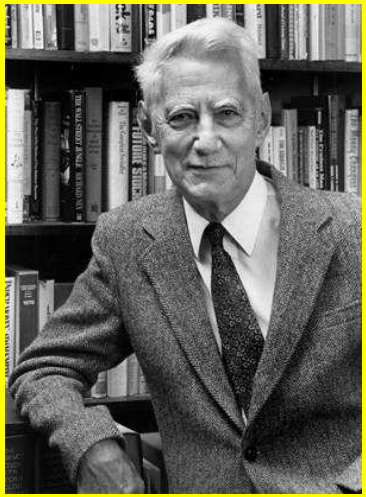
Symbolická regresia optimálnych Boolovych funkcií

Vlado Kvasnička
UAI FIIT STU

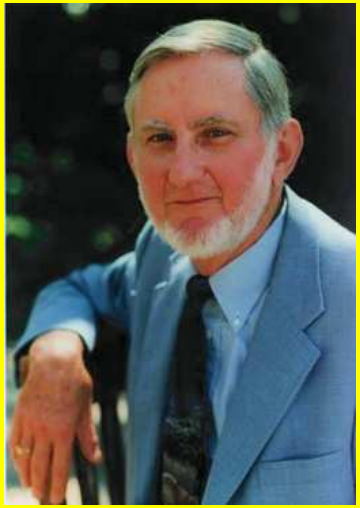
Význam výrokovej logiky v informatike

- Pre umelú inteligenciu a kognitívnu vedu tvorí teoretický základ pre algoritmizáciu procesov usudzovania.
- Pre informatiku (computer science) tvorí formálny základ pre „***logic design***“ a-alebo „***switching theory***“.
- .....

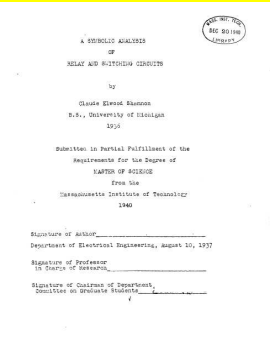


Claude Shannon (*1916 – †2001)

Willard Van Orman Quine
(*1908 – †2000)



Edward J. McCluskey (*1929)



A SYMBOLIC ANALYSIS
OF
RELAY AND SWITCHING CIRCUITS

by

Claude Elwood Shannon
B.S., University of Michigan
1936

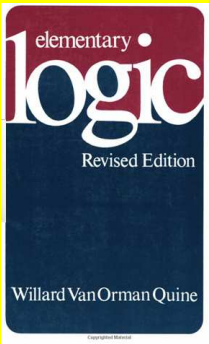
Submitted in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
from the
Massachusetts Institute of Technology
1940

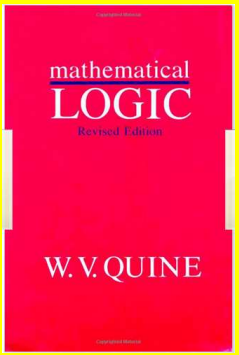
Signature of Author
Department of Electrical Engineering, August 10, 1937

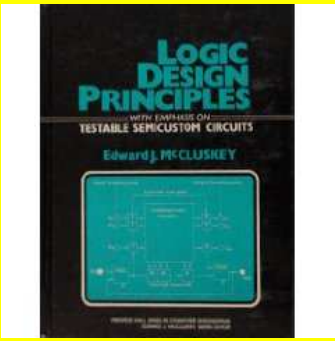
Signature of Professor
in Charge of Research

Signature of Chairman of Department,
Committee on Graduate Students

Prvá stránka MSc. Thesis
z r. 1937







Priesvitka 3

Boolove funkcie

Boolova funkcia je definovaná ako binárne zobrazenie

$$f:\{0,1\}^n \rightarrow \{0,1\}^m$$

kde

$$\left(\underbrace{(y_1, y_2, \ldots, y_n)}_{\boldsymbol{y}} = f\underbrace{(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m})}_{\boldsymbol{x}}\right) \equiv \left(\boldsymbol{y} = f(\boldsymbol{x})\right)$$

Táto Boolova funkcia je zadaná tabuľkou funkčných hodnôt

	Argumenty				*funkčné hodnoty*			
#	x_1	X_2	...	x_n	y_1	y_2	...	y_m
1	0	0	...	0	0	1	...	1
2	0	0	...	1	1	0	...	0
...	...	...	...	...	...	...	...	...
2^n-1	0	1	...	1	1	1	...	0
2^n	1	1		1	0	1	...	0

Štandardný prístup k riešeniu problému „syntézy“ Boolovych funkcii je založený na použití formúl

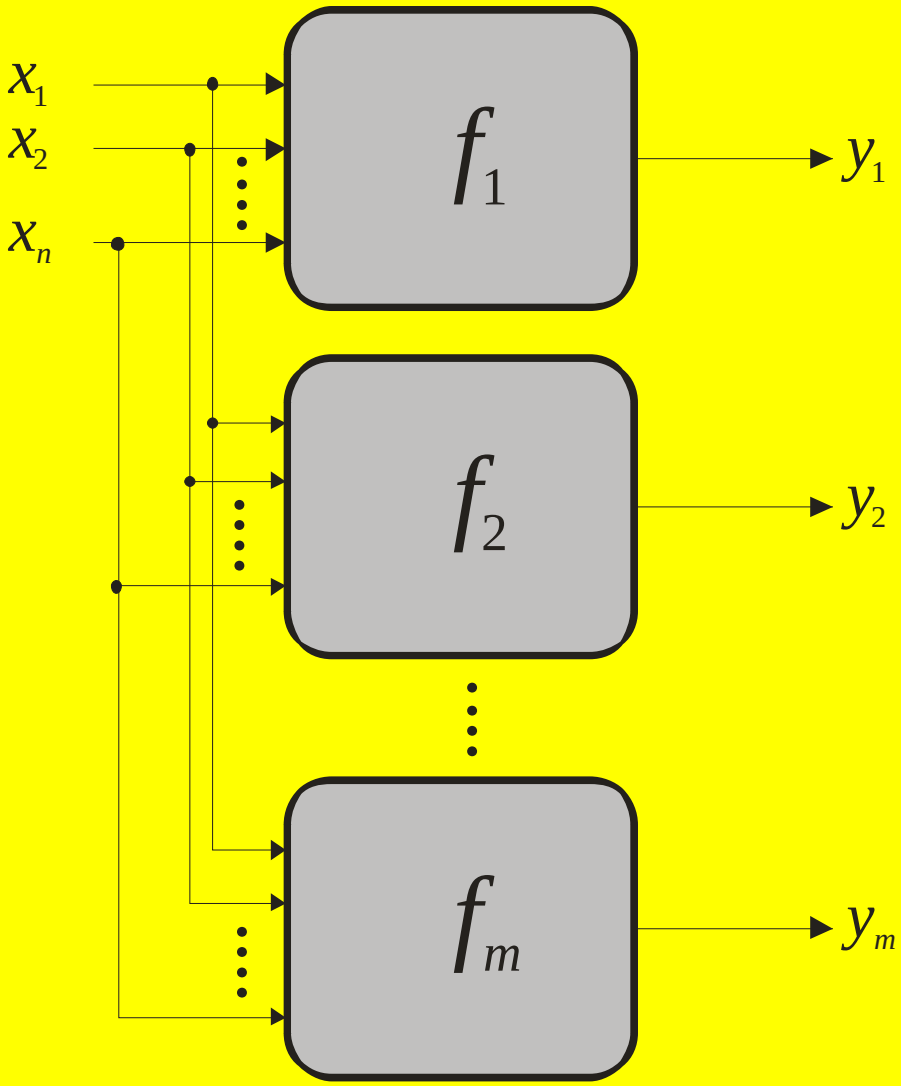
$$\varphi_{DNF} = \bigvee_{\substack{\tau \in \{0,1\}^n \\ (val_\tau(\varphi)=1)}} x_1^{(\tau_1)} \wedge x_2^{(\tau_2)} \wedge \ldots \wedge x_n^{(\tau_n)}$$

$$\varphi_{KNF} = \bigwedge_{\substack{\tau \in \{0,1\}^n \\ (val_\tau(\varphi)=0)}} x_1^{(1-\tau_1)} \vee x_2^{(1-\tau_2)} \vee \ldots \vee x_n^{(1-\tau_n)}$$

$$\varphi_{KNF}(x_1, x_2, \ldots, x_n) = \neg\varphi_{DNF}(\neg x_1, \neg x_2, \ldots, \neg x_n)$$

kde $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_n)$ a

$$x_i^{(\tau')} = \begin{cases} x_i & (\text{ak } \tau' = 1) \\ \neg x_i & (\text{ak } \tau' = 0) \end{cases}$$



Použitie Boolovych funkcií v tvare DNF **obvykle** produkuje ich striktnú blokovú štruktúru, ktorá sa neprekrýva

Priesvitka 6

Ilustračný príklad (dvojitý sumátor)

$$x_1 + x_2 + x_3 = y_1 y_2$$

#	x_1	x_2	x_3	y_1	y_2
1	0	0	0	0	0
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	1	0
5	1	0	0	0	1
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	1	1

Úloha: Hľadáme také Boolove funkcie, ktoré reprodukujú túto tabuľku

Boolove funkcie v **DNF** tvare

$$y_1 = (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3)$$
$$= \overline{x}_1 x_2 x_3 + x_1 \overline{x}_2 x_3 + x_1 x_2 \overline{x}_3 + x_1 x_2 x_3$$

$$y_2 = (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3)$$
$$= \overline{x}_1 \overline{x}_2 x_3 + \overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 \overline{x}_3 + x_1 x_2 x_3$$

Boolove funkcie v **KNF** tvare

$$y_1 = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$
$$= (x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + x_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)$$

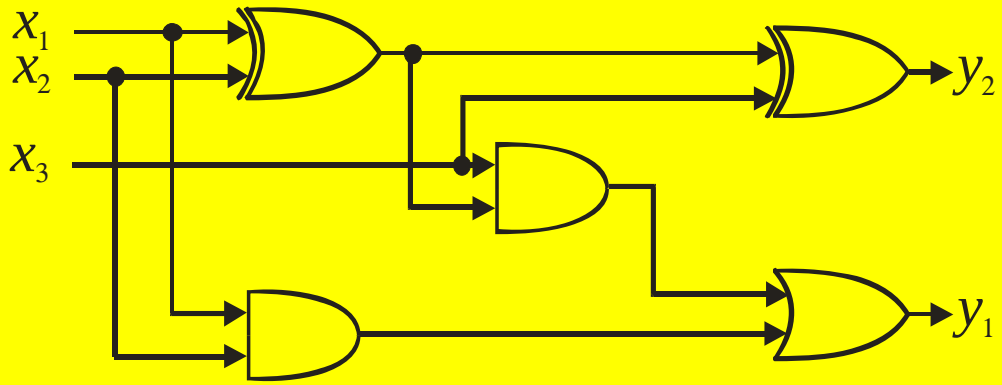
$$y_2 = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$
$$= (x_1 + x_2 + \overline{x}_3)(x_1 + \overline{x}_2 + x_3)(\overline{x}_1 + x_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)$$

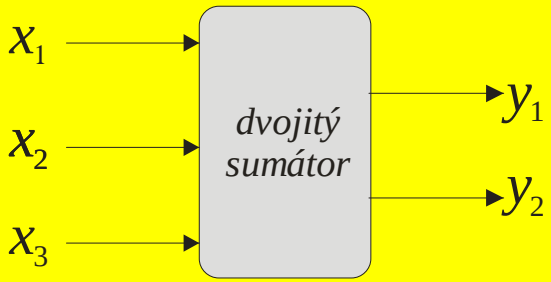
Poznámka: Optimalizačná metóda **Quine & McCluskey** je v tomto prípade nepoužiteľná.

$$y_1 = \overline{x}_1 x_2 x_3 + x_1 \overline{x}_2 x_3 + x_1 x_2 \overline{x}_3 + x_1 x_2 x_3 = \left(x_1 \oplus x_2 \right) x_3 + x_1 x_2$$

$$y_2 = \overline{x}_1 \overline{x}_2 x_3 + \overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 \overline{x}_3 + x_1 x_2 x_3$$

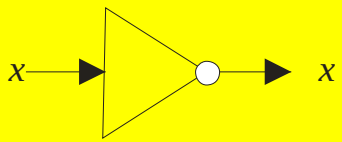
$$= \underbrace{\left(\overline{x}_1 x_2 + x_1 \overline{x}_2 \right)}_{x_1 \oplus x_2} \overline{x}_3 + \underbrace{\left(\overline{x}_1 \overline{x}_2 + x_1 x_2 \right)}_{\overline{x_1 \oplus x_2}} x_3 = x_1 \oplus x_2 \oplus x_3$$



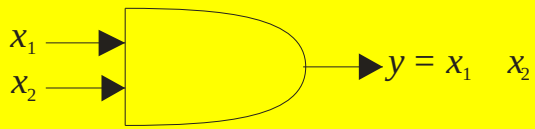


Elementárne Boolove funkcie – logické spojky

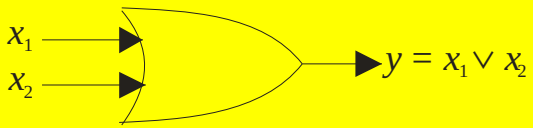
x_1	x_2	A $\neg x_1$	A $\neg x_2$	B $x_1 \wedge x_2$ (AND)	C $x_1 \vee x_2$ (OR)	D $x_1 \oplus x_2$ (XOR)	E $x_1 \uparrow x_2$ (NAND)	F $x_1 \downarrow x_2$ (NOR)
0	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0
1	0	0	1	0	1	1	1	0
1	1	0	0	1	1	0	0	0



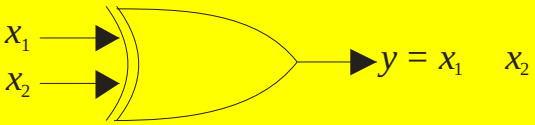
A (*brána negácie*)



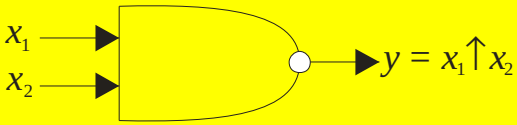
B (*brána AND*)



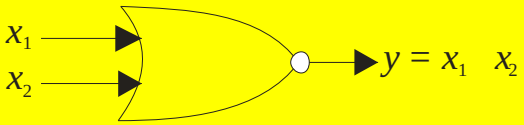
C (*brána OR*)



D (*brána XOR*)



E (*brána NAND*)



F (*brána NOR*)

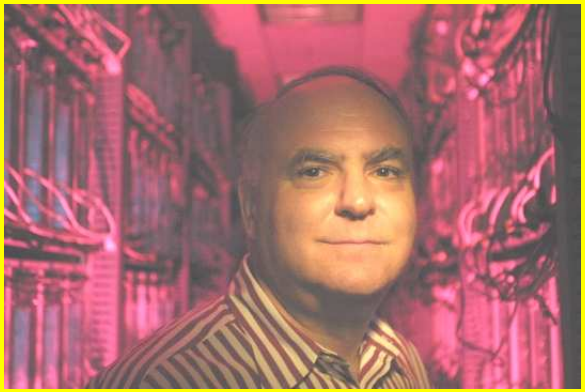
Poznámka

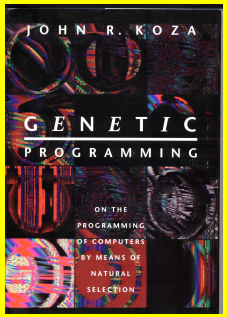
- V klasických prístupoch ku konštrukcii optimálnych Boolovych funkcií sa dajú používať len štandardné Boolove elementárne spojky negácie, konjunkcie a disjunkcie. Ďalšie „exotické“ spojky, akými sú napr. NAND, NOR a XOR sú v tomto prístupe len veľmi obťažne použiteľné.

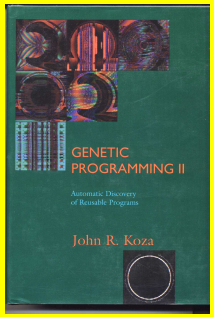
- Metafora – heuristika genetického programovania je veľmi flexibilným prístupom ku konštrukcii optimálnych Boolových funkcií vyjadrených pomocou týchto „exotických“ Boolových elementárnych spojok.

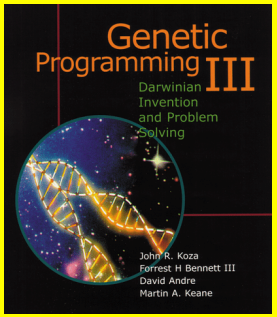
Čo je genetické programovanie?

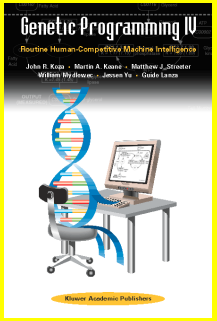
Tvorcom genetického programovania je americký informatik **John Koza** (University of Stanford)











Tieto knihy sú dostupné na ruských pirátskych internetovských portáloch

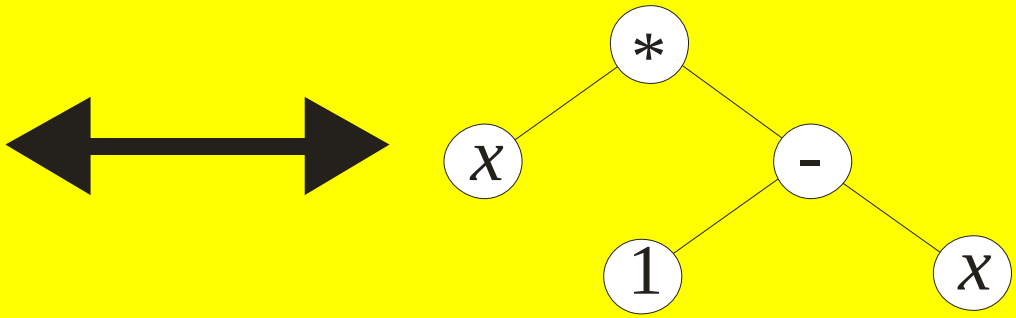
ktorý v r. 1988 podal žiadosť o patentovanie tejto metódy (Koza, John R. *Non-Linear Genetic Algorithms for Solving Problems*. US Patent 4,935,877)

Genetické programovanie je taká verzia genetického algoritmu, kde

(1) lineárne binárne chromozómy sú nahradené syntaktickými stromami rekurentné vypočítateľných funkcií,

(2) operácie mutácie a kríženia nad binárnymi reťazcami sú nahradené podobnými operáciami nad syntaktickými stromami

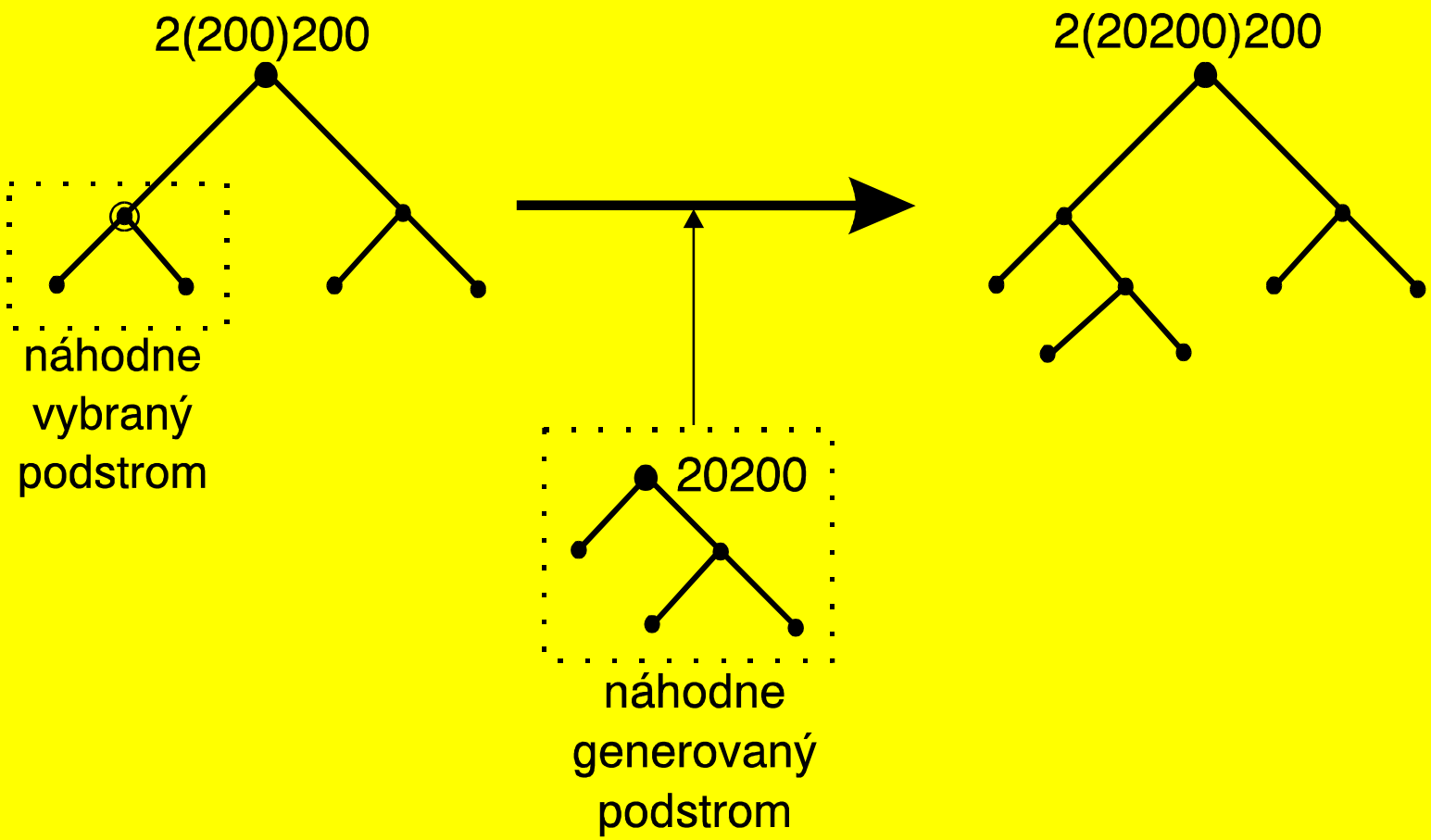
0011001111010101001 1



binárny chromozóm

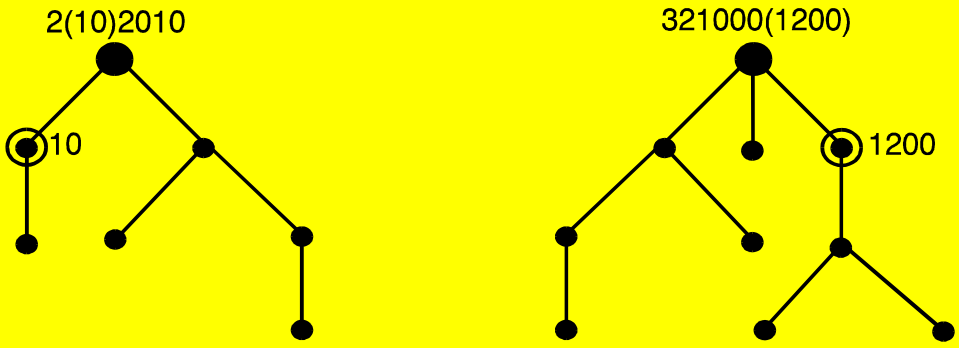
syntaktický strom
funkcie $f(x) = x*(1-x)$

Nutácia syntaktického stromu



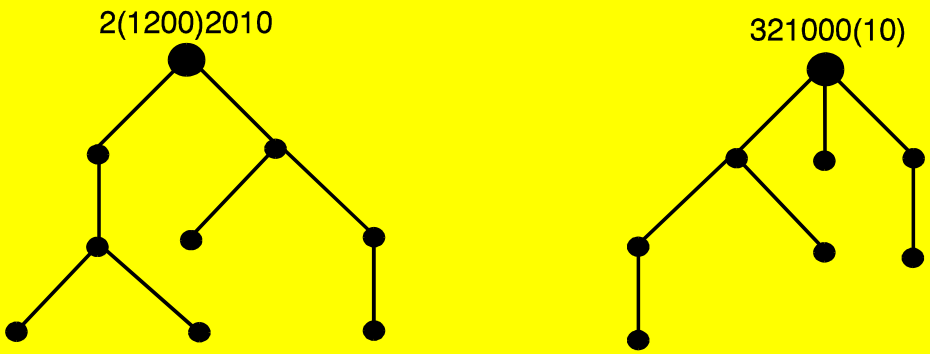
Kríženie dvoch syntaktických stromov

koreňové stromy - rodičia

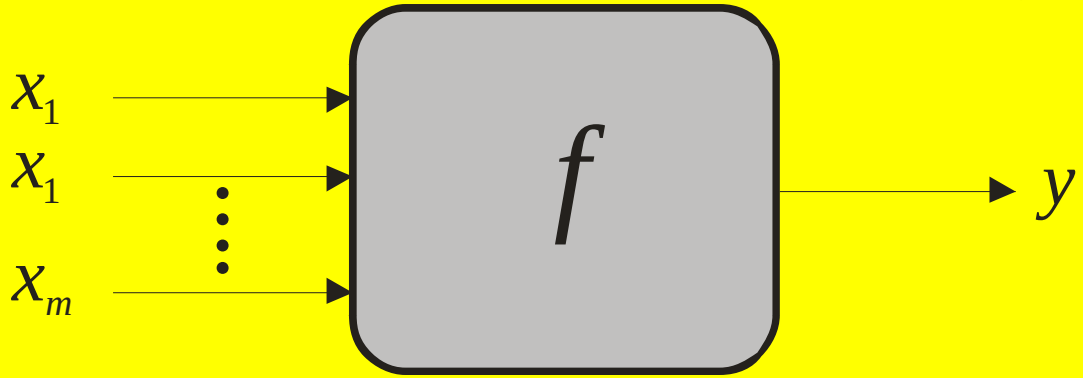


kríženie

koreňové stromy - potomkovia

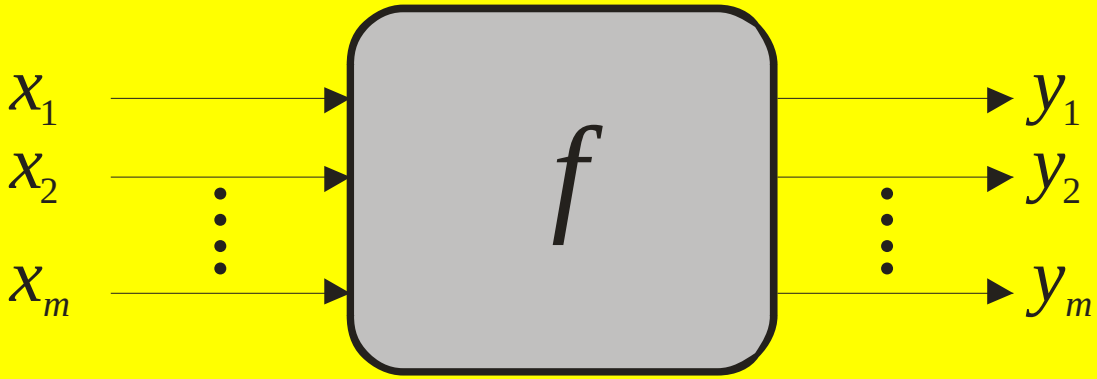


Poznámka: takto formulované genetické programovanie (podľa J. Kozu) je schopné zostrojiť len Také Boolové funkcie, ktoré majú jednu výstupnú hodnotu



Zadanie nášho problému je konštrukcia všeobecnej Boolovej funkcie typu

$$f : \{0,1\}^n \rightarrow \{0,1\}^m$$



Riešenie tohto problému je nahradiť koreňnové stromy acyklickými orientovanými grafmi

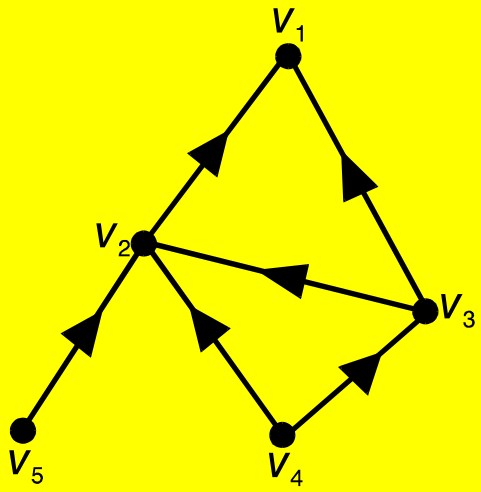
Indexovanie vrcholov acyklického orientovaného grafu

Definícia. ***Indexovanie*** vrcholov orientovaného grafu G je realizované pomocou zobrazenia φ, ktoré priradí 1-1-značne každému vrcholu celé číslo

$$\varphi : V \to \{1,2,\ldots,p\}$$

Celé číslo φ(v) sa nazýva ***index*** vrcholu v.

Veta. Orientovaný graf ***G***=(V,E) je ***acyklický*** vtedy a len vtedy, ak jeho vrcholy môžu byť indexované tak, že platí $\forall (v,v') \in E: \quad \varphi(v) < \varphi(v')$



Orientovaný graf $G=(V,E)$ obsahujúci päť vrcholov a šesť orientovaných hrán

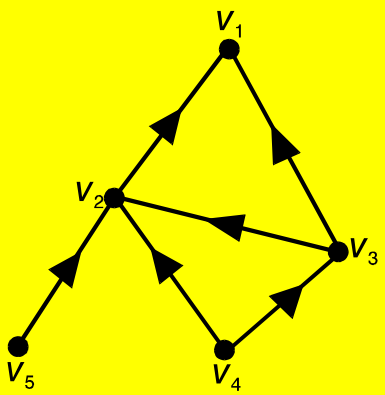
- Množina vrcholov V obsahuje päť vrcholov, $V=\{v_1,v_2,v_3,v_4,v_5\}$, množina hrán obsahuje šesť orientovaných hrán, $E=\{e_1,e_2,e_3,e_4,e_5,e_6\}$.
- Orientovaná $e\in E$ je určená pomocou usporiadanej dvojice dvoch vrcholov $v,v'\in E$, $e=(v,v')$.
- Hovoríme, že orientovaná hrana e začína v o vrchole v (hrana e vzchádza z vrcholu v) a končí vo vrchole v' (hrana e vchádza do vrcholu v').
- Graf je acyklický, neobsahuje cyklickú cestu, ktorá by sa skladala z postupnosti rovnako orientovaných hrán.

Vrcholy acyklického grafu sú špecifikované takto:

(a) ***vstupné vrcholy***, ktoré sú susedné len s vychádzajúcimi hranami ($val_{in} = 0$, $val_{out} > 0$),

(b) ***prechodné vrcholy***, ktoré sú susedné aspoň s jednou vychádzajúcou hranou a aspoň s jednou alebo dvomi hranami ($0 < val_{in} \leq 2$ a $val_{out} = 0$)

(c) ***výstupné vrcholy***, ktoré nie sú susedné so žiadnou vychádzajúcou hranou.



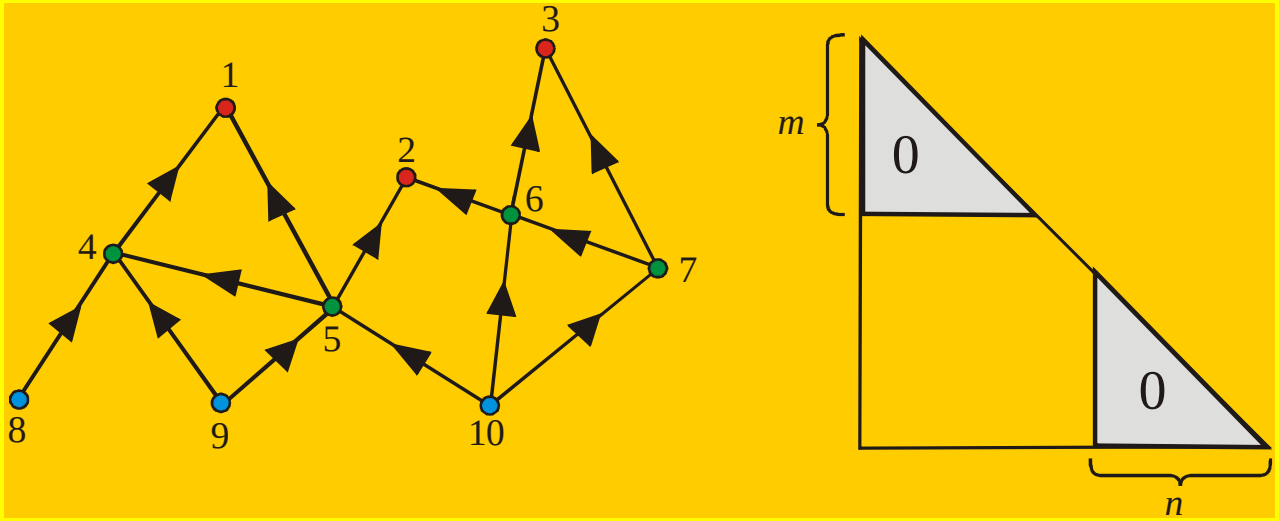
(1) Vstupné vrcholy, $V_{input} = \{4,5\}$

(2) Prechodné vrcholy, $V_{trans} = \{2,3\}$

(3) Výstupné vrcholy, $V_{output} = \{1\}$

Definícia. Indexovanie acyklického orientovaného grafu, ktorý obsahuje *m* vstupných vrcholov a *n* výstupných vrcholov sa nazýva ***kanonické*** vtedy a len vtedy, ak prvých *m* indexov je priradených výstupným vrcholom a posledných *n* indexov je priradených vstupným vrcholom.

Dôsledok: Ak acyklický graf má *m*(=3) výstupných vrcholov a *n*(=4) vstupných vrcholov , potom matica susednosti ***A*** v dolnom trojuholníku obsahuje v prvých *m* riadkoch a v posledných *n* stĺpcoch len nulové prvky



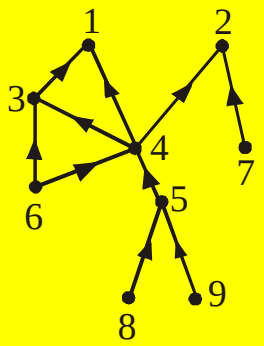
Boolova funkcia $f:\{0,1\}^n \rightarrow \{0,1\}^m$ môže byť vo všeobecnosti vyjadrená pomocou acyklického grafu, ktorý má viac ako jeden výstupných vrcholov.

Dôsledok: Pri aplikácii genetického programovania budeme hľadať optimálne riešenie Boolovej funkcie v tvare acyklického orientovaného grafu, ktorý obsahuje *m* vstupných a *n* výstupných vrcholov s kanonickou indexáciou vrcholov.

Základný predpoklad pre použitie genetického programovania pre konštrukciu optimálneho funkcionálneho tvaru danej Boolovej funkcie,

Kódovania acyklických orientovaných grafov pomocou stĺpcových tabuliek

Použitie acyklického orientovaného grafu



acyklický graf s kanonickým indexovaním

m=2

0								
0	0							
1	0	0						
1	1	1	0					
0	0	0	1	0				
0	0	1	0	0	0			
0	1	0	0	0	0	0		
0	0	0	0	1	0	0	0	
0	0	0	0	1	0	0	0	0

n=4

matica susednosti A

*polohy '1' elementov v danom stĺpci matice **A***

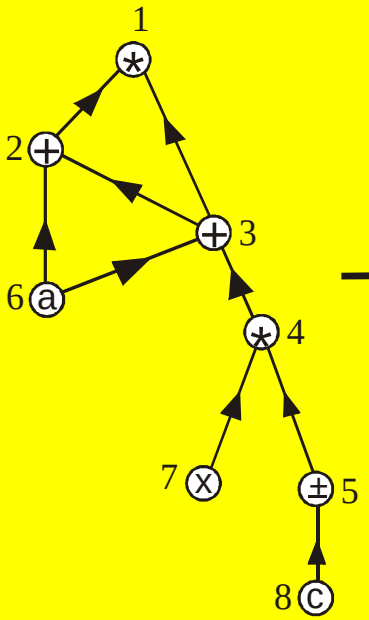
val_{in}		
2	3	4
2	4	7
2	4	6
1	5	0
2	8	9
0	0	0
0	0	0
0	0	0
0	0	0

stlpcová tabul ka

Základný objekt (chromozóm) genetického programovania

Kódovanie rekurentne vypočítateľnej funkcie

Pre účely symbolickej regresie, jednotlivé vrcholy acyklického grafu sú ohodnotené elementárnymi funkciami. Toto je realizované jednoduchým rozšírením stĺpcovej tabuľky o nový 0-tý stĺpec.

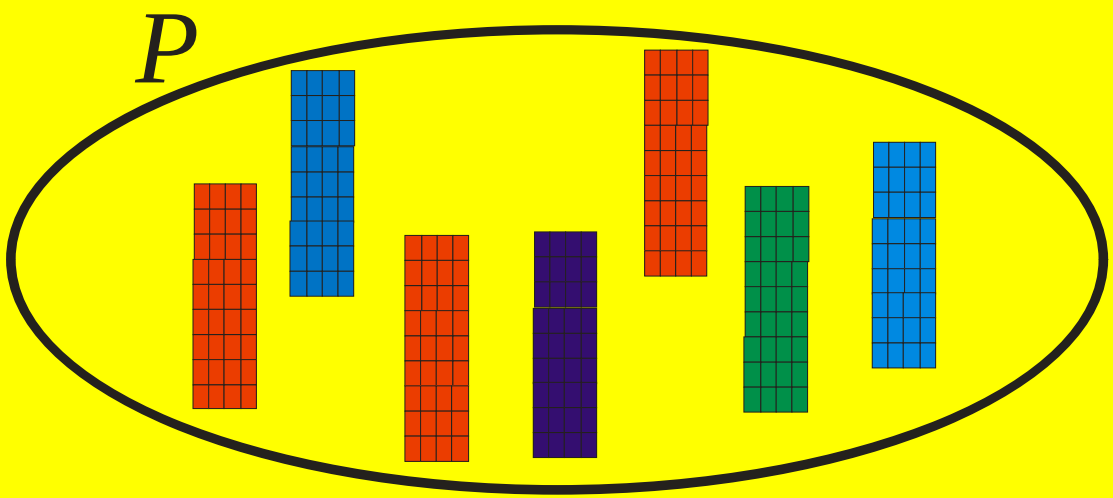


*	2	2	3
+	2	3	5
+	2	4	5
*	2	5	7
±	1	8	0
a	0	0	0
x	0	0	0
c	0	0	0

$x1=x2*x3=(2a+cx)(a+cx)$
$x2=x3+x6=(a+c*x)+a$
$x3=x6+x4=a+c*x$
$x4=x7*x5=c*x$
$x5=-c$
$x6=a$
$x7=-x$
$x8=c$

acyklický graf ***stĺpcová tabuľka*** ***rekurentná konštrukcia funkčných hodnôt***

Populácia v genetickom programovaní je tvorená multimnožinou chromozómov – rozšírených stĺpcových tabuliek



Ohodnotenie chromozómov fitness

- Nech $f:\{0,1\}^n \rightarrow \{0,1\}^m$ Boolova funkcia špecifikovaná tabuľkou je funkčných hodnôt

$$f_{tab} = \left\{ \boldsymbol{y}_{req}(\boldsymbol{x}) \in \{0,1\}^m ; \boldsymbol{x} \in \{0,1\}^n \right\}$$

t. j. pre každý binárny argument $\boldsymbol{x} \in \{0,1\}^n$ poznáme vektor funkčných hodnôt $\boldsymbol{y}(\boldsymbol{x}) \in \{0,1\}^m$ (t. j. požadované funkčné hodnoty).

- Nech $\boldsymbol{T} \in P$ je stĺpcová tabuľka z aktuálnej populácie P, Boolovu funkciu špecifikovanú touto tabuľkou označíme $f_{\boldsymbol{T}}$.

- Fitnes chromozómu - stĺpcovej tabuľky je špecifikované takto

$$E(\boldsymbol{T}) = \sum_{\boldsymbol{x}\in\{0,1\}^n} \left| f_{\boldsymbol{T}}(\boldsymbol{x}) - \boldsymbol{y}(\boldsymbol{x}) \right| + \alpha \left| \boldsymbol{T} \right|$$

$$fitness(\boldsymbol{T}) = \frac{1}{\omega + E(\boldsymbol{T})}$$

penalizačný člen, ktorý penalizuje tabuľku $\boldsymbol{T}$ za počet vrcholov, α je kladná penalizačná konštanta

ω male kladné číslo, ktoré špecifikuje max. hodnotu fitness

- Optimalizačný problém

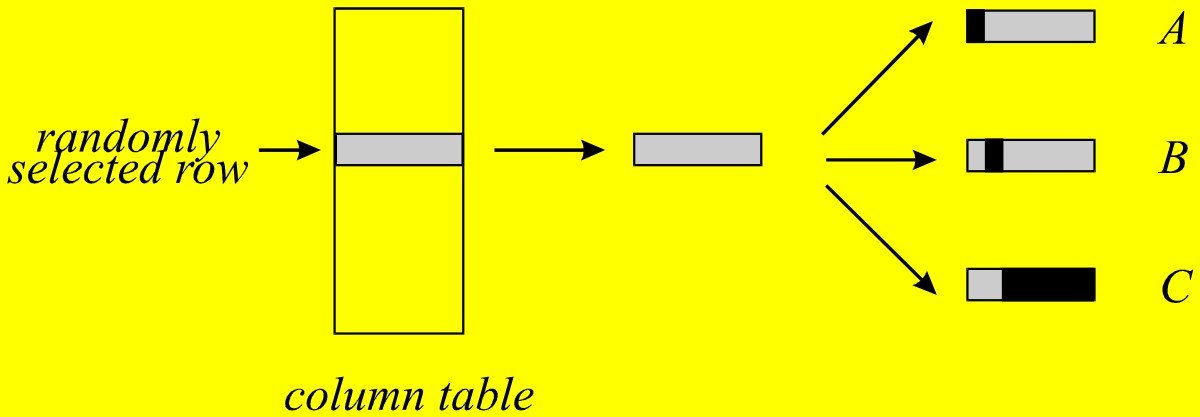
$$\boldsymbol{T}_{opt} = arg \min_{\boldsymbol{T}\in\mathcal{T}} E(\boldsymbol{T})$$

Mutation of column tables

A mutation is a stochastic operation that transforms a table onto another table

$$\boldsymbol{T}' = O_{mut}(\boldsymbol{T})$$

This operator is realized so that randomly selected row in $\boldsymbol{T}$ is randomly modified by one of the following three ways

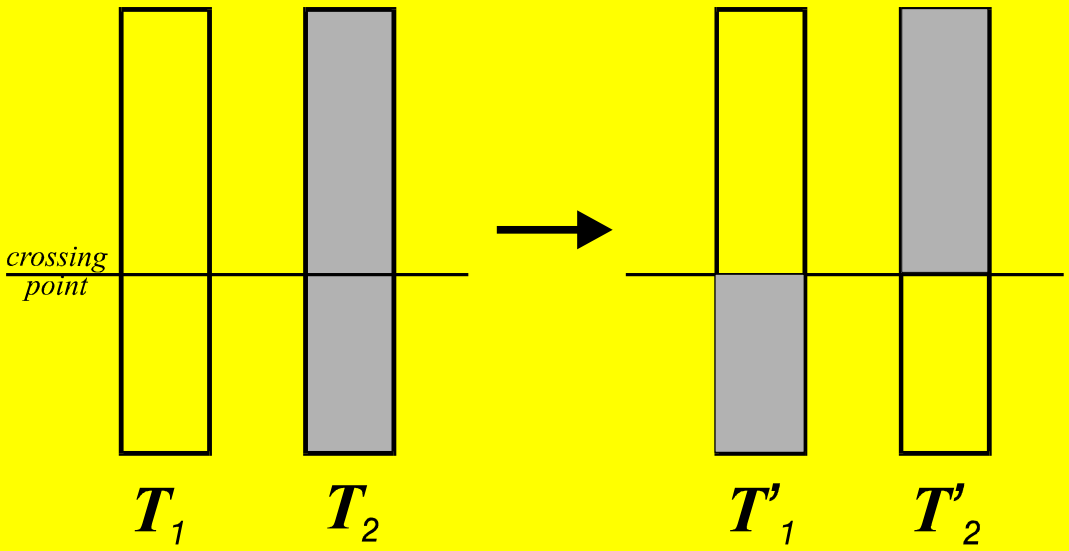


mode *A* : 0-th row entry is mutated, i.e. function is changed
mode *B* : 1-st row entry is mutated, i.e. number of predecessors is changed
mode *C* : randomly selected index of a predecessor is changed

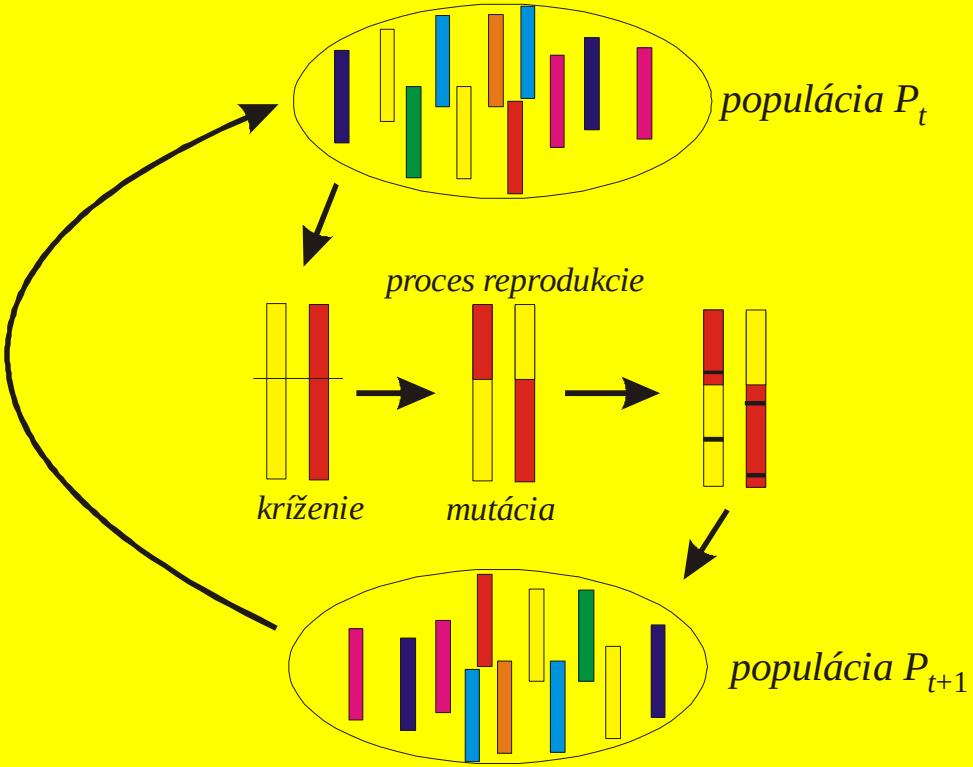
Crossover of two column tables

A crossover of two tables $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ is a stochastic transformation that creates two new tables $\boldsymbol{T}'_1$ and $\boldsymbol{T}'_2$

$$(\boldsymbol{T}_1', \boldsymbol{T}_2') = O_{cross}(\boldsymbol{T}_1, \boldsymbol{T}_2)$$



Schematické znázornenie genetického programovania s binárnou reprodukciou



Vychádza sa z aktuálnej populácie P_t , z ktorej opakovane vyberáme dvojice rodičovských replikátorov (pravdepodobnosť výberu je úmerná ich fitnes). V reprodukčnom procese ich replikácia je založená na tzv. krížení a mutácii. Takto vytvorené replikátory vytvárajú novú populáciu potomkov, ktorá, keď obsahuje rovnaký počet replikátorov ako rodičovská populácia, sa osamostatní a podobný proces reprodukcie sa opakuje.

Preliminary conclusions

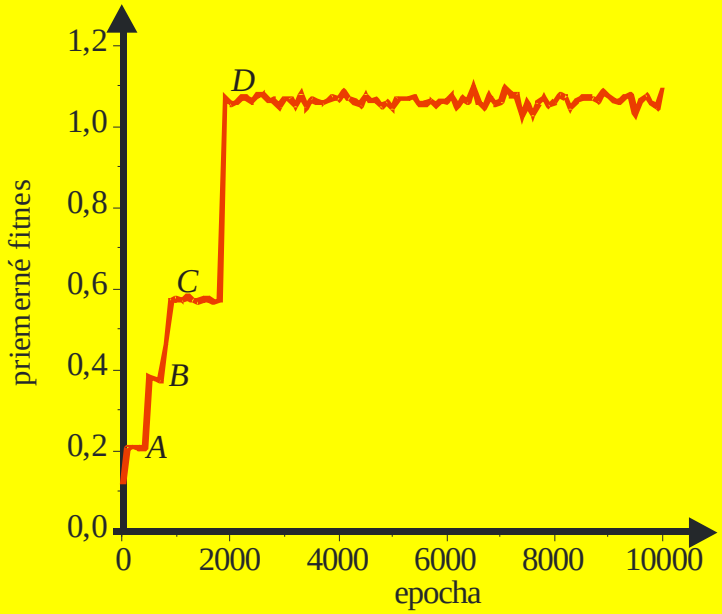
- Column tables offers simple formal tool how to **code functions** that are expressible through simple elementary unary or binary functions (negation, conjuctio, disjunction, XOR, NAND, NOR,…).
- Simple stochastic operation (mutation and crossover) over column tables are naturally determined.
- The method is straightforwardly implemented for broad class of stochastic and/or evolutionary algorithms.
- Column tables also code feed-forward neural networks, therefore a combination of the coding with an evolutionary algorithm might be of interesting for design of topology of neural networks.

Prvý ilustračný príklad – dvojitý sumátor

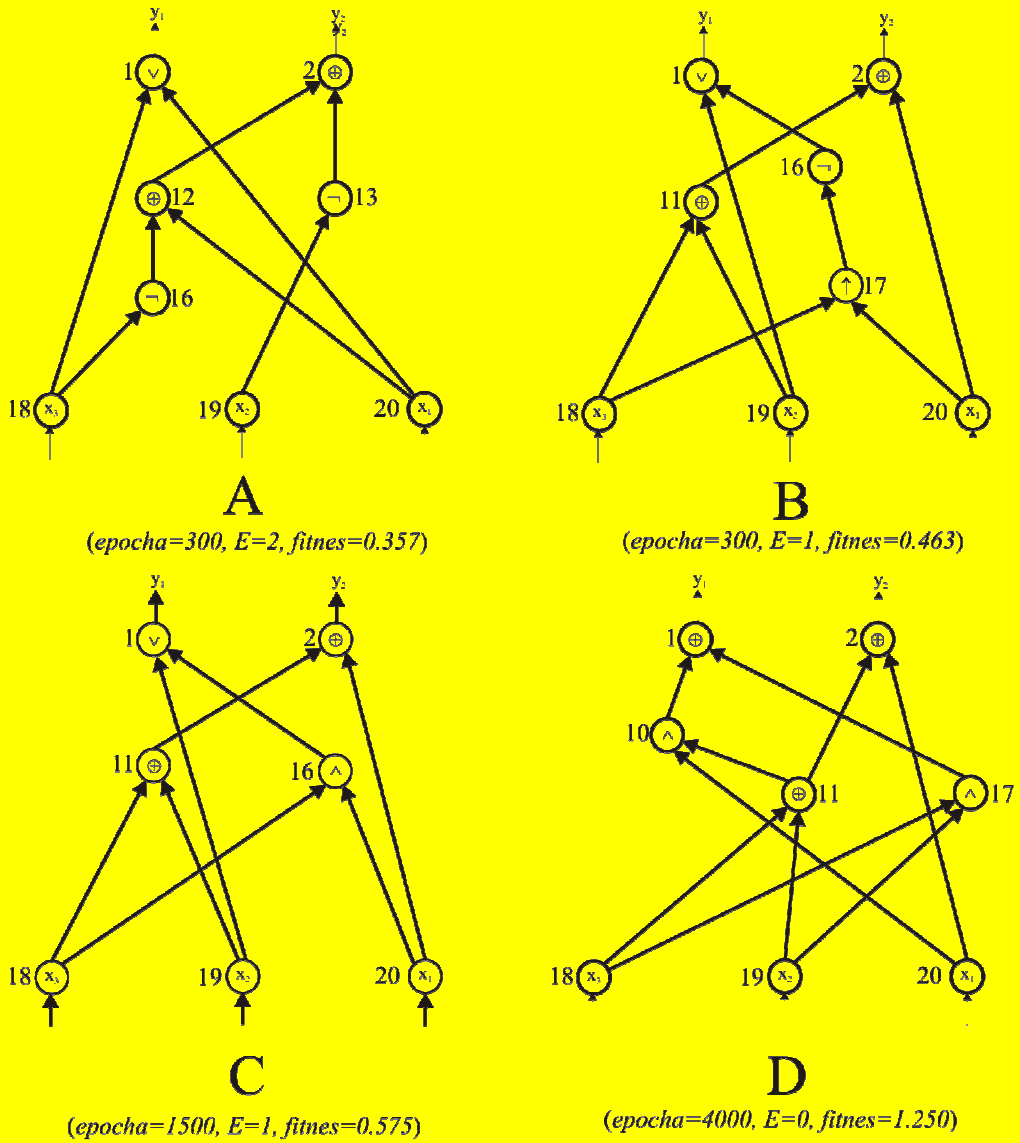
$$x_1 + x_2 + x_3 = y_1 y_2$$

#	x_1	x_2	x_3	y_1	y_2
1	0	0	0	0	0
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	1	0
5	1	0	0	0	1
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	1	1

Základné paramtre GP *parameter*	*hodnota*
veľkosť populácie P	400
pravdepodobnosť mutácie P_{mut}	0.01
pravdepodobnosť kríženia P_{cross}	0.7
Maximálna dĺžka tabuľky p_{max}	2
penalizačný parameter ω	0.1
maximálny počet epoch	3×10^4



Priebeh funkcie fitnes pri symbolickej regresii obvodu dvojitého sumátora pomocou genetickom programovaní

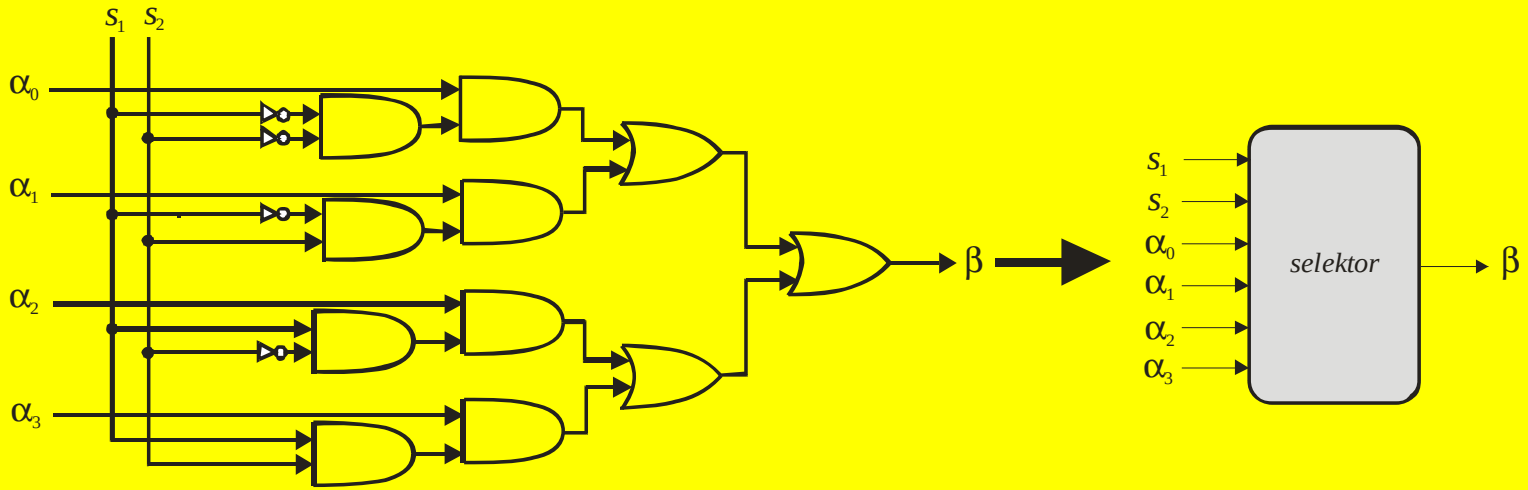


Reprezentanti jednotlivých etáp genetického programovania. Finálny diagram D obsahuje 5 funkčných vrcholov a tri vstupné vrcholy.

Druhý ilustračný príklad – multiplexor

Zariadenie nazývane *multiplexor* zo štyroch možných vstupov $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ vyberá na výstup práve jedno v závislosti od riadiacich vstupov s_1 a s_2

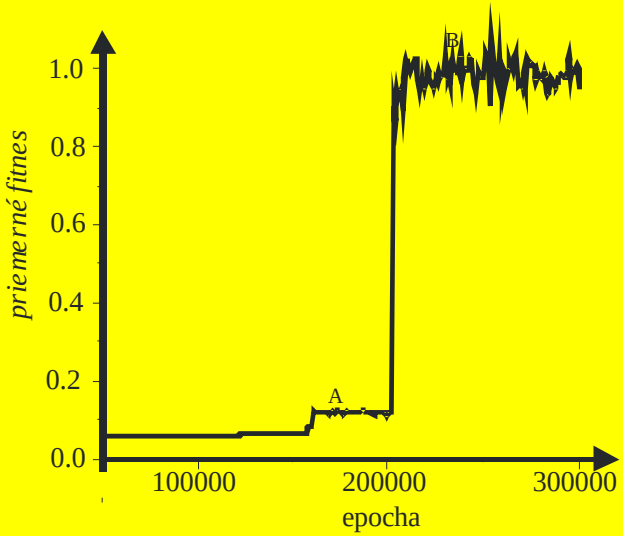
$$\beta = f\left(s_1, s_2; \alpha_0, \alpha_1, \alpha_2, \alpha_3\right) = \beta$$
$$\beta = \alpha_i \quad \left(\text{kde } i = 2s_1 + s_2\right)$$



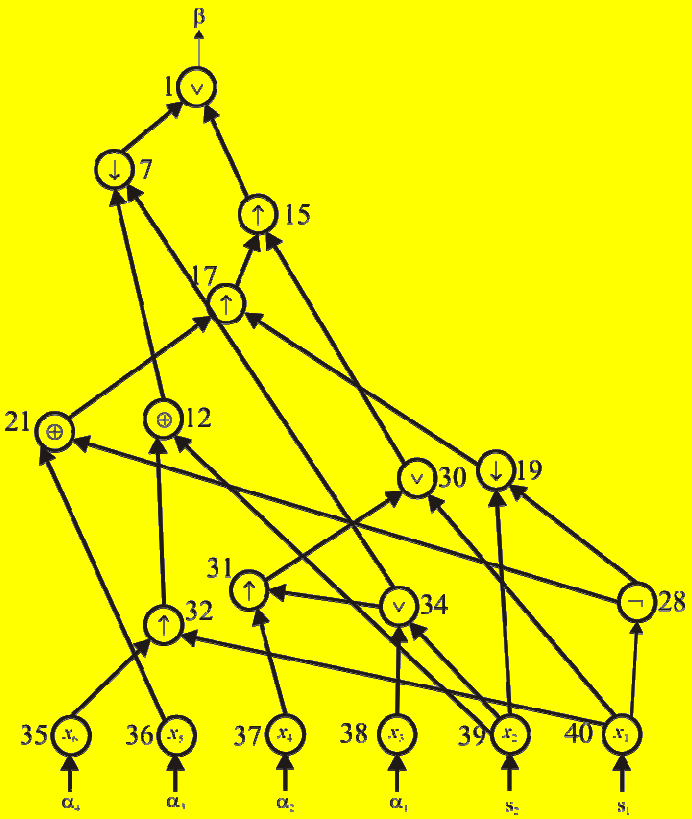
Logický obvod zložený z 11 binárnych brán a zo 4 unárnych brán negácie, ktorý simuluje funkciu selektora

Základné parametre GP pre druhý ilustračný príklad

parameter	*hodnota*
veľkosť populácie P	100
pravdepodobnosť mutácie P_{mut}	0.01
pravdepodobnosť kríženia P_{cross}	0.7
maximálna dĺžka tabuľky p_{max}	40
Penalizačný parameter ω	0.1
maximálny počet epoch	3×10^5

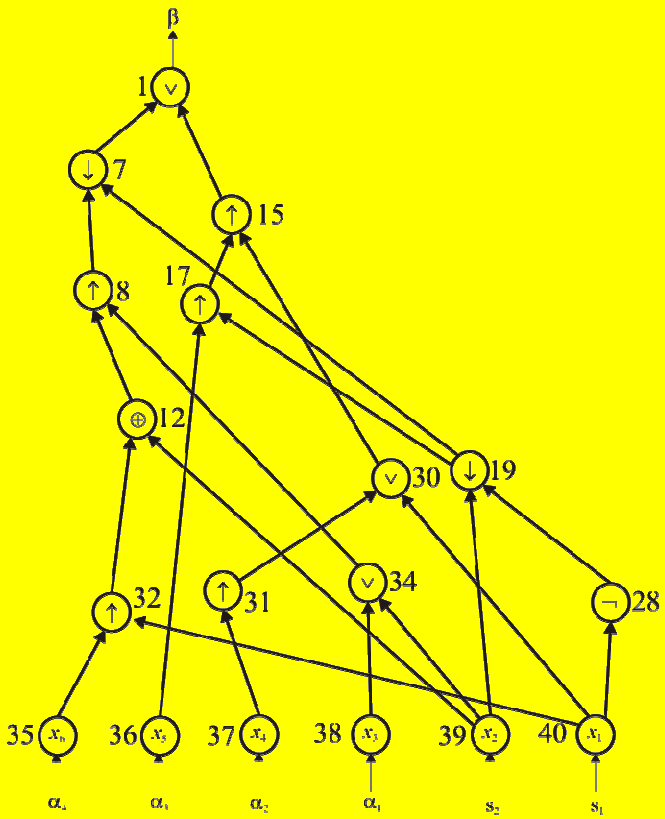


Priebeh funkcie fitnes pri genetickom programovaní pri symbolickej regresii selektora.



A

(*epocha=15000, E=2, fitnes=0.459*)



B

(*epocha=20000, E=0, fitnes=5.556*)

Reprezentanti posledných dvoch etáp genetického programovania. Finálny diagram B obsahuje 11 binárnych brán a 1 unárnu bránu negácie.

Tretí ilustračný príklad – dekodér

Budeme študovať zariadenie nazývane *dekodér*, ktoré dekóduje binárny vstup $\alpha_1, \alpha_2, \alpha_3$ na celé nezáporné číslo z intervalu $\langle 0,7 \rangle$ podľa predpisu

$$f(\alpha_1, \alpha_2, \alpha_3) = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7)$$

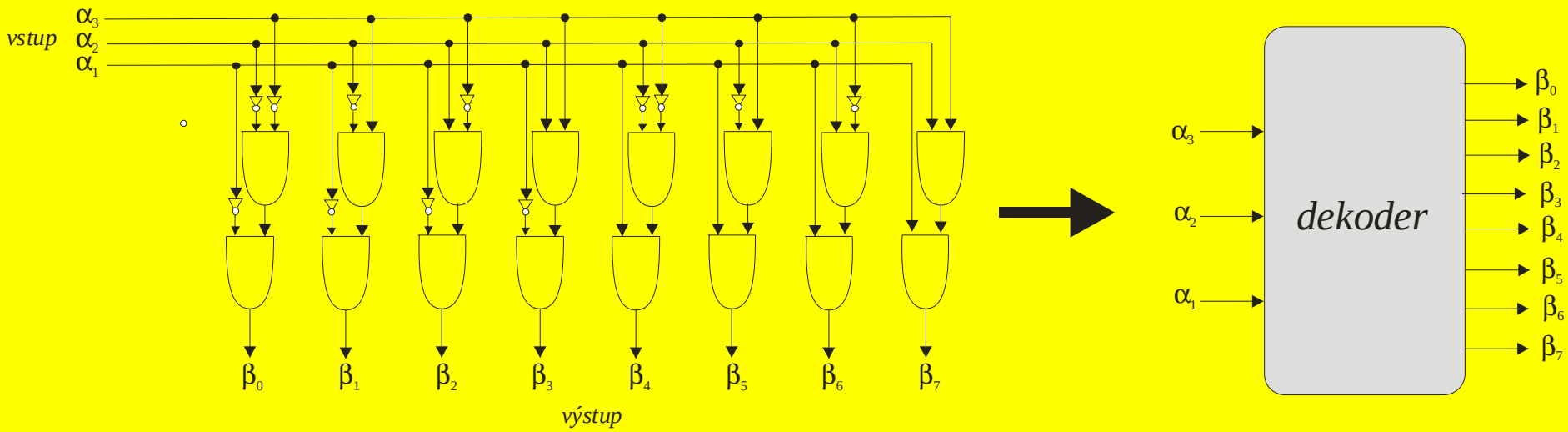
kde pre daný vstup $\alpha_1, \alpha_2, \alpha_3$ elementy výstupu sú určené takto:

$$\beta_i = \begin{cases} 1 & (pre\ i = 4\alpha_1 + 2\alpha_2 + \alpha_3) \\ 0 & (pre\ i \neq 4\alpha_1 + 2\alpha_2 + \alpha_3) \end{cases}$$

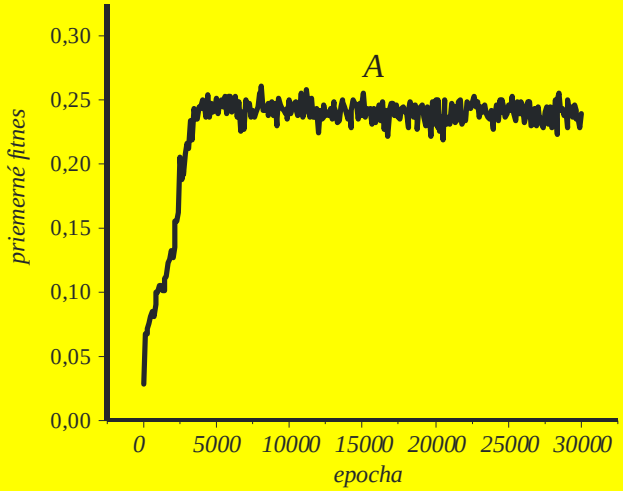
Špecifikácia vstupu a výstupu dekodéra

#	α_1	α_2	α_3	β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7
1	0	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	1	0	0	0	0	0	0
3	0	1	0	0	0	1	0	0	0	0	0
4	0	1	1	0	0	0	1	0	0	0	0
5	1	0	0	0	0	0	0	1	0	0	0
6	1	0	1	0	0	0	0	0	1	0	0
7	1	1	0	0	0	0	0	0	0	1	0
8	1	1	1	0	0	0	0	0	0	0	1

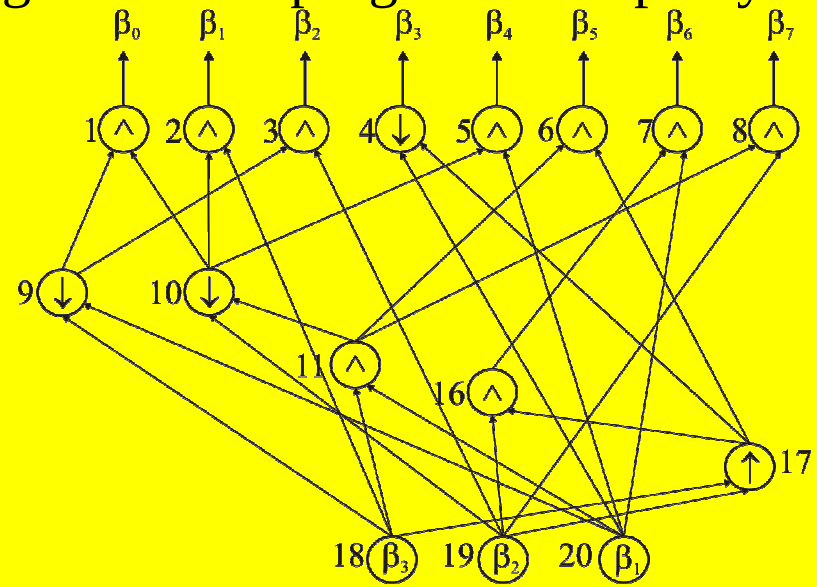
Binárny paralelný dekodér, ktorý dekóduje binárny vstup $\alpha_1, \alpha_2, \alpha_3$ do tvaru nezáporného celého čísla 0 – 7. Tak napríklad, vstupný signál od kombinácie 101 je dekódovaný ako číslica 5 (t. j. všetky výstupy β_i sú nulové okrem výstupu β_5, ktorý je jednotkový).



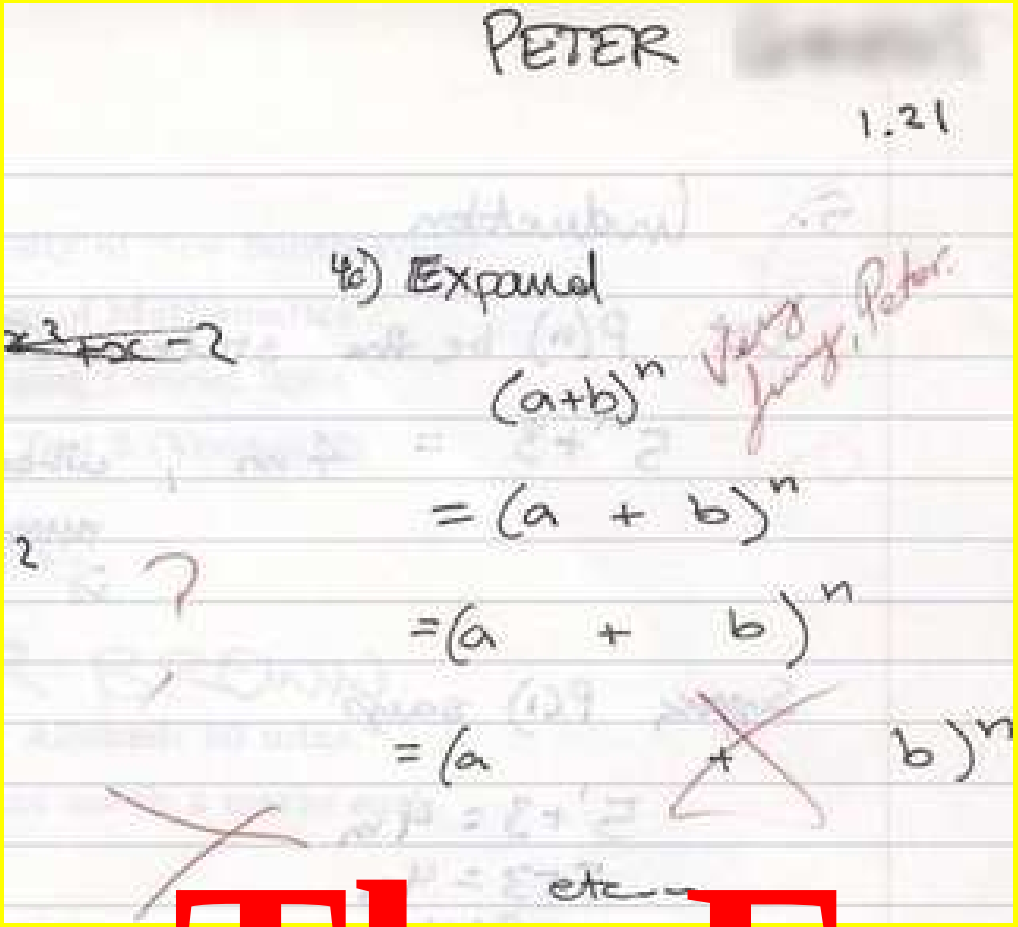
Genetické programovanie bolo uskutočnené pre parametre špecifikované v vprvom ilustračnom príklade.



Priebeh funkcie fitnes pri genetickom programovaní pri symbolickej regresii dekodéra.



Other examples of cognitive jokes



"More decisive? How can I be more decisive?
- I live by the uncertainty principle!"

The End