Iskanje procesne platforme prihodnosti

Miha Mraz

Jesen 2015

Kazalo

Predgovor

Življenje v prihodnosti bo nedvomno pogojeno z našo obkroženostjo z avtomatiziranimi sistemi, ki bodo zadovoljevali naše potrebe. Cena, dimenzija in hitrost teh sistemov bo neposredno pogojena s platformo na kateri bo procesiranje teklo v službi uporabnika. V pričujočem delu skušamo nakazati revolucionarne iztočnice prihodnosti, ki jim bomo na nivoju procesiranja priča ne samo kot uporabniki, temveč tudi kot snovalci sistemov.

prof.dr.Miha Mraz, Ljubljana, v septembru 2015

Poglavje 1

Trovrednostna logika

1.1 Uvod

Klasična *dvovrednostna* ali *dvostanjska logika* temelji na dveh različnih možnih vrednostih *logične spremenljivke* ali dveh različnih možnih *stanjih nosilca* (npr. električnega napetostnega nivoja) logične spremenljivke. Zalogo vrednosti logične spremenljivke lahko simbolično zapišemo z množico $A = \{0, 1\}$. V domeno dvovrednostne logike sodijo tudi *dvovrednostne logične funkcije* ali *dvovrednostne logične operacije*. Dobro so nam poznane enostavne dvovrednostne logične funkcije kot so AND, OR, NEG, NOR, NAND, implikacija, ekvivalenca itd. V domeni dvovhodne dvovrednostne logične funkcije, v katero vstopata logični spremenljivki x_1 in x_2 za katerima se skrivata možni vstopajoči logični vrednosti 0 ali 1, pridemo do pravilnostne tabele štirih različnih vhodnih vektorjev (glej tabelo 1.1). Pri tem nam različne možne vrednosti izhodov (y_1, ... $y_4 \in \{0, 1\}$) omogočajo postavitev šestnajstih različnih dvovrednostnih dvovhodnih logičnih funkcij ($2^4 = 16$).

V domeni dvovrednostne logike nas zanimajo minimalne množice enostavnih logičnih funkcij, s katerimi lahko realiziramo poljubno kompleksnejšo večvhodno logično funkcijo. Minimalno število gradnikov je pomembno z vidika realizacije, saj imamo v tem primeru opravka z manjšim številom različnih logičnih vrat, ki realizirajo opazovano kompleksno logično funkcijo. Posamezen nabor logičnih

x_1	x_2	y
0	0	y_1
0	1	y_2
1	0	y_3
1	1	y_4

Tabela 1.1: Pravilnostna tabela poljubne dvovhodne dvovrednostne logične funkcije.

x_1	x_2	y
0	0	y_1
0	$\frac{1}{2}$	y_2
0	1	y_3
$\frac{1}{2}$	0	y_4
$\frac{1}{2}$	$\frac{1}{2}$	y_5
$\frac{1}{2}$	1	y_6
1	0	y_7
1	$\frac{1}{2}$	y_8
1	1	y_9

Tabela 1.2: Pravilnostna tabela poljubne dvovhodne trovrednostne logične funkcije.

funkcij iz predhodno definirane minimalne množice imenujemo za *poln funkcijski sistem*. Tipični tovrstni nabori v dvovrednostni logiki so $\{NAND\}$, $\{NOR\}$, $\{AND, OR, NEG\}$ itd.

Poleg klasične dvovrednostne logike, na kateri danes temeljijo *računalniško pomnjenje*, *računalniško procesiranje* in *računalniški prenos podatkov*, obstaja tudi množica drugačnih logik, ki so se začele razvijati že v začetku prejšnjega stoletja. Tako poznamo večje število različnih *trovrednostnih* ali *trostanjskih logik* (angl. *ternary logic*) in *večvrednostnih logik* (angl. *multiple valued logic*). V pričujočem poglavju si bomo na kratko ogledali nekaj od naštetih logik in skušali razjasniti smisel njihovega obstoja.

1.2 Trovrednostna logika

Z razliko od dvovrednostne logike pri *trovrednostni* (angl. *ternary logic*) zalogo vrednosti logične spremenljivke ali nosilca stanj logične spremenljivke razširimo na tri elemente. Predpostavimo, da je zaloga teh vrednosti podana z množico $A = \left\{0, \frac{1}{2}, 1\right\}$. Po zgledu tabele 1.1 za dvovhodno dvovrednostno logično funkcijo, lahko napravimo tabelo tudi za dvovhodno trovrednostno funkcijo (glej tabelo 1.2). Pri tem nam različne možne vrednosti izhodov ($y_1, \ldots y_9 \in \left\{0, \frac{1}{2}, 1\right\}$) omogočajo postavitev 19683 različnih trovrednostnih dvovhodnih logičnih funkcij ($3^9 = 19683$).

1.2.1 Značilnosti trovrednostnega zapisa podatkov

Trovrednostni zapis podatkov ima z vidika pomnjenja podatkov zanimive značilnosti, ki so opisane v delu [1]. Predpostavimo, da smo soočeni s problemom pomnjenja naravnih števil. Zapis poljubnega naravnega števila m ($m \in \mathbb{N}$)

lahko formalno zapišemo s polinomskim izrazom

$$m = \sum_{i=0}^{\infty} d_i r^i, \tag{1.1}$$

pri čemer r predstavlja *bazo zapisa* ($r \in \mathbb{N}$), d_i pa posamezen *znak* ali *digit* zapisa ($\forall i : d_i \in \{0, ..., r-1\}$). S spremembo spodnje meje vsote iz 0 na $-\infty$ v izrazu (1.1), bi lahko problem zapisa naravnega števila m razširili na problem zapisa poljubnega realnega števila.

Najpogostejši zapis naravnih števil v realnem svetu je desetiški ali decimalni [2], pri čemer znaki ali digiti prihajajo iz množice $D_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Slednje po [1] izhaja iz načina štetja, ki pri človeku temelji na številu prstov na obeh rokah. V računalništvu je v zadnje pol stoletja najpogostejši dvojiški ali dvovrednostni zapis, pri čemer znaki imenovani *biti* (angl. *binary digit*) prihajajo iz množice $D_2 = \{0, 1\}$. Število 19 tako v desetiškem sistemu ($r = 10$) zapišemo z izrazom

$$19_{10} = 1 * 10^1 + 9 * 10^0, \tag{1.2}$$

v dvojiškem sistemu ($r = 2$) pa z izrazom

$$19_{10} = 10011_2 = 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0. \tag{1.3}$$

Zapis števila 19 v trovrednostnem - trojiškem sistemu ($r = 3$) se ob upoštevanju množice znakov

$$D_3 = \{0, 1, 2\}, \tag{1.4}$$

ki jih imenujemo za *trite*, zapiše z izrazom

$$19_{10} = 201_3 = 2 * 3^2 + 0 * 3^1 + 1 * 3^0. \tag{1.5}$$

Zapis poljubnega naravnega števila na osnovi digitov lahko ovrednotimo na osnovi različnih kriterijev. Dve najpogostejši cenilki sta

- *število uporabljenih digitov*, kar sovpada z velikostjo baze r in
- *dolžina zapisa* w opazovanega števila m.

Glede na izbrano cenilko se nam ponujajo različne rešitve načina zapisa. Po kriteriju iz prve alineje so najugodnejši zapisi z minimalno možno bazo digitov ($r = 1$). V tem primeru imamo v bazi samo en znak in dolžina zapisa v digitih sovpada z velikostjo zapisanega števila. Omenjeni pristop v praksi lahko rezultira v zelo dolge zapise. Omenimo, da opisani *unarni* zapis ne sovpada z formalno definicijo zapisa števila iz izraza (1.1).

Po kriteriju iz druge alineje so najugodnejši zapisi z zelo velikimi bazami. Tako bi lahko pri bazi $r = 1.000.000$ poljubno število iz intervala od 0 do 999.999 zapisali z enim samim digitom, pri čemer smo v tem primeru soočeni z eksplozijo števila digitov in vprašljivo je, kako izredno veliko število različnih hipotetičnih stanj (digitov) prenesti na nosilni signal prenosa, procesiranja ali medij pomnjenja logičnih spremenljivk.

Osnovna hipoteza avtorja članka [1] temelji na predpostavki, da je za zapis naravnih števil baza $r = 2$ premajhna, baza $r = 10$ pa prevelika. Avtor za idealen zapis smatra tisti zapis, kjer je razmerje med dolžino zapisa (w) in velikostjo potencialne zaloge digitov (r) *ustrezno*. Slednje po avtorju dosežemo, ko je dosežen minimum funkcije iz izraza

$$r * w, \tag{1.6}$$

pri zadrževanju konstantne vrednosti funkcije iz izraza

$$r^w. \tag{1.7}$$

Problem je analitično najlažje rešljiv, če sta števili r in w realni ($r, w \in \mathbb{R}$), kar nam za rezultat ponudi doseganje minimuma pri pogoju, da je r enak Eulerjevevi konstanti ($r = e = 2,718281...$). Samo matematično izpeljavo si bralec lahko ogleda v delu [1]. Glede na to, da je najbližje celo število 3, predpostavimo, da baza $r = 3$ predstavlja najučinkovitejši zapis podatkov. Avtor na matematičen način potrdi hipotezo, da je trojiški zapis tisti, ki je za zapisovanje numeričnih podatkov najbolj ekonomičen.

1.2.2 Vrste trovrednostnih logik

V preteklosti je prišlo do porajanja več različnih ternarnih ali trovrednostnih logičnih sistemov. V nadaljevanju si bomo ogledali *trovrednostno logiko po J. Łukasiewiczu* in *uravnoteženi trovrednostni logični sistem.*

Trovrednostna logika po J. Łukasiewiczu

Trovrednostna logika se razvije po zaslugi Poljaka J. Łukasiewicza v dvajsetih letih prejšnjega stoletja. Osnovna ideja, ki jo avtor uvede v svojo logiko, je uvedba tretje logične vrednosti, ki odraža stanje *nedoločenosti* vrednosti logične spremenljivke (angl. *indeterminate*), poleg nam že znanih logičnih vrednosti *true* in *false* [3]. Łukasiewicz poveže logične vrednosti z kodnim zapisom $D_3 = \{0, 1, 2\}$ na osnovi izraza

$$false \equiv 0, indeterminate \equiv 1, true \equiv 2. \tag{1.8}$$

Na osnovi interpretacije logičnih vrednosti s kodnim zapisom vzpostavi pravilnostne tabele za osnovne logične operatorje, kot so NEG, AND, OR in implikacija. Najdemo jih v tabelah v nadaljevanju 1.3, 1.4, 1.5 in 1.6.

Izkaže se, da definirane trovrednostne funkcije v tabelah 1.3, 1.4, 1.5 in 1.6 ne definirajo polnega funkcijskega nabora [3]. Łukasiewicz v ta namen uvede še funkcijo T, s čimer dobimo funkcionalno polno algebro, definirano s šestorčkom

$$\langle E = \{0, 1, 2\}, \rightarrow, NEG, T(), 0, 1\rangle, \tag{1.9}$$

pri čemer je funkcija T definirana s pravilnostno tabelo 1.7.

x	$\overline{x}$
0	2
1	1
2	0

Tabela 1.3: Pravilnostna tabela Łukasiewiczeve trovrednostne negacije.

x_1	x_2	y
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	1
2	0	0
2	1	1
2	2	2

Tabela 1.4: Pravilnostna tabela Łukasiewiczeve trovrednostne konjunkcije.

x_1	x_2	y
0	0	0
0	1	1
0	2	2
1	0	1
1	1	1
1	2	2
2	0	2
2	1	2
2	2	2

Tabela 1.5: Pravilnostna tabela Łukasiewiczeve trovrednostne disjunkcije.

x_1	x_2	y
0	0	2
0	1	2
0	2	2
1	0	1
1	1	2
1	2	2
2	0	0
2	1	1
2	2	2

Tabela 1.6: Pravilnostna tabela Łukasiewiczeve trovrednostne implikacije.

x	$T(x)$
0	1
1	1
2	1

Tabela 1.7: Pravilnostna tabela Łukasiewiczeve trovrednostne T funkcije.

Uravnoteženi trovrednostni logični sistem

Uravnoteženi trovrednostni logični sistem (angl. *balanced ternary sistem*) temelji na izboru kodne množice treh znakov, ki omogoča ekonomičnejši zapis podatkov. Primer takšnega kodnega nabora je množica $A = \{-1, 0, 1\}$. Z vpeljavo negativno predznačenega digita pride do poenostavitve zapisa predznaka zapisanega števila in do posploštive nekaterih računskih operacij (npr. odštevanje se realizira kot seštevanje, pri čemer se le zamenjajo predznačeni biti). Posebnega znaka za predznačitev podatka tako ne potrebujemo več.

Preostali trovrednostni logični sistemi

V strokovni literaturi najdemo množico različnih trovrednostnih logik. V delu [3] so tako opisane Postova logika, Bochvarjevea logika, Kleenova logika, Allenova in Givoneova logika, Pradhanova logika itd.

1.2.3 Trovrednostno procesiranje

Trovrednostno procesiranje (angl. *ternary computing*) se je v preteklosti uporabljalo tudi v realnih računalniških sistemih. V prejšnjem stoletju je bilo v petdesetih letih v nekdanji Sovjetski zvezi zgrajenih kar nekaj računalnikov, ki so izvajali operacije v trovrednostnem zapisu in na osnovi trojiške logike. Najbolja znana družina tovrstnih sistemov je bila serija računalnikov Setun (1.generacija l.1958, 2.generacija l.1970), razvitih na Moskovski državni univerzi. Temeljili

so na uravnoteženi množici digitov zapisovanja $\{-1, 0, 1\}$. Osnovna enota zapisa sorodna današnjemu bitu je bil *trit*, šest tritov pa je predstavljajo *tryte*, ki je soroden današnjemu byte-u. Obstajajo splošne ocene, da je zapis v tryte-u ekvivalenten podatkovni vsebini zapisani v 9,5 bitih. Prvi emulator trovrednostnega računalnika v ZDA je bil implementiranem l.1973 na Borroughs B1700 platformi.

Področje trojiškega zapisa in ustrezne logike še danes buri duhove znastvene srenje s področja računalništva. Eden od pionirjev računalništva Donald Knuth trdi, da se bo ternarno procesiranje zaradi svojih dobrih plati nedvomno vrnilo v računalniške strukture. Mnogo zanimivih informacij z opisanega področja najdemo na spletni strani www.trinary.cc. Ob evidentnem skrajšanju zapisa podatkov, dobimo krajše enote obdelave, kar omogoča hitrejšo obdelavo podatkov, manj dostopov do pomnilnika, manjšo kompleksnost priključkov, itd.

Literatura

[1] B. Hayes, "Third base," *American Scientist*, vol. 89, no. 6, 2001.

[2] H. Stocker, *Matematični priročnik z osnovami računalništva.* Tehniška založba, Slovenija, 2006.

[3] D. Miller and M. Thornton, *Multiple Valued Logic, Concepts and Representations.* Morgan and Claypool Publishers, USA, 2008.