Българска академия на науките
Институт по биофизика и биомедицинско инженерство
Секция „Биоинформатика и математическо моделиране“

АВТОРЕФЕРАТ

на дисертационен труд, представен за присъждане на
образователна и научна степен „Доктор“
по професионално направление
4.6. „Информатика и компютърни науки“
Научна специалност: 01.01.12 „Информатика“

на тема:

Програмна реализация на обобщени мрежи и приложения за моделиране

Нора Ангелова Ангелова
Научен ръководител: чл.-кор. проф. дмн дтн Красимир Атанасов

София, 2017

Дисертационният труд е обсъден и допуснат до защита на разширено заседание на секция „Биоинформатика и математическо моделиране“ на ИБФБМИ, състояло се на 9 декември 2016 г.

Дисертационният труд съдържа 178 страници, в които 22 фигури и библиография, включваща 110 заглавия.

Защитата на дисертационния труд ще се състои на …….. от …… часа в заседателната зала на Институт по биофизика и биомедицинско инженерство – Българска академия на науките, на открито заседание на научното жури в състав:

Председател:
Доц. д-р Татяна Илкова

Членове:
Акад. проф. дтн Иван Попчев
Чл.-кор. проф. дмн дтн Красимир Атанасов
Проф. дтн Людмил Даковски
Доц. д-р Александър Димов

Автор:
Нора Ангелова Ангелова

Заглавие:
Програмна реализация на обобщени мрежи и приложения за моделиране

Увод

Обобщените мрежи са разширение на мрежите на Петри [81]. Те са създадени през 1982 година, като до сега, в теорията са въведени седем различни аспекта: алгебричен, топологичен, логически, операторен, програмен, методологичен и дидактически. Преглед на направените публикации е направен в две статии от 2002 и 2007 година [13, 84].

Обобщените мрежи са средство за моделиране на реални паралелни процеси с дискретни събития, които се развиват с течение на времето. Те могат ефективно и ефикасно да моделират различни аспекти на процесите, чието поведение с течение на времето (т.е. динамично) се задейства и/или повлиява от някои външни и вътрешни събития. Това ги прави изключително полезен инструмент в много различни области като медицина, транспорт, производствени процеси, изкуствен интелект и други.

Настоящият дисертационен труд се фокусира върху програмната реализация на обобщените мрежи и приложения за моделиране на реални процеси от непознати до момента области на приложение. Дисертацията е структурирана в четири глави, заключение, библиография, списък на публикациите по дисертационния труд, списък на проекти с участието на докторанта. Направен е преглед на основните публикации в теорията и програмната реализация на обобщените мрежи, като на база на тях е формирана цел и задачи на дисертационния труд.

Във връзка с програмната реализация на обобщените мрежи са предложени две консервативни разширения на обобщените мрежи, формален метод за верификация, нов оптимизиран алгоритъм за функциониране на преход, който за първи път отчита възможността за разцепване и сливане на ядра. Последната функционалност позволява намаляване на броя на позициите и фиктивните ядра в мрежата, като предоставя възможност да се реализира разделяне на ядро в различни изходни позиции с различни характеристики (например: пациент и дадената кръвна проба) и сливането на ядра при определени дефинирани условия, за да бъдат обработвани заедно (например: обработване на различни утайки, получени от пречистването на водата). Получените и имплементираните в софтуера резултати са използвани за създаването и симулирането на обобщеномрежови модели.

Глава 1. Въведение в обобщените мрежи и тяхната програмна реализация

В Глава 1 са разгледани историческите бележки за развитието на обобщените мрежи и тяхната програмна реализация, основните дефиниции, свързани с тях, и алгоритмите за движение на ядрата.

1.1. Основни публикации в теорията на обобщените мрежи

Понятието *обобщена мрежа* (ОМ) е въведено за първи път през 1982 г. от К. Атанасов в [39]. Обзорът се спира на поредица от статии, разширяващи концепцията на ОМ и доказващи, че всяка М-мрежа, всички мрежи на Петри и всички техни разширения могат да си опишат чрез ОМ.

В началото на 90-те години основните резултати на К. Атанасов са изложени в книга [25]. В монографията се описват съществуващите към онзи момент консервативни разширения като интуиционистки размити обобщени мрежи, цветните ОМ, ОМ с оптимизационни компоненти и други. В нея са дефинирани оператори върху обобщени мрежи и се поставят основите на четири от седемте аспекта на теорията – алгебричен, топологичен, логически и методологичен.

Обзорът подробно се спира на резултатите описани в [34] - концепцията на обобщените мрежи, възможните техни приложения, редуцираните обобщени мрежи, консервативните разширения, алгебричният, операторният и методологичният аспект на обобщените мрежи и две допълнения свързани с понятието *индексирани матрици* и интуиционистки размита логика и интуиционистки размити множества.

След 2007 г. са въведени няколко важни консервативни разширения. Това са обобщени мрежи с характеристики на позициите (ОМХП), интуиционистки размити обобщени мрежи с характеристики на позициите от тип 1 (ИРОМХП1) и такива от тип 3 (ИРОМХП3) [1], обобщени мрежи с приоритети зависещи от времето (ОМПЗВ) и с динамични приоритети (ОМЯДП). През 2015 г. се дефинира и оптимизиран алгоритъм за функциониране на преход, когато сливането на ядра е позволено, при ОМХП, ИРОМХП1 и ИРОМХП2 [1*].

Голяма част от приносите в теорията са свързани с връзката на обобщените мрежи и областта на изкуствения интелект, като се посочват редица статии, описващи постигнатите резултати.

Друга част от изследванията в теорията на обобщените мрежи е свързана с тяхното реално приложение. Освен приносите в теорията на обобщените мрежи, за последните над 30 години са създадени и стотици обобщеномрежови модели. В дисертационния труд са описани част от статиите, описващи обобщеномрежови модели от различни научни направления, като генетични алгоритми, медицина, електронно обучение, телекомуникации, транспорт и други.

1.2. Описание и дефиниция на обобщена мрежа

Представени са дефиниции и графично представяне на понятието преход и обобщена мрежа.

1.3. Алгоритми за функционирането на преход и обобщена мрежа

През годините алгоритмите за функциониране на обобщена мрежа са променяни многократно с цел оптимизация. Тъй като основният компонент на една обобщена мрежа са преходите, алгоритъмът за функциониране на обобщена мрежа зависи от алгоритъма за функциониране на преход. В дисертационния труд са представени последните версии на базовия алгоритъм за функциониране на преход и на този за функциониране на обобщена мрежа, заедно с техните блок-схеми.

1.4. Преглед на текущата програмна реализация на апарата на обобщените мрежи

Първите опити за създаване на софтуерен продукт датират от 1983 година, имплементиран на езика Фортран. Прегледът описва всички софтуерни реализации до 2000 г.: създаденият през 1987-1989 г. от Л. Атанасова и Е. Димитров софтуерен продукт на езика Turbo Prolog 2.0; *Program Package for Generalized Nets* създаден през 90-те години от Р. Христов; *Software Tools for Generalized Nets* от 1993 г.; най-добрият до този момент софтуерен продукт за обобщени мрежи – *Gennete; SIM2000* създаден през 2000 г. от Н. Николов.

Новото поколение софтуерни продукти за изграждане на обобщени мрежи започва през 2003 г. с изграждането на софтуерен интерпретатор за обобщени мрежи *GNTicker*, а по-късно и на софтуера *GNLite*. За разлика от описаните до тук софтуерни продукти, *GNLite* е първият софтуер проектиран в обектно-ориентиран стил, позволяващ надграждане на функционалността му, с дефинирани стандарти за описание и комуникация с *GNTicker* на обобщеномрежовите модели. Изграждането на стандарти е изключително ключов елемент в създаването на софтуерни решения върху апарата на обобщените мрежи. Това позволява разглеждането на софтуера по компоненти и замяната на един компонент с друг, който отговоря на същите стандарти.

В следствие на създаването на *GNTicker*, *GNLite* и дефинираните стандарти за описанието на моделите се дефинира и файловият формат XGN за описание на обобщеномрежов модел.

Като резултат на описаните горе разработки се създават множество компоненти за работа с обобщени мрежи. Един от основните е *TickerServer*. Той представлява симулатор на обобщени мрежи, комуникацията с който се извършва посредством специален TCP/IP протокол. Избраната архитектура е от тип „клиент-сървър“. За него са създадени и специален език за описание на предикати *GNTCFL* (*GNTicker Characteristic Function Language*), протокол за комуникация между *TickerServer* и неговите клиенти *GNTicker Trace Protocol* (*GNTP*), а моделите се описват чрез XGN файлов формат. *GNTCFL* езикът обаче се оказва един от сериозните недостатъци на *TickerServer*, тъй като езикът е с LISP синтаксис, труден за обработка и тестване от неквалифицирани потребители, а от друга страна – изключително непознат на програмистите. Вследствие на това, създаването на приложно-програмен интерфейс (*Application Programming Interface*, *API*) [6] и нови симулатори на различни софтуерни езици на практика се оказва трудоемка и неоправдана задача.

Последната софтуерна реализация е интегрираната среда за разработка на обобщени мрежи *GN IDE* [56]. Разработката е направена на езика Java. *GN IDE* е създаден като графична среда за *GNLite* и *TickerServer*. *GN IDE* поддържа възможност за визуализация на модели описани в *XGN* файлове.

Пакетът обещава интерактивна симулация на обобщеномрежови модели чрез външен компонент (симулационен сървър *TickerServer*), следене на параметрите на модела по време на симулация и неговата история, визуално редактиране на моделите, съхраняване на моделите в *XGN* файлове.

В началото на разработката се разчита, че всички предикати и характеристични функции могат да се задават на езика на *GNTCFL*. Впоследствие се започва разработка на *GN IDE* симулатор, работещ с характеристични функции на езика JavaScript. Така се стига до „кръпка“, позволяваща изпълнение на алгоритъм за функциониране на обобщена мрежа с изпълнение на функциите на JavaScript. Алгоритъмът не е пълен, не е интегриран в *TickerServer* и на практика не може да се изпълнява алгоритъм с разцепване на ядра без допълнителна намеса директно през интегрираната среда за разработка (*Integrated Development Environment*, *IDE*). От друга страна, създадената „кръпка“ на алгоритъма не позволява използването на *GNTCFL* и *TickerServer*. Последната част от разработката прави *GN IDE* независим симулатор, включващ симулационна част, а не графична среда за изчертаване на модели. Някои от положителните характеристики на *GN IDE* са:

- платформената независимост (Windows, Linux чрез инсталиране на Java Runtime Environment (JRE));
- наличието на възможност за експортиране на моделите до TEX формат;
- възможността за изчертаване на графика на стойностите на определени характеристики на ядрата.

Някои от основните недостатъци на *GN IDE* са:

- отсъствието на възможност за автоматизирано проиграване на алгоритми за функциониране на преход и обобщена мрежа, в които е позволено разцепването и сливането на ядра;
- невъзможността за създаване на различни тестови сценарии върху единствен обобщеномрежов модел чрез прилагане на редуциращи оператори върху него;
- невъзможността за изпълнението на повече от една стъпка от алгоритъма за функциониране на обобщена мрежа чрез функцията *step*;
- изчертаването чрез графичния редактор е практически неизползваемо без допълнително оформяне на XGN файла, което изисква допълнително познаване на структурата на файла и езика XML;
- неприложимостта на приложението в уеб пространство и на таблетни устройства, което прави интерфейса и визуализацията на моделите ограничени и далеч от съвременните тенденции;
- строгата дефинираност и сложността на интеграцията на Windows (Linux) приложение с друг независим софтуер. Например научна и обобщеномрежова симулация, задвижвани с общи конфигурационни данни, проиграванни в обща среда;
- липсата на дефиниран програмно-приложен интерфейс (*API*), позволяващ изграждането и симулацията на модели, изцяло чрез средствата на код, чрез предварително дефиниран интерфейс, което да позволи симулацията на модели като част от по-големи системи и решения;
- липсата на реализирана възможност за промяна на данните в реално време, както и връзка с база данни, файлов източник и други;
- неподдържането на векторен и обектен тип характеристики на ядрата, невъзможността за създаване на характеристика с история от тип безкрайност (*IntegerInf*) и други.

1.5. Заключение

Постигнатите резултати в първа глава на дисертационния труд:

1. Направен е обзор на основните публикации в теорията на обобщените мрежи към настоящия момент.
2. Дадена е пълна дефиниция на понятието преход и обобщена мрежа.
3. Дадена е пълна дефиниция на основните алгоритми за функционирането на преход и функциониране на обобщена мрежа, като и двата алгоритъма са описани и чрез блок-схеми.
4. Направен е обзор на основните публикации в развитието на програмната реализация на обобщените мрежи към настоящия момент и на текущото състояние на интегрираната среда на разработка за обобщени мрежи (*GN IDE*).

Въз основа на представения обзор и анализ на публикуваните до сега резултати могат да се направят следните по-важни изводи:

- изграждането на програмна реализация отначало, от нов екип, без решаването на проблемите в текущия софтуерен продукт и създаването на стратегия за бъдещи разработки, често води до реализиране на недостатъчно актуален софтуерен продукт, който не имплементира новостите в теорията на обобщените мрежи и/или не покрива спецификите на изгражданите обобщеномрежови модели.
- текущата реализация на интегрираната среда за разработка на обобщени мрежи (*GN IDE*) трябва да бъде доразвита, като се решат или се предложат решения за бъдещи разработки на гореописаните проблеми;
- необходимо е да се дефинират нови разширения на понятието *обобщена мрежа*, които да разширят възможностите на обобщеномрежовите симулации и не налагат усложняване на алгоритмите за функциониране на преход и за функциониране на обобщена мрежа.
- важна част от изследванията в теорията на обобщените мрежи е свързана с тяхното реално приложение за симулиране на реални процеси и връзката им с други области, като изкуствен интелект.
- създаването на обобщеномрежови модели от области, в които сигурността е от съществено значение изисква създаването на методи за верификация на класове обобщени мрежи.

Цели и задачи

Въз основа на извършения анализ и направените изводи за основна цел на дисертационния труд се определя: „Разширяването на апарата на обобщените мрежи и имплементирането на новите възможности за изграждането на пълна версия на интегрираната среда за моделиране и симулация на обобщеномрежови модели“.

За постигане на поставената цел се дефинират следните задачи:

1) Да се проучат съществуващите консервативни разширения в теорията на обобщените мрежи и създадените алгоритми за тяхното функциониране. Да се създаде алгоритъм за функциониране на преход, при който сливането на ядра е позволено.
2) Да се дефинират разширения на обобщените мрежи, позволяващи динамична промяна на приоритетите на ядрата, да се докаже тяхната консервативност и да се покаже, че запазват алгоритмите за движение на ядрата в тях.

3) Да се предложи решение на проблем за верификация на обобщените мрежи, възникващ при реализацията на по-сложни симулации, използването на обобщени мрежи като част от по-големи софтуерни реализации и в области като програмното осигуряване.
4) Да се изгради и актуализира софтуерната реализация на обобщените мрежи, като се имплементира симулационна част в средата *GN IDE*, която позволява автоматизирано разцепване и сливане на ядра и се изчисти излишната функционалност свързана с поддържането на езика *GNTCFL*. Да се създаде *GN API* с подходящ интерфейс, който да позволи използването на обобщените мрежи като част от по-големи софтуерни реализации.
5) Да се изгради възможност за прилагане на редуциращи оператори върху обобщени мрежи. Да се подготви възможност за реализация на оператори и релации в *GN IDE*, като се изведат теореми за запазване на операциите и релациите над редуцираните обобщени мрежи.
6) Да се дефинира обобщеномрежов модел, реализиращ защита на програмното осигуряване, и да се извърши верификация на този модел посредством създадения в задача 3) метод.
7) Да се опишат и симулират обобщеномрежови модели, реализиращи пречистване на отпадни води и измерване на плоскопаралелни краищни мерки за дължина, където се отчита субективният фактор при измерването.

Глава 2. Приноси в теорията на обобщените мрежи

В Глава 2 на дисертационния труд се описват част от направените приносите в теорията на обобщените мрежи.

Първо в точка 2.1 е представена реализация на алгоритъм за функциониране на преход, където сливането на ядра е позволено.

След това се описват две от дефинираните разширения на обобщените мрежи – обобщени мрежи с приоритети, зависещи от времето и обобщени мрежи с ядра с динамични приоритети. За всяко едно от предложените разширения се показва, че запазва алгоритъма за движение на ядрата и се привежда доказателство на теорема за консервативност.

Накрая се дефинира метод за верификация на всяка стандартна обобщена мрежа, независеща от времето и се обобщават всички постигнати резултати в теорията на обобщените мрежи.

2.1. Алгоритъм за функциониране на преход, където сливането на ядра е позволено

В точка 2.1 дисертацията предлага нов модифициран алгоритъм за функциониране на преход, където сливането на ядра е позволено.

В предложения алгоритъм се отчита възможността едно ядро да се слее с друго при преминаването му от дадена входна позиция в дадена изходна.
Пълният вид на алгоритъма е следният:

Алгоритъм за функциониране на преход, където сливането на ядра е позволено

Алгоритъм А

В началото се прави уточнението, че едно ядро α има начална характеристика от следния вид: $x_0^{\alpha'} = \langle\{\beta_1, \beta_2, \dots, \beta_k\}, x_0^{\alpha}\rangle$, където x_0^{α} е стандартната начална характеристика на ядрото, а $\{\beta_1, \beta_2, \dots, \beta_k\}$ е множеството от ядра, с които α може да се слее.

(A01) Входните позиции на прехода се нареждат по приоритет в низходящ ред. Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A02)**.

(A02) Ядрата във входните позиции се нареждат по приоритет. За всяка входна позиция се създават два списъка. Първият списък първоначално съдържа всички ядра наредени по приоритет, а вторият списък е празен. Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A03)**.

(A03) Конструира се празна индексирана матрица R, съответстваща на индексираната матрица на условието на прехода – r. Един елемент на матрицата $r_{i,j}$ се запълва със стойност 0 или още „лъжа“, ако е изпълнено едно от следните условия:

1. i-ят ред съответства на празна входна позиция, т.е. няма ядра, които могат да преминат в изходна позиция на прехода.
2. $m_{i,j} = 0$, т.е. ако текущият капацитет на дъгата между i-тата входна и j-тата изходна позиция е 0.

Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A04)**.

(A04) Избира се входна позиция i от сортираните по приоритет входни позиции, която е с най-висок приоритет, има поне едно ядро в нея и от което ядро не е преминало в изходна позиция по време на текущата времева стъпка. След това се избира ядрото с най-висок приоритет в избраната входна позиция. Ако ядрото може да се разцепва,

изпълнението на алгоритъма продължава с изпълнение на стъпка **(A05)**, а в противен случай на **(A06)**.

(A05) (a) Избира се предикат от матрицата R, който е от реда, съответстващ на избраната входна позиция i, различен от 0 и не е проверяван по време на текущата времева стъпка. Ако няма такъв предикат се преминава към (e). Ако такъв предикат е открит се преминава към (b).
(b) Проверява се дали изходната позиция е пълна, и ако да, алгоритъмът продължава към (c), а в противен случай към (d).
(c) Проверява се дали избраното ядро може да се слее с ядро от пълната изходна позиция. Ако това е възможно се преминава към (d). В противен случай се задава стойност 0, отговаряща на стойност „лъжа“, в елемента на матрицата R и алгоритъмът се връща към стъпка (a).
(d) Избраният предикат от съответния ред в индексираната матрица R се проверява. Ако предикатът се оценява до „истина“, в съответния елемент на матрицата R се записва стойност 1, а в противен случай 0. Алгоритъмът продължава към (a).
(e) Ако получените стойности в съответния ред на матрицата R са само нули, стъпка **(A05)** приключва и изпълнението на алгоритъма продължава с изпълнение на стъпка **(A07).** В противен случай ядрото се разцепва и новополучените ядра се преместват във всичките допустими изходни позиции (допустими са тези позиции в реда на R, които имат стойност 1). Изчисляват се стойностите на характеристичната функция Φ за всички изходни позиции, в които е постъпило ядро. Стойностите се записват като характеристики на постъпилите ядра в съответните позиции. Стъпка **(A05)** приключва и изпълнението на алгоритъма продължава с изпълнение на стъпка **(A08).**

(A06) (a) Избира се предикат от матрицата R, който е от реда съответстващ на избраната входна позиция (i), различен от 0 и не е проверяван по време на текущата времева стъпка. Ако няма такъв предикат изпълнението на алгоритъма продължава с изпълнение на стъпка **(A07)**, а в противен случай на (b).
(b) Проверява се дали изходната позиция е пълна, и ако да, алгоритъмът продължава към (c), а в противен случай към (d).
(c) Проверява се дали избраното ядро може да се слее с ядро от пълната изходна позиция. Ако това е възможно се преминава към (d). В противен случай се задава стойност 0, отговаряща на стойност „лъжа“, в елемента на матрицата R и алгоритъмът се връща към стъпка (a).
(d) Избраният предикат от съответния ред в индексираната матрица R се проверява. Ако предикатът се оценява до „истина“, алгоритъмът продължава към (e), а в противен случай се връща към (a).
(e) Ядрото се премества в изходната позиция (съответстваща на предиката оценен до истина в d). Изчислява се стойността на характеристичната функция Φ за изходната позиция, в която е постъпило ядро. Стойността се записва като характеристика на ядрото. Стъпка **(A06)** приключва и изпълнението на алгоритъма продължава с изпълнение на стъпка **(A08).**

(A07) Ако ядро не може да премине на текущата стъпка от изпълнението на алгоритъма, то се премества във втория списък от ядра на входната позиция. Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A08)**.

(A08) Текущият брой на ядрата във всяка изходна позиция се увеличава с 1, за всяко едно ядро постъпило в нея, което не се е сляло с друго ядро в тази позиция по време на текущата стъпка. Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A09)**.

(A09) Текущият брой на ядрата във всяка входна позиция се намаля с 1, за всяко едно ядро излязло от нея по време на текущата стъпка. Ако текущият брой на ядрата във

входната позиция е станал равен на 0, на всички елементи в съответния ред на R се записва стойност 0. Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A10).**

(A10) Капацитетите на всички дъги, по които е преминало ядро, се намаляват с 1. Ако капацитетът на дъгата е станал равен на 0, стойност 0 се записва в елемента на матрицата R, който съответства на дъгата. Изпълнението на алгоритъма продължава с изпълнение на стъпка **(A11)**.

(A11) Проверява се дали има входна позиция с приоритет по-нисък от текущата, от която не са преминавали ядра към изходни позиции на текущата стъпка от алгоритъма. Ако има такава позиция, изпълнението на алгоритъма продължава с изпълнение на стъпка **(A04)**, а в противен случай – на стъпка **(A12).**

(A12) Текущото време се увеличава с елементарната времева стъпка t^0.

(A13) Проверява се дали текущото време е равно на t_1+t_2, т.е. дали е достигната максималната продължителност на активното състояние на прехода. Ако да, то изпълнението на алгоритъма продължава с изпълнение на стъпка **(A14)**, а в противен случай – на стъпка **(A04)**.

(A14) Край на активното състояние на прехода.

2.2. Обобщени мрежи с ядра с приоритети, зависещи от времето

Идеята за създаването на това разширение идва от необходимостта за изграждане на по-сложни обобщеномрежови модели в програмния аспект на теорията. В стандартната дефиниция на обобщена мрежа, всяко ядро получава приоритет в началото и той не се променя по време на изпълнението на симулацията. Изграждане на симулация, която изисква промяна на приоритета на ядрата по време на изпълнението ѝ, може да се пресъздаде единствено чрез добавяне на допълнителна характеристика на всяко ядро. Новата характеристика следва да замести предварително зададения приоритет на ядрото и да се променя на всяка стъпка от алгоритъма.

Въпреки това алгоритъмът за функциониране на преход работи с предварително зададения приоритет и на практика той по подразбиране не отчита динамичните приоритети на ядрата. За да бъде възможно пресъздаването на подобна симулация, използвайки същия алгоритъм, е необходимо усложняване на предикатите за преминаване на ядра, за да се позволи отчитането на характеристиката с динамичния приоритет (предикатът не позволява преминаване на ядро, което не е с най-висок приоритет спрямо допълнителната му характеристика). От програмна гледна точка, това усложнява функционалността, идеята за обектно-ориентирано и модулно програмиране и ще доведе до по-сложни условия за преход на ядрата, приоритетът на ядрото получен в началото става на практика неизползваем, и други.

Втората възможност за пресъздаване на подобна симулация е промяна на алгоритъма за функциониране на преход така, че сортирането на ядрата да става по неговата допълнителна характеристика, което отново ще направи началния приоритет на практика излишен и ще доведе до усложняване и необходимост от последваща интеграция в софтуерните продукти, което е недопустимо.

Поради тези причини се създава разширение на обобщените мрежи – *обобщени мрежи с приоритети, зависещи от времето* [5*]. Разширението използва същия алгоритъм за функциониране на преход и позволява оптимално изпълнение на симулации с приоритети на ядрата зависещи от времето.

Обобщените мрежи с приоритети, зависещи от времето, се дефинират като наредената четворка:

$$E = \langle \langle A, \pi_A, \pi_L, c, f, \theta_1, \theta_2 \rangle, \langle K, \pi_{K,T}, \theta_k \rangle, \langle T, t^0, t^* \rangle, \langle X, \Phi, b \rangle \rangle$$

Първият елемент на четворката се отнася до преходите на дадена ОМ:

- A – множеството от всички преходи на мрежата.
- π_A – функция, задаваща приоритетите на преходите. π_A приема като параметър преход от множеството A и връща неговия приоритет от множеството на естествените числа $\mathbb{N}$:

$$\pi_A: A \rightarrow \mathbb{N}.$$

- π_L – функция, задаваща приоритетите на позициите. L е множеството от всички позиции на мрежата. π_L приема като параметър позиция от множеството L и връща нейния приоритет от множеството на естествените числа $\mathbb{N}$:

$$\pi_L: L \rightarrow \mathbb{N}.$$

- c – функция, задаваща капацитетите на позициите. Капацитетът на дадена позиция е естествено число, което определя максималния брой на ядрата, които могат да се намират в един момент в дадена позиция. Функцията c приема като параметър позиция от множеството на всички позиции L и връща нейния капацитет от множеството на естествените числа:

$$c: L \rightarrow \mathbb{N}.$$

- f – функция, задаваща вярностната стойност на предикатите за прехода. Функцията приема като параметър множеството от предикати и връща булева стойност „истина“ или „лъжа“.
- θ_1 – функция, задаваща следващия момент t, в който преходът Z може да се активира. Стойността се изчислява в момента, в който приключи текущото активно състояние на прехода – t':

$$\theta_1(t') = t, t \in [\, t', T + t^*].$$

- θ_2 – функция, задаваща продължителността на текущото активно състояние на даден преход Z. Продължителността се изчислява в момента, в който преходът се активира – t':

$$\theta_2(t') = t, t \in [\, 0,\ t^*].$$

Вторият елемент от дефиницията на обобщената мрежа се отнася до ядрата:

- K – множество от ядрата на обобщената мрежа.
- $\pi_{K,T}$ – функция, задаваща приоритетите на ядрата. Функцията приема като параметър ядро от множеството на ядрата K и текущото време, което се мени в интервала $[\, T, T + t^*]$ (от началния момент на функционирането на обобщената мрежа до нейния край) и връща приоритет – число от множеството на естествените числа:

$$\pi_{K,T}: K\ x\ [\, T, T + t^*] \rightarrow \mathbb{N}.$$

- θ_k – функция, задаваща момента от време, в който едно ядро влиза в мрежата:

$$\theta_k(\alpha) = t, \alpha \in K, t \in [\, T, T + t^*].$$

При дефиницията на обобщената мрежа, времето се определя по фиксирана скала.

Третият елемент от наредената четворка описва тези компоненти от обобщената мрежа, които са свързани с времевата скала:

- T – момент от време, в който обобщената мрежа започва да функционира.
- t^0 – елементарна времева стъпка на времевата скала.
- t^* – продължителност на функционирането на обобщената мрежа.

Продължителността се измерва в елементарни времеви стъпки.

Последната част от дефиницията на обобщената мрежа се отнася до характеристиките:

- X – функция, задаваща началните характеристики на ядрата, които се получават при постъпването им в мрежата.
- Φ – функция, задаваща нова характеристика на едно ядро при постъпването му в изходна позиция на даден преход. X и Φ се наричат още характеристични функции.
- b – функция, задаваща максималния брой на характеристиките, които едно ядро може да носи по време на движението си в обобщената мрежа. Функцията приема като параметър ядро от множеството на ядрата K и връща максимален брой на характеристиките от множеството на естествените числа. В теорията се позволява тази стойност да бъде безкрайност, т.е. всяко ядро може да притежава неограничен брой характеристики:

$$b: K \rightarrow \mathbb{N}.$$

Разширението променя втория компонент от стандартната дефиниция на обобщена мрежа, като разширява значението на функцията, присвояваща приоритета на едно ядро, и я прави зависима от текущото време.

Накратко ще означаваме обобщените мрежи с приоритети, зависещи от времето, с абревиатурата – ОМПЗВ. Нека $\Sigma_{\text{ПЗВ}}$ е класът от всички обобщени мрежи с приоритети, зависещи от времето.

В областта на обобщените мрежи, една обобщена мрежа E се нарича консервативно разширение на обобщена мрежа G, ако резултатът от работата на E е идентичен с резултата от работата на G, но по някои от своите компоненти (приоритет на позициите, приоритет на ядрата, характеристични функции и др.) E представлява разширение на G.

Формулират се и се доказват следните теореми:

Теорема 2.2.1. (за представимост на стандартна обобщена мрежа като ОМПЗВ)

Всяка стандартна обобщена мрежа е обобщена мрежа с приоритети зависещи от времето, т.е. е в сила релацията:

$$\Sigma_{\text{ПЗВ}} \vdash \Sigma.$$

Теорема 2.2.2. (за консервативност на ОМПЗВ)

Класът $\Sigma_{\text{ПЗВ}}$ е консервативно разширение на класа на стандартните обобщени мрежи Σ, т.е.

$$\Sigma_{\text{ПЗВ}} \equiv \Sigma.$$

2.3. Обобщени мрежи с ядра с динамични приоритети

Въвежда се второ разширение на понятието обобщена мрежа – *обобщена мрежа с динамични приоритети* [6*].

Идеята за създаването на това разширение от една страна се дължи на програмния аспект на теорията, а от друга – на възможността за изграждане на обобщени мрежи с по-сложни зависимости между отделните компоненти. Например модел на диагностицирането и лечението на дадено заболяване. Подобен модел изисква промяна на приоритета на прием на даден пациент както по време, така и по резултати от направени изследвания, влошаване на здравословното състояние на пациента и много други.

Обобщените мрежи с динамични приоритети се дефинират аналогично на тези с приоритети, зависещи от времето. При тях функцията задаваща приоритетите на ядрата $\pi_{K,\varphi}$ приема като параметър ядро от множеството на ядрата K и произволен израз, и връща приоритет – число от множеството на естествените числа:

$$\pi_{K,\varphi}: K\ x\ \varphi\ \rightarrow \mathbb{N}.$$

Обобщените мрежи с динамични приоритети също е консервативно разширение на понятието обобщена мрежа.

Формулират се и се доказват следните теореми:

Теорема 2.3.1. (за представимост на стандартна обобщена мрежа като ОМЯДП)

Всяка стандартна обобщена мрежа е обобщена мрежа с динамични приоритети, т.е. е в сила релацията:

$$\Sigma_{\text{ЯДП}} \vdash \Sigma.$$

Теорема 2.3.2. (за консервативност на ОМЯДП)

Класът $\Sigma_{\text{ЯДП}}$ е консервативно разширение на класа на стандартните обобщени мрежи Σ. т.е.

$$\Sigma_{\text{ЯДП}} \equiv \Sigma.$$

2.4. Оператори и релации върху редуцирани обобщени мрежи

В тази част на дисертацията се формулират и доказват две теореми, свързани с операции и релации над редуцирани обобщени мрежи [9*].

За тази цел в дисертационния труд първо се описват операциите и релациите, дефинирани над преходи и обобщени мрежи в техния нередуциран вариант и се предоставя дефиницията на редуцирана обобщена мрежа.

Операции над редуцирани обобщени мрежи

В тази част на дисертационния труд се доказва, че операциите, дефинирани над обобщени мрежи, могат да бъдат дефинирани и върху редуцирани обобщени мрежи. Доказва се следната теорема:

Теорема 2.4.1. За всеки две обобщени мрежи $E_1, E_2 \in \Sigma$ и $\forall\ Y \in \Omega$ е в сила, че:

$$R_Y(E_1 \cup E_2) = R_Y(E_1) \cup R_Y(E_2)$$

Резултатът от прилагането на редуциращ оператор върху обединението на две обобщени мрежи е равно на обединението на съответните редуцирани обобщени мрежи.

Следствие 1. За всеки две обобщени мрежи $E_1, E_2 \in \Sigma$ и $\forall\ Y \in \Omega$ е в сила, че:

$$R_Y(E_1 \circ E_2) = R_Y(E_1) \circ R_Y(E_2)$$

Релации над редуцирани обобщени мрежи

В тази част на дисертационния труд се доказва, че релациите дефинирани над обобщени мрежи могат да бъдат дефинирани и върху редуцирани обобщени мрежи. Доказва се следната теорема:

Теорема 2.4.2. За всеки две обобщени мрежи $E_1, E_2 \in \Sigma$, всяка релация Rel, дефинирана над обобщени мрежи, и $\forall\, Y \in \Omega$ е в сила, че:

$$Rel(E_1, E_2) \Rightarrow Rel(R_Y(E_1), R_Y(E_2))$$

Ако две обобщени мрежи са в релация, прилагането на редуциращи оператори върху тях запазват релацията.

2.5. Верификация на обобщени мрежи без времеви компонент на преходите

Определението за верификацията на програмното осигуряване според стандарта на софтуерна верификация и валидация IEEE 1012-2004 [65] определя верификацията на програмното осигуряване като техника, която проверява дали артефактите (техническо задание, модел на предметна област, описание на архитектурата, програмен код, потребителска документация и др.), създадени в хода на разработката на програмното осигуряване, съответстват на други артефакти, определени като начални (входни) данни, а също дали тези артефакти съответстват на правилата и стандартите.

В точка 2.5 на дисертацията, се дефинира първият метод за верификация на обобщени мрежи. Методът е формален, а обобщените мрежи, които ще могат да бъдат верифицирани чрез него, са мрежите с преходи, независещи от времето (без времеви компонент). Едновременното (паралелното) изпълнение на някои преходи се осъществява чрез компонента □ (тип) на преходите.

В разглежданите обобщени мрежи се дефинират три променливи от тип вектор:

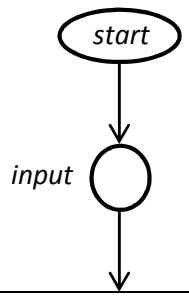
1. Входен вектор $\bar{x} = (x_1, x_2, \dots, x_n)$, който задава стойности на инициализиращите характеристики $x_1, x_2, \dots, x_n$ на ядрата на обобщената мрежа.
2. Работен вектор $\bar{y} = (y_1, y_2, \dots, y_m)$, който се използва като временна памет за пресмятанията и описва промяната на стойностите на характеристиките $y_1, y_2, \dots, y_m$ на ядрата на обобщената мрежа.
3. Изходен вектор $\bar{z} = (z_1, z_2, \dots, z_k)$, който задава стойностите на изходните характеристики $z_1, z_2, \dots, z_k$ на ядрата след завършване изпълнението на обобщената мрежа.

Съответно различаваме три области:

1. Входна област $D_{\bar{x}} = D_{x_1} \times D_{x_2} \times \dots \times D_{x_n}$.
2. Работна област $D_{\bar{y}} = D_{y_1} \times D_{y_2} \times \dots \times D_{y_m}$.
3. Изходна област $D_{\bar{z}} = D_{z_1} \times D_{z_2} \times \dots \times D_{z_k}$.

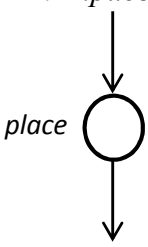
Обобщените мрежи, които ще верифицираме чрез техника, следваща метода на Флойд за верификация на блок-схеми [73], се състоят от следните сегменти:

1. *start сегмент*



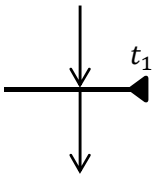
В позиция *input* влиза ядро с характеристика $(\bar{x}, \bar{y}) = (\bar{x}, f(\bar{x}))$, а $f(\bar{x})$ е тотална функция: $D_{\bar{x}} \to D_{\bar{y}}$.

2. *присвояващ сегмент*



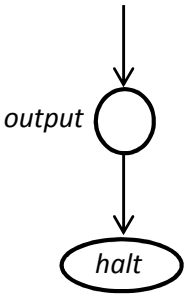
В позиция *place* влиза ядро с характеристика $\bar{y}$ и се променя в $h(\bar{x}, \bar{y})$, т.е. в позиция *place* се изпълнява операторът за едновременно присвояване $\bar{y} = h(\bar{x}, \bar{y})$.

3. *проверяващ сегмент* (*сегмент за преход*)



Матрицата r на прехода t_1 задава условията за преход. Още веднъж ще отбележим, че преходът t_1 не зависи от времето.

4. *halt сегмент*



Ядрото, което постъпва в позиция *output*, задава стойност за $\bar{z}$, т.е. реализира присвояването $\bar{z} = g(\bar{x}, \bar{y})$.

Верификацията на обобщена мрежа зависи от два зададени предиката:

1. Входен предикат, който ще бележим с $\varphi(\bar{x})$. Предикатът е тотален върху $D_{\bar{x}}$ и описва кои елементи (данни) могат да бъдат използвани като стойности на инициализиращите характеристики $x_1, x_2, \dots, x_n$ на ядрата на обобщената мрежа.
2. Изходен предикат, който ще бележим с $\psi(\bar{x}, \bar{z})$. Предикатът е тотален върху $D_{\bar{x}} \times D_{\bar{z}}$ и описва връзките между входните и изходните стойности на характеристиките на ядрата на ОМ при завършване изпълнението на обобщената мрежа.

Предикатите $\varphi(\bar{x})$ и $\psi(\bar{x}, \bar{z})$ задават входно-изходната спецификация, относно която ще се верифицира обобщената мрежа.

Дефиниция 1

Казваме, че една обобщена мрежа P от вида, описан по-горе, е *частично коректна* относно предикатите $\varphi(\bar{x})$ и $\psi(\bar{x}, \bar{z})$, ако за всеки вектор $\bar{\varepsilon}$, за който входният предикат $\varphi(\bar{\varepsilon})$ е $true$ и изпълнението на обобщената мрежа P завърши, то и изходният предикат $\psi(\bar{\varepsilon}, P(\bar{\varepsilon}))$ е $true$.

Дефиниция 2

Казваме, че една обобщена мрежа P от вида, описан по-горе, завършва изпълнението си над φ, ако за всеки вход $\bar{\varepsilon}$, за който $\varphi(\bar{\varepsilon})$ е $true$, изпълнението на обобщената мрежа P завършва (спира).

Дефиниция 3

Казваме, че една обобщена мрежа P от вида, описан по-горе, е тотално коректна относно предикатите $\varphi(\bar{x})$ и $\psi(\bar{x}, \bar{z})$, ако за всеки вектор $\bar{\varepsilon}$, за който входният предикат $\varphi(\bar{\varepsilon})$ е $true$, изпълнението на обобщената мрежа завършва изпълнението си над $\varphi(\bar{x})$ и изходният предикат $\psi(\bar{\varepsilon}, P(\bar{\varepsilon}))$ е $true$.

Верификацията на една обобщена мрежа от вида, описан по-горе, относно зададената входно-изходна спецификация, се свежда до доказване на нейната тотална коректност относно тази спецификация.

Частична коректност

Ще представим техника за доказване, че една обобщена мрежа, от вида описан по-горе, е частично коректна относно входния предикат $\varphi(\bar{x})$ и изходния предикат $\psi(\bar{x}, \bar{z})$.

Извършват се следните три стъпки:

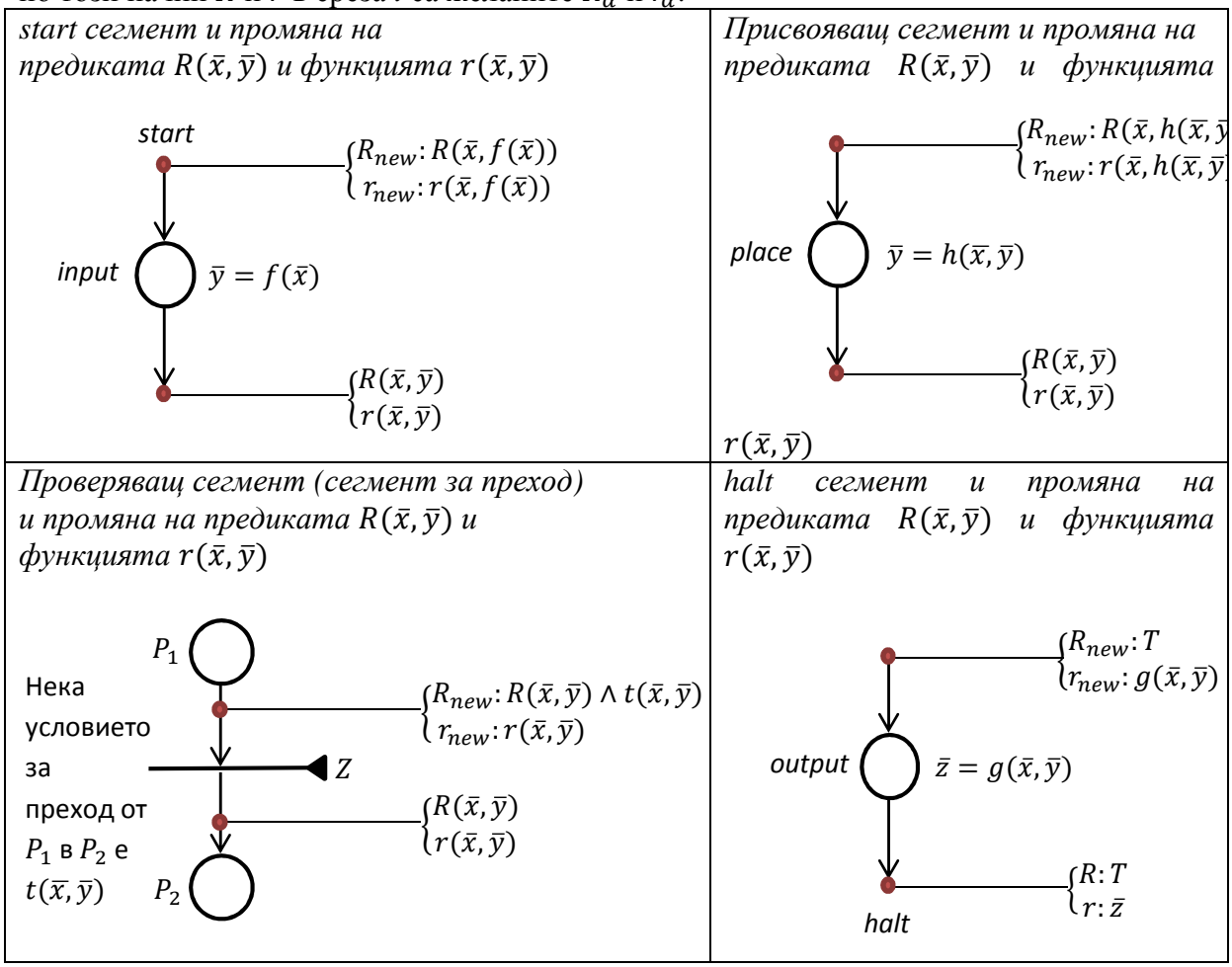
Стъпка 1. Срезове

Всяка примка на обобщената мрежа се свързва със срез. Към тези срезове се добавят *start* и *halt* срезовете. Така обобщената мрежа се покрива с краен брой пътища – започващи от срез, завършващи в срез и несъдържащи междинни срезове.

За всеки път α от срез i до срез j се намират предикат $R_\alpha(\bar{x},\bar{y})$ и вектор $r_\alpha(\bar{x},\bar{y})$. Предикатът $R_\alpha(\bar{x},\bar{y})$ изразява условието за преминаване през този път, а векторът $r_\alpha(\bar{x},\bar{y})$ описва промяната на стойностите на $\bar{y}$, предизвикана от преминаването на пътя α. Така $R_\alpha(\bar{x},\bar{y})$ е предикат върху $D_{\bar{x}} \times D_{\bar{y}}$, а $r_\alpha(\bar{x},\bar{y})$ е функция, която изобразява $D_{\bar{x}} \times D_{\bar{y}}$ в $D_{\bar{y}}$.

R_α и r_α се представят посредством функциите и условията в сегментите, съставящи пътя α. Прост метод за тяхното пораждане е да се използва техниката на заместване при движение в обратна посока [73].

Първоначалните стойности на R_α и r_α (стойностите в среза j) са съответно $true$ (ще го означаваме съкратено с T) и $\bar{y}$. След това на всяка стъпка, след движение в обратна посока (към среза i) се използват старите стойности на R и r и се получават нови. Получаването на нови стойности в сегментите е показано на Фиг. 1. Получените по този начин R и r в среза i са желаните R_α и r_α.



Фиг. 1 Промяна на предиката $\boldsymbol{R(\bar{x},\bar{y})}$ и функцията $\boldsymbol{r(\bar{x},\bar{y})}$ в различните видове сегментите

Стъпка 2. Индуктивни твърдения

С всеки срез i се свързва предикат $p_i(\bar{x},\bar{y})$ наричан *индуктивно твърдение*. Неговото предназначение е да характеризира връзката между стойностите на характеристиките $\bar{x}$ и $\bar{y}$ на ядрата в тази точка, т.е. $p_i(\bar{x},\bar{y})$ трябва да има свойството всеки път, когато изпълнението на обобщената мрежа достигне точка i, $p_i(\bar{x},\bar{y})$ да бъде $true$ за текущите

стойности на $\bar{x}$ и $\bar{y}$ в тази точка. Входният предикат $\varphi(\bar{x})$ се свързва със среза START, а изходният предикат $\psi(\bar{x},\bar{z})$ – със среза HALT.

Стъпка 3. Верифициращи условия

Построяват се и се доказва верността на следните верифициращи условия:

1. За всеки път α, за който i е срезът START,
   $\forall\bar{x}\,[\varphi(\bar{x}) \wedge R_\alpha(\bar{x}) \Rightarrow p_j(\bar{x}, r_\alpha(\bar{x}))]$
2. За всеки път α, започващ от срез i и завършващ в срез j,
   $\forall\bar{x}\forall\bar{y}\,[p_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow p_j(\bar{x}, r_\alpha(\bar{x},\bar{y}))]$
3. За всеки път α, за който j е срезът HALT,
   $\forall\bar{x}\forall\bar{y}\,[p_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow \psi(\bar{x}, r_\alpha(\bar{x},\bar{y}))]$

Ако построените условия за всички пътища, покриващи обобщената мрежа, са удовлетворени, обобщената мрежа е частично коректна относно $\varphi(\bar{x})$ и $\psi(\bar{x},\bar{z})$. Така формулираме следната теорема.

Теорема 2.5.1. (Метод на индуктивните твърдения на Флойд)

За дадена обобщена мрежа P от вида, описан по-горе, за входен предикат $\varphi(\bar{x})$ и за изходен предикат $\psi(\bar{x},\bar{z})$ се извършват следните стъпки:

1) Срязват се примките.
2) Намира се подходящо множество от индуктивни твърдения.
3) Построяват се верифициращи условия.

Ако всички верифициращи условия имат стойност *true*, то P е частично коректна спрямо φ и ψ.

Доказателството на формулираната теорема за частична коректност на обобщена мрежа е подобно на това в [44]. В [44] се доказва още, че методът на Манна е пълен.

Завършване на изпълнението

Представи техника за доказване, че една обобщена мрежа от вида описан по-горе, завършва изпълнението си относно входния предикат $\varphi(\bar{x})$. Техниката е предложена от Флойд за блок-схеми.

Извършваме следните три стъпки:

Стъпка 1. Добри твърдения

Избира се множество от точки (срезове), които срязват примките, и с всеки срез i се свързва твърдение $q_i(\bar{x},\bar{y})$, което е добро твърдение [73] , т.е.

1. За всеки път α от среза START до среза j (несъдържащ междинни срезове) е изпълнено
   $$\forall\bar{x}[\varphi(\bar{x}) \wedge R_\alpha(\bar{x}) \Rightarrow q_j(\bar{x}, r_\alpha(\bar{x}))].$$
2. За всеки път α от среза i до среза j (несъдържащ междинни срезове) е изпълнено
   $$\forall\bar{x}\forall\bar{y}[q_i(\bar{x},\bar{y}) \wedge R_\alpha(\bar{x},\bar{y}) \Rightarrow q_j(\bar{x}, r_\alpha(\bar{x},\bar{y}))].$$

Стъпка 2. Добре ограничено множество

Избира се добре ограничено множество $(W,\prec)$ [73] и с всеки срез i се свързва частична функция $u_i(\bar{x},\bar{y})$, изобразяваща $D_{\bar{x}} \times D_{\bar{y}}$ в W, която е добра функция [73] , т.е. за всеки срез i е изпълнено
$$\forall\bar{x}\forall\bar{y}[q_i(\bar{x},\bar{y}) \Rightarrow (u_i(\bar{x},\bar{y}) \in W)].$$

Стъпка 3. Проверка на условията за завършване на изпълнението

Показва се, че условията за завършване на изпълнението са в сила. Това означава, че за всеки път α от срез i до срез j (несъдържащ междинни срезове), който е част от някаква примка, е изпълнено

$$\forall \bar{x} \forall \bar{y} [q_i(\bar{x}, \bar{y}) \wedge R_\alpha(\bar{x}, \bar{y}) \Rightarrow (u_i(\bar{x}, \bar{y}) \succ u_j(\bar{x}, r_\alpha(\bar{x}, \bar{y})))].$$

Това означава, че след всяко изпълнение на път, който е част от примка, стойностите на функциите u_i, които са свързани със срезовете му, строго намаляват. Тъй като $(W, \prec)$ е добре ограничено множество, т.е. не съществува безкрайна строго намаляваща редица от елементи на W, то броят на изпълнението на пътя е краен. Описаното по-горе може да се обобщи чрез следната теорема.

Теорема 2.5.2 (Метод на добре ограничените множества (Флойд))
За дадена обобщена мрежа P от вида, описан по-горе, и входен предикат $\varphi(\bar{x})$ се прилагат следните стъпки:

1) Примките се срязват и се намират „добри“ индуктивни твърдения.
2) Избира се добре ограничено множество и се намират „добри“ частични функции.
3) Проверяват се условията за завършване на изпълнението.

Ако всички условия за завършване на изпълнението са *true*, то P завършва изпълнението си над φ.

Доказателството на формулираната теорема за завършване на изпълнението на ОМ е подобно на това в [67, 74].

В дисертационния труд се показва и пример илюстриращ метода за верификация на обобщени мрежи без времеви компонент на проходите.

2.6. Постигнати резултати в Глава 2

Постигнатите резултати в Глава 2 могат да се причислят към развитието на теорията на обобщените мрежи. В тази глава се прави анализ и се решават три от поставените задачи в дисертацията.

В точка 2.1 се предлага нов оптимизиран алгоритъм, който да позволи имплементацията на симулации от различен тип, реализирайки автоматизирана възможност за сливане на ядра.

Дефинираните консервативни разширения също разширяват значително възможностите на изграждените модели чрез промяна на приоритети на ядрата спрямо външни фактори, без да е необходима промяна на алгоритмите за функциониране на преход и обобщена мрежа, което е особено важно във връзка с изграждането на софтуерния продукт. Динамичната промяна на приоритетите от своя страна позволява и изграждане на модели, които могат да бъдат симулирани в реално време и да протичат по различен начин в зависимост от промяна на условията.

В точка 2.4 се доказват две теореми за запазване на операциите и релациите върху редуцирани обобщени мрежи, което дава добра перспектива за изграждането в началото на редуциращи оператори, а по-късно и на релации и оператори в софтуерния продукт.

Накрая се дефинира нов метод за верификация на обобщени мрежи без времеви компонент, който увеличава сигурността при изградените модели. Това позволява създаването на симулации като част от по-големи софтуерни продукти и в области, където това е задължително. В този контекст авторът въвежда понятията частична коректност, завършване на изпълнението и тотална коректност на обобщени мрежи и доказва две независими теореми, които указват необходимите и достатъчните условия за частична коректност и завършване на изпълнението на обобщените мрежи.

Глава 3. Приноси в програмната реализация на апарата на обобщените мрежи

В Глава 3 на дисертационния труд се описват част от направените приноси в програмната реализация на апарата на обобщените мрежи.

В началото, в точка 3.1 е представена имплементацията на възможността за динамично разцепване и сливане на ядра като част от интеграцията на модифицирания алгоритъм за функциониране на преход, където сливането на ядра е позволено.

В точка 3.2 се дефинира понятието *GN API*, описва се неговата реализация и интерфейс и се дава пример за негово използване.

В точка 3.3 се представят две въведени разширения на софтуерния продукт *GN IDE* - разпознаване на класове редуцирани обобщени мрежи и редуциращи оператори.

Накрая се обобщават всички постигнати резултати в програмната реализация на апарата на обобщените мрежи и се дефинират нови задачи за бъдеща работа.

3.1. Интегриране на алгоритъм за функциониране на преход, където сливането на ядра е позволено

Една от задачите на дисертационния труд е подобряване на симулационната част на интегрираната среда за разработка на обобщеномрежови модели *GN IDE* [2*]. Някои от интегрираните среди за разработване на софтуер (IDE) представляват софтуерни приложения, които предоставят цялостна среда за разработка на софтуер, включително: редактор на код, автоматизирано построяване на изходното приложение, дебъгер, компилатор и интерпретатор. В този смисъл подходът, избран за реализацията на *GN IDE* е да се позволява изграждането, стартирането и извършването на симулация единствено чрез самото IDE, без да е необходимо свързването към отдалечен сървър, използвайки мрежа. За тази цел в *GN IDE* първо се премахва цялата остаряла функционалност като връзката с *TickerServer*, използването на езика GNTCFL и други. След това се изгражда симулационна част, наречена *EmbeddedSimulation*, чиято роля е извършването на определен брой стъпки от симулацията и връщане на събитията, които са се случили вследствие на тях.

Изграждането на класа *EmbeddedSimulation* е започнато от Димитър Димитров, което още тогава противоречи на идеята за използване на *TickerServer*. Изграждането не е завършено, тъй като имплементацията не позволява нормалното протичане на симулация без да е необходима външна намеса в кода на приложението, или заобикаляйки проблемните места, чрез използването на характеристичните функции, предикатите и други. Един подобен пример е разцепването на ядрата.

Поради тази причина се създава нова версия на симулационната част на *GN IDE*, като се реализира пълният алгоритъм за функциониране на преход, където сливането на ядра е позволено, и алгоритъмът за функциониране на обобщена мрежа. Към симулационната част (изградения алгоритъм) се добавя и функционалност, която създава всички настъпили в мрежата събития, вследствие на изпълнението на стъпки от нейното функциониране. Тези събития са особено важни за изграждането на *GN API*, където те трябва да се връщат на потребителя в удобен вид за обработка.

Важно пояснение е, че при изграждането на симулацията се правят и няколко малки разширения на:

- `CharacteristicHistory` - позволява се създаването на характеристика с история, чийто капацитет е стойност цяло число или безкрайност. Последното е важно,

тъй като е възможно броят на стойностите на определени характеристики да не е ясен преди извършването на симулацията.

- `Characteristic` – започва да се поддържа векторен и обектен тип характеристики на ядрата, например:

Векторният тип може да бъде много подходящ за случаите, когато характеристиката носи резултати от множество измервания, никое от които не е важно само по себе си.

Обектният тип може да бъде подходящ за случаите, когато се използва *GN API* или моделът е част от по-голяма система, където данните се съхраняват в обекти.

3.2. Generalized Net API

Една от основните идеи на дисертационния труд е да предостави възможност за изграждане на реални, практически приложими обобщеномрежови модели от всякакъв вид и използването им като част от по-големи софтуерни решения. За тази цел трябва да бъде възможно конструиране и проиграване (симулиране) на модели изцяло чрез средствата на програмен код. Тази част от дисертацията решава този проблем, а именно описва създадения за целта обобщеномрежов приложно-програмен интерфейс – *Generalized Net Application Programming Interface* (*GN API*) [3*].

GN API предоставя възможност за изграждане и симулиране на модели, написани на Java. Характеристичните и предикатните функции на всеки модел, по подобие на *GN IDE*, могат да бъдат дефинирани на езика JavaScript.

GN API и *GN IDE* могат да се използват като независими софтуерни продукти. Кодът е публикуван в Github под лиценз MIT [80]. Двете реализации използват общи модели и симулационна част. За автоматизация на build процеса, в *GN IDE*, се използва Ant скрипт [14]. Скриптът представлява XML файл (build.xml), в който са дефинирани различни цели (*targets*), една от които е създаване на JAR файл. След създаването на *GN API* се добавя нова цел в скрипта, която генерира друг JAR файл, включващ единствено тази част от кода, която е необходима на *GN API*. Това позволява всяка оптимизация по алгоритъма за функциониране на обобщена мрежа и всяко разширение на моделите на *GN IDE* да бъдат „видими“ и в *GN API* единствено чрез разширяване (ако това е необходимо) на интерфейса и обратно.

Структурата на *GN API* се състои от два компонента *GN Builder* и *GN API* интерфейс.

Първият компонент *GN Builder*, се състои от един Java клас – `GeneralizedNetBuilder`. Той реализира цялата функционалност по изграждането и промяната на един обобщеномрежови модел. Основните стъпки по реализацията на един обобщеномрежов модел е последователното създаване и добавянето към модела на преходи, позиции, ядра и техните свойства.

За целта `GeneralizedNetBuilder` реализира статични класове `TransitionBuilder`, `PlaceBuilder` и `TokenBuilder`, всеки от които има по един капсулиран обект от съответния тип (`Transition`, `Place`, `Token`) и методи за добавяне и промяна на неговите свойства и елементи. Реализацията е направена така, че всеки един от статичните класове разширява предишния, което в комбинация с факта, че методите връщат обекти от тип `Builder`, позволява верижно извикване на методи на произволно ниво. Например дава се възможност за добавяне на преход към мрежата, като се използва строител на позиция и т.н.

Дисертационният труд подробно описва `GeneralizedNetBuilder`, `TransitionBuilder`, `PlaceBuilder` и `TokenBuilder` класовете и се показва кодът необходим за изграждане на примерна обобщена мрежа.

GN API интерфейсът предоставя възможност за създаване и контрол на обобщеномрежова симулация и получаване на резултата от изпълнението ѝ в подходяща за обработване форма.

Дисертационният труд описва класовете реализиращи симулацията (`JavaSimulation`), нейното стартиране (`GeneralizedNetFacade`), слушател на събитията настъпили в следствие на симулацията (`SimulationEventsListener`) и различните Java обобщеномрежови събития (`JavaEnterEvent, JavaLeaveEvent, JavaMoveEvent`).

Освен интерфейса, свързан със създаването и управлението на симулацията, *GN API* предоставя функционалност за създаване на Java обекти и достъп до техните свойства. Основните обекти, които трябва да могат да се създават и да се достъпват, са Java обобщена мрежа, преход, позиция, ядро, характеристика и различни функции. За целта в дисертационния труд се дефинират и описват следните класове - `JavaGeneralizedNet`, `JavaTransition`, `JavaPlace`, `JavaToken`, `JavaCharacteristic`, `JavaFunction`, `JavaFunctionReference`, `JavaFunctionRunner`.

За да се покажат част от възможностите на *GN API* интерфейса, дисертационният труд описва как се стартира симулация по създадената `JavaGeneralizedNet gn` мрежа в двата описани дотук варианта (със слушател на събития и без такъв) и демонстрира как може да се симулира една стъпка от изпълнението на мрежата.

3.3. Разпознаване на класове редуцирани обобщени мрежи и интегриране на редуциращи оператори в *GN IDE*

В точка 3.3 се представя софтуерната реализация на две нови функционалности в *GN IDE*. Първата е свързана с възможността за редуциране на компоненти от вече създадена обобщена мрежа, без загуба на информация така, че да бъде възможно премахване на приложените редуциращи оператори [4*]. Втората функционалност е свързана с възможността за разпознаване на класове редуцирани обобщени мрежи при зареждането им в GN IDE [4*]. Реализацията на двете нови функционалности позволява проиграването на различни тестови сценарии чрез изграждането на един единствен обобщеномрежов модел. Нещо повече, редуцирането на определени компоненти води до оптимизация на функционирането на преходите и съответно на цялата обобщена мрежа чрез намаляване на броя на извършваните операции от алгоритъм за функциониране на преход, където сливането на ядра е позволено.

Интегриране на редуциращи оператори в GN IDE

Интеграцията на редуциращите оператори в *GN IDE* е свързано с промяна на схемата на обобщеномрежовия модел. За реализацията се поставят две изисквания:

- съвместимост на реализираните до момента обобщеномрежови модели с новата схема, което да позволи тяхната симулация и редуцирането им след направените промени;
- възможност за премахване на приложените редуциращи оператори и възвръщане на мрежата в нейно предишно състояние, т.е. редуциращите оператори не трябва да изменят моделите.

Понастоящем всяка обобщена мрежа представлява GN XML файл (XGN), който съдържа описанието на обобщеномрежовия модел в дефинираната схема.

За реализация на редуциращите оператори се добавя нов компонент в схемата *reducedComponents* (Фиг. 9).

```
<xsd:element name="reducedComponents" minOccurs="0" maxOccurs="1">
    <xsd:complexType>
        <xsd:sequence maxOccurs="1">
            <xsd:element name="T1" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
            <xsd:element name="T2" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
            <xsd:element name="M" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
            <xsd:element name="TRANSITION_TYPE" minOccurs="0" maxOccurs="1">
                <xsd:complexType/>
            </xsd:element>
```

Фиг. 9 Новият компонент *reducedComponents* в XGN схемата

Пример за *reducedComponents* в обобщеномрежовия XML файл е показан по-долу.

```
<reducedComponents>
        <M/>
        <TRANSITION_PRIORITIES/>
        <T/>
</reducedComponents>
```

В описанието на *reducedComponents*, участват множество тагове (деца на *reducedComponents*), всеки от които е празен и се описва единствено чрез своето име. Компонентът *reducedComponents* не е задължителен за схемата, т.е. може да отсъства изцяло или да има едно, или повече деца. Липсата на таг с определено име означава, че елементът с това име не е редуциран.

Последното условие гарантира съвместимостта на реализираните до този момент обобщеномрежови модели с новата схема, като по подразбиране се приема, че те не са редуцирани.

В допълнение, софтуерната реализация не позволява редуцирането на преходите (A), техните входни и изходни позиции (A_1, A_2) и множеството на ядрата (K). Тези компоненти задават структурата на обобщената мрежа и са базови за нейната цялостност и функциониране, поради което те не подлежат на редуциране.

Представянето на списъка от редуциращи оператори в кода на *GN IDE* се реализира посредством масив от битове, като реализацията на редуцирането се осъществява в нов независим слой (layer) и се извършва чрез промяна на стойностите на битовия масив. Впоследствие наличието на редуцирани компоненти се отчита единствено от алгоритмите за функциониране на преход и обобщена мрежа, като ако е възможно, определени стъпки от тях са пропускани от симулацията. Тази реализация позволява редуциращите оператори да бъдат реализирани напълно независимо от останалата част на софтуерния продукт, да могат да се прилагат в комбинация с различни оператори, релации и други, да е възможно премахване на приложен редуциращ оператор без загуба на информация за обобщеномрежовия модел.

В дисертационния труд се описват и двете възможности за прилагането и премахването на редуциращи оператори върху обобщеномрежови модел:

- чрез директна модификация на XGN файла;
- чрез потребителския интерфейс на GN IDE.

Разпознаване на класове редуцирани обобщени мрежи

Разпознаването на класа редуцирани обобщени мрежи на един обобщеномрежов модел се изразява в това да се провери кои от компонентите на мрежата липсват. Важно уточнение е, че една мрежа се счита за редуцирана тогава и само тогава, когато пряко е указано прилагане на редуциращ оператор върху мрежата. Използването на стойностите по подразбиране на някой от компонентите не се счита за редуциране, тъй като е възможно това да бъде случайно следствие от текущите данни на симулацията и не може да се приеме за цялата мрежа.

Разпознаването на класове редуцирани обобщени мрежи може да се осъществи отново след прочитане директно на XGN файла, като се провери кои тагове присъстват в *reducedComponents* компонента на схемата или чрез потребителския интерфейс. Нещо повече, при редуциране на компонент от обобщената мрежа, той и цялата информация свързана с него не е видима за потребителя в никой от изгледите на *GN IDE*. Тоест, независимо, че е възможно премахването на редуциращите оператори и че те не въздействат пряко върху моделите на обобщената мрежа, потребителският интерфейс предоставя напълно обновен изглед, който не включва никаква информация, която няма да бъде използвана при симулацията на редуцираната обобщена мрежа.

3.4. Постигнати резултати в Глава 3

Постигнатите резултати в Глава 3 могат да се причислят към развитието на програмния аспект на обобщените мрежи. В главата се прави анализ и се решават две от поставените задачи в първа глава на дисертацията.

В точка 3.1, се описват всички направени нововъведения по симулационната част на *GN IDE* и интеграцията на новия алгоритъм за функциониране на преход, където сливането на ядра е позволено.

В точка 3.2 за първи път се описва реализацията на *Generalized Net API*. *GN API* разполага с интерфейс, който позволява изграждане и симулацията на обобщеномрежови модели изцяло чрез код. В частност, моделите се описват на Java, а характеристичните и предикатните функции на всеки модел, по подобие на *GN IDE*, са дефинирани на езика JavaScript.

В точка 3.3 се описва и реализацията на две разширения на *GN IDE* – въвеждането на редуциращи оператори, като продължение на доказаните в точка 2.4 теореми и разпознаване на класове редуцирани обобщени мрежи. Направената реализация разширява съществуващата схема, като запазва съвместимостта на реализираните до този момент обобщеномрежови модели с новата схема. Прилагането на редуциращи оператори върху моделите не води до загуба и промяна на данни в моделите, което позволява изграждането на различни тестови сценарии с реализацията на единствен обобщеномрежов модел.

Всички тези промени и множеството малки подобрения в софтуера предоставят възможност за по-лесното изграждане на симулации от различни области без да се налага модификация на кода за тяхното осъществяване.

Глава 4. Приложения на обобщените мрежи за моделиране на реални процеси

В Глава 4 на дисертационния труд, се илюстрират възможностите на обобщените мрежи като мощен и всеобхватен инструмент за моделиране, анализ и дизайн на всички видове процеси с дискретни събития, които се развиват с течение на времето.

Обобщените мрежи могат ефективно и ефикасно да моделират различни аспекти на процесите, чието поведение с течение на времето (т.е. динамично) се задейства и/или повлиява от някои външни и вътрешни събития. Това ги прави изключително полезен инструмент в много различни области. Поради тази причина в тази глава се излагат три реализирани обобщеномрежови модели на различни процеси, като всеки един от тях се описва от гледна точка на теорията и се разглежда неговата програмна реализация. Спираме се на три нови области на приложение на апарата на обобщените мрежи, а именно: пречистване на отпадни води, защита на програмното осигуряване и измервателен процес на плоскопаралелни краищни мерки за дължина. Илюстрирано е прилагане на метода за верификация на обобщени мрежи без времеви компонент на преходите, описан в точка 2.5 за верификация на обобщена мрежа, реализираща защита на програмното осигуряване.

В края се извършва обобщение на всички постигнати резултати вследствие на разработените обобщеномрежови модели в различните научни направления.

4.1. Обобщеномрежов модел за пречистване на отпадни води

Основната задача, в тази част на дисертационния труд, е да се реализира изследване на процесите на пречистване на отпадни води, на база на което да се създаде симулация с възможност за анализ на данни на същите процеси.

В най-общ план процесите на пречистване на отпадни води могат да бъдат групирани в три основни технологични етапа:

- Механична обработка на водите;
- Физикохимична обработка на водите;
- Биологична обработка на водите.

Първият етап на пречистване е механичната обработка. Целта на този етап е премахване на всички по-големи замърсители. След механичното пречистване водата влиза във втори етап на пречистване, а именно физикохимична обработка. Тук се осъществяват процесите на коагулация и флотация. Накрая пречистената вода се подава в звено за биологично пречистване (биобасейн с въглеродна асимилация). Тук се осъществяват всички процеси от въглеродната асимилация като CO_2, H_2O и NH_4 се получават като крайни продукти на окислението. Пречистената отпадъчна вода се отделя от активната утайка в така наречен вторичен радиален утаител. Потокът от обработваната вода попада в изходната позиция като пречистена вода. Всички утайки биват преработвани и попадат в друга изходна позиция.

Тази задача е реализирана чрез средствата на обобщените мрежи. Избраният инструмент е подходящ, тъй като процесът е дискретен, с възможност за множество асинхронни операции. Нещо повече, процесът зависи от времето, моделът трябва да може да бъде разширен (позволява се от обобщените мрежи, тъй като един преход може да се замести с цяла обобщена мрежа), да се извършва симулацията му в реално време или по определи данни.

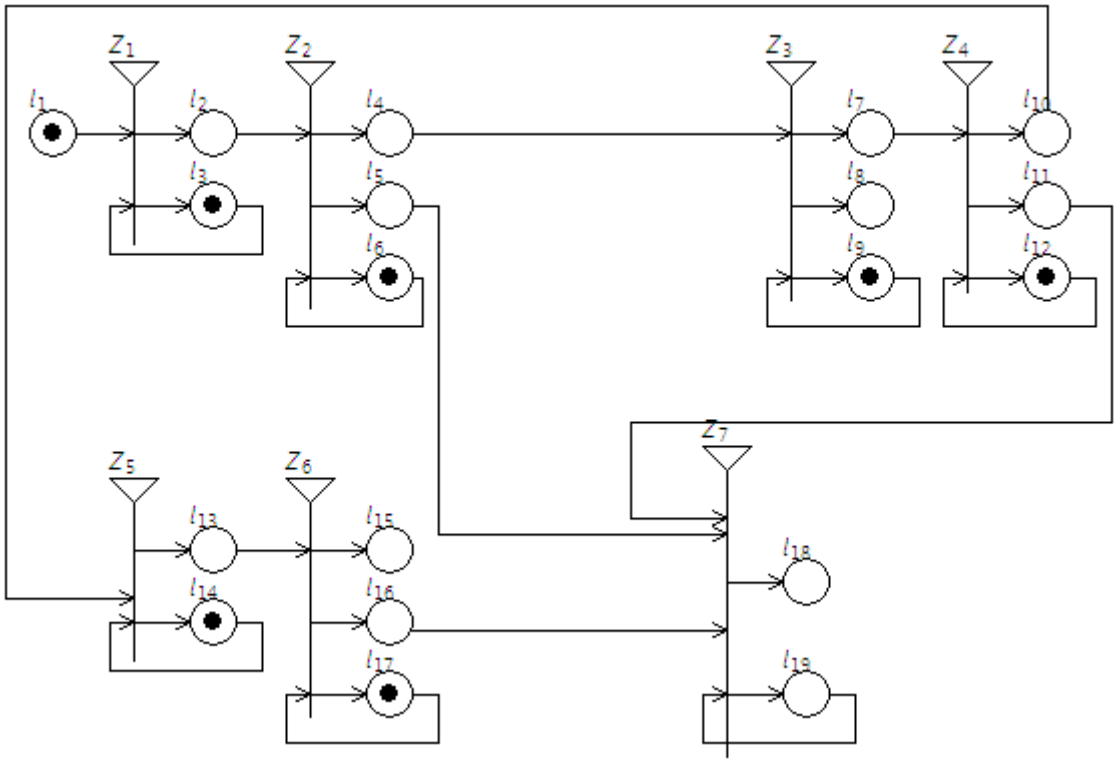
Целта на тази практическа реализация е създаването на обобщеномрежов модел, описващ целия процес на пречистване на водата, симулирането на този модел в

интегрираната среда за разработка *GN IDE*, описана по-горе, и демонстрирането на възможностите, предоставени от софтуера за наблюдения на динамиката на модела и визуализация на резултатите от различни етапи на пречистването [8*].

За създаването на обобщеномрежовия модел отначало задълбочено беше проучен процесът, който трябваше да бъде моделиран.

За целите на разработваната обобщеномрежова симулация са използвани данни за пречистване на водата от реален индустриален обект, които се събират от измерванията, правени всеки ден в период от дванадесет месеца. Експерименталните данни дават информация за количеството вода, pH, химично потребен кислород (ХПК), бензин, различни механични примеси, PO_4P, NH_4N и други.

Разработеният обобщеномрежов модел симулира описаната по-горе схема на пречистване на водата. Моделът е показан на Фиг. 13.



Фиг. 13 Обобщеномрежов модел на пречистване на водата

Обобщената мрежа се състои от седем прехода с имена съответно $Z_1 \ldots Z_7$.

$$A = \{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\}$$

- Z_1 – преходът описва постъпването на замърсени отпадни води.
- Z_2 – преходът реализира филтрация на отпадните води, през решетки (сита), което води до отстраняване на едрите примеси.
- Z_3 – преходът описва пречистването на отпадни води от масла, нефт и мазнини чрез нефтоутаител.
- Z_4 – преходът реализира първичния утаител. Той пречиства водата от първичните утайки и отпадъци.

Първите четири прехода реализират първата стъпка от процеса на пречистване на водата – механична обработка на водите.

- Z_5 – преходът реализира процеса на коагулация и флотация, което е втората стъпка от процеса на пречистване на замърсените отпадни води – физикохимична обработка на водите.

- Z_6 – преходът реализира процеса на пречистване в биобасейн с въглеродна асимилация.
- Z_7 – преходът реализира процеса на пречистване на получените от различните етапи утайки.

Последните два прехода реализират последната, трета стъпка от процеса на пречистване на водата – биологична обработка на водите.

Накрая на симулацията, пречистената вода влиза в изходната позиция l_{15} на обобщената мрежа. Всички натрупани от пречистването утайки също се обработват и влизат в изходна позиция l_{18}.

Първоначално β, χ, δ, ε, ϕ_1, γ ядра стоят съответно в позиции l_3, l_6, l_9, l_{12}, l_{14} и l_{17}. Тези ядра изпълняват ролята на слушатели на събития в обобщената мрежа. Те стоят в тези позиции през цялото време на функциониране на обобщената мрежа, като могат да получават нови стойности за своите характеристики. Описаните ядра имат следните начални характеристики:

- β - ядро в позиция l_3 с характеристики „*Количество вода за пречистване*“, „*Нефт*“, „*pH*“;
- χ - ядро в позиция l_6 с характеристика „*Механични примеси*“;
- δ - ядро в позиция l_9 с характеристики „*Нефт след (нефто) уловител*“, „*Механични примеси след (нефто) уловител*“;
- ε - ядро в позиция l_{12} с характеристики „*Нефт след първичен утаител*“, „*Механични примеси след първичен утаител*“;
- ϕ_1 - ядро в позиция l_{14} с характеристики „*Нефт след флотатор*“, „*Механични примеси след флотатор*“;
- γ - ядро в позиция l_{17} с характеристики „*pH изход*“, „*ХПК*“, „*Нефт изход*“, „*Механични примеси изход*“, „*PO4P*“, „NH4N“.

Обобщената мрежа е редуцирана от клас $\Sigma^{t_1,t_2,M}$.

Подробно описание на отделните преходи на обобщената мрежа може да се намери в [6*].

Обобщеномрежовият модел е създаден директно с графичния редактор на *GN IDE*. Неговата схема представлява XML файл, който описва всички преходи, позиции, ядра и данни за симулацията. Част от кода, дефиниращ прехода Z_1, е представен по-долу.

```
<gn     xmlns="http://www.clbme.bas.bg/GN"     name="WastewaterTreatmentSim"
time="256" timeStart="0" timeStep="1" language="JavaScript" root="true">
    <transitions>
        <transition id="Z1" name="Z_{1}" priority="0" startTime="0"
                    lifeTime="-1" positionX="90" positionY="60" sizeY="100">
            <inputs>
                <input ref="11">
                    <arc>
                        <point positionX="40" positionY="90"/>
                        <point positionX="90" positionY="90"/>
                    </arc>
                </input>
                <input ref="13">
                    <arc>
                        <point positionX="140" positionY="135"/>
                        <point positionX="170" positionY="135"/>
                        <point positionX="170" positionY="165"/>
                        <point positionX="75" positionY="165"/>
                        <point positionX="75" positionY="135"/>
```

```
                        <point positionX="90" positionY="135"/>
                    </arc>
                </input>
            </inputs>
            <outputs>
                <output ref="12">
                    <arc>
                        <point positionX="90" positionY="90"/>
                        <point positionX="140" positionY="90"/>
                    </arc>
                </output>
                <output ref="13">
                    <arc>
                        <point positionX="90" positionY="135"/>
                        <point positionX="140" positionY="135"/>
                    </arc>
                </output>
            </outputs>
            <predicates>
                <predicate input="11" output="12">true</predicate>
                <predicate input="11" output="13">false</predicate>
                <predicate input="13" output="12">false</predicate>
                <predicate input="13" output="13">true</predicate>
            </predicates>
        </transition>
    ...
    </transitions>
    <places>
        <place id="11" name="1_{1}" priority="0" capacity="-1"
merge="false" positionX="40" positionY="90"/>
        <place id="12" name="1_{2}" priority="0" capacity="-1"
char="add_Q3andPh_char" merge="false" positionX="140" positionY="90"/>
        <place id="13" name="1_{3}" priority="0" capacity="-1"
char="add_Q3andPh_history" merge="false" positionX="140" positionY="135"/>
        ...
    </places>
    <tokens>
        <generator type="conditional" predicate="hasWastewater" period="1"
id="alpha1" name="alpha_{1}" priority="0" host="11" entering="0" leaving="-
1"></generator>
        <token id="beta" name="q3_token" priority="0" host="13"
entering="0" leaving="-1"></token>
        ...
    </tokens>
    <functions>
        ...
    </functions>
</gn>
```

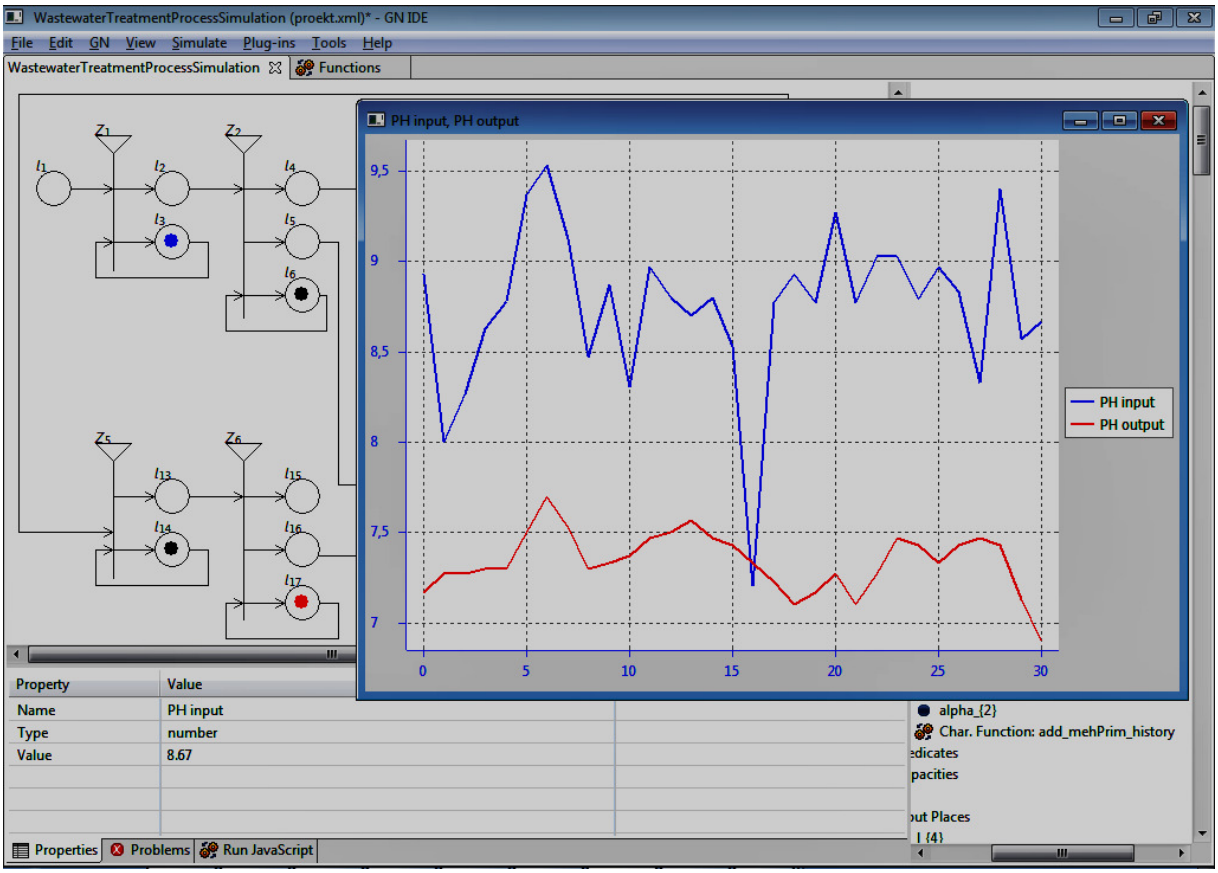
От показания код се вижда, че мрежата има времева стъпка 1, преходите са дефинирани с продължителност на активното състояние безкрайност, което се задава със стойност “-1”. Причината за това е, че редуциращите оператори не изменят моделите и там стоят стойностите, зададени при дефинирането на мрежата.

Последователно са описани входните и изходните позиции, като ясно се вижда ядрото β с начална позиция (хост) l_3. В позиция l_1 стои условен генератор на ядра. В този случай се използва генератор, заради изискването да постъпва ядро в обобщената мрежа в момента, в който постъпи отпадна вода за пречистване. Това от една страна

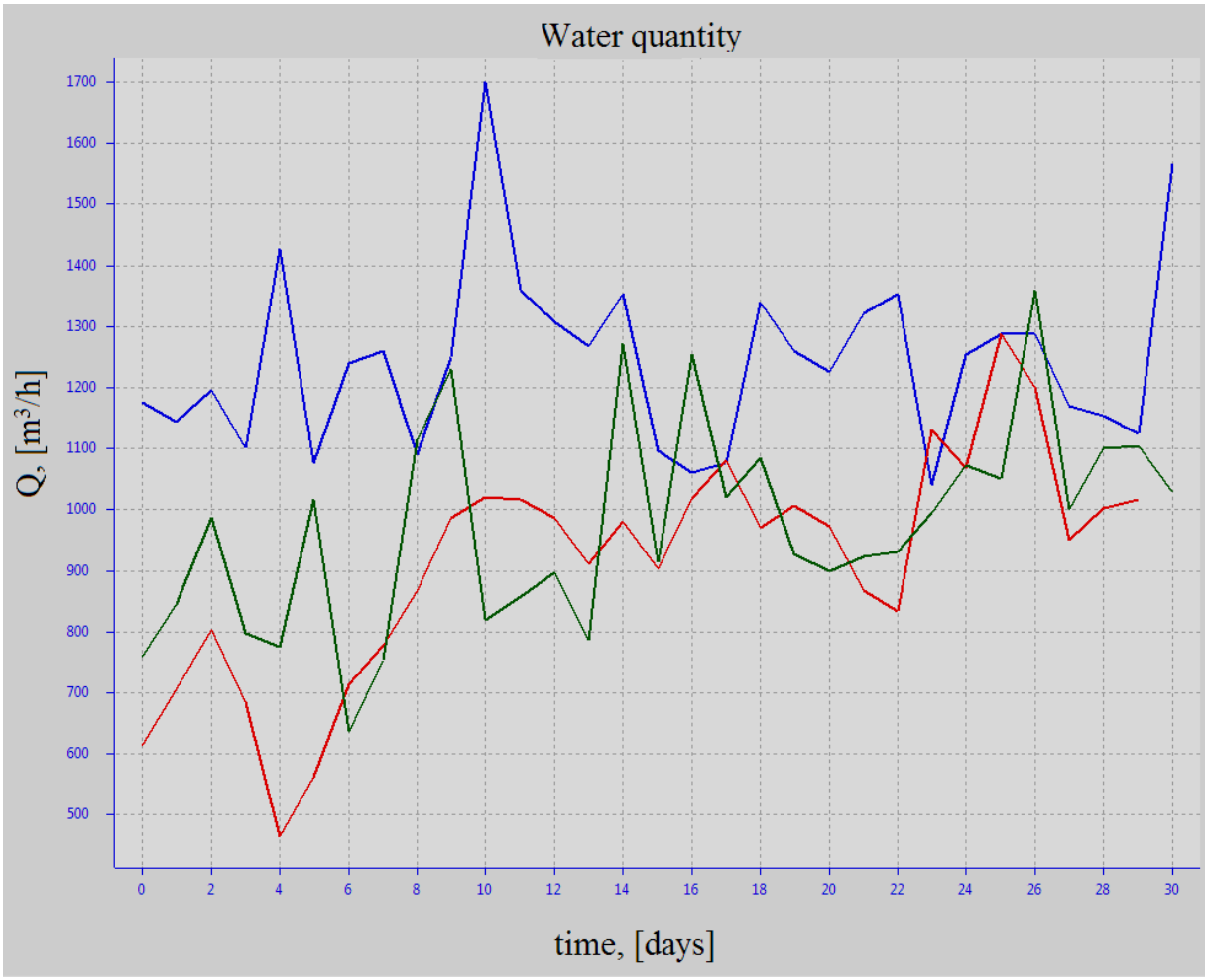
позволява процесът да протича в реално време, от друга – отделя логиката на мрежата от тази на данните, като позволява започването на процеса на пречистване да зависи единствено от дефинирания в генератора предикат.

Освен реалната симулация на процеса, *GN IDE* позволява оцветяване на ядрата в различни цветове и извеждане на графики на получените в характеристиките резултати. За обобщената мрежа за пречистване на водата са направени множество графики на данните от характеристиките на ядрата, слушатели на събития (β, χ, δ, ε, ϕ_1, γ), които запазват данните за характеристиките на водата след приключване на различните етапи.

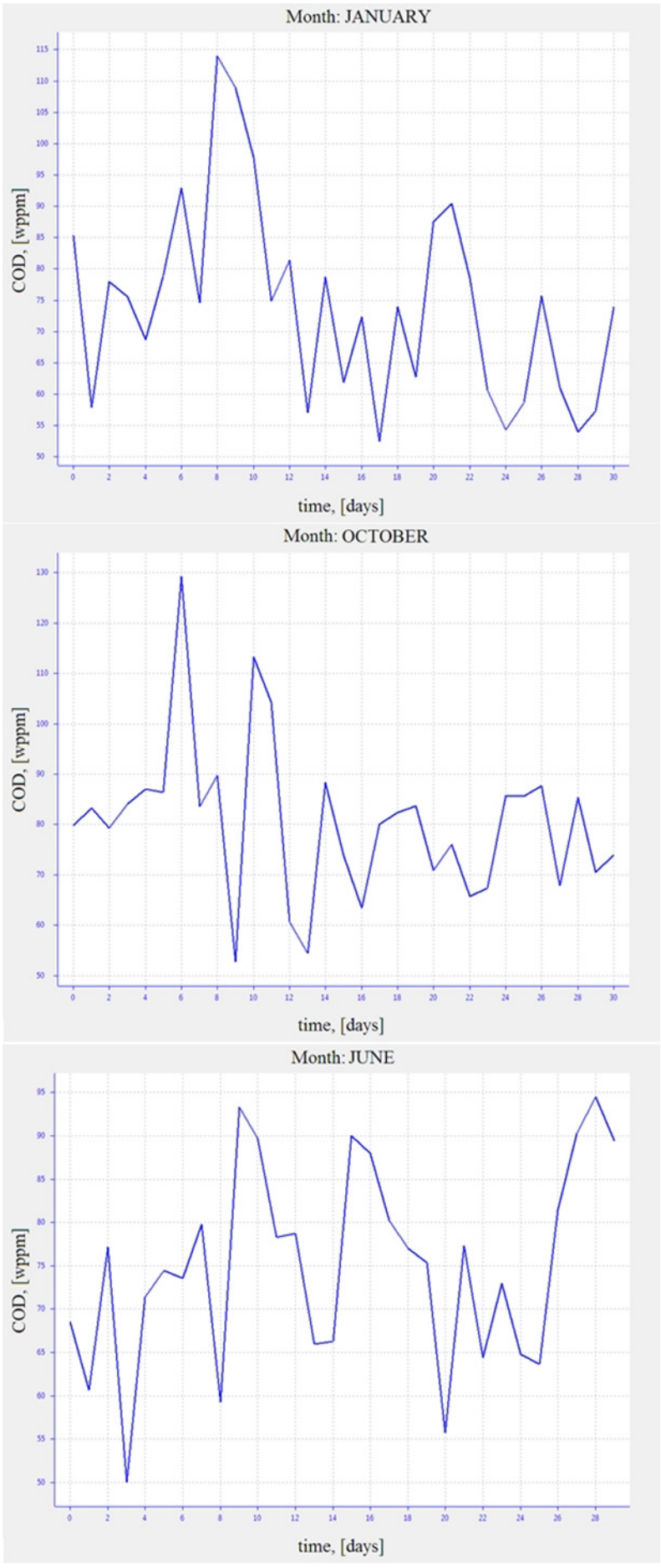
За направата на графиките са използвани различни тестови сценарии като месечно сравнение, данни на годишна база, сравнение на едни и същи характеристики на водата на различните етапи от пречистването и други. Част от графиките са показани на Фиг. 14, Фиг. 15 и Фиг. 16.



Фиг. 14 Сравнение на количеството pH във водата, при постъпването ѝ за пречистване (ядро $\boldsymbol{\beta}$ в позиция $\boldsymbol{l_3}$), и в пречистената вода (след процеса на пречистване в биобасейн с въглеродна асимилация, ядро γ в позиция l_{17}). Графиката визуализира сравнението за месец януари.



Фиг. 15 Сравнение на количеството отпадни води постъпили за пречистване, за месеците Януари, Юни и Октомври.



Фиг. 16 ХПК на пречистената вода, за месеците Януари, Юни и Октомври.

4.2. Защита на програмно осигуряване чрез използване на обобщени мрежи

За първи път дисертационният труд въвежда и нова област на приложение на обобщените мрежи, а именно защита на програмното осигуряване.

Атаки като обратно инженерство (*reverse engineering*), *modification* (чрез софтуер устойчив на разрушаване), *program-based attacks* и *break once run everywhere* могат да се възползват от недостатъците на недобре защитените софтуерни системи и да причинят големи загуби за икономиката. Това мотивира учените да разработват формални подходи за защита на софтуер, както и хардуерни и софтуерни инструменти за защита. Това мотивира също и големите софтуерни компании да влагат огромни средства в разработването на методи и техники за анализ на заплахите, на стандарти, метрики и нови подходи и механизми за защита на софтуера.

Изследванията, представени в точка 4.2 са свързани с атаките от тип *reverse engineering* [54].

Описаният по-долу метод може да се причисли към защитите, реализиращи втория подход. При него усложняването на търсенето на мястото, където се осъществява преход към защитавания софтуер, се осъществява, като се изгражда обвивка, която пакетира защитавания софтуер. Обвивката е реализирана чрез обобщени мрежи. Направеният избор е подходящ, тъй като обобщените мрежи са изключително мощен инструмент за моделиране на асинхронни процеси с динамично генериране на ядра с константни, динамични или зависещи от времето приоритети [5*], [6*] и характеристични функции. Проверката за валидност на ключа не се осъществява от един единствен преход, а е резултат от постъпково изпълнение на множество преходи на обобщената мрежа, реализиращи пълно обхождане на цифрите на въведения код.

4.2.1. Описание на подхода за защита на програмното осигуряване чрез средствата на обобщените мрежи

Защитата на софтуера, обект на направеното изследване, се осъществява от обобщената мрежа илюстрирана на Фиг. 17.

Чрез този обобщеномрежов модел ще се разпознава дали въведен от потребителя регистрационен (оторизационен) номер е правилен. Регистрационният номер е представен чрез символен низ с дължина k, където k е естествено число ($k \in \mathbb{N}$), и е съставен от десетични цифри. Стойността на k се определя от необходимата степен на безопасност на софтуера.

В следващото изложение правилния регистрационен номер ще означаваме с редицата $a_0 a_1 \dots a_{k-1}$, а съответно неговата стойност в десетична бройна система има следния вид: $R_k = a_0 10^{k-1} + a_1 10^{k-2} + \dots + a_{k-2} 10 + a_{k-1}$. Въведеният от потребителя регистрационен номер, който следва да бъде проверяван за коректност, ще означаваме с редицата $b_0 b_1 \dots b_{n-1}$. За да не усложняваме обобщената мрежа от Фиг. 17, приемаме, че n е равно на k. В противен случай считаме, че регистрационният номер $b_0 b_1 \dots b_{n-1}$ не е коректен.

Определянето на правилния регистрационен номер, е свързано с дефинирането на матрица M. В дисертационния труд е предложен метод за запълване на матрицата M. Важно е да се отбележи, че за дефинирането на M може да се използват и други алгоритми с произволна сложност, като единственото ограничение е да се гарантира, че *единствено* за правилния регистрационен номер $a_0 a_1 \dots a_{k-1}$ сумата от елементите на редицата $M[0][a_0], M[1][a_1], \dots, M[k-1][a_{k-1}]$ е равна на R_k.

Тъй като правилният регистрационен номер не трябва да се вижда в обобщената мрежа, която ще използваме за защита, той ще се добавя като характеристика, на всяко постъпващо в мрежата ядро със стойност $h(R_n)$, където h е целочислена биективна функция, подходящо избрана от създателя на обобщената мрежа за защита. С R'_k ще означаваме $h(R_k)$. Освен това на всяка стъпка от изпълнението на обобщеномрежовия модел, $h(R_k)$ ще се променя, като над него ще се прилага и целочислената биективна функция *f*. В изложението по-долу чрез f^k ще означаваме k-кратното последователно прилагане на *f*.

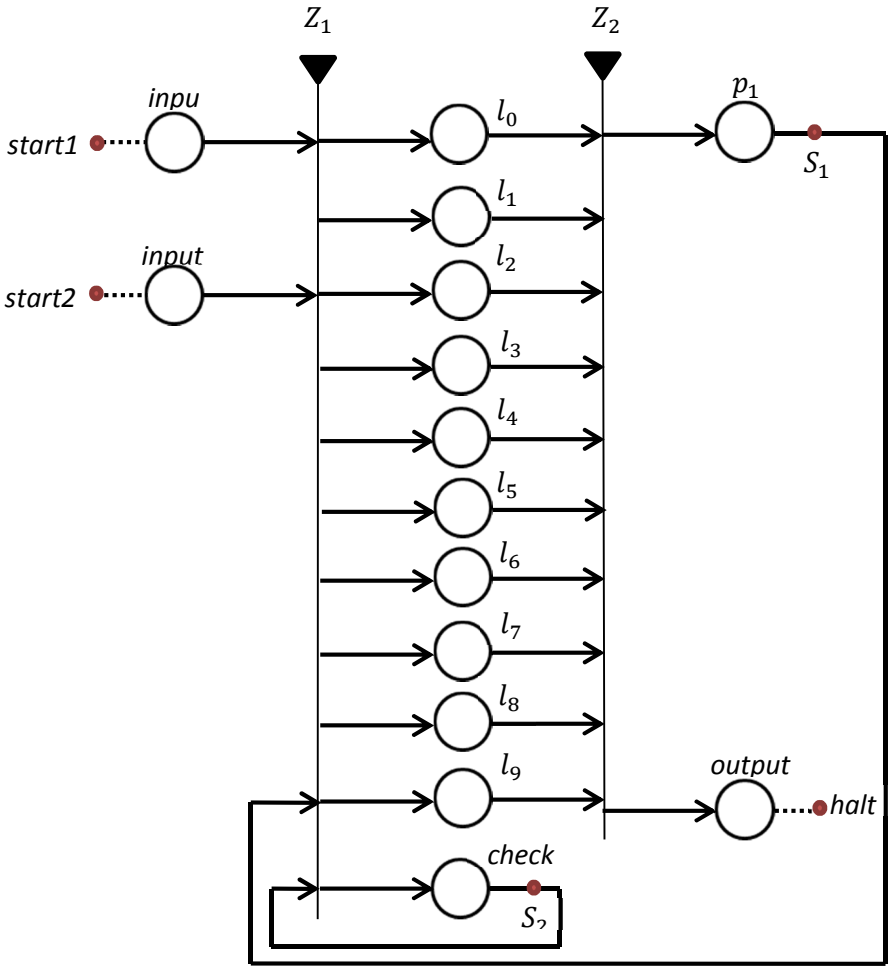
4.2.2. Дефиниция на обобщената мрежа за защита

Инициализация на входа

В позиция $input$ на обобщената мрежа от Фиг. 17 се генерира ядро $alpha$ с множество от характеристики $(password, checkIndex, modCorrectPassword)$ с начални стойности съответно:

- въведеният от потребителя регистрационен номер $- b_0 b_1 \dots b_{k-1}$.
- поредният номер на обработвания символ от регистрационния номер, първоначално равен на 0 (регистрационният номер съдържа поне един символ).
- $h(R_k)$ - стойност на правилния регистрационен номер в десетична бройна система, с приложена върху него целочислена биективна функция h.

Регистрационният номер $b_0 b_1 \dots b_{k-1}$ може да бъде правилният, както и да е напълно случаен (неправилен).



Фиг. 17 Обобщена мрежа, реализираща защита на софтуера

В позиция $input2$ се генерира ядро $gamma$ с две характеристики (*checkSum, checkIndex*) с начални стойности (0, 0). Стойността на *checkSum* е неотрицателно цяло число. *CheckSum* е инициализирана с 0 и на всяка стъпка към стойността ѝ се добавят стойностите, записани в $M[0][b_0], M[1][b_1], \dots, M[k-1][b_{k-1}]$ на матрицата M. Ако въведеният от *input* регистрационен номер $b_0 b_1 \dots b_{k-1}$ е правилният, след сканирането на символите му стойността на *checkSum* ще е равна на R_k. Тъй като R_k не е известна на обобщената мрежа за защита (известна е стойността на *modCorrectPassword*, която след сканирането на входния низ е равна на $f^k(h(R_n))$), проверката дали регистрационният номер $b_0 b_1 \dots b_{k-1}$ е коректният ще се осъществи, като се провери дали след сканирането на входния низ стойността на *modCorrectPassword* е равна на $f^k(h(checkSum))$. Ако $b_0 b_1 \dots b_{k-1}$ не е правилният регистрационен номер, след сканирането на символите му в *checkSum* ще се натрупа стойност, различна от R_k (т.е. стойността на *modCorrectPassword* ще е различна от $f^k(h(checkSum))$) и обобщената мрежа няма да го разпознае за правилен.

Дефиниция на преходите Z_1 и Z_2 на обобщената мрежа

Обобщената мрежа за защита от Фиг. 17 има два прехода Z_1 и Z_2.

За двата прехода на мрежата, времевите компоненти t_1 (текущ времеви момент, при който преходът се активира), t_2 (текущата продължителност на активното състояние на прехода) ще имат стойност съответно 0 и –1, т.е. преходът се активира със старта на функционирането на мрежата и има безкрайна продължителност на активното състояние. Индексираните матрици на капацитетите на дъгите M_1 и M_2, ще бъдат запълнени със стойности „безкрайност“, тъй като не трябва да се налага ограничение на количеството ядра, преминаващи по дъгите.

Z_1 променя стойностите на характеристиките *checkIndex*, *checkSum* и *modCorrectPassword* на ядрата *alpha* и *gamma*. Преходът Z_2 реализира проверка дали символният низ $b_0 b_1 \dots b_{k-1}$ е изчерпан или не, и променя характеристиката *modCorrectPW*$_i$ на ядрото *alpha*$_i$, което преминава в позиция p_1. В случай, че низът $b_0 b_1 \dots b_{k-1}$ е изчерпан, изпълнението на обобщената мрежа завършва с разпознаване на регистрационния номер, ако $f^k(h(checkSum))$ е равно на $modCorrectPassword$ или с неразпознаване на регистрационния номер, ако $f^k(h(checkSum))$ не е равно на $modCorrectPassword$. И в двата случая ядрото $alpha_0$ преминава в позиция $output$ и получава множество от характеристики ($password, checkIndex, modCorrectPassword, result$) със стойност ($b_0 b_1 b_2 \dots b_{k-1}, k, modCorrectPW_0, modCorrectPassword == f^k(h(checksum))$). Характеристиката $result$ на ядрото в $output$ определя дали регистрационният номер е разпознат или не.

4.2.3. Надеждност на подхода

Надеждността на подхода се определя от неговата коректност и от това доколко дълго той може да осигури забавяне на процеса софтуерът да стане свободно достъпен.

На този етап не са правени практически експерименти доколко успешно широкоизвестните методи за разрушаване на защитата ще работят в случая на предложения метод. Все пак ако на кодоразбивача не е известен описаният метод за защита на програмния код, единственият начин за намиране на кода за достъп е пълно изчерпване на всички възможни стойности за регистрационния номер. При

регистрационен номер с дължина k, броят на различните регистрационни номера, които трябва да бъдат проверени, е равен на 10^k.

4.2.4. Верификация на обобщената мрежа за защита

За да верифицираме обобщената мрежа на (Фиг. 17) ще използваме метода на индуктивните твърдения на Флойд [73]. Ще дефинираме входния, работния и изходния вектор по следния начин:

- $\bar{x} = (k, b_0 b_1 \dots b_{k-1}, a_0 a_1 \dots a_{k-1}, M, R_n, f, h)$;
- $\bar{y} = (password, checkIndex, modCorrectPassword, checkSum, symbol_i, modCorrectPW_i)$;
- $\bar{z} = (result, checkIndex, modCorrectPassword, checkSum)$.

Ще отбележим, че в случая променливите $checkIndex, modCorrectPassword$ и $checkSum$ се използват както в $\bar{y}$, така и в $\bar{z}$. Направили сме го, за да не усложняваме излишно означенията.

С всяка позиция на обобщената мрежа (Фиг. 17) ще свържем оператори за едновременно (паралелно) присвояване, описващи как се променят характеристиките на ядрата при постъпване в съответните позиции.

Инициализацията на ядрото $alpha$ в позиция $input$ има следния вид:

$(password, checkIndex, modCorrectPassword) = \big(b_0 b_1 \dots b_{k-1}, 0, h(R_n)\big)$, където

$password[\,j\,] == b[\,j\,]$ за $0 \le j \le k-1$

Инициализацията на ядрото $gamma$ в позиция $input2$ има следния вид:

$(checkSum, checkIndex) = (0,0)$

Промяната на характеристиките на ядрото $gamma$ в позиция $check$ има следния вид:

$(checkSum, checkIndex)$

$= (checkSum + M[checkIndex][\,password[checkIndex]\,], checkIndex + 1)$

Промяната на характеристиките на ядрата $alpha_i$, в позиции l_i ($i = 0, 1, \dots, 9$) има следния вид:

$(password, checkIndex, modCorrectPassword, symbol_i, modCorrectPW_i) =$

$(password, checkIndex + 1, f(modCorrectPassword) + i, (\,password[\,checkIndex\,] + i\,)\ mod\ 10\,, f(\,modCorrectPassword\,) + i)$

Промяната на характеристиките на ядрото $alpha$ в позиция P_1 има следния вид:

$(password, checkIndex, modCorrectPassword) =$

$(password, checkIndex, modCorrectPW_i\ - i)$

Промяната на характеристиката на ядрото $alpha_0$ в позиция $output$ има следния вид:

$(password, checkIndex, modCorrectPassword, result) =$

$(password, checkIndex, modCorrectPW_0, modCorrectPW_0 == f^k(h(checkSum)))$

Входният предикат $\varphi(\bar{x})$ трябва да зададе условието, относно което ще верифицираме обобщената мрежа от Фиг. 17. В този случай, той е свързан с входните позиции $input$ и $input2$ и ядрата, които влизат в тях при опит за вход в системата.

За входен предикат избираме:

$\varphi(\bar{x}) = k \ge 1\ \wedge\ (0 \le b_i \le 9$ за $i = 0, \dots, k-1\,)\ \wedge\ (\,$за $\forall\ b_0 b_1 \dots b_{k-1} \ne a_0 a_1 \dots a_{k-1}$

е в сила $\mathrm{M}[0][b_0] + \mathrm{M}[1][b_1] + \cdots + \mathrm{M}[k-1][b_{k-1}] \neq R_n \wedge$
$\mathrm{M}[0][a_0] + \mathrm{M}[1][a_1] + \cdots + \mathrm{M}[k-1][a_{k-1}] == R_n$),

където $b_0 b_1 \ldots b_{k-1}$ е въведената стойност за *password,* а $a_0 a_1 \ldots a_{k-1}$ е коректната парола.

Изходният предикат $\psi(\bar{x}, \bar{z})$ задава условието, което трябва да бъде в сила, когато ядрото $alpha_0$ попадне в позиция $output$ на обобщената мрежа, т.е. когато завърши изпълнението на обобщената мрежа.

В случая, за изходен предикат избираме:

$$\psi(\bar{x}, \bar{z}) = (\ checkIndex == k\ ) \wedge$$
$$(result == \left(modCorrectPassword == f^k\left(h(checkSum)\right)\right)),$$

където $result$ е булева променлива, която указва дали въведената от потребителя парола е вярна или не. Условието $modCorrectPassword == f^k\left(h(checkSum)\right)$ в позиция *output* има вида:

$$f^k\left(h(R_n)\right) == f^k\left(h\left(\sum_{l=0}^{k-1} M[l][b_l]\right)\right)$$

Тъй като функциите *f* и *h* са биекции, ако условието е *true,* редицата $b_0 b_1 \ldots b_{k-1}$ съвпада с редицата $a_0 a_1 \ldots a_{k-1}$.

Освен входният и изходният предикати, трябва да се дефинират и още два предиката, т.нар. индуктивни твърдения, в срезовете S_1 и S_2 на примките на обобщената мрежа (Фиг. 17). За индуктивни твърдения избираме:

- P_{S_1} в срез S_1:

$$(\ 1 \leq checkIndex < k\ ) \wedge \left(\ modCorrectPassword == f^{checkIndex}\left(h(R_n)\right)\right)$$

- P_{S_2} в срез S_2:

$$(\ 1 \leq checkIndex \leq k\ ) \wedge \left(checkSum == \sum_{l=0}^{checkIndex-1} M[\ l\ ][\ b_l\ ]\right)$$

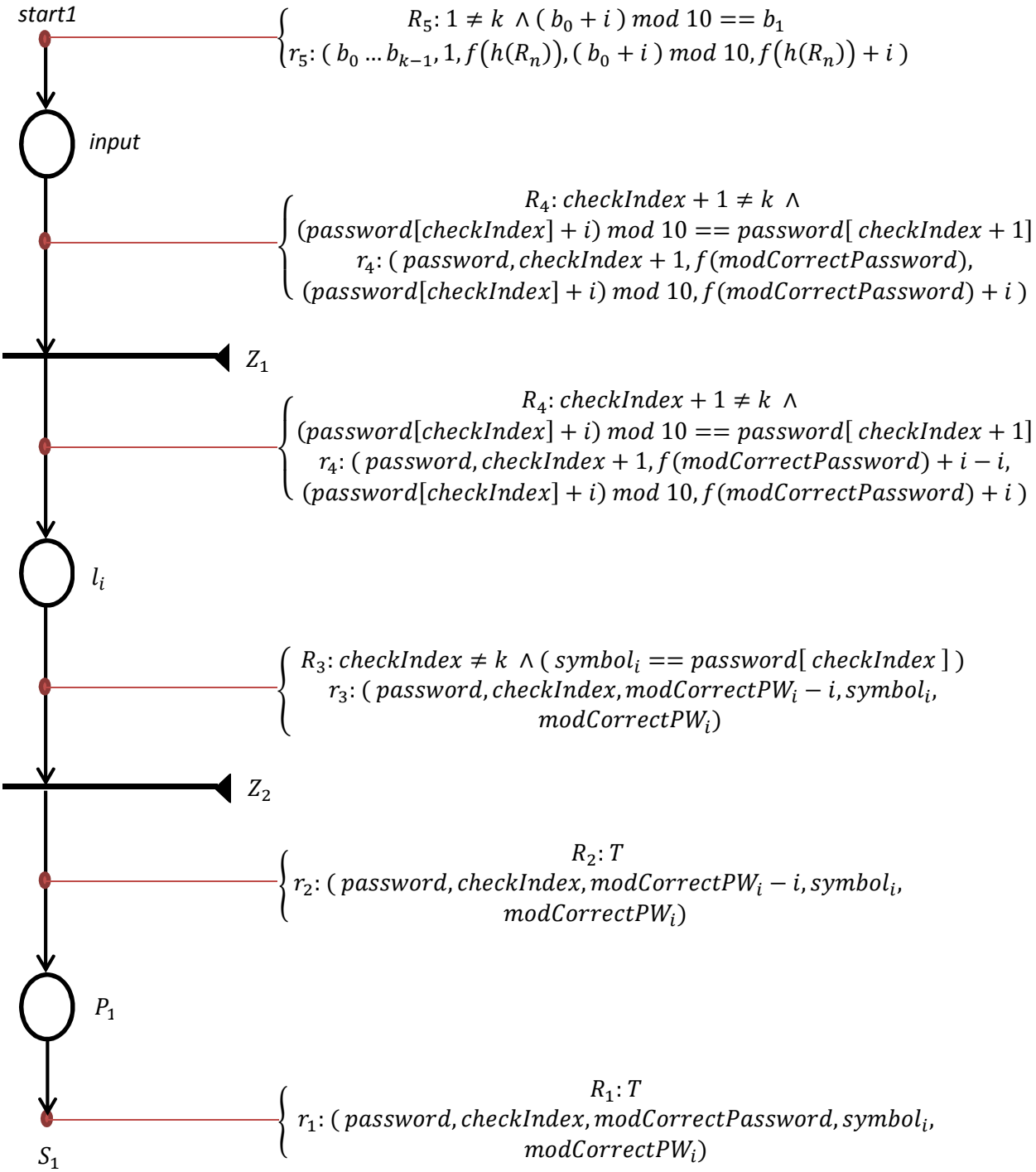
За да верифицираме обобщената мрежа, ще докажем, че тя е частично коректна относно φ и ψ и че завършва изпълнението си при φ.

Частична коректност

За да докажем, че обобщената мрежа за защита е частично коректна относно входния предикат $\varphi(\bar{x})$ и изходния предикат $\psi(\bar{x}, \bar{z})$ срязваме примките на обобщената мрежа в срезовете S_1 и S_2. Обобщената мрежа се покрива от следните пътища:

1) $start1 \rightarrow l_i \rightarrow S_1$ за $i = 0, 1, \ldots, 9$
2) $start2 \rightarrow S_2$
3) $S_1 \rightarrow l_i \rightarrow S_1$ за $i = 0, 1, \ldots, 9$
4) $S_2 \rightarrow S_2$
5) $start1 \rightarrow l_0 \rightarrow halt$
6) $S_1 \rightarrow l_0 \rightarrow halt$

Верификация на пътища $start1 \rightarrow\ l_i \rightarrow S_1$ за $i\ = 0, 1, \dots, 9$



Ще докажем верифициращото условие за тези пътища:

$$\varphi(\bar{x}) \wedge R_5 \Rightarrow (\ 1 \leq 1 < k\ )\ \wedge \Big(\ f\big(h(R_n)\big) ==\ f^1\big(h(R_n)\big)\Big)$$

От $\varphi(\bar{x})$ знаем, че $k \geq 1$, от R_5 следва, че $1 \neq k$ и следователно $1 < k$ е в сила. Следователно верифициращото условие за пътищата $start1 \rightarrow\ l_i \rightarrow S_1$ е изпълнено. Верификацията на останалите пътища е описана в дисертационния труд.

Завършване на изпълнението

Стъпка 1

Срязваме примките на обобщената мрежа в точките S_1 и S_2 и избираме следните твърдения за срезовете S_1 и S_2:

$$q_{S_1}: 1 \leq checkIndex < k,$$

$$q_{S_2}: 1 \leq checkIndex \leq k$$

Тъй като q_{S_1} и q_{S_2} се съдържат съответно в индуктивните твърдения, които са в сила в S_1 и S_2 и за които доказахме условията за частична коректност, q_{S_1} и q_{S_2} са добри твърдения.

Стъпка 2

Избираме $(W, \prec) \equiv (\mathbb{N}, <)$, $\mathbb{N}$ е множеството на естествените числа, които са положителни (≥ 1). С S_1 и S_2 свързваме съответно

$u_{S_1}(\bar{x}, \bar{y}) = k - checkIndex,$

$$u_{S_2}(\bar{x}, \bar{y}) = k + 1 - checkIndex$$

Трябва да се докажат:

а) $q_{S_1} \Rightarrow (u_{S_1} \in \mathbb{N})$, т.е.

$$1 \leq checkIndex < k \Rightarrow k - checkIndex \in \mathbb{N}$$

Тъй като $k - checkIndex \in \mathbb{N}$ означава, че $(k - checkIndex) \geq 1$, трябва да се докаже, че е в сила импликацията

$$1 \leq checkIndex < k \Rightarrow (k - checkIndex) \geq 1$$

Последното е вярно, защото $k - checkIndex > 0$ е еквивалентно на $k - checkIndex \geq 1$

б) $q_{S_2} \Rightarrow (u_{S_2} \in \mathbb{N})$, т.е.

$1 \leq checkIndex \leq k \Rightarrow (k + 1 - checkIndex) \geq 1,$

което очевидно е в сила, защото $(k + 1 - checkIndex) \geq 1$ е еквивалентно на $checkIndex \leq k$, което се съдържа в лявата част на импликацията.

Стъпка 3

а) За пътищата $S_1 \rightarrow l_i \rightarrow S_1$ $(i = 0, 1, \ldots, 9)$

Условието

$q_{S_1}(\bar{x}, \bar{y}) \wedge checkIndex + 1 \neq k \wedge ((password[checkIndex] + i) \bmod 10 == password[checkIndex + 1]) \Rightarrow$

$[u_{S_1}(\bar{x}, password, checkIndex, modCorrectPassword, symbol_i, modCorrectPassword_i) >$

$u_{S_1}(\bar{x}, password, checkIndex + i, f(modCorrectPassword),$

$(password[checkIndex] + 1) \bmod 10, f(modCorrectPassword) + i))$

има вида:

$(1 \leq checkIndex < k) \wedge (checkIndex + 1 \neq k) \wedge$

$((password[checkIndex] + i) \bmod 10 == password[checkIndex + 1]) \Rightarrow$

$$k - checkIndex > k - (checkIndex + 1),$$

което очевидно е в сила.

б) За път $S_2 \rightarrow S_2$

Условието

$q_{S_2}(\bar{x}, \bar{y}) \wedge 1 \leq checkIndex < k \wedge (\,(\,password[checkIndex] + i\,)\; mod\; 10 ==$
$password[\,checkIndex + 1] \Rightarrow [u_{S_2}(\bar{x}, checkSum, checkIndex) > u_{S_2}(\bar{x}, checkSum +$
$M[checkIndex][password[checkIndex]], checkIndex + 1)]$
има вида:
$(1 \leq checkIndex \leq k) \wedge (1 \leq checkIndex < k) \wedge$
$(\,(\,password[checkIndex] + i\,)\; mod\; 10 == password[\,checkIndex + 1] \Rightarrow$

$$k + 1 - checkIndex > k + 1 - (checkIndex + 1),$$

което също е в сила.

4.3. Обобщеномрежов модел на процес на измерване на плоскопаралелни краищни мерки за дължина

По дефиниция ненаграфена (краищна) мярка за дължина се нарича еднозначна мярка, дължината на която се определя от най-късото разстояние между противоположните и измервателни повърхнини [7].

Плоскопаралелните краищни мерки за дължина (ППКМД) са най-често използваните ненаграфени мерки за дължина. Най-разпространените ППКМД имат форма на правоъгълен паралелепипед, чиято височина се приема за дължина на мярката.

ППКМД се използват като еталонни мерки за дължина, за да предават единицата за дължина от еталонните до работните мерки и измервателните уреди и като работни мерки за дължина, за да се използват като мерки за настройка на измервателните средства. В областта на геометричните измервания плоскопаралелните краищни мерки за дължина заемат важно място във веригата за метрологична проследимост на основната единица за дължина – метър. Използват се за съхранение и предаване на единицата за дължина, за калибриране и проверка на различни мерки, калибри и др.

В тази част на дисертационния труд, процесът на измерване на плоскопаралелни краищни мерки за дължина (ППКМД), обект на направеното изследване, се осъществява чрез обобщената мрежа, представена на Фиг.19. Идеята на реализирания обобщеномрежов модел е да покаже всички етапи по измерването на плоскопаралелните краищни мерки за дължина, като за първи път в него се отчита и наличието на субективен фактор, свързан с нивото на компетенциите на субекта и връзката между получените резултати и него [7*].

Обобщеномрежовият модел на измервателен процес на ППКМД е съобразен с дейностите, извършвани в Централна лаборатория за измервателна техника към Института по отбрана „Професор Цветан Лазаров“, като е направена верификация на работата му, чрез извършване на реално калибриране на краищна мярка за дължина с размер 50 mm. Описаните процеси са на базата на реално извършени експерименти.

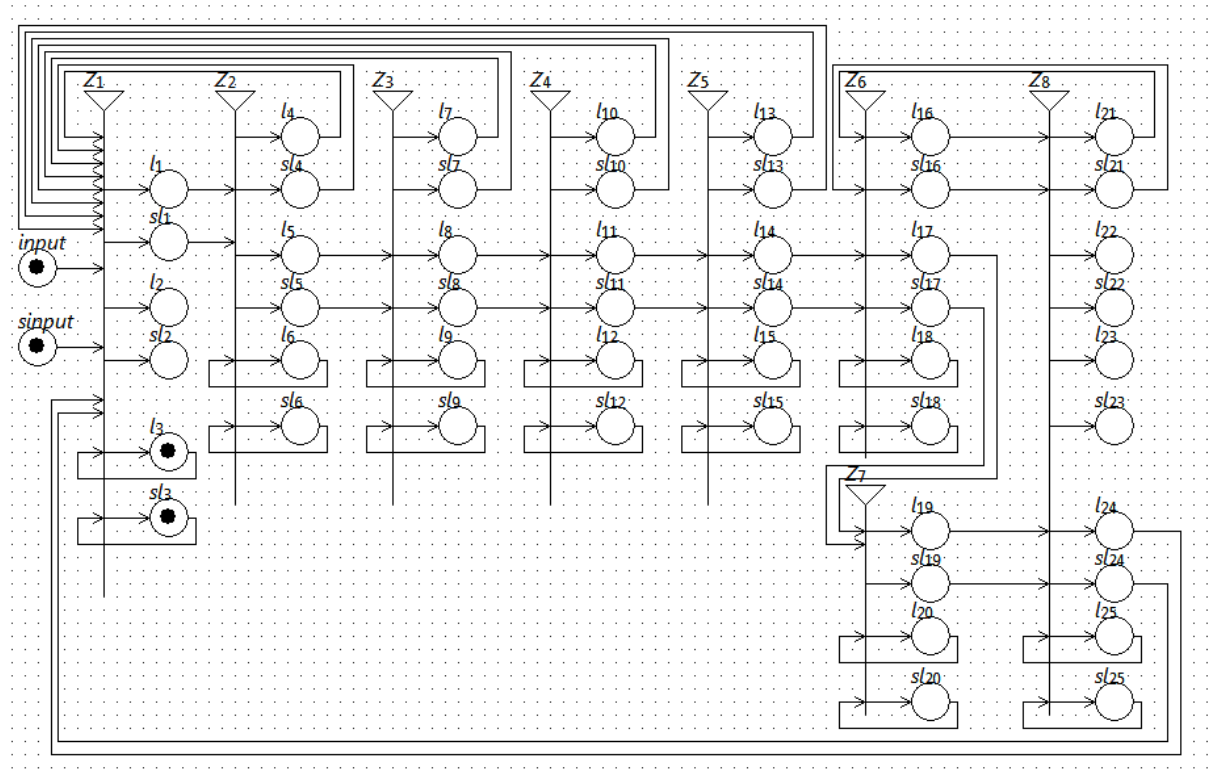
Всяка плоскопаралелна краищна мярка за дължина, подлежаща на измерване в обобщената мрежа, ще се представя чрез ядро, означено с „*GBObject*“, а субектът участващ в измервателния процес с ядро, означено с „*subject*“. Процесът на измерване се разделя на няколко етапа (подпроцеса):

- анализ на началното състояние;
- обработка и подготовка на постъпилата за калибриране ППКМД;
- темпериране на ППКМД и участващата в измерването апаратура;
- подготовка и настройка на измервателната апаратура;
- измерване на параметрите на обекта;
- обработка на данните за получаване на оценка на изходната величина;

- обработка на данните за получаване на разширена неопределеност на измерването;
- анализ на резултата.

4.3.1. Дефиниция на обобщена мрежа на процеса на измерване на ППКМД

Разработеният обобщеномрежови модел симулира описаната по-горе схема на измерване на ППКМД, като отчита и субективния фактор. Моделът е показан на Фиг. 19.



Фиг. 19 Обобщеномрежови модел на измерване на ППКМД

Инициализация на входа

В позиция $input$ на обобщената мрежа от Фиг. 19 се генерира ядро $GBObject$ с множество от характеристики ($state, serialNumber, accuracy, nominal, material, status$) с начални стойности:

- $state$ (текущото състояние на измервателния процес на ППКМД) - „*Плоскопаралелна краищна мярка, постъпила за измерване*“,
- $serialNumber$ – сериен (фабричен) номер на ППКМД,
- $accuracy$ – клас на точност на комплект ППКМД,
- $nominal$ – номинална стойност,
- $material$ – материал,
- $status$ – външно състояние,

В позиция $sinput$ на обобщената мрежа се генерира ядро $subject$ с множество от характеристики:
($state, name, experianceDegree, objectStateDegree, objectWorkingConditionsDegree, subjectAreaKnowledgeDegree, personalQualitiesDegree, resourcesDegree, regulationsDegree, motivationDegree, complexDegree$) с начални стойности съответно:

- $state$ – „*Субект, приел да извърши измерването на ППКМД с фабричен номер XXXXX*“.
- $name$ – име на субекта, извършващ измерването.
- $experienceDegree$ – степен на продължителност на работа на субекта в областта на измерванията и по-конкретно, за определения вид измерване.
- $objectStateDegree$ – степен на познаване на информацията за състоянието на обекта (метрологичните характеристики на конкретния обект през годините, както и данни от анализ на началното състояние).
- $objectWorkingConditionsDegree$ – степен на познаване на необходимата информация по отношение на калибрирания обект.
- $subjectAreaKnowledgeDegree$ – степен на притежаване на необходимите знания, образование и квалификация**.**
- $personalQualitiesDegree$ – степен на притежание на личностни (професионални и нравствени) качества.
- $resourcesDegree$ – степен на познаване на необходимото еталонно и спомагателно оборудване, инструментариум и др.
- $regulationsDegree$ – степен на познаване и прилагане на използваните нормативни документи в ежедневната си дейност.
- $motivationDegree$ – степен на мотивация, насочена към постигане на определени резултати, цели, позиции, ценности, които са причина за поведението на субекта.
- $complexDegree$ – степен на комплексните качества на субекта по отношение на това, доколко е подходящ за изпълнение на задачата „измерване на ППКМД“.

Степените са числа в затворения интервал [0,1], което позволява да се използват интуционистки размити оценки в анализа на получените резултати.

Първоначално ядрата *listenGBObject* и *listenSubject* стоят в позициите l_{26} и sl_{26}. Ядрото *listenGBObject* притежава три характеристики $expandedUncertaintyHistory$ $deviationFromAverageSizeHistory$ и $deviationFromFlatnessHistory$, а ядрото *listenSubject* е с множество от характеристики:
($experienceDegreeHistory, objectStateDegreeHistory,$
$objectWorkingConditionsDegreeHistory, subjectAreaKnowledgeDegreeHistory,$
$personalQualitiesDegreeHistory, resourcesDegreeHistory,$
$regulationsDegreeHistory, motivationDegreeHistory, complexDegreeHistory$).

Те ще стоят в тези позиции през цялото време на функциониране на обобщената мрежа и ще натрупват в своите характеристики съответно стойностите от измерванията и оценките за субекта. Тези две ядра ще бъдат използвани за натрупване на история на направените измервания.

Дефиниция на преходите на обобщената мрежа

Обобщената мрежа, симулираща процеса на измерване на ППКМД, съобразен с дейностите, извършвани в Централната лаборатория за измервателна техника към Института по отбрана, се състои от осем прехода с имена съответно $Z_1, \dots, Z_8$:

$$A = \{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7, Z_8\},$$

където:

- Z_1 описва процеса на анализ на началното състояние.
- Z_2 описва процеса на обработка и подготовка на еталонните и калибрирани ППКМД.

- Z_3 описва процеса на темпериране на ППКМД и участващата в измерването апаратура.
- Z_4 описва процеса на подготовка и настройка на измервателната апаратура.
- Z_5 описва процеса на измерване на параметрите на обекта (ППКМД).
- Z_6 описва процеса на обработка на данните за получаване на оценка на изходната величина или грешка на измерването.
- Z_7 описва процеса на обработка на данните за получаване на неопределеността на измерването.
- Z_8 описва процеса на анализ на резултата от измерване (множество от стойности на величина, преписани на измерваната величина заедно с всяка друга налична приложима информация).

По подразбиране се приема, че всяка ППКМД постъпила за измерване (ядра *GBObject*), е с по-нисък приоритет от тези, приети преди нея. Съответно всички позиции на мрежата, реализиращи изчакване на необходимите ресурси и/или фактори на заобикалящата среда, са с по-висок приоритет.

Подробно описание на отделните преходи и обобщената мрежа е направено в дисертационния труд.

4.3.2. Симулация на обобщеномрежови модел на измерване на ППКМД в GN IDE

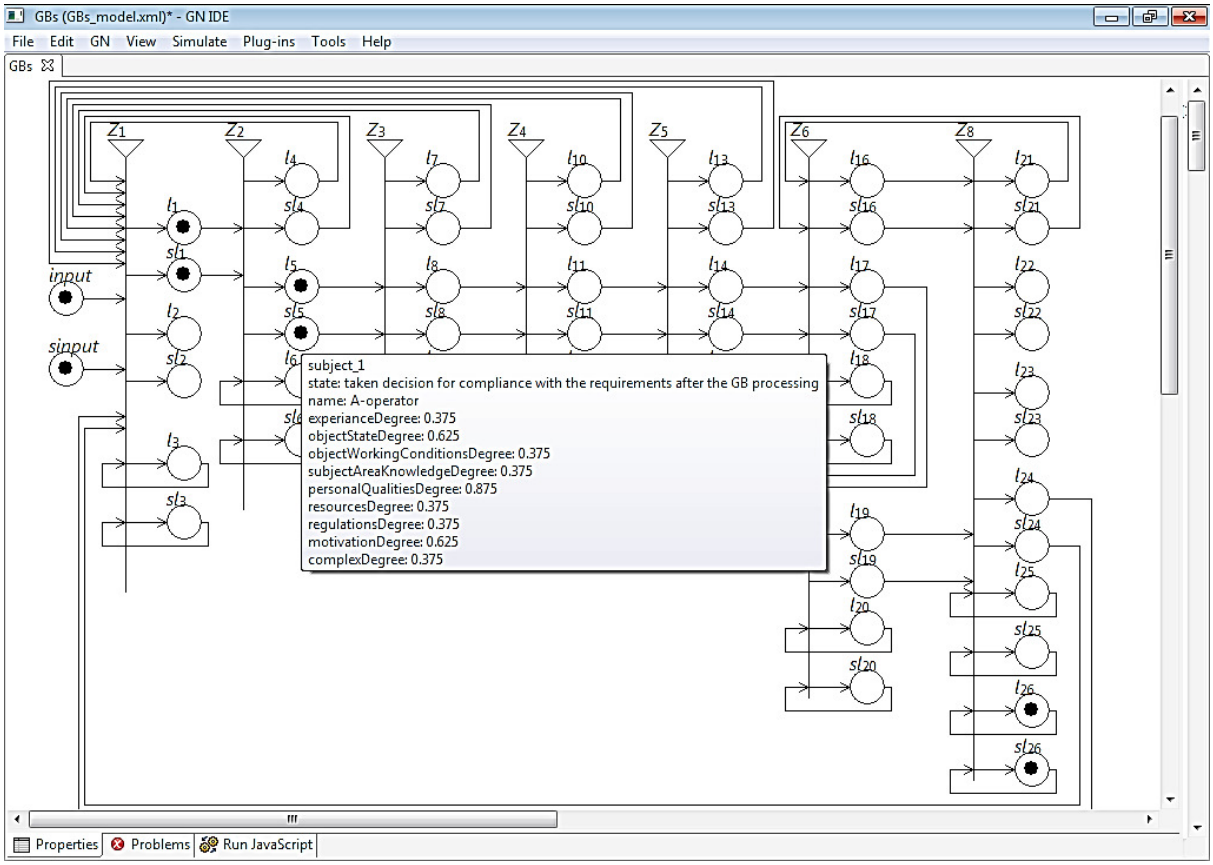
Реализацията на симулацията се осъществява чрез GN IDE.

Част от кода реализиращ обобщеномрежовия модел на измерване на ППКМД е представен в дисертацията.

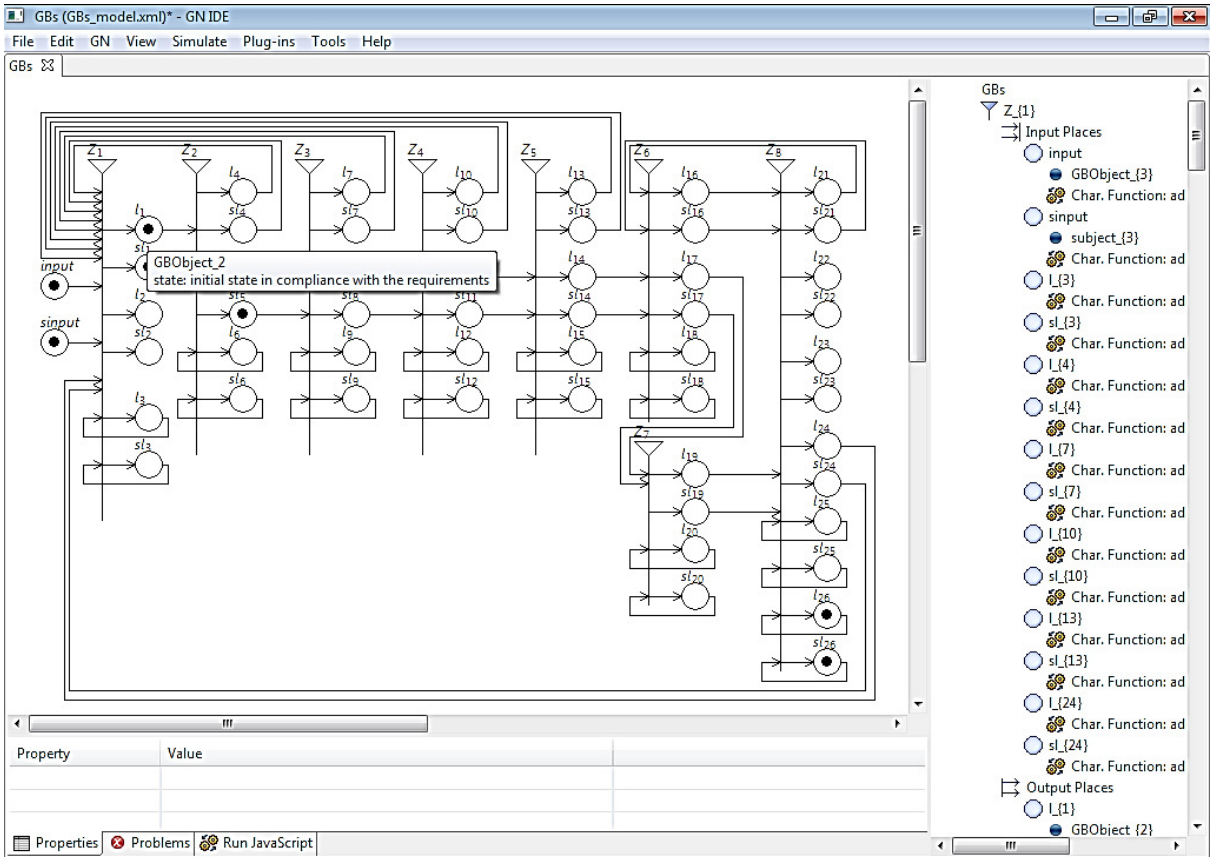
Предикатите и характеристичните функции са реализирани на езика JavaScript, като обхващат всички заложени в описанието по-горе изисквания за наличие на ресурси, фактори на заобикалящата среда и текущо състояние на обекта.

В симулацията се използват данни от реално направени тестове на 6-ма души, означени с A-оператор, B-оператор, C-оператор, D-оператор, E-оператор, F-оператор.

Движението на ядрата *subject* и *GBObject* през обобщената мрежа и промяната на техните характеристики са показани на Фиг. 20 и Фиг. 21.

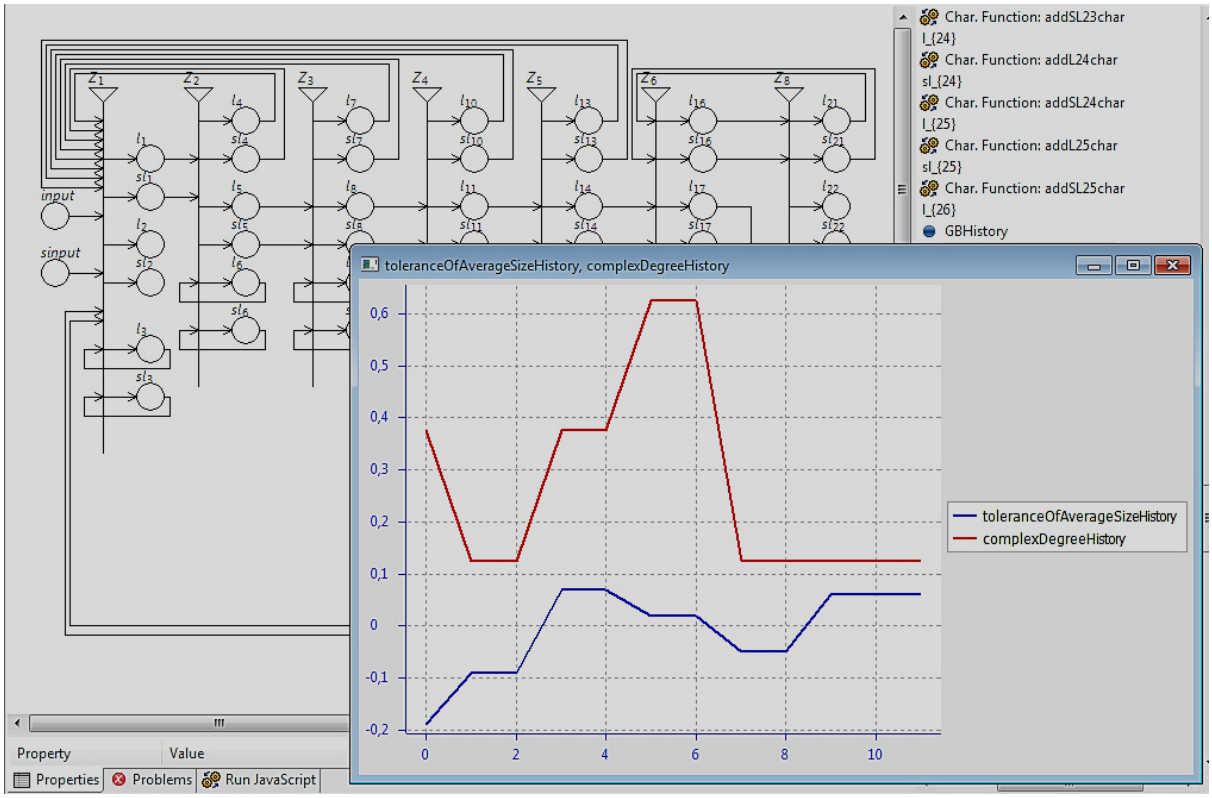


Фиг. 20 Движение на ядрото *subject* и неговите характеристики в позиция $\boldsymbol{sl_5}$



Фиг. 21 Движение на ядрото *GBObject* и стойността на неговата характеристика *state* в позиция $\boldsymbol{l_1}$

Основната цел на направената симулация е да даде връзка между оценката на субективния фактор на различните оператори (субекти) и получените от тях резултати от измерването. Връзката между комплексната оценка на субектите и отклонението от средния размер на ППКМД при направеното от тях измерване е показано на Фиг. 22



Фиг. 22 Графика на комплексната оценка на субектите и отклонението от средния размер на ППКМД

4.4. Постигнати резултати в Глава 4

В Глава 4 се решават последните две от поставените в Глава 1 задачи в дисертацията, като се показва, че съществуват нови области на приложение на обобщените мрежи и ги посочва като подходящ и надежден инструмент за моделиране, анализ и симулация на всички видове процеси с дискретни събития, които се развиват с течение на времето.

В точките 4.1 и 4.3 се описват две симулации на процес на пречистване на отпадните води и измерване на плоскопаралелни краищни мерки. За двете симулации се използват реални данни и се извеждат графики, които показват резултатите от извършените симулации.

В точка 4.2 за първи път се реализира модел, чиято цел не е симулация на процес, а се използва като нов инструмент за защита на програмното осигуряване. Създаденият модел е примерен, но дава идея как обобщените мрежи могат да бъдат използвани като част от по-големи софтуерни продукти. В точка 4.2.4 е реализирана и първата верификация на обобщена мрежа по дефинираните в част 2.5 теореми за частична коректност и завършване на изпълнението на обобщена мрежа.

Заключение

В първа глава на дисертационния труд е направен е преглед на основните публикации в теорията и програмната реализация на обобщените мрежи, като на база на тях е формирана цел и седем задачи на дисертационния труд. Реализацията на задачите се осъществява в Глава 2, Глава 3 и Глава 4 на дисертацията.

В Глава 2 се описват направените приноси в разширяването на апарата на обобщените мрежи. Предлагат се и две консервативни разширения на понятието обобщена мрежа, които разширяват възможностите за симулиране на реални процеси, при които приоритетите на ядрата могат да се променят, но не променят алгоритмите за функциониране на преход и за функциониране на обобщена мрежа. Дефинирани са и три нови понятия – частична коректност, завършване на изпълнението и тотална коректност на обобщена мрежа и са формулирани две теореми за доказване на частична коректност и завършване на изпълнението. Един от приносите в теорията на обобщените мрежи е дефинирането на нов формален метод за верификация на обобщени мрежи без времеви компонент на преходите, като се показва, че верификацията на една обобщена мрежа от този вид, относно зададената входно-изходна спецификация, се свежда до доказване на нейната тотална коректност относно тази спецификация. Предлага се и нов алгоритъм за функциониране на преход, при които се отчитат възможностите за сливане и разцепване на ядра и са формулирани и доказани теореми за запазване на поведението на оператори и релации върху редуцирани обобщени мрежи.

В Глава 3 на дисертационния труд се описват направените нововъведения в програмната реализация на обобщените мрежи. Реализират се структурни промени в интегрираната среда за разработка и симулация на обобщени мрежи *GN IDE*, като се премахва старата и неизползваема функционалност и връзките на приложението с нея. Освен това се извършва разширение на XML схемата на обобщените мрежи, като се добавя възможност за редуциране на определени компоненти от един обобщеномрежов модел. Направеното разширение запазва съвместимостта на реализираните до момента обобщеномрежови модели с новата схема и предоставя възможност за премахване на приложените редуциращи оператори и възвръщане на мрежата в нейно предишно състояние, т.е. редуциращите оператори не изменят моделите.

В последната Глава 4 се описват три обобщеномрежови модела от неизвестни до този момент области на приложение. Два от моделите са симулирани в *GN IDE*, като за симулацията са използвани реални данни. Другият модел реализира метод на защита на програмното осигуряване.

При изготвянето на дисертационния труд се формулират редица идеи за бъдещи изследвания. Вследствие на предложения метод за верификация на обобщени мрежи без времеви компонент на преходите, възниква идеята за изграждане на методи за верификация на обобщени мрежи зависещи от времето, т.е. използващи темпорална логика, както и за създаване на програмна реализация, която извежда всички пътища и техните условия, подлежащи на верификация от експерт. Поражда се и идея за изграждане на уеб версия на *GN IDE* и създаване на реактивна система за тези цели. Друг пример е въпросът за възможностите да се конструират идентични („синонимни“) по обхват и действие симулации, обобщеномрежова и научна, които използват едни и същи данни и алгоритми, но се различават по своите изгледи. Тези проблеми не са били обект на разглеждане до момента.

ПРИНОСИ НА ДИСЕРТАЦИОННИЯ ТРУД

Приносите в настоящия дисертационен труд са от научен, научно-приложен и приложен характер.

Научните приноси представени в дисертационния труд са следните:

1. Предложен е формален метод за верификация на обобщеномрежови модели без времеви компонент на преходите.

2. Дефинирани са две нови разширения на стандартните ОМ – обобщени мрежи с ядра с приоритети зависещи от времето (ОМПЗВ) и обобщени мрежи с ядра с динамични приоритети (ОМЯДП).

3. Предложен е алгоритъм за функциониране на преход, където сливането на ядра е позволено.

4. Доказани са две теореми за запазване на поведението на оператори и релации върху редуцирани обобщени мрежи.

Научно-приложните приноси са следните:

1. Създадени са два обобщеномрежови модела на процес на пречистване на отпадни води и на процес на измерване на плоскопаралелни краищни мерки за дължина, в който се отчита субективният фактор. Реализирани са и две обобщеномрежови симулации в *GN IDE* с реални данни, като са изготвени различни тестови сценарии.

2. Създаден е обобщеномрежов модел реализиращ метод на защита на програмното осигуряване. Извършена е верификация на обобщената мрежа по дефинирания в дисертационния труд метод.

Приложните приноси са следните:

1. Интегрира се новият алгоритъм за функциониране на преход, където сливането на ядра е позволено в *GN IDE*.

2. Създава се *GN API*, описва се неговият интерфейс и се показва пример за неговото използване.

3. Създават се оператори за редуциране и функционалност за разпознаване на класове редуцирани обобщени мрежи в *GN IDE*.

Библиография

Номерацията в библиографията към автореферата следва номерацията в библиографията към дисертационния труд.

[1] Андонов, В., *Обобщени мрежи с характеристики на позициите*, Дисертационен труд за присъждане на ОНС доктор по информатика, Институт по биофизика и биомедицинско инженерство, БАН, 2014

[6] Приложно-програмен интерфейс https://bg.wikipedia.org/wiki/Приложно-програмен_интерфейс

[7] Радев, Х., *Метрология и измервателна техника*, Софтрейд, София, Том 1, 2008.

[13] Alexieva, J., E. Choy, E. Koycheva. *Review and bibloigraphy on generalized nets theory and applications*. In:- A Survey of Generalized Nets (E. Choy, M. Krawczak, A. Shannon and E. Szmidt, Eds.), Raffles KvB Monograph No. 10, 2007, 207-301

[14] Apache Ant. http://ant.apache.org/ (last visited Jun 2016).

[25] Atanassov, K., *Generalized Nets.* World Scientific. Singapore, London (1991)

[34] Atanassov, K., *On Generalized Nets Theory*. Prof. M. Drinov Academic Publishing House, Sofia, 2007.

[39] Atanassov, K., *Theory of Generalized nets (an algebraic aspect).* Advances in Modelling & Simulation, AMSE Press Vol. 1 (1984), No. 2, 27-33.

[44] Bakker, J.W. de, Meertens L.G. L.T, *On the Completeness of the Inductive Assertion Method,* JOURNAL OF COMPUTER AND SYSTEM SCIENCES 11, 323-357 (1975)

[54] Chikofsky, E.,J. I. Cross, *Reverse engineering and design recovery: A taxonomy*. IEEE Software, 7(1):13–17, 1990.

[56] Dimitrov, D. G. *GN IDE – A Software Tool for Simulation with Generalized Nets*. Proceedings of Tenth Int. Workshop on Generalized Nets. Sofia, 5 December 2009, 70-75.

[65] IEEE 1012-2004 Standart for Software Verification and Validation. IEEE, 2005, http://pesona.mmu.edu.my/~wruslan/SE2/Readings/detail/Reading-7.pdf

[67] Katz, S., Z. Manna. Acts Informatica (1975), 5:333. doi:10.1007/BF00264565

[73] Manna, Z., *Mathematical theory of computation*, New York, 1974.

[74] Manna, Z., N. Dershowitz. *Proving termination with multiset ordering*. Communications of the ACM, 22(8):465–476, Aug 1979.

[80] Open Source Initiative. https://opensource.org/licenses/MIT (last visited in Dec 2016)

[81] Petri, C.A., "*Kommunikation mit Automaten*. Bonn:Institut fuer Instrumentelle Mathematik, Schriffen des IIM Nr.3, 1962. Also, English translation: "Communication

with Automata", New York: Griffiss Air Force Base. Tech. Rep. RADC-TR-65-377, vol.1, Suppl. 1, 1966. *Conf. of the Union of Bulg. Math.,* Sunny Beach, April 1984, 219-226. Proc. of II Int. Symp. Automation and Scienti c Instrumentation, Varna, May 1983, 391-396..

[84] Radeva, V., M. Krawczak, E. Choy. *Review and bibliography on generalized nets theory and Applications.* Advanced Studies in Contemporary Mathematics, Vol. 4, 2002, No. 2, 173–199.

Списък на публикациите по дисертационния труд

[1*] Andonov V., N. Angelova, *Modifications of the algorithms for transition functioning in GNs, GNCP, IFGNCP1 and IFGNCP3 when merging of tokens is permitted*, Springer Series "Studies in Fuzziness and Soft Computing", 2015, 275–288, ISBN 978-3-319-26301-4.

[2*] Angelova N., M. Todorova, K. Atanassov, *GN IDE: Implementation, Improvements and Algorithms*, Comptes rendus de l'Academie bulgare des Sciences, Tome 69, 2016, No. 4, 411–420, ISSN 1310-1331.

[3*] Angelova, N., D. Dimitrov, *Generalized Nets API*, Issues in IFSs and GNs, Vol. 12, 2015/2016, 93–113, ISBN 978-83-61551-13-3.

[4*] Angelova, N., D. Zoteva, *Implementation of the reducing operators over generalized nets in GN IDE*, Issues in IFSs and GNs, Vol. 12, 2015/2016, 80–92, ISBN 978-83-61551-13-3.

[5*] Angelova, N., *Generalized nets with time dependent priorities*, Proc. of 15th Int. Workshop on Generalized Nets, Burgas, 16 October 2014, 17–22, ISSN: 1313–6860.

[6*] Angelova, N., *Generalized Nets with Dynamic Priorities*, Issues in IFSs and GNs, Vol. 11, 2014, 15–22, ISBN: 978-83-61551-10-2.

[7*] Angelova, N., N. Radoeva, *GN Model of a Length-Gauge Measurement Process*, Issues in IFSs and GNs, Vol. 13, 2017 (in press)

[8*] Georgieva, V., N. Angelova, O. Roeva, T. Pencheva, *Simulation of Parallel Processes in Wastewater Treatment Plant Using Generalized Net Integrated Development Environment*, Comptes rendus de l'Academie bulgare des Sciences, Tome 69, No 11, 2016, 1493–1502, ISSN 1310-1331.

[9*] Zoteva, D., N. Angelova, *Operations and Relations over Reduced Generalized Nets*, Issues in IFSs and GNs, Vol. 13, 2017 (in press)

Списък на проекти с участието на докторанта

ДФНИ-И-02-5/2014 „Интеркритериален анализ – нов подход за вземане на решения“, Фонд „Научни изследвания“, ръководител: чл.-кор. Красимир Атанасов

ДН 02/10 от 17.12.2016 г. „Нови инструменти за извличане на знания от данни и тяхното моделиране“, Фонд „Научни изследвания“, ръководител: доц. д-р Сотир Сотиров