ŽILINSKÁ UNIVERZITA V ŽILINE

Fakulta riadenia a informatiky

AUTOREFERÁT

DIZERTAČNEJ PRÁCE

Žilina, 2024

Ing. Michal Mrena

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky

Ing. Michal Mrena

Autoreferát dizertačnej práce

Reliability analysis of complex systems using decision diagrams

na získanie akademického titulu **„philosophiae doctor“ (v skratke PhD.)**
v študijnom programe doktorandského štúdia
aplikovaná informatika

v študijnom odbore:
informatika

Žilina, apríl 2024

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia **na katedre informatiky** na **Fakulte riadenia a informatiky Žilinskej univerzity v Žiline**.

Predkladateľ: **Ing. Michal Mrena**
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Školiteľ: **doc. Ing. Miroslav Kvaššay, PhD.**
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Oponent: **doc. Ing. Michal Koháni, PhD.**
Katedra matematických metód a operačnej analýzy
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Oponent: **prof. Ing. Radim Briš, CSc.**
Faculty of Electrical Engineering and Computer Science
VSB – Technical university of Ostrava

Oponent: **assoc. prof. Nicolae Brinzei**
Research Center for Automatic Control of Nancy
ENSEM, University of Lorraine

Autoreferát bol rozoslaný dňa: 12. 07. 2024

Obhajoba dizertačnej práce sa koná dňa 21. 8. 2024 o 10:30 h. pred komisiou pre obhajobu dizertačnej práce schválenou pracovnou skupinou odborovej komisie v študijnom odbore **informatika** v študijnom programe **aplikovaná informatika**, vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity v Žiline.

prof. Ing. Ľudmila Jánošíková, PhD.
predsedníčka pracovnej skupiny odborovej komisie
v študijnom odbore **informatika**
v študijnom programe **aplikovaná informatika**

Fakulta riadenia a informatiky
Žilinská univerzita
Univerzitná 8215/1
010 26 Žilina

Abstrakt

Analýza spoľahlivosti systémov je zložitý proces zahŕňajúci mnoho úloh. Mnohé systémy, s ktorými sa v praxi stretávame, označujeme ako komplexné systémy. Okrem bežných úloh a problémov analýzy spoľahlivosti sa musíme pri analýze komplexných systémov vysporiadať s veľkým rozsahom takýchto systémov, rôznorodosťou komponentov a výberom efektívnych algoritmov. Kľúčovým nástrojom analýzy je štruktúrna funkcia, ktorá popisuje topológiu systému. Efektívna reprezentácia štruktúrnej funkcie komplexných systémov je preto dôležitou súčasťou analýzy. V práci sa zameriavame na reprezentáciu štruktúrnej funkcie pomocou rozhodovacích diagramov, ktoré dokážu reprezentovať aj rozsiahle funkcie. Efektívna tvorba a spracovanie diagramov sú preto hlavnými témami tejto práce. Skúmanie vlastností štruktúrnej funkcie nám umožňuje skúmať vlastnosti systému, ktorý funkcia popisuje. Práca sa preto venuje analýze a efektívnej implementácii existujúcich algoritmov. Ďalej práca predstavuje niekoľko vylepšení existujúcich algoritmov, ktoré majú za cieľ zrýchlenie algoritmov alebo uľahčenie ich použitia. Hlavnými prínosmi práce sú predstavenie nového univerzálneho algoritmu na výpočet logických derivácií, úprava existujúcich algoritmov na pravdepodobnostnú analýzu, ktorá umožňuje použitie týchto algoritmov s časovo závislými pravdepodobnosťami stavov komponentov s použitím symbolických výpočtov. Posedným dôležitým prínosom je implementácia softvérového nástroja na analýzu spoľahlivosti s použitím rozhodovacích diagramov, ktorý implementuje všetky navrhnuté a upravené algoritmy.

Kľúčové slová: analýza spoľahlivosti; binárny rozhodovací diagram; časovo závislá analýza spoľahlivosti; pravdepodobnostná analýza spoľahlivosti; softvérové spracovanie rozhodovacích diagramov; štruktúrna funkcia; viachodnotový rozhodovací diagram

Abstract

System reliability analysis is a complicated process involving various tasks. Many of the systems we encounter in practice are referred to as complex systems. In addition to the usual reliability analysis tasks and problems, when analyzing complex systems, we have to deal with the large scale of such systems, the variety of their components, and the selection of efficient algorithms. A key analysis tool is the structure function, which describes the topology of the system. An efficient representation of the structure function of complex systems is, therefore, an important part of the analysis. The thesis focuses on the representation of the structure function using decision diagrams, which can also represent large-scale functions. Efficient diagram creation and processing are therefore the main topics of this thesis. Exploring the properties of the structure function allows for the investigation of the properties of the system that the function describes. Therefore, the thesis deals with the analysis and efficient implementation of existing algorithms. Furthermore, the thesis presents several improvements to the existing algorithms that aim to make the algorithms faster or easier to use. The main contributions of the thesis are the introduction of a new universal algorithm for the computation of logical derivatives, and the modification of existing algorithms for probabilistic analysis, which allows the use of these algorithms with time-dependent probabilities of the states of the components using symbolic computations. Finally, an important contribution is the implementation of a software library for reliability analysis using decision diagrams. The open-source library implements all the proposed and modified algorithms.

Keywords: Binary Decision Diagram; Multi-valued Decision Diagram; probabilistic reliability analysis; reliability analysis; software processing of decision diagrams; structure function; time-dependent reliability analysis

Obsah

Úvod

Analýza spoľahlivosti je dôležitou súčasťou životného cyklu takmer všetkých systémov. Je dôležitá už vo fáze návrhu systémov, kedy nám pomáha zostrojiť systém tak, aby dokázal plniť požadovanú funkcionalitu dostatočne dlhý čas s požadovanou spoľahlivosťou. Nemenej dôležitá je aj pri plánovaní údržby systémov alebo pri identifikácii komponentov kritických pre fungovanie systému.

Prvým krokom analýzy je identifikácia počtu stavov systému. Ďalším krokom je vytvorenie matematického popisu systému. V tejto práci sa zameriavame na popis systému tzv. štruktúrnou funkciou [1]. Štruktúrna funkcia priradí danému stavu komponentov stav systému – popisuje závislosť stavu systému na stave jeho komponentov. Vo všeobecnosti je štruktúrna funkcia diskrétnou funkciou. Jej konkrétna forma závisí od počtu stavov systému a od počtu stavov komponentov systému.

Skúmaním vlastností štruktúrnej funkcie získavame informácie o vlastnostiach skúmaného systému. Jednou z vlastností, ktoré štruktúrna funkcia preberá je komplexnosť systému. Tá môže byť spôsobená napríklad veľkých počtom komponentov systému, rôznou povahou komponentov alebo komplikovanými vzťahmi medzi komponentami. Obzvlášť pri veľkom počte komponentov je preto potrebné štruktúrnu funkciu efektívne reprezentovať. Vhodná reprezentácia musí zvládnuť popísať aj rozsiahle systémy a musí tiež umožňovať efektívne spracovanie v počítači.

Rozhodovací diagram [2], [3] je štruktúra, ktorá spĺňa obe uvedené vlastnosti. Ide o acyklický graf, ktorý bol navrhnutý na efektívnu reprezentáciu diskrétnych funkcií. Rozhodovacie diagramy sa vo všeobecnosti považujú za veľmi efektívny spôsob reprezentácie štruktúrnej funkcie. Povaha komplexných systémov a neustály nárast zložitosti však vytvárajú tlak na neustále zlepšovanie existujúcich techník a navrhovanie nových prístupov. Voľne dostupné softvérové nástroje v súčasnosti poskytujú algoritmy na všeobecnú prácu s rozhodovacími diagramami. Algoritmy na analýzu spoľahlivosti sú však často implementované iba pre konkrétny prípad použitia alebo nie sú voľne dostupné.

Hlavným cieľom tejto práce je preto optimalizácia aplikácie rozhodovacích diagramov pri analýze spoľahlivosti zložitých systémov, z čoho vyplývajú nasledujúce výskumné témy:

- analýza existujúcich prístupov a algoritmov využívaných pri reprezentácii štruktúrnej funkcie pomocou rozhodovacieho diagramu a pri následnej analýze;
- implementácia výkonnej a robustnej softvérovej knižnice na tvorbu a manipuláciu s rozhodovacími diagramami zameranej na využitie diagramov v analýze spoľahlivosti;
- návrh, úprava a zlepšenie existujúcich algoritmov na tvorbu a manipuláciu s rozhodovacími diagramami;
- vytvorenie nových algoritmov a metód založených na využití rozhodovacích diagramov špecializovaných na analýzu spoľahlivosti.

1 Analýza spoľahlivosti

1.1 Počet stavov systému

Pred začiatkom analýzy systému je potrebné identifikovať počet stavov systému. Najjednoduchší prístup je popisovať iba dva stavy systému – systém funguje a systém zlyhal – ktoré popisujeme číslami 1 a 0 v tomto poradí. Takto popísaný systém nazývame dvojstavový systém (BSS z angl. „Binary-State System“) [1], [4]. Pre systémy, ktoré sú zo svojej povahy dvojstavové je takýto prístup postačujúci. Príkladom takéhoto systému je logický obvod. Rovnako je vhodný aj pre systémy, v ktorých môže aj veľmi malé zhoršenie stavu spôsobiť škody na technike alebo ohroziť zdravie ľudí. V takomto prípade môže ísť napríklad o riadiaci systém elektrárne.

Mnohé systémy však dokážu plniť svoju úlohu aj po zhoršení ich stavu. Príkladom môže byť transportná sieť, ktorá funguje s menšou prenosovou kapacitou. Takéto systémy nazývame viacstavové (MSS z angl. „Multi-State System“) [5]. Stavy takýchto systémov popisujeme číslami 0 pre stav, v ktorom systém nefunguje až po číslo $m-1$ pre stav, v ktorom systém funguje bez obmedzení, kde m je celkový počet stavov. Pri tomto type systémov ďalej rozlišujeme homogénne systémy, v ktorých je počet stavov všetkých komponentov a počet stavov systému rovnaký a nehomogénne systémy, v ktorých môžu mať rôzne komponenty a celý systém rozdielny počet stavov. Táto vlastnosť je typická pre systémy zložené z komponentov rôznej povahy – napr. z technických zariadení a ľudí.

Výhodou BSS je ich jednoduchosť a stým spojená jednoduchosť modelov, ktoré ich popisujú – či už z pohľadu veľkosti modelov alebo z pohľadu výpočtovej zložitosti algoritmov. Výhodou je tiež dostupnosť väčšieho množstva algoritmov a nástrojov. Na druhej strane, ich nevýhodou môže byť až prílišné zjednodušenie v prípade popisu systémov, ktoré nie sú vo svojej povahe dvojstavové. V takomto prípade je pre získanie presnejších výsledkov potrebné popisovať takéto systémy ako viacstavové – avšak za cenu zložitejšieho modelu a väčšej výpočtovej zložitosti.

1.2 Štruktúrna funkcia

Štruktúrna funkcia je zobrazenie, ktoré každému stavu komponentov priradí prislúchajúci stav systému. Vo všeobecnosti ide o diskrétnu funkciu, ktorá má v prípade nehomogénneho MSS nasledovnú podobu [6]:

$$\begin{aligned}\phi(x_1, x_2, \dots, x_n) = \phi(\boldsymbol{x}): \{0,1,\dots,m_1-1\} \times \dots \times \{0,1,\dots,m_n-1\} \\ \rightarrow \{0,1,\dots,m-1\},\end{aligned} \tag{1}$$

kde n je počet komponentov systému, x_i popisuje stav i-teho komponentu pre $i = 1,2,\dots,n$; m je počet stavov systému, m_i je počet stavov i-teho komponentu a $x = (x_1, x_2, \dots, x_n)$ je stavový vektor, ktorý obsahuje stav všetkých komponentov.

Definícia (1) popisuje najvšeobecnejší prípad a súhlasí s definíciou celočíselnej funkcie [7]. V špeciálnom prípade homogénneho MSS, kedy platí $m_i = m_j = m$ pre $i, j = 1,2,\dots,n$; má nasledovnú, jednoduchšiu, podobu [6]:

$$\phi(x_1, x_2, \dots, x_n) = \phi(\boldsymbol{x}): \{0,1,\dots,m-1\}^n \rightarrow \{0,1,\dots,m-1\}, \tag{2}$$

ktorá je zhodná s definíciou viachodnotovej logickej funkcie [7]. Ak navyše platí, že $m = 2$, štruktúrna funkcia popisuje BSS a má nasledovnú formu [1]:

$$\phi(x_1, x_2, \dots, x_n) = \phi(\boldsymbol{x}): \{0,1\}^n \to \{0,1\}, \tag{3}$$

ktorá je zhodná s definíciou booleovskej funkcie [8].

Z definícií (1), (2) a (3) je zrejmé, že definícia (3) predstavuje najvšeobecnejší prípad a definície (1) a (2) predstavujú iba špeciálne prípady. V ďalšom popise budeme preto uvažovať štruktúrnu funkciu vo forme (3).

1.3 Logické derivácie

Skúmaním vlastností štruktúrnej funkcie získavame informácie o systéme, ktorý popisuje. Logický diferenciálny počet [7], [9] – podobne ako klasický diferenciálny počet – umožňuje skúmať dynamické vlastnosti diskrétnych funkcií. Dôležitým nástrojom pre analýzu spoľahlivosti sú tzv. logické derivácie, ktoré popisujú, ako sa mení hodnota funkcie pri konkrétnej zmene hodnoty premennej.

Základnou deriváciou je smerová logická derivácia celočíselnej funkcie $f(x)$ podľa premennej x_i, ktorú definujeme nasledovným spôsobom [7]:

$$\frac{\partial f(j \to h)}{\partial x_i(s \to r)} = \begin{cases} 1, & \text{ak } f(s_j, \boldsymbol{x}) = j \text{ a } f(r_j, \boldsymbol{x}) = h \\ 0, & \text{inak,} \end{cases} \tag{4}$$

kde $s, r, j, h \in \{0,1, \dots, m-1\}$, $s \neq r$ a $j \neq h$. Notácia $f(s_j, \boldsymbol{x})$ predstavuje kofaktor funkcie f podľa premennej x_j s hodnotou s. Kofaktor je funkcia $n-1$ premenných, ktorú získame zafixovaním hodnoty premennej x_j na hodnotu s. Podobne je derivácia (4) funkcia $n-1$ premenných, ktorá nadobúda hodnotu 1 iba v bodoch, v ktorých zmena hodnoty premennej x_i z hodnoty s na hodnotu r spôsobí zmenu hodnoty funkcie f z hodnoty j na hodnotu h.

Derivácia (4) popisuje jednu špecifickú zmenu hodnoty funkcie. Pri celočíselnej funkcii však existuje vzhľadom na prípustné hodnoty s, r, j, h relatívne veľký počet konkrétnych derivácií, ktoré je možné vyhodnotiť. Keďže jednotlivé derivácie popisujú iba zlomok všetkých situácií, ich použitie by bolo pomerne nepraktické. Pre získanie obsiahlejšieho pohľadu na správanie funkcie preto používame integrované smerové logické derivácie [10]. Literatúra popisuje tri typy týchto derivácií a to:

- typ I definovaný nasledovne:

$$\frac{\partial f(j \searrow)}{\partial x_i(s \to r)} = \begin{cases} 1, & \text{ak } f(s_i, \boldsymbol{x}) = j \text{ a } f(s_i, \boldsymbol{x}) < j \\ 0, & \text{inak,} \end{cases} \tag{5}$$

- typ II definovaný nasledovne:

$$\frac{\partial f(\searrow)}{\partial x_i(s \to r)} = \begin{cases} 1, & \text{ak } f(s_i, \boldsymbol{x}) > f(r_i, \boldsymbol{x}) \\ 0, & \text{inak,} \end{cases} \tag{6}$$

- a typ III definovaný nasledovne:

$$\frac{\partial f(h_{\geq j} \to h_{<j})}{\partial x_i(s \to r)} = \begin{cases} 1, & \text{ak } f(s_i, \boldsymbol{x}) \geq j \text{ a } f(r_i, \boldsymbol{x}) < j \\ 0, & \text{inak.} \end{cases} \tag{7}$$

Definície (5), (6) a (7) sa v rámci jednotlivých typov môžu líšiť napr. smerom zmeny hodnoty funkcie. Povaha zmeny sa však pre konkrétny typ nemení.

Zmena hodnoty premennej a následná zmena hodnoty štruktúrnej funkcie zodpovedajú zmene stavu komponentu a následnej zmene stavu systému. Logické derivácia preto predstavujú veľmi silný nástroj pre skúmanie vplyvu komponentov na stav systému. V nasledujúcich sekciách preto popíšeme využitie derivácií pri výpočte rôznych ukazovateľov spoľahlivosti.

1.4 Topologická analýza

Štruktúrna funkcia popisuje topológiu systému, na základe ktorej dokážeme vykonať topologickú analýzu. Základným topologickým ukazovateľom je relatívna frekvencia stavov systému vzhľadom na stav j definovaná nasledovne [11]:

$$Fr^{\geq j} = \mathrm{TD}(\phi(\boldsymbol{x}) \geq j), \tag{8}$$

kde $\phi(\boldsymbol{x})$ je štruktúrna funkcia, $j \in \{0,1,\dots,m-1\}$ a notácia $TD(.)$ označuje tzv. hustotu pravdivosti argumentu – relatívny počet vstupných vektorov, pre ktoré argument (funkcia s booleovským výstupom) nadobúda hodnotu 1. Frekvencia stavov systému tak popisuje relatívny počet možných stavov komponentov, pre ktoré je systém v stave j alebo v stave lepšom ako j. Frekvenciu stavov systému môžeme použiť na jednoduché porovnanie dvoch rôznych konfigurácií systému napríklad vo fáze návrhu systému.

Frekvencia stavov systému popisuje celý systém jedným číslom a nehovorí nič o vplyve jednotlivých komponentov systému. Takúto informáciu poskytuje jeden z tzv. ukazovateľov dôležitosti [12] zvaný štruktúrna dôležitosť. Štruktúrnu dôležitosť (SI z angl. „Structural Importance“) je možné definovať viacerými spôsobmi. Z pohľadu vyhodnocovania je veľmi výhodná definícia pomocou logickej derivácie [13]:

$$\mathrm{SI}_i = \mathrm{TD}\left(\frac{\partial f\left(h_{\geq j} \to h_{<j}\right)}{\partial x_i(s \to r)}\right). \tag{9}$$

SI (9) popisuje relatívny počet situácií kedy zmena stavu i-teho komponentu zo stavu s do stavu r spôsobí zmenu stavu systému popísanú deriváciou. V definícii (9) sme použili integrovanú smerovú logickú deriváciu typu III (7). Pre výpočet SI je však možné použiť aj ostatné typy derivácií. Presný význam SI potom závisí od použitej derivácie.

1.5 Pravdepodobnostná analýza

Nevýhodou topologickej analýzy je, že predpokladá rovnakú pravdepodobnosť stavov komponentov. Stavy komponentov sa však v praxi vyskytujú s rôznymi pravdepodobnosťami. Pre získanie presnejších charakteristík systému je preto potrebné vziať do úvahy aj pravdepodobnosti stavov komponentov. Ďalšou dôležitou vlastnosťou je, že pravdepodobnosť stavov komponentov sa v čase mení. Preto rozlišuje časovo závislé a časovo nezávislé pravdepodobnostné charakteristiky.

Časovo nezávislé pravdepodobnosti stavov komponentov označujeme nasledovne:

$$p_{i,k} = \Pr\{x_i = k\}, \tag{10}$$

kde $i = 1,2,\dots,n$ a $k = 0,1,\dots,m_i - 1$. Notácia (10) označuje pravdepodobnosť, že komponent i je v stave k. Časovo závislé pravdepodobnosti stavov komponentov označujeme podobne:

$$p_{i,s}(t) = \Pr\{Z_i(t) = s\}, \tag{11}$$

kde $s = 0,1, \dots m_i - 1$, premenná t reprezentuje čas a funkcia $Z_i(t)$ popisuje stav i-teho komponentu v čase t.

Základnou pravdepodobnostnou charakteristikou systému je dostupnosť vzhľadom na stav systému j. Časovo nezávislú dostupnosť MSS definujeme nasledovne [5]:

$$A^{\geq j}(\boldsymbol{p}) = \Pr\{\phi(\boldsymbol{x}) \geq j\}, \tag{12}$$

kde $j \in \{1,2, \dots, m-1\}$ a $\boldsymbol{p}$ je matica pravdepodobností stavov komponentov. Dostupnosť (12) zodpovedá pravdepodobnosti, že systém je v stave j alebo v lepšom stave. Komplementárnym ukazovateľom k dostupnosti je nedostupnosť systému vzhľadom na stav j definovaná nasledovne [5]:

$$U^{\geq j}(\boldsymbol{p}) = \Pr\{\phi(\boldsymbol{x}) < j\}, \tag{13}$$

ktorá zodpovedá pravdepodobnosti, že systém je v stave horšom ako j.

Podobne ako frekvencia stavov systému (8) popisujú dostupnosť a nedostupnosť celého systému jednou pravdepodobnosťou. Keďže však môžu byť rôzne komponenty rôzne spoľahlivé, neponúkajú žiadnu informáciou o dôležitosti jednotlivých komponentov. Takúto informáciou poskytujú rôzne ukazovatele dôležitosti. Jedným z bežne používaných ukazovateľov je Birnbaumova dôležitosť (BI z angl. „Birnbaum importance“). Podobne ako pri SI (9) je z praktického hľadiska výhodná definícia pomocou logickej derivácie [13]:

$$\mathrm{BI}_{i,s}^{\geq} = \Pr\left\{\frac{\partial\phi\left(h_{\geq j} \to h_{<j}\right)}{\partial x_i(s \to s-1)} \leftrightarrow 1\right\}. \tag{14}$$

BI (14) udáva pravdepodobnosť, že zhoršenie stavu i-teho komponentu zo stavu s to stavu $s-1$ spôsobí zmenu stavu systému popisovanú deriváciou. Notácia $. \leftrightarrow 1$ zodpovedá javu kedy ľavý argument nadobúda hodnotu 1. V definícii (14) sme použili integrovanú smerovú logickú deriváciu typu III (7). Pre výpočet BI je však možné použiť aj ostatné typy derivácií. Presný význam BI potom závisí od použitej derivácie.

Definície časovo závislých verzií vyššie popísaných pravdepodobnostných ukazovateľov sú podobné ich časovo nezávislým ekvivalentom. Zásadným rozdielom je však argument, ktorý už nie je jednoduchá matica pravdepodobností, ale je ním vektor stavových funkcií.

Časovo závislá dostupnosť systému vzhľadom na stav systému j je definovaná nasledovne [5]:

$$A^{\geq j}(t) = \Pr\{\phi(\boldsymbol{Z}(t)) \geq j\}, \tag{15}$$

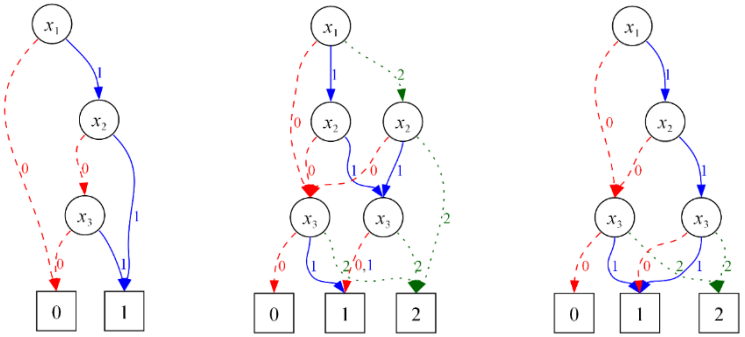
kde $\boldsymbol{Z}(t) = \left(Z_1(t), Z_2(t), \dots, Z_n(t)\right)$ je vektor stavových funkcií jednotlivých komponentov.

2 Rozhodovacie diagramy

Pre zostrojenie a vyhodnotenie štruktúrnej funkcie je potrebné vybrať vhodný spôsob jej reprezentácie. Reprezentácia musí umožniť efektívne spracovanie v počítači a zároveň umožniť efektívne reprezentovať aj rozsiahlejšie funkcie popisujúce komplexné systémy. Rozhodovací diagram je grafová štruktúra navrhnutá na reprezentáciu diskrétnych funkcií, ktorá spĺňa obe uvedené vlastnosti. Binárny rozhodovací diagram [2] (BDD z angl. „Binary Decision Diagram“) je najjednoduchší typ rozhodovacieho diagramu navrhnutý na reprezentáciu booleovských funkcií. Jeho zovšeobecnením je viachodnotový rozhodovací diagram [3] (MDD z angl. „Multi-valued Decision Diagram“) navrhnutý na reprezentáciu viachodnotových logických funkcií (2) a celočíselných funkcií (1). Vo zvyšku textu budeme uvažovať najvšeobecnejší prípad MDD, ktorý reprezentuje celočíselnú funkciu.

2.1 Štruktúra diagramu

MDD je orientovaný acyklický graf, ktorý sa skladá z vnútorných vrcholov, ktoré reprezentujú premenné a koncových vrcholov, ktoré reprezentujú hodnoty funkcie. Vrcholy diagramu sú uložené na úrovniach. Jedna úroveň obsahuje vnútorné vrcholy reprezentujúce rovnakú premennú, s výnimkou poslednej úrovne, ktorá obsahuje koncové vrcholy. Vnútorný vrchol reprezentujúci premennú x_i má m_i výstupných hrán, ktoré vedú do vrcholov na nižších úrovniach. Koncový vrchol nemá žiadne výstupné hrany a počet koncových vrcholov je najviac m. Ukážku všetkých štruktúry rôznych typov MDD môžeme vidieť na Obr. 1.



Obr. 1 Vľavo: BDD reprezentujúci booleovskú funkciu, uprostred: MDD reprezentujúci viachodnotovú logickú funkciu, vpravo: MDD reprezentujúci celočíselnú funkciu

2.2 Tvorba diagramov

Kľúčovou vlastnosťou MDD je unikátnosť a zdieľanie vrcholov. Tieto vlastnosti sú prakticky zabezpečené udržiavaním tzv. tabuľky unikátnych vrcholov – pred vytvorením nového vrcholu sa najprv kontroluje, či už požadovaný vrchol neexistuje v tabuľke.

Základom vytvorenia MDD je tzv. priama tvorba – MDD reprezentujúci funkciu jednej premennej alebo konštantnú funkciu môžeme vytvoriť jednoducho bez potreby sofistikovanejšieho algoritmu. MDD reprezentujúci komplikovanejšie funkcie môžeme následne získať spájaním priamo

vytvorených diagramov pomocou binárnych operácií ($\wedge$,$\vee$,$\oplus$, $min, max, \ldots$). Takýto prístup nazývame dynamická tvorba MDD. Na spájanie používame algoritmus *apply* [2] ktorého vstupom sú dva MDD a binárna operácia a výstupom je nový MDD, ktorý reprezentuje novú funkciu získanú spojením vstupných funkcií binárnou operáciou. Opakovaným použitím algoritmu *apply* tak môžeme vytvoriť MDD reprezentujúci ľubovoľnú funkciu.

Pre úplnosť spomenieme ďalší možný prístup k tvorbe MDD zvaný statická tvorba, ktorá spočíva v transformácii pravdivostnej tabuľky na MDD [14]. Tento prístup je vhodný pre tvorbu menších MDD, ktoré môžu slúžiť ako vstup do procesu dynamickej tvorby.

2.3 Zefektívnenie tvorby diagramov

Vytvorenie MDD reprezentujúceho štruktúrnu funkciu je nutný krok k následnej analýze systému. V prípade komplexných systémov môže byť štruktúrna funkcia rozsiahla a komplikovaná, preto je potrebné vytvorenie diagramu vykonať čo najefektívnejšie. Časť práce sa preto venuje zefektívneniu a zjednodušeniu tohto procesu.

2.3.1 Zovšeobecnenie spájania diagramov

Algoritmus *apply* môžeme považovať za binárnu operáciu. Mnohé funkcie/podsystémy, ktoré chceme popisovať sú však d-árne (kde $d \in \mathbb{N}$). Príkladom môže byť trojvstupové logické hradlo (hradlo AND na Obr. 2) alebo paralelný (pod)systém (Obr. 2). Bežne používaným riešením je viacnásobné použitie binárneho algoritmu apply. V prípade logického obvodu na Obr. 2 by tak volanie *apply* mohlo vyzerať nasledovne:

$$\text{APPLY}(\text{APPLY}(\text{APPLY}(X_1, X_2, \wedge), X_3, \wedge), X_4, \vee),$$

kde zápis X_i predstavuje MDD reprezentujúci funkciu jednej premennej x_i. Oveľa prirodzenejšie by však bolo volanie *apply* nasledovným spôsobom:

$$\text{APPLY}(\mathbf{Apply}(\boldsymbol{X_1}, \boldsymbol{X_2}, \boldsymbol{X_3}, \wedge), X_4, \vee).$$

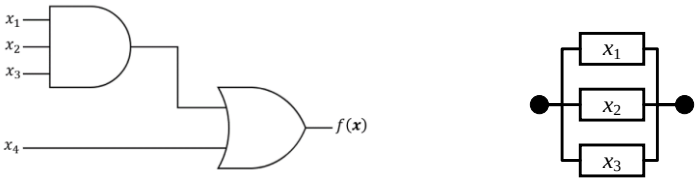
Vo zvýraznenej časti výrazu môžeme vidieť pomyselnú ternárnu verziu *apply*.

V práci sme preto navrhli zovšeobecnenú verziu algoritmu *apply*, ktorú sme pomenovali *extended apply*. Podobne ako základná verzia algoritmu je tento postavený na vzťahu, ktorý popisuje vnútorný vrchol diagramu pomocou Shannonovej expanzie [15]. Nami zovšeobecnený vzťah má nasledovnú podobu:

$$\bigodot_d (f_1, f_2, \ldots, f_d)(\boldsymbol{x}) = \sum_{k=0}^{m_i-1} \Big(\{x_i \leftrightarrow k\} * \big(\bigodot_d (f_1, f_2, \ldots, f_d)(k_i, \boldsymbol{x}) \big) \Big), \tag{16}$$

kde výraz $\{x_i \leftrightarrow k\}$ predstavuje tzv. logický bikondicionál, ktorý nadobúda hodnotu 1 práve vtedy a len vtedy, ak premenná x_i nadobúda hodnotu k a $\bigodot_d$ je d-árna asociatívna operácia. Členy súčtu sa dajú stotožniť s výstupnými hranami vnútorného vrcholu a celý výraz s vnútorným vrcholom MDD.

Navrhnutý algoritmus sme experimentálne porovnali so základnou verziou [16]. V experimente sme vytvárali MDD reprezentujúce náhodné d-cestné stromy pomocou nášho algoritmu extended apply s rôznou aritou. Výsledky porovnania môžem vidieť v Tab. 1.



Obr. 2 Príklady systémov, na popis ktorých je vhodnejšie použiť ternárne funkcie; vľavo jednoduchý logický obvod; vpravo paralelný systém s tromi komponentami

Výsledky porovnania ukazujú, že základná verzia dosahuje lepšie výsledky v porovnaní s rozšírenými verziami – a teda použitím rozšíreného algoritmu nedosahujeme výrazné zrýchlenie. Na druhej strane, verzia s aritou 3 sa ukázala byť rovnako rýchla a efektívnejšia z hľadiska počtu krokov algoritmu, čo naznačuje, že môžu existovať prípady použitia, v ktorých je rozšírená verzia vhodnejšia.

Tab. 1 Priemerný čas v milisekundách potrebný na vytvorenie MDD z výrazového stromu popisujúceho sériovo-paralelný systém zložený z n_{max} komponentov

n_{max}	d			
	2	**3**	**4**	**5**
20 000	79	78	94	113
40 000	177	176	209	252
60 000	280	278	333	401
80 000	384	383	457	550
100 000	500	495	592	712

2.3.2 Poradie spájania diagramov

Pri tvorbe diagramov je často potrebné spojiť rádovo desiatky až stovky MDD rovnakou operáciou $\odot$. Jedno použitie algoritmu *extended apply* pri takto vysokom počte MDD nie je vhodné kvôli veľkej výpočtovej náročnosti analyzovanej v texte práce. Riešením je preto viacnásobné použitie či už základnej alebo rozšírenej verzie *apply*. Kvôli asociativite operácie $\odot$ je možné toto spojenie vykonať mnohými spôsobmi. V práci sme analyzovali prístup zvaný *left-fold*, v ktorom diagramy spájame sekvenčne zľava doprava:

$$(((D_1 \odot D_2) \odot D_3) \odot D_4) \odot D_5$$

a prístup *tree-fold*, v ktorom diagramy spájame hierarchicky postupne po dvojiciach:

$$((D_1 \odot D_2) \odot (D_3 \odot D_4)) \odot D_5,$$

kde notácia D_i predstavuje počiatočný MDD.

V práci sme skúmali, či a ako poradie spájania ovplyvňuje rýchlosť vytvorenia diagramu [17], [18]. V Tab. 2 môžeme vidieť výsledky jedného z experimentálnych porovnaní, v ktorom sme vytvárali BDD reprezentujúce binárne sčítačky popísané PLA súbormi [91].

Výsledky experimentu ukázali, že v konkrétnom prípade použitého benchmakru dokáže poradie spájania výrazne ovplyvniť rýchlosť vytvorenia výsledného diagramu. V tomto konkrétnom prípade sa prístup tree-fold (ktorý je menej intuitívny) ukázal ako výrazne rýchlejší. Je však dôležité spomenúť, že v ďalší experimentoch s inými dátami sa, naopak, prístup left-fold ukázal ako rýchlejší. Z výsledkov našich experimentov preto usudzujeme, že pre nástroje na prácu s diagramami je výhodné implementovať oba prístupy.

Tab. 2 Priemerný čas v milisekundách potrebný na vytvorenie BDD reprezentujúcich výstupy binárnej sčítačky

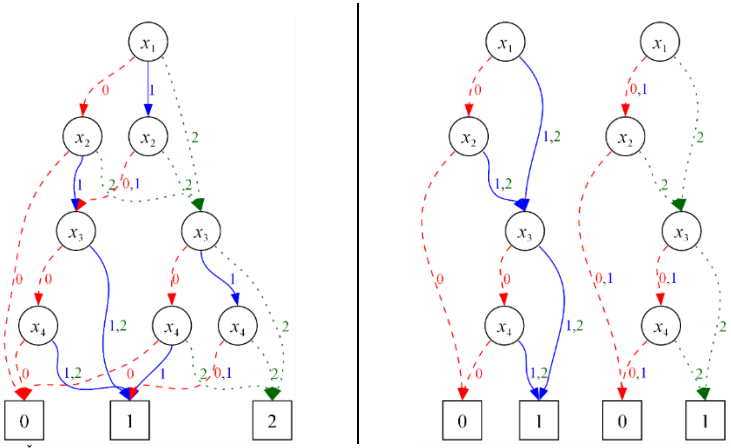
Počet bitov sčítačky	Počet spájaných diagramov	Left fold [ms]	Tree fold [ms]
10-adder-col	10 191	163	71
11-adder-col	20 427	477	180
12-adder-col	40 911	1 828	448
13-adder-col	81 867	5 342	1 084
14-adder-col	163 783	16 579	2 753

3 Rozhodovacie diagramy v analýze spoľahlivosti

Výsledky, ktoré sme popísali v predchádzajúcej sekcii je možné aplikovať na tvorbu diagramov vo všeobecnosti. Práca je však zameraná na špecifické využitie diagramov v analýze spoľahlivosti. Vo zvyšku textu sa preto venujeme tejto problematike.

3.1 Reprezentácia štruktúrnej funkcie

Primárne a pre ďalšiu analýzu nevyhnutné využitie diagramov v analýze spoľahlivosti spočíva v reprezentácii štruktúrnej funkcie. Základný a intuitívny prístup spočíva v reprezentácii celej štruktúrnej funkcie $\phi(\boldsymbol{x})$ jedným MDD. V prípade MSS však existuje aj alternatívny prístup, kedy každý stav systému popíšeme individuálne funkciami $\phi(x) \geq 1, \phi(x) \geq 2, \ldots, \phi(x) \geq m-1$ [19]. Vizuálne porovnanie týchto dvoch prístupov môžeme vidieť na Obr. 3. Nevýhodou tohto prístupu je potreba vytvorenia niekoľkých MDD namiesto jedného. Na druhej strane, výhodou môže byť zjednodušenie popisu jednotlivých stavov systému, keďže každý je popísaný samostatne.



Obr. 3 Štruktúrna funkcia $\phi(\boldsymbol{x})$ reprezentovaná jedným diagramom (vľavo) a sériou diagramov (vpravo) pozostávajúcej z funkcií $\phi(\boldsymbol{x}) \geq 1$ a $\phi(\boldsymbol{x}) \geq 2$ v tomto poradí (vpravo)

Zaujímavou otázkou je tiež porovnanie veľkosti resp. celkového počtu vrcholov MDD potrebných na reprezentáciu systému pri použití daných prístupov. Na zodpovedanie tejto otázky sme vykonali experiment, v ktorom sme porovnávali oba prístupy pri reprezentácii náhodne generovaných sériovo-paralelných systémov – ktoré považujeme za komplexné pri veľkom počte komponentov, kedy je veľkosť reprezentácie obzvlášť dôležitá. Výsledky porovnania môžeme vidieť v Tab. 3.

Tab. 3 Priemerný počet vrcholov v jednom MDD a v sérii MDD v závislosti od počtu komponentov systému (n) v prípade homogénnych sériovo-paralelných 3, 4 a 5 stavových MSS

n	Jeden MDD			Séria MDD		
	3	4	5	3	4	5
500	1 995	5 138	10 756	1 002	1 502	2 002
1 000	4 232	11 436	24 926	2 002	3 002	4 002
1 500	6 562	18 210	40 591	3 002	4 502	6 002
2 000	8 959	25 346	57 424	4 002	6 002	8 002
2 500	11 406	32 749	75 128	5 002	7 502	10 002

Výsledky experimentu jednoznačne ukázali, že v prípade sériovo-paralelných systémov je reprezentácia pomocou série diagramov výrazne výhodnejšia ako použitie jedného diagramu. Tento výsledok je konzistentný aj pre systémy rôznych veľkostí s rozdielnym počtom stavov.

3.2 Výpočet frekvencie stavov systému

Frekvencia stavov systému (8) je jedna zo základných topologických charakteristík systému, ktorou dokážeme jednoducho porovnať systémy s rôznymi topológiami. Jej výpočet spočíva vo vyhodnotení jednoduchého zlomku:

$$Fr^{\geq j} = \frac{\alpha_{\phi,j}}{\alpha_{\phi}}, \tag{17}$$

kde $\alpha_{\phi,j}$ je počet stavových vektorov (situácií), kde je systém v stave $\geq j$ a α_{ϕ} je celkový počet stavových vektorov. Na vyčíslenie čitateľa môžeme využiť exitujúci algoritmus *satisfy-count* [2] a menovateľ vypočítame jednoduchým súčinom $\prod_{i=1}^{n} m_i$. Praktický problém takéhoto výpočtu je, že aj keď je výsledok z intervalu [0,1], čitateľ a menovateľ zlomku sú často čísla, ktoré nie je možné reprezentovať 64-bitovými údajovými typmi.

V prípade BSS je možné problém s limitovaným rozsahom údajových typov vyriešiť vypočítaním logaritmov čitateľa a menovateľa a jednoduchou úpravou vzťahu (17), ktorý popisujeme v práci. Tento postup sa však nedá aplikovať vo všeobecnosti na MSS. V práci sme preto popísali špecializovanú verziu existujúceho algoritmu na výpočet pravdepodobností, ktorá nie je limitovaná rozsahom údajových typov. Otázkou zostávalo, či má zmysel aplikovať tento algoritmus aj na BSS alebo či je v tomto prípade postup využívajúci logaritmy rýchlejší. Na zodpovedanie otázky sme vykonali experimentálne porovnanie spomínaných prístupov vrátane základného na BDD reprezentujúcich náhodne generované systémy. Výsledky porovnania môžeme vidieť v Tab. 4.

Tab. 4 Priemerný čas v mikrosekundách potrebný na výpočet frekvencie stavov systému pomocou rôznych prístupov

n	Satisfy-count [µs]	Satisfy-count-log [µs]	Náš algoritmus [µs]
10	12	14	10
30	1 387	1 823	1 337
60	59 157	69 047	57 107
80	[1] 471 942	191 991	161 950
90	[1] 788 309	329 563	287 166
100	[1] 1 057 991	470 303	407 762

[1] S použitím knižnice GMP pre výpočty s neobmedzenou presnosťou

Výsledky experimentu ukázali, že náš algoritmus funguje lepšie ako prístup využívajúci logaritmy. Potvrdil aj náš predpoklad, že hoci je možné použiť základný prístup založený na algoritme *satisfy-count* – ktorý má porovnateľný výkon pre $n < 63$ – výpočty zahŕňajúce celé čísla s neobmedzenou presnosťou sú podstatne pomalšie. Preto sme dospeli k záveru, že je lepšie použiť náš algoritmus aj pre špeciálny prípad BSS.

3.3 Efektívny výpočet logických derivácií

Logické derivácie sú veľmi užitočný nástroj pre výpočet mnohých ukazovateľov spoľahlivosti, ako napr. rôznych ukazovateľov dôležitosti [12] alebo napríklad minimálnych rezných vektorov [20]. Ich efektívny výpočet je preto pre proces analýzy komplexných systémov veľmi dôležitý.

Deriváciu funkcie reprezentovanej MDD môžeme relatívne jednoducho vypočítať nasledovaním jej definície (napr. (7)) a s použitím existujúcich algoritmov na manipuláciu MDD konkrétne *cofactor* [2], *transform* a *apply* [2]. Tento prístup funguje pomerne dobre, avšak využitie všeobecných algoritmov nemôže naplno využiť špecifiká výpočtu derivácií. Jeho nevýhodou je tiež že, v základnej podobe nie je univerzálny – pre výpočet rôznych derivácií musíme uvedené algoritmy kombinovať iným spôsobom s rôznymi parametrami.

V práci sme preto navrhli špecializovaný algoritmus na výpočet ľubovoľnej derivácie [21]. Náš algoritmus vychádza z nasledujúcej univerzálnej definície logickej derivácie:

$$\frac{\partial f(\Lambda)}{\partial x_i(s \to r)} = \Lambda\big(f(s_i, \boldsymbol{x}), f(r_i, \boldsymbol{x})\big), \tag{18}$$

kde notácia Λ predstavuje funkciu, ktorá popisuje, či zmena funkcie daná jej argumentami predstavuje požadovanú zmenu – v tom prípade vráti hodnotu 1 – alebo nie a vráti hodnotu 0. Funkcia Λ je parametrom nášho algoritmu, ktorý tak môžeme použiť na výpočet ľubovoľnej logickej derivácie.

Po technickej stránke náš algoritmus kombinuje algoritmy *cofactor*, *transform* a *apply*, ktoré sa v základom prístupe používajú samostatne, do jedného kroku. Na základe tejto vlastnosti sme predpokladali, že náš algoritmus by mal byť vo výpočet derivácií rýchlejší. Na kvantifikovanie rozdielu v rýchlosti sme preto vykonali experimentálne porovnanie základného prístupu a nášho algoritmu pri výpočte integrovanej derivácie typu III (7) z MDD reprezentujúcich náhodne generované systémy. Výsledky porovnania môžeme vidieť v Tab. 5.

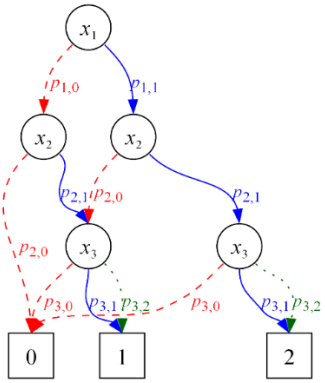
Tab. 5 Priemerný čas v milisekundách potrebný na výpočet IDPLD typu III pre každú premennú pomocou základného postupu a pomocou nášho algoritmu

m	n	Počet vrcholov	Základný prístup [ms]	Náš algoritmus [ms]	Náš algoritmus / základný prístup
2	32	128 322	3 570	1 538	0,43079
3	23	531 698	6 005	2 978	0,49597
4	20	1 591 344	18 540	9 625	0,51917
5	17	1 401 163	13 208	6 874	0,52042

Experimentálne porovnanie ukazuje, že náš algoritmus je približne o 50% rýchlejší ako základný prístup. Keďže výpočet logických derivácií je jedným zo základných krokov analýzy spoľahlivosti a dôležitosti komponentov, náš algoritmus môže výrazne urýchliť proces analýzy komplexných systémov.

3.4 Pravdepodobnostné vyhodnotenie diagramov

Pravdepodobnostná analýza poskytuje v porovnaní s jednoduchou topologickou analýzou oveľa presnejší popis správania systému a vplyvu jednotlivých komponentov prostredníctvom ukazovateľov akými sú napr. dostupnosť systému (12) alebo Birnbaumova dôležitosť (14) a mnohých iných. Pri použití MDD na reprezentáciu štruktúrnej funkcie je pri vyhodnocovaní všetkých pravdepodobnostných ukazovateľov kľúčovou úlohou výpočet pravdepodobnosti navštívenia [22] koncového vrcholu MDD (NTP z angl. „Node Traversing Probability“). MDD je našťastie pre výpočet pravdepodobností veľmi výhodnou štruktúrou. Pravdepodobnosti stavov komponentov môžeme pomyselne stotožniť s hranami MDD, ako môžeme vidieť na Obr. 4. Výpočet pravdepodobností je následne záležitosťou vhodnej prehliadky diagramu a násobenia vhodných pravdepodobností.



Obr. 4 Pravdepodobnostný rozhodovací diagram s pravdepodobnosťami stavov komponentov znázornenými na hranách diagramu

V literatúre existujú dva zásadné prístupy/algoritmy k výpočtu NTP zvané *bottom-up* a *top-town*. Tieto dva prístupy sa líšia typom prehliadky, ktorú pri výpočte používajú. Preto ich v práci označujeme aj ako *post-order* a *level-order* algoritmy. Jedným z výsledkov prezentovaných v práci je porovnanie týchto prístupov pri výpočte rôznych pravdepodobnostných ukazovateľov. Výsledky nášho porovnania môžeme vidieť v Tab. 6.

Tab. 6 Priemerný čas v milisekundách potrebný na výpočet pravdepodobnosti všetkých stavov systému

n	m	Bottom-up	Top-down [ms]
50 000	3	15	20
10 000	5	58	30
40	3	1 621	1 925
20	5	573	330

Algoritmy sme porovnávali pri výpočte pravdepodobností všetkých stavov m-stavového systému zloženého z n-komponentov so sériovo-paralelnou topológiou (prvé dva riadky tabuľky) a s náhodnou topológiou (posledné dva riadky tabuľky). Výsledky ukázali, že pri vyššom počte stavov systému je výhodnejšie použiť *top-down* algoritmus a, naopak, pri nižšom počte stavov systému je výhodnejší *bottom-up* algoritmus. Ďalším rozdielom, ktorý v práci popisujeme, a ktorý je

potrebné pri výbere algoritmu zvážiť je, že jedno vykonanie *top-down* algoritmu umožňuje vypočítať individuálne pravdepodobnosti stavov systému ako aj dostupnosť pre rôzne úrovne. Na druhej strane jednoduchší *bottom-up* algoritmus umožňuje pri jednom vykonaní výpočet iba jednej charakteristiky.

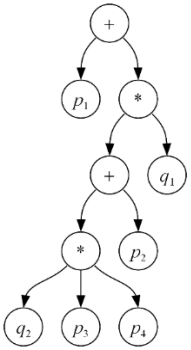
3.5 Časovo závislé pravdepodobnostné výpočty

Vo vyššie popísanom porovnaní prístupov k výpočtu NTP sme pracovali s konštantnými pravdepodobnosťami stavovo komponentov. Posledná časť práce sa venuje modifikácií uvedených algoritmov, ktorá umožní pracovať aj s pravdepodobnosťami stavov komponentov, ktoré už nie sú jednoduchými konštantami, ale sú funkciami času.

Prvým riešeným problémom je výpočet NTP (ktorá súhlasí s vybraným pravdepodobnostným ukazovateľom) v mnohých časových okamihoch. Na riešenie tohto problému sme identifikovali dva prístupy, ktoré sme nazvali základný a symbolický. Základný prístup využíva algoritmus *bottom-up* alebo *top-down* bez modifikácií. V rámci dodatočného kroku však potrebuje vyhodnotiť všetky pravdepodobnosti stavov komponentov v danom čase t. Tento prístup môžeme stručne zosumarizovať nasledovným pseudokódom, ktorý prezentuje funkciu na výpočet dostupnosti systému vo všetkých časových okamihoch uložených v zozname *timePoints*:

function EVALUATEBASIC(*diagram, j, timePoints*, $\mathbb{P}$)
  for $\forall\, t \in$ *timePoints* **do**
    $\mathbb{P}_t \leftarrow$EVALUATEDISTRIBUTIONS($\mathbb{P}$, t)
    values $\leftarrow \{a \mid j \leq a < m\}$
    $A^{\geq j}(t) \leftarrow$CALCULATENTPPOSTSTEP(*diagram, values*, $\mathbb{P}_t$)
  end for
end function.

Symbolický prístup je založený na využití symbolických výpočtov, podľa ktorých nesie svoje pomenovanie. Podobne ako základný prístup využíva jeden z dvojice algoritmov. Tento však musí byť upravený alebo vhodne implementovaný tak, aby dokázal sčítavať a násobiť symbolické výrazy, ktoré môže byť reprezentované napríklad výrazovým stromom. Príklad takéhoto stromu môžeme vidieť na Obr. 5.



Obr. 5 Výrazový strom reprezentujúci výraz, ktorý popisuje dostupnosť BSS; premenné p_i a q_i reprezentujú pravdepodobnosti, že i-ty komponent funguje a nefunguje v tomto poradí

Symbolický prístup tak najprv využije jeden z algoritmov na získanie výrazového stromu reprezentujúceho vybraný pravdepodobnostný ukazovateľ (napr. dostupnosť systému). Vstupom algoritmu je v tomto prípade matica symbolických výrazov. Následne už vyhodnocuje iba získaný strom v každom časovom okamihu. Podobne ako pri základom prístupe môžeme symbolický prístup zosumarizovať nasledovným pseudokódom:

```
function EVALUATESYMBOLIC(diagram, j, timePoints, ℙ)
    exprTree ←CREATETREE(diagram, ℙ)
    for ∀ t ∈ timePoints do
        A^≥j (t) ←EVALUATETREE(exprTree, t)
    end for
end function.
```

V našej knižnici TeDDy [23] sme symbolický prístup implementovali pomocou knižnice GiNaC [24], ktorá podporuje prácu so symbolickými výrazmi v jazyku C++.

Zaujímavou otázkou je ako sa dva uvedené prístupy líšia z pohľadu časovej náročnosti na výpočet pravdepodobnostného ukazovateľa v mnohých časových okamihoch. Za účelom preskúmania tohto rozdielu sme vykonali experimentálne porovnanie týchto prístupov pri výpočte dostupnosti náhodne generovaných sériovo-paralelných BSS. Výsledky tohto porovnania môžeme vidieť v Tab. 7. Stĺpce |BDD| a |Strom| obsahujú veľkosť danej štruktúry. Zvyšné stĺpce obsahujú celkový priemerný čas v nanosekundách potrebný na vyhodnotenie dostupnosti systému v 10 časových okamihoch.

Tab. 7 Porovnanie základného a symbolického prístupu pri výpočte dostupnosti náhodne generovaných sériovo-paralelných systémov

n	**|BDD|**	**|Strom|**	**Základný prístup [ns]**	**Vytvorenie stromu [ns]**	**Symbolické výpočty [ns]**
10	12	599	1 739	26 187	3 823 367
20	22	15 218	3 606	51 791	101 280 004
30	32	546 208	6 020	82 222	3 595 178 608
40	42	11 494 828	7 401	103 151	72 100 562 769

Výsledky experiment ukázali, že základný prístup funguje oveľa lepšie ako symbolický prístup, ak berieme do úvahy rýchlosť vyhodnotenia NTP. Podobné výsledky sme získali aj z iných experimentov, ktoré vyhodnocovali tisícky rôznych časových okamihov. Hoci výsledky sú špecifické pre našu implementáciu – knižnicu TeDDy a knižnicu GiNaC na manipuláciu s výrazmi – relatívny rozdiel medzi oboma prístupmi je značný. Preto je nepravdepodobné, že by sa pri iných implementáciách výrazne zmenil.

Symbolický prístup dosahuje v porovnaní so základným pomerne zlé výsledky. Na druhej strane nám však poskytuje možnosti, ktoré nemôžeme základným prístupom dosiahnuť. Jednou z nich je napríklad možnosť získať výraz popisujúci pravdepodobnostný ukazovateľ. Výraz môžeme analyzovať napríklad použitím knižnice GiNaC alebo ho môžeme exportovať do systému ako napr. Matlab alebo wxMaxima.

Záver

Práca sa zaoberala aplikáciou rozhodovacích diagramov pri analýze spoľahlivosti komplexných systémov. Poskytla obsiahly prehľad krokov procesu analýzy spoľahlivosti so zameraním na algoritmy založené na využití rozhodovacích diagramov reprezentujúcich štruktúrnu funkciu. Ďalej predstavila niekoľko nových algoritmov a poskytla rôzne vylepšenia a zovšeobecnenia existujúcich algoritmov. Prínosom práce sú výsledky riešenia nasledujúcich výskumných problémov:

- Analýza existujúcich prístupov a algoritmov používaných pri reprezentácii štruktúrnej funkcie pomocou rozhodovacích diagramov a ich následná analýza:
  - ✓ v úvodnej časti práca predstavila všeobecné prístupy používané pri analýze spoľahlivosti;
  - ✓ ďalej opísala diskrétne funkcie ako matematický základ štruktúrnej funkcie a spôsoby ich analýzy a reprezentácie;
  - ✓ nakoniec úvodnej časti sa zaoberala rozhodovacími diagramami a ich aplikáciami v analýze spoľahlivosti.
- Implementácia výkonnej a robustnej softvérovej knižnice na tvorbu a manipuláciu s rozhodovacími diagramami:
  - ✓ prvá časť jadra práce predstavila základné aspekty softvérovej knižnice na tvorbu a manipuláciu s rozhodovacími diagramami;
  - ✓ všetky algoritmy a techniky opísané v práci boli implementované v open-source knižnici TeDDy.
- Vyhodnocovanie, úprava a zlepšovanie existujúcich algoritmov na tvorbu a manipuláciu s rozhodovacími diagramami;
  - ✓ praktická časť práce poskytla experimentálne porovnanie rôznych spôsobov poradia vyhodnocovania algoritmu aplikácie;
  - ✓ ďalej vyhodnotila rôzne prístupy k reprezentácii štruktúrnej funkcie sériovo-paralelných systémov;
  - ✓ práca tiež predstavila univerzálny algoritmus na výpočet stavových frekvencie stavov systému.
- Vytvorenie nových algoritmov a metód rozhodovacích diagramov špecializovaných na prípad použitia topologickej a pravdepodobnostnej analýzy spoľahlivosti:
  - ✓ práca predstavila zovšeobecnenú verziu algoritmu na dynamickú tvorbu rozhodovacích diagramov;
  - ✓ nakoniec predstavila nový univerzálny algoritmus na efektívny výpočet ľubovoľných logických derivácií.

Záverom možno konštatovať, že hlavným prínosom tejto práce je opis optimalizácie tvorby a manipulácie s rozhodovacími diagramami pre analýzu spoľahlivosti rozsiahlych komplexných systémov. Opis zahŕňa existujúce techniky a algoritmy, ako aj nové algoritmy navrhnuté v tejto práci. Napokon, významným praktickým prínosom je softvérová knižnica s otvoreným zdrojovým kódom špecializovaná na analýzu spoľahlivosti pomocou rozhodovacích diagramov.

Zoznam použitej literatúry

[1] E. Zio, *An Introduction to the Basics of Reliability and Risk Analysis*. World Scientific Publishing Company, 2007. doi: 10.1142/6442.

[2] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Comput.*, vol. 35, no. 8, pp. 677–691, Aug. 1986, doi: 10.1109/TC.1986.1676819.

[3] A. Srinivasan, T. Ham, S. Malik, and R. K. Brayton, "Algorithms for discrete function manipulation," in *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, 1990, pp. 92–95. doi: 10.1109/ICCAD.1990.129849.

[4] M. Rausand and A. Høyland, *System Reliability Theory, 2nd ed*. Hoboken, NJ: John Wiley & Sons, Inc., 2004.

[5] B. Natvig, *Multistate Systems Reliability Theory with Applications*. Chichester, UK: John Wiley & Sons, Inc., 2011. doi: 10.1002/9780470977088.

[6] A. Lisnianski and G. Levitin, *Multi-State System Reliability: Assessment, Optimization and Application*. Singapore, SG: World Scientific Publishing Company, 2003. doi: 10.1142/5221.

[7] S. Yanushkevich, D. Miller, V. Shmerko, and R. Stankovic, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*. Boca Raton, FL: CRC Press, 2006. doi: 10.1201/9781420037586.

[8] Y. Crama and P. Hammer, *Boolean Functions: Theory, Algorithms, and Applications*. 2011. doi: 10.1017/CBO9780511852008.

[9] Steinbach Bernd and Posthoff Christian, *Boolean Differential Calculus*. Morgan & Claypool, 2017. doi: 10.2200/S00766ED1V01Y201704DCS052.

[10] M. Kvassay, E. Zaitseva, and V. Levashenko, "Importance analysis of multi-state systems based on tools of logical differential calculus," *Reliab Eng Syst Saf*, vol. 165, pp. 302–316, 2017, doi: https://doi.org/10.1016/j.ress.2017.03.021.

[11] M. Kvassay and E. Zaitseva, "Topological analysis of multi-state systems based on direct partial logic derivatives," *Springer Series in Reliability Engineering*, vol. 0, no. 9783319634227, pp. 265 – 281, 2018, doi: 10.1007/978-3-319-63423-4_14.

[12] W. Kuo and X. Zhu, "Importance Measures in Reliability, Risk, and Optimization: Principles and Applications," *Importance Measures in Reliability, Risk, and Optimization: Principles and Applications*, Apr. 2012, doi: 10.1002/9781118314593.

[13] E. Zaitseva and L. Vitaly, "Importance analysis by logical differential calculus," *Automation and Remote Control*, vol. 74, 2013, doi: 10.1134/S000511791302001X.

[14] T. Krkoška, "Softvérová knižnica pre manipuláciu s binárnymi rozhodovacími diagramami," University of Žilina, 2019.

[15] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Transactions of the American Institute of Electrical Engineers*, vol. 57, no. 12, pp. 713–723, 1938, doi: 10.1109/T-AIEE.1938.5057767.

[16] M. Mrena, M. Kvassay, and S. Stankevich, "Dynamic Binary Decision Diagram Creation Using an Extended Apply Algorithm," in *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 2023, pp. 513–518. doi: 10.1109/IDAACS58523.2023.10348726.

[17] M. Mrena, P. Sedlacek, and M. Kvassay, "Linear Fold and Tree Fold in Creation of Binary Decision Diagrams of Standard Benchmarks," in *Proceedings of the 11th IEEE*

International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, 2021. doi: 10.1109/IDAACS53288.2021.9660940.

[18] M. Mrena and M. Kvassay, "Comparison of Left Fold and Tree Fold Strategies in Creation of Binary Decision Diagrams," in *International Conference on Information and Digital Technologies 2021, IDT 2021*, 2021. doi: 10.1109/IDT52577.2021.9497593.

[19] L. Xing and Y. S. Dai, "A new decision-diagram-based method for efficient analysis on multistate systems," *IEEE Trans Dependable Secure Comput*, vol. 6, no. 3, pp. 161–174, 2009, doi: 10.1109/TDSC.2007.70244.

[20] M. Kvassay, E. Zaitseva, V. Levashenko, and J. Kostolny, "Minimal cut vectors and logical differential calculus," *Proceedings of The International Symposium on Multiple-Valued Logic*, pp. 167–172, 2014, doi: 10.1109/ISMVL.2014.37.

[21] M. Mrena and M. Kvassay, "Efficient Computation of Logic Derivatives Using Multi-valued Decision Diagrams," in *ISMVL 2024 IEEE International Symposium on Multiple-Valued Logic*, 2024, p. to appear.

[22] S. Nagayama, A. Mishchenko, T. Sasao, and J. Butler, "Exact and heuristic minimization of the average path length in decision diagrams," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 11, Aug. 2005.

[23] M. Mrena, M. Kvassay, and E. Zaitseva, "TeDDy: Templated Decision Diagram Library," *SoftwareX*, p. to appear, 2024.

[24] C. Bauer, A. Frink, and R. Kreckel, "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language," *J Symb Comput*, vol. 33, no. 1, pp. 1–12, 2002, doi: https://doi.org/10.1006/jsco.2001.0494.

Zoznam publikovaných prác

1. M. Mrena and M. Kvassay, “Comparison of left fold and tree fold strategies in creation of binary decision diagrams,” in *2021 International Conference on Information and Digital Technologies (IDT)*, 2021, pp. 341–352.
2. M. Mrena, P. Sedlacek, and M. Mrena, “Linear fold and tree fold in creation of binary decision diagrams of standard benchmarks,” in *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, 2021, pp. 1120–1125.
3. P. Rusnak and M. Mrena, “Time dependent reliability analysis of the data storage system based on the structure function and logic differential calculus,” in *Reliability Engineering and Computational Intelligence*, C. van Gulijk and E. Zaitseva, Eds. Cham: Springer International Publishing, 2021, pp. 179–198
4. M. Mrena, M. Kvassay, and S. Czapp, “Single and series of multi-valued decision diagrams in representation of structure function,” in *Lecture Notes in Networks and Systems, vol. 484 LNNS*, 2022, pp. 176–185.
5. M. Mrena and M. Kvassay, “Experimental analysis of decision diagrams used to represent structure functions of series-parallel multi-state systems,” in *Proceedings of the 32nd European Safety and Reliability Conference (ESREL 2022)*, 2022, pp. 1707–1714.
6. M. Mrena, M. Varga, and M. Kvassay, “Experimental comparison of array-based and linked-based list implementations,” in *2022 IEEE 16th International Scientific Conference on Informatics (Informatics)*, 2022, pp. 231–238.
7. M. Mrena and M. Kvassay, “Comparison of single MDD and series of MDDs in the representation of structure function of series-parallel MSS,” in *2022 IEEE 16th International Scientific Conference on Informatics (Informatics)*, 2022, pp. 225–230.
8. P. Galcik, M. Mrena, L. Piatrikova, and S. Stankevich, “Advanced priority queues in the optics clustering algorithm,” in *2023 International Conference on Information and Digital Technologies (IDT)*, 2023, pp. 257–266.
9. M. Mrena and M. Kvassay, “Experimental survey of algorithms for the calculation of node traversal probabilities in multi-valued decision diagrams,” in *Reliability Engineering and Computational Intelligence for Complex Systems: Design, Analysis and Evaluation*, C. van Gulijk, E. Zaitseva, and M. Kvassay, Eds. Cham: Springer Nature Switzerland, 2023, pp. 3–20.
10. M. Mrena, M. Kvassay, and S. Stankevich, “Dynamic binary decision diagram creation using an extended apply algorithm,” in *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, 2023, pp. 513–518.
11. M. Mrena and M. Kvassay, “Generating monotone Boolean functions using Hasse diagram,” in *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, 2023, pp. 793–797.
12. M. Mrena and M. Kvassay, “Efficient computation of logic derivatives using multi-valued decision diagrams,” in *ISMVL 2024 IEEE International Symposium on Multiple-Valued Logic*, 2024, to appear.
13. M. Mrena, M. Kvassay, and E. Zaitseva, “Teddy: Templated decision diagram library,” *SoftwareX*, to appear, 2024.