

Evoluční syntéza elektronických obvodů

Evolutionary synthesis of electronic circuits

Bc. Martin Jašek

Diplomová práce
2012



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2011/2012

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin JAŠEK**
Osobní číslo: **A10315**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**

Téma práce: **Evoluční syntéza elektronických obvodů**

Zásady pro vypracování:

1. Vypracujte literární rešerši zaměřenou na dané téma.
2. Vytvořte návrhový systém na bázi evolučních algoritmů, který bude umět navrhovat elektronické obvody metodou symbolické regrese.
3. Systém naprogramujte v prostředí Mathematica.
4. Závěrem zhodnoťte výsledky.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. ZELINKA, Ivan. Evoluční výpočetní techniky: Principy a aplikace. 1. vyd. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3 (váz.).
2. OPLATKOVÁ, Zuzana. Analytic programming. Zlín, 2003. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Vedoucí práce prof. Ing. Ivan Zelinka, Ph.D.
3. HYNEK, Josef. Genetické algoritmy a genetické programování. 1. vyd. Praha: Grada, 2008. ISBN 978-80-247-2695-3 (brož.).
4. MILLER, Julian F. Cartesian Genetic Programming. 1st Edition. Heidelberg: Springer, 2011. ISBN 978-3-642-17309-7.
5. TVRDÍK, Josef. Evoluční algoritmy. Ostrava, 2004. Dostupné z: http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf. Učební text. Ostravská univerzita.
6. SHIFRIN, Leonid. Mathematica programming: an advanced introduction. Version 1.01. San Francisco, 2008. Dostupné z: <http://www.mathprogramming-intro.org/download/MathProgrammingIntro.pdf>.

Vedoucí diplomové práce:

prof. Ing. Ivan Zelinka, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

24. února 2012

Termín odevzdání diplomové práce:

15. května 2012

Ve Zlíně dne 24. února 2012



prof. Ing. Vladimír Vašek, CSc.

děkan



doc. RNDr. Vojtěch Křesálek, CSc.

ředitel ústavu

ABSTRAKT

Tato práce se zabývá způsoby návrhu elektronických, především kombinačních, obvodů se zaměřením na evoluční techniky. Praktickým výstupem je vlastní návrhový systém realizovaný pomocí prostředí Wolfram Mathematica s využitím kartézského genetického programování CGP.

V teoretické části se nachází studie kombinačních obvodů, programovacího prostředí Wolfram Mathematica a stávajících evolučních i vybraných neevolučních metod. Praktická část seznamuje s vytvořeným návrhovým systémem CGP synthesis, jeho uživatelským prostředím, zdrojovým kódem a také se skutečnými výsledky tohoto systému.

Klíčová slova: elektronické obvody, kombinační obvody, CGP, evoluce, Mathematica, CGP synthesis

ABSTRACT

This thesis deals with electronic, mainly combinatorial, circuit synthesis with a specialization in evolutionary techniques. As a thesis practical output my design system realized using Wolfram Mathematica environment and Cartesian genetic programming was created.

In a theoretical part, there is a combinatorial circuits, Wolfram Mathematica environment, existing evolutionary and also non-evolutionary methods study. A practical part documents the created design system called CGP synthesis, its user interface, source code and real results.

Keywords: electronic circuits, combinatorial circuits, CGP, evolution, Mathematica, CGP synthesis

Rád bych tímto poděkoval svému vedoucímu diplomové práce, prof. Ing. Ivanu Zelinkovi, Ph.D., za cenné rady a připomínkování během vytváření návrhového systému, uvedeném v praktické části této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	8
I TEORETICKÁ ČÁST	10
1 VYBRANÉ ELEKTRONICKÉ OBVODY	11
1.1 KOMBINAČNÍ LOGICKÉ OBVODY	11
1.2 LOGICKÁ HRADLA	12
2 SYNTÉZA ELEKTRONICKÝCH OBVODŮ	14
2.1 KONVENČNÍ EXAKTNÍ ŘEŠENÍ: QUINE–MCCLUSKEY.....	15
2.2 KONVENČNÍ HEURISTICKÉ ŘEŠENÍ: ESPRESSO	18
2.3 EVOLUČNÍ ŘEŠENÍ	22
3 EVOLUČNÍ TECHNIKY	23
3.1 ZÁKLADNÍ POJMY	23
3.2 SYMBOLICKÁ REGRESE.....	24
3.3 GENETICKÉ PROGRAMOVÁNÍ.....	24
3.4 KARTÉZSKÉ GENETICKÉ PROGRAMOVÁNÍ CGP.....	28
4 WOLFRAM MATHEMATICA	30
4.1 ÚVOD DO PROSTŘEDÍ.....	30
4.2 VYBRANÉ FUNKCE.....	31
4.3 MATHEMATICA VS. CDF PLAYER VS. PLAYER PRO.....	33
5 STÁVAJÍCÍ NÁVRHOVÝ SYSTÉM: TOOLS4CGP	35
5.1 MODUL PRO NAČTENÍ VSTUPNÍCH DAT	35
5.2 IMPLEMENTACE CGP	36
5.3 PROHLÍZEČ VÝSLEDKŮ CGPVIEWER	37
II PRAKTICKÁ ČÁST	38
6 VYTVOŘENÝ NÁVRHOVÝ SYSTÉM: CGP SYNTHESIS	39
6.1 ÚVODNÍ SEZNÁMENÍ.....	39
6.2 UŽIVATELSKÉ ROZHRAŇÍ	41
7 NÁHLED DO ZDROJOVÉHO KÓDU	47
7.1 GENEROVÁNÍ JEDINCE	47
7.2 VYTVOŘENÍ ROVNICE Z JEDINCE	48
7.3 OHODNOCENÍ POPULACE	48
7.4 VIZUALIZACE ZAPOJENÍ.....	48
8 SROVNÁNÍ A ZHODNOCENÍ VÝSLEDKŮ	49
8.1 VÝSLEDKY EXPERIMENTŮ – METODA 10 POKUSŮ.....	49
8.2 VÝSLEDKY EXPERIMENTŮ – METODA 10 VÝSLEDKŮ S FITNESS 0	51
8.3 ZHODNOCENÍ VÝSLEDKŮ SYSTÉMU CGP SYNTHESIS.....	53
ZÁVĚR	54
ZÁVĚR V ANGLIČTINĚ	55
SEZNAM POUŽITÉ LITERATURY	56
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	58
SEZNAM OBRÁZKŮ	59
SEZNAM TABULEK	61
SEZNAM PŘÍLOH	62

ÚVOD

Téma diplomové práce, které mi vedoucí po předchozí spolupráci nabídl, jsem přijal především z důvodu zajímavě vyhlížejících principů evolučních algoritmů a také možnosti pokračovat ve tvůrčí činnosti ve funkčně bohatém a uživatelsky příjemném programovacím prostředí Wolfram Mathematica.

Evoluční principy a algoritmy se v dnešní době čím dál více objevují v odborných textech a aplikacích určených mnoha oborům lidské činnosti (průmysl, doprava, logistika, matematika, elektronika, stavebnictví apod.). Mezi známější evolučně řešené problémy patří např. *obchodní cestující* (určení trasy s co největší úsporou prostředků), *návrh družicové antény a přítlačného křídla automobilu* (s co nejlepšími aerodynamickými vlastnostmi). Evoluční algoritmy tak lze jistě označit za značně multioborové a přitom principiálně jednoduché s často nezanedbatelně lepšími výsledky než s dříve běžně využívanými technikami.

Syntéza kombinačních obvodů se velmi efektivně realizuje pomocí kartézského genetického programování (CGP) vlivem charakteru interpretace hledaného výsledku formou maticového zobrazení. I přes tuto skutečnost funkčních ucelených systémů s podporou výše uvedeného mnoho není, resp. podařilo se mi nalézt pouze jeden (zmiňuji se o něm v kapitole 5).

Hlavním cílem mé práce je **vytvořit funkční systém** pro návrh kombinačních obvodů pomocí implementace CGP, **který by však zahrnoval přehledné vložení vstupních parametrů a grafické vyjádření nalezeného řešení** (v ideálním případě formou obvodového schématu) **v rámci jediné spuštěné aplikace naprogramované v prostředí Wolfram Mathematica**. Jako dílčí cíle bych uvedl vypracování stručné studie stávajícího řešení, seznámení s ovládáním a způsobem získání výsledků vytvořeného systému, zhodnocení výkonnosti a funkčnosti na vybraných úlohách.

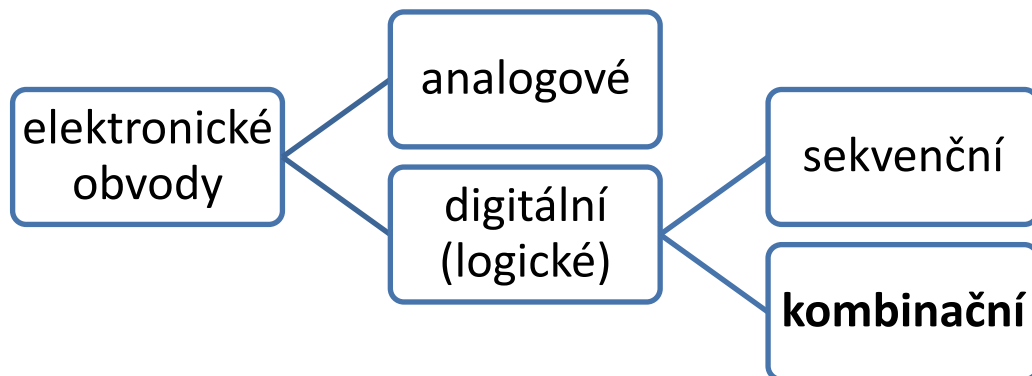
V teoretické části nejdříve vymezím pojem *elektronické obvody* na specifickou zkoumanou oblast, následně vyhotovím pojednání o způsobech navrhování elektronických obvodů. Zde není záměrem zabývat se všemi dostupnými technikami, blíže seznámím pouze se zástupci z daných neevolučních kategorií s pozdějším zaměřením na evoluční. Další kapitoly přiblíží software Wolfram Mathematica, vč. ukázky některých použitých funkcí, a stávající systém *TOOLS4CGP*.

Praktická část již výhradně zdokumentuje vlastní vytvořený systém *CGP synthesis*. Na počátku provede postupem pro správné načtení zdrojového kódu, vložení požadovaných parametrů hledaného problému i samotné evoluce, spuštění výpočetního programu. Poté obeznámí s výstupními daty – řádkovým i grafickým tvarem hledaného řešení. V další kapitole krátce uvedu do stěžejních sekcí zdrojového kódu. Na závěr zveřejním výsledky provedených experimentů za účelem zhodnocení funkčnosti systému.

I. TEORETICKÁ ČÁST

1 VYBRANÉ ELEKTRONICKÉ OBVODY

Oblast elektronických obvodů je velmi široká, tato práce se zaměřuje pouze na její specifickou část – **kombinační logické obvody**.

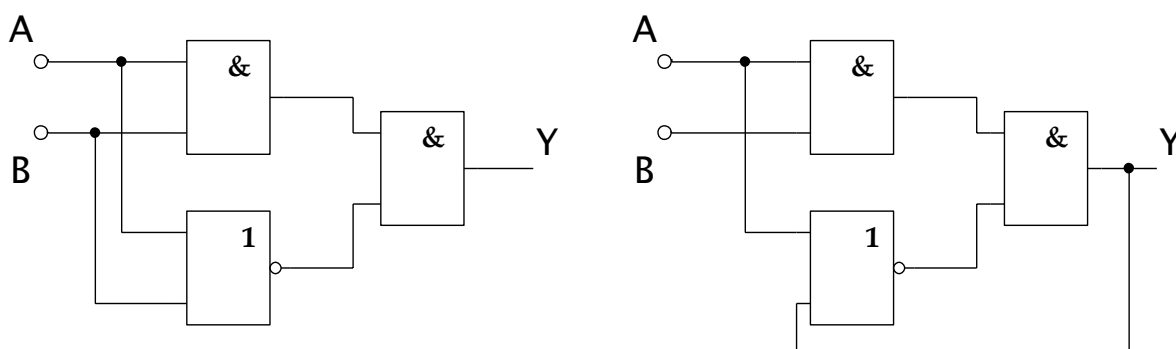


Obr. 1. Systemizace kombinačních logických obvodů.

1.1 Kombinační logické obvody

Logické (číslicové, digitální) obvody pracují s **diskrétními hodnotami**, určitými napěťovými hladinami. Zpravidla se používají dvě hladiny – hodnota blízká 0 V pro vyjádření logické úrovně *Low* (0) a hodnota blízká napájecímu napětí pro vyjádření logické úrovně *High* (1). Tím se liší od analogových obvodů pracujících se spojitými hodnotami.

Stav výstupů *kombinačních* logických obvodů **závisí pouze na okamžitých stavech vstupů**, neprojevuje se zde žádný paměťový efekt (na rozdíl od sekvenčních obvodů).



Obr. 2. Principiální rozdíl mezi kombinačním (vlevo) a sekvenčním (vpravo) obvodem.

Jak uvádí RAM¹, kombinační obvody jsou logické obvody, jejichž činnost může být **zcela popsána pravdivostní tabulkou**, realizují se pomocí hradel AND, OR, NAND, NOR. Mezi kombinační obvody patří např. sčítačky, komparátory, (de)multiplexery, (de)kodéry, paměti ROM.

1.2 Logická hradla

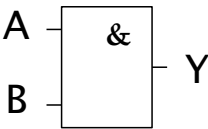
Jedná se o základní prvky logických obvodů. Každé hradlo má určitý počet vstupů (dle typu zpravidla jeden až tři, pro více vstupů využíváme *hradlové sítě*²) a jeden výstup závislý na vstupech. Tento vztah můžeme popsat logickou operací (součet, součin, negace apod.).

Postupem času vznikala různá doporučení pro symbolické zakreslování – české ČSN, německé DIN, americké ANSI, **mezinárodní IEC**. V této práci využívám částečně značení podle IEC.

Dříve se hradla realizovala pomocí elektronek, diod, tranzistorů. V dnešní době k tomu stačí jeden **integrováný obvod**.

Logický součin (konjunkce) – logická funkce AND

Tab. 1. Specifikace logického součinu.

matematická formulace	pravdivostní tabulka	značení IEC	symbolika v Mathematice															
$Y = A \cdot B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1		$Y = \text{And}[A, B]$ $Y = A \ \&\& \ B$ $Y = A \wedge B$
A	B	Y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																

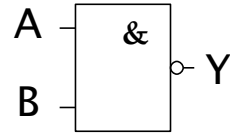
¹ LOGIC DESIGN: **Combinational and Sequential Circuits**. RAM, B. *Computer fundamentals: architecture and organization*. Rev. 3rd ed. New Delhi: New Age International, 2000. ISBN 812241267X.

² **Hradlová síť**: propojení hradel stejného typu za účelem navýšení vstupů, více zde: MAREŠ, Martin. 5. Hradlové sítě. Praha, 2011. Dostupné z: <http://mj.ucw.cz/vyuka/1112/ads2/5-hradla.pdf>

Negovaný logický součin (negace konjunkce) – logická funkce NAND

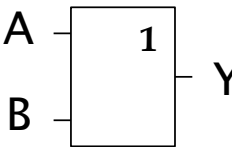
Nejčastěji používané, univerzální pro jakoukoliv funkci.

Tab. 2. Specifikace negovaného logického součinu.

matematická formulace	pravdivostní tabulka	značení IEC	symbolika v Mathematice															
$Y = \overline{A \cdot B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0		$Y = \text{Nand}[A, B]$ $Y = A \bar{\wedge} B$
A	B	Y																
0	0	1																
0	1	1																
1	0	1																
1	1	0																

Logický součet (disjunkce) – logická funkce OR

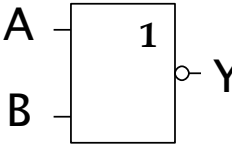
Tab. 3. Specifikace logického součtu.

matematická formulace	pravdivostní tabulka	značení IEC	symbolika v Mathematice															
$Y = A + B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1		$Y = \text{Or}[A, B]$ $Y = A B$ $Y = A \vee B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																

Negovaný logický součet (negace disjunkce) – logická funkce NOR

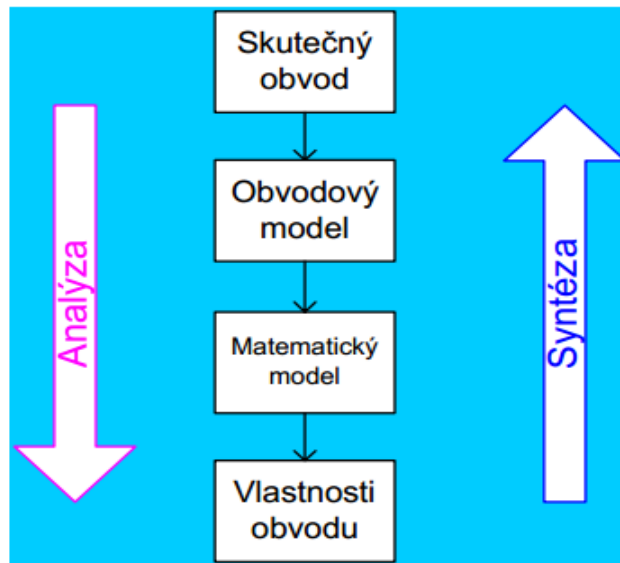
Stejně jako NAND univerzální pro jakoukoliv funkci.

Tab. 4. Specifikace negovaného logického součtu.

matematická formulace	pravdivostní tabulka	značení IEC	symbolika v Mathematice															
$Y = \overline{A + B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0		$Y = \text{Nor}[A, B]$ $Y = A \bar{\vee} B$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	0																

2 SYNTÉZA ELEKTRONICKÝCH OBVODŮ

Potřebujeme-li navrhnout vhodný elektronický obvod splňující požadované vlastnosti, které zadáme *booleovskou funkcí*³, využijeme metody zvané **syntéza elektronických (kombinačních) obvodů**. Zjednodušený algoritmus syntézy je uveden na *obr.3*, kde lze rovněž spatřit rozdíl mezi pojmy *analýza* a *syntéza* ve vztahu ke zkoumané problematice.



Obr. 3. DOSTÁL, Tomáš. Řešení obvodů. In:⁴

Způsobů, kterými můžeme syntézu realizovat, je více. Různé pohledy na jejich klasifikaci nabízí např. SEKANINA⁵, DE MICHELI⁶ a BÍLEK⁷ (*obr.4*).

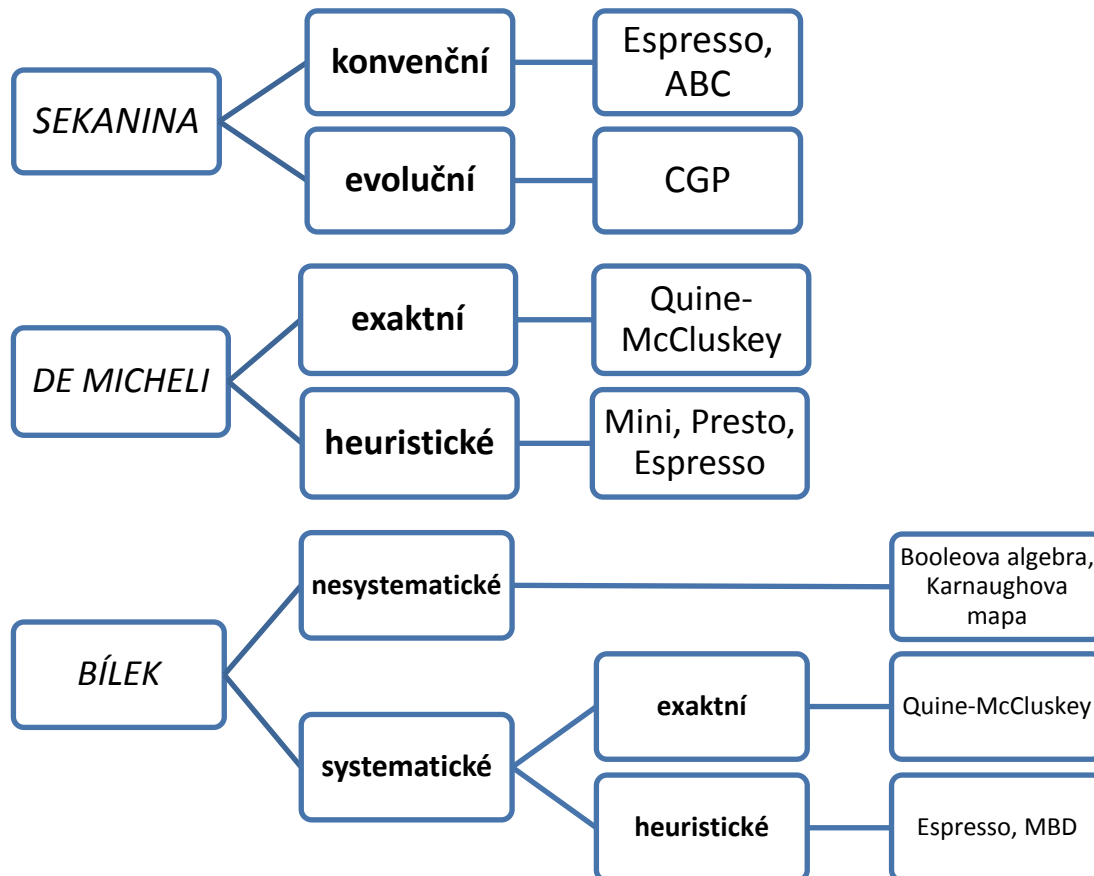
³ **Booleovská funkce:** Závislost výstupních hodnot na vstupních vyjádřená logickou hodnotou pro každou kombinaci vstupů. V pravdivostní tabulce zpravidla sloupec „Y“.

⁴ DOSTÁL, Tomáš. *Analogové elektronické obvody*. Brno, 2004.
Dostupné z: http://dalkove2008-2013.wz.cz/ek_vut_brno.pdf. Učební text. VUT Brno.

⁵ SEKANINA, Lukáš. **Evoluční návrh elektronických obvodů**.
Automa: časopis pro automatizační techniku. Praha: FCC Public, 2010, č. 1. ISSN 1210-9592.
Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=40365

⁶ DE MICHELI, Giovanni. *Two-level Logic Synthesis and Optimization* [online]. Lausanne [cit. 2012-03-12]. Dostupné z: <http://www.ece.iupui.edu/~johnlee/ECE495/Two-Level.Exact.Minimization.ppt>. EPF.

⁷ BÍLEK, Jan. *Minimalizace neúplně určených logických funkcí pomocí modifikovaných binárních rozhodovacích diagramů*. Praha, 2007.
Dostupné z: <http://service.felk.cvut.cz/vlsi/dip/Bilek07/dp-mbd.pdf>. Diplomová práce. ČVUT, FEL.



Obr. 4. Grafická konfrontace pohledů na klasifikaci metod syntézy.

Výsledkem většiny těchto metod je **minimalizovaná logická rovnice**, jediné CGP pracuje na úrovni obvodového modelu. Není cílem této práce zabývat se všemi výše uvedenými metodami. V následujících podkapitolách uvádím na ukázkou zástupce z konvenčních metod a hlubší studii evolučních metod, promítá se zde pohled *SEKANINY*⁵ a *DE MICHELIHO*⁶.

2.1 Konvenční exaktní řešení: Quine–McCluskey

Podle NIEDOBY⁸ se **nejedná o plně minimalizační techniku**, jelikož je nutné následně využít metodu zvanou *mřížka prostých implikantů*, která výsledek ještě více zminimalizuje.

⁸ NIEDOBA, Pavel. *Minimalizace Booleových funkcí pomocí Quineovy-McCluskeyovy metody*. Brno, 2010. Dostupné z: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=28728. Bakalářská práce. VUT Brno, Fakulta strojního inženýrství.

Předchozí tvrzení potvrzuje ŠEDA⁹, podle kterého samotné Quine-McCluskey **nedokáže garantovat nalezení optimálního řešení**, pro složitější problém doporučuje využít heuristické metody.

NIEDOBA⁸ svou práci pojal velmi odborně, domnívám se však, že méně zasvěcený čtenář bude potřebovat poněkud více času na pochopení zmíněné metody, jejíž postup podrobně popisuje. V následujících řádcích **vysvětlím příklad**, který NIEDOBA⁸ použil, jednodušší formou.

vstupní funkce:
$$F = \bar{x}z + \bar{x}\bar{z} + x\bar{y}\bar{z} \quad (1)$$

převod na *úplný disjunktí tvar*¹⁰:

- vyhledáme v tabulce se všemi kombinacemi proměnných příslušné řádky, ve kterých se nalézají jednotlivé členy vstupní funkce (1)
- celý řádek opišeme (2)
- nalézá-li se kombinace ve více řádcích, opišeme všechny zúčastněné

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

úplný disjunktí tvar:
$$F = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}\bar{z} \quad (2)$$

úplný disjunktí tvar podle NIEDOBY⁸:

$$F = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}\bar{z} \quad (3)$$

Srovnáním (2) a (3) docházím k závěru, že NIEDOBA⁸ se zmýlil v posledním členu, kde opomněl negaci proměnné z. Stanovisko jsem si rovněž ověřil pomocí software *Wolfram Mathematica*, kde jsem nadeřinoval výše uvedené tvary vstupní funkce (1) (2) (3) a dále studoval výstupní hodnotu v závislosti na různých kombinacích proměnných. Výsledek předkládám na *obr.5*.

⁹ ŠEDA, Miloš. **Heuristic Set-Covering-Based Postprocessing for Improving the Quine-McCluskey Method**. In: *International Journal of Computational Intelligence Enformatika* [online]. 2007 [cit. 2012-03-13]. ISSN 1304-2386. Dostupné z: <http://www.waset.org/journals/waset/v29/v29-47.pdf>

¹⁰ **úplný disjunktí tvar:** součet součinů obsahujících všechny vstupní proměnné

```
In[34]:= Fvstup[True, False, False]
          Fjasek[True, False, False]
          Fniedoba[True, False, False]

Out[34]= True

Out[35]= True

Out[36]= False
```

Obr. 5. Ověření svého stanoviska.

převod do binárního zápisu:

$$F = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}\bar{z} \quad (4)$$

$$000 \quad 001 \quad 010 \quad 011 \quad 100 \quad (5)$$

soupis s dekadickými hodnotami sloužícími dále jako indexy:

0		0	0	0
1		0	0	1
2		0	1	0
3		0	1	1
4		1	0	0

(6)

nyní se aplikuje metoda „pokus-omyl“ za účelem snížení počtu řádků s těmito pravidly:

- řádky, které se liší pouze v jednom bitu, můžeme pokrátit a místo odlišných hodnot dosadíme „-“ symbolizující vykrácení jedné proměnné (např. z řádků 0 a 1 v (6) může vzniknout jediný řádek 0,1s údaji 0 0 -)
- jednotlivé řádky lze využívat pro konfrontaci s jiným řádkem opakovaně
- ze zkušenosti doporučuji vypsát si veškeré možné variace krácení (7)
- objeví-li se dva řádky se shodnými údaji, nadbytečné odstraníme – **náš cíl**

redukování řádků:

0,1		0	0	-
0,2		0	-	0
2,3		0	1	-
0,4		-	0	0
1,3		0	-	1

0,1,2,3		0	-	-
0,2,1,3		0	-	-
0,4		-	0	0

(7)

výsledek:

0,2,1,3		0	-	-
0,4		-	0	0

(8)

Zbylé řádky (8) převedeme zpět na proměnné (9).

$$F = \bar{x} + \bar{y}\bar{z} \quad (9)$$

Výsledek jsem opět ověřil softwarem *Wolfram Mathematica*. Výstupní hodnoty se **shodují** s hodnotami na výstupu rovnice (4). Výsledek (9) již patrně další úpravy nepotřebuje, čímž částečně vyvracím tvrzení NIEDOBY⁸ a přikláním se k výroku ŠEDÉHO⁹.

2.2 Konvenční heuristické řešení: Espresso

Espresso bývá označováno jako **nejúspěšnější heuristická** metoda na minimalizaci, což potvrzuje BÍLEK¹¹ a dodává, že jako každá heuristika nehledá cíleně nejlepší řešení, spokojí se s vyhovujícím. Na druhou stranu se podle ZELENKY¹² navýšila rychlost minimalizace.

BÍLEK¹¹ rovněž zmiňuje důležitý fakt:

„Espresso je určeno pro počítačové zpracování, a tak není příliš vhodné pro manuální minimalizaci.“

Podrobně se touto metodou zabývají MEUER A BRÜCK¹³, jejich studii částečně přiblížím v následujících řádcích.

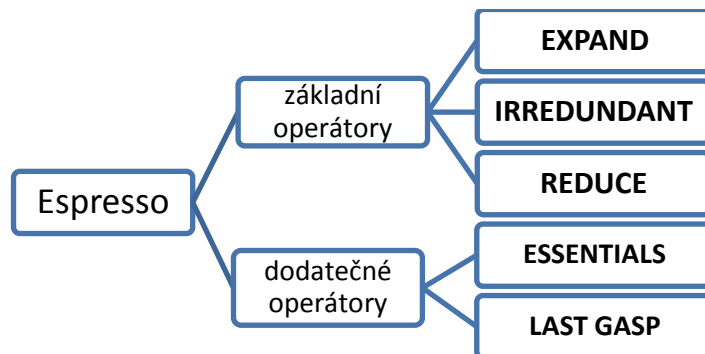
¹¹ BÍLEK, Jan. *Minimalizace neúplně určených logických funkcí pomocí modifikovaných binárních rozhodovacích diagramů*. Praha, 2007.

Dostupné z: <http://service.felk.cvut.cz/vlsi/dip/Bilek07/dp-mbd.pdf>. Diplomová práce. ČVUT, FEL.

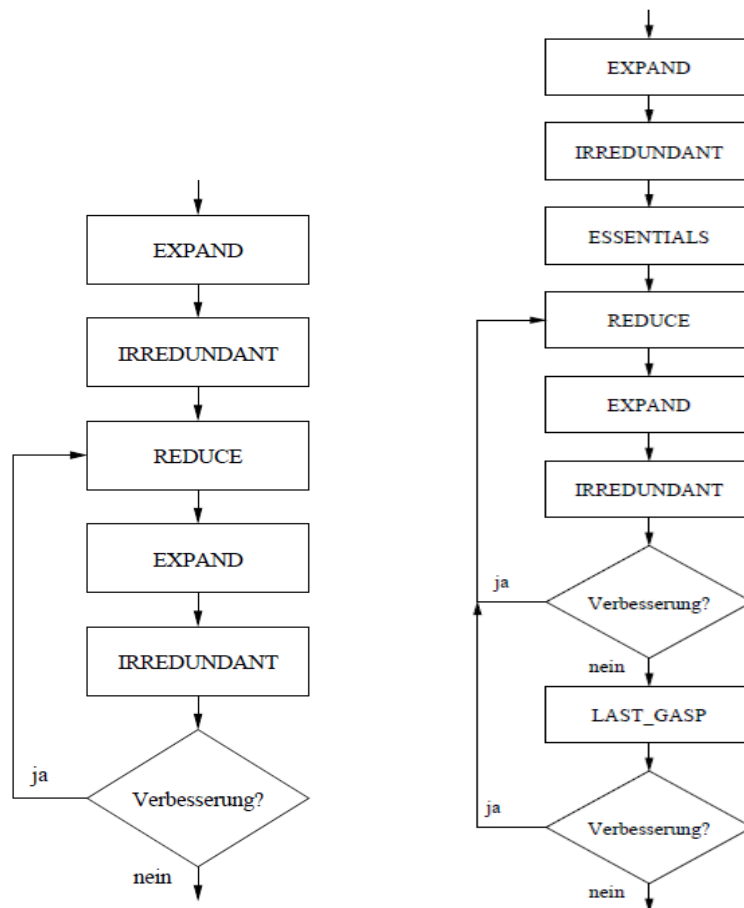
¹² ZELENKA, Jakub. *Vizualizace dvouúrovňové minimalizace logických funkcí*. Praha, 2010. Dostupné z: <https://service.felk.cvut.cz/vlsi/dip/Zelenka10/text/dp.pdf>. Diplomová práce. ČVUT v Praze, Fakulta elektrotechnická.

¹³ MEURER, Benedikt a BRÜCK. *Grundlagen und Verfahren für den Mikrosystementwurf* [online]. Siegen, 2008 [cit. 2012-03-14]. Dostupné z: <http://benediktmeurer.de/files/mse2008.pdf>. Universität Siegen.

Algoritmus Espresso se skládá z **dílčích operátorů** se specifickou úlohou (obr.6.) a zpravidla se vyjadřuje vývojovým diagramem (obr.7.).



Obr. 6. Operátory algoritmu Espresso.



Obr. 7. MEURER, Benedikt a BRÜCK. Der ESPRESSO Algorithmus (vlevo) a Der erweiterte¹⁴ ESPRESSO Algorithmus (vpravo). In.¹³

¹⁴ **erweiterte**: v překladu z německého jazyka do českého znamená „pokročilý“
verbesserung: v překladu z německého jazyka do českého znamená „zlepšení“

EXPAND

Operátor EXPAND pracuje na principu **redukování proměnných**.

před

	A	B		
C	0	0	0	0
C	0	1	1	0

$$Y = \bar{A}\bar{B}C + A\bar{B}C$$

po

	A	B		
C	0	0	0	0
C	0	1 1		0

$$Y = \bar{B}C$$

Obr. 8. Princip operátoru EXPAND.

IRREDUNDANT

Operátor IRREDUNDANT pracuje na principu **redukování nadbytečných členů**.

před

	A	B		
C	0	0	1	1
C	1	1	1	0

$$Y = \bar{A}C + \bar{B}C + A\bar{B} + A\bar{C}$$

po

	A	B		
C	0	0	1	1
C	1	1	1	0

$$Y = \bar{A}C + A\bar{B} + A\bar{C}$$

Obr. 9. Princip operátoru IRREDUNDANT.

REDUCE

Operátor REDUCE pracuje na principu **přidávání proměnných**.

před

	A	B		
C	0	0	0	0
C	0	1	1	0

$Y = \bar{B}C$

po

	A	B		
C	0	0	0	0
C	0	1	1	0

$Y = \bar{A}\bar{B}C + A\bar{B}C$

Obr. 10. Princip operátoru REDUCE.

ESSENTIALS

Operátor ESSENTIALS pracuje na principu **vyhledávání všech možných řešení**.

varianta 1

	A	B		
C	0	0	1	1
C	1	1	1	0

$Y = \bar{A}C + \bar{B}C + A\bar{C}$

varianta 2

	A	B		
C	0	0	1	1
C	1	1	1	0

$Y = \bar{A}C + A\bar{B} + A\bar{C}$

Obr. 11. Princip operátoru ESSENTIALS.

LAST GASP

Operátor LAST GASP hlídá běh algoritmu, zabraňuje vzniku horších výsledků.

2.3 Evoluční řešení

Evoluční algoritmy, využívající modelů evolučních procesů, jsou vhodné pro hledání **řešení náročných a rozsáhlých úloh**. Nalézají tak uplatnění v široké škále oborů. BURIAN¹⁵ doporučuje využít těchto metod při návrhu obvodů pro následnou implementaci bez nutnosti dynamické změny jeho parametrů, dále říká:

*„Aby mělo použití evolučního návrhu smysl, výsledné obvodové řešení **musí mít přínos oproti standardním návrhovým technikám**, např. již zmíněnou úsporu hardwarových prostředků nebo lepší chování z hlediska časování.“*

SEKANINA¹⁶ dodává, že evolucí lze získat výsledky, kterých konvenčními metodami nedosáhneme. Nevýhody vidí především ve vysoké výpočetní náročnosti, **negarantování dostatečně kvalitního řešení**, často těžší interpretaci výsledku.

Evolučních algoritmů známe více druhů, srovnání publikoval ZELINKA a kol.¹⁷ Vybraným se blíže zabývá následující kapitola.

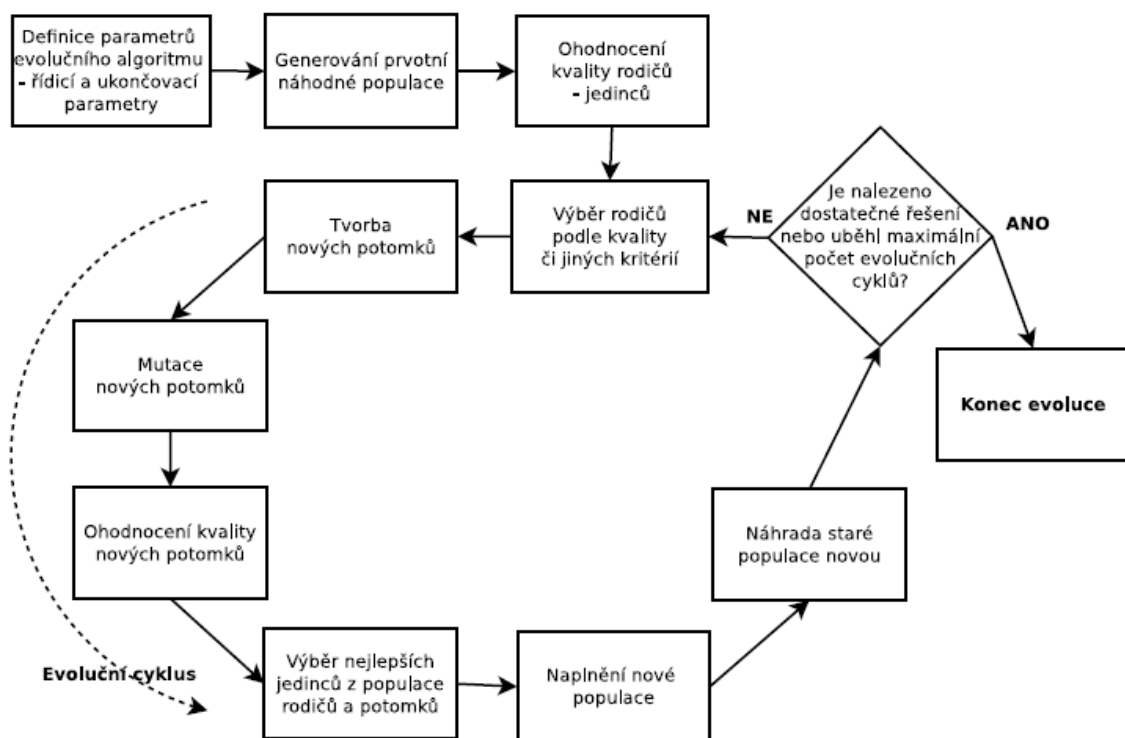
¹⁵ BURIAN, Petr. **Návrh číslicových obvodů za pomoci evolučních výpočetních technik.** *Automatizace*. Praha: Automatizace, 2009, č. 3. ISSN 0005-125X.
Dostupné z: <http://www.automatizace.cz/article.php?a=2484>

¹⁶ SEKANINA, Lukáš. **Evoluční návrh elektronických obvodů.** *Automa: časopis pro automatizační techniku*. Praha: FCC Public, 2010, č. 1. ISSN 1210-9592.
Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=40365

¹⁷ ZELINKA, Ivan, Zuzana OPLATKOVÁ, Miloš ŠEDA, Pavel OŠMERA a František VČELAŘ. **Evoluční výpočetní techniky: principy a aplikace.** 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.

3 EVOLUČNÍ TECHNIKY

Evoluční algoritmy, klíčový prvek evolučních technik, jsou založeny **na modelech přírodních evolučních procesů**, které významně napomáhají k nalezení optimálního řešení náročných úloh. Vygenerovaná řešení se postupně vylepšují v evolučním cyklu (obr.12.). Tématu se věnuje celá řada autorů (např. ZELINKA a kol.¹⁷, ŠIKULOVÁ¹⁸), tato práce se dotýká pouze specifické části – metod **symbolické regrese**.



Obr. 12. ŠIKULOVÁ, Michaela. *Obecný cyklus evolučního algoritmu*. In:¹⁸

3.1 Základní pojmy

Ve své diplomové práci dále využívám termínů specifických pro oblast evolučních technik, jejich význam stručně vysvětluji v této podkapitole.

¹⁸ ŠIKULOVÁ, Michaela. *Symbolická regrese a koevoluce*. Brno, 2011. Symbolická regrese a koevoluce. Dostupné z: <http://www.fit.vutbr.cz/study/DP/rpfile.php.cs?id=7960&y=0>. Diplomová práce. VUT v Brně, FIT.

Jedinec – nalezené řešení problému

Populace – množina jedinců

Rodič – vybraný jedinec z populace pro následné křížení/mutaci, zpravidla s nejlepší fitness

Potomek – jedinec vzniklý výsledkem evolučního operátoru (křížení, mutace...)

Křížení – evoluční technika pro vznik nových jedinců kombinací rodičů

Mutace – evoluční technika pro změnu náhodné části jedince

Stagnace – stav, kdy evoluční cyklus nenalézá lepší řešení

Fitness – hodnota představující ohodnocení jedince, do jaké míry se jedná o vhodné řešení

Fitness funkce – funkce realizující přiřazení hodnot fitness jedincům

3.2 Symbolická regrese

Evoluční algoritmy se zpravidla snaží najít globální minimum zadané funkce. Symbolická regrese si klade jiný cíl – **pro zadaná data hledá vhodnou regresivní funkci**. Jinými slovy hledá funkci, která zajistí požadované výstupní hodnoty v závislosti na vstupních.

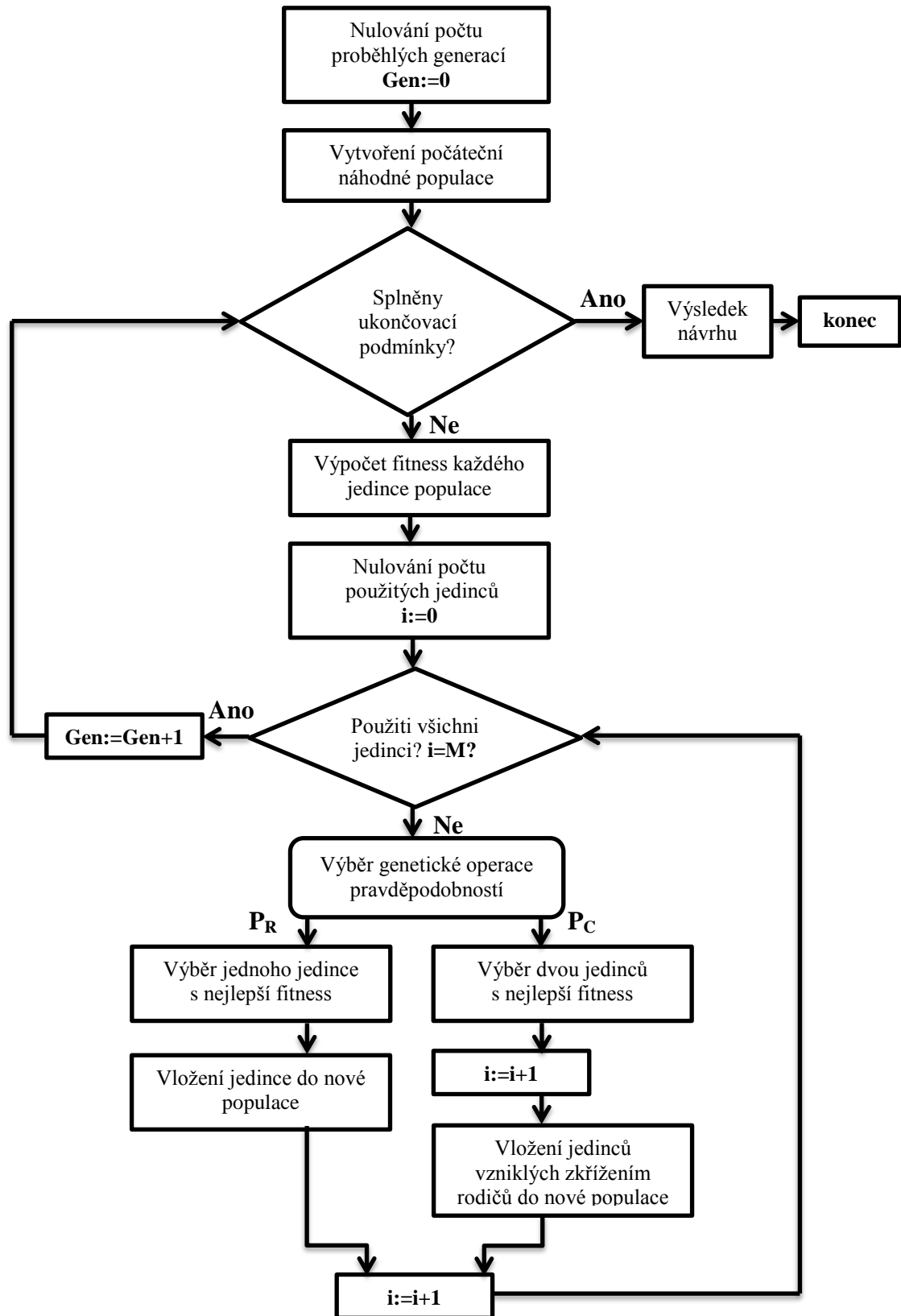
Používají se především tři základní algoritmy – **genetické programování**, gramatická evoluce a analytické programování. Pro syntézu kombinačních obvodů je velmi vhodně uzpůsobena především **kartézská varianta** prvního jmenovaného díky maticové reprezentaci jedinců.

3.3 Genetické programování

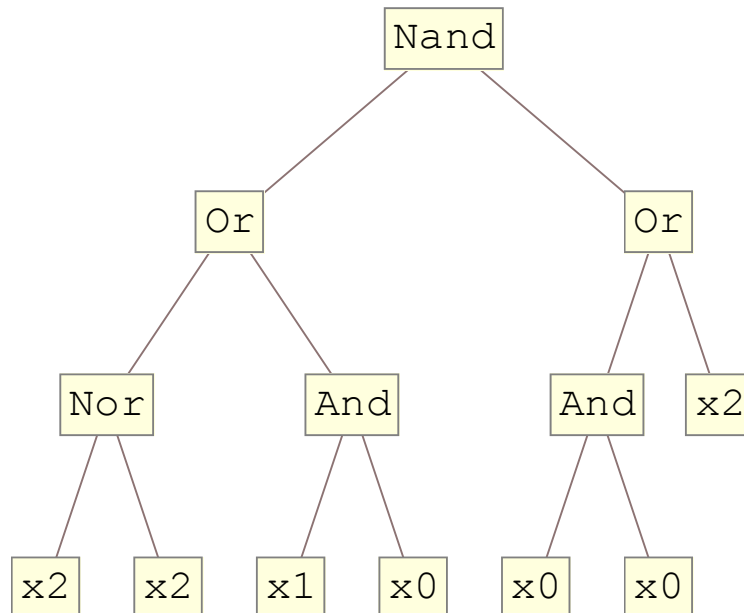
Genetické programování (dále GP) vzniklo modifikací *genetických algoritmů*¹⁹ pro aplikování symbolické regrese, využívání symbolických objektů. Algoritmem GP se podrobně zabývá KOZA²⁰ (*obr.13.*).

¹⁹ **genetické algoritmy**: varianta evolučních algoritmů, více informací např. zde: HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-2695-3.

²⁰ **KOZA**, John R. *Genetic programming: On the programming of computers by means of natural selection*. Cambridge: Bradford Book, 1992, 818 s. ISBN 02-621-1170-5.

Obr. 13. Algoritmus GP podle KOZY²⁰.

Jak uvádí POLI, LANGDON a MCPHEE²¹, jedinec se zpravidla vyjadřuje pomocí **syntaktických stromů** (obr.14.) skládajících se z *terminálů* (proměnné a konstanty) a *funkcí* (např. Plus, Sin, Nand). Vrchol stromu označujeme pojmem *kořen*, stromový zápis čteme zleva zdola směrem ke kořenu.



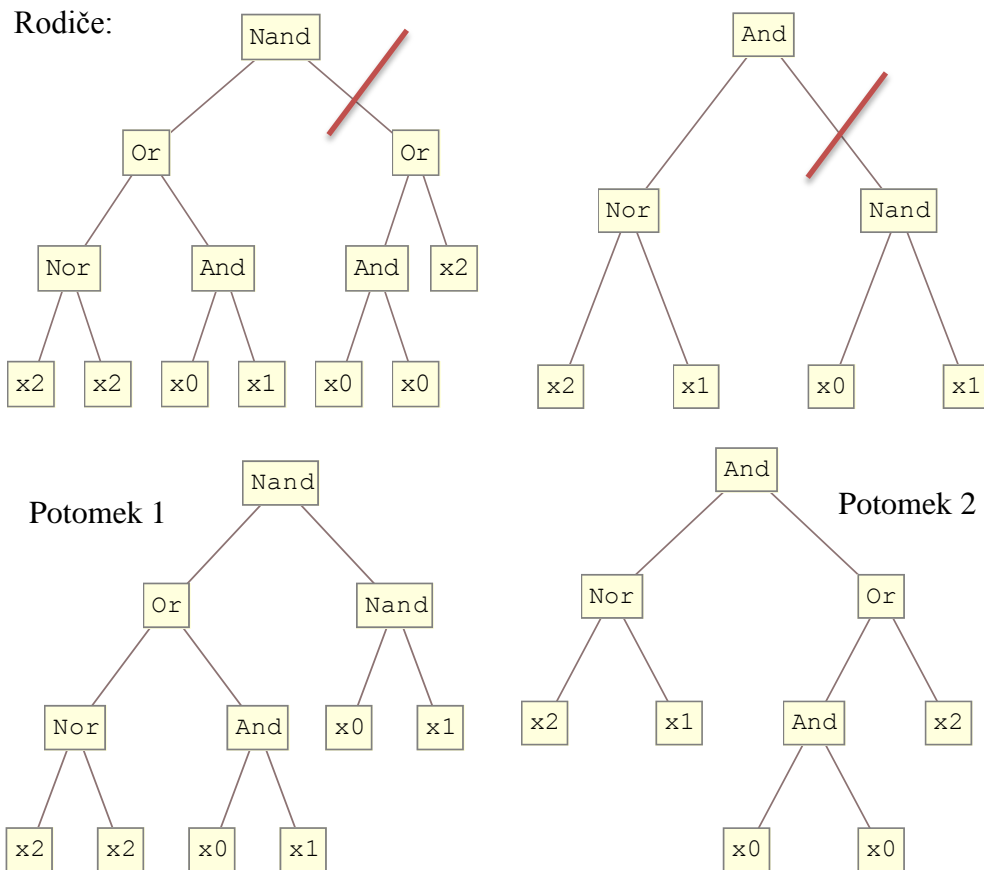
Obr. 14. Ukázka syntaktického stromu.

V GP probíhá evoluce jedince pomocí operátorů **křížení** (obr. 15.) a **mutace** (obr. 16.). Volba operátora je řízena parametry *pravděpodobnost křížení* a *pravděpodobnost mutace*.

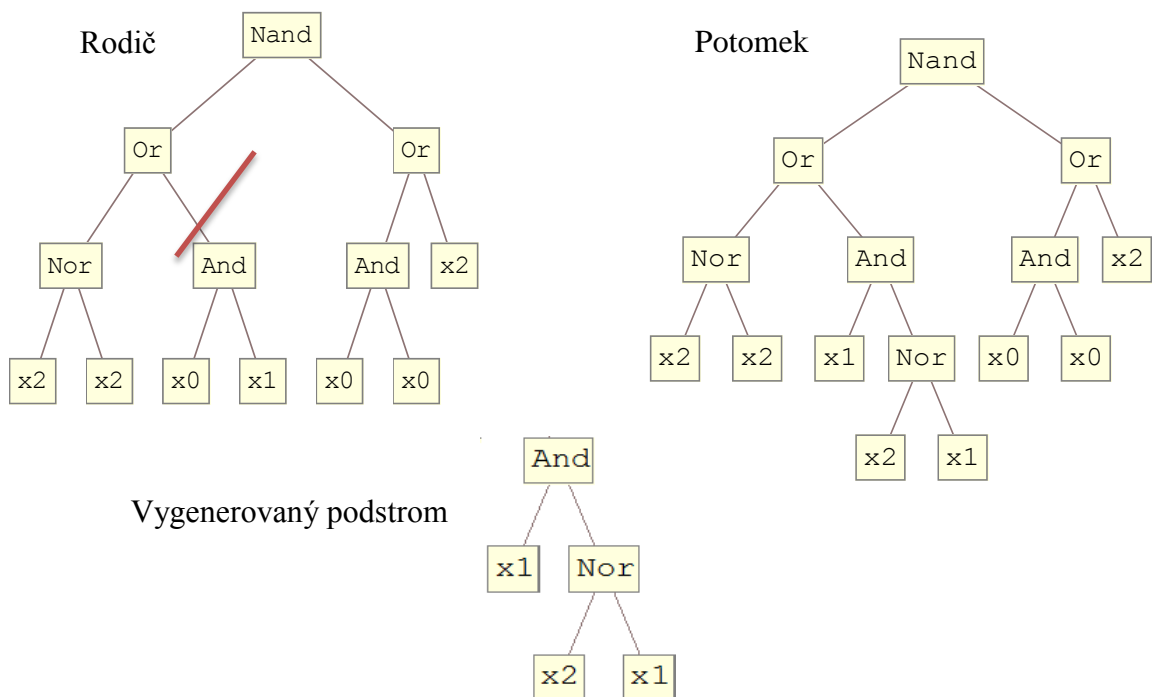
křížení ... nový jedinec vzniká záměnou podstromů náhodně vybraných uzlů rodičů

mutace ... nový jedinec vzniká nahrazením podstromu v náhodně zvoleném uzlu nově vygenerovaným podstromem

²¹ POLI, Riccardo, W LANGDON a Nicholas F MCPHEE. *A field guide to genetic programming*. [S.l.: Lulu Press], 2008, 233 s. ISBN 978-1-4092-0073-4. Dostupné z: http://www.lulu.com/items/volume_63/2167000/2167025/2/print/book.pdf

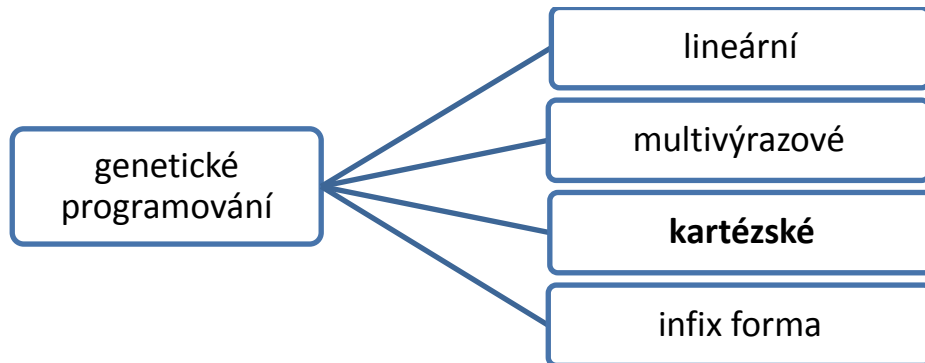


Obr. 15. Evoluční operátor křížení.



Obr. 16. Evoluční operátor mutace.

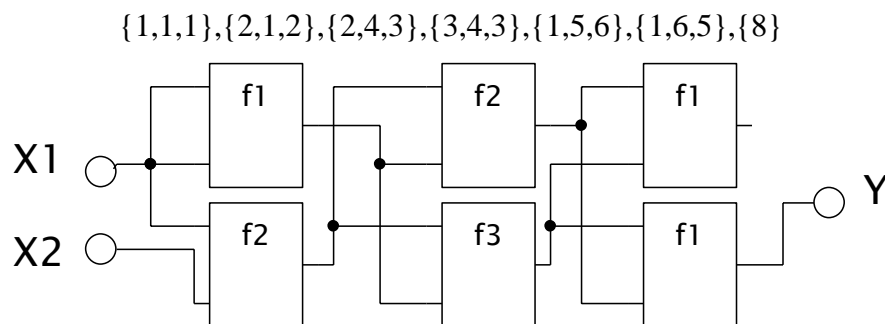
Genetické programování se vyskytuje ve více podobách lišících se kódováním jedince, uvádí to ve své knize ZELINKA a kol.²² **Kartézskou modifikaci uplatňuji v praktické části této práce.**



Obr. 17. Podoby genetického programování podle ZELINKY a kol.²⁰

3.4 Kartézské genetické programování CGP

Kartézské genetické programování (dále CGP) se liší od GP reprezentací jedince. CGP pracuje s číselnými vektory hodnot představující **maticově uspořádaný graf** s pevným počtem uzlů (obr. 18.), čímž se jedná o velmi vhodný nástroj pro syntézu kombinačních obvodů. Podle MILLERA²³ umožňuje CGP rovněž generovat nové obrázky, filtrovat šumové pixely v již zhotovených atd.

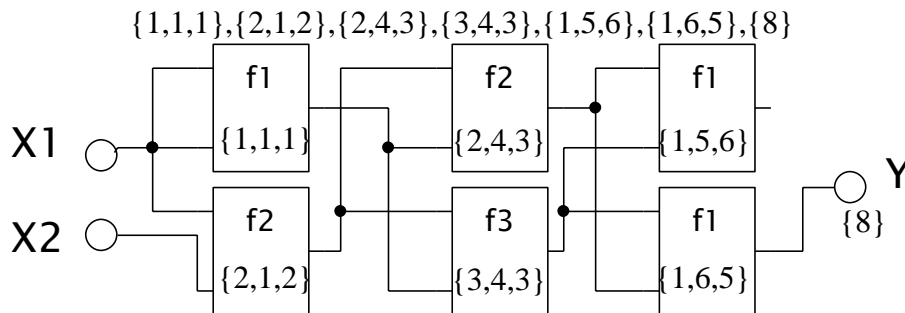


Obr. 18. Ukázkový jedinec CGP.

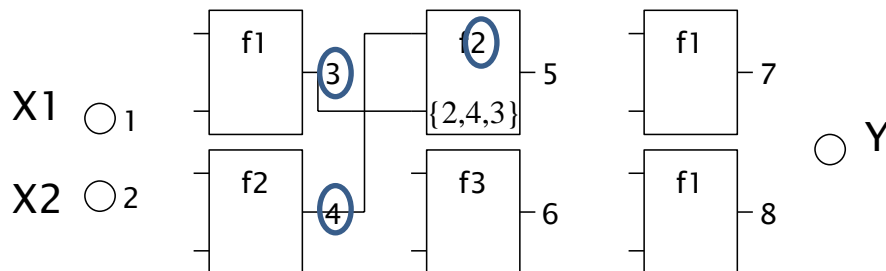
²² ZELINKA, Ivan, Zuzana OPLATKOVÁ, Miloš ŠEDA, Pavel OŠMERA a František VČELAŘ. *Evoluční výpočetní techniky: principy a aplikace*. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.

²³ MILLER, J. F. *Cartesian genetic programming*. New York: Springer Berlin Heidelberg, c2011, 344 s. Natural computing series. ISBN 978-3-642-17309-7.

Převod řádkového tvaru na maticový probíhá **přiřazením vektorů jednotlivým prvkům** (obr. 19.) a **interpretací členů vektorů** (obr. 20.) – první číslo zastupuje funkci (např. And, Or), druhé a třetí označují výstupy předchozího sloupce, které se připojí na vstup aktuálně řešeného bloku.

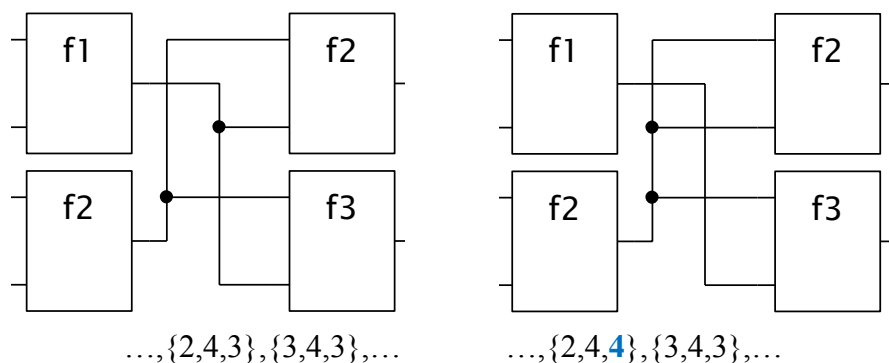


Obr. 19. Přiřazení vektorů řádkového tvaru prvkům v maticovém tvaru.



Obr. 20. Interpretace členů vektoru do maticového tvaru.

Uživatel zasahuje do chodu CGP nastavením počtu řádků a sloupců a také parametrem ***l-back*** (*level-back*), který řídí propojení bloků v maticovém tvaru jedince – určuje **počet předchozích sloupců**, ze kterých můžeme připojit vstup bloku v aktuálním sloupci. Tento parametr významně ovlivňuje výsledek. CGP dále využívá evolučního operátoru **mutace** (obr.21.).



Obr. 21. CGP – před mutací (vlevo) a po mutaci (vpravo) jedince.

Fitness jedince bývá vyjádřena **sumou rozdílů** výstupních hodnot pravdivostní tabulky a výstupních hodnot vygenerovaného obvodu.

4 WOLFRAM MATHEMATICA

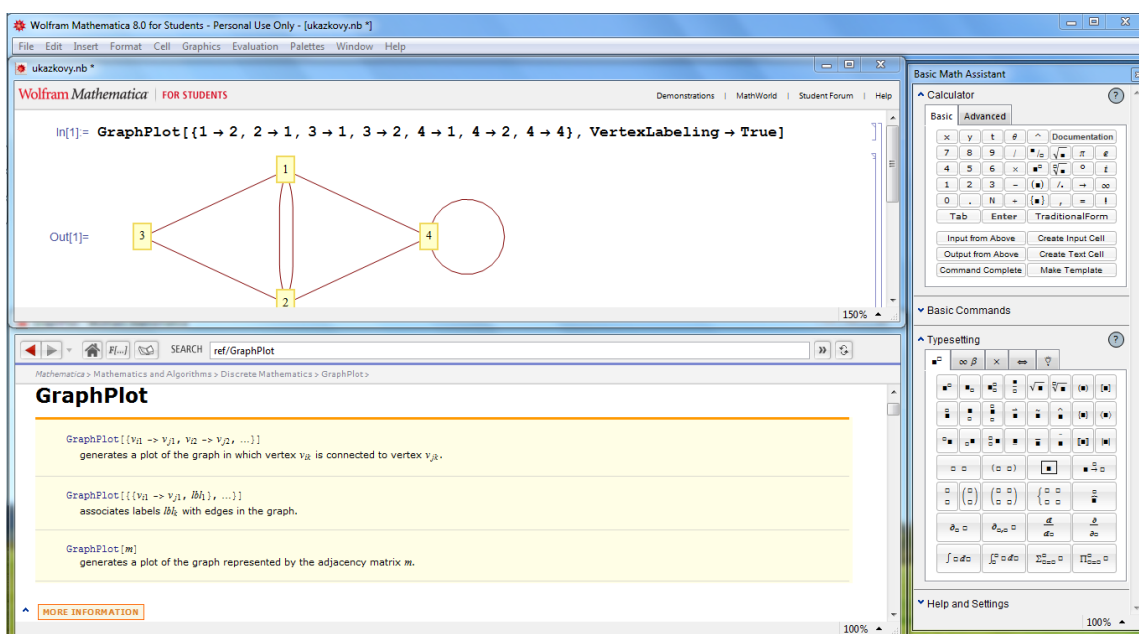
Software **Wolfram Mathematica 8 for Students** (dále WM) hraje důležitou roli v praktické části mé práce, seznámení s ním přináší tato kapitola.

4.1 Úvod do prostředí

WM vyvíjí společnost Wolfram Research. Jedná se o **výpočetní a programovací prostředí s vlastním jazykem**. Integruje velké množství nástrojů a funkcí:

- nástroje pro symbolické a numerické výpočty
- palety nástrojů (*obr. 22. vpravo*) pro snadnější práci se symbolickými prvky (zlomky, matice, integrály apod.)
- grafické nástroje pro vizualizaci výsledků
- CHRAMCOV²⁴ zmiňuje také možnost propojení s externími aplikacemi (Excel, SQL, Java atd.)

K dispozici máme také velmi **propracovanou interaktivní technickou dokumentaci** (*obr. 22. dole*), jenž podrobně popisuje dostupné funkce a příkazy s ukázkami použití a možnostmi nastavení.



Obr. 22. Prostedí Wolfram Mathematica 8 for Students.

²⁴ CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica*. Vyd. 1. Ve Zlíně: Univerzita Tomáše Bati ve Zlíně, 2005, 122 s. ISBN 80-7318-268-8.

4.2 Vybrané funkce

Uvádím zde stručný přehled vybraných příkazů využitých v praktické části. Více informací o nich lze nalézt v nápovědě WM. Pro přehlednost nedávám popisky.

logické funkce

And, Nand, Or, Nor – vstupní hodnoty *True / False* `In[1]:= And[True, False]`
 – výstupem je příslušná logická operace `Out[1]= False`

Simplify – vrací nejjednodušší podobu vloženého výrazu

```
In[2]:= Simplify[(((x2 &Agrave; x0) && (x2 &Agrave; x0)) || (x0 && x2)) &Agrave;
((x2 &Agrave; x0) || x2 || x0 || (x1 &Agrave; x1))]
```

`Out[2]= x0 À x2`

funkce s podmínkami a cykly

If – na základě stanovené podmínky provede zadané příkazy, `In[3]:= If[10 < 3, a, b]`
 když podmínka je / není splněna `Out[3]= b`

Do – provede zadané příkazy se zvoleným `In[4]:= Do[Print[n], {n, 0, 10, 5}]`
 rozmezím vstupních hodnot a všechny dílčí `0`
 výsledky vyobrazí `5`
`10`

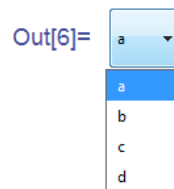
Which – na základě stanovené podmínky provede zadané příkazy, je-li podmínka splněna
 – umožňuje vložit více bloků podmínka-příkaz současně, jakmile narazí na první blok se splněnou podmínkou, příkaz provede a další podmínky již netestuje

```
In[5]:= Which[1 == 2, a, 2 - 1 == 1, b, 1 == 1, c]
```

`Out[5]= b`

ovládací funkce

PopupMenu – rozbalovací menu pro výběr `In[6]:= PopupMenu[a, {a, b, c, d}]`
 jedné z několika hodnot



Slider – plynulý výběr hodnot z nastaveného rozmezí pomocí jezdece

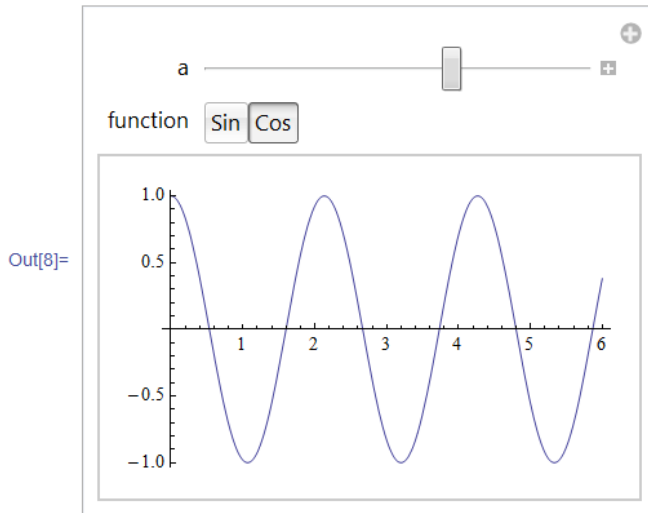
```
In[7]:= Slider[Dynamic[n], {0, 100, 1}] Dynamic[n]
```



Manipulate – funkce integrující ovládání a vizualizaci

- možnost vkládat různé typy ovládacích prvků
- vizualizační část se mění v závislosti na ovládacích v reálném čase

```
In[8]:= Manipulate[Plot[function[a x], {x, 0, 6}], {a, 1, 4}, {function, {Sin, Cos}}]
```

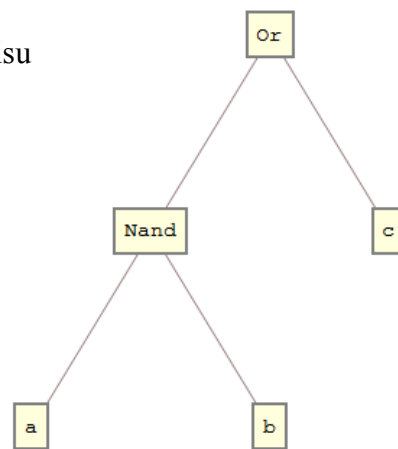


```
In[10]:= TreeForm[Or[Nand[a, b], c]]
```

```
Out[10]//TreeForm=
```

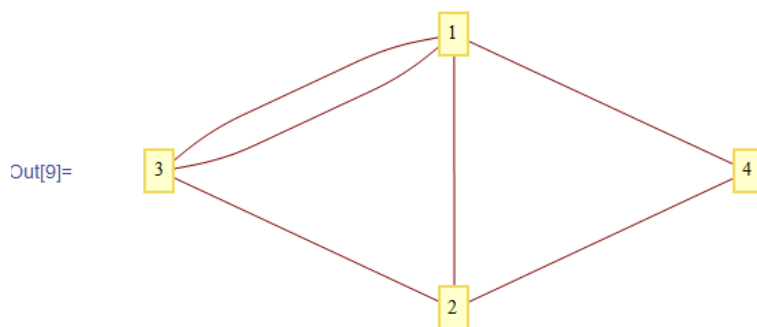
grafické funkce

TreeForm – stromové zobrazení řádkového zápisu



GraphPlot – vytváří graf s volitelnou konektivitou

```
In[9]:= GraphPlot[{1 → 3, 2 → 1, 3 → 1, 3 → 2, 4 → 1, 4 → 2}, VertexLabeling → True]
```



další funkce

Part – výběr konkrétního prvku vektoru `In[11]= {a, b, c, d, e, f}[[4]]`
– zastupuje symbolické značení `[[n]]` `Out[11]= d`

RandomInteger – generuje náhodné celé číslo ze zvoleného rozsahu

`In[12]= RandomInteger[{1, 20}]`

`Out[12]= 1`

4.3 Mathematica vs. CDF Player vs. Player Pro

Společnost Wolfram Research²⁵ nabízí krom svého hlavního produktu Mathematica rovněž **alternativní aplikace**, díky kterým můžeme pracovat se soubory vytvořenými ve WM. Alternativní aplikace však nabízí podstatně méně funkcí, jejich srovnáním se zabývá tato podkapitola a také samotný výrobce na svých webových stránkách²⁶. WM dává defaultně souborům příponu `.nb`, alternativní aplikace vyžadují spíše soubory s příponou `.cdf` (*computable document format*). Do formátu CDF umí konvertovat WM verze 8, k dispozici také bezplatný online nástroj²⁷.

Wolfram Mathematica

- placený produkt, vysoké školy však zpravidla nabízí svým studentům licence zdarma
- po registraci na webových stránkách²⁸ výrobce k dispozici zdarma ke stažení 15-denní trial verze bez omezení funkcí
- hlavní produkt, umožňuje plnou editaci a spuštění souborů s příponou `.nb` a `.cdf`

²⁵ **Wolfram Research: Mathematica, Technical and Scientific Software** [online]. 2012 [cit. 2012-03-27]. Dostupné z: <http://www.wolfram.com/>

²⁶ **srovnání:** <http://www.wolfram.com/player-pro/how-player-pro-compares.html>

²⁷ **online konverze formátu:** <http://www.wolfram.com/solutions/interactivedeployment/publish/>

²⁸ **trial verze Wolfram Mathematica:** <http://www.wolfram.com/mathematica/trial/>

Wolfram CDF Player

- zdarma
- po registraci lze stáhnout na webových stránkách výrobce²⁹ časově neomezenou verzi
- slouží k prohlížení souborů .nb a .cdf, kód však spustíme pouze se soubory .cdf
- neumožňuje spustit dynamický obsah vyžadující znovunačtení zdrojového kódu, **není proto vhodný pro systém vytvořený v praktické části této práce**

Wolfram Player Pro

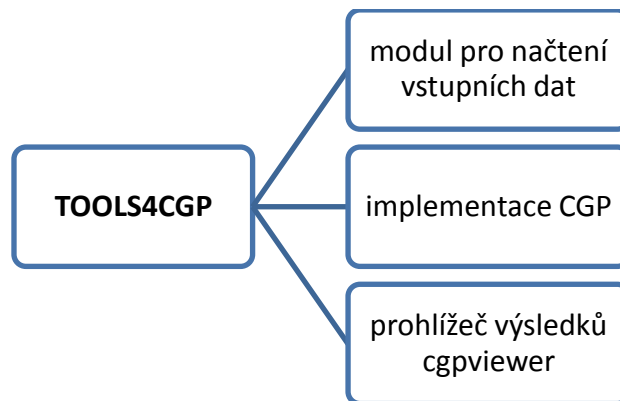
- placený produkt³⁰ bez možnosti stáhnutí trial verze
- obdobně jako u CDF Playeru slouží k prohlížení souborů .nb a .cdf, kód však spustíme pouze se soubory .cdf
- navíc však umožňuje spustit také dynamický obsah vyžadující znovunačtení zdrojového kódu

²⁹ **Wolfram CDF Player:** <http://www.wolfram.com/cdf-player/>

³⁰ **Wolfram Player Pro:** <http://www.wolfram.com/player-pro/>

5 STÁVAJÍCÍ NÁVRHOVÝ SYSTÉM: TOOLS4CGP

TOOLS4CGP³¹ je sada nástrojů – externích modulů (obr. 23.) pro evoluční návrh kombinačních obvodů. Systém vytvořen v jazyce C, evolučně založen na CGP.



Obr. 23. Nástroje TOOLS4CGP.

5.1 Modul pro načtení vstupních dat

Počáteční parametry uživatel zadává formou **externího souboru**. Pro vložení pravdivostní tabulky je k dispozici modul *tab2h*, který převádí pomocí příkazového řádku tabulku z textového formátu .txt do hlavičkového .h³² (obr. 24.).

```

C:\TOOLS4CGP\tab2h\median5.txt
# median z 5b
# 5 vstupy, 1 vystup
#i i4,i3,i2,i1,i0
#o out
00000 : 0
00001 : 0
00010 : 0
00011 : 0
00100 : 0
00101 : 0
00110 : 0
00111 : 1
01000 : 0
01001 : 0
01010 : 0
01011 : 0

C:\TOOLS4CGP\tab2h\median5.h
#define POPIS "# median z 5b 5 vstupy, 1 vystup"
//Pocet vstupu a vystupu
#define PARAM_IN 5 //pocet vstupu komb. ob
#define PARAM_OUT 1 //pocet vystupu komb. ob
//Inicializace dat. pole
#define init_data(a) \
a[0]=0xffff0000;\
a[1]=0xff00ff00;\
a[2]=0xf0f0f0f0;\
a[3]=0xcccccccc;\
a[4]=0xaaaaaaaa;\
a[5]=0xfeee8880;
//Pocet prvku pole
#define DATASIZE 6
  
```

Obr. 24. TOOLS4CGP – převod pravdivostní tabulky (vlevo .txt, vpravo .h).

³¹ **TOOLS4CGP** – zdarma ke stažení

VAŠÍČEK, Zdeněk a Lukáš SEKANINA. **Nástroje pro kartézské genetické programování**. *Fakulta informačních technologií VUT v Brně* [online]. 2008 [cit. 2012-03-27]. Dostupné z: http://www.fit.vutbr.cz/research/view_product.php?id=61¬itle=1

³² **hlavičkový soubor .h**: podpůrný soubor pro program vytvořený jazykem C

Parametry řídící běh evoluce – *l-back*, počet generací, velikost populace atd. se nachází v jiném, již zkompilevaném souboru (obr. 25.).

```

C:\TOOLS4CGP\cgp\cgp.h
0 10 20 30 40 50 60 70
#define POPULACE_MAX 5 //maximalni pocet jedincu populace
#define MUTACE_MAX 3 //max pocet genu, ktery se muze zmutovat b

#define PARAM_M 5 //pocet sloupcu
#define PARAM_N 5 //pocet radku
#define L_BACK 1 //1 (pouze predchozi sloupec) .. param_m

#define PARAM_GENERATIONS 50000 //max. pocet generaci evoluce
#define PARAM_RUNS 10 //max. pocet behu evoluce
#define FUNCTIONS 4 //max. pocet pouzitych funkci bloku (v
#define PERIODICLOGG (PARAM_GENERATIONS/2) //po kolika krocich se ma v
#define xPERIODIC_LOG //zda se ma vypisovat populace

```

Obr. 25. TOOLS4CGP – parametry řídící běh evoluce.

5.2 Implementace CGP

Modul se spouští souborem s příponou .exe, vyžaduje ve složce *data* existenci hlavičkového souboru s pravdivostní tabulkou. Po spuštění se krátce vyobrazí příkazový řádek (obr. 26.) a vytvoří se 10 nových souborů s příponou .chr, uzpůsobených pro vizualizaci v modulu *cgpviewer*, s **uloženými nejlepšími jedinci**.

```

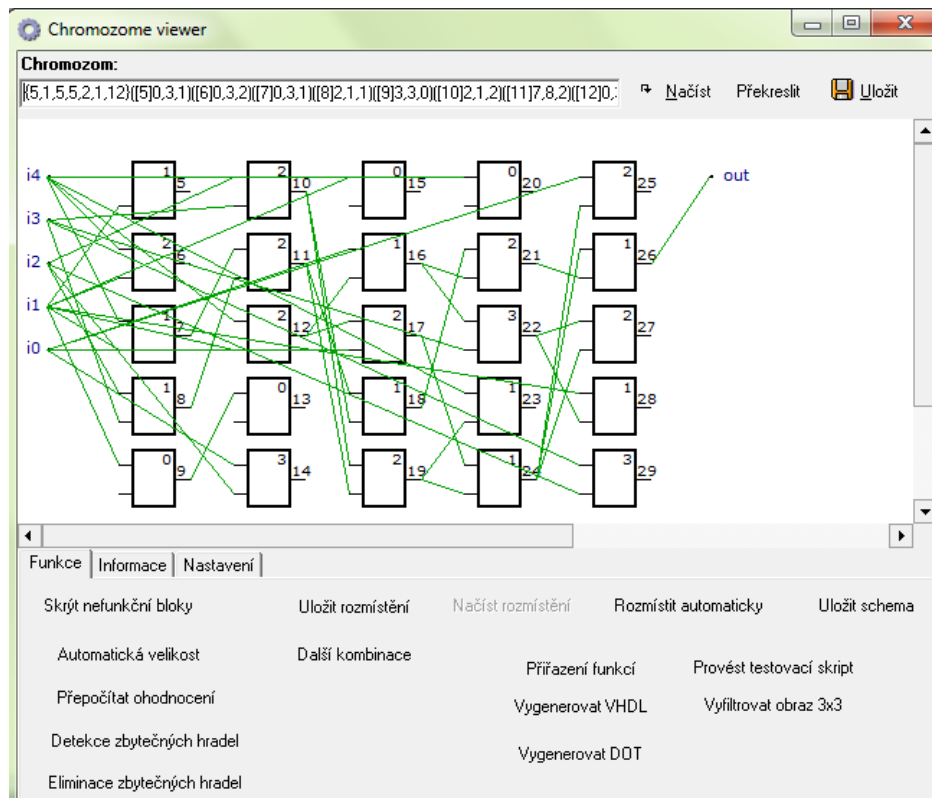
C:\windows\system32\cmd.exe
Run: 1 Tue Apr 03 19:14:46 2012
-----
Generation:78 Fitness: 24/32
Generation:228 Fitness: 26/32
Generation:379 Fitness: 27/32
Generation:6312 Fitness: 28/32
Generation:7791 Fitness: 29/32
Generation:9618 Fitness: 30/32
Generation:9625 Fitness: 31/32
Generation:26038 bestblk b:13
Generation:32657 bestblk b:12
Generation:32829 bestblk b:11
Best chromosome fitness: 32/32
Best chromosome: (5,1, 5,5, 2,1,11)<(1513,4,3)<(1612,2,0)<(1713,1,1)<(1810,3,3)<(1914,0,1)<(1010,4,2)<(1119,7,2)<(1211,3,2)<(1314,9,1)<(1410,7,3)<(15112,14,3)<(16110,12,1)<(1712,11,1)<(1811,2,0)<(1912,11,2)<(20116,15,0)<(2110,0,3)<(22117,16,2)<(23119,17,1)<(24117,19,2)<(25124,22,1)<(26122,24,0)<(2710,3,1)<(2812,21,0)<(29124,24,0)<(25)
Run: 2 Tue Apr 03 19:15:09 2012
-----
Generation:58 Fitness: 26/32
Generation:150 Fitness: 27/32
Generation:821 Fitness: 28/32
Generation:3215 Fitness: 29/32
Generation:4535 Fitness: 30/32
Generation:4961 Fitness: 31/32
Generation:11212 bestblk b:11
Best chromosome fitness: 32/32
Best chromosome: (5,1, 5,5, 2,1,11)<(1512,1,1)<(1611,2,2)<(1714,3,2)<(1813,2,2)<(1914,3,1)<(1017,5,3)<(1111,4,2)<(1215,9,2)<(1317,9,1)<(1416,7,1)<(15112,14,1)<(1610,14,1)<(17114,12,0)<(18114,12,2)<(1913,13,2)<(20118,4,3)<(21118,4,0)<(2210,15,2)<(2314,15,3)<(24116,1,2)<(25123,2,3)<(2614,0,2)<(2713,20,3)<(28124,24,2)<(29121,22,1)<(29)
Run: 3 Tue Apr 03 19:15:10 2012

```

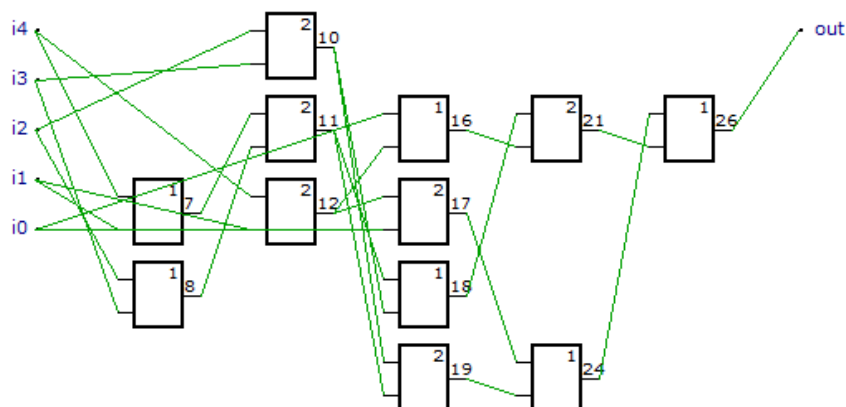
Obr. 26. TOOLS4CGP – běžící modul pro implementaci CGP.

5.3 Prohlížeč výsledků cgpviewer

Tento vizualizační modul slouží ke **grafickému zobrazení výsledků** (obr. 27.) předchozího modulu. Obsahuje rovněž podpůrné funkce, např. eliminaci nadbytečných spojů a hradel (obr. 28.).



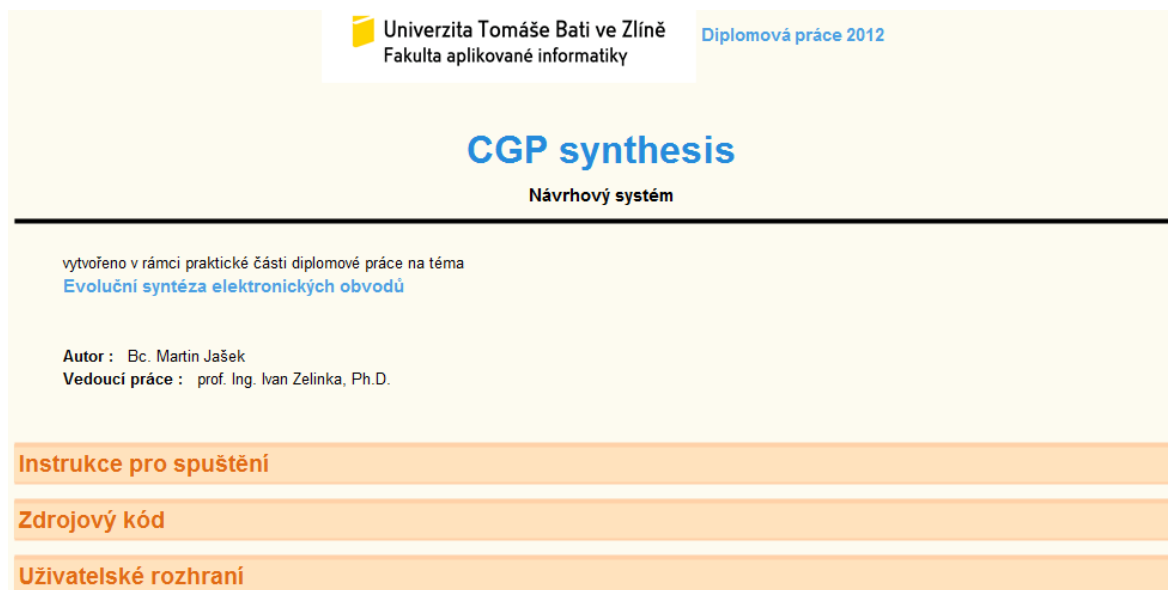
Obr. 27. TOOLS4CGP – nástroj cgpviewer po načtení jedince.



Obr. 28. TOOLS4CGP – demonstrace funkcí nástroje cgpviewer.

II. PRAKTICKÁ ČÁST

6 VYTVOŘENÝ NÁVRHOVÝ SYSTÉM: CGP SYNTHESIS



Obr. 29. CGP synthesis – náhled.

6.1 Úvodní seznámení

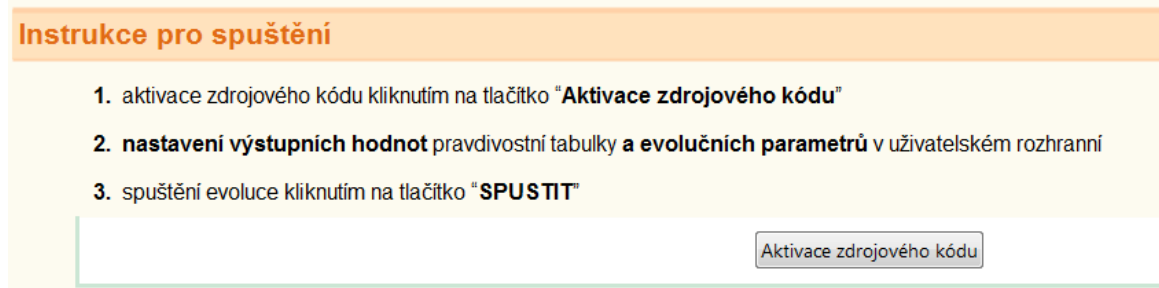
Tento nový návrhový systém elektronických obvodů jsem vytvořil v prostředí *Wolfram Mathematica 8 for Students* (studentská licence v rámci UTB ve Zlíně).

Integrované nástroje:

- **přehledné vkládání** vstupních údajů – pravdivostní tabulka, evoluční parametry
- evoluční algoritmus **CGP**
- řádkové vyjádření nejlepšího jedince s možností zobrazení stromové struktury
- **grafické vyjádření nejlepšího jedince** s barevným rozlišením a možností zobrazit část schématu
- grafické zobrazení **generačního vývoje** hodnot fitness.

Testoval jsem systém, konvertovaný do formátu CDF, ve zdarma dostupném software *Wolfram CDF Player*. Nevykazoval zde však správnou činnost, jelikož zmíněný přehrávač nepodporuje opětovnou evaluaci (znovunačtení) zdrojového kódu za účelem aplikování změn evolučních parametrů.

Pokyny pro správné spuštění programu (*obr.30.*). nalezne uživatel v první sekci zvýrazněné oranžovou barvou. Sekce se defaultně nachází zabaleny, rozbalují se pouhým kliknutím levého tlačítka myši.



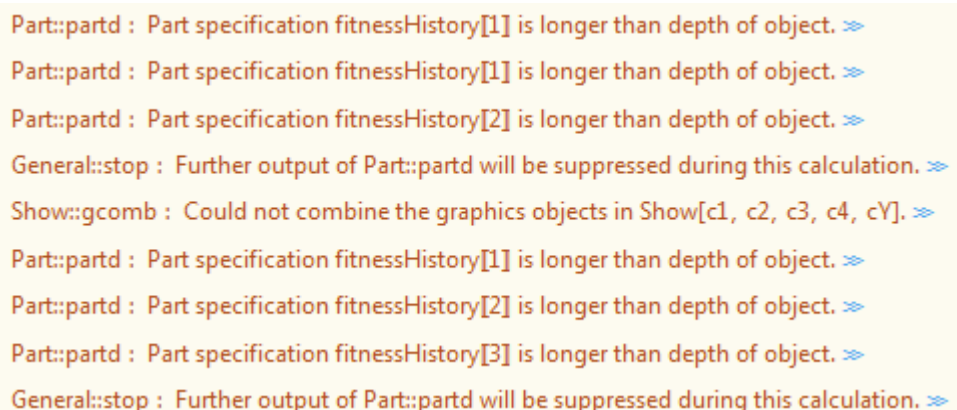
Instrukce pro spuštění

1. aktivace zdrojového kódu kliknutím na tlačítko "Aktivace zdrojového kódu"
2. nastavení výstupních hodnot pravdivostní tabulky a evolučních parametrů v uživatelském rozhraní
3. spuštění evoluce kliknutím na tlačítko "SPUSTIT"

Aktivace zdrojového kódu

Obr. 30. CGP synthesis – instrukce pro spuštění.

Nedodržením výše uvedeného spouštěcího postupu se mohou objevit chybové zprávy (*obr. 31.*) vlivem dosud nenačteného kódu a chybějících hodnot.



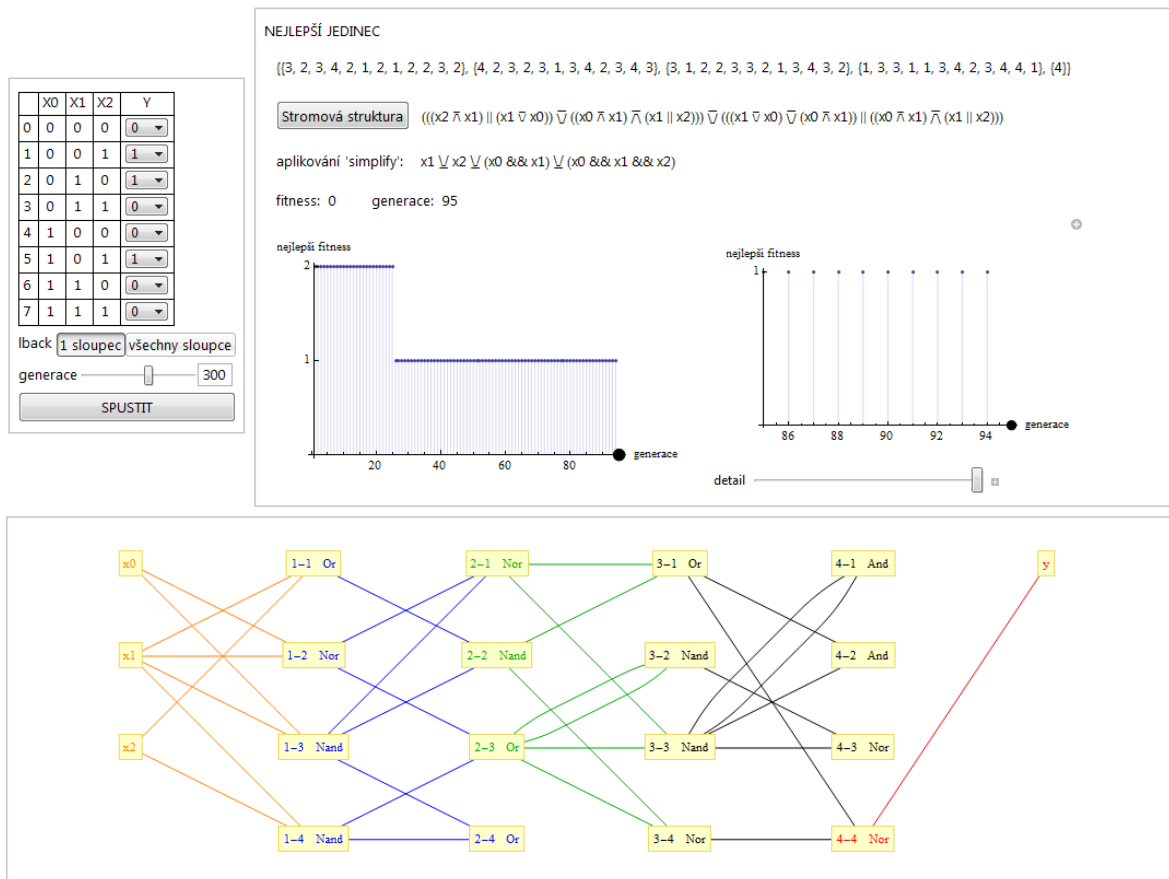
Part::partd : Part specification fitnessHistory[1] is longer than depth of object. >>
Part::partd : Part specification fitnessHistory[1] is longer than depth of object. >>
Part::partd : Part specification fitnessHistory[2] is longer than depth of object. >>
General::stop : Further output of Part::partd will be suppressed during this calculation. >>
Show::gcomb : Could not combine the graphics objects in Show[c1, c2, c3, c4, cY]. >>
Part::partd : Part specification fitnessHistory[1] is longer than depth of object. >>
Part::partd : Part specification fitnessHistory[2] is longer than depth of object. >>
Part::partd : Part specification fitnessHistory[3] is longer than depth of object. >>
General::stop : Further output of Part::partd will be suppressed during this calculation. >>

Obr. 31. CGP synthesis – chybová zpráva.

6.2 Uživatelské rozhraní

Uživatelské rozhraní

■ podpůrný kód

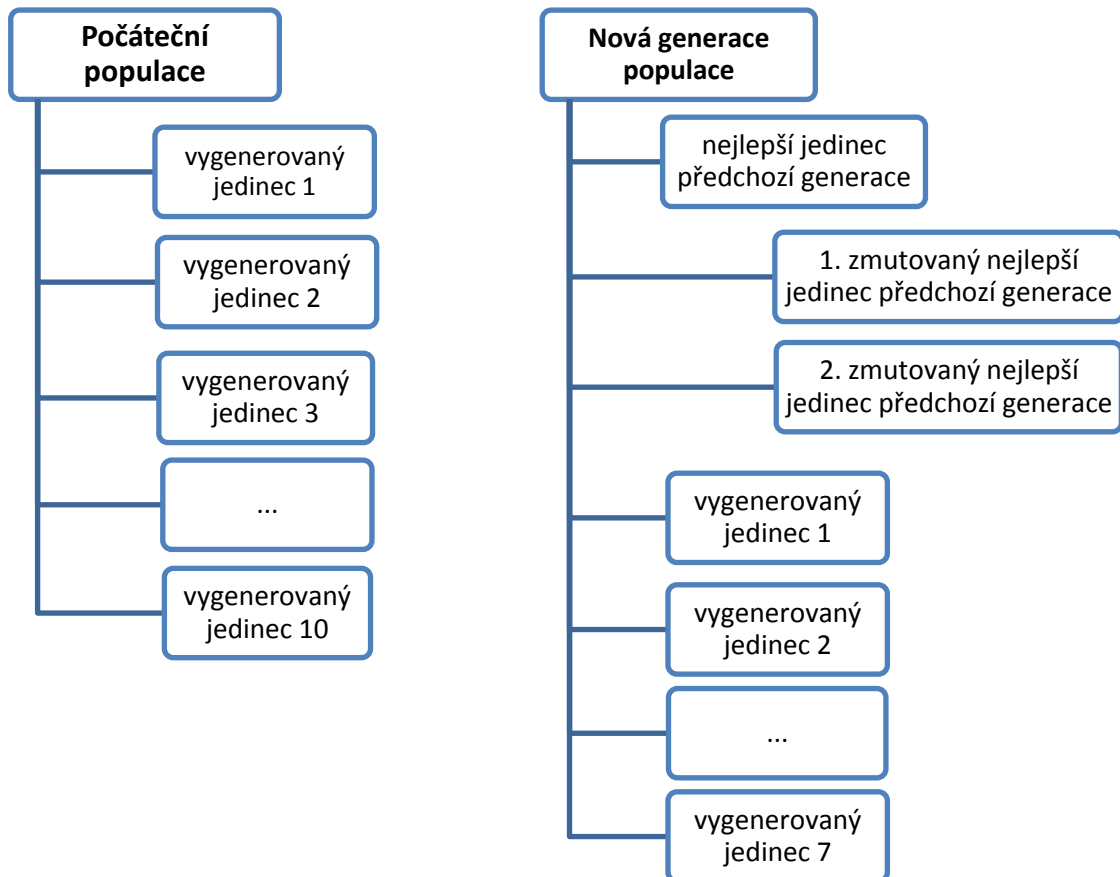


Obr. 32. CGP synthesis – náhled do uživatelského rozhraní.

Uživatelské rozhraní se skládá ze tří oken. Horní levé slouží k nastavení vstupních parametrů a spuštění, zbylé části k vizualizaci nejlepšího jedince a průběhu evoluce.

Nastavení vstupních parametrů

System umožňuje **přehledné zadání požadovaných parametrů**. Lze nastavit výstupní hodnoty pravdivostní tabulky, parametr *lback* (efekt demonstrován v odstavci *Grafické zobrazení nejlepšího jedince*), počet generací (význam na obr.33.). Poslední uvedené volíme posuvným jezdce v rozmezí hodnot <1;500>, případně manuálním vložením číslic do pole vedle jezdce.



Obr. 33. CGP synthesis – význam parametru „počet generací“.

Řádkové zobrazení nejlepšího jedince

NEJLEPŠÍ JEDINEC

{{3, 2, 2, 3, 3, 2, 4, 2, 3, 3, 3, 1}, {1, 4, 3, 4, 2, 2, 4, 4, 3, 3, 2, 1}, {1, 3, 3, 1, 1, 2, 1, 4, 3, 4, 3, 4}, {1, 3, 4, 4, 4, 2, 3, 4, 1, 3, 1, 2}, {4}}

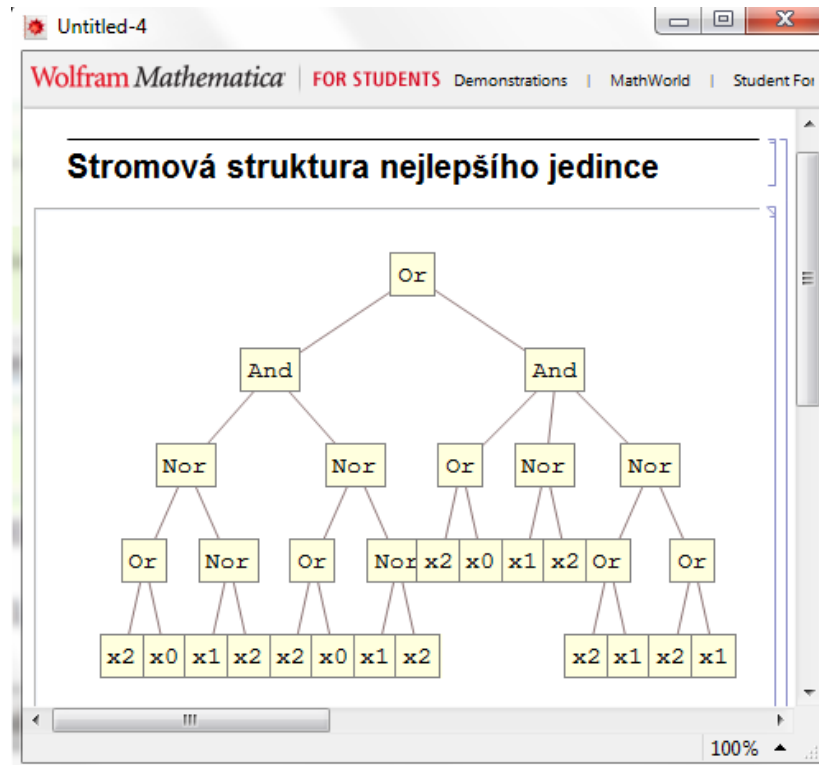
Stromová struktura $((x2 \parallel x0) \bar{\vee} (x1 \bar{\vee} x2)) \&\& ((x2 \parallel x0) \bar{\vee} (x1 \bar{\vee} x2))) \parallel ((x2 \parallel x0) \&\& (x1 \bar{\vee} x2) \&\& ((x2 \parallel x1) \bar{\vee} (x2 \parallel x1)))$

aplikování 'simplify': $x0 \bar{\vee} x1 \bar{\vee} (x0 \&\& x2) \bar{\vee} (x1 \&\& x2)$

fitness: 0 generace: 64

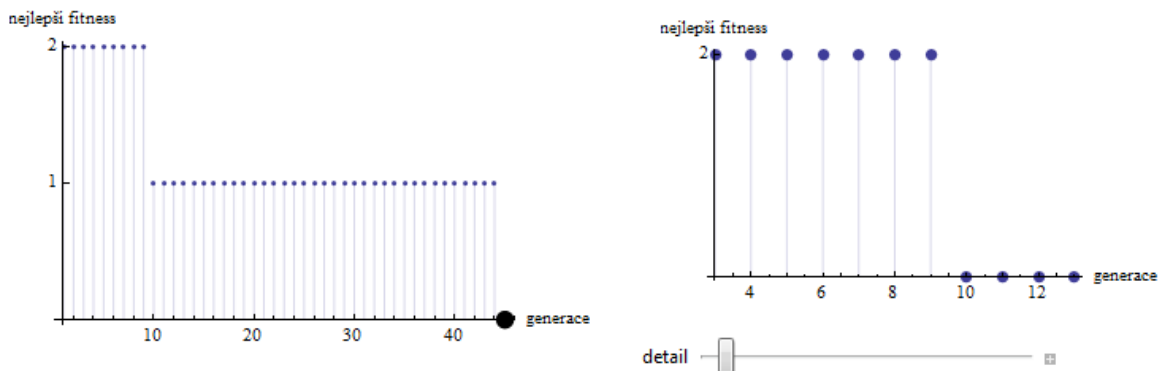
Obr. 34. CGP synthesis – řádkové zobrazení nejlepšího jedince.

Zobrazení sestává ze čtyř řádků. V prvním se nachází jedinec v CGP tvaru, v druhém ve formě booleovské rovnice s možností převodu do stromové struktury (obr. 35.) kliknutím na příslušné tlačítko. Následuje zjednodušení předchozí informace funkcí *Simplify* za účelem zhodnocení nalezeného řešení z hlediska rozsahu. Poslední řádek obsahuje údaje o hodnotě **fitness (0 = nejvhodnější)** a generaci, ze které jedinec pochází. **Algoritmus se zastaví při dosažení nulové fitness nebo zadaného počtu generací.**



Obr. 35. CGP synthesis – stromová struktura jedince.

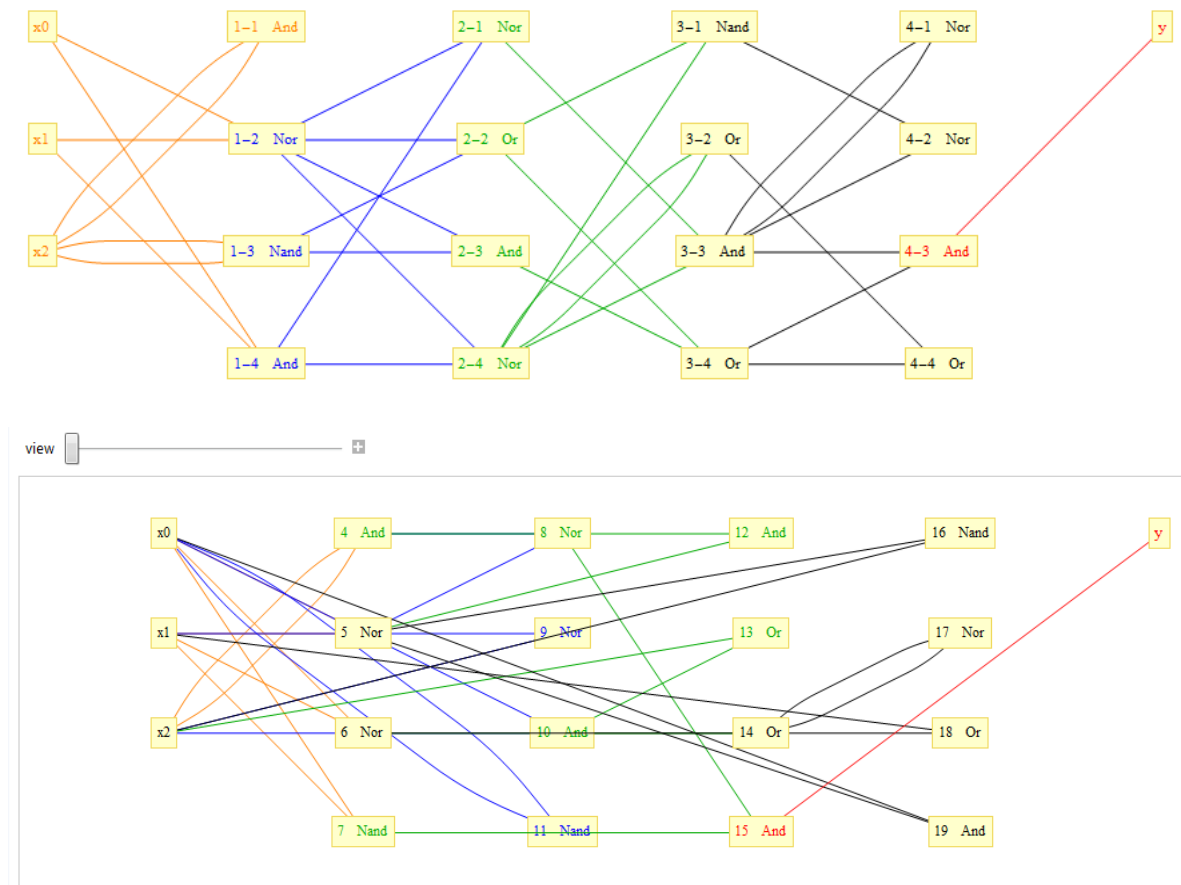
Grafické zobrazení průběhu evoluce



Obr. 36. CGP synthesis – grafické zobrazení průběhu evoluce.

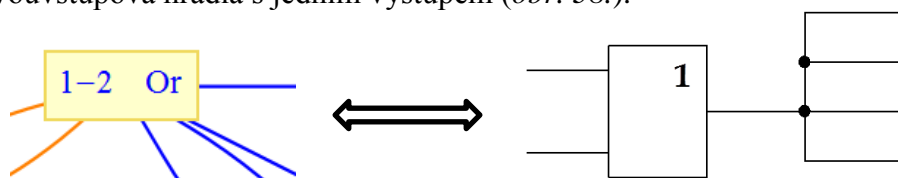
Výše uvedené grafy se zabývají **generačním vývojem fitness** nejlepšího jedince, sledují, v kterých generacích evoluce dochází ke zlepšení. Graf umístěný vlevo zajišťuje celkový pohled vývoje od první do poslední generace. Druhý graf se objeví pouze při počtu generací větším než 30 a slouží k detailnějšímu zkoumání vývoje s možností plynulého posunu jednotlivých etap.

Grafické zobrazení nejlepšího jedince



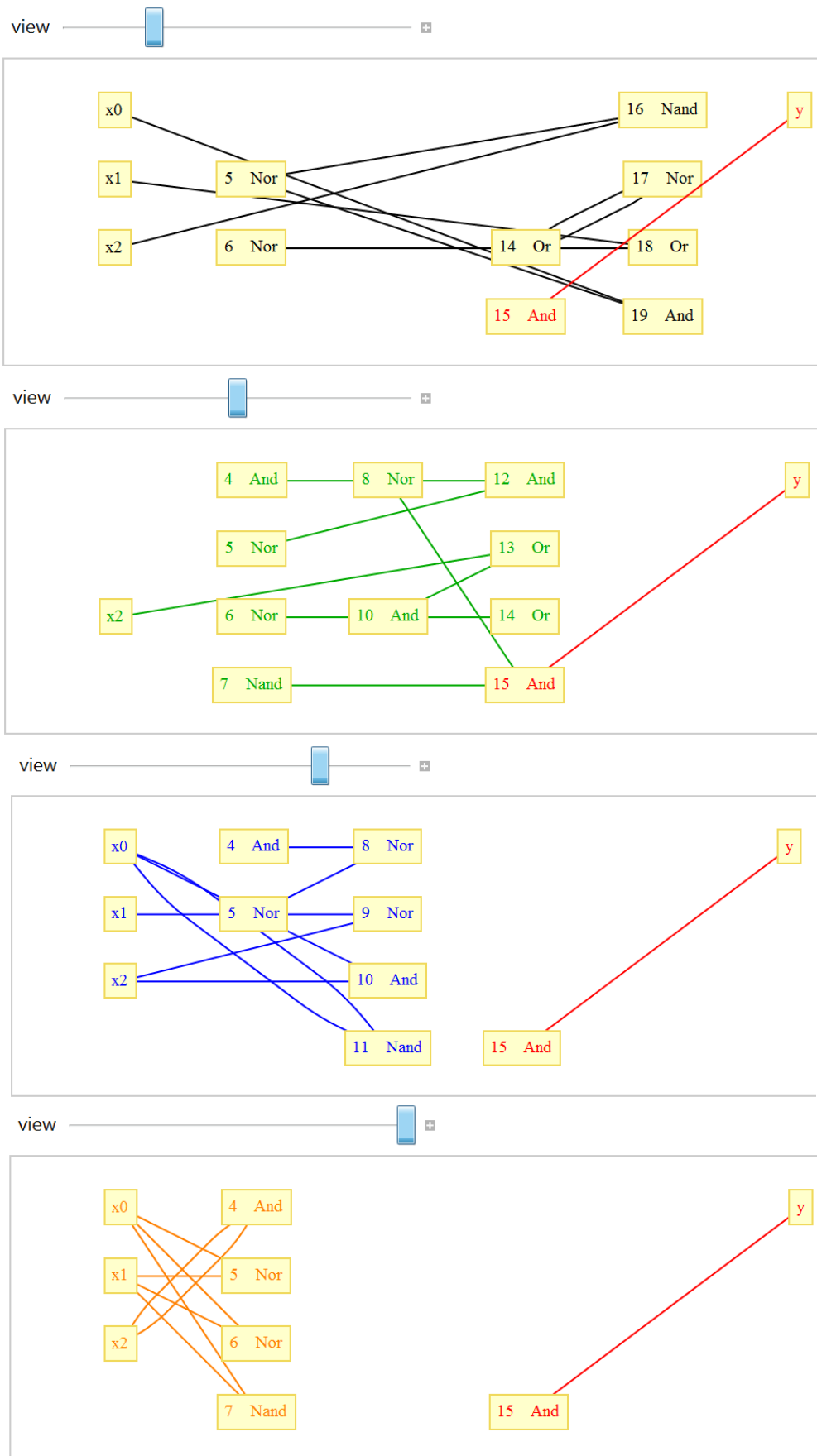
Obr. 37. CGP synthesis – grafické zobrazení nejlepšího jedince při parametru $lback=“1\text{ sloupec}“$ (nahore) a $lback=“vsechny\text{ sloupce}“$ (dole).

Grafické zobrazení nejlepšího jedince představuje **reálné schéma zapojení hradel**. Místo značek podle normy IEC jsem využil vestavěné funkce *GraphPlot* prostředí Mathematica. Funkce jednotlivých hradel jsou patrné přiřazeným textovým popisem v pravé části. Každý sloupec dostal svou jedinečnou barvu pro odlišení od ostatních, číselné označení slouží k jednoznačné identifikaci v rámci maticového rozmístění. Vždy se jedná o dvouvstupová hradla s jedním výstupem (obr. 38.).



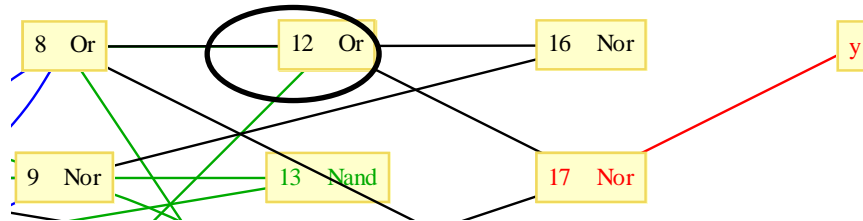
Obr. 38. CGP synthesis – interpretace značení.

V důsledku nepřehlednosti při parametru $lback=“vsechny\text{ sloupce}“$ jsem do systému implementoval funkci *view* (obr. 39.) za účelem **čitelnějšího zkoumání** zapojení.



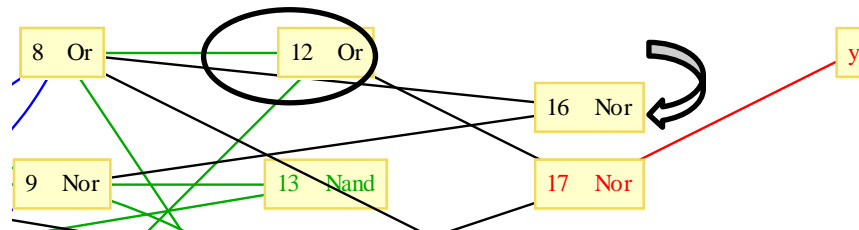
Obr. 39. CGP synthesis – módy funkce „view“.

V některých případech se může vyskytnout **zdánlivá absence jednoho ze dvou vstupů** symbolického hradla (*obr.40.*) způsobená souběžným vedením více propojujících čar najednou.



Obr. 40. CGP synthesis – zdánlivá absence vstupu.

Tento jev lze odhalit již zmiňovanou funkcí *view* (*obr. 39.*) nebo prostou **změnou polohy**³³ některého zasahujícího prvku (*obr.41.*).

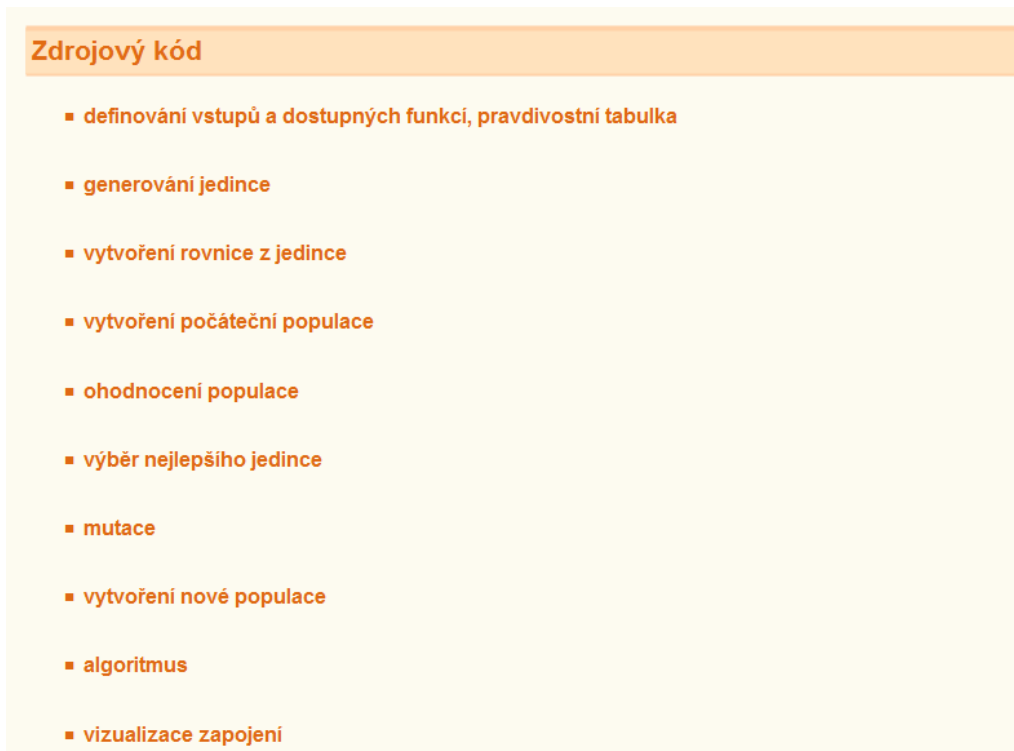


Obr. 41. CGP synthesis – odhalení mystifikace posunutím.

³³ **změna polohy některého zasahujícího prvku:**

Rychle klikneme 3x levým tlačítkem myši na prvek, který chceme posunout, čímž jej dostaneme do editovatelného stavu (po obvodu spatříme malé akční místa znázorněné čtverci). Následně tradičním způsobem (stisk tlačítka myši) přetáhneme objekt na nové umístění.

7 NÁHLED DO ZDROJOVÉHO KÓDU



Obr. 42. CGP synthesis – kategorizovaný zdrojový kód.

Zdrojový kód je **přehledně rozdělen do kategorií dle svého účelu** (obr. 42.). Do stěžejních částí vybraných přináší náhled tato kapitola. Některé (neuvedeny níže) obsahují funkci *Quiet* tlumící počáteční chybové zprávy vlivem absence vstupních parametrů mezi aktivací zdrojového kódu a prvotním spuštěním evoluce.

7.1 Generování jedince

```
1 rand3 := RandomInteger[{1, 3}];
2 rand4 := RandomInteger[{1, 4}];
3 rand5 := RandomInteger[{1, 5}];
4 rand7 := RandomInteger[{1, 7}];
5 rand11 := RandomInteger[{1, 11}];
6 rand15 := RandomInteger[{1, 15}];
7 rand19 := RandomInteger[{1, 19}];
```

Definování funkcí pro generování náhodných čísel ve specifickém rozsahu.

```

8 individual[lback_] := {{rand4, rand3, rand3, rand4, rand3,
  rand3, rand4, rand3, rand3, rand4, rand3, rand3},
9 {rand4, Which[lback==0, rand4, lback==1, rand7],
  Which[lback==0, rand4, lback==1, rand7], rand4,
...

```

Generování CGP tvaru jedince v závislosti na parametru lback.

7.2 Vytvoření rovnice z jedince

```

...
30 column2={#1[#2,#3],#4[#5,#6],#7[#8,#9],#10[#11,#12]}&[
31 functions[[population[[pos]][[2]][[1]]]],
32 inputsANDcolumn1[[population[[pos]][[2]][[2]]]],
33 inputsANDcolumn1[[population[[pos]][[2]][[3]]]],
...

```

Přiřazení funkce hradlům ve sloupci a nadefinování napojení vstupů v rámci vybraného jedince populace.

7.3 Ohodnocení populace

```

1 valuationPop:= Do[equation[pos]; fitnessvalue=0; tableY;
2 out=output /. x0 -> 0 /. x1 -> 0 /. x2 -> 0 // Simplify;
3 If[Boole[out][[1]][[1]]!=tableY[[1]],fitnessvalue++]
  /.Boole[1]->1/.Boole[0]->0;

```

Definování funkce pro porovnání výstupní hodnoty vygenerované rovnice s výstupní hodnotou z pravdivostní tabulky. Při nesrovnalosti navýší hodnotu fitness o 1.

7.4 Vizualizace zapojení

```

...
18 c1=GraphPlot[{c11->Which[bestIndividual[[1]][[2]]==1,"x0",
  bestIndividual[[1]][[2]]==2,"x1",bestIndividual[[1]][[2]]
  == 3,"x2"]},
...

```

Definování propojení bloků funkce GraphPlot v závislosti na nejlepším jedinci.

8 SROVNÁNÍ A ZHODNOCENÍ VÝSLEDKŮ

8.1 Výsledky experimentů – metoda 10 pokusů

Podkapitola obsahuje deset po sobě jdoucích výsledků konkrétních úloh při parametru *generace=200*.

Tab. 5. Experiment metodou 10 pokusů – úloha 1.

úloha					pokus	lback="1 sloupec"	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 18	fitness 0 v generaci 42
0	0	0	0	1 ▾	2.	fitness 0 v generaci 15	fitness 0 v generaci 9
1	0	0	1	0 ▾	3.	fitness 0 v generaci 26	fitness 0 v generaci 10
2	0	1	0	1 ▾	4.	fitness 0 v generaci 45	fitness 0 v generaci 19
3	0	1	1	0 ▾	5.	fitness 0 v generaci 15	fitness 0 v generaci 12
4	1	0	0	0 ▾	6.	fitness 0 v generaci 14	fitness 0 v generaci 8
5	1	0	1	0 ▾	7.	fitness 0 v generaci 2	fitness 0 v generaci 16
6	1	1	0	0 ▾	8.	fitness 0 v generaci 12	fitness 0 v generaci 135
7	1	1	1	0 ▾	9.	fitness 0 v generaci 15	fitness 0 v generaci 10
					10.	fitness 0 v generaci 4	fitness 0 v generaci 41

Tab. 6. Experiment metodou 10 pokusů – úloha 2.

úloha					pokus	lback="1 sloupec"	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 nedosaženo	fitness 0 nedosaženo
0	0	0	0	1 ▾	2.	fitness 0 v generaci 30	fitness 0 nedosaženo
1	0	0	1	0 ▾	3.	fitness 0 nedosaženo	fitness 0 nedosaženo
2	0	1	0	1 ▾	4.	fitness 0 v generaci 86	fitness 0 nedosaženo
3	0	1	1	0 ▾	5.	fitness 0 nedosaženo	fitness 0 nedosaženo
4	1	0	0	0 ▾	6.	fitness 0 nedosaženo	fitness 0 nedosaženo
5	1	0	1	1 ▾	7.	fitness 0 v generaci 36	fitness 0 nedosaženo
6	1	1	0	1 ▾	8.	fitness 0 nedosaženo	fitness 0 nedosaženo
7	1	1	1	0 ▾	9.	fitness 0 nedosaženo	fitness 0 nedosaženo
					10.	fitness 0 nedosaženo	fitness 0 nedosaženo

Tab. 7. Experiment metodou 10 pokusů – úloha 3.

úloha					pokus	lback="1 sloupec"	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 19	fitness 0 nedosaženo
0	0	0	0	0 ▾	2.	fitness 0 v generaci 3	fitness 0 v generaci 29
1	0	0	1	0 ▾	3.	fitness 0 nedosaženo	fitness 0 v generaci 24
2	0	1	0	1 ▾	4.	fitness 0 v generaci 159	fitness 0 nedosaženo
3	0	1	1	1 ▾	5.	fitness 0 nedosaženo	fitness 0 v generaci 107
4	1	0	0	1 ▾	6.	fitness 0 v generaci 130	fitness 0 nedosaženo
5	1	0	1	1 ▾	7.	fitness 0 nedosaženo	fitness 0 nedosaženo
6	1	1	0	1 ▾	8.	fitness 0 v generaci 83	fitness 0 nedosaženo
7	1	1	1	0 ▾	9.	fitness 0 nedosaženo	fitness 0 v generaci 7
					10.	fitness 0 v generaci 83	fitness 0 nedosaženo

Tab. 8. Experiment metodou 10 pokusů – úloha 4.

úloha					pokus	lback="1 sloupec"	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 2	fitness 0 v generaci 3
0	0	0	0	1 ▾	2.	fitness 0 v generaci 2	fitness 0 v generaci 4
1	0	0	1	1 ▾	3.	fitness 0 v generaci 2	fitness 0 v generaci 4
2	0	1	0	1 ▾	4.	fitness 0 v generaci 2	fitness 0 v generaci 3
3	0	1	1	1 ▾	5.	fitness 0 v generaci 2	fitness 0 v generaci 8
4	1	0	0	1 ▾	6.	fitness 0 v generaci 2	fitness 0 v generaci 8
5	1	0	1	1 ▾	7.	fitness 0 v generaci 2	fitness 0 v generaci 20
6	1	1	0	1 ▾	8.	fitness 0 v generaci 2	fitness 0 v generaci 28
7	1	1	1	1 ▾	9.	fitness 0 v generaci 2	fitness 0 v generaci 2
					10.	fitness 0 v generaci 2	fitness 0 v generaci 2

Tab. 9. Experiment metodou 10 pokusů – úloha 5.

úloha					pokus	lback="1 sloupec"	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 2	fitness 0 nedosaženo
0	0	0	0	1 ▾	2.	fitness 0 nedosaženo	fitness 0 nedosaženo
1	0	0	1	0 ▾	3.	fitness 0 v generaci 159	fitness 0 v generaci 74
2	0	1	0	1 ▾	4.	fitness 0 v generaci 86	fitness 0 nedosaženo
3	0	1	1	0 ▾	5.	fitness 0 nedosaženo	fitness 0 nedosaženo
4	1	0	0	1 ▾	6.	fitness 0 nedosaženo	fitness 0 nedosaženo
5	1	0	1	0 ▾	7.	fitness 0 v generaci 112	fitness 0 nedosaženo
6	1	1	0	0 ▾	8.	fitness 0 nedosaženo	fitness 0 nedosaženo
7	1	1	1	1 ▾	9.	fitness 0 nedosaženo	fitness 0 nedosaženo
					10.	fitness 0 v generaci 32	fitness 0 v generaci 91

8.2 Výsledky experimentů – metoda 10 výsledků s fitness 0

Zde zkoumám počet pokusů nutných k dosažení deseti výsledků konkrétních úloh při parametru $generace=500$.

Tab. 10. Experiment metodou 10 výsledků s fitness 0 – úloha 1.

úloha					pokus	lback="1 sloupec"	pokus	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 2	1.	fitness 0 v generaci 71
0	0	0	0	1	2.	fitness 0 v generaci 59	2.	fitness 0 v generaci 39
1	0	0	1	0	3.	fitness 0 v generaci 4	3.	fitness 0 v generaci 29
2	0	1	0	1	4.	fitness 0 v generaci 8	4.	fitness 0 v generaci 47
3	0	1	1	0	5.	fitness 0 v generaci 9	5.	fitness 0 v generaci 15
4	1	0	0	0	6.	fitness 0 v generaci 23	6.	fitness 0 v generaci 67
5	1	0	1	0	7.	fitness 0 v generaci 7	7.	fitness 0 v generaci 17
6	1	1	0	0	8.	fitness 0 v generaci 5	8.	fitness 0 v generaci 62
7	1	1	1	0	9.	fitness 0 v generaci 8	9.	fitness 0 v generaci 19
					10.	fitness 0 v generaci 8	10.	fitness 0 v generaci 90

Tab. 11. Experiment metodou 10 výsledků s fitness 0 – úloha 2.

úloha					pokus	lback="1 sloupec"	pokus	lback="všechny sloupce"
	X0	X1	X2	Y	2.	fitness 0 v generaci 42	>30.	fitness 0 nedosaženo
0	0	0	0	1	9.	fitness 0 v generaci 124		
1	0	0	1	0	11.	fitness 0 v generaci 304		
2	0	1	0	1	15.	fitness 0 v generaci 462		
3	0	1	1	0	16.	fitness 0 v generaci 154		
4	1	0	0	0	22.	fitness 0 v generaci 79		
5	1	0	1	1	25.	fitness 0 v generaci 334		
6	1	1	0	1	26.	fitness 0 v generaci 98		
					34.	fitness 0 v generaci 103		
					43.	fitness 0 v generaci 25		

Tab. 12. Experiment metodou 10 výsledků s fitness 0 – úloha 3.

úloha					pokus	lback="1 sloupec"	pokus	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 45	5.	fitness 0 v generaci 102
0	0	0	0	0	2.	fitness 0 v generaci 12	7.	fitness 0 v generaci 497
1	0	0	1	0	5.	fitness 0 v generaci 39	9.	fitness 0 v generaci 102
2	0	1	0	1	6.	fitness 0 v generaci 453	12.	fitness 0 v generaci 91
3	0	1	1	1	7.	fitness 0 v generaci 263	17.	fitness 0 v generaci 213
4	1	0	0	1	8.	fitness 0 v generaci 383	23.	fitness 0 v generaci 291
5	1	0	1	1	10.	fitness 0 v generaci 278	24.	fitness 0 v generaci 53
6	1	1	0	1	11.	fitness 0 v generaci 129	25.	fitness 0 v generaci 165
7	1	1	1	0	12.	fitness 0 v generaci 361	28.	fitness 0 v generaci 247
					13.	fitness 0 v generaci 115	32.	fitness 0 v generaci 11

Tab. 13. Experiment metodou 10 výsledků s fitness 0 – úloha 4.

úloha					pokus	lback="1 sloupec"	pokus	lback="všechny sloupce"
	X0	X1	X2	Y	1.	fitness 0 v generaci 2	1.	fitness 0 v generaci 2
0	0	0	0	1	2.	fitness 0 v generaci 4	2.	fitness 0 v generaci 24
1	0	0	1	1	3.	fitness 0 v generaci 2	3.	fitness 0 v generaci 3
2	0	1	0	1	4.	fitness 0 v generaci 2	4.	fitness 0 v generaci 4
3	0	1	1	1	5.	fitness 0 v generaci 2	5.	fitness 0 v generaci 12
4	1	0	0	1	6.	fitness 0 v generaci 2	6.	fitness 0 v generaci 6
5	1	0	1	1	7.	fitness 0 v generaci 2	7.	fitness 0 v generaci 31
6	1	1	0	1	8.	fitness 0 v generaci 2	8.	fitness 0 v generaci 3
7	1	1	1	1	9.	fitness 0 v generaci 2	9.	fitness 0 v generaci 8
					10.	fitness 0 v generaci 2	10.	fitness 0 v generaci 6

Tab. 14. Experiment metodou 10 výsledků s fitness 0 – úloha 5.

úloha					pokus	lback="1 sloupec"	pokus	lback="všechny sloupce"
	X0	X1	X2	Y	2.	fitness 0 v generaci 38	8.	fitness 0 v generaci 66
0	0	0	0	1	3.	fitness 0 v generaci 332	12.	fitness 0 v generaci 366
1	0	0	1	0	4.	fitness 0 v generaci 18	23.	fitness 0 v generaci 249
2	0	1	0	1	5.	fitness 0 v generaci 30	24.	fitness 0 v generaci 84
3	0	1	1	0	7.	fitness 0 v generaci 257	46.	fitness 0 v generaci 212
4	1	0	0	1	8.	fitness 0 v generaci 13	53.	fitness 0 v generaci 160
5	1	0	1	0	9.	fitness 0 v generaci 214	>60.	
6	1	1	0	0	10.	fitness 0 v generaci 136		
7	1	1	1	1	11.	fitness 0 v generaci 98		
					12.	fitness 0 v generaci 87		

8.3 Zhodnocení výsledků systému CGP synthesis

Na základě výše uvedených experimentů **docházím k následujícím závěrům:**

- Vytvořený systém **funguje v pořádku**, vhodné řešení, vyhovující pravdivostní tabulce, vždy našel (ač někdy ne hned na první spuštění).
- **Počet generací** nutných k nalezení řešení s fitness 0 **se odvíjí podle zadaného problému a počáteční vygenerované populaci**. V některých případech vystačíme s první desítkou generací, jindy jich potřebujeme stovky (*tab. 12. vs tab. 13.*).
- Parametr **lback** **značně ovlivňuje efektivitu** systému. Očekával jsem lepší výsledky s parametrem *lback*="všechny sloupce" pro větší variabilitu při sestavování obvodu, experimenty však poukázaly na odlišnou skutečnost. Při *lback*="1 sloupec" systém našel dříve vhodné řešení a uspěl i tehdy, když varianta s *lback*="všechny sloupce" selhala (*tab. 11.*). Domnívám se, že je to způsobeno systematičností, kdy se sestavení výsledného obvodu účastní každý sloupec maticového schématu.
- **Doba běhu** spuštěného programu (po kliknutí na tlačítko „SPUSTIT“) do zobrazení řešení je úměrně **závislá na nastavené hodnotě počtu generací** a na generaci, ve které byl nalezen jedinec s fitness 0. Stane-li se tak dříve, než v zadané maximální generaci, program se v aktuální generaci ukončí – tzn. výpočetní doba se zkrátí.
- **Navolení vyššího počtu generací zvyšuje šanci** na nalezení vhodného jedince.

ZÁVĚR

Hlavním cílem diplomové práce bylo vytvořit v prostředí Wolfram Mathematica funkční systém pro návrh kombinačních obvodů s přehledným zadáváním vstupních parametrů a grafickým vyjádřením nalezeného řešení (nejlépe formou obvodového schématu) v rámci jediné spuštěné aplikace. **Tento cíl se mi podařilo splnit**, systém je přehledný a plně funkční (dosaženými výsledky se zabývá kapitola 8). V některých případech je sice potřeba k nalezení vhodného řešení více pokusů, nicméně řešení se podaří nalézt vždy (platí především při parametru `lback="1 sloupec"`). Rovněž obvodové schéma je úspěšně implementováno, ač nevyužívá normovaných symbolů IEC.

Nepodařilo se mi vytvořit funkci, která by v obvodovém schématu zobrazila pouze akční hradla, mající vliv na výstupní hodnotu, z důvodu složitosti naprogramování a zřejmé následné robustnosti kódu. Zmíněná funkce měla sloužit pouze jako doplněk bez přímého vztahu k cíli práce. Také snaha o možnost spouštění systému ve zdarma dostupném software Wolfram CDF Player nevedla k dobrým výsledkům, jelikož uvedený přehrávač nepodporuje opětovnou evaluaci kódu, s kterou systém pracuje.

Dílní cíle definované v úvodu hodnotím taktéž jako splněné, práce obsahuje studii dosavadního i nového návrhového systému.

ZÁVĚR V ANGLIČTINĚ

The thesis main goal was to create the working combinatorial circuits synthesis system with Wolfram Mathematica environment. The system should have a transparent input parameters entering and graphical found solution (best as a circuit diagram) within one running application. **This goal has been accomplished**, the system is transparent and full working (results are presented in chapter 8). Although to find a suitable solution there is a need for more experiments in some cases, the solution has always been found (especially in case of `lback="1 column"` parameter). The circuit diagram function was also implemented successfully (without normed IEC symbols).

I failed a function that should show only action gates (with output effect) creating because of programming complexity and resultant robust source code. Mentioned function should be used only as an addon without a direct relation to the thesis goal. An effort to make a system running in free Wolfram CDF player software possibility has not tended to right resolutions too, because the software do not support a code reevaluation my CGP synthesis system works with.

The introduction sub-goals have been accomplished as well, the thesis contains an existing and new created design system study.

SEZNAM POUŽITÉ LITERATURY

- [1] RAM, B. *Computer fundamentals: architecture and organization*. Rev. 3rd ed. New Delhi: New Age International, 2000. ISBN 812241267X.
- [2] DOSTÁL, Tomáš. *Analogové elektronické obvody*. Brno, 2004. Dostupné z: http://dalkove2008-2013.wz.cz/ek_vut_brno.pdf. Učební text. VUT Brno.
- [3] SEKANINA, Lukáš. Evoluční návrh elektronických obvodů. *Automa: časopis pro automatizační techniku*. Praha: FCC Public, 2010, č. 1. ISSN 1210-9592. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=40365
- [4] DE MICHELI, Giovanni. *Two-level Logic Synthesis and Optimization* [online]. Lausanne [cit. 2012-03-12]. Dostupné z: <http://www.ece.iupui.edu/~johnlee/ECE495/TwoLevel.Exact.Minimization.ppt>. EPF.
- [5] BÍLEK, Jan. *Minimalizace neúplně určených logických funkcí pomocí modifikovaných binárních rozhodovacích diagramů*. Praha, 2007. Dostupné z: <http://service.felk.cvut.cz/vlsi/dip/Bilek07/dp-mbd.pdf>. Diplomová práce. ČVUT, FEL.
- [6] NIEDOBA, Pavel. *Minimalizace Booleových funkcí pomocí Quineovy McCluskeyovy metody*. Brno, 2010. Dostupné z: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=28728. Bakalářská práce. VUT Brno, Fakulta strojního inženýrství.
- [7] ŠEDA, Miloš. Heuristic Set-Covering-Based Postprocessing for Improving the Quine-McCluskey Method. In: *International Journal of Computational Intelligence Enformatika* [online]. 2007 [cit. 2012-03-13]. ISSN 1304-2386. Dostupné z: <http://www.waset.org/journals/waset/v29/v29-47.pdf>
- [8] ZELENKA, Jakub. *Vizualizace dvouúrovňové minimalizace logických funkcí*. Praha, 2010. Dostupné z: <https://service.felk.cvut.cz/vlsi/dip/Zelenka10/text/dp.pdf>. Diplomová práce. ČVUT v Praze, Fakulta elektrotechnická.
- [9] MEURER, Benedikt a BRÜCK. *Grundlagen und Verfahren für den Mikrosystementwurf* [online]. Siegen, 2008 [cit. 2012-03-14]. Dostupné z: <http://benediktmeurer.de/files/mse2008.pdf>. Universität Siegen.

- [10] BURIAN, Petr. Návrh číslicových obvodů za pomoci evolučních výpočetních technik. *Automatizace*. Praha: Automatizace, 2009, č. 3. ISSN 0005-125X. Dostupné z: <http://www.automatizace.cz/article.php?a=2484>
- [11] ZELINKA, Ivan, Zuzana OPLATKOVÁ, Miloš ŠEDA, Pavel OŠMERA a František VČELARĚ. *Evoluční výpočetní techniky: principy a aplikace*. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.
- [12] ŠIKULOVÁ, Michaela. *Symbolická regrese a koevoluce*. Brno, 2011. Symbolická regrese a koevoluce. Dostupné z: <http://www.fit.vutbr.cz/study/DP/rpfile.php.cs?id=7960&y=0>. Diplomová práce. VUT v Brně, FIT.
- [13] HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-2695-3.
- [14] KOZA, John R. *Genetic programming: On the programming of computers by means of natural selection*. Cambridge: Bradford Book, 1992, 818 s. ISBN 02-621-1170-5.
- [15] POLI, Riccardo, W LANGDON a Nicholas F MCPHEE. *A field guide to genetic programming*. [S.l.: Lulu Press], 2008, 233 s. ISBN 978-1-4092-0073-4. Dostupné z: http://www.lulu.com/items/volume_63/2167000/2167025/2/print/book.pdf
- [16] MILLER, J. F. *Cartesian genetic programming*. New York: Springer Berlin Heidelberg, c2011, 344 s. Natural computing series. ISBN 978-3-642-17309-7.
- [17] CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica*. Vyd. 1. Ve Zlíně: Univerzita Tomáše Bati ve Zlíně, 2005, 122 s. ISBN 80-7318-268-8.
- [18] *Wolfram Research: Mathematica, Technical and Scientific Software* [online]. 2012 [cit. 2012-03-27]. Dostupné z: <http://www.wolfram.com/>
- [19] VAŠÍČEK, Zdeněk a Lukáš SEKANINA. Nástroje pro kartézské genetické programování. *Fakulta informačních technologií VUT v Brně* [online]. 2008 [cit. 2011-09-30]. Dostupné z: http://www.fit.vutbr.cz/research/view_product.php?id=61¬itle=1

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ABC	Artificial Bee Colony
ANSI	American National Standards Institute
CDF	Computable Document Format
CGP	Cartesian Genetic Programming
ČSN	České technické Normy
DIN	Deutsches Institut für Normung
GP	Genetické programování
IEC	International Electrotechnical Commission
MBD	Minimum Bandwidth Drop
ROM	Read Only Memory
WM	Wolfram Mathematica

SEZNAM OBRÁZKŮ

Obr. 1. Systemizace kombinačních logických obvodů.	11
Obr. 2. Principiální rozdíl mezi kombinačním (vlevo) a sekvenčním (vpravo) obvodem.	11
Obr. 3. DOSTÁL, Tomáš. Řešení obvodů.	14
Obr. 4. Grafická konfrontace pohledů na klasifikaci metod syntézy.	15
Obr. 5. Ověření svého stanoviska.	17
Obr. 6. Operátory algoritmu Espresso.	19
Obr. 7. MEURER, Benedikt a BRÜCK. Der ESPRESSO Algorithmus (vlevo) a Der erweiterte ESPRESSO Algorithmus (vpravo).	19
Obr. 8. Princip operátoru EXPAND.	20
Obr. 9. Princip operátoru IRREDUNDANT.	20
Obr. 10. Princip operátoru REDUCE.	21
Obr. 11. Princip operátoru ESSENTIALS.	21
Obr. 12. ŠIKULOVÁ, Michaela. Obecný cyklus evolučního algoritmu.	23
Obr. 13. Algoritmus GP podle KOZY.	25
Obr. 14. Ukázka syntaktického stromu.	26
Obr. 15. Evoluční operátor křížení.	27
Obr. 16. Evoluční operátor mutace.	27
Obr. 17. Podoby genetického programování podle ZELINKY a kol.	28
Obr. 18. Ukázkový jedinec CGP.	28
Obr. 19. Přiřazení vektorů řádkového tvaru prvkům v maticovém tvaru.	29
Obr. 20. Interpretace členů vektoru do maticového tvaru.	29
Obr. 21. CGP – před mutací (vlevo) a po mutaci (vpravo) jedince.	29
Obr. 22. Prostředí Wolfram Mathematica 8 for Students.	30
Obr. 23. Nástroje TOOLS4CGP.	35
Obr. 24. TOOLS4CGP – převod pravdivostní tabulky (vlevo .txt, vpravo .h).	35
Obr. 25. TOOLS4CGP – parametry řídící běh evoluce.	36
Obr. 26. TOOLS4CGP – běžící modul pro implementaci CGP.	36
Obr. 27. TOOLS4CGP – nástroj cgpviewer po načtení jedince.	37
Obr. 28. TOOLS4CGP – demonstrace funkcí nástroje cgpviewer.	37
Obr. 29. CGP synthesis – náhled.	39
Obr. 30. CGP synthesis – instrukce pro spuštění.	40

Obr. 31. CGP synthesis – chybová zpráva.....	40
Obr. 32. CGP synthesis – náhled do uživatelského rozhraní.....	41
Obr. 33. CGP synthesis – význam parametru „počet generací“.....	42
Obr. 34. CGP synthesis – řádkové zobrazení nejlepšího jedince.....	42
Obr. 35. CGP synthesis – stromová struktura jedince.....	43
Obr. 36. CGP synthesis – grafické zobrazení průběhu evoluce.....	43
Obr. 37. CGP synthesis – grafické zobrazení nejlepšího jedince při parametru lback=“1 sloupec“ (nahore) a lback=“všechny sloupce“ (dole).....	44
Obr. 38. CGP synthesis – interpretace značení.....	44
Obr. 39. CGP synthesis – módy funkce „view“.....	45
Obr. 40. CGP synthesis – zdánlivá absence vstupu.....	46
Obr. 41. CGP synthesis – odhalení mystifikace posunutím.....	46
Obr. 42. CGP synthesis – kategorizovaný zdrojový kód.....	47

SEZNAM TABULEK

Tab. 1. Specifikace logického součinu.	12
Tab. 2. Specifikace negovaného logického součinu.	13
Tab. 3. Specifikace logického součtu.	13
Tab. 4. Specifikace negovaného logického součtu.	13
Tab. 5. Experiment metodou 10 pokusů – úloha 1.	49
Tab. 6. Experiment metodou 10 pokusů – úloha 2.	49
Tab. 7. Experiment metodou 10 pokusů – úloha 3.	50
Tab. 8. Experiment metodou 10 pokusů – úloha 4.	50
Tab. 9. Experiment metodou 10 pokusů – úloha 5.	50
Tab. 10. Experiment metodou 10 výsledků s fitness 0 – úloha 1.	51
Tab. 11. Experiment metodou 10 výsledků s fitness 0 – úloha 2.	51
Tab. 12. Experiment metodou 10 výsledků s fitness 0 – úloha 3.	52
Tab. 13. Experiment metodou 10 výsledků s fitness 0 – úloha 4.	52
Tab. 14. Experiment metodou 10 výsledků s fitness 0 – úloha 5.	52

SEZNAM PŘÍLOH

PI: CD

PŘÍLOHA P I: CD

Diplomová práce v elektronické podobě.

fulltext.pdf

Vytvořený program spustitelný v prostředí Wolfram Mathematica (vytvořen ve verzi 8).

CGP_synthesis.nb

Vytvořený program uložený ve standardu CDF, přizpůsobeném Wolfram přehrávačům³⁴.

CGP_synthesis.cdf

³⁴ Pro plnou funkcionalitu systému CGP synthesis **nepostačí** zdarma dostupný nástroj *Wolfram CDF Player*. Nutný placený produkt *Wolfram Mathematica*, popř. *Wolfram Player Pro*. Způsobeno interaktivním dynamickým obsahem. <http://www.wolfram.com/player-pro/how-player-pro-compares.html>