საქართველოს ტექნიკური უნივერსიტეტი

გია სურგულაძე, დავით გულუა,
ეკატერინე თურქია

ბიზნესპროცესების მოდელირება პეტრის ქსელებით

დამტკიცებულია:
სტუ-ს სარედაქციო-
საგამომცემლო
საბჭოს მიერ

თბილისი
2014

უაკ 004.5

სახელმძღვანელოში გადმოცემულია კორპორაციული სისტემების ბიზნეს-პროცესების მოდელირებისა და ანალიზის თეორიული საფუძვლები და მათი რეალიზაციის ინსტრუმენტული საშუალებანი პეტრის ქსელების ბაზაზე. განხორციელებულია პეტრის ქსელების კლასიფიკაცია, აღწერილია ელემენტარული და სისტემური პეტრის ქსელების ძირითადი პრინციპები და გამოყენების სფეროები. განსაკუთრებით გამახვილებულია ყურადღება მართვის საინფორმაციო სისტემებში კოლექტიური მოხმარების რესურსების ეფექტური მართვის ამოცანის გადაწყვეტაზე, მათი ბიზნეს-პროცების მოდელირებასა და ანალიზზე.

განკუთვნილია პირველ რიგში, მართვის საინფორმაციო სისტემების (Management Information Systems) სპეციალობის მაგისტრანტებისათვის, აგრეთვე თეორიული და პრაქტიკული ინფორმატიკისა და სხვა დარგების სპეციალისტებისათვის, დოქტორანდებისა და სტუდენტებისათვის, რომლებიც დინამიკური ტექნოლოგიური პროცესების მოდელირებასა და კვლევას აწარმოებენ.

რეცენზენტი:

- ტ.მ.დ.,პროფ. **ზ. გასიტაშვილი**

პროფ. გ. სურგულაძის რედაქციით

© სტუ-ს „IT-კონსალტინგის ცენტრი“ (online გამოცემა 2014)

ISBN 978-9941-14-125-6 (print გამოცემა 2008**)**

Georgian Technical University

**GIA SURGULADZE, DAVID GULUA,
EKATERINA TURKIA**

MODELLING OF BUSINESS-PROCESSES WITH PETRI NETWORK

Supported by DAAD (Germany)

The description of theoretical bases of modelling and the analysis business-processes and tool means for their realization on the basis of Petri networks is offered. Classification of Petri networks is given. Main principles and sphere of use of elementary and system Petri networks are described. Special attention are given questions of the decision of a problem of efficient control by computing resources of collective using in information management systems, on the basis of modelling and the analysis their business-processes.

ISBN 978-9941-14-125-6 (print, 2008)

გია სურგულაძე

სტუ-ს ინფორმატიკის ფაკულტეტის „მართვის ავტომატიზებული სისტემების“ მიმართულების ხელმძღვანელი, სრული პროფესორი, ტექნიკის მეცნიერებათა დოქტორი. 200-ზე მეტი სამეცნიერო ნაშრომის და წიგნის ავტორი ინფორმაციულ ტექნოლოგიათა სფეროში.

ეკატერინე თურქია

სტუ-ს ინფორმატიკის ფაკულტეტის „მართვის ავტომატიზებული სისტემების“ კათედრის ასოცირებული პროფესორი, ტექნიკის მეცნიერებათა კანდიდატი. 20-ზე მეტი სამეცნიერო ნაშრომის და წიგნის ავტორი ბიზნეს-პროცესების მოდელირება - დაპროექტების სფეროში Web-ტექნოლოგიებით.

დავით გულუა

ბერლინის ჰუმბოლდტის უნივერსიტეტის კომპიუტერული ცენტრის სერვერების მენეჯერი. სტუ-ს „მართვის ავტომატიზებული სისტემების“ კათედრის ყოფილი ასპირანტი, ტექნიკის მეცნიერებათა კანდიდატი. 20-ზე მეტი სამეცნიერო ნაშრომის და წიგნის ავტორი მართვის საინფორმაციო სისტემების დაპროგრამება-მოდელირების სფეროში სისტემური პეტრის-ქსელებით.

შინაარსი

შესავალი

ბიზნეს-პროცესების სამუშაო ნაკადების (**Workflow**) მოდელირება თანამედროვე მოდელირების ერთ-ერთი უმნიშვნელოვანესი სფეროა, რომელიც მჭიდრო კავშირშია პეტრის ქსელების თეორიასთან [1,2]. პეტრის ქსელები ერთნაირი წარმატებით გამოიყენება სამუშაო ნაკადებისა და ბიზნეს-პროცესების მოდელირების, სიმულაციისა და ვერიფიკაციის ეტაპებზე. მეორე მხრივ, ბიზნეს-პროცესების შესრულების ენა **(Buziness Process Execution Language; BPEL)** წარმოადგენს დე ფაქტო სტანდარტს ვებ-სერვისებზე დაფუძნებული ბიზნეს-პროცესების აღსაწერად. იგი შეიქმნა 2003 წელს **IBM**- და **Microsoft**-ფირმების ერთობლივი მუშაობის შედეგად, ხოლო ენის თეორიული საფუძვლები და სტანდარტიზაციის პროცედურა ევროპის რამდენიმე უნივერსიტეტის ფარგლებში განხორციელდა, მათ შორის ბერლინის ჰუმბოლდტის უნივერსიტეტის „თეორიული დაპროგრამების“ კათედრაზე (ხელმძღვ. პროფ. ვ. რეისიგი) [1,3].

სამუშაო ნაკადებისა და ბიზნეს-პროცესების პეტრის ქსელის მოდელები **BPEL**-ენაზე „ითარგმნება“ აბსტრაქტული **BPEL**-პროცესების სახით, რომლებიც შემდგომ საკმაოდ მარტივად შეიძლება გარდაიქმნას შესრულებად **BPEL**-პროცესებად. ეს სქემა **UML**-ტექნოლოგიაში **UML**-დიაგრამების ობიექტ-ორიენტირებულ კლასებში ტრანსფორმაციას შეიძლება შევადაროთ და იგი აქტუალური მიმართულებაა დღეს [4,5].

არსებობს პეტრის ქსელების სხვადასხვა კლასები, რომლებსაც ერთმანეთთან მჭიდრო კავშირი აქვს და მრავალი ცალკეული ტიპის პეტრის ქსელებისგან შედგება. შესაბამისად, უკვე გასული საუკუნის ბოლოდან აქტუალური გახდა მთელი ამ მრავალფეროვნების მოწესრიგებისა და სტანდარტიზების პრობლემა, რათა სხვადასხვა ტიპის პეტრის ქსელებთან მომუშავე მეცნიერებმა ან პროგრამებმა ერთმანეთს უკეთესად „გაუგონ“. შემოთავაზებულია რამდენიმე მიდგომა პეტრის ქსელების სტანდარტიზაციისთვის, თუმცა რაიმე საბოლოო შედეგზე საუბარი ჯერ ნაადრევია პეტრის ქსელების მრავალსახიანობისა და შესაბამისად, დასამუშავებელი ინფორმაციის დიდი მოცულობის გამო [2].

სტანდარტიზაციის საერთაშორისო ორგანიზაციაში **(ISO)** პეტრის ქსელების სტანდარტიზაციის პროცედურა 1995 წელს დაიწყო და ჯერ არ დასრულებულა. მწვავე დისკუსიების საგანი ხდება ერთი შეხედვით ისეთი მარტივი საკითხიც კი, როგორიცაა, მაგალითად, პეტრის ქსელის პოზიციის აღმნიშვნელ სიმბოლოდ **S**–ის ან **P**-ს გამოყენება. პირველს გერმანელი მეცნიერები მოითხოვენ (გერმანულ **Stelle**-ზე დაყრდნობით), რადგან თეორიის ავტორი გერმანელი დოქტორი კარლ ადამ პეტრია და საერთოდ გერმანელები ამ თეორიის განვითარების საქმეში ჯერჯერობით მთავარ როლს თამაშობენ. სხვა ქვეყნის წარმომადგენელთა აზრით, ინგლისური **Place**-ს საწყისი სიმბოლოს გამოყენება მსოფლიოს მასშტაბით უფრო გასაგები და ადვილად აღსაქმელი იქნება.

თეორიულმა პროგრესმა ბოლო ათწლეულებში პრაქტიკული ნაყოფიც გამოიღო და პეტრის ქსელების 100–ზე მეტი ინსტრუმენტიდან (სიმულატორები) მრავალი უკვე კომერციული პროდუქტია, რომლებიც დიდი წარმატებით გამოიყენება სრულიად განსხვავებულ საპრობლემო სფეროებში საწარმოო პროცესების მოდელირებისთვის.

ინტერნეტის მასობრივი განვითარების პირობებში სხვადასხვა სიმულატორების თავსებადობის პრობლემა მეტად გამწვავდა. სადღეისოდ სიმულატორები იქმნება უნივერსიტეტებში თითქმის სრულიად დამოუკიდებელი ალგორითმებით, ისინი სხვა სიმულატორების მონაცემთა ანალიზს, ასახვას და დამუშავებას ვერ ახერხებს, რაც ახალი სიმულატორების შექმნისას ხშირად სამუშაო დროის ფუჭი ხარჯვის მიზეზი შეიძლება გახდეს.

სადღეისო ამოცანაა ინსტრუმენტებისთვის ერთგვარი საბაზო პლატფორმის შექმნა, სადაც პეტრის ქსელის ახალი ტიპის განსაზღვრა იქნება შესაძლებელი, ხოლო ძველი ტიპების თვისებათა თავიდან დაპროგრამების საჭიროება აღარ იარსებებს.

წინამდებარე სახელმძღვანელოში გადმოცემულია პეტრის ქსელის თეორიული საფუძვლებისა და პრაქტიკული მოდელირების ინსტრუმენტის აღწერა. იგი მოიცავს მასალას ელემენტარული პეტრის ქსელებიდან - სისტემურ პეტრის ქსელებამდე. პეტრის ქსელების სტანდარტიზაციის საკითხთან დაკავშირებითაც შემოთავაზებულია პრობლემის გადაწყვეტის ერთი მცდელობა.

წიგნი 15 თემისგან შედგება, რომლებშიც ლოგიკური თანამიმდევრობითაა გადმოცემული როგორც ბიზნეს-პროცესების არსი, მათი მოდელირების პრობლემები და ამოცანები, ასევე პეტრის ქსელების თეორიული და გამოყენებითი ასპექტები ამ პროცესების მოდელირებისათვის. თავიდან შემოთავაზებულია პეტრის ქსელების სამყაროს სადღეისო ვითარების ანალიზი და ისმება ამოცანა მისი უნიფიცირებისათვის. შემდეგ აღიწერება პეტრის ქსელების ახალი გაცვლითი ფორმატი, პეტრის ქსელების ენა **(PNML Petri Net Marcup Language),** რომელიც სადღეისოდ ყველაზე გავრცელებულ ინტერნეტ **XML**-ტექნოლოგიაზეა დამყარებული.

განიხილება პეტრის ქსელების მოდელების აგება დინამიკური სისტემებისთვის, ისეთი აქტუალური საპრობლემო სფეროებისთვის, როგორიცაა ოპერაციული სისტემები, ქსელური პროტოკოლები, ცენტრალიზებული მონაცემთა ბაზები, სერვერთა ვირტუალიზაცია (სერვერ-კლასტერების ბაზაზე) და სხვა, შესრულდება მოდელების ტრანსფორმაცია ახალ გაცვლით ფორმატში. აღიწერება ნახსენები მოდელების პრაქტიკული რეალიზების ნიმუშები.

ბოლოს გადმოცემულია მიღებული თეორიული შედეგების პრაქტიკულად გამოყენების საკითხები ობიექტ-ორიენტირებული მოდელირებისა და ვიზუალურ-კომპონენტური დაპროგრამების ინსტრუმენტების საფუძველზე, ბიზნეს-პროცესების მოდელირების ამოცანების გადასაწყვეტად. აქვე განიხილება ახალი მიმართულება ინფორმაციულ ტექნოლოგიებში, როგორიცაა BPMN, BPEL და PetNet ურთიერთკავშირი [5].

იმედია, წიგნი სასარგებლო და საინტერესო იქნება მკითხველთათვის. მისი გამოყენება შეუძლიათ მაგისტრანტებს, დოქტორანტებს და სტუდენტებსაც, რომლებიც ეწევიან სამეცნიერო-კვლევით სამუშაოებს კომპიუტერული მოდელირების სფეროში.

1. ბიზნეს-ობიექტები და ბიზნეს პროცესები. მათი ფორმალიზაციისა და მოდელირების პრობლემები, ამოცანები და თანამედროვე ინფორმაციული ტექნოლოგიები

წინამდებარე პარაგრაფში განხილულია ბიზნეს-პროცესების მოდელირების თანამედროვე პრინციპები და საშუალებები BPMN-ის (Business Process Modeling Notation - ბიზნეს-პროცესების მოდელირების ნოტაცია) ბაზაზე. მისი მიზანია ბიზნეს-სტრუქტურების დაპროექტებისა და აგებისთვის მოდელირების გრაფიკული ელემენტების სტანდარტიზაცია და ერთიანი საინფორმაციო ტექნოლოგიური ინფრასტრუქტურის შექმნა. წარმოდგენილი კონცეფცია ხელს უწყობს ბიზნეს-ოპერაციების შეფასებას და ბიზნესის მუდმივ, ეტაპობრივ ოპტიმიზაციას, სტრუქტურულ და არასტრუქტურულ მონაცემთა ინტეგრაციასა და მათ შემდგომ ანალიზს, სისტემების მონიტორინგს, პროცედურებისა და პროცესების ვიზუალიზაციასა და ვერსიების მართვას [4,5].

90-იანი წლების ბოლოს ჩამოყალიბებულმა ბიზნეს-რესტრუქტურიზაციის პროცესმა საბოლოოდ დაამკვიდრა საინფორმაციო ტექნოლოგიების აუცილებლობა ბიზნესის მართვასა და განვითარებაში, რაც ნებისმიერი ბიზნეს-პროცესის ავტომატიზაციის იდეოლოგიას ატარებს. ბიზნეს-რესტრუქტურიზაციის პროცესმა პრაქტიკულად სათავე დაუდო, კონკრეტულად ბიზნესის დარგისთვის საინფორმაციო ტექნოლოგიების პლატფორმის შექმნას. ამ პლატფორმის მიზანია გახადოს ბიზნესი ინტელექტუალური და ავტომატიზებული. იგი ითვალისწინებს ბიზნეს-გარემოს ადაპტაციას საინფორმაციო ტექნოლოგიებთან, რისი შედეგეციაა ელექტრონული კომერციისა და ელექტრონული ბიზნესის სისტემები, კორპორაციული საინფორმაციო სისტემები და ა.შ. ფაქტობრივად, ამ იდეოლოგიას შემდგომში ეწოდა ბიზნეს-პროცესების მართვის საინფორმაციო ტექნოლოგია, რომელიც მოიცავს ავტომატიზებული სისტემის დაპროექტების, მოდელირებისა და აგების ყველა ეტაპს.

ნებისმიერი ავტომატიზებული საწარმოო პროცესი, წარმოების განვითარებასა და შესაბამის ცვლილებასთან ერთად

საჭიროებს ამ ცვლილებების ასახვას უკვე დანერგილ ავტომატიზებულ სისტემაში. გარდა სხვა ტექნიკური დეტალებისა, მნიშვნელოვანია უკვე არსებული სისტემის სრული სურათის ფლობა და მისი შემდგომი განვითარებისთვის თითოეული საქმიანი პროცესის დეტალური ანალიზი. ავტომატიზებული სისტემის რეალიზაციისას, საქმიანი პროცესის დეტალური პროცედურული ანალიზი, პრაქტიკულად, წარმოებს განცალკევებულად, რომლის მონაწილე მხარეები იყოფა – სისტემის ანალიტიკოსებად (ექსპერტებად) და ტექნიკურ პერსონალად (დამპროექტებლები, პროგრამისტები). თუმცა, ზოგადად, სისტემის აგება და მართვა საჭიროებს ბიზნეს-პროცესების (საქმიანი პროცესების) მთლიანი სასიცოცხლო ციკლის, არქიტექტურის, ამ პროცესებში მონაწილე როლებისა და რესურსების, ინფორმაციის, დოკუმენტების მოძრაობის, გაფორმებისა და შესრულების სრულ კონტროლს და ანალიზს. ამდენად, სისტემის მოდელის შექმნა უნდა წარმოებდეს ბიზნეს-სფეროს ყველა ძირითადი მონაწილისთვის – დაწყებული ბიზნეს-ანალიტიკოსებიდან, რომლებიც ქმნიან პროცესების პირველად ესკიზებს, ტექნიკურ დამმუშავებლებისთვის, რომლებიც პასუხისმგებელნი არიან ტექნოლოგიის დანერგვაზე, პროცესებისა და მონაცემების დამუშავებაზე, და ბოლოს, თვით ბიზნეს-მენეჯერებისთვის, რომლებიც უშუალოდ მართავენ ამ პროცესებს და ახორციელებენ მათ მონიტორინგს [6].

მიუხედავად იმისა, რომ ბიზნეს-პროცესების მოდელირებისთვის დღეისათვის საკმაოდ მოქნილ ტექნოლოგიად ითვლება უნიფიცირებული მოდელირების ენა (UML), იგი არ ასახავს ბიზნეს-სტრუქტურების სრულ სასიცოცხლო ციკლსა და ერთიან, ზოგად მოდელს. ამავდროულად, UML ენა შესაძლებლობას აძლევს დამპროექტებელს მოახდინოს სისტემის დეტალური აღწერა დეკომპოზიციური დიაგრამების სახით და ორიენტირებულია პროგრამული პროდუქტის შექმნაზე. თუმცა, ბიზნეს-სისტემების აგება საჭიროებს ბიზნესის ინტეგრალური სურათის ფლობას ანუ დეკომპოზიციური დიაგრამების კომპოზიციას, განზოგადებული, მეტა-მოდელის შექმნას, რომელიც გასაგები იქნება თვით ბიზნესის დარგის სპეციალისტებისთვისაც.

პრაქტიკულად, ბიზნეს-პროცესების მართვის საინფორმაციო სისტემები მოითხოვს უნიფიცირებული მოდელირების ენის

სრულყოფას ბიზნეს-პროცესების დაპროექტებისთვის. ამ სრულყოფის მნიშვნელოვანი ფაქტორებია: ბიზნეს-პროცესების დაპროექტების ერთიანი სივრცის შექმნა ბიზნეს-ოპერაციების შეფასებისა და ბიზნესის მუდმივი, ეტაპობრივი ოპტიმიზაციისთვის; სტრუქტურულ და არასტრუქტურულ მონაცემთა ინტეგრაცია და მათი შემდგომი ანალიზი; სისტემის მონიტორინგი; პროცედურებისა და პროცესების ვიზუალიზაცია და ვერსიების მართვა.

კიდევ ერთი პრობლემა, დღევანდელ დღეს UML-ენაზე ბაზირებული მოდელირების სისტემების სიჭარბეა, რაც მოკლებულია ერთი სრული სტანდარტის არსებობას. ამ კუთხით, ბიზნეს-პროცესების მოდელირებისთვის შეიქმნა სპეციალური სტანდარტი - ბიზნეს-პროცესების მოდელირების ნოტაცია (BPMN- Business Process Modeling Notation), რომელშიც გაერთიანებულ იქნა სისტემების მოდელირების არსებული საუკეთესო კონცეფციების (მაგალითად, UML Activity Diagram, UML EDOC Business Processes, ARIS, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM, and Event-Process Chains (EPCs) და ა.შ.) სხვადასხვა ნოტაციები, ინსტრუმენტები და მეთოდები ერთი სტანდარტული ფორმით. იგი, პრაქტიკულად ბიზნეს-სფეროს მონაწილეებიის დამაკავშირებელ ბირთვს წარმოადგენს ბიზნეს-პროცესების დამუშავებასა და რეალიზაციას შორის [7].

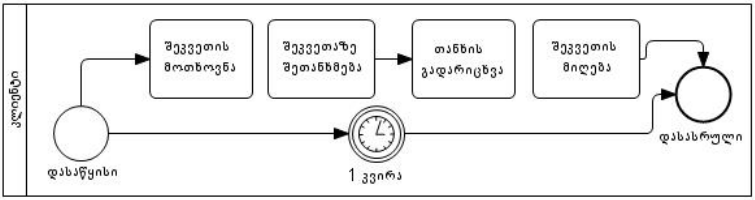
BPMN - ის მთავარი ფოკუსი პროცესზე ორიენტირებული მიდგომაა. BPMN სტანდარტის ძირითადი ბირთვია თვით ბიზნეს-პროცესი, რომლის მიმართებაშიც ხდება შემდგომში პროცესის დეტალიზაციის და ქცევის განსაზღვრა (მაგალითად, როლები, რესურსები და ა. შ.). მისი ერთ-ერთი ძირითადი მიზანია სტანდარტული გრაფიკული ნოტაციის მიხედვით ბიზნეს-პროცესების ჩაშენება ბიზნეს-ნაკადების მართვის სისტემაში ანუ სისტემის ტექნოლოგიურ პროცესში (WFMS – Workflow Management System).

ბიზნეს-პროცესების მოდელირების ნოტაციაში პრიორიტეტულია მოდელირების გრაფიკული ელემენტების ვიზუალური მხარე და დიაგრამების თავსებადობა. ამ თავსებადობის საფუძველი არის ბიზნეს-პროცესების მოდელირების ენა (BPML – Business Process Modeling Language) და ბიზნეს-პროცესების

შესრულების ენა (BPEL – Business Process Execution Language) [1], რომელიც ბაზირებულია XML (Extensible Markup Language) ენაზე და წარმოადგენს ბიზნეს-პროცესების გრაფიკულად ასახვისა და მათი ურთიერთქმედების პროტოკოლების ფორმალური აღწერის ენას, რაც ბიზნეს-მოდელისა და საინფორმაციო მოდელის სინქრონიზაციის საშუალებას იძლევა [1].

ბიზნეს-პროცესების მოდელირებისა და შესრულების ენები საშუალებას იძლევა გრაფიკულად აიგოს გამჭოლი ბიზნეს-პროცესები. არსებობს სამი ძირითადი ტიპი გამჭოლი მოდელის ქვემოდელების ფარგლებში:

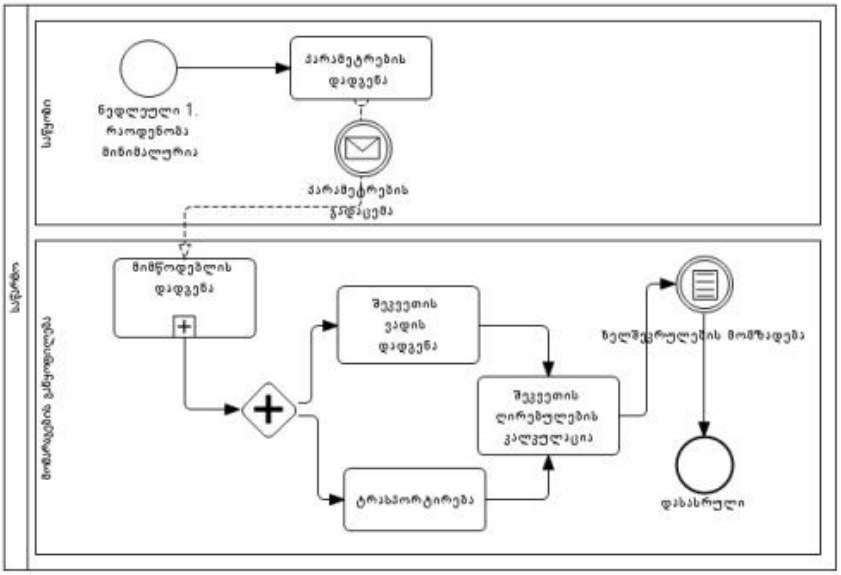
კერძო (შიგა) ბიზნეს-პროცესი, რომელიც აღწერს ტექნოლოგიურ პროცესს ანუ საქმიან ნაკადს. კერძო ბიზნეს-პროცესის მოდელის ფრაგმენტი წარმოდგენილია 1.1 ნახაზზე.



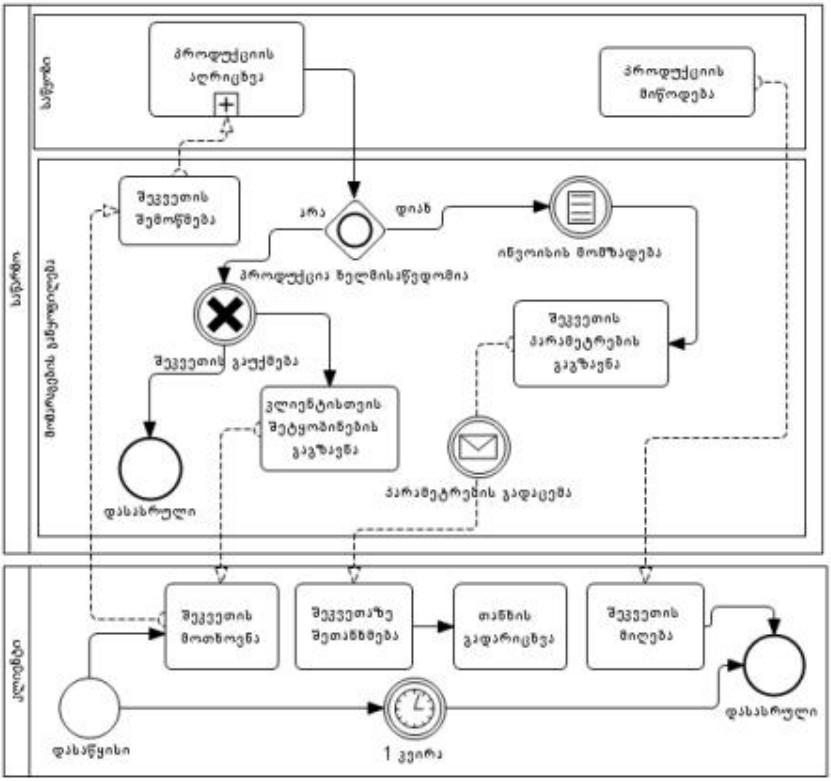
ნახ.1.1. კერძო ბიზნეს-პროცესის მოდელის ფრაგმენტი

▪ აბსტრაქტული (ღია) ბიზნეს-პროცესი. იგი, აღწერს დამოკიდებულებას ორ ან მეტ კერძო პროცესს შორის ან პროცესსა და რესურს შორის. აბსტრაქტულად ითვლება, მხოლოდ ის პროცესები, რომელთა ქმედება აუცილებლად უკავშირდება კერძო ბიზნეს-პროცესს. ამდენად, აბსტრაქტული პროცესი ასახავს იმ შეტყობინებათა გადაცემის თანამიმდევრობას, რომლებიც ურთიერთქმედებს კონკრეტულ ბიზნეს-პროცესთან (ნახ.1.2).

▪ ერთობლივი (გლობალური) ბიზნეს-პროცესი, რომელიც ასახავს ურთიერთქმედებას ორ ან მეტ ბიზნეს-ობიექტს შორის და აერთიანებს აბსტრაქტულ ბიზნეს-პროცესებს. იგი წარმოადგენს, ფაქტობრივად, მეტა-მოდელს, რომელიც ქმნის კონკრეტული ბიზნეს-სტრუქტურის ერთიან სურათს (ნახ.1.3).



ნახ.1.2. აბსტრაქტული ბიზნეს-პროცესის ფრაგმენტი



ნახ.1.3. ერთობლივი ბიზნეს-პროცესის ფრაგმენტი

ბიზნეს-პროცესების მოდელირებისა და შესრულების ენებში მოდელირების ძირითად სემანტიკურ ერთეულად განიხილება ოპერაციები და შეტყობინებები, რის შედეგადაც წარმოებს დანართების სხვადასხვა ფუნქციონალური მოდულების ანუ სერვისების ურთიერთკავშირი.

პრაქტიკულად, ბიზნეს-სტრუქტურების დაპროექტებისა და აგების თანამედროვე კონცეფციაა სხვადასხვა საინფორმაციო ტექნოლოგიების ინტეგრაცია ერთ საერთო სტანდარტში. ამ კუთხით ბიზნეს-პროცესების მართვის საინფორმაციო ტექნოლოგია აყალიბებს ბიზნეს-პროცესების გამოყენების ძირეულ სპექტრს:

- ბიზნეს-პროცესების მოდელირების ენა, რომლის ბაზისია პროცეს-ორიენტირებული მოდელირება - ბიზნეს-პროცესების ანალიზითა და პროცესების იმიტაციით;

- ბიზნეს-პროცესების რეალიზაციის ენა, რის საფუძველზეც წარმოებს ორგანიზაციული პროცესების დოკუმენტირება, ვიზუალიზაცია, მათი კომუნიკაციის მხარდაჭერა და თავსებადობა;

- ბიზნეს-რესურსების ინტელექტუალური მართვის ტექნოლოგია, რაც გამოყენების პროცესზე ორიენტირებული პროგრამული უზრუნველყოფის განვითარებას მოიცავს.

2. პეტრის ქსელების თეორიული საფუძვლები: სიმრავლეები და მულტისიმრავლეები

პეტრის ქსელების ისტორია 1962 წლიდან იწყება, როცა გერმანელმა ინჟინერმა, კარლ ადამ პეტრიმ დარმშტადტის ტექნიკურ უნივერსიტეტში დაიცვა სადოქტორო დისერტაცია თემაზე "კომუნიკაცია ავტომატებით".

ამ ნაშრომში მან პირველად ჩამოაყალიბა და დაასაბუთა იდეა ორი განსხვავებული ტიპის კვანძებითა და მათი დამაკავშირებელი მიმართული რკალებით აგებული მუშა ქსელების შესახებ, რომლებიც ერთი მოდელის ფარგლებში გააერთიანებდა კონკრეტულ და აბსტრაქტულ პროცესებს და მონაცემებს.

სახელი "პეტრის ქსელები" თეორიამ მოგვიანებით მიიღო ავტორის პატივსაცემად. თავისი თეორიის საფუძვლად პეტრიმ სასრული ავტომატების, სიმრავლეთა და გრაფების თეორიის ელემენტები გამოიყენა.

არსებობის 44-წლიანი ისტორიის მანძილზე გაიბა კავშირები პეტრის ქსელებსა და სხვა მრავალ მათემატიკურ თუ არამათემატიკურ დისციპლინებთან.

ერთის მხრივ პეტრის ქსელები ფართოვდება თეორიულად, სულ უფრო მძლავრი ხდება მისი მათემატიკური აპარატი, იქმნება ახალი თეორიული კლასები, ხოლო მეორეს მხრივ მატულობს პეტრის ქსელების პრაქტიკული გამოყენების სიხშირე

ინფორმატიკის მოწინავე მიმართულებებთან მჭიდრო თანამშრომლობის შედეგად, რაც სქემატურად 2.1 ნახაზზეა გამოსახული.

ნახ.2.1. პეტრის ქსელები და მისი გარემოცვა

2.1. სიმრავლეები

სიმრავლეთა თეორია პეტრის ქსელების ერთერთი ბაზისია. განვიხილოთ მოკლედ მისი ძირითადი ელემენტები.

საწყისი აღნიშვნები:

N = {0,1,...} – ნატურალურ რიცხვთა სიმრავლე;

Z = {…,-1,0,1,…} – მთელ რიცხვთა სიმრავლე

Boolean = {true,false} – ბულის სიმრავლე

სიმრავლე არაერთგვაროვან ობიექტთა ერთობ-ლიობაა. მათ სიმრავლის **ელემენტები** ეწოდება.

a არის **A**-სიმრავლის ელემენტი, თუკი ფლობს თვისებას **a∈A** („მიეკუთვნება“). სიმრავლე მოიცემა ელემენტთა ჩამონათვალით $\mathbf{A = \{a_1, a_2, ..., a_n\}}$, ან გარკვეულ **p(a)** ფუნქციაზე დაყრდნობით, რომლის შედეგი სიმრავლის ელემენტისთვის აუცილებელ პირობას აკმაყოფილებს:

$$\mathbf{A = \{a \mid p(a)\}.}$$

ცარიელი სიმრავლე **Ø**-სიმბოლოთი აღინიშნება და გამოისახება პირობით **Ø = {a | a≠a},** რადგან პირობა **a≠a** ყოველთვის მცდარია.

სიმრავლეთა თეორიაში განისაზღვრება შემდეგი ძირითადი დამოკიდებულებები და ოპერაციები: **ქვესიმრავლე (A⊆B), ჭეშმარიტი ქვესიმრავლე (A⊂B), გაერთიანება (A∪B), თანაკვეთა (A∩B), სხვაობა (A\B),** სადაც:

A⊆B, როცა ნებისმიერი **a∈A**–თვის მართებულია **a∈B**

A⊂B, როცა **A⊆B** და **A≠B**

A∪B = { a | a∈A ან **a∈B }**

A∩B = { a | a∈A და **a∈B }**

A\B = { a | a∈A და **a∉B }**

ცარიელი სიმრავლე ნებისმიერი არაცარიელი სიმრავლის ქვესიმრავლეა: **Ø⊂A. A** და **B** სიმრავლეებს **განცალკევებული სიმრავლეები** ეწოდება, თუ **A∩B=Ø.**

სიმრავლე შეიძლება შეიცავდეს ელემეტებს, რომლებიც თავადაა სიმრავლეები**. A**-სიმრავლის ყველა შესაძლო ქვესიმრავლეთა სიმრავლე **Π(A)**-თი აღინიშნება, ნატურალურ რიცხვთა სიმრავლე 0-ის ჩათვლით – **N**-ით, ლოგიკურ მნიშვნელობათა (ჭეშმარიტი ან მცდარი) სიმრავლე – **B**-თი.

$A_1,A_2,...,A_n$ (**n∈N**) სასრულ სიმრავლეთა **პროდუქტი (დეკარტული ნამრავლი)** განისაზღვრება შემდეგნაირად:

$$A_1xA_2x...xA_n = \{ (a_1,a_2,...,a_n) \mid a_i \in A_i \ i=1,...,N \}$$

პროდუქტის ცალკეულ ელემენტს **n**-კორტეჟი ეწოდება. ყოველი **i**-სთვის, სადაც **1<i<n**, a_i-ს კორტეჟის **i**-ური ელემენტი ეწოდება ($a_1,...,a_n$). **წყვილი** განისაზღვრება, როგორც **n**-კორტეჟის კერძო შემთხვევა, 2-კორტეჟი (ბინარული).

თუ ყველა სიმრავლე **$A_1=A_2=...A_n=A$** მსგავსია, პროდუქტი ჩაიწერება A^n სახით. გარდა ამისა, $A^1 = A$ და $A^0 = Ø$.

2.2. მულტისიმრავლეები (კომპლექტები)

მულტისიმრავლე განისაზღვრება სიმრავლეში ერთი და იმავე ელემენტის რამდენჯერმე ასახვისთვის. მაგალითად, პეტრის ქსელის პოზიციაში რამდენიმე მსგავსი მარკერის აღსაწერად.

მულტისიმრავლე B არაცარიელ საბაზო A სიმრავლეზე, ეწოდება ფუნქციას:

$$B:A\to N$$

სადაც საბაზო **A** სიმრავლის ყოველი **a∈A** ელემენტის **სიხშირე B** მულტისიმრავლეში აისახება ფორმატით **B(a).** სიხშირის სიდიდე შეიძლება 0-ის ტოლიც იყოს. სიმრავლე მულტისიმრავლის სპეციალური შემთხვევაა, სადაც სიხშირის მნიშვნელობებია 0 ან 1.

მულტისიმრავლის ასახვის გაფართოებული ფორმა შემდეგია: **[a,a,...,a,b,...,b,...],** სადაც ყოველი ელემენტი თავისი სიხშირის მიხედვით მეორდება.

მულტისიმრავლეთა სიმრავლე საბაზო **A** სიმრავლეზე აღინიშნება **μA**-თი. მულტისიმრავლე ასევე შეიძლება გამოისახოს სიმბოლური ჯამის სახით, რომელიც **a∈A** ელემენტის სიხშირეს და სახელს შეიცავს:

$$B=\sum_{a\in A}B(a)a$$

თუ **B(a) = 1,** მაშინ ჯამურ ასახვაში იგი საერთოდ გამოიტოვება და იწერება მხოლოდ **a**.

B∈μA მულტისიმრავლეში**, a∈A** ელემენტს ეწოდება **B**-ს წევრი და ჩაიწერება **a∈B,** თუ **B(a)>0** და პირიქით, თუ **B(a)<0,** მაშინ **a∉B.** ცარიელი მულტისიმრავლე ∅ წევრებს არ შეიცავს: **∀a∈A, ∅(a)=0.**

მულტისიმრავლის **სიმძლავრე (კარდინალურობა)** მისი ყველა ელემენტის **სიხშირეთა ჯამს** ეწოდება და განისაზღვება შემდეგნაირად:

$$|B|=\sum_{a\in A}B(a)$$

თუ **|B|** სასრულია, მაშინ მულტისიმრავლე **B**-ს **სასრული მულტისიმრავლე** ეწოდება.

ორი მულტისიმრავლე, B_1 და B_2 **ტოლია ($B_1=B_2$),** თუ **∀a∈A, $B_1(a)=B_2(a)$.**

B_1 **ნაკლებია ან ტოლია** B_2-ის (ანუ B_2 მოიცავს B_1-ს) თუ **∀a∈A, $B_1(a)\le B_2(a)$.**

მულტისიმრავლეებზე ძირითად ოპერაციებს წარმოადგენს:

შეკრება: $\mathbf{B=B_1+B_2}$, თუ $\forall \mathbf{a \in A, B(a) = B_1(a) + B_2(a)}$

გამოკლება: $\mathbf{B=B_1-B_2}$, თუ $\forall \mathbf{a \in A, ((B_1(a) \geq B_2(a)) \wedge ((B(a) = B_1(a) - B_2(a))}$

სკალარული ნამრავლი: მულტისიმრავლის $\mathbf{B_1 \in \mu A}$ და ნატურალური რიცხვის $\mathbf{n \in N}$ სკალარული ნამრავლი განისაზღვრება როგორც $\mathbf{B=n*B_1}$, თუ $\forall \mathbf{a \in A, B(a) = n*B_1(a)}$, სადაც **"*"** არითმეტიკული გამრავლების ოპერაციაა.

2.3. პეტრის ქსელების ძირითადი ცნებები

პეტრის ქსელების ნოტაციას საბოლოო სახე ჯერ არ მიუღია. მისი მრავალმხრივი განვითარების პროცესში წარმოიშვა შეუსაბამისობების მთელი რიგი, რაც ნოტაციის სხვადასხვანაირ ინტერპრეტირებას უკავშირდებოდა.

პეტრის ქსელების სტანდარტიზაციის პროცედურა ეგრეთ წოდებული **პროექტ 15909**-ის ფარგლებში 1995 წლიდან მიმდინარეობს. მასში მონაწილეობენ ისეთი ავტორიტეტული ორგანიზაციები, როგორიცაა **სტანდარტიზაციის საერთაშორისო ორგანიზაცია (ISO), ინფორმაციულ ტექნოლოგიათა ერთიანი ტექნიკური კომიტეტი (JTC1) და საერთაშორისო ელექტროტექნიკური კომისია (IEC)**. პროექტი 3 ნაწილისგან შედგება:

1. მაღალი დონის პეტრის ქსელები – კონცეფცია, განსაზღვრებები და გრაფიკული ნოტაცია;
2. პეტრის ქსელების გაცვლითი ფორმატი, სავარაუდოდ **XML**-ის ბაზაზე;
3. მოდულური პეტრის ქსელების კონსტრუქციები – იერარქიულობა, დროითი და სტოქასტური გაფართოებები.

ქვემოთ მოცემულია პეტრის ქსელის ნოტაცია უკვე ფაქტობრივ სტანდარტად დამკვიდრებული „პროექტ 15909"-ს პირველი ნაწილის საფუძველზე [5].

ძირითადი ცნებები:

საბაზო სიმრავლე (Basis Set). ობიექტების საწყისი სიმრავლე მულტი-სიმრავლეების (კომპლექტების) შესაქმნელად.

მულტისიმრავლე ანუ კომპლექტი (Multiset). ობიექტების ნაკრები, სადაც ერთგვაროვანი ობიექტების განმეორება შესაძლებელია.

მულტისიმრავლის კარდინალურობა (Cardinality). მულტისიმრავლის ელემენტების საერთო რაოდენობა.

პოზიცია (Place). ქსელის ტიპიზებული კვანძი. ქსელის გრაფში წრით ან ელიფსით გამოისახება.

გადასასვლელი (Transition). ქსელის არატიპიზებული კვანძი, რომელიც მართკუთხედით გამოისახება.

რკალი (Arc). ქსელის მიმართული კავშირის ხაზი, რომელიც აერთებს პოზიციებს გადასასვლელებთან (შემავალი რკალი) ან პირიქით (გამომავალი რკალი).

შემავალი პოზიცია (Input Place). გადასასვლელთან შემავალი რკალით შეერთებული პოზიცია.

გამომავალი პოზიცია (Output Place). გადასასვლელთან გამომავალი რკალით შეერთებული პოზიცია.

პოზიციის ტიპი (Place Type). პოზიციასთან დაკავშირებულ მონაცემთა ელემენტების არაცარიელი სიმრავლე.

მარკერი (Marker). პოზიციასთან დაკავშირებული და შესაბამისი პოზიციის ტიპის მონაცემთა ელემენტი.

მარკირება (Marking). ყველა პოზიციაში შემავალ მარკერთა ერთობლიობა.

საწყისი მარკირება (Initial Marking). ყველა პოზიციაში შემავალ მარკერთა ერთობლიობა ქსელის მუშაობის დასაწყისში.

პოზიციის მარკირება (Place Marking). პოზიციაში მოთავსებულ მარკერთა მულტისიმრავლე.

გადასასვლელის გახსნის პირობა (Transition Condition). გადასასვლელთან დაკავშირებული ლოგიკური (ბულის) ტიპის გამოსახულება.

გადასასვლელის გახსნის რეჟიმი (Transition Mode). ცვლადების დაკავშირება გადასასვლელის გახსნის პირობასთან ისე, რომ გადასასვლელის გახსნა ნებადართული გახდეს.

გადასასვლელის გახსნის ნებართვა (Enabling a Transition). გადასასვლელი რომ გაიხსნას, ყოველი პოზიციის მარკირება უნდა აკმაყოფილებდეს მისი და გადასასვლელის დამაკავშირებელი რკალის მოთხოვნას (რკალის გამოსახულება), რაც ნიშნავს, რომ

მარკირება შეიცავს მარკერების მინიმუმ იმავე მულტისიმრავლეს, რაც რკალის გამოსახულებაზეა ასახული.

გადასასვლელის გახსნა (Transition Occurence). თუ გადასასვლელის გახსნა ნებადართულია, იგი შეიძლება გაიხსნას. ამ დროს გადასასვლელის ყოველი შემავალი პოზიციიდან მოიხსნება მარკერები გახსნის რეჟიმის შესაბამისად, ხოლო ყოველ გამომავალ პოზიციაში გამომავალი რკალების გამოსახულებათა შესაბამისი მარკერები ჩაემატება. პოზიცია შეიძლება ერთდროულად შემავალი და გამომავალი იყოს (**მარყუჟი**)

გადასასვლელის ცვლადები (Transition Variables). რკალებისა და გადასასვლელის გახსნის პირობაში შემავალი ცვლადების ერთობლიობა.

რკალის ანოტაცია (Arc Annotation). გამოსახულება, რომელიც შეიძლება შეიცავდეს კონსტანტებს, ცვლადებს და ოპერატორებს რკალთან დაკავშირებული პოზიციის ტიპის მულტისიმრავლიდან.

მიღწევადი მარკირება (Reachable Marking). მარკირება, რომელიც მიიღება ქსელის საწყისი მარკირებიდან გადასასვლელთა გარკვეული მიმდევრობის გახსნის შემდეგ.

მიღწევად მარკირებათა სიმრავლე (Reachability Set). საწყისი მარკირებიდან მიღწევად მარკირებათა სიმრავლე თვით საწყისი მარკირების ჩათვლით.

ალგებრა (Algebra). მათემატიკური სტრუქტურა, რომელიც შეიცავს სიმრავლეთა სიმრავლეს და ფუნქციათა სიმრავლეს, რომლებიც ამ სიმრავლეთა დომენებსა და ქვედომენებზე მოქმედებს.

ტიპი (Sort). მონაცემთა სტრუქტურის სახელი.

არგუმენტის ტიპი (Argument Sort). ოპერატორის არგუმენტის ტიპი.

გამომავალი ტიპი (Output Sort). ოპერატორის შედეგის ტიპი.

არულობა (Arity) – ფუნქციაში შემავალი (არგუმენტები) და გამომავალი (შედეგი) ტიპები (მაგ., ბინარული, **n**-არული).

ტიპიზაცია (Typisation). ტიპის დაკავშირება პოზიციასთან.

აღწერები (**Declarations**). გამოსახულებათა სიმრავლე სიმრავლეთა, კონსტანტების, პარამეტრების მნიშვნელობათა,

ტიპიზებული ცვლადებისა და ფუნქციების განსაზღვრისათვის, რომლებიც მაღალი დონის პეტრის ქსელებზე აისახება.

ოპერატორი (Operator). სიმბოლოთა ერთობლიობა (აბრევიატურა) ფუნქციის სახელის წარმოსადგენად.

პარამეტრი (Parameter). მუდმივა (კონსტანტა), რომელიც სიმრავლეში განსაზღვრულ სიდიდეთა არეს შეიცავს.

მინიჭება (Assignment). მნიშვნელობის მინიჭება ცვლადების სიმრავლის კონკრეტული ცვლადისათვის.

სიგნატურა (Signature). ალგებრული სტრუქტურა, რომელიც ტიპების და ოპერატორების სიმრავლეებისგან შედგება.

ბულის სიგნატურა (Bool Signature). სიგნატურა, რომელიც ბულის (ლოგიკურ) ტიპს შეიცავს.

მრავალტიპური სიგნატურა (Many-sorted Signature). სიგნატურა, სადაც ტიპების სიმრავლის კარდინალურობა ერთზე მეტია.

ცვლადიანი სიგნატურა (Signature with Variables). სიგნატურა, რომელიც შეიცავს ცვლადების სახელებს, ტიპებს და ოპერატორებს.

თერმი (Term). სიგნატურის საფუძველზე შედგენილი გამოსახულება, რომელიც შეიცავს მუდმივებს, ცვლადებს და ოპერატორებს.

დახურული თერმი (Closed Term). თერმი, რომელიც შეიცავს კონსტანტებს და ოპერატორებს, მაგრამ არა ცვლადებს.

თერმის მნიშვნელობა (Term Evaluation). შედეგი, რომელიც მიიღება თერმის ცვლადებითვის მნიშვნელობების მინიჭებისა და ფუნქციათა შედეგების გამოთვლის შემდეგ.

მაღალი დონის პეტრის ქსელი (High Level Petri Net). ალგებრული სტრუქტურა, რომელიც შეიცავს: პოზიციების სიმრავლეს; გადასასვლელთა სიმრავლეს; ტიპების სიმრავლეს; ტიპების პოზიციებზე და ტიპების გადასასვლელებზე დამაკავშირებელ ფუნქციებს; პრეფუნქციებს შემავალი და პოსტფუნქციებს გამომავალი მარკირებების განსაზღვრისათვის; საწყის მარკირებას.

პეტრის ქსელის გრაფი (Petri Net Graph). მიმართული გრაფი ორი ტიპის კვანძებითა (პოზიციები და გადასასვლელები) და მათი დამაკავშირებელი რკალებით. დაშვებულია კავშირები

„პოზიცია-გადასასვლელი“ ან „გადასასვლელი-პოზიცია“, მაგრამ არა „პოზიცია-პოზიცია“ ან „გადასასვლელი-გადასასვლელი“.

მაღალი დონის პეტრის ქსელის გრაფი (High Level Petri Net Graph). ქსელის გრაფისა და ანოტაციების (წარწერების) ერთობლიობა, რომელიც შეიცავს პოზიციათა ტიპებს, რკალების ანოტაციებს, გადასასვლელთა გახსნის პირობებს, შესაბამის განსაზღვრებებს განსაზღვრებათა სიაში და ქსელის საწყის მარკირებას.

მიღწევადობის გრაფი (Reachibility Graph). მიმართული გრაფი, სადაც კვანძები მიღწევად მარკირებებს შეესაბამება, რკალები – გადასასვლელთა გახსნის ოპერაციას.

პარამეტრიზებული მაღალი დონის პეტრის ქსელის გრაფი (Parameterized High Level Petri Net Graph). მაღალი დონის პეტრის ქსელის გრაფი, რომელშიც პარამეტრები განისაზღვრება.

3. პეტრის ქსელების სემანტიკური მოდელი. მაღალი დონის პეტრის ქსელები

3.1. პეტრის ქსელი HLPN

პეტრის ქსელის სემანტიკური მოდელის აღწერის პროცესში გამოიყენება შემდეგი აბრევიატურები: **HLPN** – მაღალი დონის პეტრის ქსელი და **HLPNG** – მაღალი დონის პეტრის ქსელის გრაფი.

HLPN წარმოადგენს სტრუქტურას

$$\mathbf{HLPN = (P, T, D; Type, Pre, Post, M_0),}$$

სადაც:

- **P - პოზიციად** წოდებული ელემენტების სასრული სიმრავლეა;
- **T - გადასასვლელებად** წოდებული ელემენტების სასრული სიმრავლე, ისე, რომ $\mathbf{P \cup T = \varnothing;}$
- **D** – არაცარიელი **დომენების** სასრული სიმრავლე, რომლის ყოველ ელემენტს **ტიპი** ეწოდება;

- **Type : $P \cap T \in D$** წარმოადგენს ტიპების პოზიციებზე დაკავშირებისა და გადასასვლელის გახსნის რეჟიმის განსაზღვრის ფუნქციას;
- **Pre, Post : TRANS->μPLACE** წარმოადგენენ **წინასწარ** (გადასასვლელის გახსნამდე) და **შედეგის** (გადასასვლელის გახსნის შემდგომ) ასახვებს, სადაც
  - **TRANS = { (t,m) | t⊆T, m⊆Type(t) }**
  - **PLACE = { (p,g) | p⊆P, g⊆Type(p) }**
- **$M_0 \subseteq \mu PLACE$** მულტისიმრავლეა, რომელსაც **ქსელის საწყისი მარკირება** ეწოდება;
- **$M \subseteq \mu PLACE$** მულტისიმრავლეა, რომელსაც **ქსელის მარკირება** ეწოდება;

გადასასვლელის გაშვების რეჟიმების სასრული სიმრავლე, **$T_\mu \in \mu TRANS$** ნებადართულია **M**-მარკირებაში, თუ **$Pre(T_\mu) \leq M$,** სადაც **Pre**-ს წრფივი გაფართოება შემდეგი სახისაა:

$$Pre(T_\mu) = \Sigma_{tr \in TRANS}\, mult(tr,T_\mu)Pre(tr)$$

თუ გადასასვლელის გახსნის რეჟიმთა მულტისიმრავლე T_μ ნებადართულია **M** მარკირებაში, მაშინ გადასასვლელის გახსნის პროცედურას **ბიჯი** ეწოდება და მისი შესრულების შედეგად მიღებული ახალი მარკირება გამოისახება ფორმულით:

$$M' = M - Pre(T_\mu) + Post(T_\mu)$$

ბიჯის ფორმალური ასახვა შეიცავს საწყის და შედეგის მარკირებებს, აგრეთვე გადასასვლელთა გახსნის დაშვებული რეჟიმების მულტისიმრავლეს:

$$M \xrightarrow{T_\mu} M'$$

3.2. მაღალი დონის პეტრის ქსელის გრაფი – HLPNG

მაღალი დონის პეტრის ქსელის გრაფი წარმოადგენს სტრუქტურას:

HLPNG = (NG, Sig, H; Type, AN, M_0),

სადაც

- **NG = (P,T;F) ქსელის გრაფად** იწოდება, რომელშიც

- **P** კვანძების სასრული სიმრავლეა (**პოზიციები**);
- **T** - კვანძების სასრული სიმრავლე (**გადასასვლე-ლები**) და **P∪T=∅;**
- **F⊆(PxT)∩(TxP)** - **რკალებად** წოდებული მიმართული მონაკვეთების სიმრავლე**;**

- **Sig = (S, O, V)** წარმოადგენს გრაფის ნატურალურ-ლოგიკურ **სიგნატურას.**
- **H = (S_H, O_H)** სიგნატურისთვის განსაზღვრული მრავალსორტიანი **ალგებრაა;**
- **Type : P->S_H** ტიპების პოზიციებზე **დანიშვნის ფუნქციაა;**
- **AN = (A, TC)** ქსელის **ანოტაციათა** წყვილია, სადაც:
- **TC:T->TERM(O∩V)$_{Bool}$** წარმოადგენს **ფუნქციას**, რომელიც გადასასვლელებს ლოგიკური გამოსახუ-ლების ტიპის ანოტაციით აფართოებს;
- **M_0:P->$U_{p∈P}$μType(p),** ისე, რომ **∀p∈P, M_0(p)∈μType(p) საწყისი მარკირების ფუნქციაა**, რომელიც მარკერთა მულტისიმრავლეს ყოველი პოზიციის **ტიპთან** კორექტულად აკავშირებს.

გრაფიკულად **პოზიცია** წრეებით ან ელიფსებით გამოისახება. პოზიციის ანოტაცია (წარწერები) შედგება მინიმუმ პოზიციის სახელის, პოზიციასთან დაკავშირებული ტიპის სახელისა და საწყისი მარკირებისგან. თუ საწყისი მარკირება ცარიელია, იგი შეიძლება არ გამოისახოს.

გადასასვლელს მართკუთხედი ან შავი ხაზი გამოსახავს; გადასასვლელის ანოტაცია შედგება მინიმუმ მისი სახელისგან; თუ **გადასასვლელის გაშვების პირობა** მოცემულია, იგი მართკუთხედის შიგნით აისახება და გამოიტოვება მხოლოდ მაშინ, როცა ყოველთვის ჭეშმარიტია.

რკალს შეიძლება ჰქონდეს როგორც ერთ- ან ორმხრივმიმართული მონაკვეთის, ასევე მიმართული მრუდის სახე.

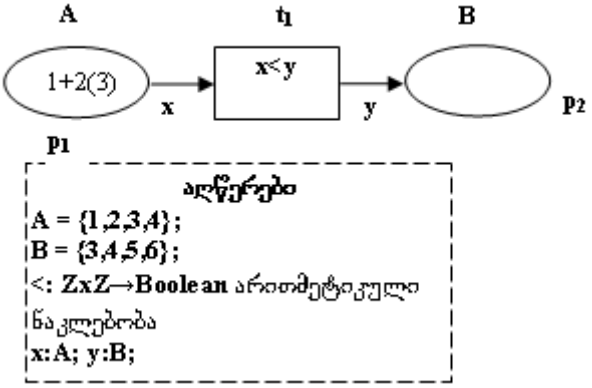
თუ ერთი და იგივე პოზიციისა და გადასასვლელის დამაკავშირებელ, შემავალ და გამომავალ რკალებს მსგავსი ანოტაციები აქვთ, ორმხრივმიმართული რკალების გამოსახვა ამ შემთხვევაში ერთი ორისრიანი რკალის სახითაც ნებადართულია.

რკალის ანოტაციად ზემოთ აღწერილი **თერმების მულტისიმრავლეები** გამოიყენება.

მარკერი პეტრის ქსელში პოზიციათა მარკირების ელემენტს წარმოადგენს და შესაბამისად, პოზიციის გვერდზე (ან შიგნით) გამოისახება მისი მარკირების ფარგლებში **სიმბოლური ჯამის** სახით.

ფრჩხილები გამოიყენება მიმდინარე მარკირებაში მარკერის **სიხშირისა** (ფრჩხილებს გარეთ) და მარკერის **მნიშვნელობის** (ფრჩხილებს შიგნით) გამოსაყოფად.

პეტრის ქსელების ზემოთმოყვანილი ნოტაციის მაგალითი მოცემულია 3.1 ნახაზზე.



ნახ.3.1. HLPN - გრაფი

მოცემულია 2 **პოზიცია (p_1 და p_2)**, 1 **გადასასვლელი (t_1)** და **დამაკავშირებელი რკალები**. აღიწერება 2 **ტიპი**, **A** და **B**, რომლებიც **საბაზო სიმრავლეებს** წარმოადგენს და ნატურალურ რიცხვთა სხვადასხვა ქვესიმრავლეებს შეიცავს. ცვლადი **x A**-ტიპისაა, **y – B**-ტიპის. გადასასვლელი შეიცავს **გახსნის პირობას x<y**, რისთვისაც აღწერების სიაში ,,ნაკლებობის'' **ოპერატორი** განისაზღვრება. რკალზე (p_1,t_1) დართულია **ანოტაცია** – ცვლადი **x**, ხოლო რკალზე (t_1,p_2) **ანოტაცია** – ცვლადი **y**.

პოზიცია **p_1 ტიპიზებულია A**-ტიპით და გააჩნია **საწყისი მარკირება 1+2(3),** რომელიც წარმოადგენს მულტისიმრავლეს **$M_0(p_1)$={(1,1),(2,0),(3,2),(4,0)},** სადაც ყოველი წყვილის პირველი

ელემენტი ნატურალური რიცხვია **A={1,2,3,4}** საბაზო სიმრავლიდან, ხოლო მეორე – მისი **სიხშირე** მულტისიმრავლეში. პოზიცია $\mathbf{p_2}$ **ტიპიზებულია B**-ტიპით და მისი **საწყისი მარკირება** ცარიელ მულტისიმრავლეს წარმოადგენს $\mathbf{M_0(p_2)=\{\varnothing\}}$.

საწყისი მარკირებიდან გამომდინარე, გადასასვლელი $\mathbf{t_1}$ **ნებადართულია** გადასასვლელის გახსნის შემდეგ **რეჟიმებში: {(1,3), (1,4), (1,5), (1,6), (3,4), (3,5), (3,5)}**, სადაც ყოველი წყვილის პირველი ელემენტი **x**-ცვლადის მნიშვნელობაა, ხოლო მეორე – **y**-ცვლადისა, ისე, რომ ყოველი წყვილისთვის **x<y**.

მოცემულ ქსელში გაშვების რეჟიმების **პარალელიზმიც** ფიქსირდება. მაგალითად, რეჟიმების მულტისიმრავლე **(1,3)+2(3,5)** $\mathbf{t_1}$ გადასასვლელს პარალელურად ნებადართულს ხდის. შეიძლება ასეთი მაგალითის განხილვაც: **(1,5)+(3,4); (1,6)+(3,5)+(3,6)**.

თუ გაიხსნება **ნებადართული მარკირება**, მაგალითად, **(1,3) რეჟიმთა მულტისიმრავლეში** (რომელიც ამ შემთხვევაში მხოლოდ 1 ტიპის 1 ელემენტისგან შედგება), მაშინ **შედეგის მარკირებებს** შემდეგი სახე ექნება:

$\mathbf{M(p_1)=\{(1,0),(2,0),(3,2),(4,0)\}}$;

$\mathbf{M(p_2)=\{(3,1),(4,0),(5,0),(6,0)\}}$;

(1,3)+2(3,5) რეჟიმთა მულტისიმრავლეში გადასასვლელის გახსნა შემდეგ **შედეგის მარკირებებს** ჩამოაყალიბებს:

$\mathbf{M(p_1)=\{\varnothing\}}$;

$\mathbf{M(p_2)=\{(3,1),(4,0),(5,2),(6,0)\}}$;

4. პეტრის ქსელების კლასიფიკაცია

4.1 პეტრის ქსელების ქვეკლასები

პეტრის ქსელების ევოლუციის პირველ ეტაპზე მათი 3 თეორიული კლასი განისაზღვრა (I, II და III დონის პეტრის ქსელები), მაგრამ დღეისათვის მათი რაოდენობა ორამდე ჩამოვიდა - განისაზღვრება **დაბალი** და **მაღალი** დონის პეტრის ქსელები, ამასთან მეორე კლასი პირველს მოიცავს.

ძველი კლასიფიკაცია პოზიციებზე, გადასასვლელებ-სა და რკალებზე იყო ორიენტირებული და განასხვავებდა მათ ისეთ

მახასიათებლებს, როგორიცაა მარკერთა მაქსიმალური რაოდენობა პოზიციაში, რკალების ჯერადობა და სხვა.

ახალ კლასიფიკაციაში ყურადღება უშუალოდ მარკერთა სემანტიკაზეა გამახვილებული.

კერძოდ, დაბალი დონის პეტრის ქსელებში დაიშვება მხოლოდ **"შავი"** მარკერები ყოველგვარი შინაგანი სტრუქტურის გარეშე, ხოლო მაღალი დონის პეტრის ქსელები დამატებით წინასწარ განსაზღვრული სტრუქტურის **"ფერად"** მარკერებსაც შეიცავს, თუმცა უნდა აღინიშნოს, რომ ტერმინები "შავი" და "ფერადი" სიმბოლურია და ლიტერატურაში მათ ხშირად განსხვავებული სახელებით მოიხსენიებენ.

მარკერთა არსი განსაზღვრავს შემდგომ პოზიციებისა და გადასასვლელების, აგრეთვე რკალების ანოტაციის შიგთავსს. "შავმარკერიან" ქსელებში მხოლოდ არატიპიზებული, ერთგვაროვანი პოზიციები და გადასასვლელებია დაშვებული, "ფერად" ქსელებში ყველა პოზიციისთვის საკუთარი ტიპი განისაზღვრება შესაბამისი ტიპის მარკერთა დომენით. ერთი ტიპის პოზიციაში მეორე ტიპის მარკერის არსებობა დაუშვებელია.

გადასასვლელები ფართოვდება გადასასვლელის გაშვების პირობებით, რომელიც ლოგიკურ გამოსახულებას წარმოადგენს და შეიძლება ჭეშმარიტი ან მცდარი იყოს.

რკალის ანოტაცია დაბალი დონის პეტრის ქსელში ან საერთოდ გამოიტოვება (რკალში ერთ გაშვებაზე მხოლოდ ერთი "შავი" მარკერი გადაადგილდება) ან ნატურალურ რიცხვს წარმოადგენს, რომელიც გადასაადგილებელ მარკერთა რაოდენობას ასახავს (რკალის ჯერადობა), მაშინ, როცა მაღალი დონის პეტრის ქსელში რკალის ანოტაცია შეიძლება შეიცავდეს უფრო რთულ მონაცემებსაც, რომლებიც ქვემოთ განიხილება.

დაბალი დონის პეტრის ქსელების ქვეკლასებიდან შეიძლება დავასახელოთ ავტომატური პეტრის ქსელები, მარკირებული გრაფები, პეტრის ქსელები თავისუფალი არჩევანით, ელემენტარული სისტემური ქსელები, C/E-ქსელები, უსაფრთხო S/T ქსელები, S/T (კლასიკური) ქსელები და სხვა, ხოლო მაღალი დონის პეტრის ქსელების ყველაზე კარგად გამოკვლეულ და განსაზღვრულ ქვეკლასს **სისტემური პეტრის ქსელები** წარმოადგენს.

ავტომატურ პეტრის ქსელებში ანუ მდგომარეობათა მანქანებში **(State Machines)** ყოველ გადასასვლელს შეიძლება ჰქონდეს მაქსიმუმ 1 შესასვლელი და 1 გამოსასვლელი. იგი **მკაცრად შენახვადი** პეტრის ქსელია (მარკერების საერთო რაოდენობა მასში არასდროს იცვლება). ავტომატური პეტრის ქსელებით შეიძლება **კონფლიქტების**, მაგრამ არა **პარალელიზმის** მოდელირება.

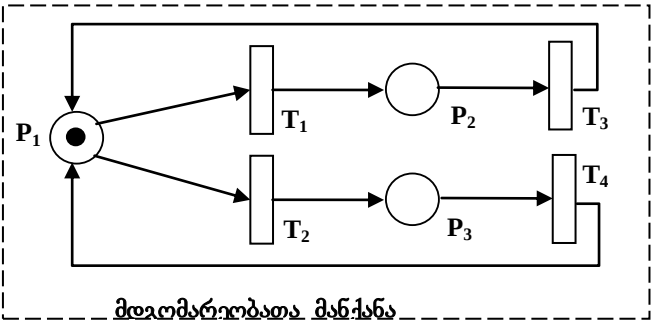
მარკირებულ გრაფებში (Marked Graphs) ყოველი პოზიცია ზუსტად 1 გადასასვლელის შესასვლელს და ზუსტად 1 გადასასვლელის გამოსასვლელს წარმოადგენს. იგი თეორიულად ავტომატური პეტრის ქსელების ორეულია, ამოდელირებს პარალელიზმს, მაგრამ კონფლიქტებს - ვერა.

მარკირებულ გრაფებში არსებობს **ციკლები** – შეკრული (ჩაკეტილი) გზა რომელიმე გადასასვლელიდან იმავე გადასასვლელამდე, რომელიც გადასასვლელთა გარკვეული მიმდევრობის გახსნით მიიღება. ციკლის გაშვების შედეგად მარკირებულ გრაფში მარკერების საერთო რაოდენობა არ იცვლება, თუმცა, ზოგადად, მარკირებული გრაფი შენახვადი არ არის (მასში მერკერების მთლიანი რაოდენობა შეიძლება იცვლებოდეს).

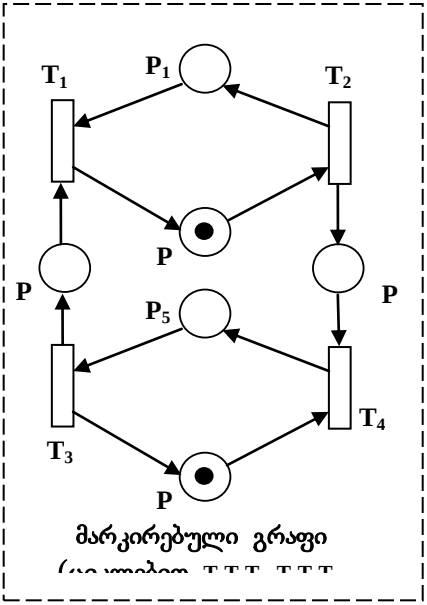
პეტრის ქსელებში თავისუფალი არჩევანით (Free Choise Petri Nets) მართვადი კონფლიქტის ცნება შემოდის: თუ რამდენიმე გადასასვლელს შემავალი პოზიციისთვის **კონფლიქტი** აქვს, პეტრის ქსელში თავისუფალი არჩევანით ისინი ყველა ნებადართული უნდა იყოს, ანუ საკონფლიქტე პოზიცია ერთადერთი შემავალი პოზიცია უნდა იყოს ყველა მოკონფლიქტე გადასასვლელისთვის.

ზემოთ აღწერილი 3 ქვეკლასის პეტრის ქსელების ნიმუშები მოცემულია 4.1 ნახაზზე.

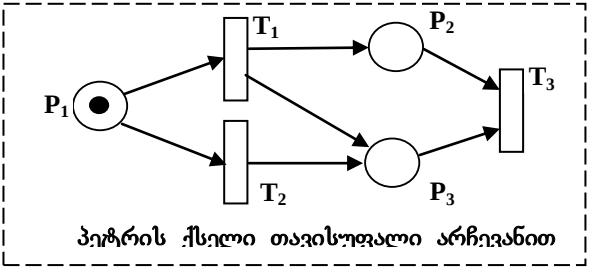
S/T-ქსელები (**State/Transition Nets**) **კლასიკური** პეტრის ქსელების წარმომადგენელია. იგი შედგება მსგავსი მარკერებისგან, რომელთა გრაფიკული ფორმა პატარა შავი წრეა პოზიციის ფარგლებში.



მდგომარეობათა მანქანა



მარკირებული გრაფი

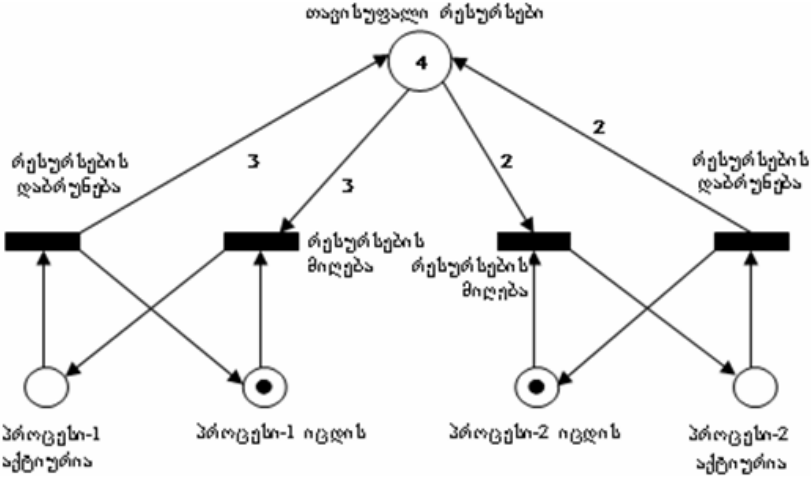


პეტრის ქსელი თავისუფალი არჩევანით

ნახ.4.1. დაბალი დონის პეტრის ქსელების 3

S/T-ქსელებში პოზიცია შეიძლება ერთზე მეტ მარკერს შეიცავდეს, ხოლო მარკერების დიდი ოდენობის შემთხვევაში პოზიციაში მათი რაოდენობა რიცხობრივად ჩაიწერება.

გადასასვლელის გაშვების აუცილებელი პირობაა ყველა შემავალ პოზიციაში დამაკავშირებელი რკალის ჯერადობაზე არანაკლები ოდენობის მარკერების მოგროვება. რკალების ჯერადობა ნატურალური რიცხვით გამოისახება (ნახ.4.2).



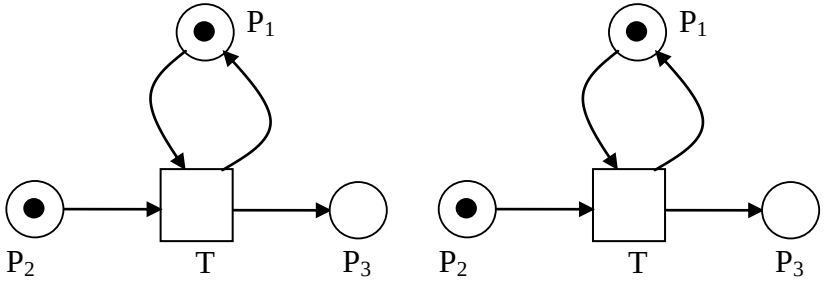
ნახ.4.2. კლასიკური პეტრის ქსელი „რესურსების განაწილების“ ამოცანისთვის

უსაფრთხო S/T-ქსელების პოზიციებში მარკერთა ოდენობა 1-ს არ უნდა აღემატებოდეს.

ელემენტარული სისტემური ქსელები (**Elementary System Nets**) **S/T**-ქსელების მსგავსად დაბალი დონის პეტრის ქსელების კლასში შედის, როგორც ყველა სხვა ქსელი, რომლებშიც მხოლოდ ერთგვაროვანი მარკერებია დასაშვები. სხვაობა ისაა, რომ ელემენტარულ სისტემურ ქსელებში ერთი პოზიცია შეიძლება არაუმეტეს ერთ მარკერს შეიცავდეს. შესაბამისად, მასში ჯერადი რკალები არ არსებობს და რკალი ჭდეს (წარწერას) არ საჭიროებს. ამასთან, ელემენტარულ ქსელურ სისტემებში აკრძალულია გადასასვლელის გაშვება, თუ ერთი მაინც გამომავალი პოზიცია მარკერს უკვე შეიცავს [15].

მისგან განსხვავებით უსაფრთხო **S/T**-ქსელებისთვის მნიშვნელობა არა აქვს გამომავალი პოზიციების მდგომარებას, მათში გადასასვლელი გაიშვება, თუ ყველა შემავალ პოზიციაში მარკერი იქნება. ამასთან, უსაფრთხო **S/T**-ქსელები კრძალავენ პოზიციაში ერთზე მეტი მარკერის არსებობას [18]. აქედან გამოდინარე, უსაფრთხო პეტრის ქსელში მარყუჟის არსებობა გადასასვლელის გახსნას ხელს ვერ უშლის, ელემენტარულ სისტემურ ქსელში კი პირიქით.

4.3 ნახაზზე გრაფიკულად სრულებით მსგავსი ელემეტარული სისტემური ქსელი და **S/T**-ქსელი განსხვავდება შესრულების მანერით: პირველში გადასასვლელი **T** ვერ გაიშვება, მეორეში იგი ნებადართულია.



ელემენტარული სისტემური ქსელი

უსაფრთხო **S/T**-ქსელი

ნახ.4.3. გრაფიკულად მსგავსი და შესრულების წესებით განსხვავებული პეტრის ქსელები

4.2. სისტემური პეტრის ქსელები

System Petri Nets - ყველა ზემოაღწერილი ქვეკლასისგან განსხვავებით, მაღალი დონის პეტრის ქსელის ქვეკლასს წარმოადგენს.

მაღალი დონის პეტრის ქსელების სტანდარტული ნოტაციის თანახმად, სისტემური პეტრის ქსელებისთვის განისაზღვრება **კონსტანტები, ცვლადები** და **ფუნქციები**, რომელთა ერთობლიობას სისტემური პეტრის ქსელის **სტრუქტურა** ეწოდება, ხოლო გადასასვლელებისთვის განისაზღვრება გახსნის პირობა, რომელსაც "**გადასასვლელის დამცავი ფუნქცია**" ეწოდება [**15**]. სისტემური პეტრის ქსელების ძირითადი განსაზღვრებები შემდეგია:

განსაზღვრება–1. ვთქვათ **Σ** ქსელია. **Σ**-ს **A** უნივერსუმი თითოეული $\mathbf{p}\in\mathbf{P_\Sigma}$ პოზიციისთვის აფიქსირებს მდგომარეობათა $\mathbf{A_p}$ სიმრავლეს, რომელსაც **A**-ში **p**-ს **დომენი** ეწოდება.

განსაზღვრება–2. ვთქვათ **Σ** ქსელია **A** უნივერსუმით, მაშინ

1. **Σ**-ს **მდგომარეობა a** თითოეული პოზიციისთვის განსაზღვრავს სიმრავლეს $\mathbf{a(p)\subseteq A_p}$;
2. ვთქვათ $\mathbf{t\in T_\Sigma}$. **ქმედება m** თითოეული მოსაზღვრე **f=(p,t)** ან **f=(t,p)** რკალისთვის განსაზღვრავს სიმრავლეს $\mathbf{m(f)\subseteq A_p}$.

ეს ნიშნავს, რომ სისტემური ქსელების პოზიციებში ცვლადის მნიშვნელობათა სიმრავლის მოთავსება შეიძლება, რკალსაც ელემენტარული პეტრის ქსელებისგან განსხვავებით ერთზე მეტი მნიშვნელობის გატარება შეუძლია, რაც მაღალი დონის პეტრის ქსელებს ახასიათებს.

სისტემური პეტრის ქსელების შინაარსი სტრუქტურების ცნებაზეა დაფუძნებული. განვსაზღვროთ თავიდან კონსტანტისა და ფუნქციის ცნებები.

განსაზღვრება–3. დავუშვათ $\mathbf{A_1,\ldots,A_k}$ არის ქვესიმრავლეთა სიმრავლე.

1. დავუშვათ $\mathbf{a\in A_i}$, რომელიღაც $\mathbf{1<i<k}$ - თვის. მაშინ **a**-ს ეწოდება **კონსტანტა** სიმრავლეში $\mathbf{A_1,\ldots,A_k}$ და $\mathbf{A_i}$-ს ეწოდება **a**-ს **კლასი**.

2. **i=1,…,n+1** - თვის დავუშვათ $\mathbf{B_i \in \{A_1,\ldots,A_k\}}$, და ვთქვათ, $\mathbf{f:B_1\times\ldots\times B_n \to B_{n+1}}$ არის ფუნქცია. მაშინ **f**-ს ეწოდება ფუნქცია $\mathbf{A_1,\ldots,A_k}$ სიმრავლეებზე. სიმრავლეები $\mathbf{B_1,\ldots,B_n}$ წარმოადგენს **f**-ის არგუმენტების ტიპებს, ხოლო $\mathbf{B_{n+1}}$ შედეგის ტიპს. **n+1** კორტეჟი $\mathbf{(B_1,\ldots,B_{n+1})}$ წარმოადგენს **f**-ის **არეს** და შემდეგნაირად ჩაიწერება $\mathbf{B_1\times\ldots\times B_n \to B_{n+1}}$.

სისტემური პეტრის ქსელის **სტრუქტურა** კონსტანტებისა და ფუნქციების სიმრავლეს ეწოდება.

განსაზღვრება–4. დავუშვათ $\mathbf{A_1,\ldots,A_k}$ არის ქვესიმრავლეთა სიმრავლე, $\mathbf{a_1,\ldots,a_l}$ კონსტანტები $\mathbf{A_1,\ldots,A_k}$-ში და $\mathbf{f_1,\ldots,f_m}$ ფუნქციები $\mathbf{A_1,\ldots,A_k}$-ზე. მაშინ $\mathbf{A=(A_1,\ldots,A_k;a_1,\ldots,a_l;f_1,\ldots,f_m)}$ წარმოადგენს **სტრუქტურას**. $\mathbf{A_1,\ldots,A_k}$ **მატარებელი სიმრავლეებია**, $\mathbf{a_1,\ldots,a_l}$ - **კონსტანტები**, ხოლო $\mathbf{f_1,\ldots,f_m}$ - **ფუნქციები**.

სტრუქტურების ფუნქციათა შემადგენლობა **თერმების** ცნებით აღიწერება. თერმები შეიცავს **ცვლადებს** და **კონსტანტებს** და ისევე როგორც ცალკეული ცვლადები, კონკრეტულ მომენტში კონკრეტულ მნიშვნელობას ღებულობს.

თერმებისა და ცვლადების უკეთ წარმოსადგენად განვიხილოთ "მოთხოვნათა მომსახურების" სისტემა. იგი შედგება მონაცემთა 3 მომხმარებლისაგან **(u,v,w),** რომელთა ციკლურ მომსახურებას აწარმოებს მონაცემთა 2 მენეჯერი (**m** და **n**).

საწყის მდგომარეობაში თითოეული მომხმარებელი ლოკალურ მდგომარეობაშია, ხოლო გარკვეულ მომენტში მომხმარებელი მოითხოვს მონაცემებს ორივე მენეჯერისგან და მხოლოდ მას შემდეგ, რაც ორივესგან დაკმაყოფილდება, უბრუნდება ლოკალურ მდგომარეობას.

მომსახურებას პირველად **u** ღებულობს, მერე **v** და ბოლოს **w**. 4.4 ნახაზზე წარმოდგენილია სისტემური პეტრის ქსელი მოცემული სისტემისათვის.

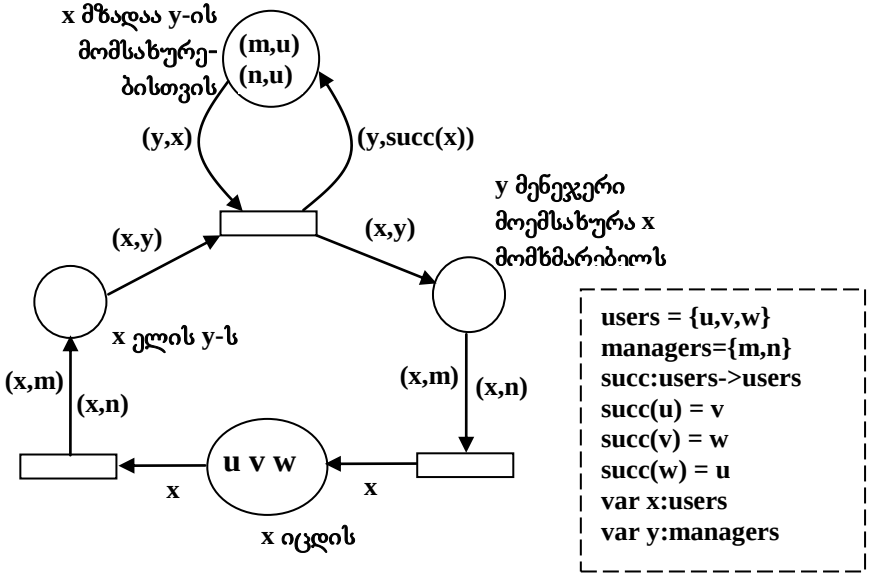
ამ შემთხვევაში პეტრის ქსელის სტრუქტურას შემდეგი სახე ექნება:

"მოთხოვნათა მომსახურება"=(users,managers,

users×managers, managers×users,u,v,w,m,n,succ)

სადაც **succ** (ინგლისური სიტყვიდან **success** - "წარმატება") მოთხოვნის წარმატებით შესრულების მაუწყებელი ფუნქციაა, მისი

არგუმენტი **x** ცვლადია, რომელიც **users** ტიპისაა და შესაბამისად, მნიშვნელობებს მხოლოდ **{u,v,w}** სიმრავლიდან იღებს, ხოლო თავად **succ**-ის მნიშვნელობა რიგში მომდევნო ადგილას მდგარი მომხმარებელია, რომელიც ფუნქციის მეშვეობით მენეჯერთან მიმართვის უფლებას ღებულობს.



ნახ.4.4. სისტემური პეტრის ქსელი „მოთხოვნათა მომსახურების ამოცანისთვის“

სისტემურ ქსელზე რკალთა ზოგიერთი წარწერა თერმია, მაგალითად, **(x,m)** და იგი შეიცავს ცვლადსაც **(x)** და კონსტანტასაც **(m)**, როგორც ზემოთ აღვნიშნეთ.

სისტემური ქსელის მუშაობის პირველ ბიჯზე, ამოცანის პირობის თანახმად, მომსახურებას ღებულობს **u** მომხმარებელი, დავუშვათ **m** მენეჯერისგან. მაშინ თერმი **(y,x)=(m,u),** თერმი **(x,m)=(u,m)** და ცვლადი **x=u**.

თუ **x**-ს სხვა მნიშვნელობას მივანიჭებთ, მაგალითად **v**-ს, გადასასვლელი „მომსახურების მოთხოვნა" კი გაიხსნება, მაგრამ თავად „მომსახურება" - ვერა, რადგან ამ დროს **(y,x)** თერმის

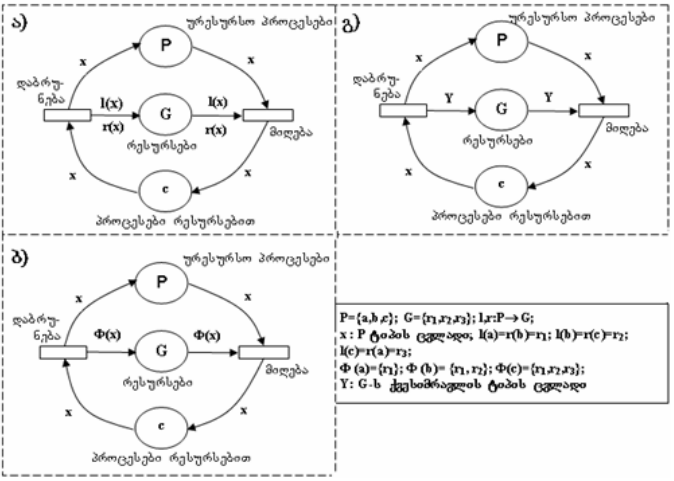
მნიშვნელობა **(m,v)** უნდა გახდეს, რის საშუალებასაც პოზიციის („**y** მზადაა **x**-ის მომსახურებისთვის") საწყისი შემცველობა არ იძლევა და შესაბამისად, ქსელი არ იმუშავებს.

უფრო რთული შემთხვევებისთვის თერმების სიმრავლე შემოიტანება (**სიმრავლე-თერმები**), რომელიც სამი ტიპისაა: **კონსტანტა-თერმი, ფუნქცია-თერმი** და **ცვლადი-თერმი**.

კონსტანტა-თერმი ქსელში არა ცალკეული კონსტანტების, არამედ კონსტანტების (მაგალითად, ნატურალური რიცხვების) სიმრავლეთა ასახვისთვის გამოიყენება და ამ შემთხვევისთვის პოზიციის აღმნიშვნელი **ჭდე** არა კონსტანტების სიმრავლეს, არამედ სიმრავლის აღმნიშვნელ სიმბოლოს (დიდ ლათინურ ასოს) წარმოადგენს, ხოლო თავად სიმრავლე ქსელის სტრუქტურის განსაზღვრათა ბლოკში აღიწერება.

ფუნქცია-თერმი რკალის ანოტაციაა, აღინიშნება **Φ**-სიმბოლოთი და პეტრის ქსელში კონსტანტების ცვლადი რაოდენობის ტრანსპორტირებას ემსახურება არგუმენტ-ცვლადის მნიშვნელობის შესაბამისად.

ცვლადი-თერმი ასევე რკალის ანოტაციას წარმოადგენს და ქსელში მთლიანად სიმრავლის ან მის ქვესიმრავლეთა გადასატანად გამოიყენება. თერმების სიმრავლეთა ტიპების მაგალითები მოცემულია 4.5 ნახაზზე.



ნახ.4.4.5. სიმრავლე-თერმები

4.3. მაღალი დონის პეტრის ქსელების ტიპები

პეტრის ქსელების ქვეკლასიდან განისაზღვრება **პეტრის ქსელის ტიპი**. ახალი ტიპის განსაზღვრის არე შეიძლება მრავალნაირი იყოს. ყოველ პეტრის ქსელს გააჩნია შემდეგი საერთო ელემენტები:

- **პოზიციები,**
- **გადასასვლელები**
- **რკალები**,

რომლებითაც პეტრის ქსელის გრაფი იქმნება.

პეტრის ქსელის ახალი **ტიპის** განსაზღვრისას საფუძვლად სწორედ პეტრის ქსელის გრაფია აღებული და იგი შემდგომი ასახვებითა და ფუნქციებით პეტრის ქსელის კონკრეტულ ტიპამდე ფართოვდება.

პეტრის ქსელის სხვადასხვა ტიპები ერთმანეთისგან შეიძლება განსხვავდებოდეს **მარკერთა ტიპებით და მათგან გამომდინარე ერთიანი მარკირების სისტემით, ქსელის ელემენტების აღწერით (ჭდეები) ან/და გადასასვლელთა გაშვების წესებით**.

ჭდეები პეტრის ქსელის ელემენტებზე, ძირითადად მხოლოდ წარწერებია და შეიცავს **ელემენტის სინტაქსს, მაგრამ არა სემანტიკას**. შესაბამისად, ისინი ქსელის შესრულების პროცესში ვერაფერს ცვლის. ჭდეების დანიშნულება პეტრის ქსელის სინტაქსური კონტროლია.

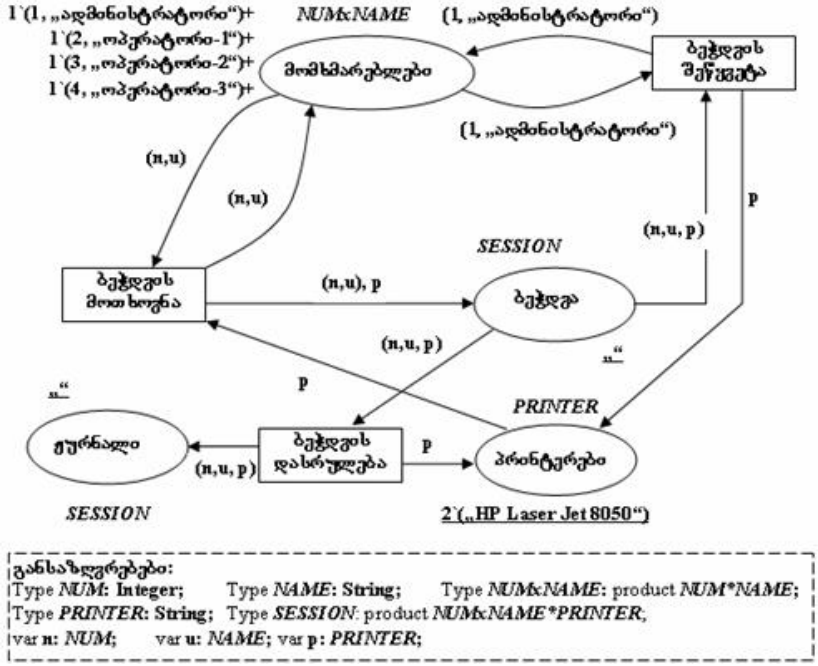
ამის მიუხედავად, ახალი ტიპის ჭდის განსაზღვრა უკვე საკმარისია იმისთვის, რომ ახალი პეტრის ქსელის ტიპი იქნეს განსაზღვრული.

პოზიციებზე, გადასასვლელებზე ან/და რკალებზე დროითი დაყოვნების განსაზღვრას დროითი პეტრის ქსელის ტიპი შემოაქვს, დაყოვნების დროთა ალბათურ განაწილებას – სტოქასტური პეტრის ქსელის ტიპი და ასე შემდეგ.

პეტრის ქსელების ტიპების განსაზღვრის უფრო ფართო გარემო მაღალი დონის პეტრის ქსელების ქვეკლასებია (მაგალითად, სისტემური პეტრის ქსელები). ქვემოთ მოკლედ

აღვწერთ ყველაზე კარგად დამუშავებულ და გავრცელებულ პეტრის ქსელის ტიპებს.

ფერადი პეტრის ქსელები (**Coloured Petri Nets**) მაღალი დონის პეტრის ქსელების კლასში შედის და სხვადასხვა ტიპის ანუ ფერის მარკერებს შეიცავს [7]. ტერმინი „ფერადი“ მხოლოდ ტრადიციისთვის გამოიყენება და ქსელში განსხვავებული მარკერების არსებობაზე მიანიშნებს - ამგვარი ქსელების დაბალი დონის პეტრის ქსელებისგან გამოსარჩევად, რომლებიც ერთგვაროვან, „შავ“ მარკერებს შეიცავს. ფერადი პეტრის ქსელის სტრუქტურა საკმაოდ რთულია და იგი მრავალი სახეობის ჭდეებს შეიცავს (ნახ.4.6).



ნახ.4.6. ფერადი პეტრის ქსელი „ბეჭდვის მოთხოვნათა მომსახურების“ ამოცანისთვის

ფერადი პეტრის ქსელის ყოველ პოზიციას გააჩნია მინიმუმ ორი ჭდე: **სახელი**, რომელიც აღმნიშვნელი წრის ან ელიფსის შიგნით იწერება და მარტივი ან შედგენილი **ტიპი** (პოზიციის გვერდით, კურსივით, გასაღებური სიტყვა **Type** ან **Color**). მაგალითად, პოზიცია „**მომხმარებლები**“ ***NUMxNAME*** ტიპისაა, რომელიც წინასწარგანსაზღვრული ***NUM*** და ***NAME*** ტიპების დეკარტული ნამრავლით წარმოიქმნება.

ფერადი პეტრის ქსელი შეიცავს „ფერად“ მარკერებს, რომლებიც კონკრეტული ტიპის შესაძლო მნიშვნელობათა სიმრავლეს ან მულტისიმრავლეს (კომპლექტს) წარმოადგენს.

განისაზღვრება კონსტანტები (გასაღებური სიტყვა **val**), ცვლადები (**var**) და ფუნქციები (**fun**). სხვადასხვა ტიპის მონაცემთა შორის კავშირების ასახვისთვის გამოიყენება სიმრავლეთა და კომპლექტების თეორიის ელემენტები.

გარდა მონაცემთა ტიპისა, ყოველი პოზიციის გვერდით შეიძლება აისახოს პოზიციაში მოცემულ მომენტში შემავალი ფერადი მარკერები.

საინიციალიზაციო მარკირება ხაზგასმული ტექსტის სახით გამოიტანება.

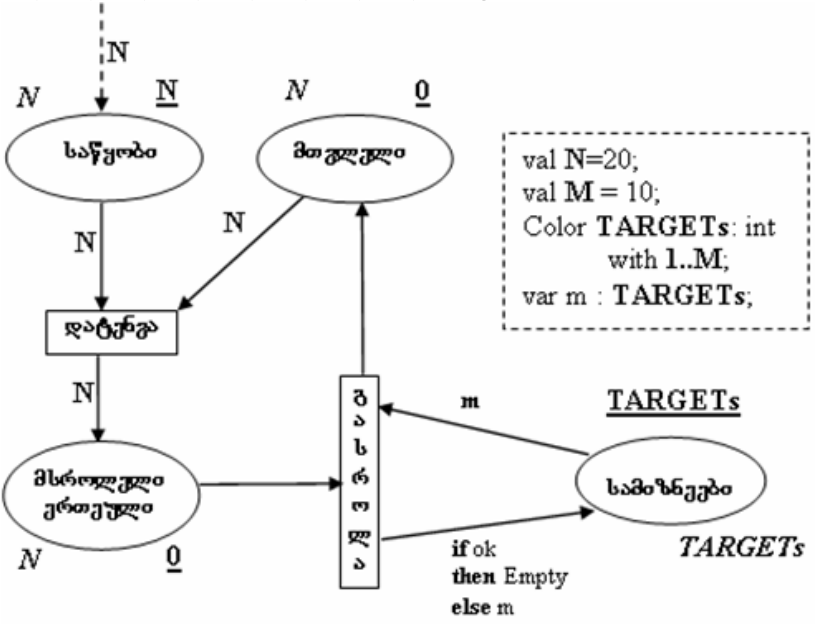
მაგალითად, საწყის მდგომარეობაში პოზიცია „მომხმარებლები“ შეიცავს ***NUMxNAME*** ტიპის ფერად მარკერთა შემდეგ სიმრავლეს (**საინიციალიზაციო მარკირება**): **{ 1`(1, „ადმინისტრატორი“), 1`(2, „ოპერატორი-1“), 1`(3, „ოპერატორი-2“), 1`(4, „ოპერატორი-3“) }**

საყურადღებოა საწყისი „1“-იანი ყოველი ელემენტის დასაწყისში (მას **კოეფიციენტი** ეწოდება), რომელიც უთითებს, რომ პოზიცია არაუმეტეს 1 მოცემულ მონაცემს შეიცავს (ანუ არსებობს მხოლოდ ერთი მომხმარებელი სახელით „ადმინისტრატორი“, რომლის რიგითი ნომერია 1). ამ შემთხვევაში გვაქვს მონაცემთა ელემენტების სიმრავლე, ხოლო **2`(„HP Laser Jet 8050“)** ნიშნავს, რომ სისტემა შეიცავს 2 ერთგვაროვან პრინტერს – ეს უკვე მულტისიმრავლე, ანუ კომპლექტია.

თუ საინიციალიზაციო მარკირების ტექსტი ძალიან გრძელია (მაგალითად მონაცემთა ბაზის ათასობით ჩანაწერს გამოსახავს), მარკერს ფსევდონიმი განესაზღვრება, რომელიც ტრადიციულად თანხვდება მარკერის ტიპის სახელს, მაგრამ იწერება ხაზგასმით.

4.7 ნახაზზე ფსევდონიმი **TARGETs** ასახავს სამიზნეთა **1..M** სიმრავლეს, ანუ ამ შემთხვევაში ნატურალურ რიცხვთა (სამიზნეთა ნომრები) სიმრავლეს 1-დან 10-მდე.

TARGETs–ის გაშლილი ვარიანტი იქნება სიმრავლე **{1`1, 1`2, 1`3, 1`4, 1`5, 1`6, 1`7, 1`8, 1`9, 1`10}**



ნახ.4.7. არადეტერმინირებული ფერადი პეტრის ქსელი „მიზანში სროლის“ ამოცანისთვის

ამავე ნახაზზეა ასახული **არადეტერმინირებული ლოგიკური გამოსახულება (პირობის ბლოკი)** ფერადი პეტრის ქსელის რკალებზე, რომელიც გადასასვლელთა გაშვების სხვადასხვა პირობებს და შედეგებს ასახავს, ანუ ლოგიკური პირობის ჭეშმარიტებისას გადასავლელს გასხვავებული მნიშვნელობა მიეწოდება (ან გადასასვლელიდან განსხვავებული მნიშვნელობა გამოვა), მცდარობისას – განსხვავებული.

ლოგიკური პირობის მნიშვნელობა სხვადასხვა შემთხვევებში სხვადასხვანაირად განისაზღვრება.

ინტერაქტიულ სიმულატორებში ჭეშმარიტება-მცდარობას თავად მომხმარებელი განსაზღვრავს, ავტომატური სიმულაციისას – შემთხვევით სიდიდეთა გენერატორი და ასე შემდეგ.

ფერადი პეტრის ქსელებში ყველაზე კარგადაა შერწყმული პეტრის ქსელებისა და დაპროგრამების (იერარქიულობა, მოდულურობა – დიდი სისტემების მოდელირებისთვის) თეორია, რაც თეორიულთან ერთად მის დიდ პრაქტიკულ ღირებულებასაც განაპირობებს თანამედროვე ინფორმაციულ ტექნოლოგიათა მრავალ სფეროში, როგორებიცაა საკომუნიკაციო პროტოკოლები, აუდიო/ვიდეო- და ოპერაციული სისტემები, აპარატურის და პროგრამების დაპროექტება, ბიზნეს-პროცესები და სხვა.

დღეისათვის ფერადი პეტრის ქსელები და მისი კომპიუტერული რეალიზაცია **„CPN-Tools“** მსოფლიოს 40 ქვეყნის 400-ზე მეტ ორგანიზაციაში გამოიყენება სისტემების მოდელირების ინსტრუმენტად (მათგან 100-მდე კომერციულ კომპანიაში) [9]. წინამდებარე სადისერტაციო სამუშაოს რიგი ექსპერიმენტები სწორედ ამ ინსტრუმენტის ლიცენზირებული ვერსიით იქნა შესრულებული.

4.4. დროითი პეტრის ქსელები

დროითი პეტრის ქსელები ფაქტობრივად ყოველი ტიპის პეტრის ქსელისთვის დროითი გაფართოების დამატებით მიიღება. დროითი გაფართოება აუცილებელია რეალური საპრობლემო სფეროს მოდელირებისთვის, მის გარეშე პეტრის ქსელი მხოლოდ სისტემის რაოდენობრივი ანალიზისთვის გამოდგება.

დროითი პეტრის ქსელი 4 ტიპის არსებობს: **პოზიციურ-დროითი** (**Timed Places Petri Nets - TPPNs**), **ტრანზაქციულ-დროითი** (**Timed Transition Petri Nets - TTPNs**), **რკალურ-დროითი** და **მარკერულ-დროითი** [11].

პოზიციურ-დროითი ტიპისთვის განისაზღვრება დაყოვნების ერთი და იგივე დრო პოზიციაში მოთავსებული ყველა მარკერისთვის და დროის ათვლა იწყება შესაბამისი გადასასვლელის გააქტიურებისთანავე (როცა მისი გახსნა

ნებადართული ხდება). ყველა შემავალი პოზიციის დაყოვნების დროის გასვლის შემდეგ გადასასვლელი გაიხსნება.

ტრანზაქციულ-დროით პეტრის ქსელებში დაყოვნების დრო გადასასვლელისთვის (ტრანზაქციის-თვის) განისაზღვრება. პეტრის ქსელების ეს ტიპი 2 ქვეტიპს შეიცავს: **წინასწარი არჩევანისა** და **შეჯიბრების** მოდელებს.

წინასწარი არჩევანის შემთხვევაში გადასასვლელი გააქტიურებისთანავე იღებს მონოპოლურ უფლებას ყველა შემავალ პოზიციაში მოთავსებულ მარკერების იმ ოდენობაზე, რაც მისი გახსნისთვის აუცილებელია (სხვა პოზიციებთან კონფლიქტში იმარჯვებს). ამის შემდგომ იწყება დაყოვნების დროის ათვლა. მისი გასვლისთანავე გადასასვლელი გაიშვება პეტრის ქსელის წესების მიხედვით, ანუ გადასასვლელის გააქტიურებას აუცილებლად მისი გახსნა მოჰყვება.

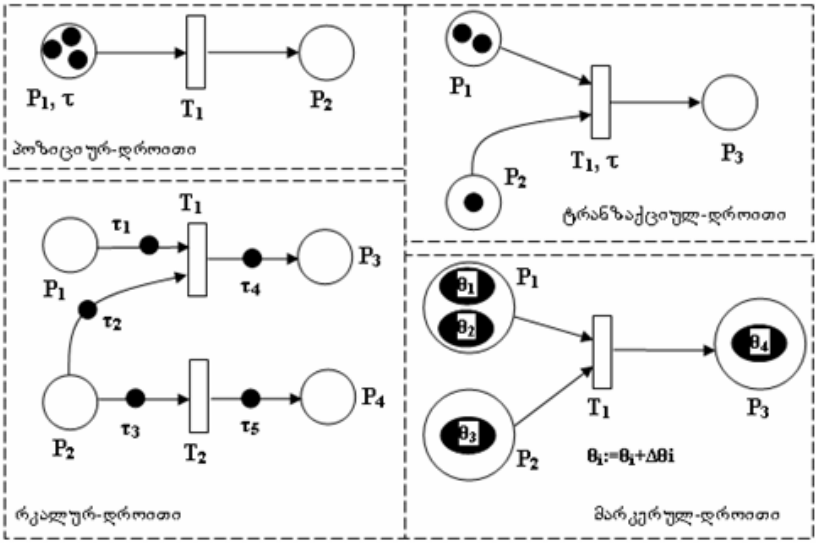
შეჯიბრის მოდელში მთავარი დროითი ფაქტორია, მარკერები ყველა აქტიურ გადასასვლელს ეკუთვნის და გაივლის მას, რომლის დაყოვნების დროც უფრო მალე გავა.

რკალურ-დროით პეტრის ქსელში დროითი დაყოვნების სიდიდეები რკალებს ენიჭება, განისაზღვრება მარკერის რკალში „მოგზაურობის“ დრო და გადასასვლელის გახსნა შესაძლებელია მხოლოდ მაშინ, როცა ყველა შემავალ რკალში მოძრავი მარკერი უწევს გადასასვლელს.

მარკერთა „მოგზაურობა“ გადასასვლელისკენ იწყება მხოლოდ მაშინ, როცა გადასასვლელის გახსნა ნებადართული ხდება. გახსნის შემდგომაც ყოველ რკალს ენიშნება მასში მარკერის „მოგზაურობის“ დრო, სანამ იგი გამომავალ პოზიციას მიაღწევს.

მარკერულ-დროითი პეტრის ქსელი ყოველი მარკერისათვის ცალკე დაყოვნების დროის განსაზღვრას მოითხოვს. ამგვარი ტიპი მოხერხებულია დროითი პრიორიტეტების მოდელირებისთვის.

დროითი პეტრის ქსელის სხვადასხვა ტიპები 4.8 ნახაზზეა მოცემული.



ნახ.4.8. დროითი პეტრის ქსელის ტიპები

ცხადია, პეტრის ქსელების დროითი გაფართოების შემოტანა მოდელირებისას ახალ პრობლემებს წარმოშობს. მაგალითად, ტრანზაქციულ-დროით პეტრის ქსელებში გასარკვევია, თუ როგორ უნდა ვმართოთ იმ გადასასვლელთა დაყოვნების დროები, რომლებმაც „შეჯიბრის“ შედეგად მარკერი დაკარგა და ხელახალ გააქტიურებას ელოდება გასახსნელად.

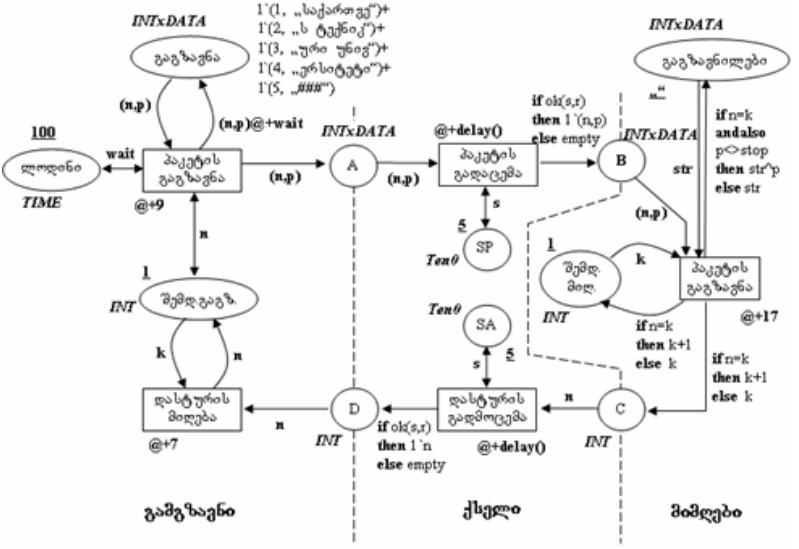
არსებობს ახალი დროითი დაყოვნების განსაზღვრის 2 ვარიანტი: დაფიქსირდება მარკერის დაკარგვისას დარჩენილი დრო (**Continue**) და გადასასვლელის შემდგომი გააქტიურებისას დროის „ჩამოყრა“ დარჩენილი დროიდან გაგრძელდება ან დროითი დაყოვნების საწყისი მნიშვნელობა ხელახლა განისაზღვრება (**Restart**). ამ ვარიანტებზე დაყრდნობით მთლიანად ტრანზაქციულ-დროითი პეტრის ქსელებითვის დროით გაფართოებათა მოდიფიცირების 3 სტრატეგია განისაზღვრება:

- **სრული რესტარტი (Resampling)** – ნებისმიერი გადასასვლელის გახსნისთანავე ქსელის ყველა გადასასვლელის დაყოვნების დრო თავიდან განისაზღვრება, არანაირი ინფორმაცია არ ინახება

• **ნაწილობრივი რესტარტი (Enabling Memory)** – მარკერწართმეული გადასასვლელების დაყოვნების დრო თავიდან განისაზღვრება (რესტარტი), ხოლო დანარჩენები (რომლებიც გააქტიურებულია) ჩვეულებრივად აგრძელებს დროის „ჩამოყრას“.

• **დროის შენახვა (Age Memory)** – გადასასვლელის გაშვებისას ყველა გადასასვლელის მიმდინარე დრო ინახება და გადასასვლელის შემდგომი გააქტიურებისას დროითი დაყოვნება შენახვის დროითი პუნქტიდან აგრძელებს შემცირებას.

უნდა აღინიშნოს, რომ დასაშვებია **ჰიბრიდული** პეტრის ქსელების აგება დროითი და არადროითი ელემენტებით, რაც ხშირად სისტემების მოდელირების და ანალიზის ყველაზე ეფექტურ ინსტრუმენტს წარმოადგენს. დროითგაფართოებიანი, ჰიბრიდული, ფერადი პეტრის ქსელის კომპლექსური მაგალითი 4.9 ნახაზზეა გამოსახული.




```
color INT = int timed; color DATA = String; color INTxDATA = Product INT*DATA;
var n,k : INT; var p,str : DATA; val stop="###";
color Ten0 = int with 1..10; color Ten1 = int with 1..10; var s:Ten0; var r:Ten1;
fun Ok(s:Ten0, r:Ten1) = (r<=s);
color NetDelay = int with 25..75 declare ran; fun Delay() = ran'NetDelay();
var wait:TIME;
```

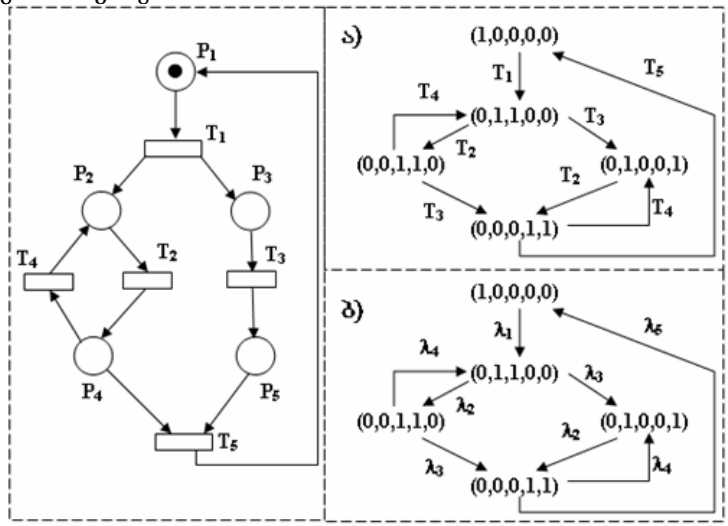
ნახ.4.9. დროითი ფერადი პეტრის ქსელი მარტივი ქსელური პროტოკოლისთვის

ტრანზაქციულ-დროით პეტრის ქსელებს, სადაც გადასასვლელის დაყოვნების დრო შემთხვევით განაწილებულ ექსპონენციალურ ფუნქციას წარმოადგენს, **ალბათური** ანუ **სტოქასტური პეტრის ქსელები** (**Stochastic Petri Nets**) ეწოდება, ხოლო სტოქასტური პეტრის ქსელი, რომელიც **დროითთან** ერთად **არადროით (მყისიერ)** გადასავლელებსაც შეიცავს, **განზოგადებულ სტოქასტურ პეტრის ქსელს** (**Generalized Stochastic Petri Nets**) წარმოადგენს [19]. ამგვარი ქსელის ქცევა **ალბათური** (მაგალითად, **მარკოვის**) **პროცესებით** აღიწერება.

მათემატიკურად სტოქასტური პეტრის ქსელი მიიღება პეტრის ქსელის განსაზღვრებაზე $\Lambda = \{\lambda_1, \lambda_2, \ldots \lambda_m\}$ სიმრავლის დამატებით, სადაც გადასასვლელთა გაშვების დრო ექსპონენციალურადაა განაწილებული და შემთხვევითი λ_i სიდიდის განაწილება:

$$F_{\lambda_i}(x) = 1 - e^{-\lambda_i x}$$

სტოქასტური პეტრის ქსელის მაგალითი მოცემულია 1.12 ნახაზზე, სადაც გადასასვლელი T_1 ნებადართულია $M_0=(1,0,0,0,0)$ საწყის მარკირებაში.



ნახ.4.10. სტოქასტური პეტრის ქსელი: ა - მიღწევად მარკირებათა „სიით“; ბ - ეკვივალენტურ მარკოვის ჯაჭვით

გადასასვლელის დაყოვნების დრო ექსპონენციალუ-რადაა განაწილებული და დამოკიდებულია λ_1 სიდიდეზე (გადასასვლელის კოეფიციენტი), ისე რომ გადასასვლელის გახსნის საშუალო დროა $\frac{1}{\lambda_1}$.

T_1-ის გახსნის შემდეგ მიიღება მარკირება M_1**=(1,0,0,0,0)** სადაც პარალელურად ნებადართული გადასასვლელებია T_2 და T_3. თუ T_2 პირველად გაიხსნა, მიიღება მარკირება M_2 **= (0,1,1,0,0)**, ხოლო თუ T_3 – M_3 **= (0,1,0,0,1)**. მომდევნო მარკირებები უკვე იმაზეა დამოკიდე-ბული, ,,შეჯიბრს'' რომელი გადასასვლელი მოიგებს. ალბათობა იმისა, რომ T_2 პირველად გაიხსნება, უდრის:

$$P[T_2] = P[\lambda_2 < \lambda_3] = \int_0^\infty \left(\int_0^x \lambda_2 e^{-\lambda_2 y} dy\right) \lambda_3 e^{-\lambda_3 x} dx = \int_0^\infty (1 - e^{-\lambda_2 x}) \lambda_3 e^{-\lambda_3 x} dx =$$
$$= \frac{\lambda_2}{\lambda_2 + \lambda_3}$$

ანალოგიურად:

$$P[T_3] = \frac{\lambda_3}{\lambda_2 + \lambda_3}.$$

ამ ფორმულებით ცხადი ხდება ისიც, რომ მარკირებათა ცვლილების ალბათობები გარკვეულ წინა მარკირებებში ყოფნის დროზე (''წინაისტორია'') არ არის დამოკიდებული. სტოქასტური პეტრის ქსელების წარმოდგენა და რაოდენობრივი ანალიზი შეიძლება შესაბამისი მარკოვის პროცესების ანალიზით განხორციელდეს, რაც ასევე 4.10 ნახაზზეა ასახული. ამ მძლავრი მათემატიკური აპარატის ინტეგრაცია სტოქასტურ პეტრის ქსელებს მიმზიდველ მოდელირების საშუალებად აქცევს, განსაკუთრებით **კონფლიქტების** მოდელირებისთვის.

4.6. ობიექტური პეტრის ქსელები

Object Petri Nets (და მისი გაფართოებები: **ობიექტ-ორიენტირებული** და **მაჩვენებლიანი** პეტრის ქსელები) პეტრის ქსელების თეორიისა და ობიექტ-ორიენტირებული დაპროგრამების თეორიის შეჯერებით მიღებული პეტრის ქსელის ტიპია [9]. ობიექტური პეტრის ქსელი ერთი **სისტემური** და რამდენიმე **ობიექტური** ქსელისგან შედგება, სადაც ობიექტური ქსელები **მარკერთა** როლში გამოდის. ფაქტობრივად, მიიღება პეტრის ქსელების სიმრავლე ერთი პეტრის ქსელის პოზიციებში.

ობიექტური ქსელები **ელემენტარულ სისტემურ ქსელებს** წარმოადგენს, ხოლო სისტემური ქსელი **მაღალი დონის პეტრის ქსელია**, რომლის პოზიციებშიც დაშვებულია როგორც ობიექტური ქსელების, ასევე ჩვეულებრივი შავი მარკერების არსებობა, ოღონდ არა ერთსა და იმავე პოზიციაში. შესაბამისად, რკალის ანოტაცია შეიძლება იყოს ნატურალური რიცხვი შავი მარკერებისთვის ან ობიექტური ქსელების განსაზღვრულ იდენტიფიკატორთა სიმრავლე.

სისტემური ქსელის ყოველ გადასასვლელს შეუძლია ობიექტური ქსელის გადატანა **არაუმეტეს** ერთი შემავალი პოზიციიდან **არაუმეტეს** ერთ გამომავალ პოზიციაში. ამასთანავე ერთ გაშვებაზე არაუმეტეს ერთი ობიექტური ქსელის გადატანაა ნებადართული.

ობიექტურ პეტრის ქსელებს სხვა ტიპის პეტრის ქსელებისგან გადასასვლელის როლის ზრდაც გამოარჩევს: სისტემური და ობიექტური ქსელების ზოგიერთ გადასასვლელს ემატება სპეციალური ფუნქცია, რომელსაც **ინტერაქცია** ეწოდება. ინტერაქცია 2 ტიპისაა: **სისტემ-ობიექტური** და **ობიექტ-ობიექტური**. პირველი სისტემური და ობიექტური ქსელების გადასასვლელთა სინქრონულ ურთიერთობას უზრუნველყოფს, მეორე – ობიექტური ქსელების ურთიერთსინქრონიზაციას.

სისტემ-ინტერაქციული გადასასვლელის გაშვების წესი შემდეგია: თუ სისტემური ქსელის გადასასვლელი ინტერაქციულია და მისი ინტერაქცია ქსელში მარკერის სახით მოძრავი ობიექტური ქსელის ნებადართულ ინტერაქციას თანხვდება, მაშინ

სისტემური ქსელის ინტერაქციული გადასასვლელის გაშვებისას ქსელში მოძრაობის პარალელურად გაიხსნება ობიექტური ქსელის ინტერაქციული გადასასვლელიც.

სხვა შემთხვევაში (თუ ინტერაქციები არ თანხვდება, ან სისტემური ან ობიექტური ქსელის გადასასვლელები ინტერაქციებს არ შეიცავს), ობიექტური ქსელი სისტემურში უცვლელი სახით გადაადგილდება. სწორედ ამგვარი მიდგომა განაპირობებს ობიექტური პეტრის ქსელების **ობიექტ-ორიენტირებულ** ხასიათს.

ზემოთ განხილული ფერადი პეტრის ქსელები სტრუქტურული დაპროგრამების თეორიასთან მჭიდრო კავშირში იმყოფება, შესამაბისად, შესაძლებელია ფერადი პეტრის ქსელებიდან მოქნილი გადასვლა ობიექტურ პეტრის ქსელებზე (როგორც სტრუქტურულიდან ობიექტორიენტირებული დაპროგრამების იდეოლოგიაზე) გარკვეული ახალი თვისებების შემოტანით.

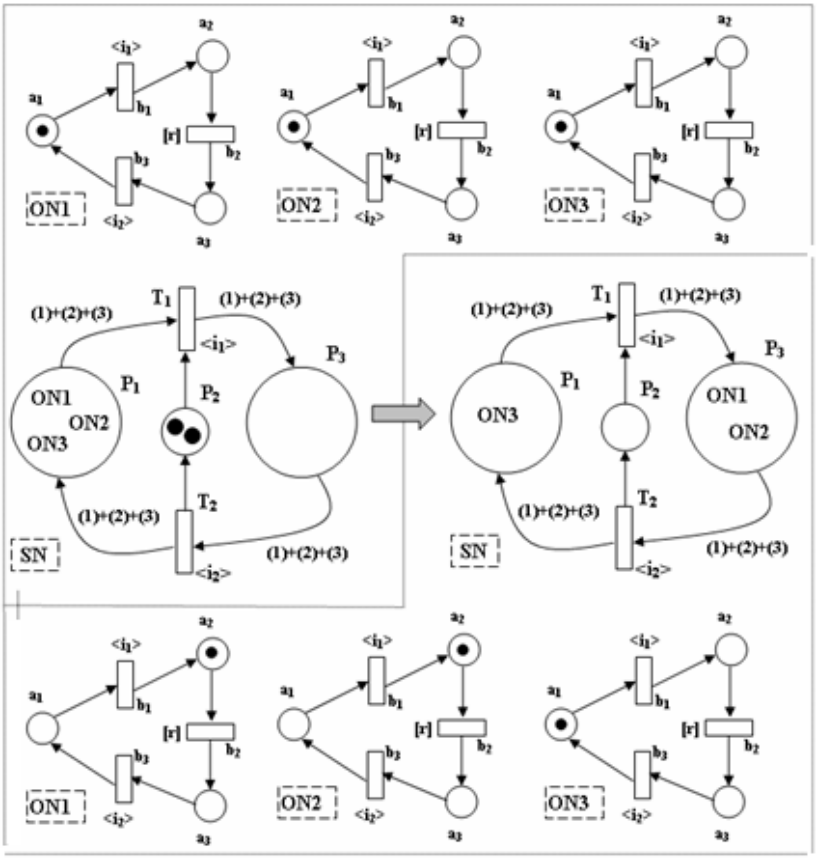
რამდენიმე მარტივი შესაბამისობა ობიექტ-ორიენტირებულ დაპროგრამებასა და ობიექტურ პეტრის ქსელებს შორის 4.1 ცხრილშია მოცემული.

ობიექტ-ორიენტირებული დაპროგრამებისა და ობიექტური პეტრის ქსელების ეკვივალენტური ელემენტები ცხრ.4.1.

ობიექტ-ორიენტირებული დაპროგრამება	ობიექტური პეტრის ქსელები
პროგრამული მოდული	სისტემური ქსელი
კლასი	ობიექტური ქსელის განსაზღვრება
ობიექტი	ობიექტური ქსელი კონკრეტული მარკირებით
ცვლადი	სისტემური ქსელის მარკერი
კლასის წევრი-ცვლადი	ობიექტური ქსელის მარკერი
გარე ფუნქცია	სისტემური ქსელის ინტერაქციული გადასასვლელი
კლასის წევრი-ფუნქცია	ობიექტური ქსელის გადასასვლელი

გავავლოთ პარალელი: გარე ფუნქციას (სისტემური ქსელის გადასასვლელი) კლასის წევრი-ცვლადის (ობიექტური ქსელის მარკერი) მოდიფიცირება (სხვა პოზიციაში გადანაცვლება) შეუძლია მხოლოდ მოცემული კლასის ობიექტის (მარკირებული ობიექტური ქსელი) გავლით, შესაბამისი წევრი-ფუნქციის

გამოძახებით (ინტერაქციული გადასასვლელის გაშვება), როგორც ეს 4.11 ნახაზზეა მოცემული.



ნახ.4.11. სისტემური და ობიექტური ქსელების მარკირებათა ცვლილებები

სისტემური ქსელის **(SN)** გადასასვლელი $\mathbf{T_1}$ და ობიექტური ქსელების **(ON1, ON2, ON3)** $\mathbf{b_1}$-გადასასვლელები შეიცავს ერთნაირ **სისტემ-ობიექტურ ინტერაქციებს (<i_1>),** რაც ნიშნავს, რომ თუ $\mathbf{T_1}$-ის გახსნისას ობიექტური ქსელი **ON1** გადაადგილდება, მაშინ იგი გამოიყენებს ერთ შავ მარკერს $\mathbf{P_2}$ პოზიციიდან და გადასასვლელის გახსნისას $\mathbf{P_1}$-დან მოხვდება $\mathbf{P_3}$

პოზიციაში და ამავდროულად ობიექტურ ქსელ **ON1**-ში მარკერი $\mathbf{a_1}$ პოზიციიდან $\mathbf{a_2}$-ში მოხვდება $\mathbf{b_1}$-ის გახსნის შედეგად.

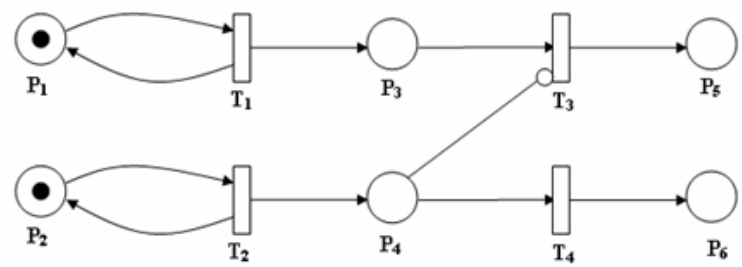
ობიექტურ ქსელებში გადასასვლელ $\mathbf{b_2}$-ისთვის განსაზღვრულია **ობიექტ-ობიექტური ინტერაქცია [r],** რომელიც ობიექტურ ქსელებს სისტემურ ქსელში პოზიციის შეუცვლელად საკუთარი შიდა მარკერების გადაადგილების საშუალებას აძლევს. მაგალითად, 4.11 ნახაზის ქვედა ნაწილში ობიექტური ქსელები **ON1** და **ON2** სისტემური ქსელის $\mathbf{P_3}$ პოზიციაში იმყოფება და რადგან მათი $\mathbf{b_2}$ გადასასვლელები ნებადართულია, ისინი გაიშვება კიდეც სინქრონულად $\mathbf{P_3}$ პოზიციის დატოვების გარეშე, რის შემდეგაც უკვე სისტემური ქსელის $\mathbf{T_2}$ გადასასვლელიც ნებადართული გახდება.

5. პეტრის ქსელების გაფართოებები

პეტრის ქსელების მთელი აღწერილი მრავალ-ფეროვნების მიუხედავად მისი მამოდელირებელი სიმძლავრე შეზღუდულია და რიგი ამოცანების მოდელირებისთვის არასაკმარისი. ამ პრობლემის გადასაჭრელად განსაზღვრულია პეტრის ქსელის **გაფართოებები,** რომლებიც სპეციფიკური საპრობლემო სფეროების მოდელირებისთვის გამოიყენება [10].

ყველაზე ფართოდ გავრცელებული გაფართოებაა **შემაკავებელი რკალი (Inhibitor Arc),** რომელიც პოზიციის „ნულზე შემოწმების“ პროცედურას ასრულებს, ანუ გადასასვლელს ნებადართულს ხდის მხოლოდ მაშინ, როცა შესაბამის შემავალ პოზიციაში არცერთი მარკერი არაა წარმოდგენილი. შემაკავებელი რკალი წრიული ბოლოთი გამოისახება. 5.1 ნახაზზე მისი დახმარებით გადასასვლელთა პრიორიტეტული გაშვების ამოცანა მოდელირდება (გადასასვლელი $\mathbf{T_3}$ ნებადართულია მხოლოდ მაშინ, როცა $\mathbf{T_4}$ არ არის ნებადართული, ანუ $\mathbf{P_4}$ პოზიციაში მარკერი არ არის), რაც კლასიკური პეტრის ქსელით შეუძლებელი იყო.

გარდა შემაკავებელი რკალებისა, პეტრის ქსელებში განისაზღვრება პოზიციების და გადასასვლელთა სხვადასხვა გაფართოებები.



ნახ.5.1. პეტრის ქსელი შემკავებელი რკალით

შეზღუდვის არე შემავალი პოზიციების გარკვეული სიმრავლეა, რომელთათვისაც გადასასვლელის გახსნის წესი შემდეგნაირად მოდიფიცირდება: გადასასვლელი შეიძლება გაიხსნას მხოლოდ მაშინ, როცა ერთი მაინც შემავალი პოზიცია ცარიელია.

გაფართოება **„გამომრიცხავი ან“** გულისხმობს გადასასვლელის გახსნას მაშინ, როცა შემავალი პოზიციებიდან მარკერი ერთსა და მხოლოდ ერთ პოზიციაშია (ნახ.5.2).

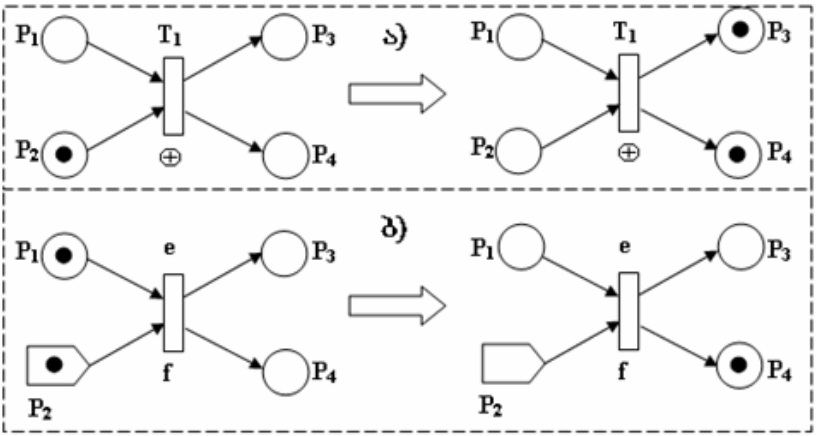
გადამრთველი გადასასვლელი შეიცავს სპეციალურ შემავალ **გადამრთველ პოზიციას** და ზუსტად 2 გამომავალ პოზიციას (**e** და **f** პოზიციები).

გადამრთველი გადასასვლელისთვის გადამრთველ პოზიციაში მარკერის არსებობა-არარსებობას მნიშვნე-ლობა არა აქვს, იგი გაიხსნება, თუ სხვა შემავალი პოზიციები შევსებულია მარკერებით.

ამასთან, გამომავალი პოზიციებიდან მარკერს მიიღებს მხოლოდ: პოზიცია **e** - თუ გადამრთველ პოზიციაში მარკერი არ არის, პოზიცია **f** – თუ მარკერი მოცემულია.

აღწერილი სამი გაფართოება, როგორც წესი, ჩანაცვლებადია შემაკავებელი რკალებით, ოღონდ ამ დროს პეტრის ქსელის სტრუქტურა რთულდება.

ზოგადად, გაფართოებები ზრდის პეტრის ქსელების **მამოდელირებელ სიმძლავრეებს**, მაგრამ იმავდროულად აბათილებს ქსელის **ანალიზის** შესაძლებლობას.



ნახ.5.2. პეტრის ქსელები: ა - „გამომრიცხავი ან“ და ბ - „გადამრთველი პოზიცია-გადასასვლელის“ გაფართოებებით (გადასასვლელთა გაშვებამდე და გაშვების შემდეგ)

ჩვენ აქ გავაანალიზეთ პეტრის ქსელების სამყაროს პრაქტიკულად მთლიანი სივრცე, თუმცა უნდა ითქვას, რომ პეტრის ქსელების სრული კლასიფიკაცია კიდევ მრავალი ტიპის ქსელებს მოიცავს.

6. პეტრის ქსელების უნიფიცირების კონცეფცია UML-ტექნოლოგიით

ზოგადად, ახალი ტიპის განსაზღვრა შესასრულებელი ამოცანის სპეციფიკაზეა დამოკიდე-ბული. რადგანაც პეტრის ქსელები ამოცანების ფართო კლასზეა გათვლილი, ახალი ამოცანის დასმისას ხანდახან მისი მიმდინარე მამოდელირებელი სიმძლავრეები არ კმარა, სწორედ ამ დროს განისაზღვრება პეტრის ქსელის ახალი ტიპი ან გაფართოება და იკვეთება პეტრის ქსელების სხვადასხვა ქვეკლასებისა და ტიპების თავსებადობის პრობლემა, რომელიც პირველ რიგში პრაქტიკული ხასიათისაა.

როგორი უნდა იყოს პროგრამული ბიბლიოთეკების ის საწყისი ნაკრები, რომელიც პეტრის ქსელის სიმულატორის ნებისმიერი ამგებისთვის სამუშაოს საწყისი პუნქტი იქნება? ამ კითხვაზე მარტივი პასუხის გაცემა, რომ ბიბლიოთეკა უნდა შეიცავდეს პეტრის ქსელების ელემენტების (პოზიციების, გადასასვლელების, რკალებისა და მარკერების) აღწერებს და მათი გამოსახვის საშუალებებს, არასაკმარისი იქნებოდა, რადგან პეტრის ქსელის სიმულატორზე მომუშავის წარმოდგენა იმაზე, თუ როგორ უნდა გამოიყურებოდეს პეტრის ქსელების ელემენტები და როგორი ტიპის ჭდეებს უნდა შეიცავდეს, ერთმანეთისგან განსხვავებული იქნება.

ამიტომ ამოცანა უნდა დაისვას ასე: შეიქმნას გარკვეული საბაზისო ინფრასტრუქტურა, რომელიც საერთო იქნება პეტრის ქსელის ნებისმიერი უკვე არსებული ტიპისთვის და რომლის საშუალებითაც სიმულატორის ამგები თავად მოახერხებს მისთვის საჭირო პეტრის ქსელის ტიპის განსაზღვრას.

ამასთან, ყველა პეტრის ქსელისთვის დამახასი-ათებელი თვისებები ცენტრალიზებულად უნდა იყოს შენახული ცალკე ბიბლიოთეკის სახით, რომელიც ახალი სიმულატორის შექმნის პროცესში კონკრეტული ტიპის პეტრის ქსელის აღწერით გაფართოვდება. ამგვარ სტრუქტურას ჩვენ **პეტრის ქსელების ბირთვის** სახით მოვიხსენიებთ.

ამოცანის პირველი ნაწილი ბირთვის თეორიული დახასიათებას წარმოადგენს, ხოლო მეორე ნაწილში მისი

პრაქტიკული რეალიზაციის ასპექტებია გამოსაკვლევი და შესასრულებელი.

ცხადია, ბირთვი უნდა აიგოს იმგვარად და ისეთ გარემოში, რომ იგი მისი ნებისმიერი მომავალი მომხმარებლისთვის ადვილად გასაგები და აღსაქმელი იყოს. წიგნის მეორე თავში ეს საკითხები განიხილება.

ნაშრომში დასმული და გამოკვლეული ამოცანის მეორე ნაწილს მართვის ავტომატიზებულ სისტემებში (მას) პეტრის ქსელების ერთიანი გაცვლითი ფორმატის გამოყენება წარმოადგენს.

პეტრის ქსელები განაწილებული სისტემებისა და ალგორითმების (მათ შორის, მას-ების) მოდელირე-ბისთვის განსაკუთრებით ეფექტური ინსტრუმენტია. შესაბამისად, პეტრის ქსელების ერთიან გაცვლით ფორმატს ამგვარი სისტემების მოდელირებისა და აგების პროცესში მნიშვნელოვანი როლის შესრულება შეუძლია.

მას-ები ,,ადამიანურ-მანქანური'' სისტემებია, მრავალრიცხოვანი, თვისობრივად განსხვავებული კომპონენტებით, რომელთა ერთ სისტემაში თავმოყრა და ორგანიზება დიდი სისტემებისთვის რთული და შრომატევადი საქმეა. სისტემის დაპროექტების ეტაპზე დაშვებული მცირე შეცდომაც კი მისი აგების, დანერგვისა და ექსპლოატაციის პროცესში მნიშვნელოვანი ფინანსური დანახარჯების მიზეზი შეიძლება გახდეს.

პეტრის ქსელები ამგვარი სისტემების მოდელირე-ბისა და ანალიზის კარგად განვითარებულ საშუალებებს შეიცავს, თუმცა ქსელურ ტექნოლოგიათა შემდგომი განვითარება (განსაკუთრებით კორპორაციული ქსელებისა და ინტერნეტის დამკვიდრება) ახალი ტიპის მართვის ავტომატიზებულ სისტემათა აუცილებლობას განაპირობებს, რომელთა მოცულობა დიდია და ხშირად ერთი ქვეყნის ფარგლებს სცილდება (ტრანსნაციონალურ კორპორაციათა ქსელები, ვებ-სერვისები), ხოლო დაპროექტებისა და აგების ვადები – შეზღუდული.

საჭირო ხდება სამუშაოთა პარალელური წარმართვა დამპროექტებელთა სხვადასხვა ჯგუფების მიერ, რომლებიც უმეტეს წილად დაპროექტების ასევე სხვადასხვა ინსტრუმენტებს იყენებენ (მაგალითად, პეტრის ქსელების სხვადასხვა ტიპებს), რაც

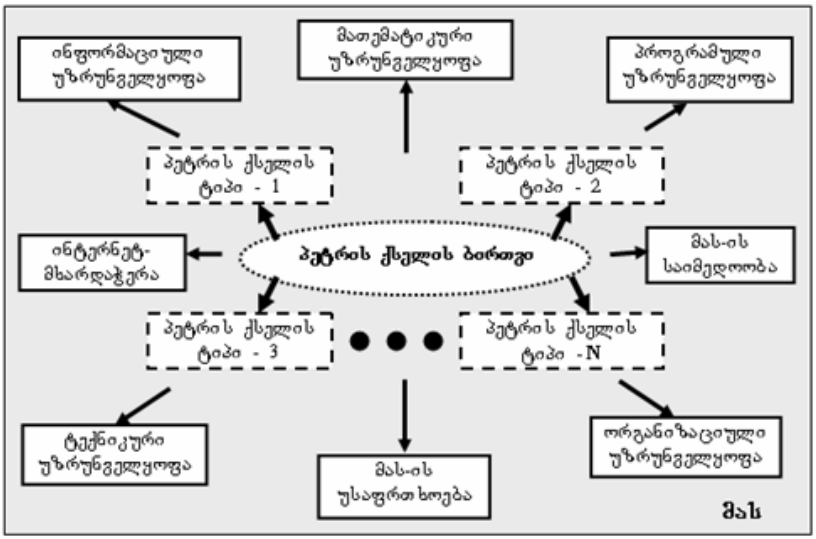
ქვესისტემათა მოდელების შემდგომი არა-თავსებადობის მიზეზი შეიძლება გახდეს.

პეტრის ქსელის ბირთვის შემოღებით მართვის ავტომატიზებული სისტემების მოდელირება უნიფიცი-რებული გახდება. მთელი სისტემის წარმოდგენა ერთიანი, ცენტრალიზებული დოკუმენტის ან დოკუმენტების ნაკრების სახით განხორციელდება, რომელთა დამუშავება ასევე ცენტრალიზებულად შესრულდება.

სხვანაირად რომ ვთქვათ, პეტრის ქსელების ერთიანი გაცვლითი ფორმატი მართვის ავტომატიზებული სისტემების დაპროექტებისა და აგების ინტეგრირებული ინფრასტრუქტურის ასაგებად უნდა იქნეს გამოყენებული. ამგვარი მიდგომის ეფექტურობა იმაშიც გამოიხატება, რომ მას-ების უკვე არსებული პროექტები (ან მათი ცალკეული ბლოკები) უფრო ადვილად გამოყენებადი იქნებოდა ახალი მას-ების ასაგებად, რაც სისტემების დაპროექტების აგების საერთო ღირებულებას შეამცირებდა.

ამრიგად, მეორე ამოცანას მართვის ავტომატიზებულ (ადამიანურ-მანქანურ) სისტემებში პეტრის ქსელების ინსტრუმენტისა და მისი უნიფიცირებული ვარიანტის – პეტრის ქსელის ბირთვისთვის შესაბამისი ადგილის მოძებნა წარმოადგენს (ნახ. 6.1).

როგორც მოცემული სქემა გვიჩვენებს, **მას**-ის სხვადასხვა ქვესისტემების მოდელირებისთვის, მათი სპეციფიკის მიხედვით პეტრის ქსელის განსხვავებული ტიპები შეიძლება იქნეს გამოყენებული. მაგალითად, თუ **მას**-ის ქსელური უზრუნველყოფის აღსაწერად ხშირად დაბალი დონის, დროითგაფართოებიანი პეტრის ქსელებიც კმარა (რომლებიც სისტემას უფრო მარტივად აღსაქმელი სახით ამოდელირებს), მონაცემთა და ცოდნის ბაზებისთვის აუცილებლად მაღალი დონის (მაგალითად, ფერადი) პეტრის ქსელებია საჭირო.



ნახ.6.1. პეტრის ქსელის ბირთვი მას-დაპროექტების და რეალიზაციის პროცესში

თუ მას-ში ალბათური პროცესები მიმდინარეობს (მაგალითად, კომპიუტერულ ქსელებში), სტოქასტური პეტრის ქსელებია აუცილებელი და ასე შემდეგ. ჩვენი ამოცანაა მთელი მას-ის დაპროექტებისა და აგების საწარმოო პროცესის ისე წარმართვა, რომ ქვესისტემებმა ერთმანეთთან კავშირის მოქნილად დამყარება მოახერხოს. ამისთვის აუცილებელია მამოდელირებელი ინსტრუ-მენტის (ჩვენს შემთხვევაში პეტრის ქსელების) იმგვარი უნიფიკაცია, რომელიც პეტრის ქსელის სხვადასხვა ტიპებს და შესაბამისად, ამ ტიპებით დაპროექტებული და აგებული მას-ის ქვესისტემებს შორის "დიალოგს", ხოლო საბოლოოდ ქვესისტემების სრულ ურთიერთ-ინტეგრაციას გახდიდა შესაძლებელს.

7. პეტრის ქსელების სიმულატორების (ინსტრუმენტების) ანალიზი

პეტრის ქსელების ინსტრუმენტების, ეგრეთ წოდებული **სიმულატორების** სრული და მუდმივგანახლებადი მონაცემთა ბაზა პეტრის ქსელების ოფიციალურ ვებ-გვერდზე მოიპოვება [11]. მონაცემთა ბაზაში 50-მდე რეგისტრირებული სიმულატორი შედის, აქედან 11 კომერციული პროდუქტია (სასწავლო დაწესებულებებისთვის ფასდაკლებით), ხოლო 39 თავისუფლად ვრცელდება ინტერნეტის ქსელში.

პლატფორმებიდან (კომპიუტერის არქიტექტურა პლუს ოპერაციული სისტემა) **PC**-არქიტექტურისთვის (**Windows** ოპერაციული სისტემით) ყველაზე მეტი სიმულატორია შექმნილი, თუმცა პეტრის ქსელის სიმულატორთა გავრცელების არე ძალიან ფართოა. არსებობს სიმულატორები თვით **SiliconGraphics** (ოპერაციული სისტემა **IRIX**) ფირმის სუპერ-კომპიუტერებისა და **HewletPackard**-ის (ოპერაციული სისტემა **HP-UX**) პლატფორმებისთვის, აგრეთვე **OS2**-სა და **BSD**-ოჯახის ოპერაციული სისტემებისთვის **(FreeBSD, NetBSD, OpenBSD).** პეტრის ქსელის რეგისტრირებულ სიმულატორთა გავრცელების სადღეისო ვითარება 7.1 ცხრილშია მოცემული.

პეტრის ქსელების სიმულატორები და პროგრამული პლატფორმები ცხრ.7.1

არქიტექტურა	ოპერაციული სისტემა	სიმულატორთა რაოდენობა
PC	Ms Dos	4
	Windows 95/98/NT/2000/XP	31
	LINUX	22
SUN	SUN OS, SOLARIS	21
Macintosh	MAC OS, MAC OS X	5
JAVA		18

ამ ცხრილში არ იანგარიშება სიმულატორთა ის ნაწილი, რომლებიც მორალურად მოძველდა და მონაცემთა ბაზიდან წაშლილ იქნა (როგორც, მაგალითად, ადრე ცნობილი გერმანული სიმულატორი **Pepsi**). კლასიკურ პეტრის ქსელებთან ერთად სულ უფრო მეტი ახალი სიმულატორი იძენს მაღალი დონის პეტრის ქსელების აგებისა და ანალიზის საშუალებებს. სიმულატორთა რაოდენობრივი განაწილება პეტრის ქსელის ტიპების მიხედვით 7.2 ცხრილში აისახება.

პეტრის ქსელების სიმულატორთა განაწილება ტიპების მიხედვით.

ცხრ.7.2

პეტრის ქსელის ტიპი	P/T	დროითი	სტოქას-ტური	ობიექ-ტური	მაღალი დონის
სიმულა-ტორთა რაოდ.	36	33	16	5	31

ზემოთ მოყვანილი ცხრილებიდან ნათელი ხდება, რომ სხვადასხვა სიმულატორები ხშირად ერთსა და იმავე სამუშაოს განმეორებით ასრულებს ერთმანეთთან შეუთავსებლობის გამო, რაც ასევე ახალი, ერთიანი გაცვლითი ფორმატის შექმნის აუცილებლობაზე მიუთითებს.

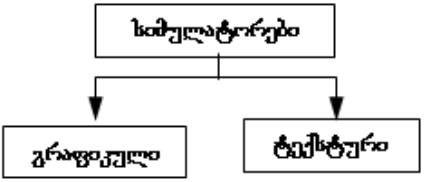
ჩამოვთვალოთ მთავარი თვისებები, რომელიც პეტრის ქსელის სიმულატორებს გააჩნიათ ან უნდა გააჩნდეთ:

ყოველი სიმულატორის უპირველესი კომპონენტი პეტრის ქსელის **აგების** და **გამოსახვის** საშუალებაა. ქსელი შეიძლება ფაილიდან ჩაიტვირთოს ან მომხმარებელთან ინტერაქტიულ დიალოგში შეიქმნას. მრავალი სიმულატორი დამატებით მარკერთა სიმულაციის ვიზუალური გრაფიკული საშუალებებითაცაა აღჭურვილი (**HPSim, CPN-Tools, VisualObjectNet++** და სხვ.). ანიმაციური ელემენტების ჩართვა სიმულატორს სიცხადეს ჰმატებს.

ანალიზის საშუალებები პეტრის ქსელის სიმულატორებისთვის ასევე მნიშვნელოვანი კომპონენტია.

იგულისხმება ინვარიანტულობის მატრიცების და მიღწევადობათა „ხის“ აგება და დამუშავება, აგრეთვე მათი საშუალებით პეტრის ქსელების სხვადასხვა ამოცანათა (**მიღწევადობა, უსაფრთხოება, შეზღუდულობა, აქტიურობა, სხვადასხვა პეტრის ქსელთა ორეულობა, ინვერსულობა** და ასე შემდეგ) გადაწყვეტა.

მხოლოდ გარეგნული თვალსაზრისით თუ გავარჩევთ, სიმულატორების 2 კლასი გამოიყოფა (ნახ.7.1).



ნახ.7.1. პეტრის ქსელის სიმულატორების კლასები

გრაფიკული სიმულატორებია **CPN-Tools, Renew, PEP** და სხვ., ხოლო ტექსტური - **INA, MARIA, LoLA**.

პირველი უფრო პეტრის ქსელების დიზაინერებსა და მომხმარებლებზეა გათვლილი, მეორე – ექსპერტებზე. ეს უკანასკნელი კლასი გრაფიკული ინტერფეისის უარყოფის სანაცვლოდ პეტრის ქსელების ანალიზის კარგად განვითარებულ საშუალებებს შეიცავს. უკვე არის ორი კლასის სიმულატორების ინტეგრირების ცალკეული მცდელობებიც.

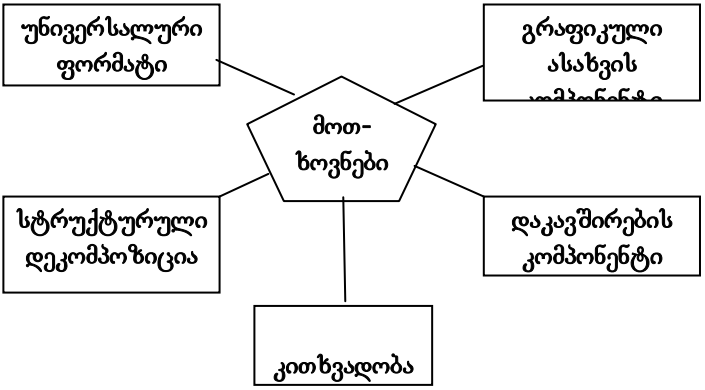
პეტრის ქსელების ერთიან მონაცემთა ფორმატზე მუშაობა მხოლოდ რამდენიმე წლის წინ დაიწყო. უკვე შექმნილ სიმულატორებში მონაცემთა ფორმატი უმრავლეს შემთხვევაში დამოუკიდებლად განისაზღვრება. დამპროგრამებელი თვითონ განსაზღვრავს პეტრის ქსელის ელემენტების შენახვისა და ასახვის მექანიზმს. შესაბამისად, სხვადასხვა ინსტრუმენტებს შორის მონაცემთა გაცვლა დამატებითი დაპროგრამების გარეშე ვერ ხერხდება. იყო მხოლოდ რამდენიმე მცდელობა ერთიანი ფორმატის შემოღებისა, ოღონდ არა პეტრის ქსელის ყველა, არამედ ზოგიერთი კონკრეტულ ტიპებს შორის (მაგალითად **PEP** და **Design/CPN**), რომლებიც საერთო ამოცანებმა დაუკავშირა ერთმანეთს. საერთო ჯამში, პეტრის ქსელის სიმულატორთა ურთიერთ-დამოუკიდებლობის და არათავსებადობის დონე მაინც უაღრესად მაღალია.

8. PNML-ის მოდელი, ძირითადი თვისებები, კომპონენტები და სტრუქტურა

წინა თავში პეტრის ქსელების ერთიანი კონცეფციის ერთერთი ვარიანტი შემოვიტანეთ. მასში მთავარი მახვილი **პეტრის ქსელის ტიპის** განსაზღვრაზეა დასმული, რომელიც **ქსელის, ჭდეებისა** და **გახსნის წესების** ერთობლიობას წარმოადგენს. ამათგან ქსელის საბაზო სტრუქტურა (**პოზიციები, გადასასვლელები, რკალები**) ფუნდამენტური და მსგავსია ყველა ტიპის პეტრის ქსელისთვის, ხოლო **ჭდეების სიმრავლე** და **გადასასვლელთა გახსნის წესები** – განსხვავებული.

წინამდებარე თავში პეტრის ქსელების სინტაკსური ასახვის პრობლემებს შევეხებით.

საჭიროა ჩამოყალიბდეს უნივერსალური აღწერის ენა, რომელიც პეტრის ქსელის ნებისმიერი ტიპის აღწერისთვის იქნება გამოსადეგი (ნახ.8.1.).



ნახ.8.1.

- პირველი მოთხოვნა, რომელიც ამგვარ ენას წაეყენება, არის **მონაცემთა უნივერსალური ფორმატი**, რომელიც ამავდროულად **მოქნილი**, **თავსებადი** და **გაფართოებადიც** იქნება

პეტრის ქსელის ახალი ტიპების უმტკივნეულოდ ინტეგრირებისთვის.

მოქნილობა გულისხმობს, რომ ენამ უნდა ასახოს პეტრის ქსელის ყოველი ტიპი თავის უნიკალური თვისებებით. მან არ უნდა შეზღუდოს ან დამალოს პეტრის ქსელის ზოგი თვისება მისი კონვერტირების პროცესში.

თავსებადობა ნიშნავს მაქსიმალური ოდენობის ინფორმაციის გაცვლის შესაძლებლობას პეტრის ქსელების სხვადასხვა ტიპებს შორის, ხოლო გაფართოებადობა პეტრის ქსელის ახალი ტიპების განსაზღვრის შესაძლებლობას.

- მეორე მოთხოვნას ენის **გრაფიკული** კომპონენტი წარმოადგენს ქსელის ელემენტების კორექტული ასახვისთვის, რადგან გრაფიკული ასახვა უმნიშვნელოვანესია პეტრის ქსელის პრაგმატიკის გასაგებად.

- ენის მესამე მოთხოვნად **დაკავშირება** მოიაზრება: ენამ უნდა უზრუნველყოს, რომ ყოველი განსაზღვრული **ჭდე** აუცილებლად ქსელის რომელიმე კომპონენტთან (**კვანძი**, **რკალი**) იყოს დაკავშირებული.

- მეოთხე მოთხოვნა **სტრუქტურიზაცია** იქნება: დიდი პეტრის ქსელურ მოდელთა ცალკეულ **მოდულებად** დაყოფა და მოდულთაშორის კავშირების განსაზღვრა

- წაკითხვადობის მოთხოვნა გულისხმობს, რომ ახალი ენით აღწერილი ქსელის წაკითხვა „შეუიარაღებელი“ თვალით იყოს შესაძლებელი, შუალედური ტრანსლატორების გარეშე.

ჩამოთვლილ მოთხოვნათა ხორცშესხმის ყველაზე ეფექტურ საშუალებად გვეჩვენება **პეტრის ქსელის აღწერის ენისთვის პეტრის ქსელის სხვადასხვა ტიპების ერთიანი გრამატიკის ფორმირება**.

ამგვარი მიდგომის მიზანია პეტრის ქსელის შიგთავსის გამოყოფა დამმუშავებელი პროგრამისგან და მისი შენახვა არა ჩვეულებრივი, არამედ სტრუქტურული (იერარქიული) ფაილის ფორმით, რაც მოხერხებულია პეტრის ქსელების ერთიანი გაცვლითი ფორმატის ასაგებადაც, რომელიც კონკრეტულ ინსტრუმენტზე აღარ იქნება დამოკიდებული და თავად მიაწვდის მას ინფორმაციას პეტრის ქსელის ელემენტების ფორმისა და შინაარსის შესახებ.

ამრიგად, საუბარი გვაქვს ერთგვარ **მონიშვნათა** ახალ ენაზე, რომელსაც **პეტრის ქსელის ფორმატირების (მონიშვნათა) ენას (Petri Net Markup Language - PNML)** ვუწოდებთ.

ფორმატირების ენები ინტერნეტის პროგრესთან ერთად სულ უფრო მეტ მნიშვნელობას იძენს. **PNML**-ის საფუძვლად **მონიშვნების გაფართოებული ენა XML** არის აღებული, რომელიც სადღეისოდ ყველაზე პოპულარულ ინტერნეტ-ტექნოლოგიას წარმოადგენს.

ერთერთ მომდევნო პარაგრაფში მოკლედ გვაქვს განხილული **XML** ენა, რადგან **PNML**-ის საფუძველს სწორედ მისი სინტაკსი წარმოადგენს.

8.1. PNML-ის კომპონენტები

ინფორმაციის ტიპები PNML-ში. **PNML**-ენის განსაზღვრისას მე-7 პარაგრაფში მოცემულ განსაზღვრებათა თანახმად 2 ტიპის ინფორმაციაა საჭირო: ქსელის ტიპისგან **დამოუკიდებელ** (ყველა ქსელისთვის უნიკალურ) და **დამოკიდებულ** ქსელის ელემენტებზე.

დამოუკიდებელ ელემენტებს მივაკუთვნებთ **კვანძებს (პოზიციები, გადასასვლელები)** და **რკალებს**. პეტრის ქსელის ტიპისგან დამოუკიდებლად განისაზღვრება აგრეთვე **გვერდები** და **მოდულები**.

ყოველი ელემენტისთვის, აგრეთვე მთლიანად პეტრის ქსელისთვის განვსაზღვრავთ უნიკალურ **იდენტიფიკატორს** (კოდი). იდენტიფიკატორების გამოყენების არე საკმაოდ ფართო იქნება: მაგალითად, რკალები გამოიყენებს პოზიციების და გადასასვლელთა იდენტიფიკატორებს ქსელში თავიანთი კავშირების იდენტიფიცირებისთვის.

პეტრის ქსელის ტიპზე დამოკიდებულ ელემენტს ვუწოდებთ **ჭდეს**, რომელიც ყველა ელემენტს შეიძლება ერთვოდეს და მის შინაარსს განსაზღვრავდეს.

თავისთავად ჭდის სინტაქსი პეტრის ქსელის ბირთვის შემადგენელი ნაწილია, ხოლო სემანტიკა (ჭდეების სახეობები, რაოდენობა და ნებადართული კომბინაციები) პეტრის ქსელის

ყოველი კონკრეტული ტიპისთვის ცალ-ცალკე უნდა განისაზღვროს.

ზოგადად, ჭდეების 2 ტიპის განსაზღვრა იქნება საჭირო: **წარწერებისა** და **ატრიბუტების**, სადაც წარწერა მნიშვნელობათა შეუზღუდავი სიმრავლის მქონე ჭდეს წარმოადგენს და ტექსტური ფორმით გამოისახება პეტრის ქსელის საბაზო ელემენტის გვერდით, ხოლო ატრიბუტის მნიშვნელობათა სიმრავლე შეზღუდულია.

პირველის მაგალითებია ჭდეები გადასასვლელთა გახსნის პირობის, პოზიციების სახელებისა და მარკირებათა ასახვისთვის, მეორის მაგალითად რკალის ტიპის **(**რომლის მნიშვნელობათა სიმრავლეა **IN, OUT, READ, INHIBITOR)** ამსახველი ჭდის დასახელება შეიძლება განვიხილოთ.

ამასთან ყოველ წარწერას თანმხლები გრაფიკული ინფორმაცია (ეკრანული კოორდინატები, დაშორება დაკავშირებული ელემენტიდან) ახლავს, ხოლო ატრიბუტს იგი არ გააჩნია, ატრიბუტი თვით ელემენტის გამოსახვის ფორმას განსაზღვრავს (მაგალითად, რკალი ატრიბუტით **INHIBITOR** შემაკავებელ რკალს განსაზღვრავს, რომელიც პატარა წრით ბოლოვდება).

გრაფიკული ინფორმაცია სხვადასხვა ტიპის შეიძლება იყოს: კვანძისთვის - პოზიციის კოორდინატები, რკალისთვის – საშუალედო პოზიციების კოორდინატთა სია, ანოტაციისთვის – ფართობითი პოზიცია შესაბამისი ობიექტის მიმართ (დაშორება).

გრაფიკულ ინფორმაციად ითვლება აგრეთვე მონაცემები ზომის, ფერის, კვანძების და რკალების ფორმის შესახებ ან ჭდეების ფერზე, შრიფტზე და შრიფტის ზომაზე.

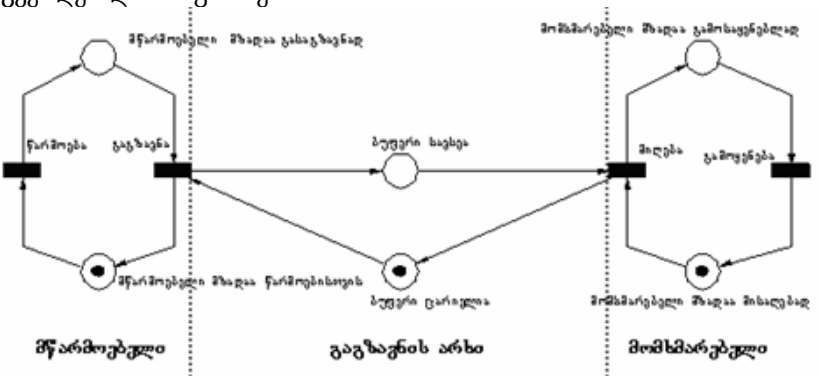
პეტრის ქსელის ზოგი ინსტრუმენტი შეიძლება შეიცავდეს **ინსტრუმენტის სპეციფიკურ ინფორმაციას**, რომელსაც სხვა ინსტრუმენტები ვერ ცნობს. ამგვარი ინფორმაციის შესანახად ყოველი ობიექტი და ჭდე უნდა აღიჭურვოს **ინსტრუმენტის სპეციფიკური ინფორმაციის ბლოკით**, რომლის ფორმატი კონკრეტულ ინსტრუმენტზეა დამოკიდებული და **PNML**-ით არ აღიწერება.

სტრუქტურიზაცია PNML-ენის სინტაქსში აუცილებე-ლია, იგი პეტრის ქსელის ცალკეულ ნაწილებად, ანუ **გვერდებად**

დაყოფას გულისხმობს. **გვერდი** არის ობიექტი, რომელიც პეტრის ქსელის ელემენტების ქვესიმრავლეს ან/და სხვა გვერდებს შეიცავს. ამასთან, გვერდის საზღვარი შეიძლება გადიოდეს კვანძებზე, მაგრამ არა რკალებზე ანუ რკალს მხოლოდ ერთი გვერდის ფარგლებში მოთავსებული კვანძების დაკავშირება შეუძლია, როგორც ეს 8.2 ნახაზზე „მწარმოებელ-მომხმარებლის“ სისტემისთვისაა ნაჩვენები.

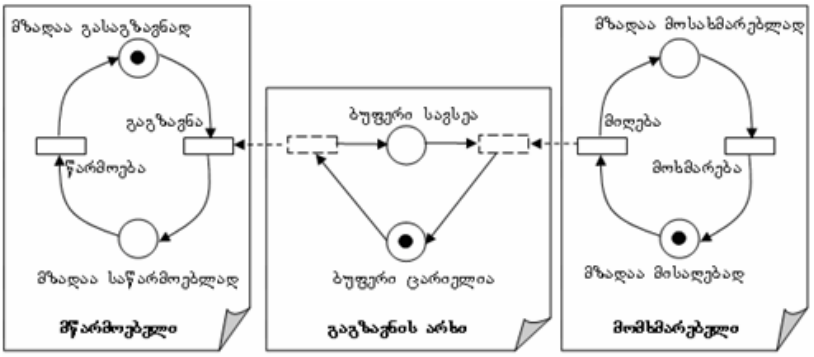
მიუხედავად იმისა, რომ გვერდების საზღვრები ყოველთვის კვანძებზე გადის, არ შეიძლება სასაზღვრო კვანძი ორივე გვერდს ეკუთვნოდეს, ამიტომ შემოგვაქვს **მაჩვენებელი კვანძის** ცნება, რომელიც **მიზნის კვანძზე** მაჩვენებელს წარმოადგენს და ზუსტად ერთ პოზიციაზე (**მაჩვენებელი პოზიცია**) ან გადასასვლელზე (**მაჩვენებელი გადასასვლელი**) მიუთითებს.

სასაზღვრო კვანძი ერთ რომელიმე გვერდზე იქნება განთავსებული, ხოლო მაჩვენებელი კვანძები მასზე სხვა გვერდებიდან მიუთითებს.



ნახ.8.2. პეტრის ქსელის დაგვერდვის მაგალითი

ამგვარად მოდერნიზებული „მწარმოებელ-მომხმარებლის“ სისტემა 8.3 ნახაზზეა წარმოდგენილი, სადაც მაჩვენებელი გადასასვლელები წყვეტილჩარჩოიანი მართკუთხედებით გამოისახება.



ნახ.8.3. გვერდებად დაყოფილი პეტრის ქსელი „მწარმოებელ-მომხმარებლის“ სისტემისთვის

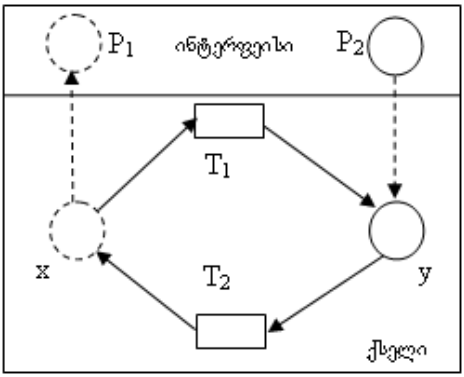
გვერდებად დაყოფილი პეტრის ქსელიდან საწყისი ქსელის მიღება ადვილია. საამისოდ მაჩვენებელი კვანძები მიზნის კვანძებს ერწყმის და გვერდების საზღვრები უგულვებელიყოფა.

უკვე არსებული პეტრის ქსელის სიმულატორებიდან დაგვერდვის ფუნქციას შეიცავს ინსტრუმენტი **CPN-Tools.**

პეტრის ქსელების სტრუქტურიზების უფრო სრულყოფილ მეთოდად **მოდულარიზაციას** წარმოვად-გენთ, რაც დაგვერდვაზე რთული პროცედურაა, რადგან მოდულთაშორის კავშირები უფრო რთული ხასიათისაა.

მოდულს განვიხილავთ, როგორც პეტრის ქსელის ნაწილს დანარჩენ პეტრის ქსელთან (მოდულის **გარემო**) ინფორმაციის გაცვლის საშუალებით. მოდული შედგება **მოდულ-ქსელური** და **ინტერფეისის** ნაწილებისგან.

პირველი მოდულში შემავალ პეტრის ქსელის ფრაგმენტს შეიცავს, მეორე – მოდულის გარემოსთან კავშირის საშუალებებს. მოდულის მაგალითი 8.4 ნახაზზეა მოცემული.



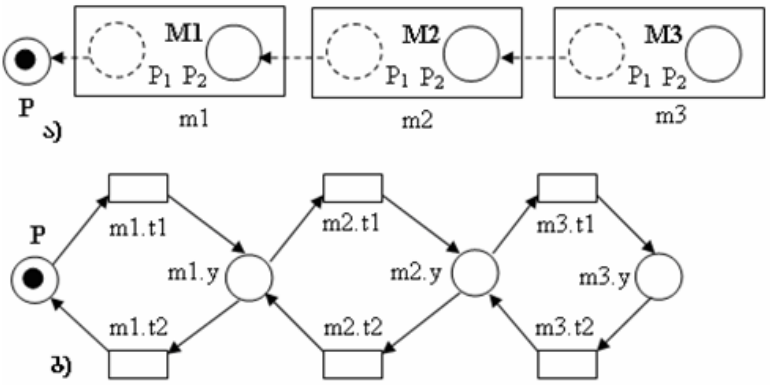
ნახ.8.4. მოდული ქსელისა და ინტერფეისის ნაწილებით

მოდულის ქსელური ნაწილიდან გარემოსთან პირდაპირი კავშირი აკრძალულია, ანუ არ შეიძლება მოდულის ქსელში იმგვარი კვანძის არსებობა, რომელზეც გარედან რომელიმე მაჩვენებელი კვანძი მიუთითებს და პირიქით, მოდულის ქსელი არ უნდა შეიცავდეს მაჩვენებელ კვანძებს, რომლებიც გარემოს კვანძებზე მაჩვენებლები იქნება.

მოდული გარემოს მხოლოდ ინტერფეისის გავლით შეიძლება დაუკავშირდეს. ჩვენს მაგალითში მოდულის ქსელი შეიცავს 2 გადასასვლელს ($\mathbf{T_1}$ და $\mathbf{T_2}$), 1 პოზიციას (**y**) და 1 მაჩვენებელ პოზიციას (**x**). ინტერფეისისთვის განვსაზღვრავთ 2 პოზიციას: $\mathbf{P_1}$-ს **იმპორტის პოზიციას** ვუწოდებთ და წყვეტილი წრეწირით გამოვსახავთ, $\mathbf{P_2}$ **ექსპორტის პოზიცია** იქნება და ჩვეულებრივი პოზიციის რგოლის სახით გამოიტანება.

პირველი გარემოდან მოდულში მონაცემთა იმპორტის ფუნქციას შეასრულებს, მეორე – მოდულიდან გარემოში ექსპორტისა. მაჩვენებელი პოზიცია **x** წარმოადგენს მაჩვენებელს იმპორტის პოზიცია $\mathbf{P_1}$-ზე, ხოლო ექსპორტის პოზიცია $\mathbf{P_2}$ მოდულური ქსელის **y**-პოზიციაზე.

ამგვარი მექანიზმით მოდულის შემავალი და გამომავალი პარამეტრები განისაზღვრება, რაც კომპლექსურ პეტრის ქსელში მოდულის მრავალჯერადი გამოყენების საშუალებას იძლევა (მოდულის **ეგზემპლარების** სახით). შესაბამისი მაგალითი 8.5 ნახაზზეა ნაჩვენები.



ნახ.8.5. ა) მოდულური პეტრის ქსელი,
ბ) მოდულური პეტრის ქსელის სემანტიკა

ნახაზზე პეტრის ქსელის ასაგებად დამოუკიდებელი პოზიცია **P** და **M1**-მოდულის 3 ეგზემპლარი გამოიყენება: **m1, m2** და **m3.** ამასთან, მოდულის ქსელური ნაწილის გამოსახვისას საკმარისია მხოლოდ მოდულის ინტერფეისული ნაწილის გამოსახვა, რადგან სწორედ იგი შეიცავს მოდულის იმპორტის და ექსპორტის **პარამეტრებს** და მის გარეშე გარემოდან მოდულზე არავითარი ზემოქმედება არ ხდება.

მოდულის ეგზემპლარი **m1** პოზიცია **p**-ს თავისი იმპორტის პოზიცია $\mathbf{P_1}$-ის **პარამეტრად** იყენებს, რასაც მოდულურ პეტრის ქსელზე გრაფიკულად $\mathbf{P_1}$-დან **P**-ზე მიმართული მაჩვენებლის რკალით გამოვსახავთ. ეგზემპლარი **m2** თავისი იმპორტის პოზიცია $\mathbf{P_1}$-ის **პარამეტრად m1**-ის ექსპორტის პოზიცია $\mathbf{P_2}$-ს იღებს და ასე შემდეგ.

ნახაზის მეორე ნაწილში მოდულური პეტრის ქსელი „გაშლილი“ სახითაა წარმოდგენილი და მისი სემანტიკაა გადმოცემული.

სხვათა შორის, ამ ქსელის ყოველი კომპონენტის უნიკალურობის დასაცავად კომპონენტები მოდულის ეგზემპლარისა და კომპონენტის სახელებით აღიწერება, რომლებიც ერთმანეთისგან წერტილითაა გამოყოფილი.

8.2. PNML-ის სტრუქტურა

პეტრის ქსელების მონიშვნათა ენის ერთიანი სტრუქტურა 3 დონისგან შედგება. ყველაზე ქვედა დონე პეტრის ქსელის ფუნდამენტური კომპონენტების აღწერას მოიცავს, რომელი ნებისმიერი კლასისა და ტიპის პეტრის ქსელებისთვის საერთოა და განისაზღვრება მხოლოდ ერთხელ.

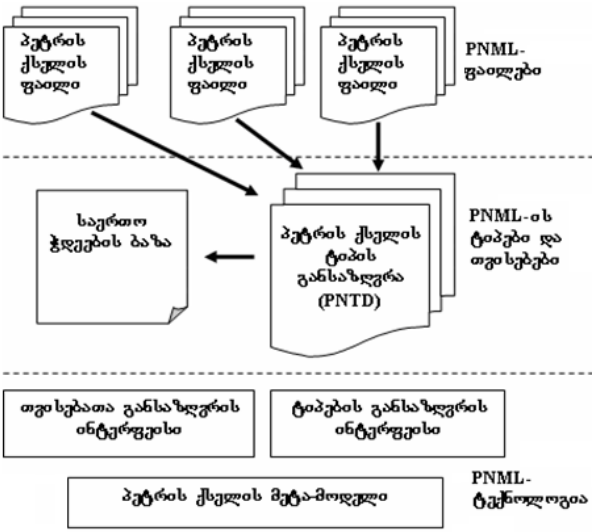
ამგვარ კომპონენტებს მივაკუთვნებთ პეტრის ქსელის **მეტა-მოდელს**, **თვისებათა განსაზღვრის ინტერფეისსა** და **ტიპების განსაზღვრის ინტერფეისს**.

შუალედურ დონეზე პეტრის ქსელების ენა პეტრის ქსელის ახალი ტიპების განსაზღვრის მექანიზმებს შეიცავს **საერთო ჭდეების ბაზითა** და **პეტრის ქსელის ტიპის განსაზღვრის** მექანიზმით, რომლებიც ერთმანეთთან მჭიდრო კავშირშია.

მესამე, გამოყენებითი დონე ქვედა დონეების საფუძველზე აგებულ პეტრის ქსელის სრულ ინფრასტრუქტურას, ე.წ. **პეტრის ქსელის ფაილებს** შეიცავს (ნახ.8.6). განვიხილოთ **PNML**-ის სტრუქტურული კომპონენტები დეტალურად.

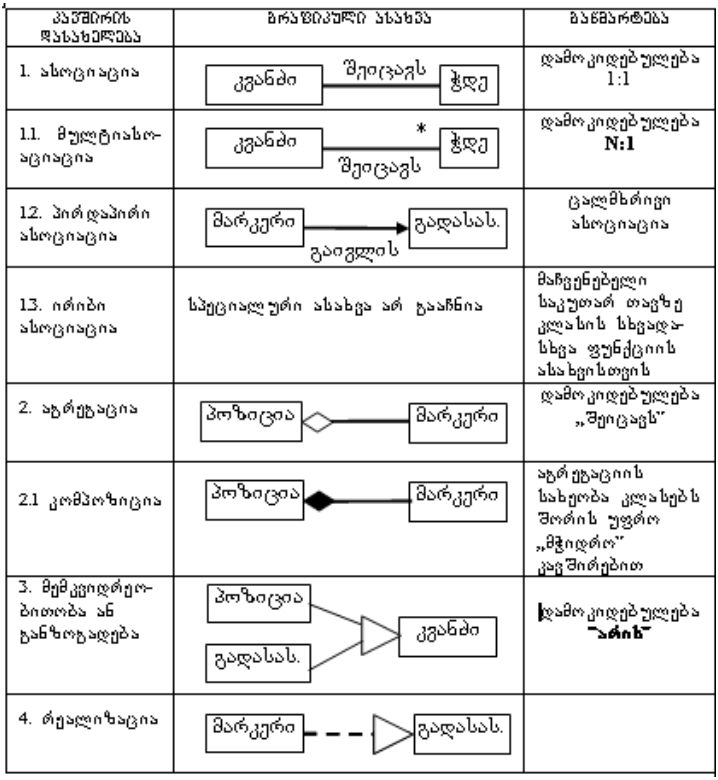
პეტრის ქსელის ფორმატირების ენის მეტა-მოდელი მის საწყის სტრუქტურას წარმოადგენს და შესრულებულია **UML**-ტექნოლოგიის გამოყენებით. **UML** იშიფრება როგორც **Unified Modelling Language** – უნიფიცირებული მოდელირების ენა.

იგი ობიექტ-ორიენტირებული იდეოლოგიის სტან-დარტიზაციის პროდუქტია და შეიცავს ობიექტ-ორიენტირებული მოდელირების სხვადასხვა საშუალე-ბათა სიმრავლეს 4 კლასში განაწილებული 9 სხვადასხვა ტიპის დიაგრამების სახით, რომლებსაც განსხვავებული ფუნქციონალური დატვირთვები გააჩნია [12].



ნახ.8.6. PNML ენის ზოგადი სქემა

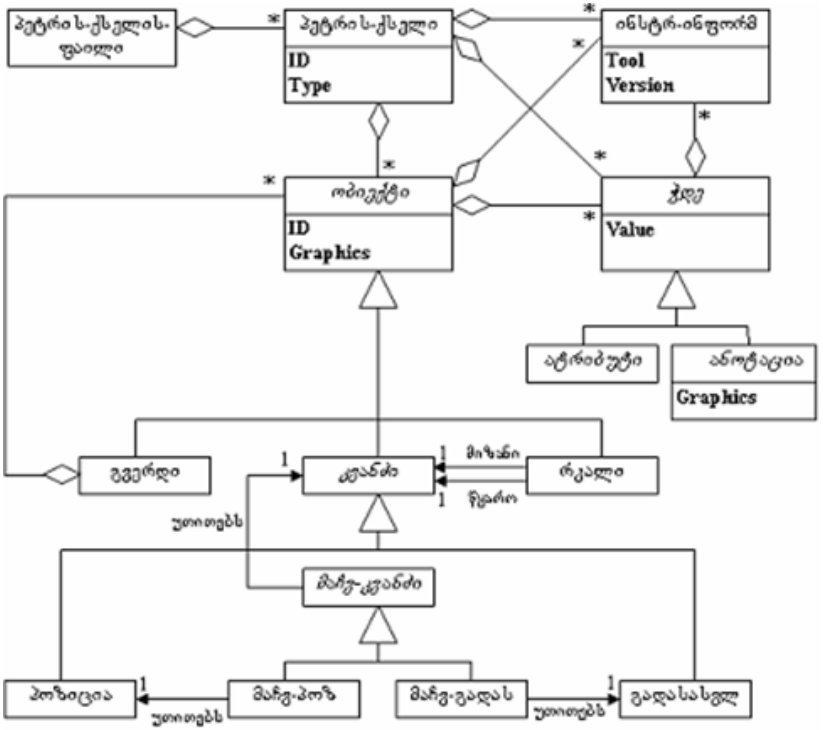
პეტრის ქსელის ენის მეტა-მოდელისთვის **კლასების დიაგრამაა** გამოყენებული, რომლის ელემენტებს **კლასები**, **ინტერფეისები** და მათ შორის დამოკიდე-ბულებები წარმოადგენს. კლასების დიაგრამაში კავშირთა ტიპები 8.7 ნახაზზეა მოცემული პეტრის ქსელების ელემენტთა მაგალითზე.



ნახ.8.7. კავშირის ტიპები კლასების დიაგრამაში

სიტუაციის მიხედვით შესაძლებელია სხვადასხვა კლასებს შორის ერთზე მეტი დამოკიდებულების არსებობა, რაც გვაქვს კიდეც კლასების დიაგრამაზე პეტრის ქსელების მეტა-მოდელისათვის (ნახ.8.8).

ზემოთ უკვე განვსაზღვრეთ პეტრის ქსელების მონიშვნათა ენის ცალკეული კომპონენტების შინაარსი და დანიშნულება, ახლა მათ შორის კავშირების სემანტიკას აღვწერთ.



ნახ.8.8. პეტრის ქსელების (PNML) მეტა-მოდელი

ყოველი **მარტივებელი პოზიცია** ან **მარტივებლი გადასასვლელი** წარმოადგენს მარტივებელ კვანძს და უთითებს შესაბამისად ერთ და მხოლოდ ერთ **პოზიციაზე** ან **გადასასვლელზე**.

პოზიცია და **გადასასვლელი** წარმოადგენს **კვანძს**, ხოლო ყოველი **რკალისთვის** არსებობს ერთი საწყისი (“**წყარო**”) და ერთი მიზნის (“**მიზანი**”) **კვანძი**.

გვერდები, **კვანძები** და **რკალები** წარმოადგენს პეტრის ქსელის **ობიექტებს**, ამასთან, **გვერდი** მოიცავს 1 ან მეტ ობიექტს (ანუ გვერდებს, კვანძებს და რკალებს).

ატრიბუტები და **ანოტაციები** არის **ჭდეები**. ობიექტი 1 ან მეტ ჭდეს, აგრეთვე პეტრის ქსელის კონკრეტული

ინსტრუმენტისთვის დამახასიათებელ ინფორმაციას შეიცავს (მინიმუმ ინსტრუმენტის დასახელებას და ვერსიას).

ყველაზე ზედა დონეს წარმოადგენს **პეტრის ქსელი**, რომელიც შედგება 1 ან მეტი ობიექტის (გვერდები, კვანძები და რკალები შესაბამისი ჭდეებით) და 1 ან მეტი საკუთარი ჭდისაგან.

და ბოლოს, **პეტრის ქსელის ფაილი** 1 ან მეტი პეტრის ქსელისგან შედგება. **ტიპების განსაზღვრის ინტერფეისი** ახალ პეტრის ქსელის ტიპს განსაზღვრავს, რომელიც მეტა მოდელის ფაილებიდან თავისთვის სასურველ სიმრავლეს იღებს.

თვისებათა განსაზღვრის ინტერფეისი პეტრის ქსელების ახალ თვისებებს განსაზღვრავს.

საერთო ჭდეების ბაზა შეიცავს პეტრის ქსელის სტანდარტულ ჭდეთა სიმრავლეს, რომლებსაც ნებისმიერი ტიპის პეტრის ქსელში უცვლელი სემანტიკა გააჩნია. საერთო ჭდეების ბაზა პეტრის ქსელის ჭდეების უმრავლესობის თავიდან განსაზღვრის პროცედურას არასაჭიროს ხდის. პეტრის ქსელის ახალი ტიპის განსაზღვრისას სტანდარტული ჭდეები შეიძლება საერთო ჭდეების ბაზიდან იქნეს აღებული, ხოლო სპეციფიკური ჭდეები პეტრის ქსელის ტიპის განსაზღვრის პროცედურით განისაზღვრება.

საერთო ჭდეების ბაზა დინამიკურად განახლებადი დოკუმენტია, რომელიც სპეციალისტთა ჯგუფის მიერ პეტრის ქსელების თეორიის განვითარებასთან ერთად ახალი სტანდარტული ჭდეებით შეიძლება შეივსოს.

პეტრის ქსელის ტიპის განსაზღვრა (PNTD – Petri Net Type Definition) წარმოადგენს პეტრის ქსელის ახალი ტიპის განსაზღვრისთვის აუცილებელ დირექტივათა ნაკრებს პეტრის ქსელის მონიშვნათა ენაზე.

პეტრის ქსელის ფაილი უკვე განსაზღვრული ტიპის პეტრის ქსელებს შეიცავს, რომლებიც თავიანთ კომპონენტებს შესაბამისი **PNTD**-ს მიხედვით აგებს.

9. პეტრის ქსელების ფორმატირების ენის სინტაქსი

9.1. XML – საფუძველი პეტრის ქსელის ფორმატირების ენისთვის

1998 წელს საერთაშორისო ორგანიზაცია **W3C**-ს მიერ ოფიციალურ სპეციფიკაციად დამტკიცების შემდეგ **XML**-ტექნოლოგია (**XML** იშიფრება, როგორც **Extensible Markup Language** – **მონიშვნათა გაფართოებადი ენა**) ინტერნეტის დაპყრობას აგრძელებს. სადღეისოდ არსებობს უამრავი ვებ-სერვერი, სადაც ინფორმაციის შესანახად და ასახვისთვის სწორედ **XML**-ს იყენებენ და შეიძლება ითქვას, ამ ტექნოლოგიამ ინტერნეტი თვისობრივადაც გარდაქმნა [25].

XML ჰიპერტექსტური მონიშვნების სისტემაა, ისევე, როგორც მისი წინამორბედი **HTML (Hyper Text Markup Language – ჰიპერტექსტების მონიშვნის ენა)**. ორივე ენა **სტანდარტული განზოგადებული მონიშვნების ენა SGML**-ის ქვეენას წარმოადგენს. ჰიპერტექსტის ცნება თეორიულად ჯერ კიდევ 1945 წელს ამერიკელმა მეცნიერმა ვანევარ ბუშმა დაამკვიდრა, ხოლო 60-იან წლებში ჰიპერტექსტებზე მომუშავე პროგრამებიც გამოჩნდა, რომლებიც ინფორმაციის არაწრფივ ასახვას და დამუშავებას უზრუნველყოფდა. ჰიპერტექსტური ტექნოლოგიის მასობრივი გავრცელება მხოლოდ მსოფლიო აბლაბუდის (**WWW – World Wide Web**) აგების პროცესში გახდა შესაძლებელი.

HTML-მა ინტერნეტის მსოფლიო ქსელად ქცევაში გადამწყვეტი როლი შეასრულა, მისი საშუალებით ინტერნეტში განთავსებული უზარმაზარი მოცულობის ინფორმაციის ძებნის პროცედურა მოწესრიგდა და გამარტივდა, მაგრამ ახალი ამოცანებისთვის საჭირო სიმძლავრეები **HTML**-ს არ გააჩნია. იგი წარმოადგენს სპეციალურ ინსტრუქციათა - **ტეგების** შეზღუდულ ნაკრებს, რომლებიც, როგორც წესი, ჩვეულებრივ ტექსტურ (**.HTML**) ფაილში ინახება.

პროგრამა-ბრაუზერები (**Internet Explorer, Netscape, Mozilla, Quanta, Opera**) წაიკითხავს **HTML**-ფაილს და მასში აღწერილი ტეგების მეშვეობით ასახავს ინფორმაციას

მომხმარებლის ეკრანზე. მაგრამ თანამედროვე ინტერნეტ-სერვისებისთვის პრინციპულ ამოცანად იქცა ინფორმაციის არამარტო ასახვა, არამედ სტრუქტურიზაციაც, რაც **HTML**-ს ფაქტობრივად არ შეუძლია. მასში მონაცემებს და ტეგებს ერთმანეთთან კავშირი არ გააჩნიათ, რაც ძლიერ ართულებს ინფორმაციის ანალიზს. მაგალითად, ინსტრუქცია:

```
<font color="red">Text</font>
```

ადვილი გასაგებია ნებისმიერი ბრაუზერისთვის, რომელიც ხვდება, რომ საჭიროა ტექსტის "**Text**" წითელი შრიფტით ასახვა, მაგრამ ამასთან მისთვის სულერთია, **HTML**-ფაილის რა ადგილას იქნება ზემორე ტეგი (****) განთავსებული, არის თუ არა იგი სხვა ტეგების ფრაგმენტი, ან თვითონ დაქვემდებარებულ ტეგებს თუ მოიცავს.

ფაქტობრივად, ინფორმაციის ძებნა და ანალიზი ჩვეულებრივი ტექსტური ფაილის ანალოგიურად უნდა მოხდეს, რაც ინფორმაციასთან მუშაობის არაეფექტურ მეთოდს წარმოადგენს.

HTML-ის მეორე ნაკლად მოუქნელობა უნდა ჩაითვალოს. მასში ტეგების სტანდარტული ნაკრები შეზღუდულია (მიუხედავად პერიოდული შევსებისა) და ბოლომდე ვერ პასუხობს სხვადასხვა ტიპის ინფორმაციულ მოთხოვნებს (მულტიმედია, მათემატიკური და ქიმიური ფორმულები და სხვა).

XML-მა ზემორე პრობლემები წარმატებით გადაჭრა. მასში არ არის წინასწარ განსაზღვრული ტეგების ნაკრები, **XML** წარმოადგენს საწყის ბაზისს (ნოტაციას) მომხმარებლის საკუთარი მონიშვნების ენისთვის [13]. **XML**-დოკუმენტიც **HTML**-ის მსგავსად ტეგების საფუძველზე აიგება, მაგრამ მისგან პრინციპულად განსხვავდება. თავისთავად **XML**-ფაილის სტრუქტურა გაურკვეველია სტანდარტული ინტერნეტ-ბრაუზერებისთვის, მისი ასახვისთვის სპეციალური პროგრამული უზრუნველყოფაა საჭირო (მაგალითად, შუალედური პროგრამა **XML**-დოკუმენტსა და ინტერნეტ-ბრაუზერს შორის).

XML-ს გააჩნია მონაცემთა საცავებთან მიმართვის შესაძლებლობები, სადღეისოდ ინტერნეტ-პროგრამირების თითქმის ყველა გავრცელებული სისტემა (**JAVA Script, VisualBasic Script, PHP, Perl** და სხვა სკრიპტული და არა მარტო

სკრიპტული ენები) ფლობს **XML**-დოკუმენტების დამუშავების საშუალებებს, რაც მანქანურ-დამოუკიდებელი პროგრამების დასაწერად საუკეთესო კომბინაციას წარმოადგენს და ინფორმაციის გაცვლის უნივერსალურ ფორმატს გვთავაზობს.

ამასთან **XML**-ში შესაძლებელია დოკუმენტებში იერარქიულად შენახული ინფორმაციის კორექტულობის კონტროლი, რომელიც სრულიად სხვადასხვა ტიპის მონაცემებისგან შეიძლება შედგებოდეს. ერთიანი კონტროლის ქონა ძალიან სასარგებლოა ინფორმაციული სისტემის შექმნის საწყის ეტაპზევე, რადგან ამით სისტემის სხვადასხვა კომპონენტების მიერ მონაცემთა სხვადასხვა ფორმატების გამოყენებით გამოწვეული შეუთავსებლობა იმთავითვე აღმოიფხვრება.

სკრიპტული ენები **XML**-დოკუმენტის დამუშავების-თვის დამატებითი პროგრამების (სკრიპტების) დაწერას მოითხოვს. ასახვის უფრო ეფექტური მეთოდია ეგრეთ წოდებული **სტილური ცხრილების გაფართოებადი ნაკრები XSL (Extensible Stylesheet Language)**, რომელიც **XML**-ის სპეციალურ ინსტრუქციათა ნაკრებს წარმოადგენს და შეიცავს **XML**-დოკუმენტის ფილტრაციის, მარტივი და რთული ძებნის და სხვა მრავალ ფუნქციას, რომლებიც **წესების** (**Rules**) მექანიზმზეა დაფუძნებული. **XSL**-ტექნოლოგია უკვე ჩანერგილია ყველა გავრცელებულ ინტერნეტ-ბრაუზერში.

XML-დოკუმენტების დამმუშავებელი პროგრამების (მარტივად რომ ვთქვათ, **XML**-ბრაუზერების) აგება და გამოყენება, როგორ წესი, შედარებით მცირე დროს და ხარჯებს მოითხოვს. ენის ნაკლოვანებებიდან პირველ რიგში მუშაობის დაბალი სიჩქარე და მეხსიერების მაღალი ხარჯი გამოირჩევა, რაც ოპტიმიზაციის სპეციალური მეთოდებით (**Compressed XML**) ნაწილობრივ გამოსწორებადია.

ზოგად შემთხვევაში **XML**-დოკუმენტი შემდეგ ფორმატს უნდა დაექვემდებაროს:

1. დოკუმენტის სათაურში ცხადდება თვით **XML**-დოკუმეტი ვერსიის ნომრითა და სხვა დამატებითი ინფორმაციით;

2. ყოველ „გამღებ" ტეგს **აუცილებლად** უნდა მოჰყვებოდეს „დამხურავი" ტეგი (**HTML**-ში ამის აუცილებლობა არ არის).

3. **XML**-ში სიმბოლოთა რეგისტრი განირჩევა;

4. ატრიბუტების მნიშვნელობები ყოველთვის ბრჭყალებში თავსდება.

5. ტეგების „ჩადგმულობა" (იერარქია) მკაცრად კონტროლირებადია, ამიტომ "გამღები" და "დამხურავი" ტეგების მიმდევრობას გადამწყვეტი მნიშვნელობა აქვს.

საწყის და საბოლოო ტეგებს შორის განთავსებულ **მთელ** ინფორმაციას **XML** განიხილავს, როგორც მონაცემებს, ამიტომ **HTML**-ისგან განსხვავებით გაითვალისწინება ფორმატირების ელემეტებიც (ცარიელი სიმბოლო, მომდევნო სტრიქონის ნიშანი, ტაბულაცია და სხვა).

თუ **XML**-დოკუმენტი ზემორე წესებს არ არღვევს, მას **ფორმალურ-სწორი** (სინტაქსურად სწორი) დოკუმენტი ეწოდება.

ენის კონსტრუქცია. ზოგადად, **XML**-დოკუმენტი ორი მთავარი კომპონენტის, **მონიშვნის ელემენტებისა** (**Markup**) და **მონაცემების** (**Content**) ერთობლიობას წარმოადგენს. თუ განვავრცობთ, **XML**-დოკუმენტი შედგება **ელემენტების ნაკრების, PCDATA და CDATA-სექციების, ანალიზატორის დირექტივების, კომენტარების, სპეცსიმბოლოებისა** და **ტექსტური მონაცემებისგან.**

ელემენტი **XML**-დოკუმენტის სტრუქტურული ბლოკია. ყოველი არაცარიელი ელემენტი აუცილებლად შედგება საწყისი და საბოლოო ტეგების, აგრეთვე მათ შორის მოთავსებული მონაცემებისგან. მონაცემი შეიძლება წარმოადგენდეს უბრალო ტექსტს, დოკუმენტის ჩადგმულ ელემენტს, **PCDATA** ან **CDATA**-სექციას, დამმუშავებელ ინსტრუქციას ან კომენტარს – ანუ პრაქტიკულად **XML**-დოკუმენტის ნებისმიერ ნაწილს. ელემენტების ნაკრები **XML**-დოკუმენტის იერარქიულ სტრუქტურას განსაზღვრავს. ერთი ელემენტი **ფესვურია**, პროგრამა-ანალიზატორი დოკუმენტის წაკითხვას მისგან იწყებს.

თუ ელემენტი ცარიელია (მონაცემებს არ შეიცავს), მისი საწყისი და საბოლოო ტეგები ერთიანდება შემდეგი ფორმატით: **<ტეგი/>**

კომენტარი ამოიცნობა ფორმატით: **<!--კომენტარი-->**

ატრიბუტი ელემენტის თვისებებს განსაზღვრავს, მაგალითად, მას მნიშვნელობას ანიჭებს. იგი საწყის ტეგში აისახება ფორმატით:

<ტეგი ატრიბუტი=მნიშვნელობა>.

სპეცსიმბოლოები ენის კონსტრუქციას განსაზღვრავს და მონაცემთა ბლოკში მათი პირდაპირი ჩართვა დოკუმენტის სტრუქტურას არევს, ამიტომ მისი რიცხვითი ან სიმბოლური იდენტიფიკატორის გამოყენება ხდება საჭირო.

ანალიზატორების დირექტივები გამოისახება ფორმატით: **<?დირექტივა?>**

CDATA (Character Data) მონაცემთა არეა, რომელსაც ანალიზატორი განიხილავს, როგორც უბრალო ტექსტს მასში ინსტრუქციებისა და სპეცსიმბოლოების განურჩევლად, მაგრამ კომენტარე-ბისგან განსხვავებით, გამოიყენებს მას სხვადასხვა მიზნებით, მაგალითად, კლიენტ-პროგრამის შესას-რულებლად. **CDATA** ბლოკში ხშირად **JAVA**-სკრიპტის ინსტრუქციები თავსდება. ფორმატი:

<![CDATA] მონაცემები]]>.

PCDATA (Parseable Character Data) – ნებისმიერი ინფორმაცია, რომელთანაც პროგრამა-ანალიზატორს მუშაობა შეუძლია.

ელემენტების განსაზღვრისას მათი იერარქიაც უნდა განისაზღვროს. მშობელი ელემენტის განსაზღვრის ბლოკში მისი შიდა ელემენტები იქვე ფრჩხილებში აღიწერება სპეციალური არააუცილებელი სიმბოლოების თანხლებით: **„+", „*", „?",** რომელთაგან "+" უთითებს, რომ ქვეელემენტი ელემენტში რამდენიმე ეგზემპლარად შეიძლება შედიოდეს, "?" – ელემენტი ოფციონალურია (შეიძლება საერთოდ არ იყოს წარმოდგენილი), " * " – შიგა ელემენტი ან ელემენტთა მიმდევრობა წარმოდგენილია რამდენჯერმე ან საერთოდ გამოიტოვება.

ენის გრამატიკასთან **ფორმალური** შესაბამისობის გარდა დოკუმენტში შეიძლება (და სასურველია) **შინაარსის** კონტროლის საშუალებებიც იყოს ჩადებული, რომლებსაც ცალკე **პროგრამა-ანალიზატორები** (ან მათი სპეციალური მოდულები) ანუ **ვერიფიკატორები** ამუშავებენ (მათ სხვანაირად **პარსერებსაც** უწოდებენ, ინგლისური სიტყვის **parse** – "ანალიზი, გარჩევა", საფუძველზე).

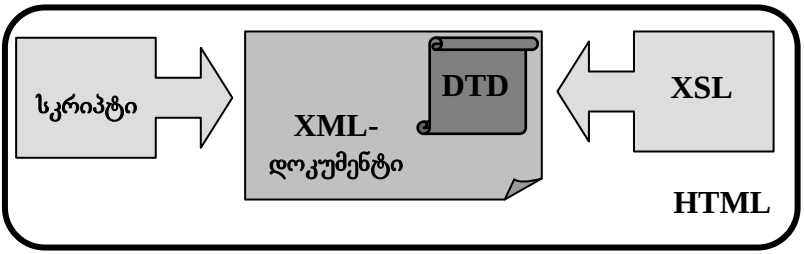
შინაარსის კონტროლი გულისხმობს **XML**-დოკუმენტის შესაბამისობის გარკვევას წინასაწარ განსაზღვრულ მონაცემთა სქემებთან.

სადღეისოდ **XML**-დოკუმენტის სისწორის შემოწმების ორი გავრცელებული მეთოდი არსებობს:

- **DTD**-განსაზღვრებები (**Document Type Definition** – დოკუმენტის ტიპის განსაზღვრა) და

- მონაცემთა სემანტიკური სქემა (**Semantic Schema**), რომლებიც შეიძლება **XML**-ფაილშივე ჩაისვას ან ცალკე ფაილის სახით გაფორმდეს (მაგალითად, **DTD**-სთვის გამოიყენება ტეგი **<!DOCTYPE "ფესვური ელემენტი" SYSTEM "DTD-ფაილის სახელი"/>).**

ზემოთ მოცემული აღწერილობიდან გამოდინარე, **XML**-დოკუმენტის (და შესაბამისად, **პეტრის ქსელების მონიშვნათა ენაზე** აღწერილი დოკუმენტის) კავშირი სხვა ინტერნეტ-ტექნოლოგიებთან 9.1 ნახაზზე მოცემული სქემით შეიძლება გამოისახოს.



ნახ.9.1. XML-დოკუმენტის დამუშავების სქემის ერთი ვარიანტი

ქვემოთ, 1-ელ ლისტინგში მოცემულია მარტივი **XML**-დოკუმენტის მაგალითი.

ფრაგმენტის პირველი ნაწილი დოკუმენტის ტიპების განსაზღვრის ინტერფეისია (**DTD**), რომელიც **XML**-დოკუმენტში გარე ფაილის სახით ჩაისმება.

დირექტივა:

```
<!DOCTYPE Saqartvelo SYSTEM "Saqartvelo.dtd">
```

განსაზღვრავს იმ ელემენტთა და მათი ატრიბუტების სიმრავლეს, რომლებიც შემდეგ **XML**-დოკუმენტში ნებადართული იქნება.

```
<!-- DTD-gansazRvrebebi -->
<!ELEMENT qvekana
          (fartobi,mosaxleoba,qalaqi)>
<!ATTLIST qvekana
Saxeli CDATA #REQUIRED>
<!ELEMENT fartobi (PCDATA)>
<!ATTLIST fartobi KvKm CDATA #REQUIRED>
<!ELEMENT mosaxleoba (PCDATA)>
<!ATTLIST mosaxleoba raodenoba CDATA
          #REQUIRED>
<!ELEMENT qalaqi (PCDATA,raioni)>
<!ATTLIST qalaqi id ID #REQUIRED>
<!ELEMENT raioni (PCDATA)*>
<!ATTLIST raioni id ID #REQUIRED>
<!-- XML-dokumenti -->
<?xml version="1.0" encoding="ISO-8859-1"
          standalone="no"?>
<!DOCTYPE Saqartvelo SYSTEM
          "Saqartvelo.dtd">
<qvekana Saxeli="Saqartvelo">
<fartobi KvKm="69700">afxazeTis da samxreT
          oseTis CaTvliT</fartobi>
<mosaxleoba raodenoba="3500000">faqtobrivi
          migraciis
          gaTvaliswinebiT</mosaxleoba>
<qalaqi id="1">Tbilisi
          <raioni id="1-
          1">Saburtalo</raioni>
          <raioni id="1-2">Vake</raioni>
</qalaqi>
<qalaqi id="3">Kutaisi</qalaqi>
<qalaqi id="4">Zugdidi</qalaqi>
</qvekana>
```

ლისტინგი-1. XML-დოკუმენტი გარე DTD-განსაზღვრებებით

პეტრის ქსელების მონიშვნის ენისთვის მისაღები ანალიზატორის შესარჩევად ჩვენ გავაანალიზეთ როგორც დე-ფაქტო სტანდარტებად მიღებული (**DTD**, მონაცემთა სემანტიკური სქემები), ასევე სხვა დამოუკიდებელი ანალიზატორები. საბოლოოდ არჩეულ იქნა ანალიზატორი **TREX (Tree Regular Expressions for XML),** რომელიც **DTD**-ზე ნაკლებ კომპაქტურია, სამაგიეროდ მოდულურობის თვისება გააჩნია, რაც მნიშვნელოვანია პეტრის ქსელების ერთიანი გაცვლითი ფორმატის შექმნისათვის [14]. მე-2 ლისტინგში მოცემულია **DTD**-განსაზღვრებათა ფრაგმენტი და მისი ექვივალენტური ფრაგმენტი **TREX**-ზე.

პეტრის ქსელის ტიპის ცნების განსაზღვრისა და **XML**-ენის ძირითადი ელემენტების განხილვის შემდეგ უკვე შეიძლება **PNML**-ენის სინტაკსის განსაზღვრა.

PNML-ს **(Petri Net Markup Language – პეტრის ქსელების მონიშვნის ენა)** განვმარტავთ, როგორც **XML**-ზე დაფუძნებულ პეტრის ქსელების დოკუმენტების აღწერის ენას, ანუ ყოველი **PNML**-დოკუმენტი იმავდროულად **XML**-დოკუმენტს წარმოადგენს.

შემდგომ პარაგრაფებში განვსაზღვრავთ პეტრის ქსელების ფორმატირების (მონიშვნათა) ენის სინტაკსს მისი სტრუქტურის შემადგენელი ნაწილებისთვის.

```
<!ELEMENT D (A | (B*, C)+)>
<!ELEMENT A EMPTY>
<!ELEMENT B EMPTY>
<!ELEMENT C EMPTY>
```

```
<grammar>
    <start>
        <element name="D">
            <choice>
                <ref name="A"/>
                    <oneOrMore>
                        <zeroOrMore>
                            <ref name="B"/>
                        </zeroOrMore>
                        <ref name="C"/>
                    </oneOrMore>
            </choice>
        </element>
    </start>
    <define name="C">
       <element name="C"><empty/></element>
    </define>
    <define name="B">
       <element name="B"><empty/></element>
    </define>
     <define name="A">
       <element name="A"><empty/></element>
     </define>
</grammar>
```

ლისტინგი 2. მონაცემთა კლასების განსაზღვრა DTD და TREX-ზე

9.2. პეტრის ქსელების მეტა-მოდელის სინტაქსი

მეტა-მოდელის სინტაქსის ელემენტები (**PNML**-ის **გასაღებური** ელემენტები) 8.6 ნახაზზე აგებული კლასების დიაგრამის საფუძველზე განისაზღვრება. პირდაპირი შრიფტით გამოსახული კლასებისთვის („კონკრეტული კლასები") ექვივალენტური **XML**-ელემენტები 9.1 ცხრილშია მოცემული.

მეტა-მოდელის ტრანსლაცია PNML–ის ელემენტებში

ცხრ.9.1

კლასი	XML-ელემენტი	XML-ატრიბუტი
პეტრის-ქსელის-ფაილი	<pnml>	
პეტრის-ქსელი	<net>	id: ID type: anyURI
პოზიცია	<place>	id: ID
გადასასვლელი	<transition>	id: ID
რკალი	<arc>	id: ID source: IDRef(Node) target: IDRef(Node)
გვერდი	<page>	id: ID
მოდული	<module>	id: ID
მაჩვენებელი-პოზიცია	<referencePlace>	id: ID ref: IDRef(Place or RefPlace)
მაჩვენებელი-გადასასვლელი	<referenceTransition>	id: ID ref: IDRef(Transition or RefTrans)
ინსტრუმენტის-ინფორმაცია	<toolspecific>	tool: string version: string
მნიშვნელობა	<value>	
გრაფიკა	<graphics>	

პეტრის ქსელის ფაილის აღმნიშვნელი ელემენტი **<pnml>** პეტრის ქსელის ენაზე შედგენილი ყველა დოკუმენტის ფესვური ელემენტია, საიდანაც პროგრამა-ანალიზატორი დოკუმენტის ანალიზს იწყებს.

როგორც აღვნიშნეთ, პეტრის ქსელის დოკუმენტი შეიძლება ერთზე მეტი პეტრის ქსელის აღწერას შეიცავდეს, ყოველი

ცალკეული პეტრის ქსელი **<net>...</net>** ელემენტის ფარგლებშია მოქცეული.

კლასთა სახელები პეტრის ქსელის მეტა-მოდელის **კონკრეტული** ელემენტებისთვის მეტა-მოდელის სქემაზე (ნახ.8.8) პირდაპირი შრიფტით გვაქვს გამოსახული.

კონკრეტულ ელემენტებს მივაკუთვნებთ მათ, რომლებიც პეტრის ქსელის გრაფში კონკრეტულადაა წარმოდგენილი და რომლებიც პეტრის ქსელის შექმნის პროცესში წინასწარ განისაზღვრება: **პოზიციებს, გადასასვლელებს, რკალებს, გვერდებს, მაჩვენებელ-პოზიციებს** და **მაჩვენებელ-გადასასვლელებს**.

კურსივით გამოვსახავთ იმ ელემენტებს, რომლებსაც პეტრის ქსელში ან კონკრეტული გამოსახულება არ გააჩნია (**კვანძები, ობიექტები**), ან მათი განსაზღვრა დამატებით უნდა შესრულდეს **პეტრის ქსელის ტიპის განსაზღვრის (PNTD)** მექანიზმით (**ჭდეები**).

კონკრეტულ ელემენტებს **PNML**-ში ექვივალენტური ელემენტები განესაზღვრებათ **(<place>, <transition>, <arc>, <page>)**. ამათგან პირველ ორს მხოლოდ იდენტიფიკაციის ატრიბუტი (**id**) გააჩნია, ისევე როგორც გვერდის ამსახველ ელემენტს, როცა **რკალის** ამსახველი ელემენტი **<arc>** ფლობს დამატებით ორ ატრიბუტს: **source** რკალის საწყის კვანძს განსაზღვრავს („წყარო"), **target** – საბოლოოს ("მიზანი"). **მაჩვენებელი პოზიცია** (**<referencePlace>**) და **მაჩვენებელი გადასასვლელი** (**<referenceTransition>**) შეიცავს სპეციალურ ატრიბუტს **ref**, რომელიც წარმოადგენს მაჩვენებელს პოზიციაზე ან გადასასვლელზე, ან მაჩვენებელს სხვა მაჩვენებლებზე.

ელემენტი **<toolspecific>** გამოსახავს კონკრეტული ინსტრუმენტის მომსახურე ინფორმაციას პეტრის ქსელის მოცემული ელემენტისთვის (პირველ რიგში მოიცემა ინსტრუმენტის დასახელება და ვერსია, ხოლო შემდეგ მხოლოდ მითითებული ინსტრუმენტისთვის დამახასიათებელი ქმედებები);

ელემენტი **<value>** პეტრის ქსელის კომპონენტის მნიშვნელობას ინახავს (მაგალითად, პოზიციის მარკირების ან რკალის ანოტაციის მიმდინარე მნიშვნელობებს).

მეტა-მოდელის განსაზღვრაში ვათავსებთ პეტრის ქსელების მონიშვნის ენის ყველა კონკრეტული ელემენტისთვის აუცილებელ

გრაფიკულ კომპონენტს (ელემენტი **<graphics>**), რომელიც თავის მომსახურე ელემენტებთან და ატრიბუტებთან ერთად ცალკე იქნება განხილული. მაგალითებში **PNML**-ის გასაღებური ელემენტები ხაზგასმით გვექნება გამოყოფილი.

9.3. ჭდეების განსაზღვრის სინტაქსი

PNML-ის ყველა ელემენტი, რომელიც მეტა-მოდელში არ არის განსაზღვრული (1-ელ ცხრილში არ არის მოცემული), განიხილება როგორც ჭდე ან ჭდეების კომბინაცია **PNML**-ის მოცემული ელემენტისთვის.

მაგალითად, **<initialMarking>** შეიძლება იყოს ჭდე პოზიციისთვის და მის საწყის მარკირებას ასახავდეს, **<name>** ობიექტის სახელის ამსახველი ჭდე იქნება, **<inscription>** - რკალის წარწერა.

ჭდე კომბინირებული ელემენტია და შეიძლება ქვეელემენტებს შეიცავდეს. მაგალითად, ჭდის მნიშვნელობა ინახება ელემენტში **<text>**, მაგრამ ამასთან, მისი გამოსახვა შეიძლება **XML**-ის ხითაც, თუკი მას რთული სტრუქტურა გააჩნია (ელემენტი **<structure>**).

PNML-ის ოფციონალური ელემენტი **<graphics>** განსაზღვრავს ჭდის გრაფიკულ პარამეტრებს.

ასევე ოფციონალურმა **<toolspecific>** ელემენტმა ჭდეს შეიძლება დამატებითი, კონკრეტული ინსტრუმენ-ტისთვის ინფორმაცია შეჰმატოს.

9.4. გრაფიკული ელემენტების განსაზღვრის სინტაქსი

PNML-ის გრაფიკული გარსი ელემენტ **<graphics>**-ის ფარგლებში აღიწერება და შედგება რამდენიმე ქვეელემენტისგან, რომლებიც 9.2 ცხრილშია მოცემული.

PMNL-ის გრაფიკული ელემენტები

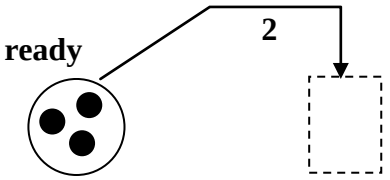
ცხრ.9.2

XML ელემენტი	ატრიბუტი	დომენი
<position>	x	decimal
	y	decimal
<offset>	x	decimal
	y	decimal
<dimension>	x	nonNegativeDecimal
	y	nonNegativeDecimal
<fill>	color	RBG-color
	image	anyURI
	gradient-color	RBG-color
	gradient-rotation	(vertical, horizontal, diagonal)
<line>	shape	(line, curve)
	color	RBG-color
	width	nonNegativeDecimal
	style	(solid, dash, dot)
<font>	family	CSS2-font-family
	style	CSS2-font-style
	weight	CSS2-font-weight
	size	CSS2-font-size
	decoration	(underline, overline, line-through)
	align	(left, center, right)
	rotation	decimal

<position> პეტრის ქსელის ელემენტის აბსოლუტურ კოორდინატებს განსაზღვრავს, **<offset>** - ელემენტის წარწერის დაშორებას ელემენტისგან. **<dimension>** წარმოადგენს კვანძის ზომებს, ხოლო **<fill>** – კვანძის გაფერადების პარამეტრებს ასახავს. **<line>** – რკალის მახასიათებელი ელემენტია მისი ფორმის, ფერის, სიგანის და სტილის პარამეტრებით და ბოლოს, ელემენტი **<font>** ანოტაციათა შრიფტის დაყენებას ემსახურება.

9.5. პეტრის ქსელის (PNML-) ფაილის მაგალითები

წინა პარაგრაფებში მოცემული პეტრის ქსელების **მეტა მოდელის, ჭდეებისა** და **გრაფიკული ელემენტების** განსაზღვრებათა ერთობლიობა საშუალებას გვაძლევს ავაგოთ „სუფთა" **PNML**-ფაილი მოცემული **P/T** (კლასიკური) კლასის პეტრის ქსელისთვის (ნახ.92).



ნახ.9.2. საილუსტრაციო პეტრის ქსელი PNML-ფაილის ასაგებად

```
<pnml>
      <net id="n1" type="PTNet">
          <name>
              <value>P/T-qselis nimuSi</value>
          </name>
            <place id="p1">
                  <graphics>
                      <position x="20" y="20"/>
                  </graphics>
                  <name>
                        <value>ready</value>
                  <graphics>
                        <offset x="10" y="-8"/>
                  </graphics>
                  </name>
                  <initialMarking>
                        <value>3</value>
                  </initialMarking>
            </place>
```

```
            <transition id="t1">
                  <graphics>
                     <position x="60" y="20"/>
                  </graphics>
  <toolspecific tool="PetriSim" version="1.1">
                        <hidden/>
  </toolspecific>
      </transition>
       <arc id="a1" source="p1" target="t1">
              <graphics>
                  <position x="30" y="25"/>
                  <position x="60" y="25"/>
              </graphics>
              <inscription>
                   <value>2</value>
                   <graphics>
                      <offset x="15" y="-2"/>
                   </graphics>
              </inscription>
       </arc>
    </net>
</pnml>
```

ლისტინგი 3. PNML-კოდი 22-ე სურათზე ასახული კლასიკური პეტრის ქსელისთვის

ამრიგად, ყოველი **PNML**-ფაილი იწყება ტეგით **<pnml>** და სრულდება ტეგით **</pnml>**. შემდეგი ტეგია კონკრეტული პეტრის ქსელის განმსაზღვრელი **<net>**, რომელიც შეიცავს 2 ატრიბუტს: უნიკალურ იდენტიფიკატორს **(id="n1")** და **პეტრის ქსელის ტიპის** დასახელებას.

პეტრის ქსელის პირველი ელემენტია **<**name**>,** რომელიც ქსელს სახელს ანიჭებს, რის შემდეგაც ქსელის კომპონენტების აღწერა იწყება.

ჩვენს მაგალითში გვაქვს პეტრის ქსელის სამივე საბაზო კომპონენტი: პოზიცია (**<**place**>**), გადასასვლელი (**<**transition**>**) და რკალი (**<**arc**>**), რომლებსაც იდენტიფიკაციის აუცილებელ ატრიბუტებთან ერთად საკუთარი (მაგალითად, გრაფიკული) ატრიბუტებიც გააჩნია.

კერძოდ, პოზიციისთვის ფრაგმენტი :

```
<graphics>
    <position x="20" y="20"/>
</graphics>
```

უჩვენებს, რომ პოზიციის გრაფიკული გამოსახულება (უფრო ზუსტად, გამოსახულების ანუ რგოლის ცენტრი), მოთავსებულია დეკარტეს კოორდინატთა სისტემის (20,20) კოორდინატზე, ხოლო ფრაგმენტი :

```
<name>
   <value>ready</value>
   <graphics>
      <offset x="10" y="-8"/>
   </graphics>
</name>
```

განსაზღვრავს პოზიციის სახელს ("**ready**") და მის დაშორებას პოზიციის რგოლის ცენტრიდან (ელემენტი **<offset>**). და ბოლოს, ფრაგმენტი:

```
<initialMarking>
    <value>3</value>
</initialMarking>
```

პოზიციის საწყის მარკირებას განსაზღვრავს.

იგივე პრინციპით განისაზღვრება გადასასვლელიც, ჩვენი მაგალითისთვის მას ემატება მხოლოდ ინსტრუმენტის სპეციფიკური ინფორმაცია (ტეგი <**toolspecific**>), რომელშიც განსაზღვრული გვაქვს პეტრის ქსელის ფიქტიური ინსტრუმენტი **PetriSim** თავისი ვერსიის ნომრით.

სპეციფიკური ინფორმაციის განყოფილება პეტრის ქსელის სიმულატორთა დამპროგრამებლებისთვისაა გამოყოფილი მხოლოდ კონკრეტული ინსტრუმენტისთვის საჭირო მოქმედებათა შესასრულებლად.

ჩვენს მაგალითში ფიქტიური ინსტრუმენტი **PetriSim** გადასასვლელის დამალვის **<hidden/>**-ელემენტს შეიცავს (შესაბამისად, გადასასვლელი სურათზე წყვეტილი მართკუთხედითაა გამოსახული).

რკალის განსაზღვრას ორი დამატებითი ატრიბუტიც თან ახლავს:

source – რკალის საწყისი კვანძის უნიკალურ იდენტიფიკატორს შეიცავს,

target – საბოლოოსას.

საყურადღებოა, რომ ჯერჯერობით რკალის განსაზღვრა არ შეიცავს ინფორმაციას რკალის ტიპის შესახებ. რკალის **ტიპი** განისაზღვრება სპეციალური უხილავი ჭდის საშუალებით, რომელსაც **რკალის ატრიბუტს** ვუწოდებთ (იხილეთ პეტრის ქსელის მეტა-მოდელის აღწერის 9.2 პარაგრაფი) და რომელიც რკალის გრაფიკულ ფორმას განაპირობებს.

მისი ექვივალენტური ატრიბუტი **PNML**-ის რკალის განსაზღვრის არეში იქნება **type**. ქვემოთ მოცემული ფრაგმენტი რკალის ტიპს ატრიბუტის დამატებით განსაზღვრავს:

```
<arc id="a1" place="p1" transition="t1" type="in">
```

მეორე ვარიანტში **<type>** განისაზღვრება, როგორც **<arc>**-ელემენტის ქვეელემენტი:

```
<arc id="a2" place="p2" transition="t2">
      <type value="in"/>
</arc>
```

შემაკავებელი რკალისთვის (**Inhibitor Arc**):

```
<arc id="a2" place="p2" transition="t2">
      <type value="inhibitor"/>
</arc>
```

ზემო 2 ფრაგმენტში ელემენტი <**type**> ხაზგასმული არ არის, რადგან იგი არ შედის **PNML**-ის გასაღებური ელემენტების რიცხვში და **პეტრის ქსელის ტიპების განსაზღვრის** მექანიზმით დამატებით უნდა იქნეს განსაზღვრული (ისევე, როგორც საწყისი მარკირების ელემენტი **<initialMarking>** მე-3 ლისტინგიდან).

შედარებით კომპლექსური **PNML**-ფაილის მაგალითი მოგვყავს 8.3 ნახაზზე გამოსახული, მრავალგვერდიანი მწარმოებელ-მომხმარებლის სისტემისათვის. იგი მე-4 ლისტინგზეა აღწერილი.

განსხვავება მხოლოდ პოზიციების და გადასასვლელების სახელებში იქნება, სადაც ქართული დასახელებები ინგლისური ექვივალენტებითაა ჩანაცვლებული.

```
<pnml>
 <net id="n1" type="PTNet">
   <name>
      <value>Consumer-Produser System</value>
   </name>
   <page id="pg1">
   <name><value>Producer</value></name>
   <place id="p1">
   <name><value>Ready to produce</value></name>
<initialMarking><value>0</value></initialMarking>
   </place>
      <transition id="t1">
         <name><value>produce</value></name>
      </transition>
   <place id="p2">
   <name><value>Ready to deliver</value></name>
<initialMarking><value>1</value></initialMarking>
   </place>
      <transition id="t2">
         <name><value>deliver</value></name>
      </transition>
      <arc id="a1" source="p1" target="t1">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a2" source="t1" target="p2">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a3" source="p2" target="t2">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a4" source="t2" target="p1">
        <inscription><value>1</value></inscription>
      </arc>
    </page>
    <page id="pg2">
      <name><value>Delivery Channel</value></name>
      <referenceTransition id="rt1" ref="t2"/>
      <place id="p5">
        <name><value>Empty</value></name>
```

```
<initialMarking><value>1</value></initialMarking>
      </place>
      <place id="p6">
        <name><value>full</value></name>
<initialMarking><value>0</value></initialMarking>
      </place>
      <referenceTransition id="rt2" ref="t3"/>
      <arc id="a5" source="rt1" target="p6">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a6" source="p6" target="rt2">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a7" source="rt2" target="p5">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a8" source="p5" target="rt1">
        <inscription><value>1</value></inscription>
      </arc>
    </page>
    <page id="pg3">
      <name><value>Consumer</value></name>
      <place id="p3">
      <name><value>Ready to recieve</value></name>
<initialMarking><value>1</value></initialMarking>
      </place>
      <transition id="t3">
        <name><value>recieve</value></name>
      </transition>
      <place id="p4">
      <name><value>Ready to consume</value></name>
<initialMarking><value>0</value></initialMarking>
      </place>
      <transition id="t4">
        <name><value>consume</value></name>
      </transition>
      <arc id="a9" source="p3" target="t3">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a10" source="t3" target="p4">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a11" source="p4" target="t4">
```

```
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a12" source="t4" target="p3">
        <inscription><value>1</value></inscription>
      </arc>
    </page>
  </net>
</pnml>
```

ლისტინგი 4. PNML-ფაილი „მწარმოებელ-მომხმარებლის“ სისტემისთვის

მოცემულ **PNML**-ფაილში ელემენტი **<page>** გვერდის ასახვას ემსახურება და საკუთარი უნიკალური იდენტიფიკატორი გააჩნია. იგი სხვა ელემენტებიდან შეიძლება გამოვყოთ <**referenceTransition**>, რომელიც ერთი გვერდიდან მეორე გვერდზე განთავსებულ გადასასვლელზე მაჩვენებელს აღწერს.

რკალის ანოტაცია <**inscription**> განსაზღვრავს რკალის ანოტაციის ჭდეს და პეტრის ქსელის მონიშვნათა ენის გასაღებურ ელემენტებს არ მიეკუთვნება (უნდა განისაზღვროს **PNTD**-თი).

PNML-ენა **მოდულების** განსაზღვრის მექანიზმსაც შეიცავს, რომლის არსი 8.1 პარაგრაფში გადმოვეცით. პრაქტიკული რეალიზაციის თვალსაზრისით **მოდულს** განვიხილავთ, როგორც **დამოუკიდებელ პეტრის ქსელის** (**PNML**-) **ფაილს**, ხოლო მოდულთაშორის კავშირებს – პეტრის ქსელის ფაილებს შორის კავშირების სახით გამოვსახავთ, რომელთა რეალიზაცია **მაჩვენებლების** მექანიზმით ხდება. შესაბამისად, საჭიროა **PNML**-ის ახალი ელემენტების განსაზღვრა **PNML**-ფაილში სხვა, გარე **PNML**-ფაილების იმპორტისთვის.

მოდულს განსაზღვრავს გასაღებური ელემენტი **<module>**. იგი შედგება ინტერფეისისა და მოდულის ქსელისგან. **ინტერფეისს** ცალკე ელემენტი განესაზღვრება იგივე სახელით **<interface>**, ხოლო **მოდულის ქსელური** ნაწილი

<interface>...</interface>

ტეგების გარეთაა განთავსებული. ამგვარად აგებული მოდულის შესაბამისი **PNML**-ფაილი 8.4 ნახაზზე აგებული მოდულური პეტრის ქსელისთვის მე-5 ლისტინგშია ნაჩვენები. ადგილის დაზოგვის მიზნით ფაილიდან ჭდეებისა (საწყისი მარკირება, ანოტაციები) და გრაფიკული ელემენტები ამოღებულია.

```
<module id="M1">
  <name><value>M1</value></name>
  <interface>
    <importPlace id="p1"/>
    <exportPlace id="p2" ref="y"/>
  </interface>
  <referencePlace id="x" ref="p1"/>
  <transition id="t1"/>
  <transition id="t2"/>
  <place id="y"/>
  <arc source="x" target="t1"/>
  <arc source="t1" target="y"/>
  <arc source="y" target="t2"/>
  <arc source="t2" target="x"/>
</module>
```

ლისტინგი 5. PNML-ფაილი M1 მოდულისთვის

როგორც ლისტინგი ცხადჰყოფს, **მოდულის ინტერფეისისთვის** განისაზღვრება **იმპორტის პოზიცია p1** (ელემენტი <**importPlace>**), რომელიც გარემოდან მოდულში მონაცემთა იმპორტს ემსახურება და ექსპორტის პოზიცია **p2** (ელემენტი <**exportPlace>**), რომელიც მოდულის ქსელიდან გარემოში მონაცემთა ექსპორტს ასრულებს და უთითებს მოდულური ქსელის პოზიცია **y**-ზე. დავუშვათ, მოცემული მოდული შენახულია მიმდინარე კატალოგის **moduleM1.pnml** ფაილში და საჭიროა მისი ეგზემპლარების გამოყენება 8.5 ნახაზზე მოცემული მოდულური პეტრის ქსელის **PNML**-ფორმატში შესანახად. შესაბამის **PNML**-ფაილს მე-6 ლისტინგზე ნაჩვენები სახე ექნება.

```
<pnml>
  <net id="n1">
    <place id="p">
    <initialMarking>
      <value>1</value>
    </initialMarking>
    </place>
    <instance id="m1" ref="M1"
uri="file:moduleM1.pnml">
      <importPlace parameter="p1" ref="p"/>
    </instance>
    <instance id="m2" ref="M1"
uri="file:moduleM1.pnml">
      <importPlace parameter="p1" instance="m1"
ref="p2"/>
    </instance>
    <instance id="m3" ref="M1"
uri="file:moduleM1.pnml">
      <importPlace parameter="p1" instance="m2"
ref="p2"/>
    </instance>
  </net>
</pnml>
```

ლისტინგი 6. PNML-ფაილი მოდულური პეტრის ქსელისთვის

ელემენტი <**instance**> **მოდულის ეგზემპლარს** განსაზღვრავს. მისი ატრიბუტი **ref** მაჩვენებელია პეტრის ქსელის შესაბამის მოდულზე, ხოლო ატრიბუტი **uri (Unified Resource Identifier)** რესურსის, ამ შემთხვევაში პეტრის ქსელის მოდულის მისამართია.

იმპორტის პოზიციისთვის განისაზღვრება **პარამეტრი** (**parameter**) და **მაჩვენებელი** (**ref**) იმპორტის კვანძზე (<**importPlace parameter="p1" ref="p"/**>) ან მოდულის ეგზემპლარზე (<**importPlace parameter="p1" instance="m1" ref="p2"/**>), საიდანაც მოდულში მონაცემთა იმპორტი სრულდება.

9.6. პეტრის ქსელის ტიპის განსაზღვრა – PNTD

წინა პარაგრაფებში ჩვენს მიერ შემოთავაზებულ და განხილულ იქნა "სუფთა" **PNML**-ტექნოლოგია, რომელშიც ხორციელდება პეტრის ქსელების ფორმატირების ენის **გასაღებური ელემენტების** სინტაკსური კონტროლი, რითაც შესაძლებელი ხდება კორექტული საბაზო სტრუქტურის აგება პეტრის ქსელის ნებისმიერი ტიპისთვის.

მომდევნო ნაბიჯი უნდა იყოს ინფრასტრუქტურის შექმნა პეტრის ქსელის კონკრეტული ტიპის განსაზღვრისთვის, რაც **პეტრის ქსელის ტიპის განსაზღვრის მექანიზმით** (**PNTD – Petri Net Type Definition**) მიიღწევა.

ეს მექანიზმი აზრობრივად **XML**-ის სემანტიკური კონტროლის საშუალებებს (**DTD**, მონაცემთა სემანტიკური სქემები, **TREX**) ეფუძნება, რომლებსაც 2.4.1 პარაგრაფში შევეხეთ, ანუ **PNTD** წარმოადგენს საწყისი წესების სიმრავლეს (**გრამატიკას**) **სემანტიკურად სწორი პეტრის ქსელის (PNML-) ფაილების** ასაგებად.

გრაფიკულად ეს დამოკიდებულება 2.3.3 პარაგრაფში, 2.7 ნახაზზეა ნაჩვენები, სადაც **PNML**-ენის ზოგადი სტრუქტურაა მოცემული. პეტრის ქსელის ფაილები მიმართავენ **PNTD**-ს პეტრის ქსელის კონკრეტული ტიპისთვის ნებადართულ ჭდეებზე ინფორმაციის მისაღებად, რის შემდეგაც მოცემული ტიპის პეტრის ქსელთან მუშაობისას მხოლოდ ამ ჭდეებით მანიპულირება შეუძლიათ.

მე-7 ლისტინგი **TREX**-გრამატიკის მეშვეობით (იხილეთ პარაგრაფი 2.4.1) შედგენილი პეტრის ქსელის ტიპის განსაზღვრას ასახავს **P/T** (კლასიკური) პეტრის ქსელისთვის.

```
<grammar ns="http://www.informatik.hu-
berlin.de/top/pnml"
   xmlns="http://www.thaiopensource.com/trex">
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/pnml.trex"/>
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/conv.trex"/>
      <define name="NetType" combine="replace">
            <string>PTNet</string>
      </define>
      <define name="Place" combine="interleave">
          <optional>
              <ref name="InitialMarkingString"/>
          </optional>
      </define>
      <define name="Arc" combine="interleave">
            <optional>
                  <ref
            name="InscriptionString"/>
            </optional>
      </define>
</grammar>
```

ლისტინგი 7. P/T (კლასიკური) პეტრის ქსელის ტიპის განსაზღვრა

TREX-გრამატიკის სინტაქსი **XML**-ორიენტირებულია (**DTD**-სგან განსხვავებით). მისი ფესვურია ელემენტია **<grammar>**, რომლის ატრიბუტი **ns** (**XML**-ის გასაღებური სიტყვიდან **namespace**) **PNTD**-დოკუმენტის წყაროს განსაზღვრავს (მის მისამართს ინტერნეტში), ხოლო **xmlns (XML namespace)** – ანალიზატორის მისამართს.

ელემენტი **<include>** ორჯერ გამოიყენება: პირველად პეტრის ქსელის მეტა მოდელის ფაილის, ხოლო მეორედ – საერთო ჭდეთა ბაზის (იხ.§-2.4.3) იმპორტისთვის ინტერნეტის მისამართებიდან.

ელემენტ **<define>**-ით ახალი ჭდე შემოიტანება სხვადასხვა ოფციების თანხლებით. მაგალითად, ფრაგმენტი:

```
<define name="NetType" combine="replace">
    <string>PTNet</string>
</define>
```

განსაზღვრავს ჭდეს **P/T** (კლასიკური) პეტრის ქსელისთვის, რომელიც შეიძლება **PNML**-ფაილში იქნეს იმპორტირებული, ისე, რომ ფაილში ქსელის განსაზღვრის საწყის ჭდეს (რომელიც ელემენტ **<net>-**ში განისაზღვრებოდა. იხ. ლისტინგი-3) ჩაანაცვლებს, რაზეც **combine**-ატრიბუტის მნიშვნელობა "**replace**" მიუთითებს. ხოლო ფრაგმენტით:

```
<define name="Place" combine="interleave">
    <optional>
        <ref name="initialMarkingString"/>
    </optional>
</define>
```

განისაზღვრება პოზიციის ჭდე **P/T**-ქსელში, სადაც ატრიბუტის მნიშვნელობა **combine="interleave"** უთითებს, რომ **PNTD**-ს ჭდის განსაზღვრა კი არ ჩაანაცვლებს, არამედ შეუჯერდება **PNML**-ფაილში მოცემული ანალოგიური ჭდის განსაზღვრას. ესე იგი, თუ **PNML**-ფაილში პოზიცია განისაზღვრებოდა ფრაგმენტით (პეტრის ქსელების ენის მეტა-მოდელიდან):

```
<place id="p"></place>
```

ხოლო **PNTD**-ში დაემატა ჩვენს მიერ ზემოთ მოცემული განსაზღვრება, მაშინ მისი საბოლოო სახე იქნება:

```
<place id="p">
```

```
            <initialMarkingString>
                <value>1</value>
            </initialMarkingString>
        </place>
```

ამასთან ერთად, **PNTD**-ში მოცემული ელემენტი **<optional>** აზუსტებს, რომ ელემენტი ოფციონალურია, ანუ იგი შეიძლება ყველა პოზიციისთვის არ განისაზღვროს (რაც ზუსტად შეესაბამება პოზიციის მარკირების ლოგიკას).

PNTD-ში განსაზღვრული ჭდეების იმპორტი **PNML**-ფაილში შესაძლებელი რომ გახდეს, **PNML**-ის ფესვურ ელემენტ **<pnml>**-ს და ცალკეული პეტრის ქსელის საწყის ელემენტ **<net>**-ს შესაბამისი მაჩვენებლები უნდა ჩაემატოს, მაგალითად, ქვემოთ მოცემული ფრაგმენტის სახით:

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
<net id="n1" type="http://www.informatik.hu-
                    berlin.de/top/pntd/ptNetb">
```

9.7. საერთო ჭდეთა ბაზის სინტაქსი

საერთო ჭდეთა ბაზაში სტანდარტული **ჭდეები** განისაზღვრება, რომლებიც საწყის ეტაპზე პეტრის ქსელის არცერთ საბაზო ელემენტს არ ეკუთვნის.

საერთო ჭდება ბაზა, ფაქტობრივად, წარმოადგენს შეთანხმებას იმ ჭდეების შესახებ, რომლებსაც სხვადასხვა ტიპის პეტრის ქსელები შეიძლება შეიცავდეს.

განსაზღვრული ჭდეების გარკვეული ქვესიმრავლე შემდგომ **პეტრის ქსელის ტიპების განსაზღვრის მექანიზმს** (**PNTD**) მიეწოდება პეტრის ქსელის ტიპის მიხედვით. **PNTD**, თავის მხრივ, ამ ჭდეს პეტრის ქსელის კონკრეტულ ელემენტს მიამაგრებს და საჭიროების შემთხვევაში **PNML**-დოკუმენტს გადასცემს.

თუ **PNTD** იყენებს ჭდეებს საერთო ჭდეების ბაზიდან, მასში ბაზის ფაილის (მაგალითად, **conv.trex**) იმპორტი აუცილებელია. შემდგომ, ახალი ჭდის განსაზღვრისას შეიძლება გამოყენებულ

იქნას შესაბამისი მითითება საერთო ჭდეების ბაზაზე, როგორც ქვემოთმოყვანილ ფრაგმენტში:

```
<grammar ns="http://www.informatik.hu-
                                    berlin.de/top/pnml"
      xmlns="http://www.thaiopensource.com/trex">
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/pnml.trex"/>
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/conv.trex"/>
...
<define name="Place" combine="interleave">
      <interleave>
            <optional>
                  <ref name="conv:PTMarking"/>
            </optional>
            <optional>
                  <ref name="conv:Name"/>
            </optional>
      </interleave>
</define>
```

ფორმალურად, საერთო ჭდეების ბაზა შეიცავს ერთმანეთისგან მეტნაკლებად დამოუკიდებელი ჭდეების განსაზღვრებათა მიმდევრობას.

ყოველ ჭდეს უნიკალური **სახელი** ენიშნება. ამასთან ჭდის სახელი შეიძლება შესაბამისი **XML**-ელემენტის სახელს არ ემთხვეოდეს.

2.5 ცხრილში საერთო ჭდეების ბაზაში განსაზ-ღვრულ ჭდეთა სიმრავლის ქვესიმრავლეა მოცემული.

საერთო ჭდეების ბაზის ქვესიმრავლის აღწერა PNML-ით. ცხრ.2.5.

ჭდის სახელი	XML ელემენტი	დომენი (ტიპი)
Name	<name>	string
PTMarking	<initialMarking>	nonNegativeInteger
ENMarking	<initialMarking>	—
HLMarking	<initialMarking>	structured
PTCapacity	<capacity>	nonNegativeInteger
HLSort	<sort>	structured

ცხრილის პირველ სვეტში ჭდის სახელია გამოტანილი, მეორეში – შესაბამისი **XML**-ელემენტი. მე-3 სვეტი ჭდის მონაცემთა ტიპს ასახავს.

ზემოაღწერილ ჭდეთა დანიშნულება შემდეგია:

Name - წარმოადგენს მომხმარებლის განსაზღვრულ იდენტიფიკატორს ელემენტის (პოზიციის, გადასასვლე-ლის, რკალის) სემანტიკის აღსაწერად, რომლის მნიშვნელობა ჩვეულებრივი ტექსტური სტრიქონის ტიპისაა (**string**). მისი ასახვისთვის სპეციალური ელემენტი **<text>** გამოიყენება;

PTMarking - პოზიციების საწყისი მარკირებაა **P/T (Place/Transition)** ტიპის პეტრის ქსელებში, რომელიც მნიშვნელობებს ნატურალურ (არაუარყოფით) რიცხვთა სიმრავლიდან იღებს (**nonNegativeInteger**);

ENMarking - ელემენტარულ სისტემურ ქსელებში **(EN Petri Nets)** პოზიციების საწყისი მარკირებას აღწერს და მისი ტიპი არ არის განსაზღვრული, რადგან ელემენტარულ სისტემურ ქსელებში პოზიციათა მარკირება ან ცარიელია ან შეიცავს შავი რგოლის სახის მქონე მარკერს, რომელიც გრაფიკულად უნდა განისაზღვროს;

HLMarking არის **თერმი** პოზიციების საწყისი მარკირებისთვის მაღალი დონის ქსელებში, რომელსაც კომპლექსური სტრუქტურა გააჩნია და აღიწერება შესაბამისი ტიპით **structured**. იგი შეიცავს **<text>** ელემენტს რამდენიმე სტრიქონად ან **XML**-ხის განშტოებას ანოტაციის ელემენტში **<structure>.**

PTCapacity წარმოადგენს **P/T** ტიპის ქსელებში პოზიციის მოცულობის აღმწერ ანოტაციას, რომელიც ნატურალური რიცხვის ტიპისაა;

HLSort მაღალი დონის პეტრის ქსელების პოზიციებში არსებულ მარკერთა სახეობებს აღწერს. იგი კომპლექსური სტრუქტურის მქონე ელემენტია.

საერთო ჭდეთა ბაზაში თავისი ადგილი გააჩნია ჭდეებს გადასასვლელებისთვის (მაგალითად, გადასას-ვლელის გახსნის პირობისთვის), რკალებს და პეტრის ქსელების მონიშვნათა ენის სხვა ელემენტებს.

ზემოთ მოყვანილი ცხრილიდან ისიც ჩანს, რომ საერთო ჭდეების ბაზის სხვადასხვაა ჭდეები ზოგჯერ **XML**-ის ერთი და იმავე ელემენტით გამოისახება (ცხრილში ელემენტი **<initialMArking>** ჭდეებისთვის **PTMarking**, **ENMarking** და **HLMarking**), რაც ბაზის ლოგიკურ სტრუქტურას არ არღვევს, რადგან ნახსენები ჭდეები პეტრის ქსელის სხვადასხვა ტიპებისთვისაა აღწერილი და ერთდროულად არასდროს გამოიყენება.

საერთო ჭდეების ბაზის ფრაგმენტი რკალის ატრიბუტებისა და ანოტაციებისთვის მე-8 ლისტინგშია მოცემული.

```
<grammar      ns="http://www.informatik.hu-berlin.de/top/pnml"
              xmlns="http://www.thaiopensource.com/trex">

    <include href="http://www.informatik.hu-
              berlin.de/top/pnml/pnml.trex"/>

              <define name="InitialMarkingString">

                        <element name="initialMarking">
                                  <ref name="Annotation"/>
                        </element>
              </define>

              <define name="InscriptionString">
                        <element name="inscription">
                                  <ref name="Annotation"/>
                        </element>
```

```
        </define>

        <define name="ArcType">
            <element name="type">
                <attribute name="value">
                    <ref name="ArcTypeValues"/>
                </attribute>
            </element>
        </define>

        <define name="ArcTypeValues">
            <choice>
                <string>normal</string>
                <string>read</string>
                <string>inhibitor</string>
                <string>reset</string>
            </choice>
        </define>
</grammar>
```

ლისტინგი 8. საერთო ჭდეთა ბაზის ფრაგმენტი

10. ბიზნეს-პროექტების მართვის სისტემის მოდელირება პეტრის ქსელებით

პეტრის ქსელის თეორია გრაფო-ანალიზურ თეორიაზე დაფუძნებული კონცეფციაა და გამოიყენება დისკრეტული სისტემების პარალელური და განაწილებული ხასიათის პროცესების კვლევისა და მოდელირებისათვის. პეტრის ქსელით, მოდელების სისტემური ანალიზი იკვლევს ობიექტის დინამიკურ თვისებებსა და რესურსულ შესაძლებლობებს.

პეტრის ქსელის საშუალებით შესაძლებელია პრაქტიკულად ნებისმიერი სისტემის ყოფაქცევის მოდელირება, გამოკვლევა და ანალიზი [3,4].

პეტრის ქსელში ძირითად მახასიათებელს წარმოადგენს მოვლენისა და პირობის ცნებები. მოვლენა - პეტრის ქსელში გარკვეული მოქმედების განხორციელებაა, ხოლო პირობა - სისტემის მდგომარეობაა. მოვლენების განსახორციელებლად საჭიროა შესრულდეს მოვლენის ე.წ. წინაპირობები. მოვლენის შესრულების შემდეგ ხორციელდება მოვლენის ე.წ. შემდგომი პირობები. პეტრის ქსელში მოვლენები აისახება პოზიციებით, ხოლო პირობები გადასასვლელებით, მოვლენის წინა პირობები წარმოიდგინება შესაბამისი გადასასვლელის შემავალი პოზიციებით და მოვლენის შემდგომი პირობები - გამომავალი პოზიციებით.

ამგვარად, პეტრის ქსელის ძირითადი ელემენტებია პოზიციები (რგოლები), გადასასვლელები (მართკუთხედები) და მათი შემაერთებელი რკალები. პოზიციები შეიძლება შეიცავდეს ერთ ან მეტ მარკერს (შავი წერტილი ან რიცხვი), რაც აუცილებელი პირობაა პეტრის ქსელის ამუშავებისთვის. ამუშავება ნიშნავს ქსელის რომელიმე გადასასვლელის გახსნას, რაც ნებადართულია მხოლოდ მაშინ, როცა გადასასვლელის შემავალი ყველა პოზიცია მინიმუმ 1 მარკერს შეიცავს. შესაძლებელია ჯერადი რკალების

არსებობა და ამ დროს შემავალ პოზიციებში მარკერების რაოდენობა რკალების ჯერადობაზე ნაკლები არ უნდა იყოს.

სხვადასხვა პეტრის ქსელის ტიპის გამოყენების თვალსაზრისით, შექმნილია პეტრის ქსელების ინსტრუმენტები. პეტრის ქსელი, მისი ავტორის, გერმანელი დოქტორის კარლ ადამ პეტრის სახელს ატარებს.

პეტრის ქსელები გამოიყენება პარალელური პროცესების შესრულებისას საერთო გამოყენების მონაცემების სინქრონიზაციისა და კონფლიქტური სიტუაციების მართვისთვის.

მულტიპროცესორულ და განაწილებულ სისტემებში განსაკუთრებული მნიშვნელობა ენიჭება პროცესების ეფექტურად ორგანიზაციის საკითხს ჩიხური სიტუაციის აღმოსაფხვრელად. პროცესი ჩიხურია, თუ იგი ელოდება ისეთი ხდომილების შესრულებას, რომელიც არასოდეს არ მოხდება. ჩიხური პროცესების არსებობისათვის ოთხი აუცილებელი პირობა იქნა განსაზღვრული: ურთიერთგამორიცხვის (პროცესებს აქვს რესურსების მონოპოლური გამოყენების უფლება), დამატებითი რესურსების მოლოდინის (პროცესებს აქვს უკვე გამოყოფილი რესურსები, მაგრამ ელოდება დამატებითსაც), არაგადანაწილებადობის (პროცესებს არ შეიძლება ჩამოერთვას რესურსები მათ საბოლოო შესრულებამდე), წრიული მოლოდინის (არსებობს პროცესების ჯაჭვი, რომელშიც ყოველი პროცესი აბლოკირებს ერთ ან რამდენიმე რესურსს, რომლებიც ესაჭიროება ჯაჭვში მომდევნო პროცესს) [2, 17].

არსებობს პეტრის ქსელების სხვადასხვა ნაირსახეობა, მაგალითად:

- პირობით/ მოვლენითი;
- მიზეზ/შედეგობრივი;
- პრედიკატ/ ტრანზიტული;
- ფერადი;
- ალგებრული;
- მაღალი დონის;

- დაბალი დონის;
- გაფართოებული და ა.შ.

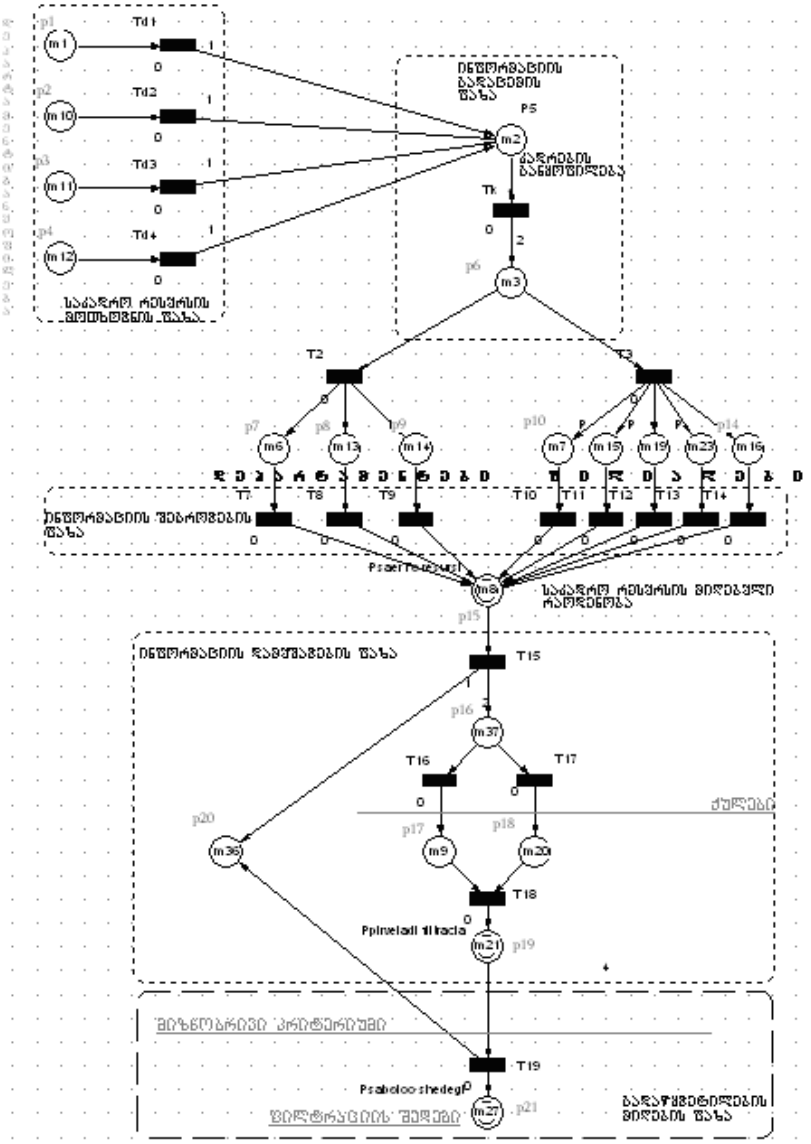
დაბალი დონის ანუ ელემენტარულ პეტრის ქსელში აისახება მხოლოდ სისტემაში მიმდინარე მოვლენები და არ შეიცავს ინფორმაციას ობიექტის რაოდენობრივი ან სხვა მახასიათებლების შესახებ, რეალური, დიდი სისტემების მოდელირებისთვის ელემენტარული პეტრის ქსელები მეტწილად არასაკმარისია.

ამ შემთხვევაში გამოყენებაშია სისტემური (გაფართოებული) პეტრის ქსელი, რომელიც ქსელის განზოგადებას გულისხმობს და მონაცემთა სტრუქტურებით მანიპულირებას უზრუნველყოფს. ტექნიკური ტერმინებით ეს ნიშნავს იმას, რომ დინამიკური ელემენტები (მარკერები) მათში არამარტო უბრალო წერტილებია, არამედ მონაცემთა გარკვეული სტრუქტურებიც. ამასთან, ერთი პრინციპით აგებული ქვექსელების სტრუქტურები ერთ სტრუქტურაშია გაერთიანებული [17].

მაგალითად, 10.1 და 10.2 ნახაზებზე ნაჩვენებია ელემენტარული პეტრის ქსელის ნიმუში, ბიზნეს-პროექტში სამუშო გუნდის სპეციალისტების მოთხოვნისა და საკადრო რესურსის შეფასების პროცესის მართვის ექსპერიმენტული კვლევისთვის.

10.1 ნახაზზე ასახული მდგომარეობებისა და გადასვლების ანუ პირობებისა და ოპერაციების აღმნიშვნელი მონაცემები ნაჩვენებია ცხრილში 10.1.

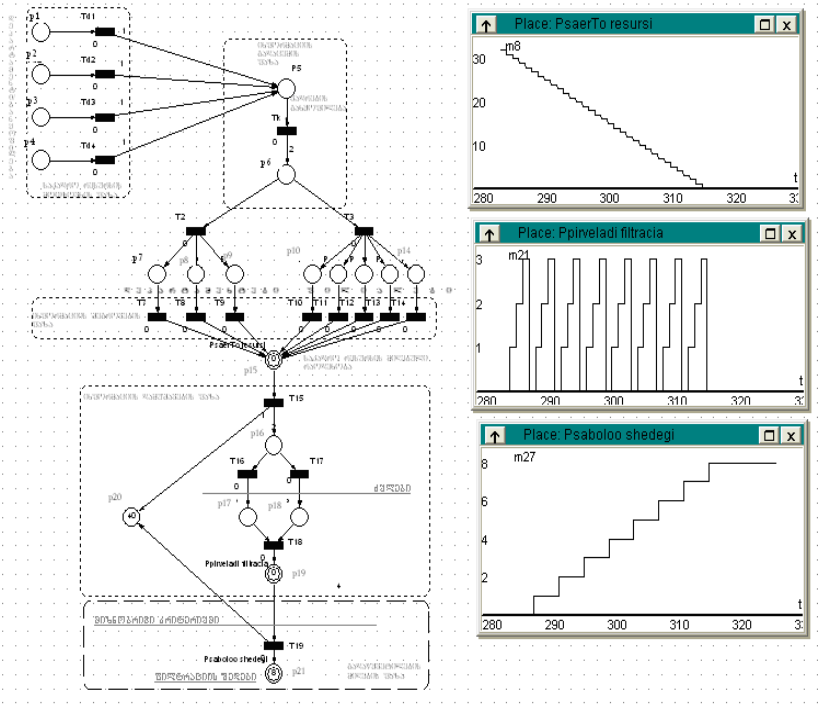
განხილულ პროცესში, დინამიკური კვლევის სიმულაცია შესაძლებლობას იძლევა გამოიკვეთოს შედეგის მიღების დროითი ეფექტურობა.



ნახ.10.1. ელემენტარული პეტრის ქსელის ნიმუში

პირობებისა და ოპერაციების აღმნიშვნელი მონაცემები ცხრ.10.1

მდგომა რეობები	მნიშვნელობა	გადასვლები	მნიშვნელობა
P_1,P_2, P_3, P_4	პროექტ-მენეჯერებისგან საკადრო რესურსის მოთხოვნის მომზადება	Td_1, Td_2, Td_3, Td_4	პროექტ-მენეჯერებისგან საკადრო რესურსის მოთხოვნის მომზადება
P_5	კადრების განყოფილება	Tk	საკადრო რესურსის პარამეტრების დამუშავება კრიტერიუმებისა და რაოდენობის გათვალისწინებით
P_6	კადრების განყოფილების თანამშრომლები	T_2, T_3	საკადრო რესურსის მოთხოვნაზე ინფორმაციის გავრცელება
P_7, P_8, P_9	სათაო ოფისის ქვეგანყოფილებები	T_7, T_8, T_9	სპეციალისტების მონაცემთა ანალიზი შეზღუდვებისა და ალტერნატივების გათვალისწინებით
P_{10}, P_{11}, P_{12}, P_{13}, P_{14}	კორპორაციის ქვედანაყოფები	T_{10}, T_{11}, T_{12}, T_{13}, T_{14}	სპეციალისტების მონაცემთა ანალიზი შეზღუდვებისა და ალტერნატივების გათვალისწინებით
P_{15}	მიღებული საერთო გასაფილტრი რესურსი	T_{15}	ფილტრაცია
P_{16}, P_{17}	პირველადი ფილტრაციის შედეგი	T_{16}, T_{17}	მიზნობრივი კრიტერიუმებით ფილტრაცია რეიტინგული შედეგების მიხედვით (იერარქიული ანალიზისა და წყვილ-წყვილად შედარების პროცედურით)
P_{18}, P_{19}, P_{20}	მიზნობრივი კრიტერიუმებით ფილტრაციის შედეგი	T_{18}	ფილტრაცია n-ჯერ უპირატესი პროცედურით
P_{21}	საბოლოო ფილტრაციის შედეგი	T_{19}	რანჟირება



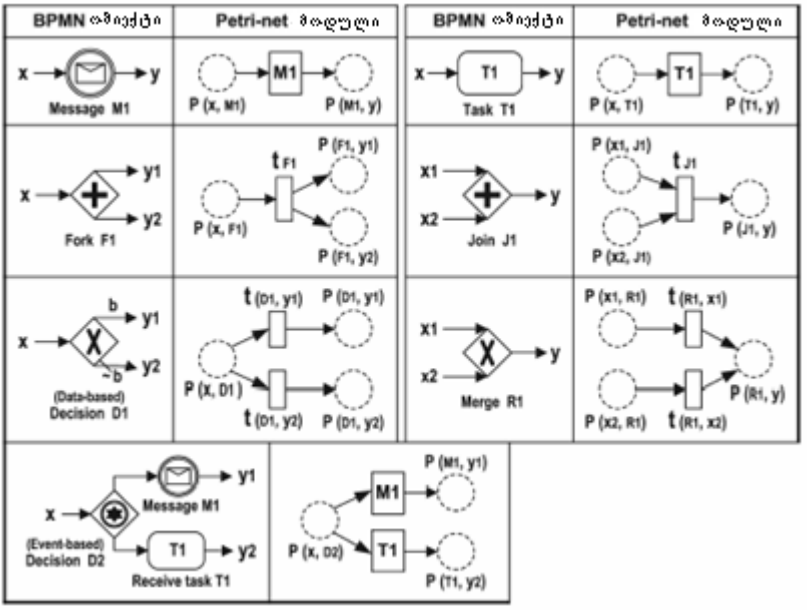
ნახ.10.2. დინამიკური კვლევის სიმულაცია ელემენტარული პეტრის ქსელის მიხედვით

სისტემური (გაფართოებული) პეტრის ქსელი რთული სისტემების ბიზნეს-პროცესების ფორმალური აღწერისა და დინამიკური მოდელის აგების მოქნილი საშუალებაა. კორპორაციული სისტემების ხასიათიდან გამომდინარე, რომელიც პირდაპირ აისახება ვებ-ტექნოლოგიური და სერვის-ორიენტირებული სტანდარტების განვითარებაზე, აუცილებელი გახდა პეტრის ქსელის მოდელების მოდერნიზაცია და დამუშავება უშუალოდ BPMN ნოტაციაზე ბაზირებული ბიზნეს-ნაკადების მართვის სისტემისთვის (მაგალითად, WorkFlowNet, BPEL2PN, WPDL - Workflow Process Definition Language, PNML-

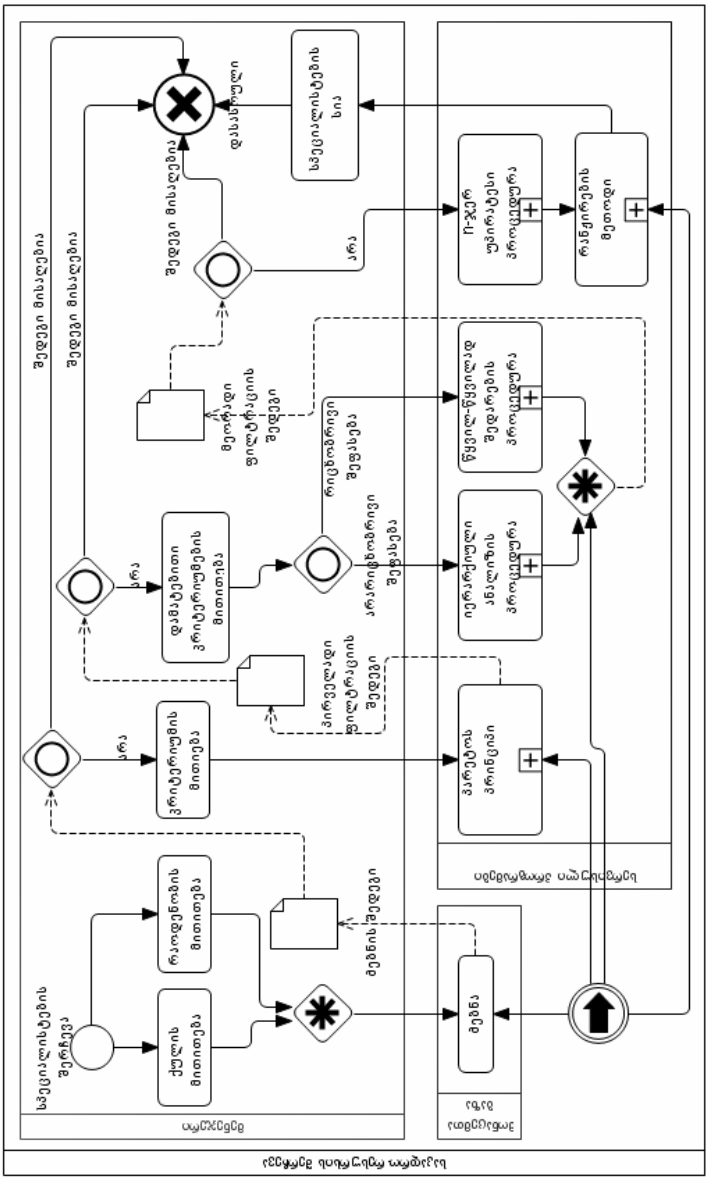
PetriNetMarkup Language, ენები). ასეთი სახის ენები ძირითადად შეიცავს პეტრის ქსელის სტანდარტულ კონცეფციებს: კვანძები, გადასვლები, პირობები და ა. შ. და დამატებით სრულყოფს მოდელების გრაფიკული წარმოდგენისა და ვიზუალიზაციის ასპექტებს [15, 16].

BPMN ნოტაციაში ბიზნეს-პროცესების დიაგრამის სემანტიკის განსაზღვრა ეფუძნება პეტრის ქსელების თეორიას, შემადგენელი ობიექტების ვიზუალური სახეცვლილებით.

10.3 ნახაზზე ნაჩვენებია BPMN ობიექტების ასახვა და ანალოგიები პეტრის ქსელის მოდულში, ხოლო 10.4 ნახაზზე წარმოდგენილია პეტრის ქსელზე ბაზირებული საპროექტო გუნდის დაკომპლექტების BPMN მოდელი.



ნახ.10.3. BPMN სტანდარტის ობიექტების ასახვა პეტრის ქსელის მოდულში



სურ. 10.4. პეტრის ქსელზე ბაზირებული საპროექტო გუნდის დაკომპლექტების BPMN მოდელი

11. ჩიხების აღმოფხვრის ალგორითმები

თავიდან ოპერაციული, ხოლო შემდეგ ქსელური სისტემების მოდელირების პროცესში ერთერთ მთავარ პრობლემას ჩიხური სიტუაციების წარმოშობა წარმოადგენს. განაწილებულ სისტემაში **ჩიხი** წარმოიშობა მაშინ, როცა ორი ქვესისტემა ერთდროულად **ურთიერთლოდინის** მდგომარეობაში იმყოფება ან კონფლიქტია რომელიმე რესურსისთვის.

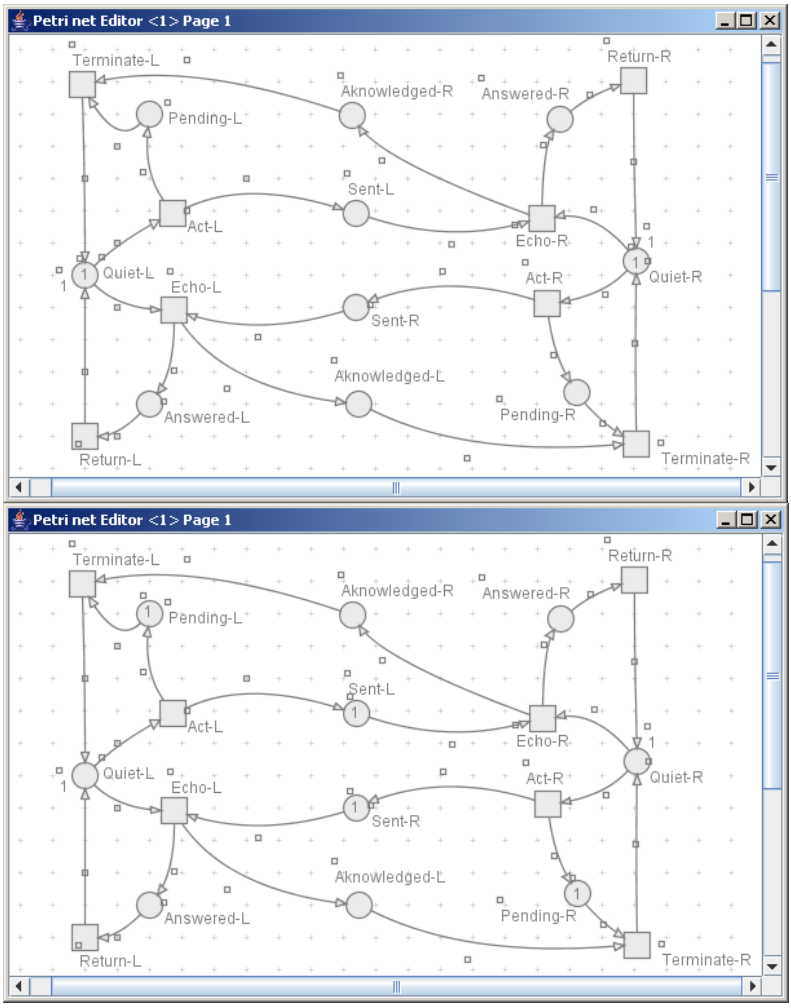
პირველი შემთხვევა სიმპტომატურია ოპერაციული სისტემების პროცესებისა და ქსელური პროტოკოლები-სთვის. 11.1 ნახაზზე მოცემულია ელემენტარული პეტრის ქსელი ტიპური შემთხვევისათვის (“ცნობების გაგზავნა-მიღების ამოცანა”).

აქ ორი პროცესი (ან ქსელის კვანძი) ერთმანეთს **ცნობებს** (**მესიჯებს**) უგზავნის ინფორმაციის გასაცვლელად, ამასთან ინფორმაციის მორიგი ულუფის გასაგზავნად აუცილებელი პირობაა **დასტურის** მიღება წინა შეტყობინების წარმატებით მისვლის შესახებ.

ინფორმაციის გაგზავნისა და დასტურის მიღების ოპერაციათა ერთობლიობას **რაუნდი** ვუწოდოთ. პროცესი (მაგალითად **L**) გზავნის **ცნობას** (გადასასვლელი **Act-L**) და გადადის **ლოდინის** მდგომარეობაში (პოზიცია **Pending-L**).

პროცესი **R** შეტყობინების მიღებისთანავე გამოდის პასიური მდგომარეობიდან (**Quiet-R**) და გზავნის მიღების **დასტურს** (გადასასვლელი **Echo-R**), რის შემდეგაც უბრუნდება **პასიურ** მდგომარეობას (გადასასვლელი **Return-R**).

დასტურის მოსვლა გაწყვიტავს **L**-პროცესის **ლოდინის** მდგომარეობას (გადასასვლელი **Terminate-L**) და დააბრუნებს მას **პასიურ** მდგომარეობაში (პოზიცია **Quiet-L**). ორივე პროცესის **პასიურ** მდგომარეობაში დაბრუნებით **რაუნდი** სრულდება.

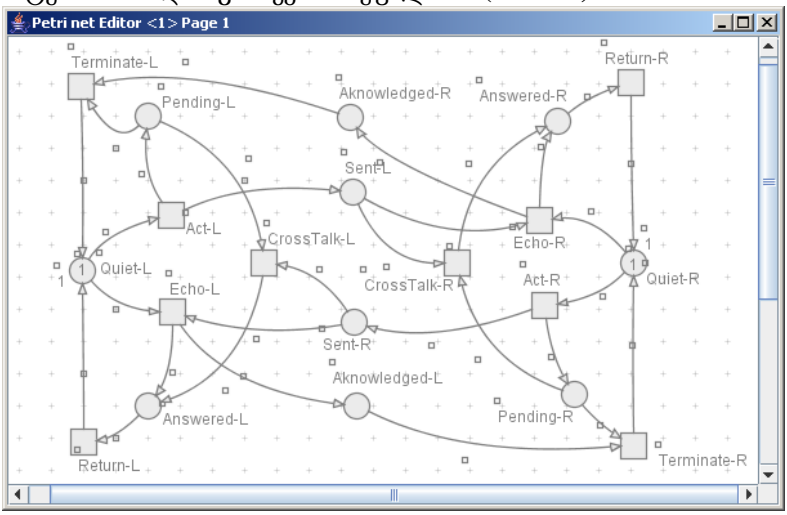


ნახ.11.1. ა) "ცნობათა გაგზავნა-მიღების" ამოცანის საწყისი მარკირება; ბ) ჩიხი

ამგვარ სისტემაში **ჩიხი** წარმოიშობა მაშინ, როცა ორივე აგენტი ერთსა და იმავე რაუნდში გადაწყვეტს ცნობის გაგზავნას (ნახ.11.1-ბ). ამ დროს ორივე პროცესი უსასრულოდ ელის

ადრესატისგან დასტურის მოსვლას (პეტრის ქსელის ყველა გადასასვლელი ბლოკირებულია).

ჩიხის თავიდან ასაცილებლად შემუშავებულია ალგორითმი **CrossTalk**, რაც პეტრის ქსელში 2 სპეციალური გადასასვლელის ჩამატებას გულისხმობს (თითო თითო პროცესისთვის), რომელთაც სისტემის ჩიხიდან გამოყვანა შეუძლიათ (ნახ.11.2).



ნახ.11.2. ჩიხის აღმოფხვრის CrossTalk-ალგორითმი

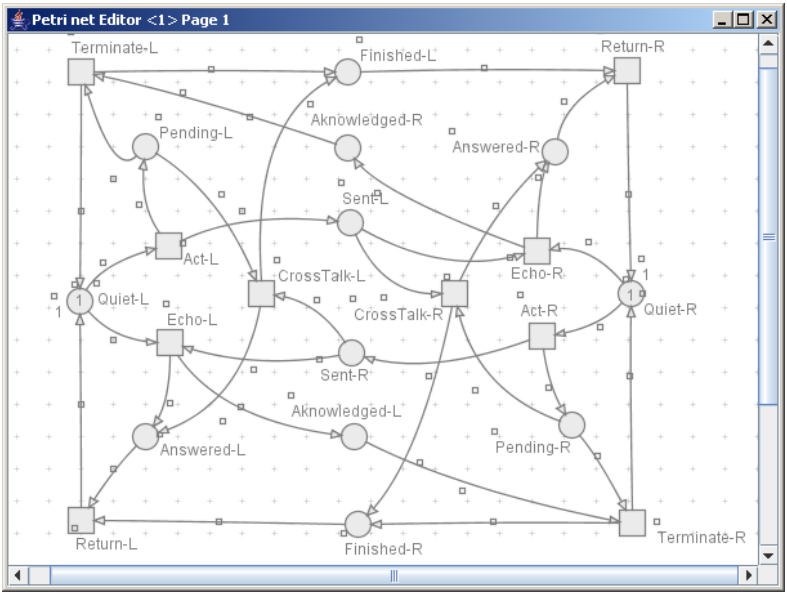
გადასასვლელი **CrossTalk** ერთ პროცესს აძლევს უფლებას მეორის მდგომარეობა შეამოწმოს და თავად ლოდინის მდგომარეობაში მყოფმა თუ დაადგინა, რომ მეორე პროცესიც იცდის, გაწყვიტოს ლოდინი და პირდაპირ დასტურის მიღების მდგომარეობაში გადავიდეს. დაპროგრამების ტერმინებით ეს ნიშნავს, რომ ლოგიკური ცვლადი, რომელიც პროცესის მდგომარეობის მნიშვნელობას ინახავს, ლოკალურიდან (მხოლოდ თავისი პროცესისთვის გამოყენებადი) გლობალურ ცვლადად უნდა გარდაიქმნას, რომელთან მიმართვა (და საჭიროებისამებრ მნიშვნელობის შეცვლაც) რამდენიმე პროცესს შეეძლება.

11.2 ნახაზზე მოცემული პეტრის ქსელიც "ცნობების გაგზავნა-მიღების" ამოცანის სრულყოფილ მოდელს ვერ წარმოადგენს. ქსელში ჩიხი უკვე გამოირიცხა, მაგრამ

სემანტიკურად იგი ჯერ კიდევ დასამუშავებელია. კერძოდ, რიგ სისტემებში (პირველ რიგში, კომპიუტერულ ქსელებში) მონაცემთა მორიგი პაკეტის გაგზავნა აკრძალულია მანამ, სანამ წინა პაკეტის წარმატებით მიღების შესახებ დასტური არ მოვა, ანუ მორიგი რაუნდის დაწყებამდე ყოველი წინა რაუნდი ბოლომდე უნდა იქნას მიყვანილი.

11.2 ნახაზზე ეს წესი ირღვევა (**რაუნდის შეცდომა**), მაგალითად, გადასასვლელთა შემდეგი მიმდევრობით გაშვებისას **"Act-L –> Echo-R –> Return-R –> Act-R –> CrossTalk-L".** ამ დროს პოზიციაში **"Sent-R"** მოხვედრილი მარკერი ეკუთვნის სისტემის ახალ (მეორე) რაუნდს (**R**-დან **L**-ში **ცნობის** გაგზავნა) და იგი აღწევს პროცეს **L**-ს მანამ, სანამ ეს უკანასკნელი პირველ რაუნდს დაასრულებდეს (ქვესისტემა **L პასიურ** მდგომარეობაში ჯერ კიდევ არ გადასულა ანუ **R**-იდან **დასტური** არ მიუღია).

პირველი რაუნდი დაუსრულებელი დარჩება, რაც პროცეს **R**-ის მიერ დაბრუნებული **დასტურის** ‘‘დაკარგვას’’ ნიშნავს. პრობლემის მოსაგვარებლად ქსელს ორი ახალი პოზიცია (**Finished-L** და **Finished-R)** ემატება, რომლებიც **"რაუნდის დასასრულის"** მდგომარეობას გამოსახავს (ნახ.11.3). აქ მოცემულ პეტრის ქსელში ახალი პოზიციები უკვე გამორიცხავს პროცესის მიერ ახალი რაუნდის დაწყებას ძველის დასრულებამდე, ამასთან მოცემული პეტრის ქსელი პროცესების **პარალელიზმსაც** ამოდელირებს: **L**- და **R**-პროცესებს ცნობების პარალელურად გაგზავნა-მიღება შეუძლია.



ნახ.11.3. რაუნდ-ორიენტირებული CrossTalk

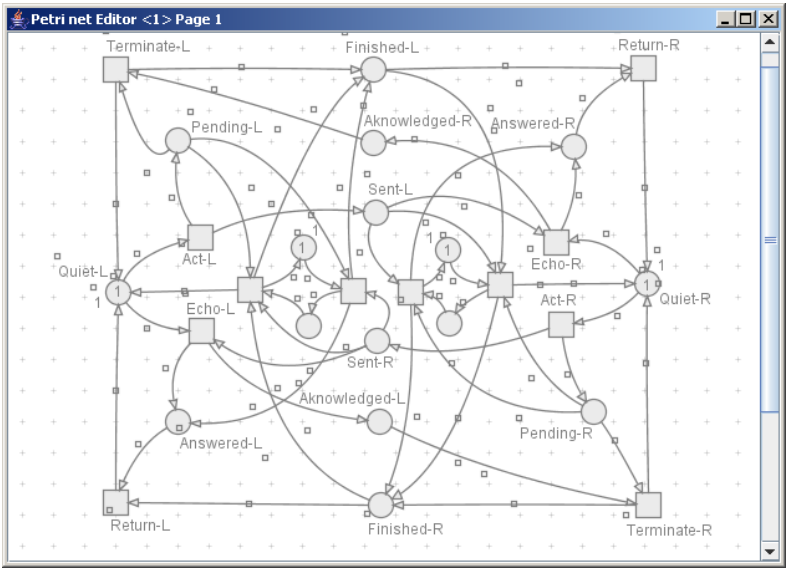
ამასთან, მოდელი ვერ წყვეტს **სინქრონიზაციის** ამოცანას, რომელიც დაისმება იმ შემთხვევაში, თუ პროცესები ცნობების გაგზავნა-მიღებისთვის არა 2 ურთიერთდამოუკიდებელ, არამედ ერთ, საერთო გადაცემის ფიზიკურ არხს იყენებს.

თუ ამ დროს ორივე პროცესი ინფორმაციას პარალელურად გაგზავნის, ისინი გზაში შეხვდება და ერთმანეთს დაამახინჯებს, ხოლო დამახინჯების ფაქტს 11.3 ნახაზზე მოცემული პეტრის ქსელი ვერ ასახავს, ამიტომ საჭიროა **პრიორიტეტების** შემოღება იმ მიზნით, რომ მოცემულ დროის მომენტში ინფორმაციის გადაცემა მხოლოდ ერთი პროცესისთვის იყოს შესაძლებელი.

მარტივ შემთხვევაში ერთი პროცესი მეორის მიმართ ცალსახად პრიორიტეტულია, ანუ ერთი პროცესის **CrossTalk**-გადასასვლელი გაიშვება მეორისაზე ადრე და პროცესის მიმდინარე რაუნდიც მეორე სისტემის შესაბამის რაუნდზე ადრე დამთავრდება.

უფრო რთული ვარიანტია პეტრის ქსელი ალტერნატიული პრიორიტეტებით, რომელიც 11.4 ნახაზზეა გამოსახული. ალტერნატიულ-პრიორიტეტებიანი პეტრის ქსელის მოდელი

ერთდროულად **პარალელიზმისა** და **სინქრონიზაციის** ამოცანებს წყვეტს, მხოლოდ იმ პირობით, რომ ქსელის მუშაობას **L**-პროცესი იწყებს.



ნახ.11.4. "ინფორმაციის გაგზავნა-მიღების" სისტემა ალტერნატიული პრიორიტეტებით

ორივე პროცესის მხრიდან **ცნობების** ერთდროული გაგზავნის შემთხვევაში **L**-პროცესი კავშირის არხის მონოპოლური მფლობელი ხდება და პირველი იყენებს ალტერნატიულ გადასასვლელს (**CrossTalk-L**) **R**-პროცესის საკონტროლო მარკერის მითვისების ხარჯზე (**Sent-R** პოზიციიდან), თავისი ბიჯის შესრულებისთანავე გადავა **Answered-L** პოზიციაში ("**R**-პროცესისგან **ცნობა** მიღებულია") და "აიძულებს" **R**-პროცესს **ცნობის** ხელახლა გაგზავნას (**Quiet-R** პოზიციიდან, რომელიც ამ ოპერაციების შედეგად მარკერს ხელახლა ღებულობს). ამ ჯერზე **L**-პროცესი ცნობას ჩიხური სიტუაციის გარეშე მიიღებს და რაუნდს დაასრულებს, რის შემდეგაც მმართველი ფუნქციას უკვე **R**-პროცესი აიღებს და მორიგი რაუნდი დაიწყება.

12. ურთიერთგამორიცხვის ალგორითმები

ურთიერთგამორიცხვის პრობლემა ყველაზე გამოკვეთილად ოპერაციული სისტემების, კომპიუტერული ქსელების და მონაცემთა ბანკების მართვის სისტემების დაპროექტებისას წარმოიშობა, როცა რამდენიმე პროცესი საერთო რესურსებს ინაწილებს. რესურსებთან ერთდროული მიმართვა ხშირად ჩიხურ სიტუაციებს წარმოშობს ან რესურსების არასასურველ განაწილებას განაპირობს.

ურთიერთგამორიცხვის ალგორითმებს **MUTEX**-ალგორითმები მათი ინგლისური სახელწოდების შემოკლებული ვარიანტის მიხედვით ეწოდება (**MUTual EXclusion** - "ურთიერთგამორიცხვა").

ალგორითმების არსის გასაგებად წარმოვიდგინოთ სისტემა, რომელიც შედგება ორი ქვესისტემისაგან (ან რომელშიც 2 პროცესი მუშაობს) - **L** და **R**. თითოეული პროცესის მოქმედება შემოზღუდულია სამ მდგომარეობაში ციკლური გადასვლებით. ესენია **პასიური, ლოდინისა და კრიტიკულ უბანზე მუშაობის** მდგომარეობები.

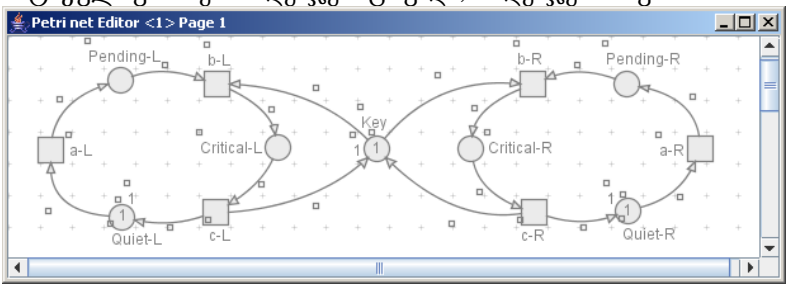
ამასთან, **პასიურიდან ლოდინის** მდგომარეობაში ორივე პროცესი ურთიერთდამოუკიდებ-ლად გადადის, ხოლო მდგომარეობაში **კრიტიკული უბანი** ორივე პროცესის ერთდროულად ყოფნა არ შეიძლება (**ურთიერთგამორიცხვის** თვისება).

ვერცერთი პროცესი ლოდინის მდგომარეობიდან პასიურში ისე ვერ დაბრუნდება, თუ კრიტიკული უბანი არ გაიარა (**ევოლუციურობის** თვისება).

არსებობს **MUTEX**-ის 2 საწყისი ალგორითმი რესურსების "უკიდურესად უსამართლო" და "უკიდურესად სამართლიანი" განაწილებისთვის. მათ მხოლოდ თეორიული ღირებულება გააჩნიათ:

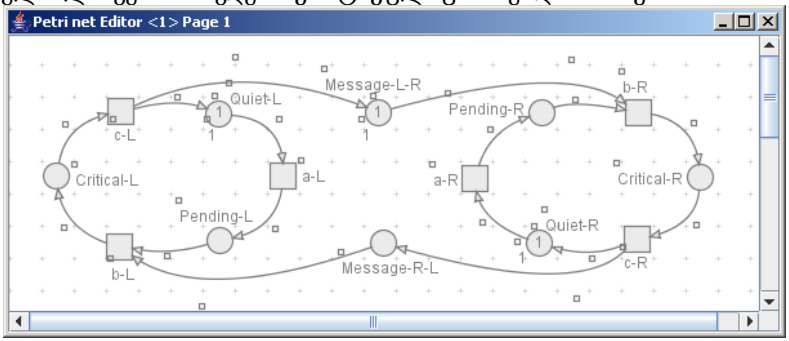
კონფლიქტური MUTEX-ალგორითმი (ნახ.12.1) ერთი პროცესისგან საერთო რესურსის მუდმივი მითვისების შესაძლებლობას ასახავს. სურათზე **key** რესურსისათვის პროცესებს **კონფლიქტი** მოსდით და მისი გადაჭრის საშუალებაც

არ ჩანს, ორივე პროცესს შეუძლია მიიტაცოს რესურსი და კრიტიკულ უბანზე რამდენჯერაც უნდა, იმდენჯერ იმუშაოს.



ნახ.12.1. კონფლიქტური MUTEX-ალგორითმი

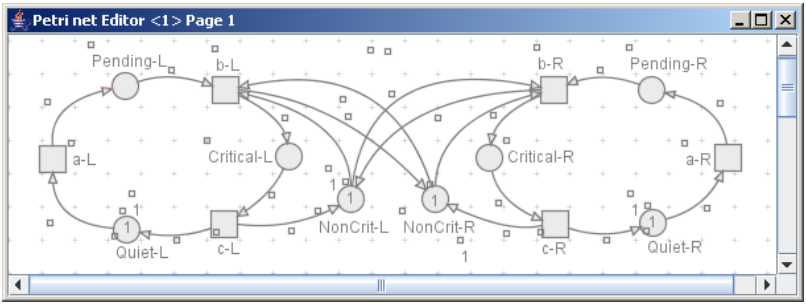
ალტერნატიულ MUTEX-ალგორითმში (ნახ.12.2) რესურსები ყოველთვის სამართლიანად ნაწილდება, ზედმეტად სამართლიანადაც. ერთი პროცესი კრიტიკული უბნიდან გამოსვლისას მეორეს **აცნობებს**, რომ კრიტიკულ უბანზე მუშაობა დაასრულა, რის შემდეგაც მეორე პროცესი **ვალდებულია** კრიტიკულ უბანზე იმუშაოს, სხვაგვარად პირველი პროცესი ხელახლა ვერ მოხვდება კრიტიკულ უბანზე და პირიქით.



ნახ.12.2. ალტერნატიული MUTEX-ალგორითმი

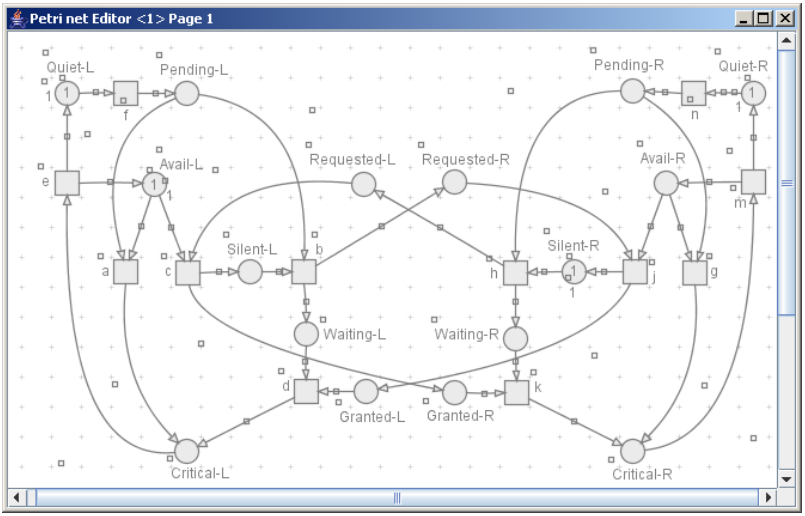
რეალური სისტემების უმრავლესობისთვის ამგვარი მიდგომა (ისევე, როგორც კონფლიქტი საერთო რესურსისთვის) მიუღებელია.

პრობლემა გადაწყდება, თუ ავაგებთ **მდგომარეობის შემოწმების MUTEX**-ალგორითმს (ნახ.12.3). მასში პროცესებს სპეციალური ალმების **noncrit-L** და **noncrit-R** საშუალებით ერთმანეთის მდგომარეობის შემოწმება შეუძლია და როცა კრიტიკული უბანი თავისუფალია, თითოეულ პროცესს მასში იმდენჯერ შეუძლია მოხვდეს, რამდენჯერაც უნდა.



ნახ.12.3. მდგომარეობის შემოწმების MUTEX-ალგორითმი

სამივე ზემოაღნიშნული ალგორითმი არასრულყოფი-ლია. გაცილებით მისაღები იქნება, პროცესებს ერთმანეთისაგან კრიტიკულ უბანზე მუშაობის უფლების მოთხოვნა და ნებართვის გადაცემა რომ შეეძლოს. ამგვარ სისტემას **მარკერული MUTEX**-ალგორითმი აღწერს (ნახ.12.4).



ნახ.12.4. მარკერული MUTEX-ალგორითმი

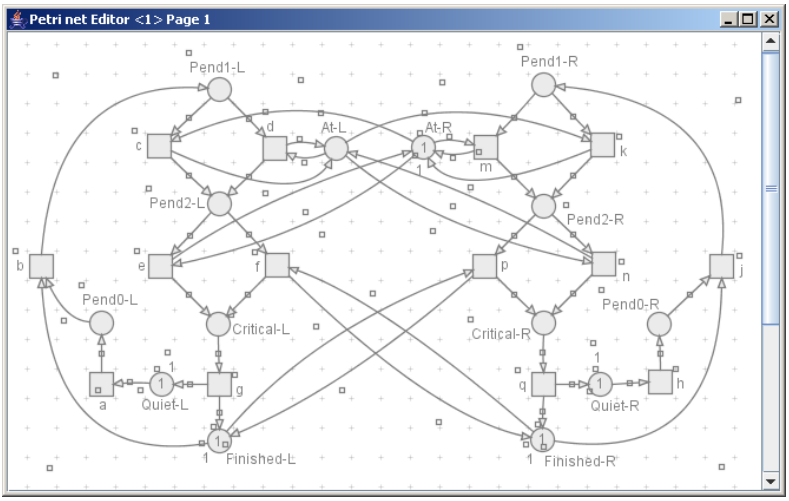
მარკერი კრიტიკულ უბანზე მუშაობის უფლებას წარმოადგენს, რომელსაც დროის მოცემულ მომენტში მხოლოდ ერთი პროცესი ფლობს (მარკერი პოზიციაში **Avail-L** ან **Avail-R**). პროცესს კრიტიკულ ზონაში მოხვედრა მხოლოდ მარკერის ფლობის შემთხვევაში შეუძლია, ხოლო თუ მარკერი არ გააჩნია, შეუძლია მეორე პროცესისგან მისი **მოთხოვნა** (პოზიცია **Requested-L** ან **Requested-R**), რომელიც მარკერის მფლობელმა პროცესმა შეიძლება **დააკმაკოფილოს** და თვითონ უმარკეროდ დარჩეს (პოზიცია **Granted-L** ან **Granted-R**) ან **არ დააკმაყოფილოს** და ისევ თვითონ გავიდეს კრიტიკულ უბანზე. 4.8 ნახაზზე მარკერს **L**-პროცესი ფლობს (პოზიცია **Avail-L**) და მოქმედების ორი ვარიანტი აქვს: **a** გადასასვლელით **კრიტიკულ** ზონაში მოხვდება (პოზიცია **Critical-L**) ან **c**-თი **პასიურ** მდგომარეობაში გადავა (**Silent-L**), რათა **R**-პროცესმა მოთხოვნის შემთხვევაში მარკერის მიღება შეძლოს. კრიტიკულ ზონაში მომუშავე **L**-პროცესი **e** გადასასვლელით საწყის მდგომარეობას დაუბრუნდება და ამასთან მარკერს **Avail-L** პოზიციაში აბრუნებს.

ამით ერთი ციკლი დასრულებულია, ხოლო მორიგ ციკლს ის პროცესი იწყებს, რომელსაც მარკერი არა აქვს (მარკერს მოითხოვს). წინა ციკლის შესრულების შედეგის მიხედვით მომთხოვნი შეიძლება იყოს ისევ **R**-პროცესი (**Pending-R**), თუ წინა ციკლში მარკერი არ გადაცემულა ან **L**-პროცესი (**Pending-L**) – თუ გადაიცა.

ურთიერთგამორიცხვის ამოცანის კიდევ უფრო მოქნილ ვარიანტს **პიტერსონის MUTEX**-ალგორითმი წარმოადგენს, სადაც რესურსის მისაღებად მზადმყოფი პროცესის ლოდინი სამ მდგომარეობადაა დანაწევრებული – მაგალითად, **L**-პროცესისთვის **Pend0-L**, **Pend1-L** და **Pend2-L** (ნახ.12.5). **Pend0-L** პოზიციას თეორიული ღირებულება არ გააჩნია, **პასიური** მდგომარეობიდან **(Quiet-L)** მასში გადასვლა ალგორითმის შესრულების საერთო სტრუქტურას არ არღვევს.

მაგრამ ამ მდგომარეობის არსებობა მოცემულ პეტრის ქსელში მაინც საჭიროა, რადგან მასში გადასვლით პროცესი გამოთქვამს **სურვილს** (და არა **პრეტენზიას**) კრიტიკულ ზონაში მუშაობის ნებართვაზე (ანუ მეორე პროცესს ჯერჯერობით ხელს არ უშლის), რაც რეალური სისტემების დაპროგრამებისას შეიძლება გახდეს საჭირო.

ალგორითმში 4 **ალამი**, იგივე საკვანძო პოზიციაა: **Finished-L** და **Finished-R (**რაუნდის **დასასრული),** **At-L** და **At-R (**კრიტიკულ ზონაში სამუშაო ნებართვის **მარკერები)**.



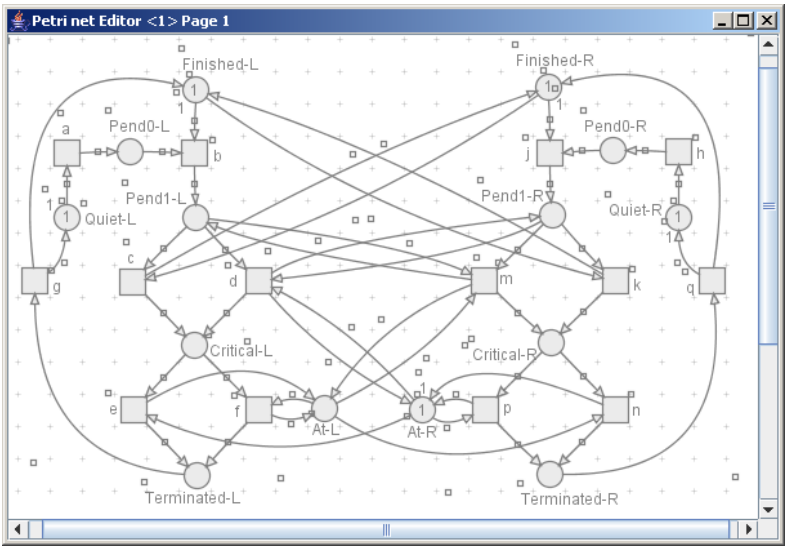
ნახ.12.5. პიტერსონის MUTEX-ალგორითმი

ამ ოთხ პოზიციაში მარკერების არსებობა განსაზღვრავს პროცესების **კრიტიკულ ზონაში** მოხვედრის მიმდევრობას. კერძოდ, თუ კრიტიკულ ზონაში მუშაობის სურვილს მხოლოდ ერთი, **L**-პროცესი გამოთქვამს (ანუ პოზიციაში **Finished-R** მარკერი შენარჩუნებულია), მაშინ **L**-პროცესი კრიტიკულ ზონას დაუბრკოლებლად აღწევს (გადასასვლელთა მიმდევრობა **a-b-c-f**), მაგრამ საკმარისია **R**-პროცესმა თვითონაც მოისურვოს კრიტიკულ ზონაში მუშაობა (გადასასვლელები **h-j**), რომ პროცესებს მოქმედებათა **სინქრონიზაცია** მოუწევს, ოღონდ სამართლიანად: პროცესები ერთმანეთს ვერ „გადაუსწრებს", რომელი პროცესიც მარკერს პირველი მოითხოვს, კრიტიკულ ზონაშიც პირველი მოხვდება იმ გარანტიით, რომ მისი კრიტიკული ზონიდან გამოსვლისთანავე მეორე პროცესიც მიიღებს მასში მოხვედრის მარკერს (უფლებას).

დეკერის MUTEX-ალგორითმი პიტერსონისას გარეგნულად საკმაოდ ჰგავს (ნახ.12.6).

ძირითადი შინაარსობრივი განსხვავება შემდეგშია: პიტერსონის ალგორითმში **At-L** პოზიცია მარკერს ღებულობს მანამ, სანამ **L**-პროცესი კრიტიკულ ზონაში მოხვდებოდეს, ხოლო

დეკერისაში - პირიქით, **L**-პროცესის კრიტიკულ ზონაში მოხვედრის შემდეგ.

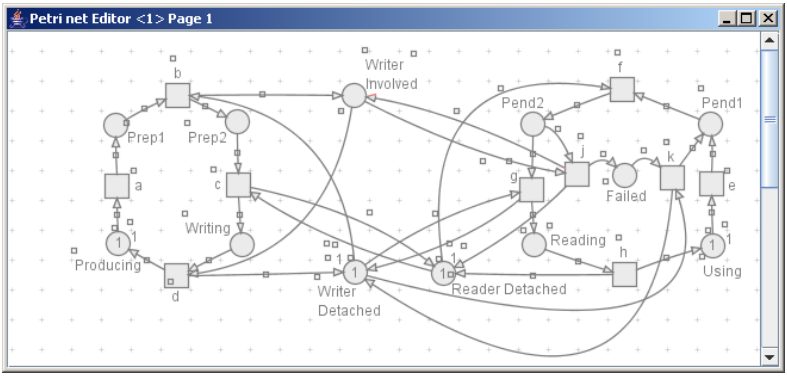


ნახ.12.6. დეკერის MUTEX-ალგორითმი

აქამდე განხილული ყველა ალგორითმი ორი **თანასწორი** პროცესის ურთიერთქმედებას აღწერდა.

რეალური სისტემების უმრავლესობაში პროცესებს **პრიორიტეტები** ენიჭებათ და საერთო რესურსებთან მიმართვისას მაღალპრიორიტეტიანი პროცესი უფლება-მოსილია დაბალპრიორიტეტიანი „გამოაძევოს“.

ოვიცკი / ლამპორტის MUTEX ალგორითმში ამგვარი სისტემის მოდელია წარმოდგენილი. მასში **განაწილებული ცვლადის წაკითხვა-განახლების** ოპერაციათა ურთიერთქმედება აისახება (ნახ.12.7).



ნახ.12.7. ოვიცკი/ლამპორტის MUTEX-ალგორითმი

სისტემა ორი პროცესისგან შედგება: **ჩამწერისა** (**Writer**) და **წამკითხველისგან** (**Reader**), რომლებიც საერთო მონაცემთა ცვლადის მნიშვნელობას ცვლის (ჩამწერი) ან ამოიკითხავს (წამკითხველი). **ჩამწერი** პროცესი პრიორიტეტულია.

ალგორითმი სამ ალამს იყენებს: **Writer Detached** (**ჩამწერი მოიხსნა**), **Writer Involved** (**ჩამწერი ჩაერია**) და **Reader Detached** (**წამკითხველი მოიხსნა**).

პირველი და მესამე ალამი შესაბამისად ჩამწერ და წამკითხველ პროცესებს მოხსნის კრიტიკული უბნიდან და მეორე პროცესს სამოქმედო გზას უხსნის, მეორე ალამი კი ჩამწერი პროცესის წამკითხველის მოქმედებაში ჩარევას ამოდელირებს.

იგი აღკვეთს **წამკითხველისგან** ცვლადის წაკითხვის ოპერაციას, თუ იმავდროულად **ჩამწერი** ცვლადის განახლებას დააპირებს.

პროცესების მიმდევრობა ასეთია: **ჩამწერს** მომზადებული აქვს ცვლადის ახალი მნიშვნელობა (პოზიცია **Producing**), რომელიც **a-b-c** გადასასვლელთა მიმდევრობის გახსნით უნდა გადაიტანოს კრიტიკულ ზონაში (პოზიცია **Writing**) და **განაწილებული ცვლადის** მნიშვნელობა განაახლოს.

დავუშვათ, წამკითხველმაც უკვე წაიკითხა და გამოიყენა განაწილებული ცვლადის წინა მნიშვნელობა (პოზიცია **Using**) და მოინდომა თავის კრიტიკულ ზონაში (პოზიცია **Reading**) **e-f-g**

გადასასვლელების გხსნის გზით გადასვლა განაწილებული ცვლადის ახალი მნიშვნელობის წასაკითხად.

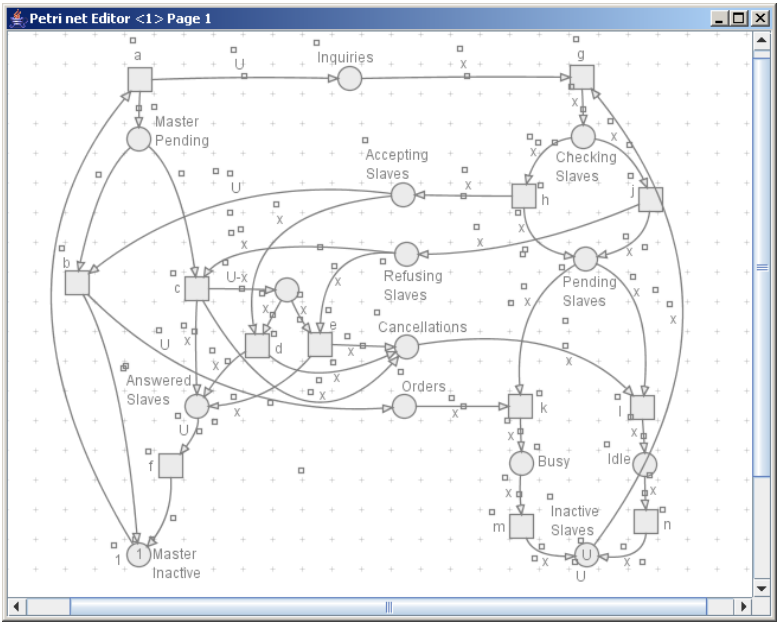
წარმოიშობა ჩიხური სიტუაცია: **g** გადასასვლელის გახსნა აკრძალულია, რადგან პოზიციაში **Writer Detached** (**ჩამწერი მოიხსნა**) მარკერი არ არის და ვერც **c** გადასასვლელი გაიხსნება, რადგან პოზიციიდან **Reader Detached (წამკითხველი მოიხსნა)** მარკერი **წამკითხველმა** აიღო. ჩიხიდან გამოსავალს პოზიცია **Writer Involved (ჩამწერი ჩაერია)** წარმოადგენს, რომელშიც არსებული მარკერის მონაწილეობით იხსნება **j** გადასასვლელი და **წამკითხველი** მდგომარეობაში **Failed** (**წაკითხვა ჩაიშალა**) გადადის, ხოლო **ჩამწერი** დაუბრკოლებლად აღწევს კრიტიკულ ზონას, განაახლებს ცვლადს და **d** გადასასვლელით თავის პირვანდელ მდგომარეობას უბრუნდება, რის შემდეგაც პოზიციას **Writer Detached** (**ჩამწერი მოიხსნა**) მარკერი უბრუნდება და **წამკითხველს** შეუძლია განაწილებული ცვლადის წასაკითხად ახალი რაუნდი წამოიწყოს (გადასასვლელთა მიმდევრობა **k-f-g**).

13. განაწილებულ მონაცემთა ბაზების განახლების „მასტერ-სლეივ" ალგორითმები

განაწილებული მონაცემთა ბაზების მართვის სისტემებში პროცესთა პარალელიზმისა და სინქრონიზაციის უზრუნველყოფა უმნიშვნელოვანესი ამოცანებია. მონაცემებთან მიმართვა და მათი დამუშავება ისე უნდა განხორციელდეს, რომ ინფორმაციის ლოგიკური მთლიანობა არ დაირღვეს (მონაცემთა სხვადასხვა ბაზებში წინააღმდეგობრივი ინფორმაცია არ იქნეს შენახული).

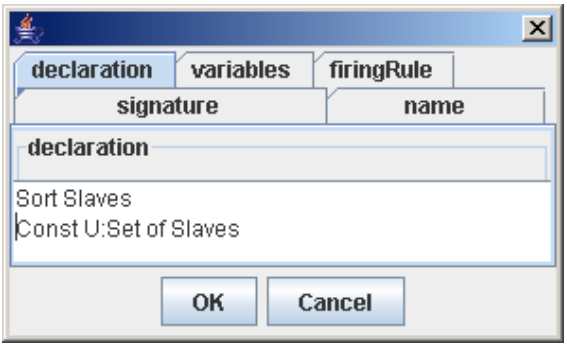
სამაგალითოდ პეტრის ქსელის ბირთის გარემოში წარმოვადგენთ მაღალი დონის (სისტემურ) პეტრის ქსელს განაწილებულ მონაცემთა ბაზებში აქტუალური **განახლების** ამოცანისთვის. მოდელი შეიცავს ერთ **სერვერულ** ("მასტერ") პროცესს, რომელიც მართავს **კლიენტების** ("სლეივ") პროცესთა სიმრავლეს, სადაც თითოეული კლიენტ-პროცესი ერთ მონაცემთა ბაზას განაახლებს.

განაწილებული ალგორითმის პირობა ასეთია: თუ ერთი მაინც კლიენტ-პროცესი მზად არ არის განახლების ოპერაციის შესასრულებლად, განახლების მთელი პროცედურა უქმდება (მოიხსნება). შესაბამისი პეტრის ქსელი 13.1 ნახაზზეა მოცემული.

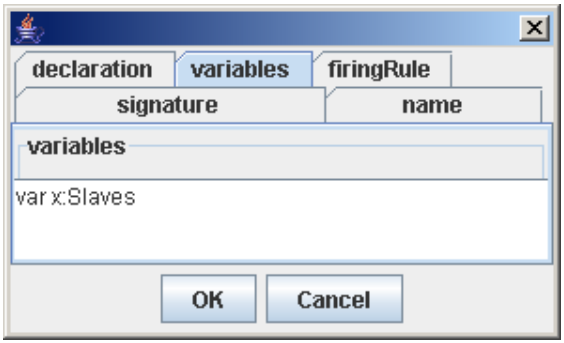


ნახ.13.1. მაღალი დონის პეტრის ქსელის მასტერ-სლეივ ალგორითმისთვის

ამოცანის მოდელირებისთვის პეტრის ქსელის ბირთვიდან **მაღალი დონის (სისტემური)** პეტრის ქსელის ინსტრუმენტები გამოიყენება (ნახ.13.2-3).



ნახ.13.2. პეტრის ქსელის მონაცემთა აღწერის ფანჯარა;



ნახ.13.3. პეტრის ქსელის ცვლადების აღწერის ფანჯარა

ნახაზებზე პეტრის ქსელის ბირთვის ინტერფეისში ქსელის **სისტემური სქემის** აღწერაა მოცემული, საიდანაც ჩანს, რომ ქსელისთვის განისაზღვრება ერთი ახალი ტიპი **Slaves** (**კლიენტ-პროცესები**), მუდმივა **U** წარმოადგენს **Slaves**-ტიპის კონსტანტას (ყველა კლიენტ-პროცესის სიმრავლე), ხოლო ცვლადი **x** – კონკრეტულ კლიენტ-პროცესს გამოსახავს. **სერვერ-პროცესი** ერთია, ამიტომ იგი შავი მარკერით გამოისახება.

ალგორითმის შესრულების დასაწყისში სერვერ- და კლიენტ-პროცესები **პასიურ** მდგომარეობაში იმყოფება (პოზიციები **Master Inactive** და **Inactive Slaves** შესაბამისად). განახლების პროცედურას სერვერ-პროცესი იწყებს, რომელიც პირველ ეტაპზე კლიენტ-პროცესების „გამოკითხვას“ ატარებს (პოზიცია **Inquiries**) განახლებისთვის მათი მზადყოფნის დასადგენად.

თუ ყველა კლიენტ-პროცესმა დადებითი პასუხი დააბრუნა (ანუ პოზიციაში **Accepting Slaves** კონსტანტა **U** მოხვდა), სერვერ-პროცესი განახლების პროცედურას დაიწყებს (გადასასვლელთა მიმდევრობა **b-k-m**), წინააღმდეგ შემთხვევაში (ერთი მაინც კლიენტ-პროცესი პოზიციაში **Refusing Slaves**) განახლების ტრანზაქცია უქმდება და შუალედური ოპერაციებით ყველა კლიენტ-პროცესი **გაუქმების** პოზიციაში იყრის თავს (პოზიცია **Cancellations**), საიდანაც ისინი **პასიურ** მდგომარეობას უბრუნდება ახალი ტრანზაქციის ინიცირებამდე სერვერ-პროცესის მხრიდან.

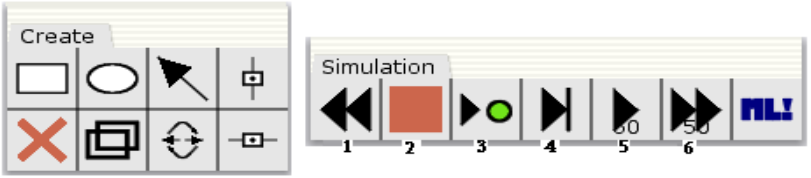
შეიძლება დავასკვნათ, რომ პეტრის ქსელის ბირთვი (**PNK**) მოსალოდნელია იქცეს ერთერთ ყველაზე მოხერხებულ საშუალებად სხვადასხვა აქტუალური საპრობლემო სფეროების მოდელირებისთვის მისი უნივერსალურობიდან და მოქნილობიდან გამომდინარე, რაც მას პოტენციურად პეტრის ქსელის დღეს არსებულ ყველა ტიპთან მომუშავე ინსტრუმენტად აქცევს.

14. მარკეტინგული პროცესების მოდელირება და ანალიზი პეტრის ქსელებით

მრავალპროცესორული ორგანიზაციული სისტემებისა და რთული სტრუქტურული ობიექტების კვლევის კომპლექსური ანალიზისთვის აღიწერება სხვადასხვა ტიპის ბიზნეს-ნაკადების მოძრაობა. არის შემთხვევები, როდესაც სხვადასხვა ტიპის ბიზნეს-ნაკადში საჭიროა გავრცელდეს ერთიდაიგივე რესურსი და ნაკადებმა იმოძრაოს საერთო მარშრუტით. ასეთი მოდელის აგებისას, კლასიკურ პეტრის ქსელში რთულია კომპლექსური დინამიკის სრული სურათის დანახვა ანუ ერთი ნაკადის მაჩვენებლების გარჩევა სხვა ნაკადის მაჩვენებლებისგან.

ამ მიზნით დანიელი მეცნიერ-ინჟინრების მიერ დამუშავდა გაფართოებული პეტრის ქსელის ვარიანტი-ფერადი პეტრის ქსელები (CPN), CPNML ენითა და CPN-ინსტრუმენტით, რომელიც იყენებს გრაფო-ანალიზური, ობიექტ-ორიენტირებული, ვიზუალური დაპროგრამების პრინციპებს. CPN ML ენა საშუალებას იძლევა აღიწეროს ქსელის ფერადი კომპონენტები (მარკერები), ცვლადები, კონსტანტები და თვით პოზიციების, გადასასვლელებისა და რკალების ტექსტური აღწერები, რაც ერთგვარ კომფორტს ქმნის ქსელის წასაკითხად და გასაგებად [18].

14.1 სურათზე ნაჩვენებია CPN გარემოში პეტრის ქსელის აგებისა და იმიტაციური მოდელირების ვიზუალური კომპონენტები. სიმულაციის მე-3 ღილაკი საშუალებას იძლევა იტერაციულად, ხელით ავამუშავოთ ჩვენთვის საჭირო გადასასვლელი (აირჩევა რამდენიმე ალტერნატიულიდან). მე-6 ღილაკი იძლევა საბოლოო მარკირების სურათს. 1-ელი ღილაკი – კი აღადგენს საწყის მარკირებას, ექსპერიმენტის თავიდან ჩასატარებლად.



სურ. 14.1. ფერადი პეტრის ქსელის შექმნისა და იმიტაციური მოდელირების ინსტრუმენტები

CPN ML ენის მიხედვით, ქსელის ყოველ პოზიციას გააჩნია მინიმუმ ორი ჭდე: სახელი, რომელიც აღმნიშვნელი წრის ან ელიფსის შიგნით იწერება და მარტივი ან შედგენილი ტიპი (პოზიციის გვერდით, კურსივით, საკვანძო სიტყვა type, color ან string) მაგალითად, პოზიცია „კონტრაქტები“ INTxDATA ტიპისაა, რომელიც წინასწარგანსაზღვრული INT და DATA ტიპების დეკარტული ნამრავლით წარმოიქმნება. ფერადი პეტრის ქსელი შეიცავს „ფერად“ მარკერებს, რომლებიც კონკრეტული ტიპის შესაძლო მნიშვნელობათა სიმრავლე ან მულტისიმრავლეა.

განისაზღვრება კონსტანტები (საკვანძო სიტყვა val), ცვლადები (var) და ფუნქციები (fun). სხვადასხვა ტიპის მონაცემთა შორის კავშირების ასახვისთვის გამოიყენება სიმრავლეთა და კომპლექტების თეორიის ელემენტები.

გარდა მონაცემთა ტიპისა, ყოველი პოზიციის გვერდით შეიძლება აისახოს მოცემულ მომენტში შემავალი ფერადი მარკერები. საინიციალიზაციო მარკირება ხაზგასმული ტექსტის სახით გამოიტანება. მაგალითად, საწყის მდგომარეობაში პოზიცია „კონტრაქტები“ შეიცავს INTxDATA ტიპის ფერად მარკერთა 5-ელემენტიან სიმრავლეს (საინიციალიზაციო მარკირება):

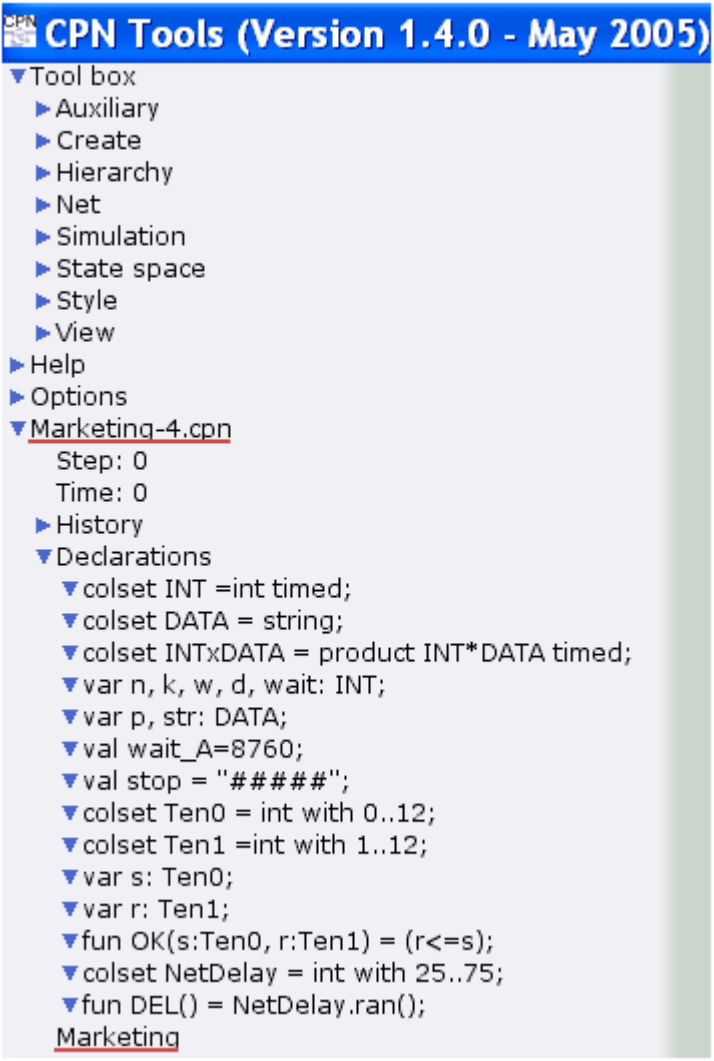
{1`(1,”კონტრაქტი_1“), 1`(2, „ კონტრაქტი_2“),
1`(3, „კონტრაქტი_3“), 1`(4, „ კონტრაქტი_4“), 1`(5, „# # # # “) }.

ბოლო, მე-5 ელემენტი შეესაბამება დასასრულის იდენტიფიკაციას - stop.

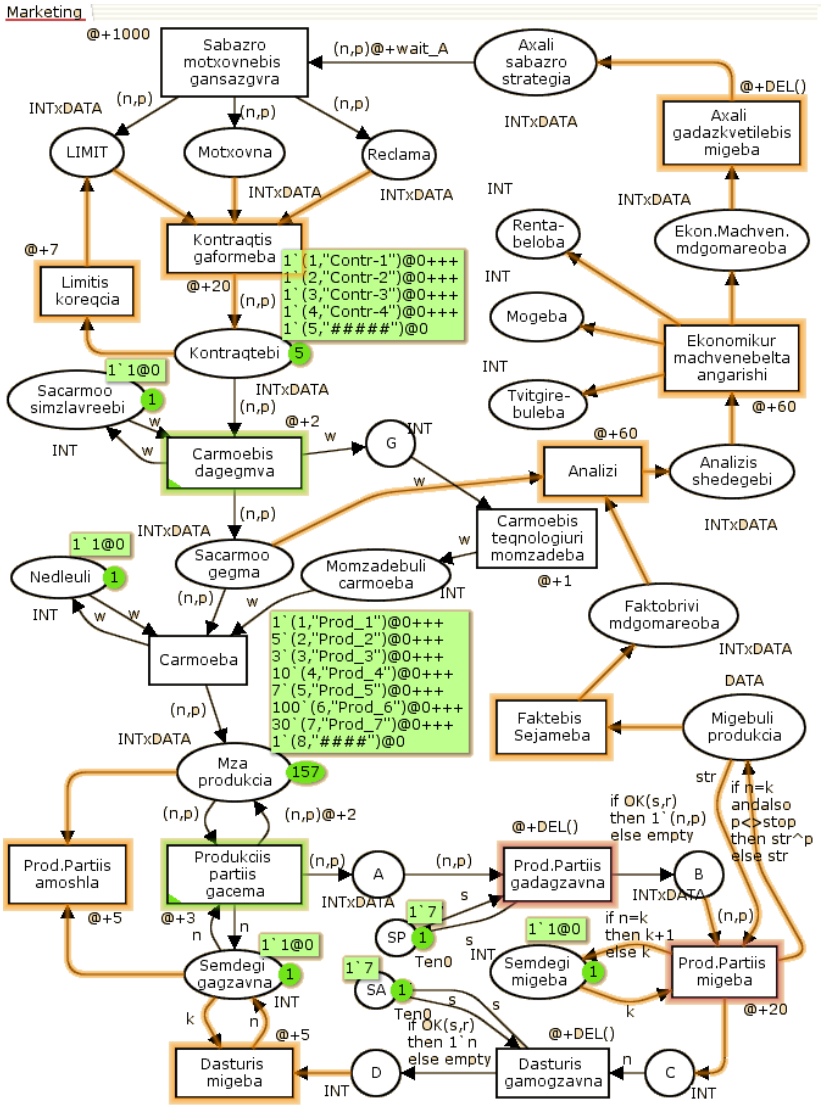
საყურადღებოა „1“-იანი ყოველი ელემენტის დასაწყისში (მას კოეფიციენტი ეწოდება), რომელიც მიუთითებს, რომ პოზიციაშია არაუმეტეს 1 ცალი მოცემული ფერის მონაცემი (ანუ არსებობს მხოლოდ ერთი კონტრაქტი ნომრით „კონტრაქტი_1“, რომლის ფერია - რიგითი ნომერია 1). ამ შემთხვევაში გვაქვს მონაცემთა ელემენტების სიმრავლე.

14.2 სურათზე წარმოდგენილია CPN ინსტრუმენტის ფუნქციების, დახმარების, ოფციებისა და აღწერის (Declarations) ნიმუში.

მაგალითის სახით 14.3 სურათზე ნაჩვენებია საწარმოს მარკეტინგული კვლევის პროცესის კომპლექსური ანალიზის ფერადი პეტრის ქსელის ფრაგმენტი.

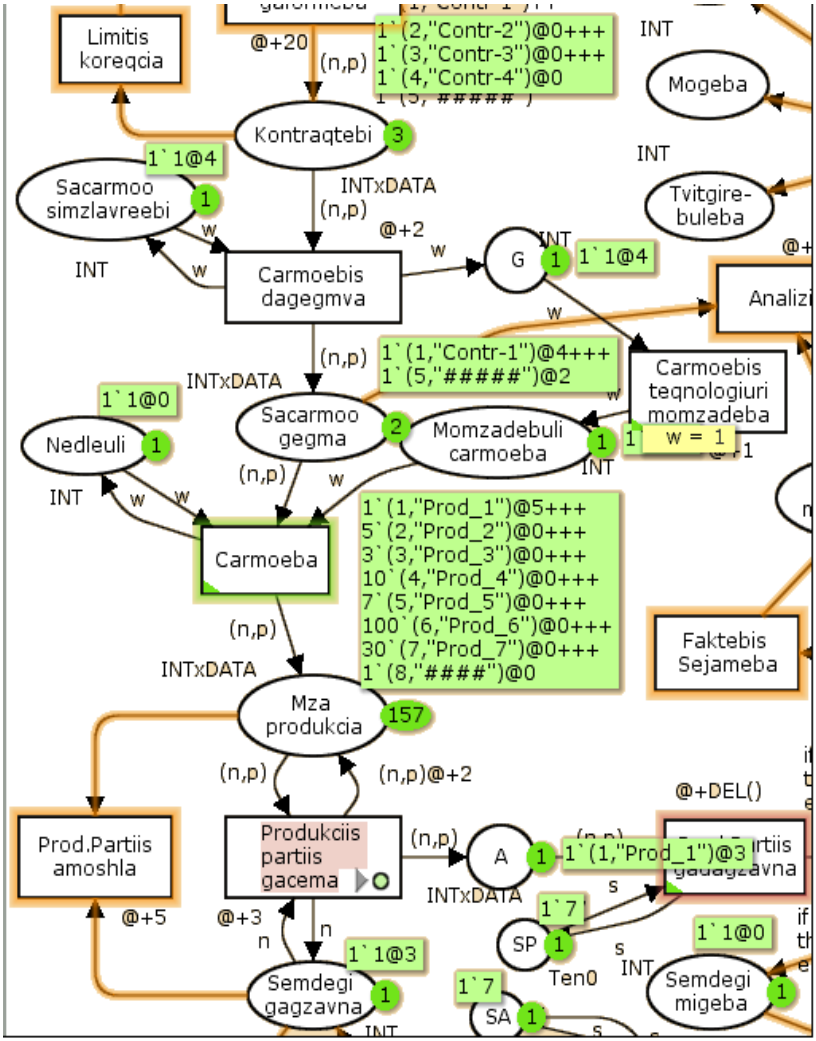


სურ. 14.2. CPN-ინსტრუმენტის სამუშაო გარემო



სურ. 14.3. მარკეტინგული პროცესის ფრაგმენტი CPN-ის გარემოში

მარკეტინგული კვლევის პროცესის წარმოდგენილი ქსელის (სურ. 14.3) დინამიკა რამდენიმე ბიჯის შემდეგ, სადაც ჩანს მარკერების შეცვლილი მდგომარეობა, ნაჩვენებია 14.4 სურათზე.



სურ. 14.4. იმიტაციური მოდელირების შუალედური ეტაპი

ფერად პეტრის ქსელებში კარგადაა შერწყმული პეტრის ქსელებისა და დაპროგრამების თეორია (იერარქიულობა, მოდულურობა – დიდი სისტემების მოდელირებისთვის), რაც მის დიდ პრაქტიკულ ღირებულებასაც განაპირობებს თანამედროვე ინფორმაციულ ტექნოლოგიათა გამოყენების მრავალ სფეროში.

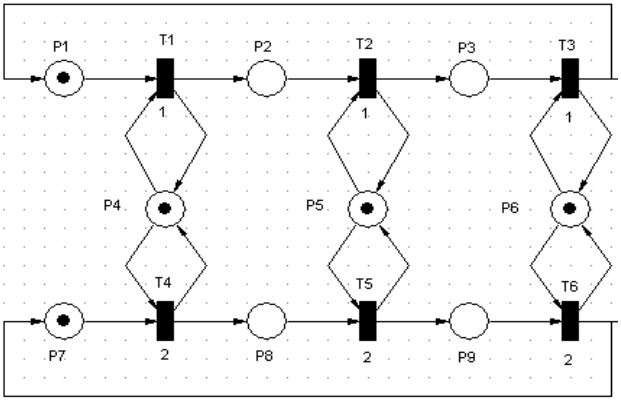
15. განაწილებული სისტემების რესურსების ადმინისტრირების ამოცანები

15.1. პროცესების კვლევა დინამიკურ რეჟიმში პეტრის ქსელებით

გავაანალიზოთ კომპიუტერული ქსელის მოქმედება დინამიკურ რეჟიმში. ამ შემთხვევაში სერვერების მიერ კლიენტთა მოთხოვნების დაკმაყოფილების პროცესი შეიძლება მოდელირებულ იქნას ტრანზიტული დროითი პეტრის ქსელის (Timed Transition Petri Net) საშუალებით [3]. პეტრის ქსელის დროითი გაფართოება ჩვენს შემთხვევაში იქნება ალბათური (სტოქასტური). ამგვარად, ასეთი ქსელის ანალიზი შესაძლებელია მარკოვის მეთოდების გამოყენებით, რომელშიც დრო ექსპონენციალურადაა განაწილებული [18,19].

სტოქასტური პეტრის ქსელის მისაღებად საჭიროა „პოზიცია-გადასასვლელების ქსელს" დაემატოს გადასასვლელთა გაშვების (დაყოვნების, მოლოდინის) დროთა მომენტები (მაგალითად, μ1, μ2, . . . , μn). განვიხილოთ კერძო მაგალითი კორპორაციული ქსელისათვის, ორი სერვერით და სამი კლიენტით.

15.1 ნახაზზე მოცემულია სტოქასტური პეტრის ქსელის საწყისი მდგომარეობა. მარკერის არსებობა S1(p1,p2,p3) და S2(p7,p8,p9) სერვერებში ნიშნავს მათ მზადყოფნაზე კლიენტების მომსახურებისათვის.



ნახ.15.1. სტოქასტური პეტრის ქსელის საწყისი მდგომარეობა

დავუშვათ, რომ C(p4,p5,p6) კლიენტის პოზიციებში მარკერები მუდმივადაა, ე.ი. მოთხოვნები არსებობს და ისინი ელოდება სერვერის მომსახურებას. როგორც აღვნიშნეთ, Tj გადასასვლელის გახსნის დროა (ანუ მომსახურების დაყოვნების დრო). Tj-ური გადასასვლელის გახსნის საშუალო დრო იქნება 1/μ, სადაც μ გადასასვლელის გახსნის ინტენსივობაა.

სისტემის მდგომარეობები, ანუ მარკერების სიმრავლე შეიძლება ასე ჩაიწეროს:

p პოზიციები და t გადასასვლელები:

M1__100111001
M2__010111001
M3__100111100
M4__010111001
M5__100111010
M6__001111100
M7__010111010
M8__001111010

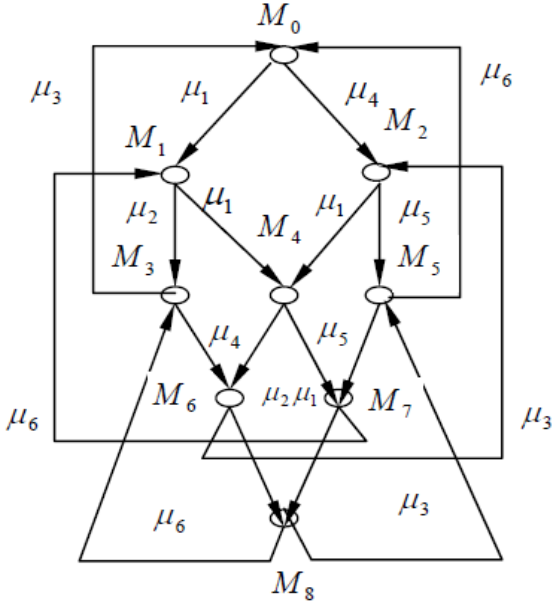
სადაც Mi, 0<=i<=K მდგომარეობებია (მარკირებები); Tj, 1<=j<=L გადასასვლელები; μj, 1<=j<=L – დაყოვნების დრო

გადასასვლელის გასაღებად. ჩვენ შემთხვევაში m=3 და n=2, ამიტომ კომბინაცია იქნება $m^n = 9$.

გადასასვლელების გახსნის ორგანიზება, როცა სისტემა ყველა მდგომარეობას გადის ნაჩვენებია 15.2 ნახაზზე, რომელსაც პეტრის ქსელის მიღწევადობის გრაფს უწოდებენ.

ასეთი სტოქასტური პეტრის ქსელის რაოდენობრივი ანალიზი შეიძლება განხორციელდეს შესაბამისი მარკოვის პროცესების ანალიზით.

განვიხილოთ მარკოვის ჯაჭვის მაგალითი, რისთვისაც ამ ნახაზზე გრაფის რკალებზე მივამაგროთ გადასასვლელების გაშვების μ კოეფიციენტები.



ნახ.15.2. პეტრის ქსელის მიღწევადობის გრაფი

ჩვენთვის საინტერესოა დავადგინოთ სისტემის თითოეულ მდგომარეობაში გადასვლის ალბათობები, ამისათვის საჭიროა შევადგინოთ კოლმოგოროვის განტოლებათა სისტემა:

$$P5*\mu6+P3*\mu3-P0*(\mu1+\mu4)=0$$

$$P7*\mu6+P0*\mu1-P1*(\mu2+\mu4)=0$$

$$P0*\mu4+P6*\mu3-P2*(\mu1+\mu5)=0$$

$$P6*\mu3+P0*\mu4-P3*(\mu1+\mu5)=0$$

$$P1*\mu4+P2*\mu1-P4*(\mu2+\mu5)=0$$

$$P2*\mu5+P8*\mu3-P5*(\mu1+\mu6)=0$$

$$P3*\mu4+P4*\mu2-P6*(\mu3+\mu5)=0$$

$$P4*\mu5+P5*\mu1-P7*(\mu2+\mu6)=0$$

$$P6*\mu5+P7*\mu2-P8*(\mu3+\mu6)=0$$

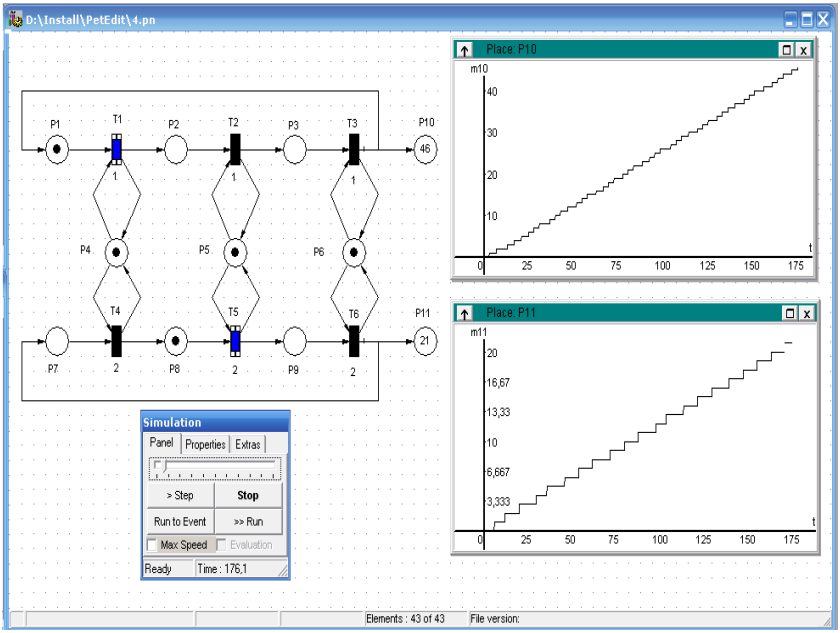
$$P0+P1+P2+P3+P4+P5+P6+P7+P8=1$$

მაგალითად, თუ დავუშვებთ, რომ μ1=3, μ2=5, μ3=2, μ4=3, μ5=1, μ6=7, მაშინ ალბათობათა მნიშვნელობები, შესაბამისად იქნება:

P0=0.11; P1=0.05; P2=0.07;
P3=0.26; P4=0.06; P5=0.02;
P6=0.37; P7=0.01; P8=0.05.

უნდა აღინიშნოს, რომ კომპიუტერული ქსელისათვის ერთი მდგომარეობიდან მეორეში გადასვლის ინტენსივობა μ არის სერვერის მიერ შესაბამისი კლიენტის მოთხოვნის მომსახურებისათვის საჭირო დროის შებრუნებული სიდიდე, ე.ი. 1/Ts. ამ განტოლებათა სისტემის ამოხსნით გაუსის მეთოდით მივიღებთ სისტემის ერთი მდგომარეობიდან მეორეში გადასვლის ალბათობებს $\mathbf{P_0, P_1, P_2, \ldots, P_8}$.

15.3 ნახაზზე წარმოდგენილი პეტრის ქსელის გრაფისთვის PetEdit რედაქტორში ავაგოთ შესაბამისი მოდელი. სერვერებისთვის შევირჩიოთ პირობითად განსხვავებული მწარმოებლურობა, კერძოდ ერთი ამუშავებს მოთხოვნებს 1 წმ-ში, მეორე კი - 2 წამში (ამ მნიშვნელობების ცვლილებით შესაძლებელია შემდგომი ექსპერიმენტების ჩატარება).



ნახ.15.3. პეტრის ქსელის სიმულაციის რეჟიმი მახასიათებლებით

15.3 ნახაზზე ნაჩვენებია მიღებული პეტრის ქსელის მოდელი და სიმულაციის შედეგები. 10 და 11 პოზიციები ასახავს სერვერების მიერ შესრულებული პროცედურების ჯამურ რაოდენობას.

როგორც დიაგრამებიდან ჩანს, პირველი სერვერის სწრაფქმედების, ან სერვისების დამუშავების პროცედურების ხანგრძლივობა თითქმის ორჯერ ნაკლებია. ამიტომაც შედეგები მე-10 პოზიციაში ორჯერ მეტია. თუ სერვერული სისტემებისთვის მოხდება სერვისული ოპერაციების დამუშავების დროის წინასწარ განსაზღვრა, მაშინ შედეგებიც შესაბამისად აისახება.

15.2. ჩიხური სიტუაციების მართვა

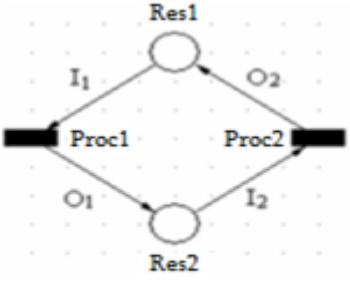
განიხილება ოპერაციულ სისტემების პლატფორმაზე მონაცემთა ბაზების ფაილების კოლექტიური გამოყენების დროს ჩიხური პროცესების არსებობისა და მათი გამორიცხვის შესაძლებლობანი. შემუშავებულია ასეთი პროცესების მართვის მოდელი პეტრის ქსელის ინსტრუმენტის საფუძველზე. ჩატარებულია ამ მოდელის იმიტაციური გამოკვლევა და აგებულია შესაბამისი დროითი მახასიათებლები [20,21].

თანამედროვე მულტიპროცესორულ სისტემებში, მათ შორის ლოკალურ კომპიუტერულ ქსელებში, რომლებიც გამოთვლითი რესურსების საერთო გამოყენების კონცეფციას ეყრდნობა, განსაკუთრებული მნიშვნელობა ენიჭება პროცესების ეფექტურად ორგანიზაციის საკითხს ჩიხური სიტუაციების აღმოსაფხვრელად.

პროცესი ჩიხურია (deadlock), თუ იგი ელოდება ისეთი ხდომილების შესრულებას, რომელიც არასოდეს მოხდება. ორი ან რამდენიმე პროცესი შეიძლება მოხვდეს ჩიხში, თუ თითოეული მათგანი აბლოკირებს რესურსებს (მაგალითად, მონაცემთა ბაზის ცხრილებს, ან მის ფრაგმენტებს), რომლებიც ესაჭიროება სხვა პროცესებს და თვითონ კი მოითხოვს ისეთ რესურსებს, რომლებიც ბლოკირებულია სხვა პროცესების მიერ. ოპერაციულ სისტემას ჩვენ განვიხილავთ როგორც გამოთვლითი რესურსების ადმინისტრატორს, ხოლო რესურსებად გვევლინება ცენტრალური პროცესორი, ოპერატიული მეხსიერება, დისკოები, ფაილური სისტემები, პროგრამები და მონაცემთა ბაზები, პრინტერები, ქსელური არხები და ა.შ.

15.4 ნახაზზე ნაჩვენებია ელემენტარული ჩიხური ოპერაციის მაგალითი, ჩაწერილი პეტრის ქსელის გრაფით. აქ Proc1, Proc2 პროცესებია, ხოლო Res1, Res2 - რესურსები. პოზიცია-გადასასვლელთა შემაერთებე-ლი რკალები შემდეგი დანიშნულე-ბითაა: I_1 : Res1-რესურსი გამოეყო Proc1-პროცესს; O_1: Proc1-

პროცესი მოითხოვს Res2-რესურსს; I_2 : Res2-რესურსი გამოეყო Proc2-პროცესს; O_2: Proc2-პროცესი მოითხოვს Res1-რესურსს.



ნახ.15.4. ჩიხური სიტუაცია

როგორც ნახაზიდან ჩანს, Proc1 პროცესს ბლოკირებული აქვს Res1 რესურსი და მუშაობის გასაგრძელებლად სჭირდება Res2 რესურსი. Proz2 პროცესს კი პირიქით, ბლოკირებული აქვს Res2 რესურსი და მუშაობის გასაგრძელებლად სჭირდება Res1 რესურსი. ამგვარად, ორივე პროცესი იმყოფება მუდმივად მოლოდინის რეჟიმში.

ჩიხური პროცესების არსებობისათვის ოთხი აუცილებელი პირობა იქნა განსაზღვრული [20]. ურთიერთგამორიცხვის (პროცესებს აქვს რესურსების მონოპოლური გამოყენების უფლება), დამატებითი რესურსების მოლოდინის (პროცესებს აქვს უკვე გამოყოფილი რესურსები, მაგრამ ელოდება დამატებითს), არაგადანაწილებადობის (პროცესებს არ შეიძლება ჩამოერთვას რესურსები მათ საბოლოო შესრულებამდე) და წრიული მოლოდინის (არსებობს პროცესების წრიული ჯაჭვი, რომელშიც ყოველი პროცესი აბლოკირებს ერთ ან რამდენიმე რესურსს, რომელიც ესაჭიროება ჯაჭვში მომდევნო პროცესს).

ჩიხური პროცესების მართვის პრობლემა ოპერაციულ სისტემებში განიხილება შემდეგი ამოცანების გადაწყვეტით:

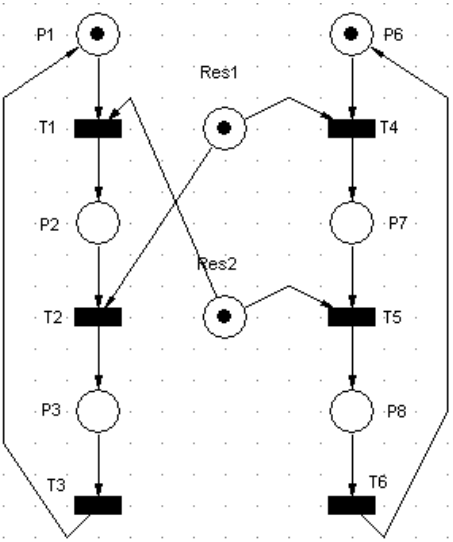
- ჩიხების თავიდან აცილება. თუ ჩიხების არსებობის აღწერილი პირობებიდან მოხერხდება ერთი ან რამდენიმე პირობის მოხსნა, მაშინ შესაძლებელია ჩიხების აღმოცენების თავიდან აცილება;

- ჩიხების გერდის ავლა. აქ პრინციპულად დასაშვებია ჩიხური სიტუაციის არსებობა, მაგრამ მისი მოახლოვებისას მიიღება შესაბამისი გამაფრთხილებელი ზომები. ამ დროს შესაძლებელია რესურსების უფრო რაციონალური გამოყენება, ვიდრე წინა შემთხვევაში;

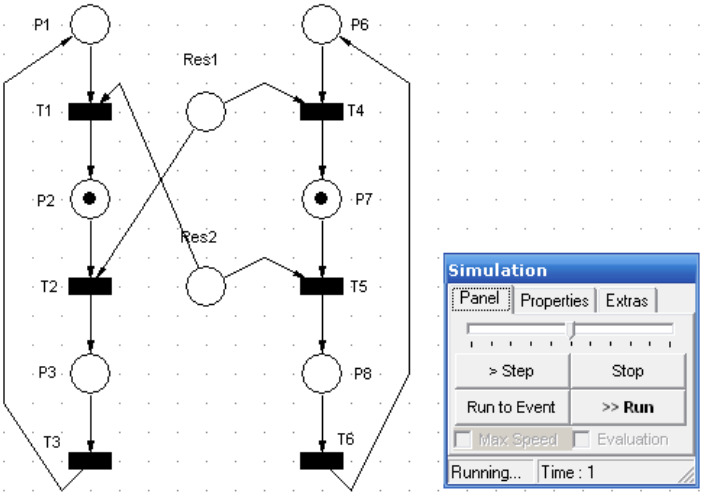
- ჩიხების აღმოჩენა. ამ დროს ჩიხური სიტუაციები ლოკალიზდება და ოპერატორს მიეწოდება სათანადო ინფორმაცია მათ შესახებ;

- ჩიხური სიტუაციის აღდგენა. ესაა ჩიხური სიტუაციიდან გამოსვლა მიმდინარე მუშაობის შედეგების გარკვეული დანაკარგებით.

ახლა განვიხილოთ კონკრეტული შემთხვევა ორი პროცესისთვის (Proc1, Proc2), რომლებიც ორი საერთო რესურსის (Res1, Res2) გამოყენებით ასრულებენ გარკვეულ პროცედურათა მიმდევრობას. 15.5 ნახაზზე წარმოდგენილია შესაბამისი პეტრის ქსელის გრაფი საწყის და შუალედურ (ჩიხურ) მდგომარეობაში.



ა)

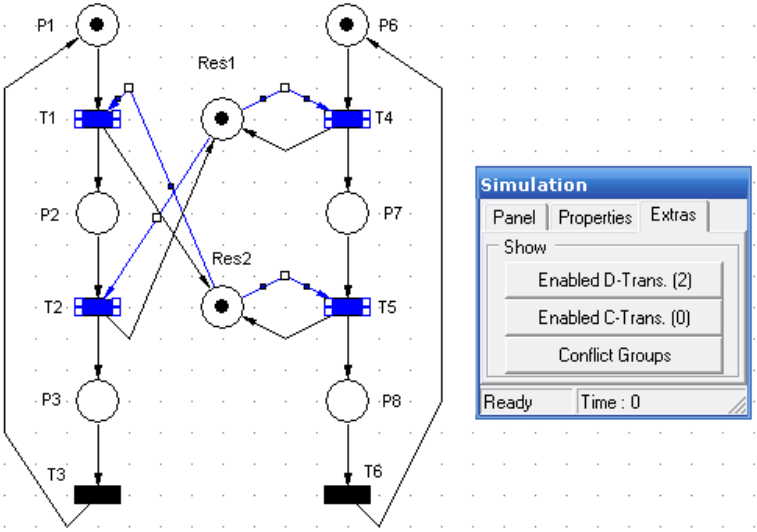


ბ)

ნახ.15.5. საწყისი მდგომარეობა (ა),
ჩიხური სიტუაცია ბლოკირებული რესურსებით (ბ)

ორივე რესურსი ბლოკირებულია შუალედურ პროცედურაში და ელოდება მეორე რესურსს. ამ შემთხვევაში პეტრის ქსელი უძლურია პროცესის გასაგრძელებლად. საჭიროა დამატებითი რკალების შემოტანა, რომლებიც უზრუნველყოფს ბლოკირებული რესურსების გათავისუფლებას.

15.6 ნახაზზე დამატებულია აღნიშნული რკალები. აქვე ნაჩვენებია კონფლიქტურ გადასასვლეთა ჯგუფი.

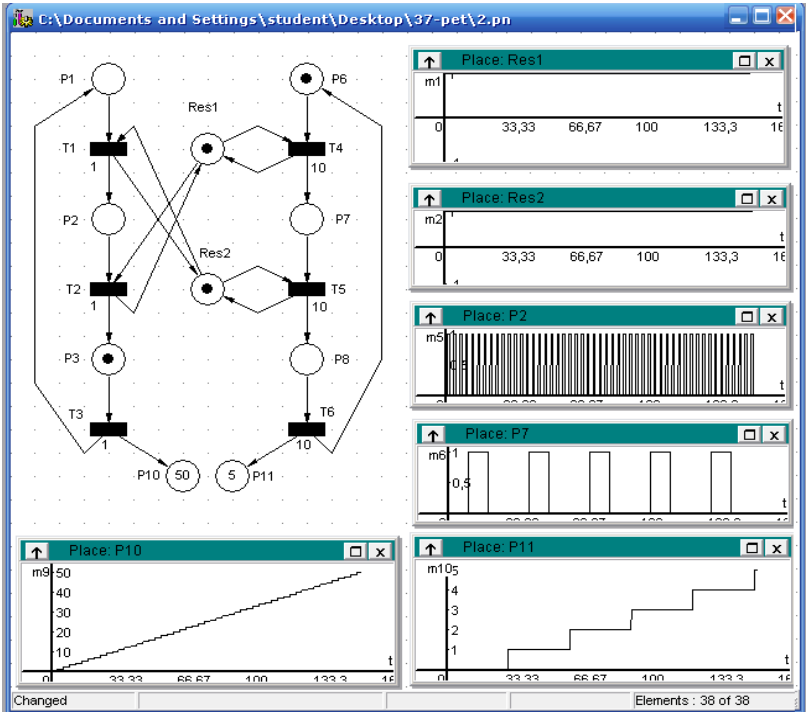


ნახ.15.6. კონფლიქტური ჯგუფის გადასასვლელები

15.7 ნახაზზე ნაჩვენებია პეტრის ქსელის გრაფის იმიტაციის პროცესის შედეგები, მათი ცალკეული პოზიციების დროითი დიაგრამებით. სქემაზე Proc1-ის გადასასვლელების (პროცედურათა შესრულების) დაყოვნების დრო არის, პირობითად, 1 წმ, ხოლო Proc2-ის 10 წმ.

მონაცემთა განაწილებული ბაზების ადმინისტრირებისათვის ნებისმიერი ოპერაციული სისტემის პლატფორმაზე, ჩიხური პროცესების მართვა, ანუ აღმოჩენა და მისი დროული გამორიცხვა

შესაძლებელია შესაბამისი პროცესების მოდელირებით პეტრის ქსელის გრაფო-ანალიზური ინსტრუმენტის საფუძველზე, რაც ზემოთ იყო ილუსტრირებული. აგებული მოდელის იმიტაციური პროცესის გამოკვლევა იძლევა შესაბამის დროითი მახასიათებლებს გარკვეული დასკვნების გასაკეთებლად.



ნახ.15.7. პროცესების შესრულების დროითი მახასიათებლები

გამოყენებული ლიტერატურა:

1. რეისიგი ვ., სურგულაძე გ., გულუა დ. Modellierung-2008: ახალი ხიდი მეცნიერებასა და წარმოებას შორის. სტუ-ს შრ. კრებ. №1(4), 2008. თბ., 9-15 გვ.

2. სურგულაძე გ., თურქია ე. ბიზნეს-გეგმის ავტომატიზებული დამუშავების პროცესის სამუშაო ნაკადების მართვის სისტემა. სტუ-ს შრომები, № 8(424), თბილისი, 1998.

3. სურგულაძე გ., გულუა დ. განაწილებული სისტემების ობიექტ-ორიენტი-რებული მოდელირება უნიფიცირებული პეტრის ქსელებით. სტუ. თბ., 2005.

4. Surguladze G., Turkia E., Gulua D. Perfection of Object-Oriented Projecting with a Process-Oriented Approach. Intern.Conf. "Education, science and economics at universities. Integration to international education area". Płock, Poland, 2008.

5. სურგულაძე გ., თურქია ე., მეგი გიუტაშვილი მ. ბიზნეს-პროცესების მოდელირების თანამედროვე პრინციპები და ინსტრუმენტული საშუალებები. სტუ-ს შრ. კრებ. №1(4), 2008. თბ., 73-77 გვ.

6. Open Management Group, Business Process Management Initiative: Business Process Modeling Notation (BPMN). http://www.bpmi.org. 2006

7. Weske M. Business Process Management: Concepts, Languages, Architectures, Springer-Verlag Berlin-Heidelberg-New York. 2007

8. Turkia E.G. and Giutashvili M. Z., Perfection of Object-Oriented Projecting with a Process-Oriented Approach., "Georgian Engineering News" 4'07, Georgia, 2007

9. Valk R. Petri Nets as Token Objects – An Introduction to Elementary Object Nets. In: Desel, Jörg (Hrsg.) ; Silva, Manuel (Hrsg.): Application and Theory of Petri Nets 1998. Berlin; Heidelberg ; New York et al : Springer, 1998 (Lecture Notes in Computer Science (LNCS) 1420), S. 1–25.

10. Дж. Питерсон: Теория Сетей Петри и моделлирование систем. Перевод с английского. Москва, «Мир», 1983

11. პეტრის ქსელების ოფიციალური ვებგვერდი. URL: http://www.informatik.uni-hamburg.de/TGI/PetriNets/

12. ვ. რეისიგი, გ. სურგულაძე, დ. გულუა: ვიზუალური ობიექტორიენტირებული დაპროგრამების მეთოდები. გამომცემლობა ,,ტექნიკური უნივერსიტეტი". თბილისი, 2002 წ.

13. Moeller A. XML as an Interchange Format. Petri Nets International Conference, Aahrus, Danmark. Presentation, 27/06/2000.

14. Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara, 1996.

15. Jablonski, S. Projek-ForFlow, http://www.ai4.uni-bayreuth.de/en/research/projects/010_forflow/index.html.

16. Surguladze G., Turkia E., Topuria N., Giutashvili M. Development and Research of the Computer System Models Supporting the Human Resource Selection for the Project Management, 3 rd Intern. Conf., Computational Intelligence (CI'09), Tbilisi, Georgia 2009.

17. М. Оутей, П. Конте, Эффективная работа SQL Server. М., Киев, Харьков, Минск, Санкт-Петербург 2002.

18. Bolch G., Greiner S., De Meer H., Trivedi K. Queueing Networks and Markov Chains, Modeling and Performance Evaluation with Computer Science Application. John Wiley & Sons, 1998. 726 S.

19. ბოლხი გ., სურგულაძე გ., პეტრიაშვილი ლ., ჩიხრაძე ბ. მულტიპროცესორული სისტემების რესურსების მართვის პროგრამული უზრუნველყოფის დამუშავება Borland_C++Builder ინსტრუმენტით. სტუ-ს შრ., 4(437), თბილისი, 2001.

20. Дейтель Г. Введение в операционные системы. Пер. с англ., Мир, М., 1987.

21. სურგულაძე გ., კაშიბაძე მ. ოპერაციული სისტემები: პროცესების მართვის კვლევა პეტრის ქსელების თეორიის გამოყენებით. სტუ. თბ., 1993.

განახლებული განმეორებითი ონლაინ გამოცემა

საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი“
თბილისი, მ. კოსტავას 77