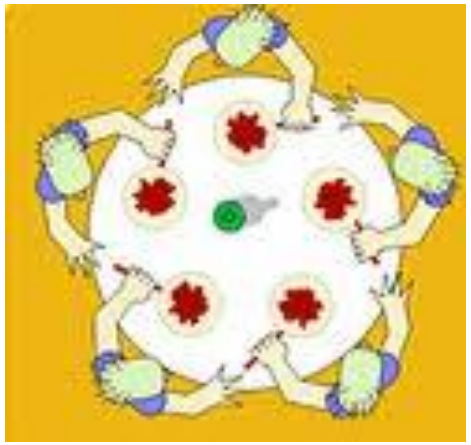
- Кад год постоји зависност између две активности потребна је координација.

- Активности у процесу: активности производње (језгро процеса) и активности координације.
  - Идентификација активности координације:
    - директним тражењем међу свим активностима,
    - тражењем учесника у процесу који координирају или
    - елиминисањем активности производње из листе свих активности. Активности које остају су или активности координације или активности које треба избацити из процеса.
  - Често су активности које користе више информација од осталих (информационо интензивне), управо активности координације.

Deadlock

- Блокада, застој, ћорсокак, мртво стање
- Скуп процеса је у блокади ако сваки процес чека на догађај који може да изазове само неки други процес.

Услови за Deadlock



1. Узајамно искључење (*Mutual exclusion*) – само један процес може да користи ресурс у једном тренутку.
2. Држи и чекај (*Hold and wait*) - процес чува ресурс и чека следећи.
3. Нема права пречег (*No preemption*) – заузети ресурс се не може преотети. Процес добровољно ослобађа ресурс након извршења.
4. Кружно чекање (*Circular wait*) – мора постојати ланац од најмање два процеса где сваки чека ресурс заузет суседним процесом.

Управљање блокадом

- Превенција – осигурати да се *deadlock* не деси тако што се обезбеди да се бар један од 4 услова не појави
- Избегавање deadlock - осигурати да се *deadlock* не деси коришћењем информација о захтевима ресурсима и динамичким избегавањем несигурних ситуација
- Детекција и опоравак – дозволити deadlock, али регистровати када се појави, опоравити процес и наставити
- Игнорисање – најлакши и најчешћи приступ.

Технике за моделирање процеса: захтеви

- Моделирање активности, ресурса и веза између њих.
- Процесна оријентација.
  - Моделирање у хијерархији или у сегментима и приказивање различитих варијанти (типова) процеса.
  - Моделирање зависности и механизама координације.
- Моделирање кашњења (стохастичког).
- Лако прилагођавање и отвореност.
- Алат.
- Разумљива и лака за употребу.
- Могућност анализе процеса.

ПЕТРИЈЕВЕ МРЕЖЕ (Petri nets- PN)

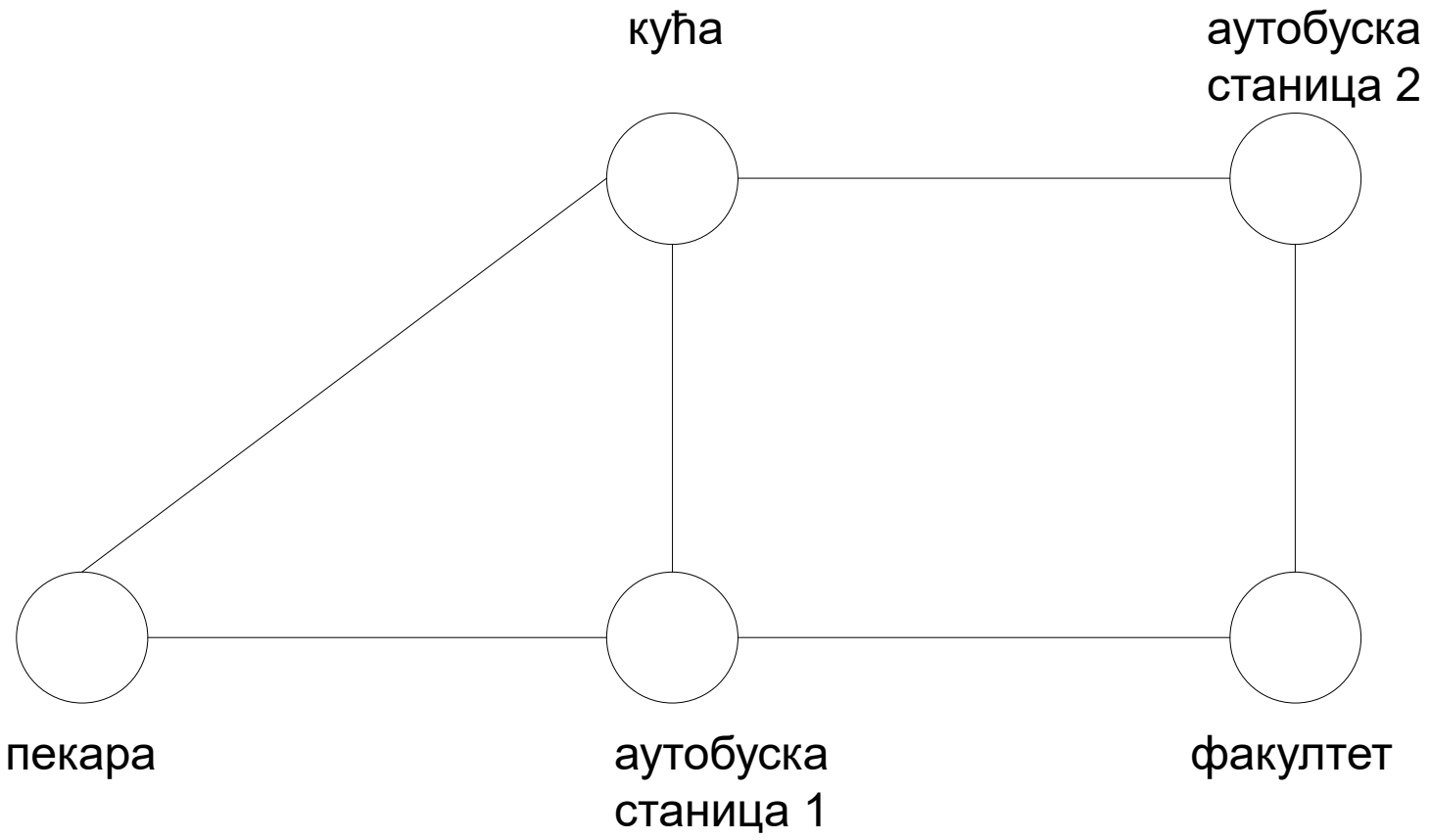
- **Петријеве мреже** су графички и математички алат за моделирање и анализу процеса и њихове динамике.

- Концепт PN је увео Carl Adam Petri 1962. године у Докторској дисертацији под називом *"Kommunikation mit Automaten"*, на Факултету за математику и физику *Uniadt* у Немачкој.

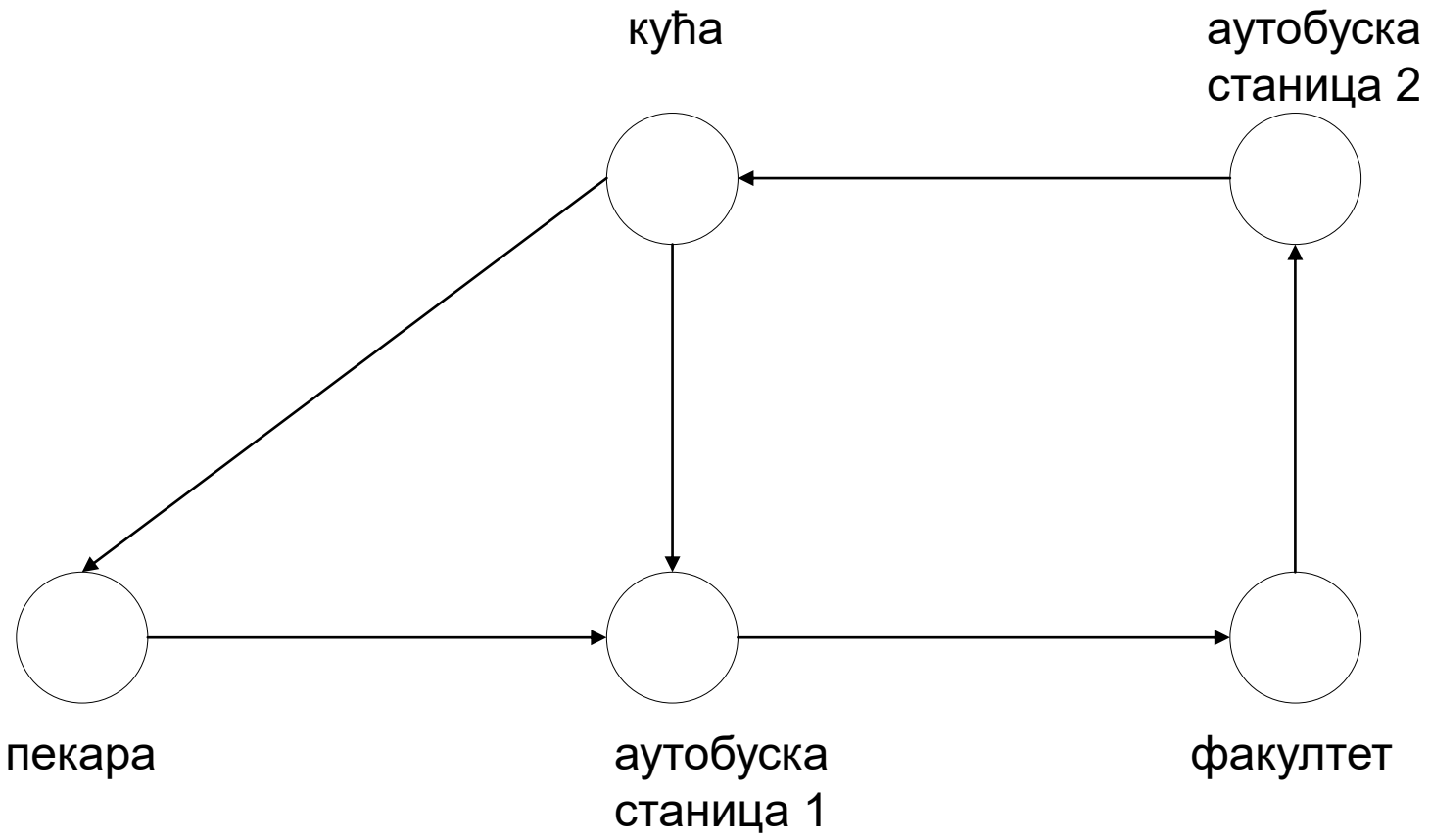
ПЕТРИЈЕВЕ МРЕЖЕ (Petri nets- PN)

- Преко 250 књига посвећених Петријевим мрежама и више од 2000 књига у којима су Петријеве мреже обрађене у оквиру поглавља или наслова.
- Преко 8500 радова посвећених Петријевим мрежама.
- Сталне конференције:
  - *International Conferences on Application and Theory of Petri Nets and Other Models of Concurrency* (1980.)
  - *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools* (1998.),
  - *The International Conference on Quantitative Evaluation of SysTems (QEST)* у оквиру кога се од 2004. године одржава *the International Workshop on Petri Nets and Performance Models* (PNPM)

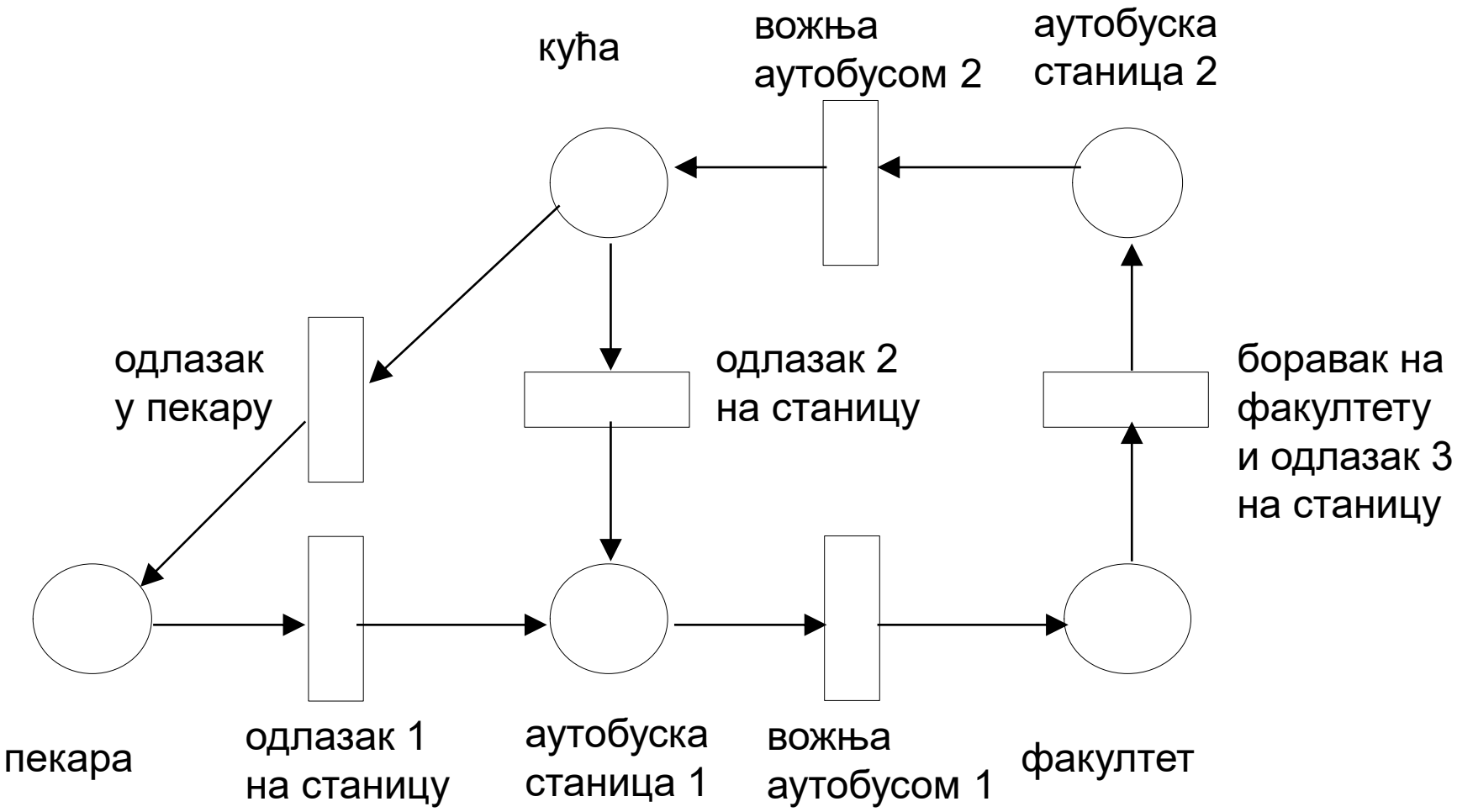
Граф (X,U)


кућа
аутобуска
станица 2
пекара
аутобуска
станица 1
факултет

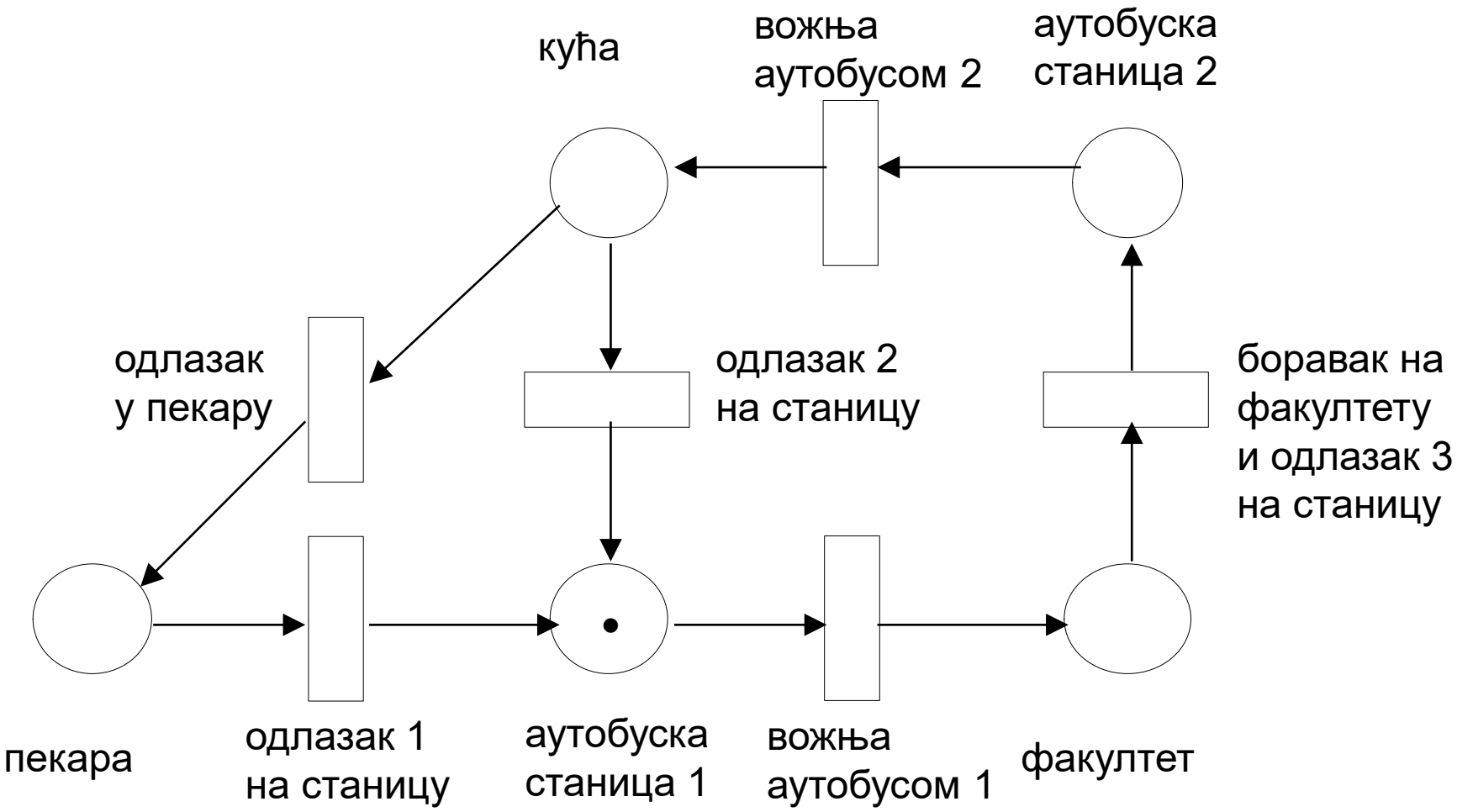
Дограф (X,U)


кућа
аутобуска
станица 2
пекара
аутобуска
станица 1
факултет

Бипартитни граф (P,T,U)



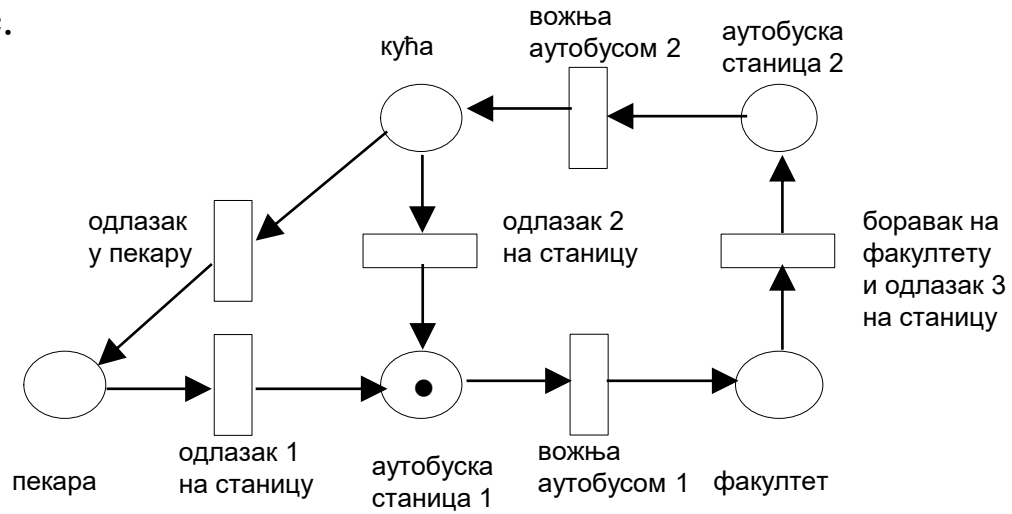
Петријева мрежа – Petri Net (PN)



Формална дефиниција PN

Елементарна (црно-бела) PN је петорка PN = $(P, T, A, W, M_0) = (N, M_0)$
где је:

$P = \{p_1, p_2, ..., p_n\}$	коначан скуп места (чворови графа)
$T = \{t_1, t_2, ..., t_q\}$	коначан скуп прелаза (чворови графа)
$A \subseteq (P \times T) \cup (T \times P)$	коначан скуп грана, при чему је $(P \cap T) = \varnothing$ и $(P \cup T) \neq \varnothing$
$W: A \rightarrow \{1,2,\ldots\}$	функција тежина придружена гранама,
$M_0: P \rightarrow \{0, 1,2,\ldots\}$	почетно маркирање.



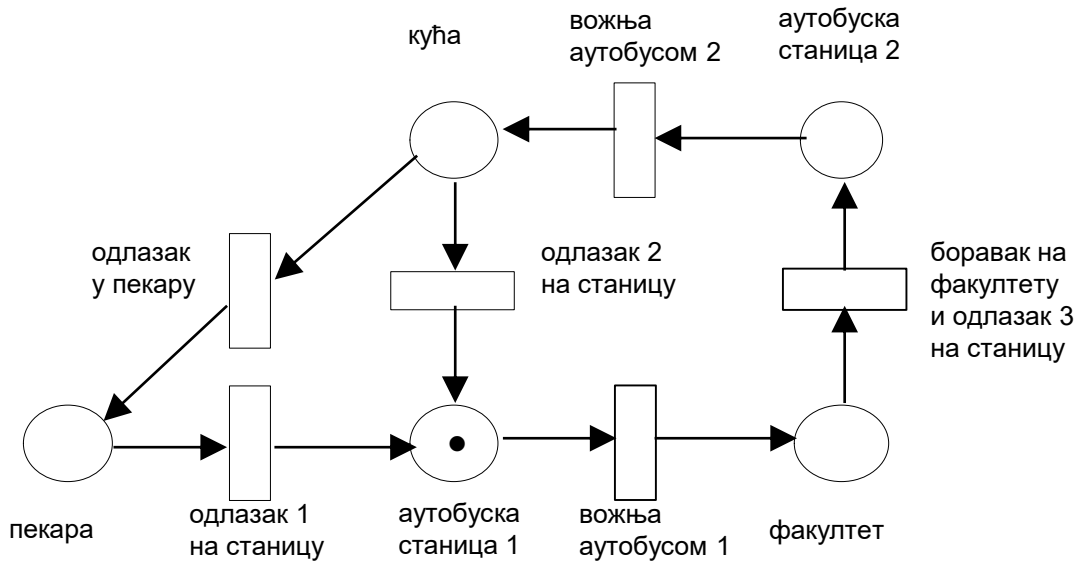
Моделирање динамике помоћу PN

^{o}t – скуп свих улазних места прелаза t, тј. $^{o}t = \{p \mid p \in P, (p,t) \in A\}$

t^{o} – скуп свих излазних места прелаза t, тј. $t^{o} = \{p \mid p \in P, (t, p) \in A\}$

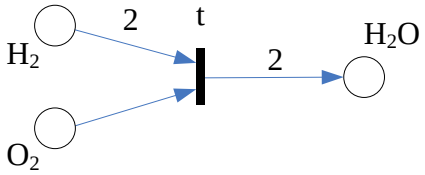
^{o}p – скуп свих улазних прелаза места p, тј. $^{o}p = \{t \mid t \in T, (t, p) \in A\}$

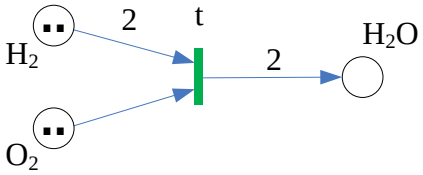
p^{o} – скуп свих излазних прелаза места p, тј. $p^{o} = \{t \mid t \in T, (p,t) \in A\}$

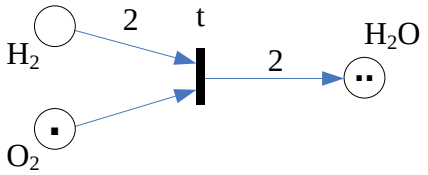


$2H_2 + O_2 \rightarrow 2H_2O$

$2H_2, 2O_2$







$PN = (P, T, A, W, M_0)$

$P = \{H_2, O_2, H_2O\}$

$T = \{t\}$

$A = \{(H_2, t), (O_2, t), (t, H_2O)\}$

$W = [2, 1, 2]$

$M_0 = [0, 0, 0]$

$\forall p \in {}^{o}t, M(p) \geq W(p,t)$

$M_0 = [2, 2, 0]$

${}^{o}t = \{H_2, O_2\}$

$M(H_2)=2,\ M(O_2)=2$

$W(H_2, t)=2,\ W(O_2, t)=1$

$$\Rightarrow \begin{matrix} M(H_2) \geq W(H_2, t) \\ M(O_2) \geq W(O_2, t) \end{matrix}$$

$\Rightarrow$ prelaz t je omogućen

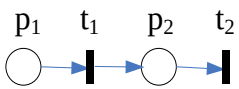
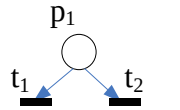
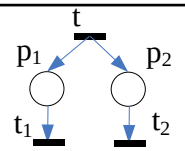
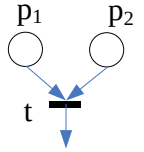
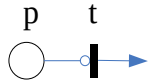
$$M'(p) = \begin{cases} M(p)\text{-W}(p,t) & \text{ako} \quad p \in {}^{0}t \\ M(p) + \text{W}(p,t) & \text{ako} \quad p \in t^{0} \\ M(p) & \text{inače} \end{cases}$$

$M'(H_2)=M(H_2)-W(H_2, t)=2-2=0$

$M'(O_2)=M(O_2)-W(O_2, t)=2-1=1$

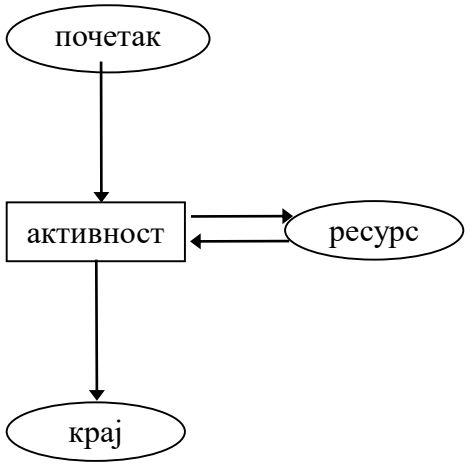
$M'(H_2O)=M(H_2O)+W(t, H_2O)=0+2=2$

Стандардни модели зависности

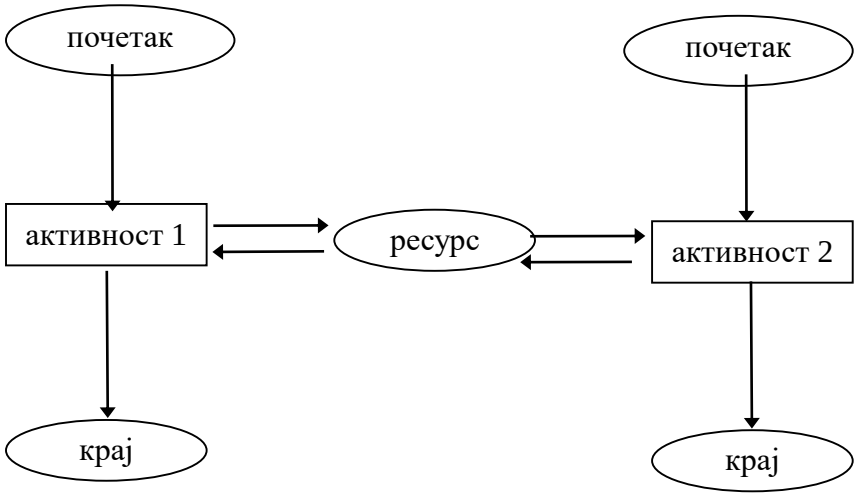
Примитив	Услов	Приказ
Секвенца	$t_1 \in p_1^0, t_1 \in {}^0p_2, t_2 \in p_2^0$ $M(p_1) \neq \varnothing$	
Конфликт	$t_1, t_2 \in p^0$ $M(p) \neq \varnothing$	
Истовременост (concurrency)	$p_1, p_2 \in t^0, t_1 \in p_1^0, t_2 \in p_2^0$	
Синхронизација	$p_1, p_2 \in {}^0t$ $M(p_1) \neq \varnothing, M(p_2) \neq \varnothing$	
Инхибитор	$p \in {}^0t,\ M(p) = \varnothing$	

Стандардни модели зависности

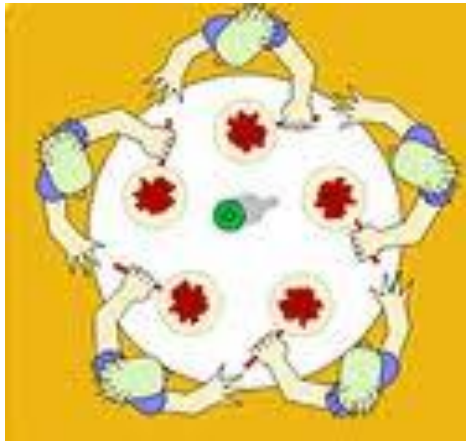
коришћење ресурса

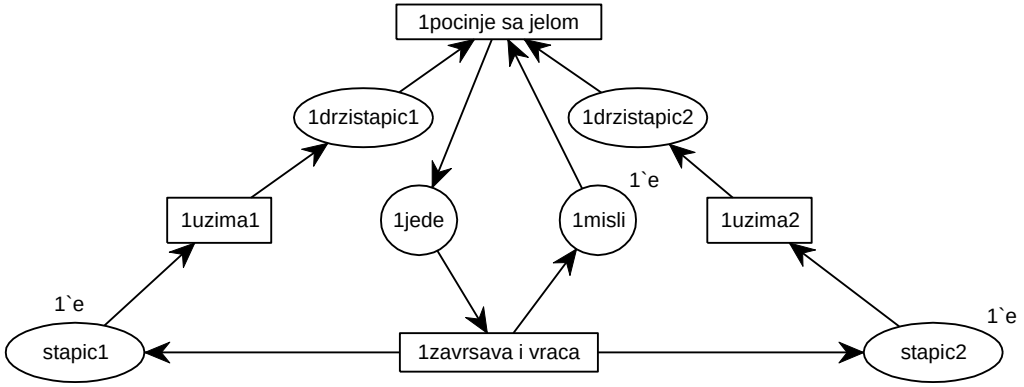


дељење ресурса

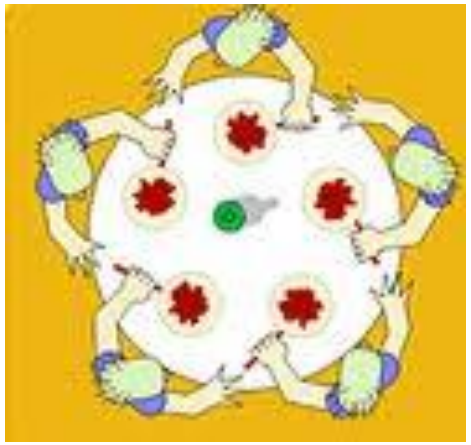


Услови за Deadlock



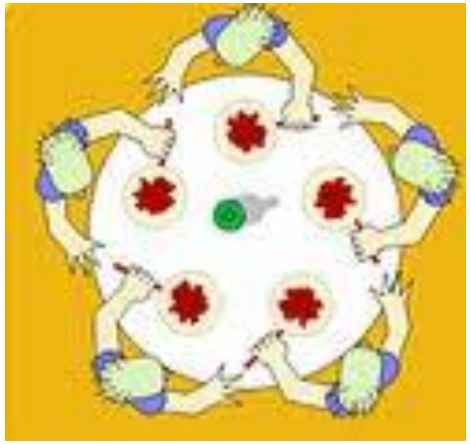


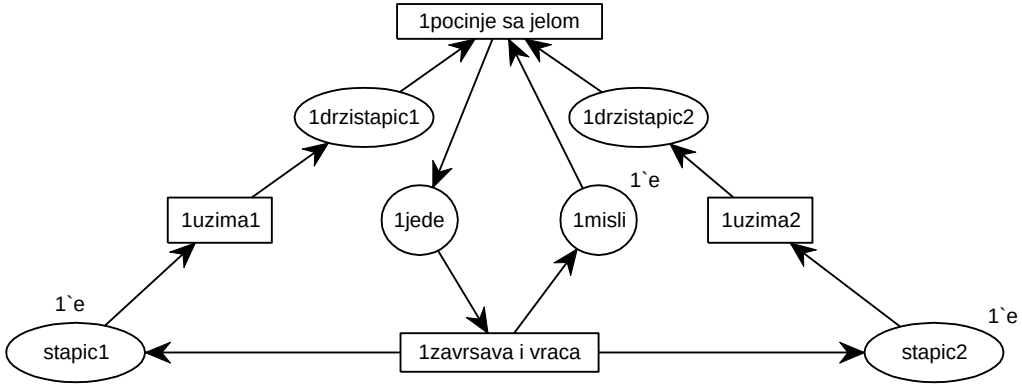
Услови за Deadlock

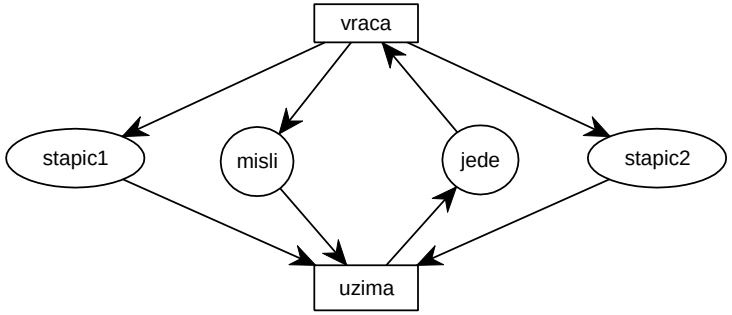


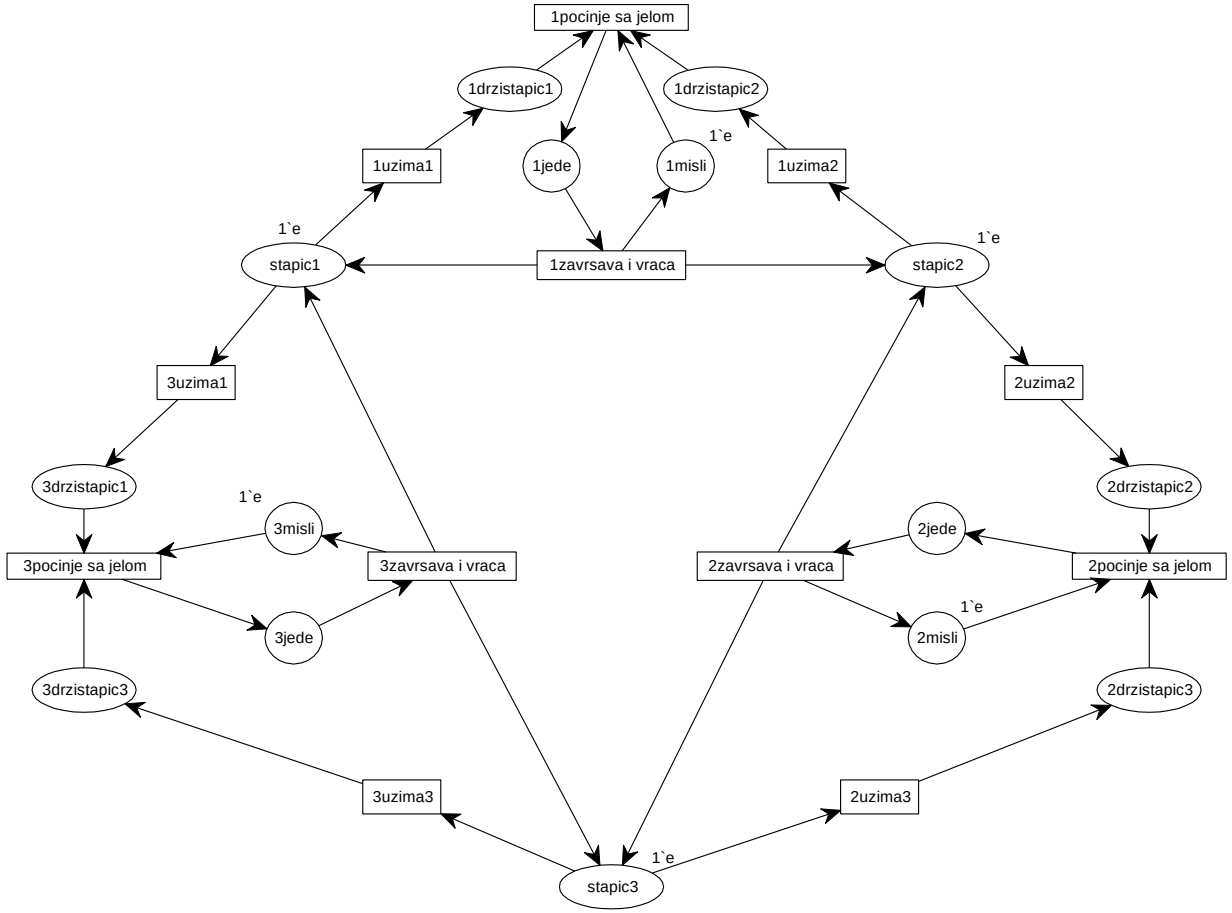
1. Узајамно искључење (*Mutual exclusion*) – само један процес може да користи ресурс у једном тренутку.
2. Држи и чекај (*Hold and wait*) - процес чува ресурс и чека следећи.
3. Нема права пречег (*No preemption*) – заузети ресурс се не може преотети. Процес добровољно ослобађа ресурс након извршења.
4. Кружно чекање (*Circular wait*) – мора постојати ланац од најмање два процеса где сваки чека ресурс заузет суседним процесом.

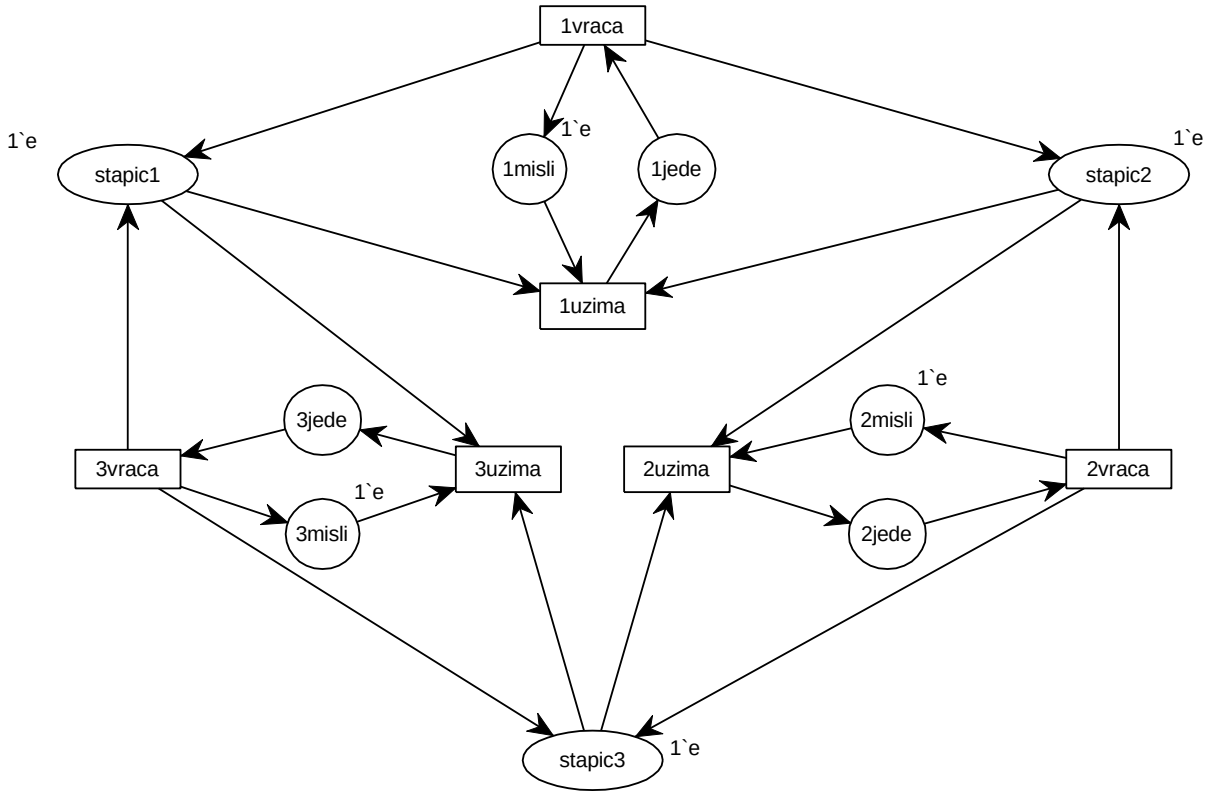
Услови за Deadlock








1pocinje sa jelom
1drzistapic1
1drzistapic2
1uzima1
1jede
1misli
1uzima2
1`e
1`e
1`e
stapic1
1zavrsava i vraca
stapic2
3uzima1
2uzima2
3drzistapic1
2drzistapic2
1`e
3misli
2jede
3pocinje sa jelom
3zavrsava i vraca
2zavrsava i vraca
2pocinje sa jelom
1`e
3jede
2misli
3drzistapic3
2drzistapic3
3uzima3
2uzima3
1`e
stapic3


1vraca
1`e
1misli
1jede
1`e
1`e
stapic1
stapic2
1uzima
1`e
3jede
2misli
3vraca
3uzima
2uzima
2vraca
1`e
3misli
2jede
stapic3
1`e

Анализа Петријевих мрежа

Инваријанте

Једначина стања

Стабло досежљивости

Симулација

- Класе PN вишег нивоа.
- Мерење перформанси процеса.

Технике за моделирање процеса: захтеви

- Моделирање активности, ресурса и веза између њих.
- Процесна оријентација.
  - Моделирање у хијерархији или у сегментима и приказивање различитих варијанти (типова) процеса.
  - Моделирање зависности и механизама координације.
- Моделирање кашњења (стохастичког).
- Лако прилагођавање и отвореност.
- Алат.
- Разумљива и лака за употребу.
- Могућност анализе процеса.

Обојене Петријеве мреже

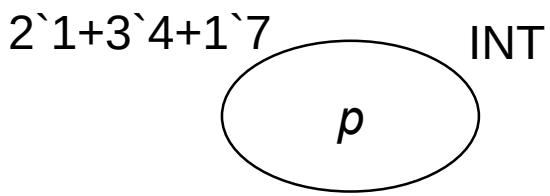
Обојене Петријеве мреже (ОПМ)

- *Colored Petri Nets* (CPN)
- Курт Јенсен (*Kurt Jensen*) 1981. године
- Сваком чвору типа место придружен је тип података који одређује врсту података које дато место може да садржи.
- Сваки жетон у месту носи вредност података која припада типу тог места.
- Вредност података се назива и боја жетона а тип података – скуп боја.

Место ОПМ

- Мултискуп $M=(S, m)$, над непразним скупом S, где се функција $m:S\rightarrow \boldsymbol{N}$ се представља као формална сума $\sum_{s\in S} m(s)`s$
- $m(s)$ представља број појављивања елемента $s\in S$ у мултскупу M.
- Маркирање у обојеним ПМ може да се представи као функција која сваком месту p придружује мултискуп жетона одговарајућег типа.

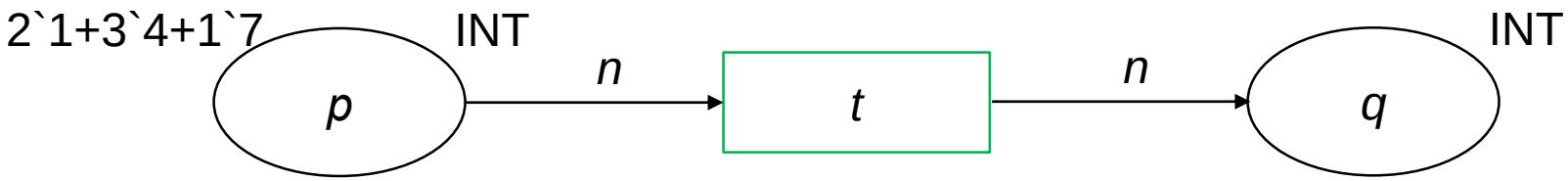
Маркирање места p које је типа цео број је: 1, 1, 4, 4, 4 и 7.



Грана ОПМ

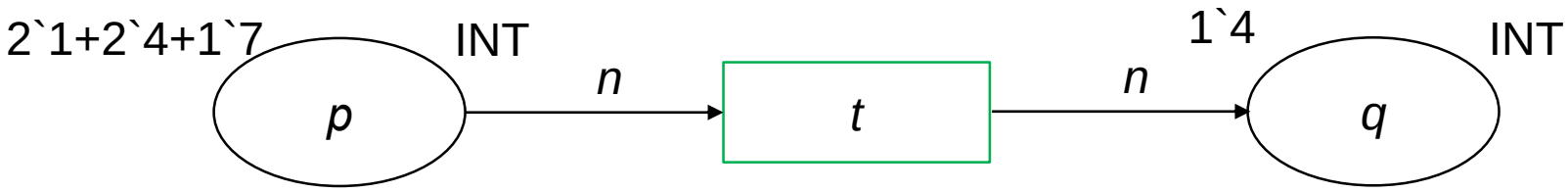
- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се паљењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.

n – променљива типа *INT*



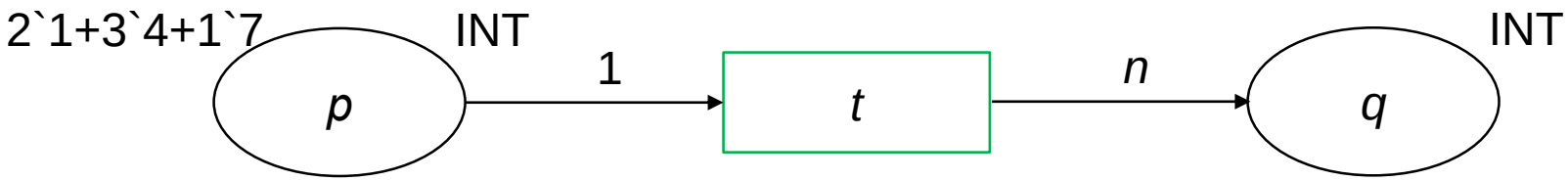
Грана ОПМ

- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се пражњењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.



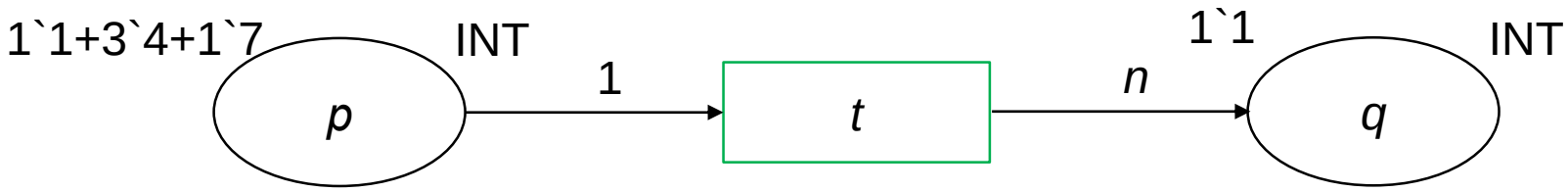
Грана ОПМ

- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се пражњењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.



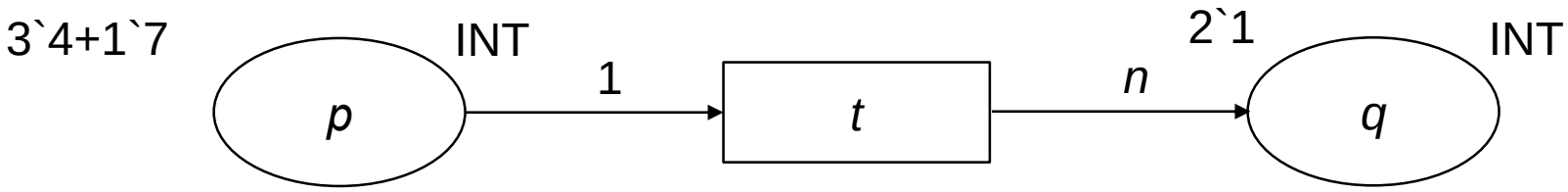
Грана ОПМ

- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се паљењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.



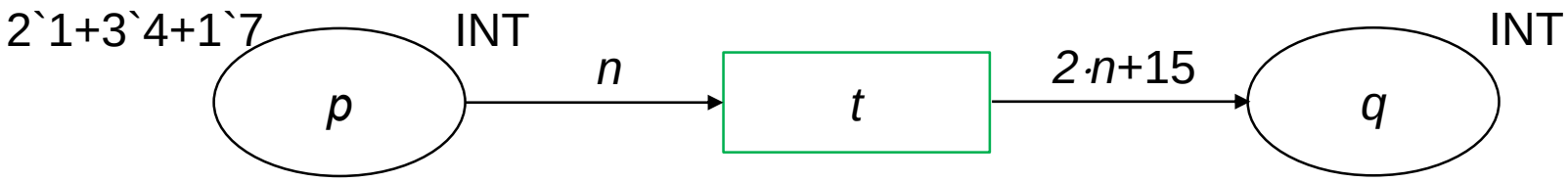
Грана ОПМ

- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се пражњењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.



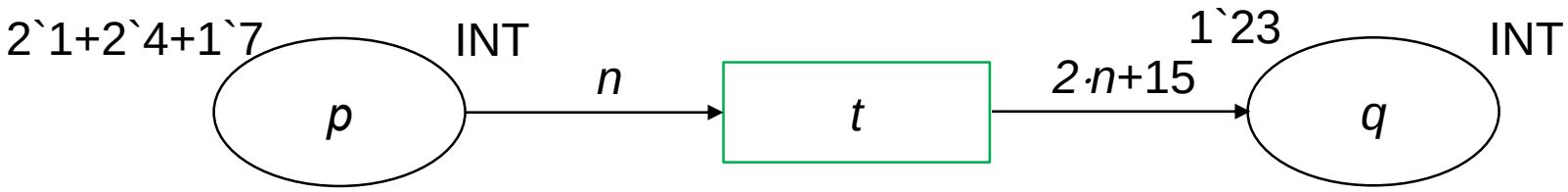
Грана ОПМ

- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се пражњењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.



Грана ОПМ

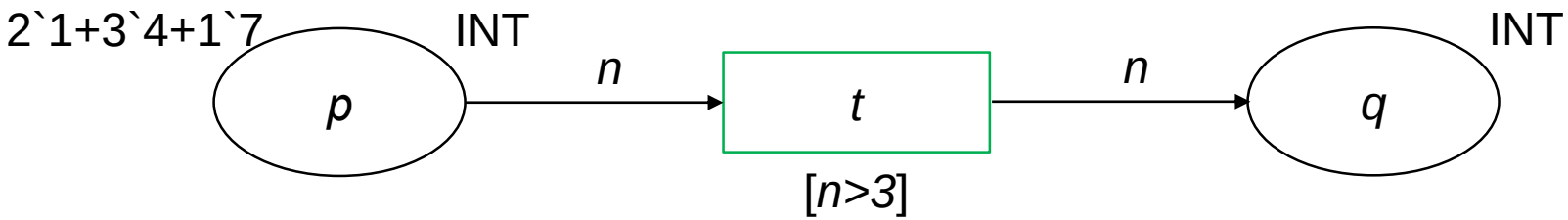
- Тежина гране - опис гране (*arc expression*).
- Опис гране одређује тачан број и врсту жетона која се паљењем прелаза t ослобађа из $p \in {}^{o}t$ и производи у $p \in t^{o}$.
- Опис гране мора бити истог типа као место са којим је повезана.
- Опис гране може бити променљива, константа или функција.



Прелаз ОПМ

- Паљење прелаза – *transition occurrence*.
- Прелазу се може доделити *guard* функција која одређује додатни услов паљења прелаза.
- Резултат *guard* функције је буловска променљива ({*true*, *false*}) а аргументи морају бити истог типа (боје) као места са којим је прелаз повезан.

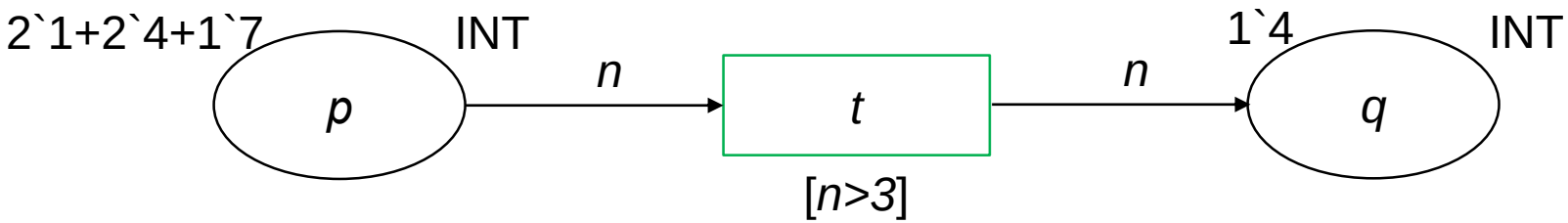
$$\forall t \in T : [\mathrm{Type}(G(t)) = \mathbf{B} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \Sigma]$$



Прелаз ОПМ

- Паљење прелаза – *transition occurrence*.
- Прелазу се може доделити *guard* функција која одређује додатни услов паљења прелаза.
- Резултат *guard* функције је буловска променљива ({*true*, *false*}) а аргументи морају бити истог типа (боје) као места са којим је прелаз повезан.

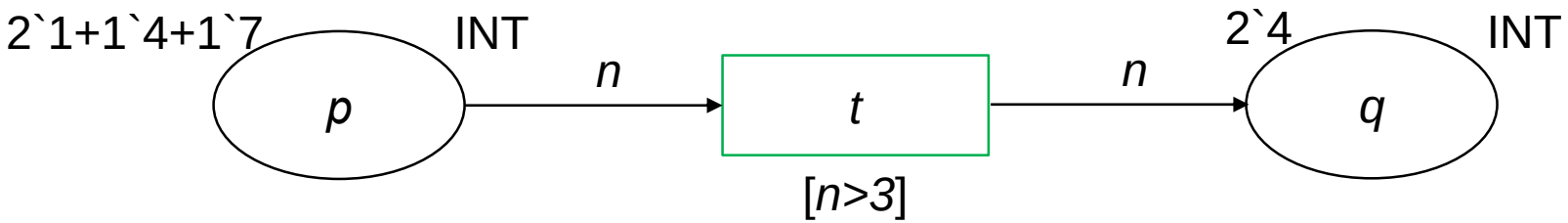
$$\forall t \in T : [\mathrm{Type}(G(t)) = \mathbf{B} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \Sigma]$$



Прелаз ОПМ

- Паљење прелаза – *transition occurrence*.
- Прелазу се може доделити *guard* функција која одређује додатни услов паљења прелаза.
- Резултат *guard* функције је буловска променљива ({*true*, *false*}) а аргументи морају бити истог типа (боје) као места са којим је прелаз повезан.

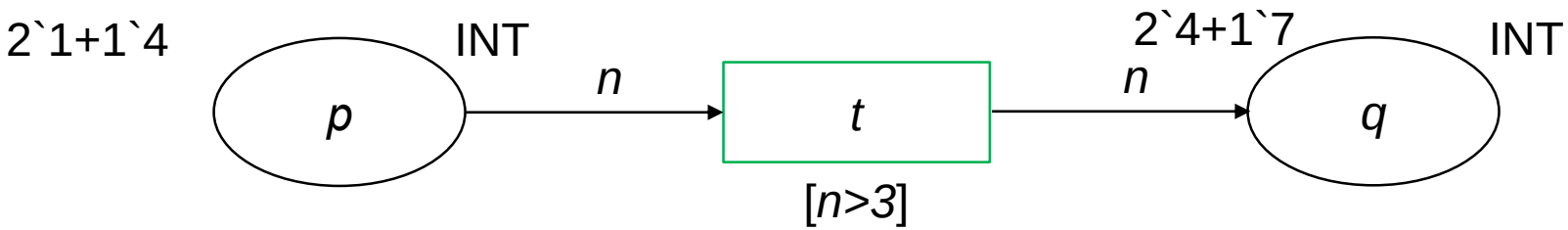
$$\forall t \in T : [\mathrm{Type}(G(t)) = \mathbf{B} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \Sigma]$$



Прелаз ОПМ

- Паљење прелаза – *transition occurrence*.
- Прелазу се може доделити *guard* функција која одређује додатни услов паљења прелаза.
- Резултат *guard* функције је буловска променљива ({*true*, *false*}) а аргументи морају бити истог типа (боје) као места са којим је прелаз повезан.

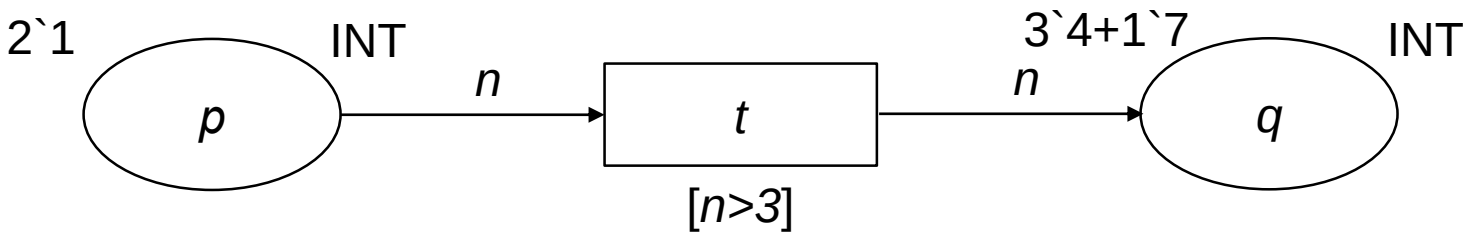
$$\forall t \in T : [\mathrm{Type}(G(t)) = \mathbf{B} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \Sigma]$$



Прелаз ОПМ

- Паљење прелаза – *transition occurrence*.
- Прелазу се може доделити *guard* функција која одређује додатни услов паљења прелаза.
- Резултат *guard* функције је буловска променљива ({*true*, *false*}) а аргументи морају бити истог типа (боје) као места са којим је прелаз повезан.

$$\forall t \in T : [\mathrm{Type}(G(t)) = \mathbf{B} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \Sigma]$$



Формална дефиниција CPN

Обојена PN је деветорка $CPN = (\Sigma, P, T, A, N, C, G, E, M_0)$ где је:

Σ	коначан скуп типова (боја)
$P = \{p_1, p_2, ..., p_n\}$	коначан скуп места
$T = \{t_1, t_2, ..., t_q\}$	коначан скуп прелаза
A	коначан скуп грана такав да $P \cap T = P \cap A = T \cap A = \varnothing$
$N: A \to (P \times T) \cup (T \times P)$	функција чворова
$C: P \to \Sigma$	функција типова (боја)

G функција чувања, која пресликава *T* у израз такав да:

$$t \in T : [Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma]$$

E опис гране, који пресликава *A* у израз такав да:

$$\forall\, a \in A : [Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$$

M_0 почетна функција, која пресликава *P* у затворени израз такав да:

$$\forall\, p \in P : [Type(M_0(p) = C(p)_{MS}]$$

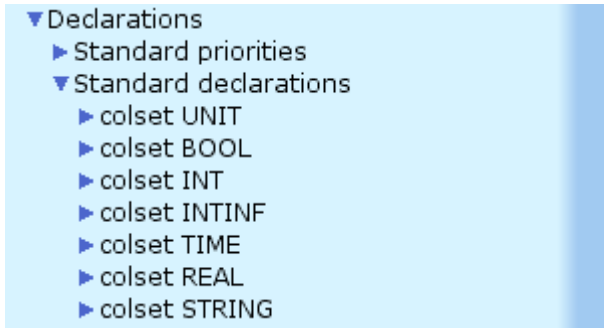
Обојена Петријевих мрежа

Обојена Петријева мрежа којом се моделира конкретан процес се састоји из 3 дела:

- Структура мреже, односно оријентисан, бипартитни граф дефинисан са (ii -v);
- Опис мреже (*declaration*);
- Ознаке на мрежи: типови места, описи грана, услови паљења прелаза и почетно маркирање на мрежи.

CPN Tools

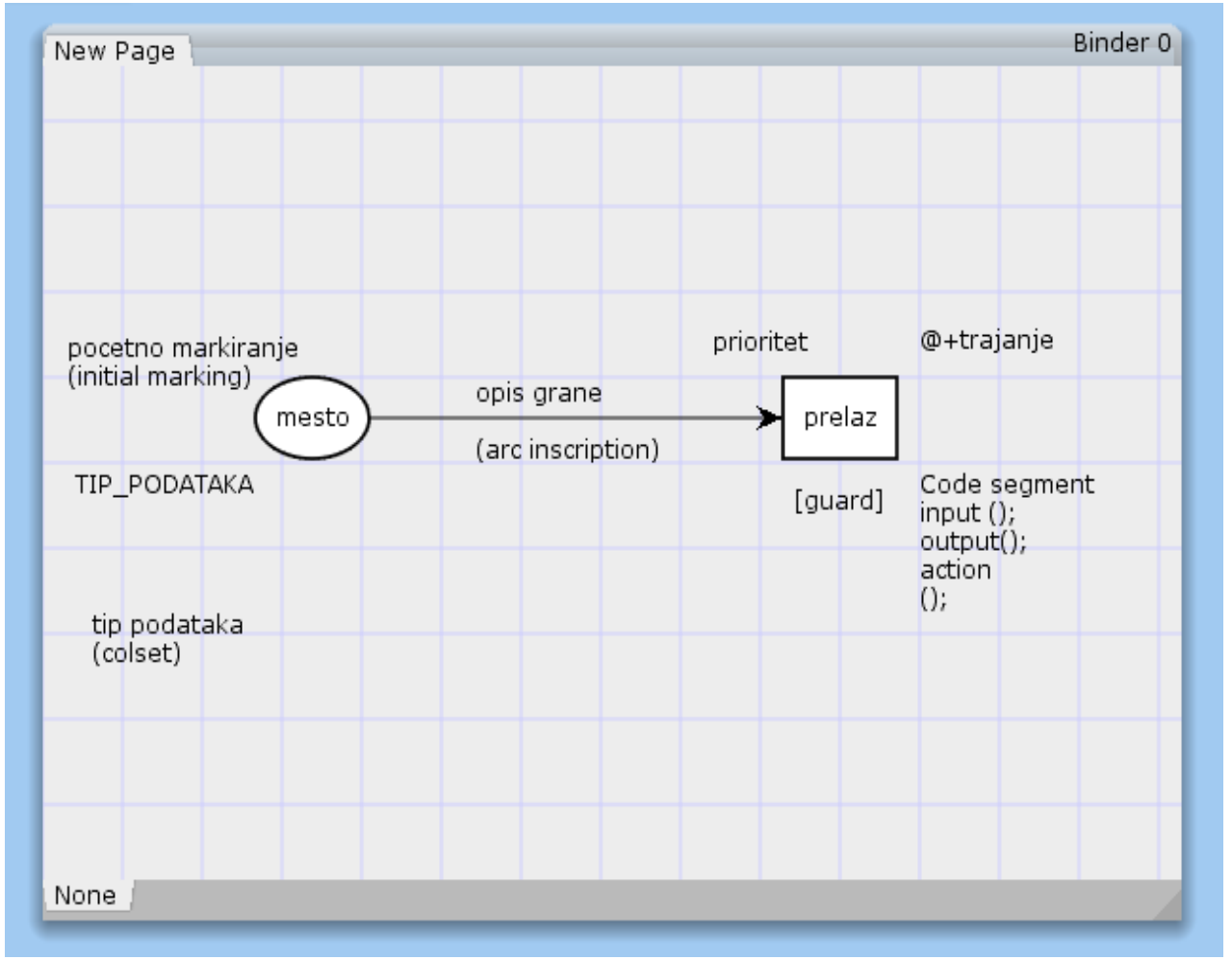
- Опис мреже (*declaration*);



- Декларишу се:
  - Типови података (colset),
  - Функције (fun),
  - Променљиве (var),
  - Константе - параметри (val).

CPN Tools

- Ознаке на мрежи (*inscription*): типови места, описи грана, услови паљења прелаза и почетно маркирање на мрежи.



CPN Tools - Основни типови података *(Simple color sets)*

colset UNIT = unit; неструктуирани тип података, вредност токена = ()

синтакса: colset name = unit [with new_unit];

пример: colset E = unit with e;

опис гране: *e* или ()

colset BOOL = bool; буловски тип података, вредност токена ∈{true,false}

синтакса: colset name = bool [with (new_false, new_true)];

пример: colset Uslov= bool with (da, ne);

var a: Uslov;

опис гране: a

CPN Tools - Основни типови података *(Simple color sets)*

colset INT = int; целобројни тип података, вредност токена $\in \mathbf{Z}$

синтакса: colset name = int [with int-exp1..int-exp2];

пример: colset Broj =int;

colset Broj =int with 1..5;

var *n*: INT;

var *n*: Broj;

опис гране: *n* или функција од *n*

colset INTINF = intinf; целобројни тип података, вредност токена $\in \mathbf{Z}$ и $\leq \infty$

синтакса: colset name = intinf [with int-exp1..int-exp2];

пример: colset Velikibroj =intinf;

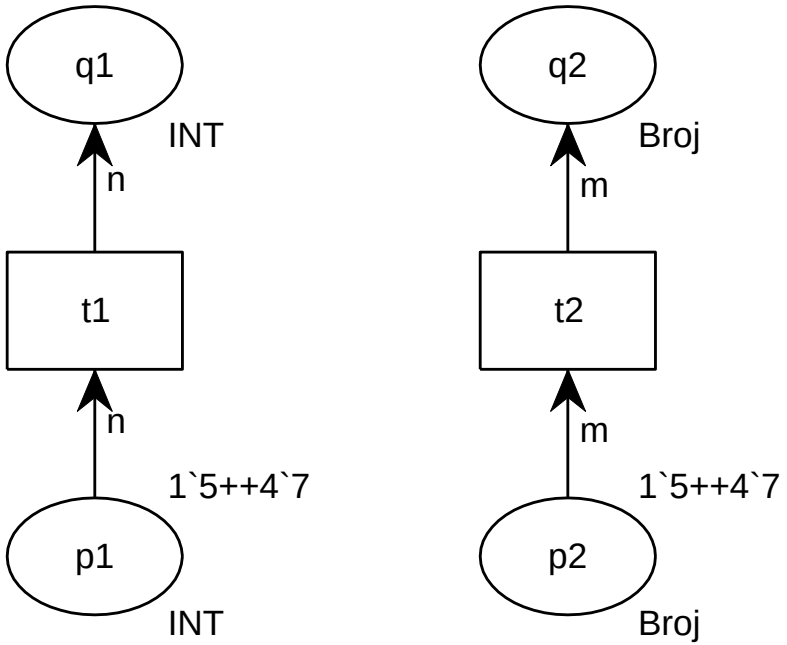
colset Velikibroj =intinf with 1..500000;

var *n*: INTINF;

var *n*: Velikibroj;

опис гране: *n* или функција од *n*

```
(* Standard declarations *)
    colset INT = int;
    colset Broj = int;
    var n:INT;
    var m:Broj;
```



Тип података Broj је тзв. *Alias color sets* – тип података који има исте вредности и својства као неки претходно дефинисани тип података. Сврха његовог постојања је прецизније описивање неког типа података и лакша чиљивост ПМ.

Пример

Испред мењачнице се налази 10 људи од којих неки хоће да замене динаре за евре (купе €) а неки да замене евре за динаре (продају €).

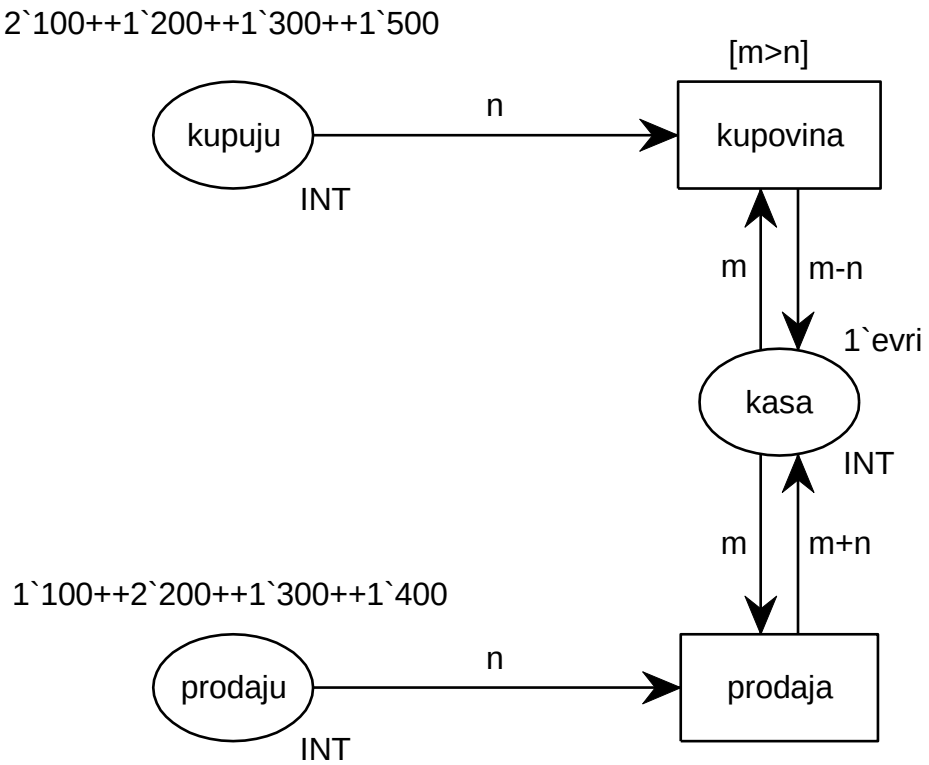
особа	A	B	C	D	E	F	G	H	I	J
купује €	100			500		200	100			300
продаје €		200	300		100			400	200	

У каси мењачнице се тренутно налази 500 € а динара има довољно.

Клијенти се услужују на случајан начин.

Моделирати помоћу ОПМ и симулирати добијену мрежу.

```
(* Standard declarations *)
    colset INT = int;
    var n, m: INT;
    val evri = 500;
```



CPN Tools - Основни типови података *(Simple color sets)*

colset REAL= real; реални тип података, вредност токена $\in$**R**

синтакса: colset name = real [with real-exp1..real-exp2];

пример: colset Broj =real;

colset Broj =real with 1.3..5.1;

var *n*: REAL;

var *n*: Broj;

опис гране: *n* или функција од *n*

colset STRING= string; текстуални тип података

синтакса: colset name = string [with string-exp1..string-exp2 [and int-exp1..int-exp2]];

пример: colset Tekst =string;

colset Tekst =string with “a”..”k”;

colset Tekst =string with “a”..”k” and 5..8;

var *a*: STRING;

var *a*: Tekst;

опис гране: *a*

CPN Tools - Основни типови података *(Simple color sets)*

colset TIME = time; вредност токена зависи од системског времена

синтакса: colset name = time;

пример: colset Vreme =time;

var *t*: TIME;

var *t*: Vreme;

опис гране: time() опис гране – резулат је системско време

Enumeration color set набројиви тип података

синтакса: colset name = with id0 | id1 | … | idn;

пример: colset Dani =with pon | ut | sre | cet | pet | sub | ned;

colset Svrha =with uplata | isplata;

var *a*: Dani;

var *a*: Svrha;

опис гране: *a*

CPN Tools – Сложени типови података (*Compound color sets*)

Product color sets — уређене *n*-торке више типова података

синтакса: colset name = product name1 * name2 * … * name*n*;

пример: colset Klijent = product Svrha * Iznos;

var *k*: Klijent; — вредност токена нпр. (uplata, 100)

var *s*: Svrha;

var *n*: Iznos;

Опис гране: *k* или (*s*,*n*)

Пример

Испред мењачнице се налази 10 људи од којих неки хоће да замене динаре за евре (купе €) а неки да замене евре за динаре (продају €).

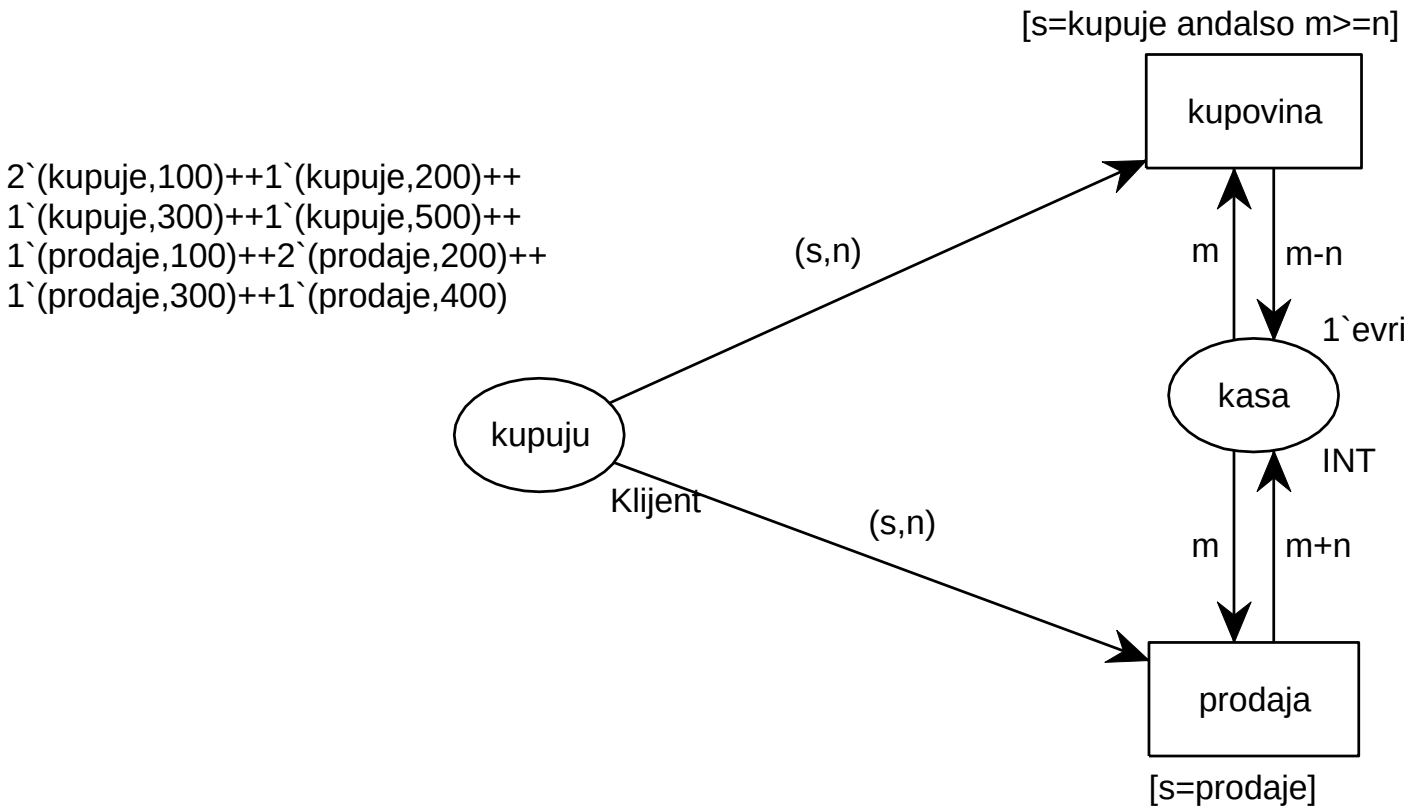
особа	A	B	C	D	E	F	G	H	I	J
купује €	100			500		200	100			300
продаје €		200	300		100			400	200	

У каси мењачнице се тренутно налази 500 € а динара има довољно.

Клијенти се услужују на случајан начин.

Моделирати помоћу ОПМ тако да постоји једно место чије почетно маркирање чине сви клијенти и симулирати добијену мрежу.

```
(* Standard declarations *)
    colset INT = int;
    colset Svrha = with kupuje | prodaje;
    colset Klijent = product Svrha * INT;
    var n, m: INT;
    var s: Svrha;
    val evri = 500;
```



Пример

Испред мењачнице се налази 10 људи од којих неки хоће да замене динаре за евре (купе €) а неки да замене евре за динаре (продају €).

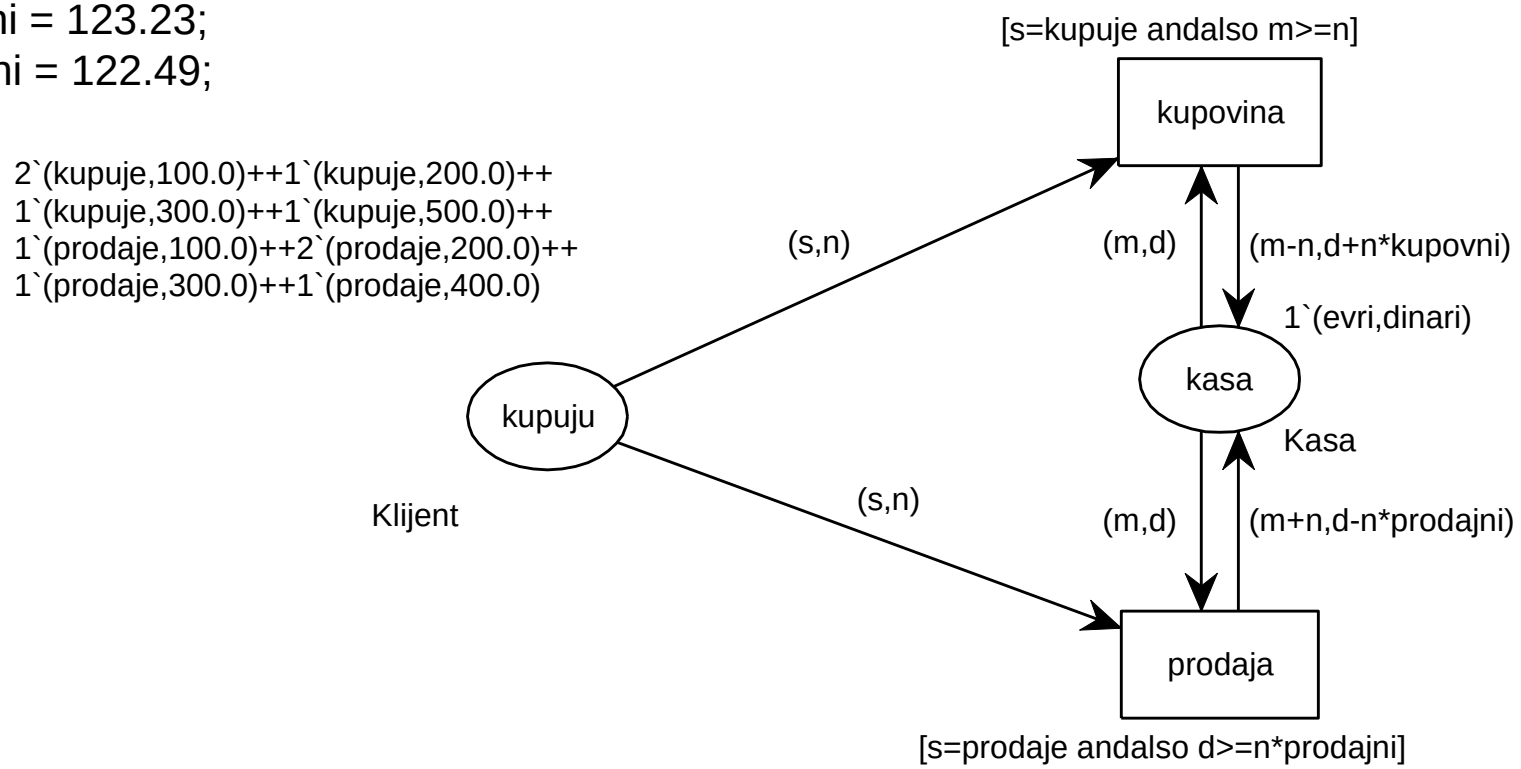
особа	A	B	C	D	E	F	G	H	I	J
купује €	100			500		200	100			300
продаје €		200	300		100			400	200	

У каси мењачнице се тренутно налази 500 € и 70000 динара. Продајни курс износи 122,49 динара а куповни 123,23 за један €.

Клијенти се услужују на случајан начин.

Моделирати помоћу ОПМ тако да постоји једно место чије почетно маркирање чине сви клијенти и симулирати добијену мрежу.

```
(* Standard declarations *)
    colset REAL = real;
    colset DINARI=real;
    colset EVRI=real;
    colset Svrha = with kupuje | prodaje;
    colset Klijent = product Svrha * EVRI;
    colset Kasa = product EVRI * DINARI;
    var n, m: EVRI;
    var s: Svrha;
    var d: DINARI;
    val evri = 500.0;
    val dinari = 500.0;
    val kupovni = 123.23;
    val prodajni = 122.49;
```



CPN Tools – Сложени типови података (*Compound color sets*)

List color sets — листа

синтакса: colset name = list name [with int-exp1..int-exp2];

пример: colset Brojlist = list INT; — неограничена листа целих бројева

colset Brojlist = list INT with 4..7; — листа целих бројева дужине од 4 до 7

colset Red = list Klijent;

var *br*: Brojlist; — вредност токена нпр. [2 1 5 7 9]

var *q*: Red; — [(uplata,100) (isplata,200) (isplata,50)]

празна листа [ ]

почетно маркирање 1`[ ]

Опис гране:

br ^^ [*n*], *q* ^^ [*k*], *q* ^^ (*s*,*n*) — убацивање елемента у листу (var *n*: INT; var *k*: Klijent)

n :: *br*, *k* :: *q*, (*s*,*n*) :: *q* — избацивање елемента из листе

Пример

Испред мењачнице се налази 10 људи од којих неки хоће да замене динаре за евре (купе €) а неки да замене евре за динаре (продају €).

особа	A	B	C	D	E	F	G	H	I	J
купује €	100			500		200	100			300
продаје €		200	300		100			400	200	

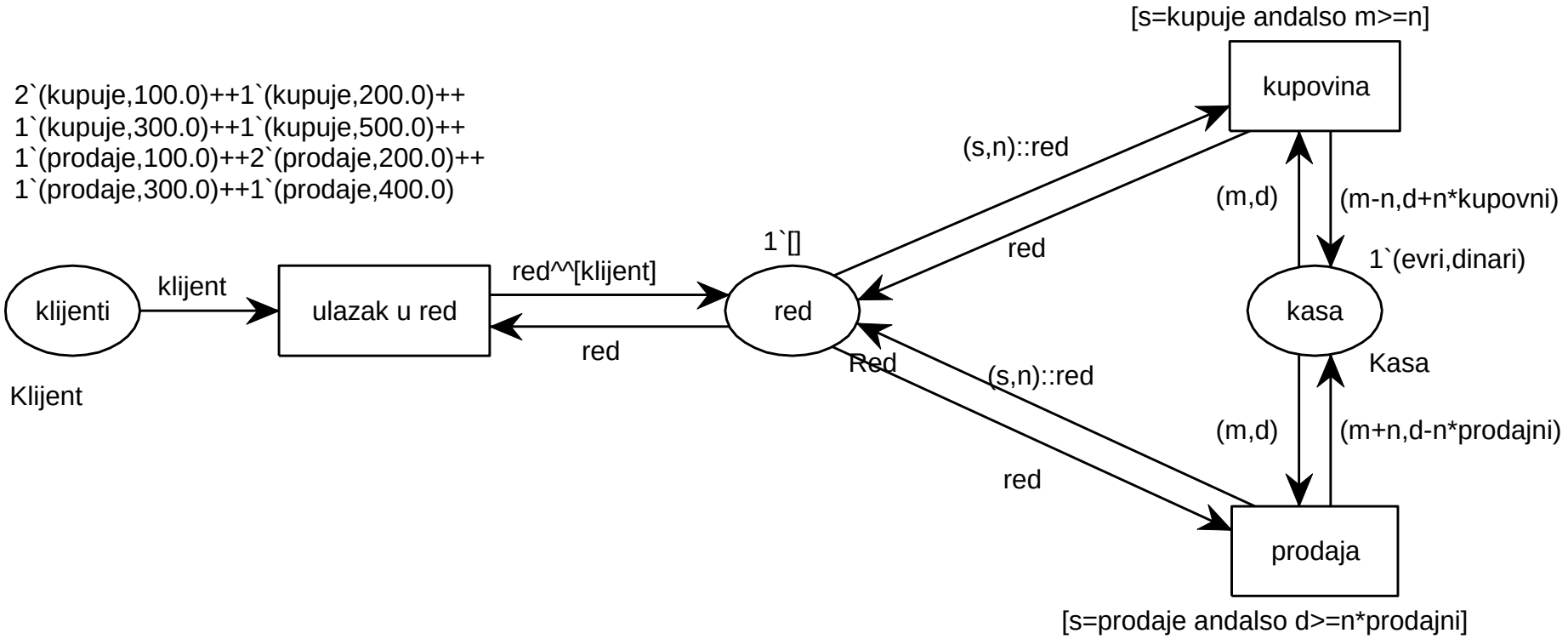
У каси мењачнице се тренутно налази 500 € и 70000 динара. Продајни курс износи 122,49 динара а куповни 123,23 за један €.

Клијенти се услужују по FIFO правилу.

Моделирати помоћу ОПМ тако да постоји једно место чије почетно маркирање чине сви клијенти и симулирати добијену мрежу.

```
(* Standard declarations *)
    colset REAL = real;
    colset DINARI=real;
    colset EVRI=real;
    colset Svrha = with kupuje | prodaje;
    colset Klijent = product Svrha * EVRI;
    colset Kasa = product EVRI * DINARI;
    colset Red= list Klijent;
```

```
var klijent:Klijent;
var red: Red;
var n, m: EVRI;
var s: Svrha;
var d: DINARI;
val evri = 500.0;
val dinari = 500.0;
val kupovni = 123.23;
val prodajni = 122.49;
```



Пример

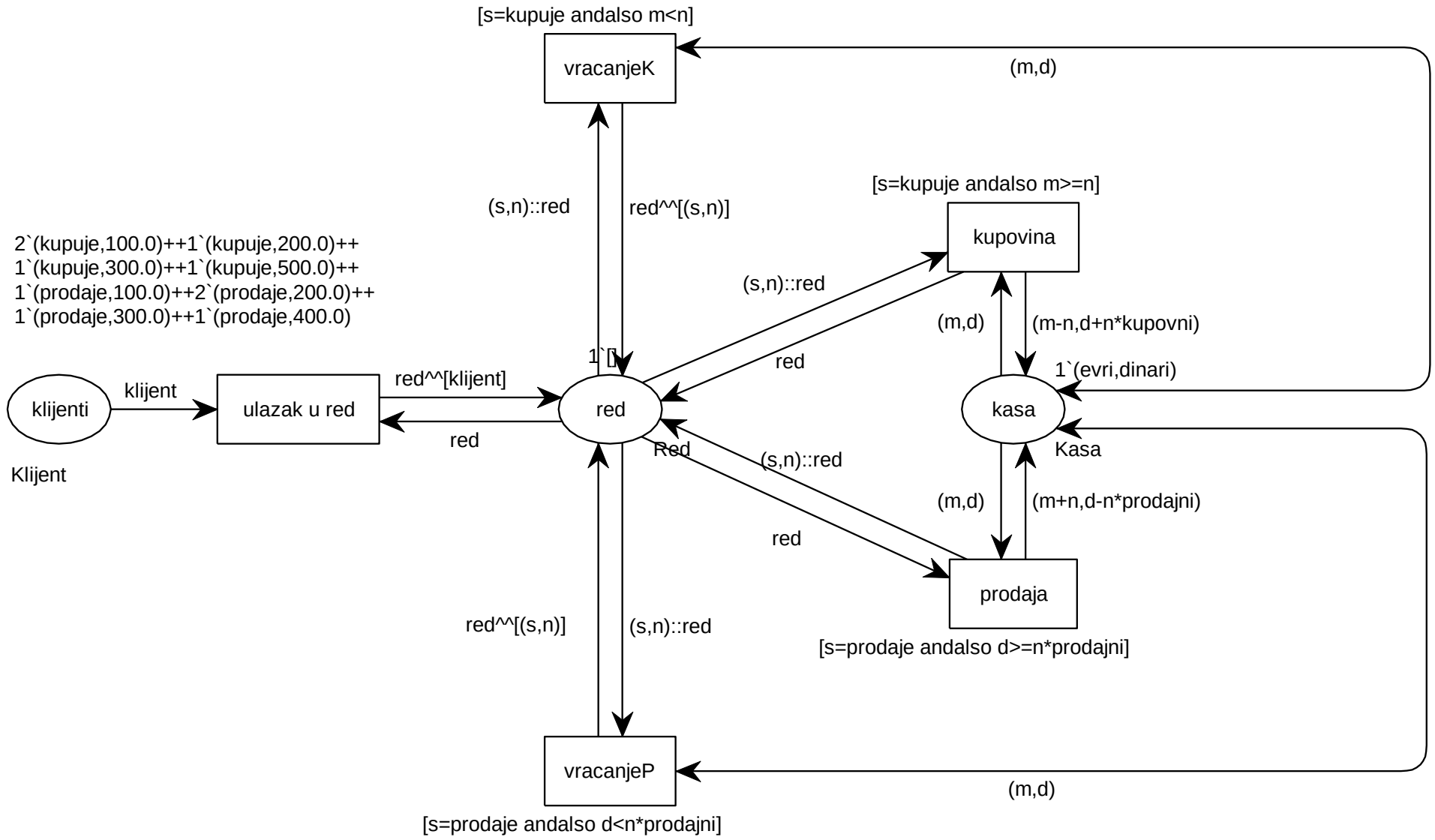
Испред мењачнице се налази 10 људи од којих неки хоће да замене динаре за евре (купе €) а неки да замене евре за динаре (продају €).

особа	A	B	C	D	E	F	G	H	I	J
купује €	100			500		200	100			300
продаје €		200	300		100			400	200	

У каси мењачнице се тренутно налази 500 € и 70000 динара. Продајни курс износи 122,49 динара а куповни 123,23 за један €.

Клијенти се услужују по FIFO правилу. Уколико мењачница нема довољно евра или динара, клијент се враћа на крај реда.

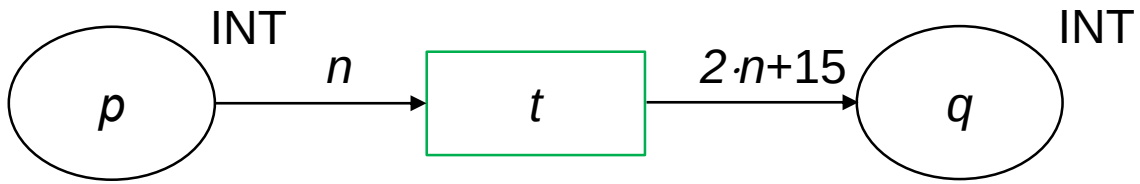
Моделирати помоћу ОПМ тако да постоји једно место чије почетно маркирање чине сви клијенти и симулирати добијену мрежу.

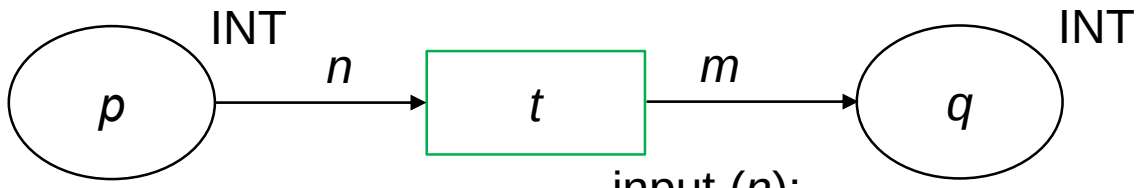

[s=kupuje andalso m<n]
vracanjeK
(m,d)
(s,n)::red
red^^[(s,n)]
[s=kupuje andalso m>=n]
kupovina
2`(kupuje,100.0)++1`(kupuje,200.0)++
1`(kupuje,300.0)++1`(kupuje,500.0)++
1`(prodaje,100.0)++2`(prodaje,200.0)++
1`(prodaje,300.0)++1`(prodaje,400.0)
(s,n)::red
red
(m,d)
(m-n,d+n*kupovni)
1`[]
1`(evri,dinari)
klijenti
klijent
ulazak u red
red^^[klijent]
red
red
kasa
Kasa
Red
(s,n)::red
(m,d)
(m+n,d-n*prodajni)
red
prodaja
Klijent
red^^[(s,n)]
(s,n)::red
[s=prodaje andalso d>=n*prodajni]
vracanjeP
(m,d)
[s=prodaje andalso d<n*prodajni]

CPN Tools – *Code segment*

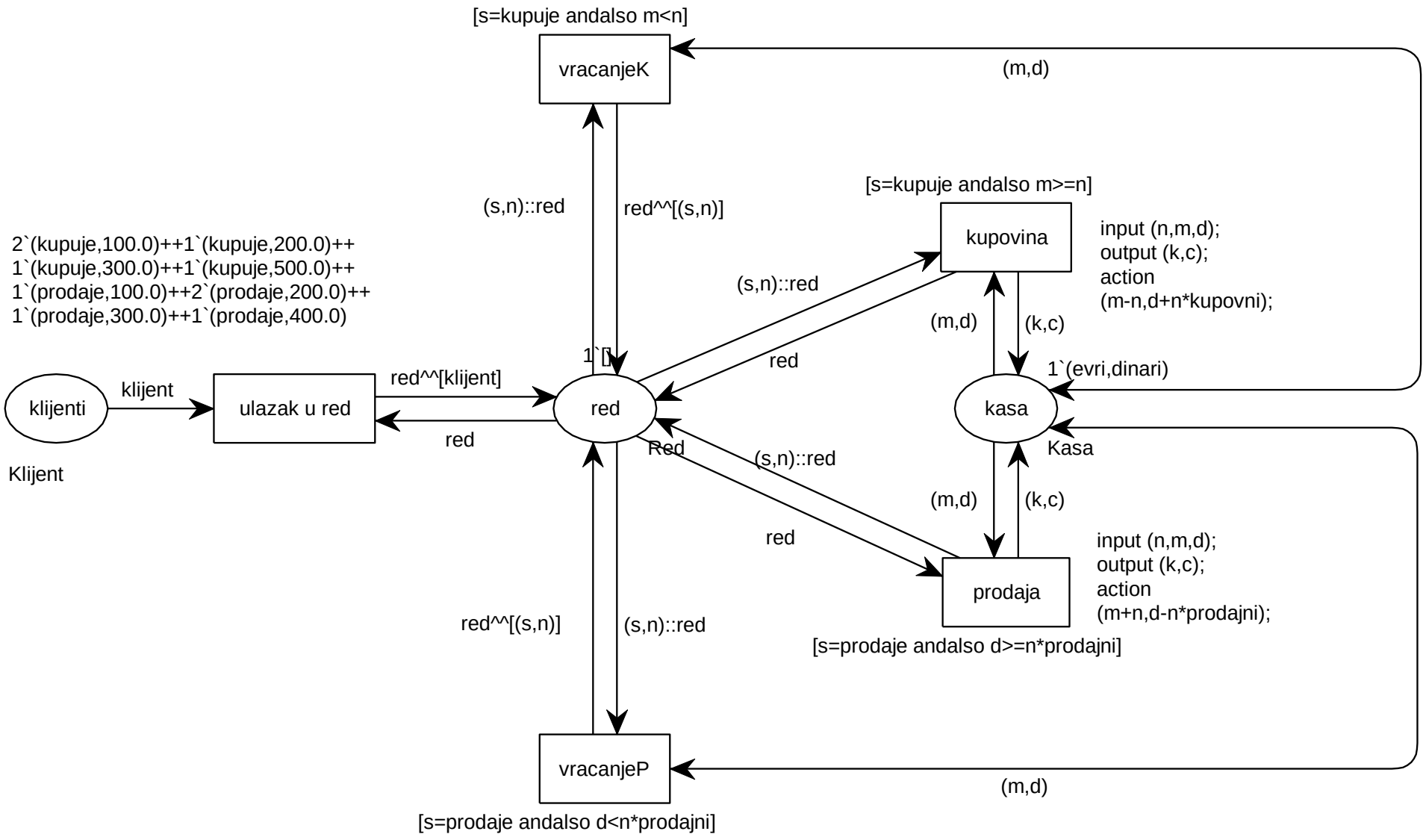
синтакса: input (); необавезни део
output (); необавезни део
action (); обавезни део

пример:





input (*n*);
output *(m);*
action (*2·n*+15);


[s=kupuje andalso m<n]
vracanjeK
(m,d)
(s,n)::red
red^^[(s,n)]
[s=kupuje andalso m>=n]
kupovina
input (n,m,d);
output (k,c);
action
(m-n,d+n*kupovni);
2`(kupuje,100.0)++1`(kupuje,200.0)++
1`(kupuje,300.0)++1`(kupuje,500.0)++
1`(prodaje,100.0)++2`(prodaje,200.0)++
1`(prodaje,300.0)++1`(prodaje,400.0)
(s,n)::red
red
(m,d)
(k,c)
1`[]
1`(evri,dinari)
klijenti
klijent
ulazak u red
red^^[klijent]
red
red
kasa
Red
Kasa
(s,n)::red
Klijent
(m,d)
(k,c)
red
input (n,m,d);
output (k,c);
action
(m+n,d-n*prodajni);
prodaja
red^^[(s,n)]
(s,n)::red
[s=prodaje andalso d>=n*prodajni]
vracanjeP
(m,d)
[s=prodaje andalso d<n*prodajni]

CPN Tools – Контролне структуре

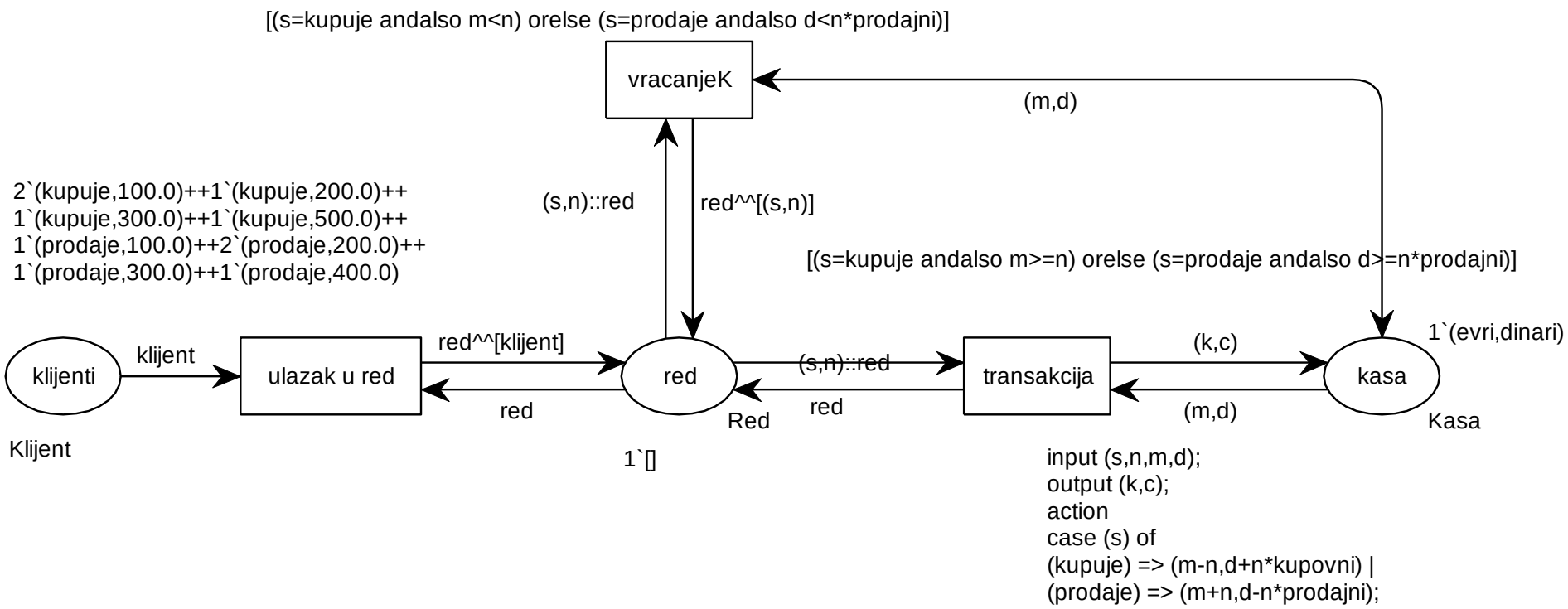
синтакса: if bool-exp then exp1 else exp2; exp1 i exp2 су истог типа

```
case exp of
pat1 => exp1
| pat2 => exp2
| ...
| patn => expn;
```

exp1,…, expn су истог типа

пример: if m>=n then m-n else m;

```
case s of
kupuje => m-n
| prodaje =>m+n;
```


[(s=kupuje andalso m<n) orelse (s=prodaje andalso d<n*prodajni)]
vracanjeK
(m,d)
2`(kupuje,100.0)++1`(kupuje,200.0)++
1`(kupuje,300.0)++1`(kupuje,500.0)++
1`(prodaje,100.0)++2`(prodaje,200.0)++
1`(prodaje,300.0)++1`(prodaje,400.0)
(s,n)::red
red^^[(s,n)]
[(s=kupuje andalso m>=n) orelse (s=prodaje andalso d>=n*prodajni)]
1`(evri,dinari)
klijenti
klijent
ulazak u red
red^^[klijent]
red
(s,n)::red
red
transakcija
(k,c)
(m,d)
kasa
Kasa
Klijent
1`[]
Red
input (s,n,m,d);
output (k,c);
action
case (s) of
(kupuje) => (m-n,d+n*kupovni) |
(prodaje) => (m+n,d-n*prodajni);

Проширења Петријевих мрежа: временска ПМ

Временским ПМ се моделира трајање (кашњење) у процесу додељивањем трајања паљењу прелаза.

Предуслов: постојање временских типова података.

У случају временског типа, жетон поред своје стандардне структуре садржи и временску вредност - најранији временски тренутак у коме се жетон може искористити за паљење неког прелаза, тј. тренутак када жетон почиње да постоји.

Да би се ово омогућило, уводи се глобални сат који моделира време.

Проширења Петријевих мрежа: временска ПМ

Прелаз може запалити (догодити) када је омогућен и спреман (*ready*).

- Прелаз је омогућен, као и у случају невременске CPN, када се у његовим улазним местима налази бар онолико жетона колико је задато описом одговарајуће гране.
- Прелаз је спреман када је временска вредност жетона у улазним местима већа или једнака тренутном времену глобалног сата.

У одређеном моменту глобалног сата пале се сви прелази који су омогућени и спремни; затим се глобални сат помера на први временски тренутак у коме је следећи прелаз омогућен и спреман.

Проширења Петријевих мрежа: временска ПМ

Нека је паљењу прелаза t придружено трајање τ, и нека паљење t почиње у тренутку T_0. Онда се паљење прелаза t састоји у:

– уклањању жетона из свих места $p \in {}^{o}t$ у тренутку T_0,

- додавању жетона у местима $p \in t^{o}$ са временским типом података у тренутку $T_0 + \tau$ и

- додавању жетона у местима $p \in t^{o}$ са невременским типом података у тренутку T_0.

CPN Tools – Временски типови података (*Timed color sets*)

Сваки тип података може бити дефинисан као временски тип.

синтакса: colset name = … timed;

пример:colset E = unit with e timed; временесски неструкт. тип под.

colset Broj = int timed; временски целобројни тип под.

colset Svrha =with uplata | isplata timed; временски набројиви тип под.

Пример

Испред мењачнице се налази 10 људи од којих неки хоће да замене динаре за евре (купе €) а неки да замене евре за динаре (продају €).

особа	A	B	C	D	E	F	G	H	I	J
купује €	100			500		200	100			300
продаје €		200	300		100			400	200	

У каси мењачнице се тренутно налази 500 € и 70000 динара. Продајни курс износи 122,49 динара а куповни 123,23 за један €.

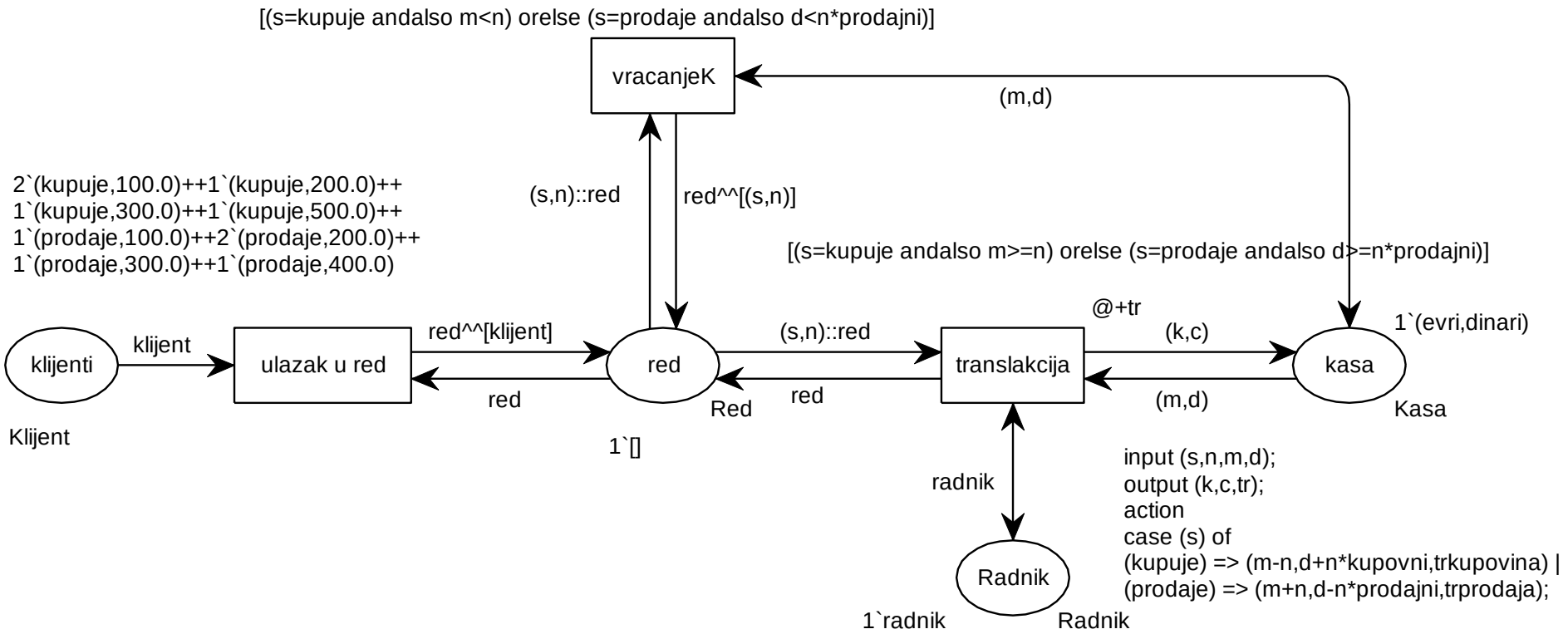
Клијенти се услужују по FIFO правилу. Уколико мењачница нема довољно евра или динара, клијент се враћа на крај реда.

У мењачници ради један радник. Куповина евра траје 3 минута а продаја 4 зато што радник мора да провери исправност новчаница.

Моделирати помоћу ОПМ и симулирати добијену мрежу.

```
(* Standard declarations *)
    colset UNIT = unit;
    colset INT = int;
    colset REAL = real;
    colset DINARI=real;
    colset EVRI=real;
    colset Svrha = with kupuje | prodaje;
    colset Klijent = product Svrha * EVRI;
    colset Kasa = product EVRI * DINARI;
    colset Red= list Klijent;
    colset Radnik = unit with radnik timed;
```

```
var klijent:Klijent;
var red: Red;
var n, m, k: EVRI;
var s: Svrha;
var d, c: DINARI;
var tr: INT;
val evri = 500.0;
val dinari = 70000.0;
val kupovni = 123.23;
val prodajni = 122.49;
val trkupovina = 3;
val trprodaja = 4;
```



Проширења Петријевих мрежа: стохастичка ПМ

Стохастичка ПМ (*Stochastic PN*) је свака ПМ у којој постоје стохастичке величине (променљиве).

Стохастичке величине могу бити:

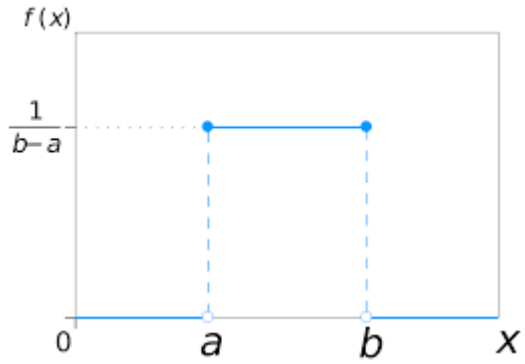
- трајање паљења прелаза,
- вредност жетона одређеног типа.

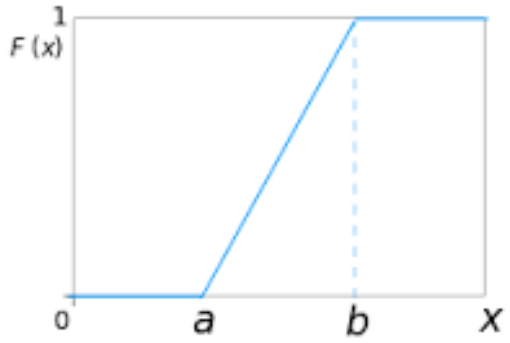
CPN Tools – Расподеле случајних величина

Uniform distribution Униформна расподела

синтакса: uniform(a:real, b:real) : real

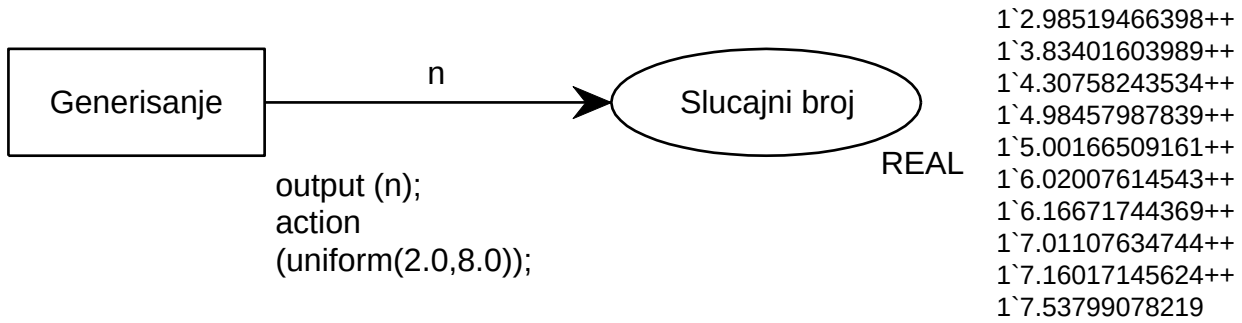
карактериситке: средина: (a+b)/2 варијанса: ((b-a)^2)/12



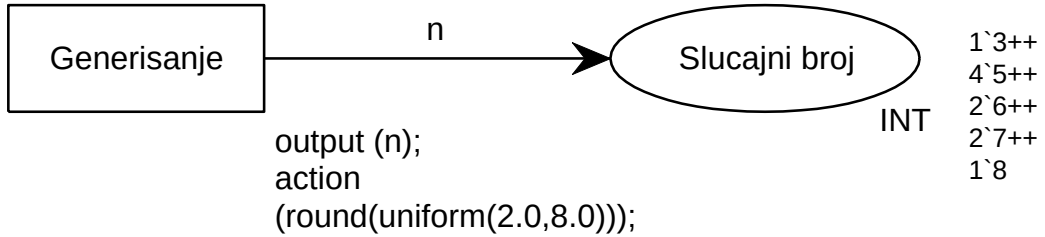


CPN Tools – Расподеле случајних величина

пример: uniform(2.0, 8.0) генерисање случајног реалног броја

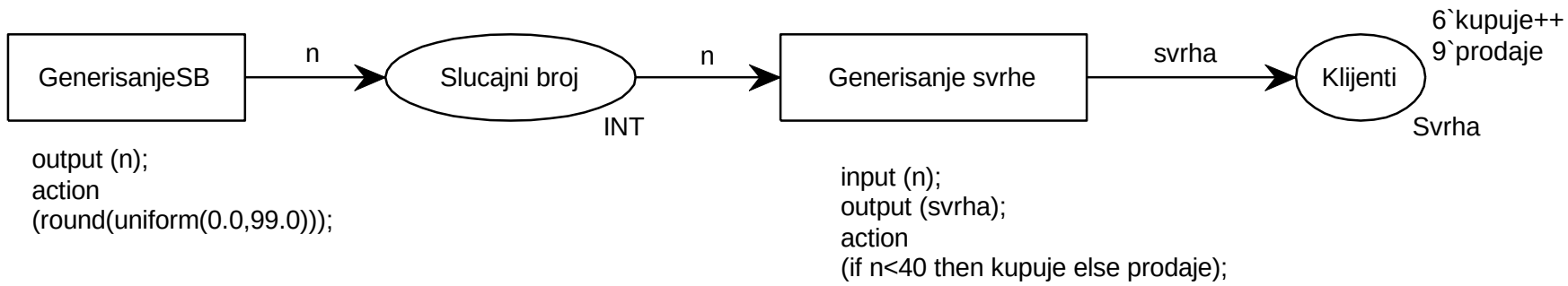


пример: round(uniform(2.0, 8.0)) генерисање случајног целог броја

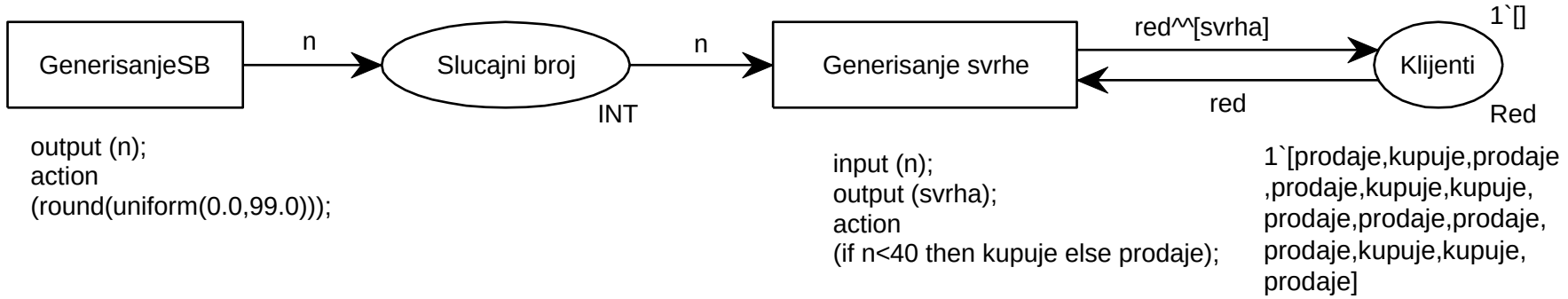


CPN Tools – Расподеле случајних величина

пример: У току дана 40% клијената долази пред мењачницу да би купило а 60% да би продало евре.

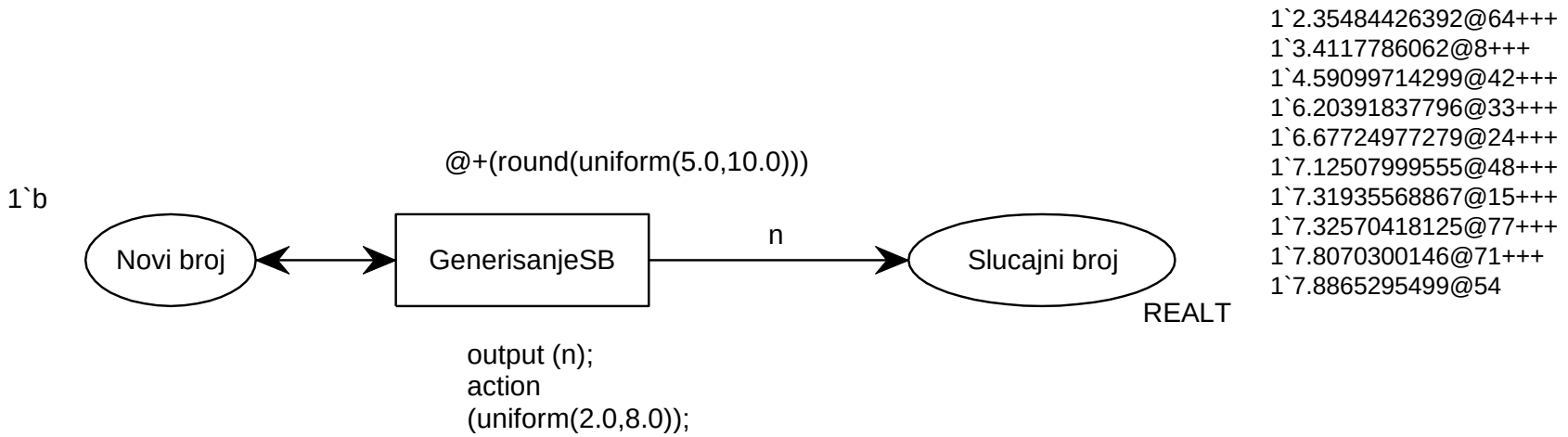


пример: Клијенти стају у ред.



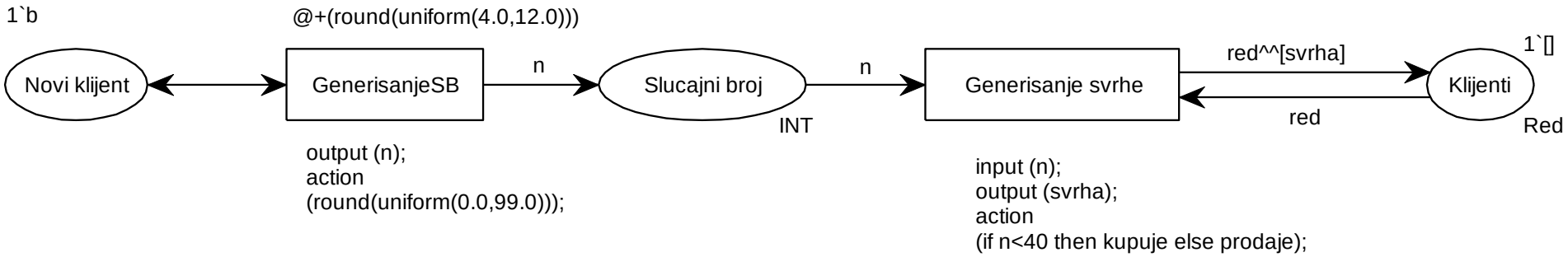
CPN Tools – Расподеле случајних величина

пример: Време генерисања новог реалног случајног броја је униформно распоређена величина између 5 и 10. Трајање паљења прелаза подлеже униформној расподели.



CPN Tools – Расподеле случајних величина

пример: Време између доласка два клијента је униформно распоређена величина између 4 и 12 минута.

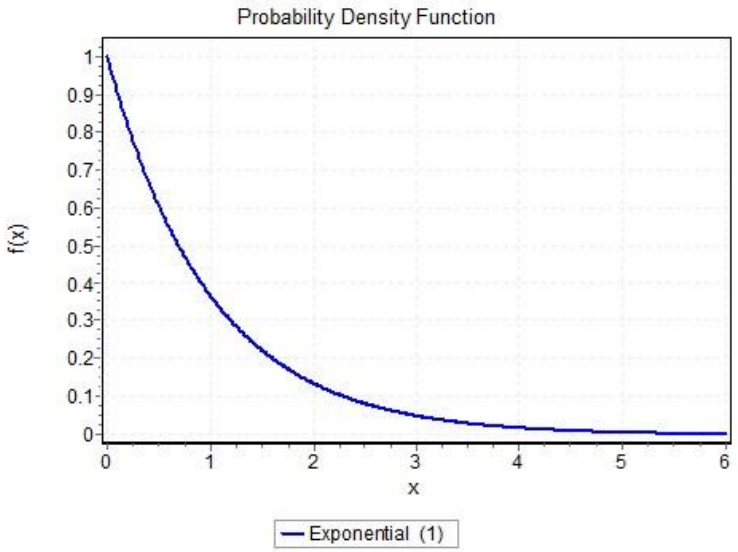


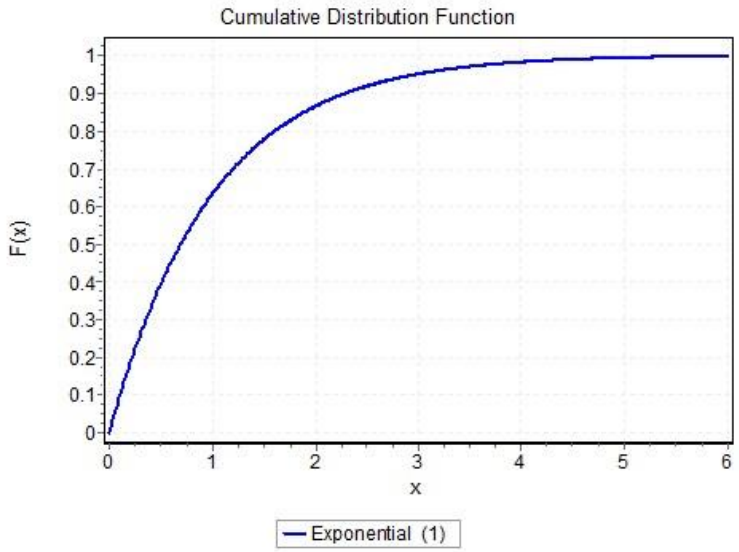
CPN Tools – Расподеле случајних величина

Exponential distribution Експоненцијална расподела

синтакса: exponential(r:real) : real

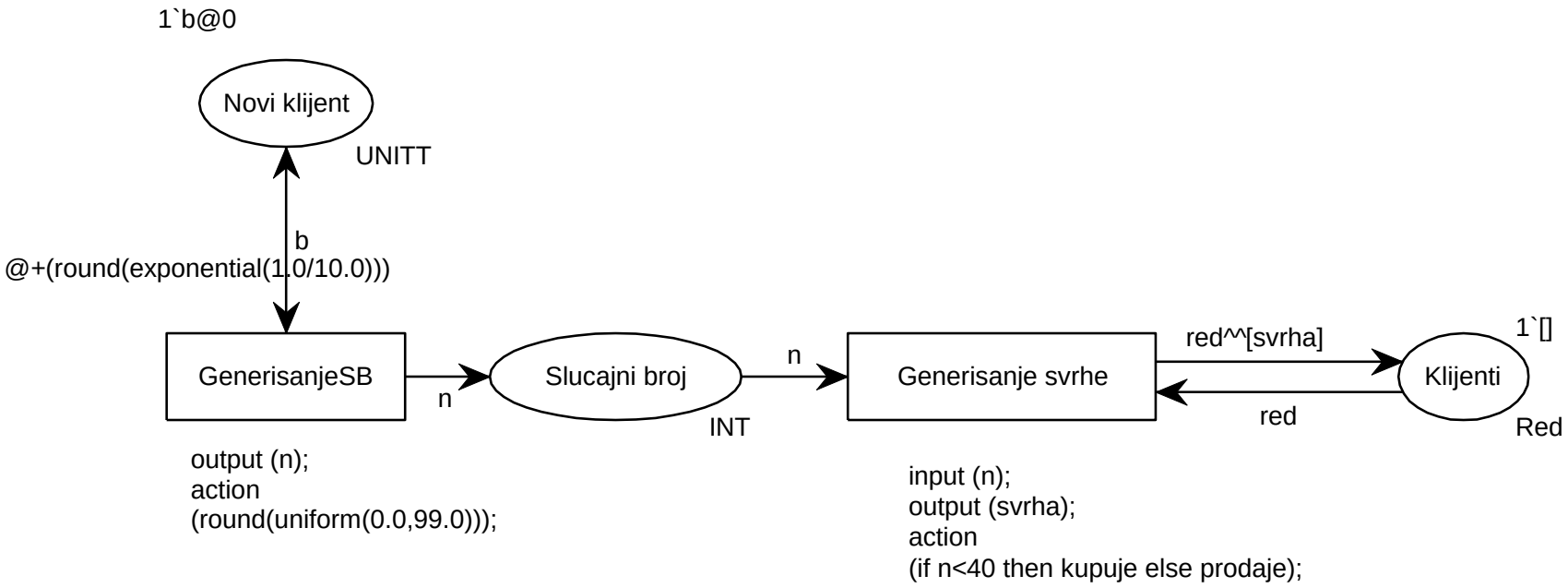
карактериситке: средина: 1/r варијанса: 1/r^2





CPN Tools – Расподеле случајних величина

пример: Средње време између доласка два клијента је 10 минута и подлеже експоненцијалној расподели.

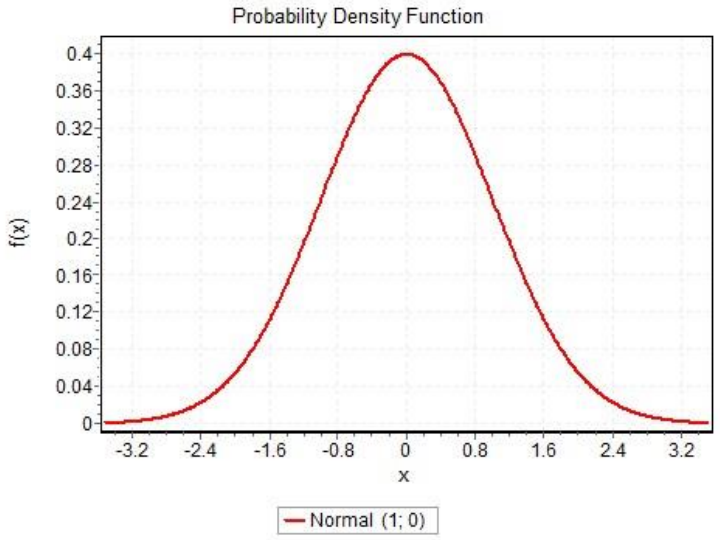


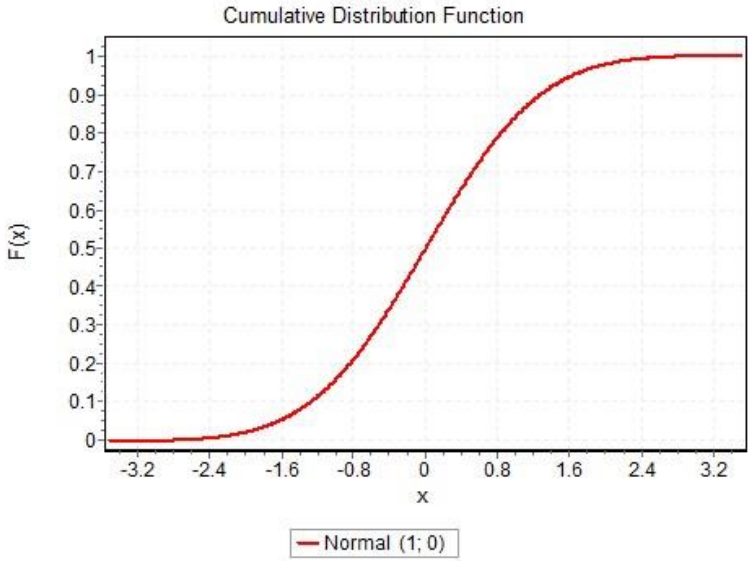
CPN Tools – Расподеле случајних величина

Normal distribution Нормална расподела

синтакса: normal(n:real, v^2:real) : real

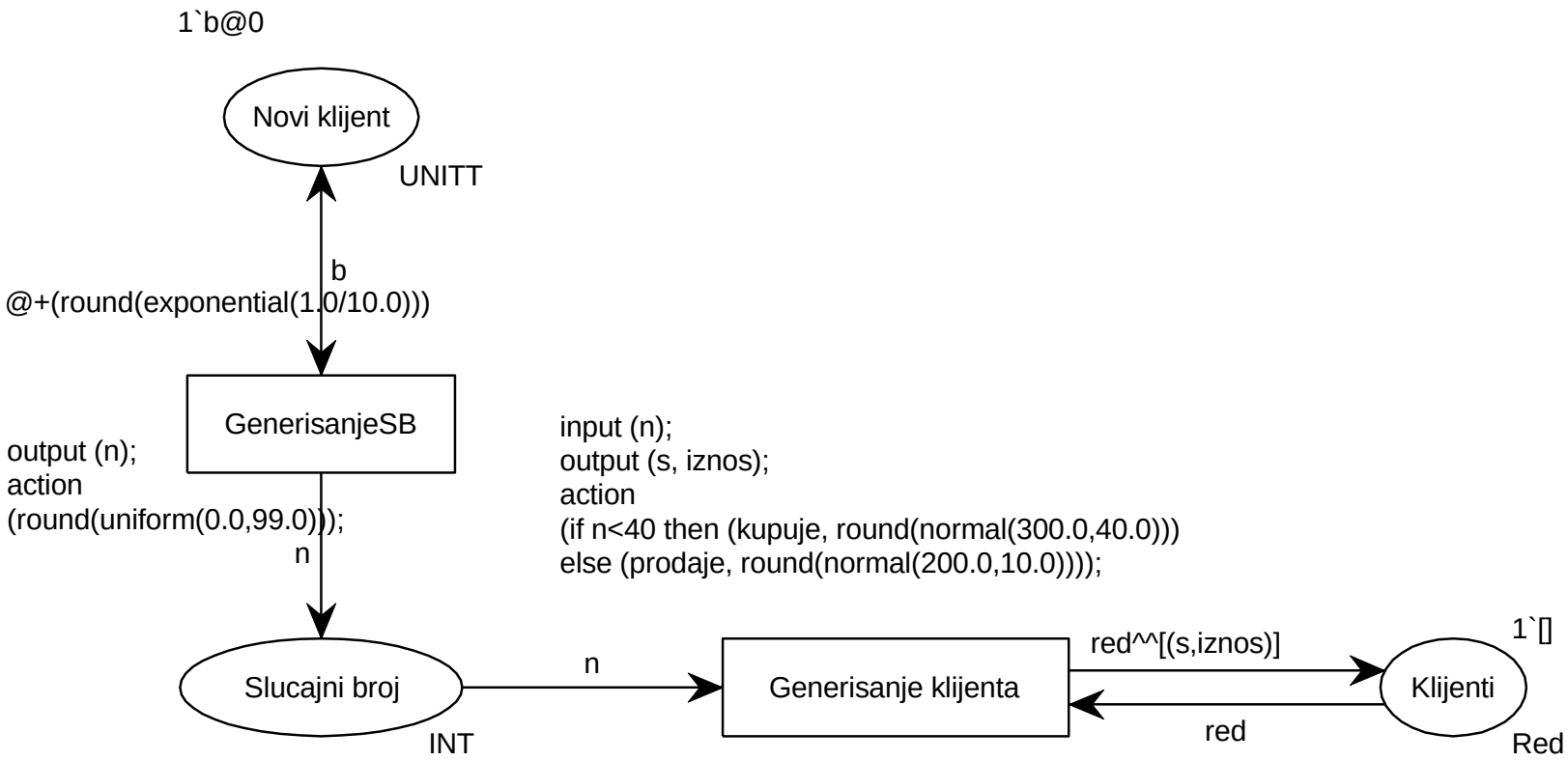
карактериситке: средина: n варијанса: v





CPN Tools – Расподеле случајних величина

пример: Износи евра коју клијенти купују и продају подлежу нормлној расподели. Средња вредност куповине износи 300 евра са варијансом 40 а продаје 200 евра са варијансом 10.



CPN Tools – Расподеле случајних величина

пример: Сваком клијенту доделити и тренутак када је ушао у систем.

```
(* Standard declarations *)
    colset UNITT = unit timed;
    colset INT = int;
    colset TIME = time;
    colset EVRI = INT;
    colset Svrha= with kupuje|prodaje;
    colset Klijent = product Svrha * EVRI * TIME timed;
    colset Red= list Klijent timed;
    var n: INT;
    var s:Svrha;
    var iznos: EVRI;
    var red: Red;
    var b: UNITT;
    var t: TIME;
```