TECHNICKÁ UNIVERZITA V KOŠICIACH
Fakulta elektrotechniky a informatiky

Prof.Ing.Štefan HUDÁK, DrSc.

TEORETICKÁ INFORMATIKA:

Katedra počítačov a informatiky FEI TU

November 2002

Obsah

Kapitola 1

Algebra algoritmov

V tejto kapitole sa budeme venovať novému prístupu k syntéze (programových) systémov, ktorý je založený na netradičnom pohľade na algoritmus, ako na objekt, pre ktorý je možno nájsť algebru, čo umožňuje manipulovať s algoritmami, ako s algebraickými objektami. Inými slovami, uvidíme, ako sa definujú operácie (transformácie) nad algoritmami, ktoré sú zvolené tak, aby sa zachovala funkcia algoritmu a menila sa iba jeho forma. Naznačený prístup je založený na pionierskych prácach kyjevskej algebraicko-logickej školy kybernetiky založenej akademikom V.M.Gluškovom a jeho spolupracovníkmi, medzi ktorými zaujíma významné miesto Prof. G.E. Cejtlin [14].

V tejto kapitole sa zaoberáme základmi teórie algoritmiky (iné pomenovanie pre informatiku, ktoré zvýrazňuje miesto algoritmu, ako ústredného pojmu vedy o počítačoch a počítaní), v duchu prác G.E. Cejtlina. Po stručnom uvedení poznatkov o algebraických systémoch v časti 1.1, sa oboznámime s teóriou tzv. graf-schém (časť **??**)a možnosťami ich reprezentácie pomocou špeciálnych algebraických termov. Príkladom algoritmických algebier je venovaná časť **??**. Teória algebry algoritmov je zovšeobecnením teórie E. Posta o funkcionálnej úplnosti, ktorá je ilustrovaná na boolovských funkciach v časti **??**. Príklady aplikácií algebry algoritmov sú predmetom časti **??**.

1.1 Modely. Algebry. Algebraické systémy

Modely a algebry patria medzi základné pojmy matematiky, ktoré na druhej strane sú používané v algoritmike pri konštrukcii tzv. *logicko-funkcionálnych modelov*, ktoré sa využívajú pri *formalizácii* zadania úlohy, ktorej riešenie je predmetom návrhu príslušného programu, alebo systému. Pomocou takých, alebo podobných modelov sa realizuje upresnenie zadania úlohy, ktoré na začiatku môže byť predstavené neformálne, alebo v pojmoch relácií. Ako ilustráciu uvedieme nasledujúce príklady.

Pri formulácii predpokladáme, že čitateľ je oboznámený s takými pojmami ako je relácia, funkcionálna relácia, či funkcia, alebo predikát (ako špeciálny prípad funkcie). Ďalej tiež predpokladáme, že je nám jasný význam pojmu *usporiadanie*, definovaného ako *binárna relácia* na niektorej univerzálnej množine prvkov. Rozlišujeme tzv. *čiastočné usporiadanie* a *úplné (lineárne) usporiadanie*.
Majme teraz univerzálnu množinu prvkov **U**, na ktorej je definovaná relácia lineárneho usporiadania $\prec$ (čítaj menší). Označíme si množinu všetkých (konečných) postupností na **U** ako *SEQ*. Majme teraz postupnosť $M \in SEQ$ a nech $M = (a_1, a_2, ...a_n)$. Budeme M volať *usporiadanou*, ak platí $a_i \prec a_j$, pre všetky i a j také, že $i \prec j, 1 \geq i, j \leq n$. Množinu všetkých usporiadaných postupností na **U** označíme ako *USQ*. Je jasné, že $USQ \subset SEQ$.

Príklad 1.1 *Definujeme si binárnu reláciu $SORT \subset SEQ \times USQ$, ktorá definuje priradenie medzi (vo všeobecnosti neusporiadanými) postupnosťami zo SEQ a usporiadanými postupnosťami z USQ. Je*

jasné, že jedna usporiadaná postupnosť z USQ môže zodpovedať viacerym postupnostiam zo SEQ. Relácia SORT priraďuje každej postupnosti $M \in SEQ$ (vzor)jedinečnú postupnosť $M' \in USQ$ (obraz), ktorá vznikne z M permutáciou prvkov tak, aby tieto boli lineárne usporiadané. Tak napríklad vzorom $M_1 = (5,2,1,6,5)$ a $M_2 = (1,2,5,6,5)$ zodpovedá jediný obraz-postupnosť $M_3 = (1,2,5,5.6)$; tu $M_1, M_2 \in SEQ$ a $M_3 \in USQ$. S binárnou reláciou SORT je asociovaná unárna (s jedným argumentom) funkcia sort taká, že $sort(M) = M'$ práve vtedy, keď postupnosti M a M' sú v relácii SORT, t.j. ak $M, M' \in SORT$.

□
(Koniec príkladu)

Všeobecne, priradenie f medzi prvkami množín A a B , pri ktorom pre každý prvok $a \in A$ existuje nie viac ako jeden prvok $b \in B$ a taký, že $f(a)=b$ sa volá unárnou (1-miesnou, 1-argumentovou) funkciou z A do B,$f : A \to B$. S každou funkciou z A do B je spriahnutá (asociovaná) binárna relácia $F \subseteq A \times B$ s vlastnosťou, že $(a,b) \in F$ práve vtedy keď $f(a)=b$. Taká binárna relácia sa volá *funkcionálnou*binárnou reláciou.
Nasleduje ďalší príklad.

Príklad 1.2 *Podľa binárnej relácie SORT možno definovať predikát S-O-R-T(x,y), kde x a y sú argumenty zodpovedajúco na množinách SEQ a USQ. Predikát S-O-R-T je definovaný takto:*

$$S-O-R-T(M,M') = \begin{cases} 1, ak\ sort(M)=M'; \\ 0\, vopanomprpade \end{cases}$$

Logická funkcia S-O-R-T je príklad 2-miestneho (2-argumentového) predikátu, ktorý jekj definovaný na karteziánskom súčine $SEQ \times USQ$ s hodnotami v obore $E_2 = \{0,1\}$

□
(Koniec príkladu)

Všeobecne platí, že s ľubovoľnou n-árnou reláciou $R^{(n)} \subset A_1 \times A_2 \times ... \times A_n$ je spriahnutý n-miestny predikát $J(x_1, x_2, ..., x_n)$ definovaný takto:

$$J(a_1, a_2, ..., a_n) = \begin{cases} 1, ak\ (a_1,a_2,...,a_n) \in R^{(n)}; \\ 0\, vopanomprpade \end{cases}$$

Predikát $J(x_1, x_2, ..., x_n)$ sa volá *charakteristickým predikátom* relácie $R^{(n)}$. V prípade *n=1* s unárnou reláciou $R^{(1)}$ je asociovaná množina A , $R^{(1)} \subseteq A$ a charakteristický predikát $J(x)$ v danom prípade má tú vlastnosť, že nadobúda hodnotu 1 na prvkoch množiny $R^{(1)}$ a hodnotu 0 na ostatných prvkoch množiny A.
Predikáty a funkcie sa využívajú tak pri formulácii úloh, ako aj pri návrhu akgoritmov ich riešenia.
V texte budeme navrhované koncepcie ilustrovať na príklade riešenia problému triedenia, ktorý je veľmi dobre známy a dôležitý problém z praktického hľadiska. V riešení tohoto problému binárna relácia *SORT*, unárna funkcia *sort* a charakteristický predikát *S-O-R-T* ustanovujú vzájomný vzťah medzi vstupnými (z hľadiska problému triedenia) postupnosťami zo *SEQ* a usporiadanými postupnosťami z *USQ*. Formulácia úlohy triedenia v uvedených pojmoch dáva prvotnú predstavu o probléme triedenia postupností (na množine **U**).
V ďalšom bude uvedená metóda a možnosti upresnenia a prehlbenia takého zadania úlohy cestou tvorby formálneho aparátu.
Teraz uvedieme definíciu konkrétnych predikátov a funkcií (operátorov), ktoré sa budú využívať pri upresňovaní zadania úlohy triedenia v ďalšom.
Pre potreby riešenia problému triedenia začneme s intuitívne dobre pochopiteľným algoritmom tzv. bublinkového triedenia, pre ktorý budeme používať akronym *BUBBLE* (z anglického originálu názvu tohoto algoritmu *Bubble Sort*). Budeme pracovať so špeciálne upravenými postupnosťami zo *SEQ*, ktoré budú

doplnenné niekoľkými špeciálnymi symbolmi:**H**(na označenie začiatku postupnosti), $\mathbf{Y}_1$(na označenie aktuálnej pozície v postupnosti), **K**(na označenie konca postupnosti). Určenie symbolov ilustruje upravená postupnosť $M = (a_1, a_2, ..., a_n)$, ktorá doplnená o uvedené symboly bude vyzerať takto:

$$\mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}_1, a_{i+1}, ..., a_n\mathbf{K} \tag{1.1}$$

Elementárne predikáty

Všetky nižšie uvádzané predikáty sú definované na označenej postupnosti **M** typu (1.1).

$$d(Y_1, K) = 1 \quad \Leftrightarrow_{df} \quad \mathbf{M}{=}\mathbf{M}^{'}{:}\mathbf{H}a_1, a_2, ..., a_n, \mathbf{Y}_1\mathbf{K} \tag{1.2}$$

$$\ell > r|Y_1 = 1 \quad \Leftrightarrow_{df} \quad \mathbf{M} = \mathbf{M}^{'}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}_1, a_{i+1}, ..., a_n\mathbf{K} \wedge a_i > a_{i+1} \tag{1.3}$$

$$UM = 1 \quad \Leftrightarrow_{df} \quad \mathbf{M} = \mathbf{M}^{'}{:}\mathbf{H}\mathbf{Y}_1, a_1, a_2, ..., a_n\mathbf{K} \wedge a_i < a_{i+1}, i = 1, 2, ..., n-1 \tag{1.4}$$

Definícia elementárnych predikátov nemá vo svojej definícii javne zahrnutý argument **M**. Dôvod je snaha po jednoduchosti zápisu a ďalší dôvod bude jasný pozdejšie. Význam predikátov je jasný z definície a $d(Y_1, K) = 1$ znamená, že ukazovateľ (indikátor)aktuálnej pozície (postupnosti **M**) dosiahol koncový marker **K**. Ak si predstavíme, že **M** je sekvenčný súbor, potom predikát $d(Y_1, K)$ je identický s predikátom *eof(M)*. Prípad $\ell > r|Y_1 = 1$ vyjadruje fakt, že prvok stojaci naľavo od indikátora $\mathbf{Y}_1$ v $\mathbf{M}(a_i)$ je väčší od prvku stojaceho bezprostredne napravo od $\mathbf{Y}_1$ (a_{i+1}). Napokon pravdivosť predikátu *UM=1* vyjadruje situáciu, že postupnosť **M** je usporiadaná, t.j. utriedená v zmysle relácie $\prec$.

Elementárne operátory

Podobne ako v predošlom prípade definujeme si elementárne operátory, ktoré budeme potrebovať v ďalšom. Definícia elementárnych operátorov taktiež nemá vo svojej definícii javne zahrnutý argument **M**. Dôvody sú rovnaké ako vyššie.

$$E \quad \Leftrightarrow_{df} \quad E(\mathbf{M}) = \mathbf{M} \tag{1.5}$$

$$\tag{1.6}$$

$$P(Y_1) \quad \Leftrightarrow_{df} \quad \mathbf{M}_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}_1, a_{i+1}, ..., a_n\mathbf{K} \wedge$$
$$\mathbf{M}_2 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, a_{i+1}, \mathbf{Y}_1, a_{i+2}, ..., a_n\mathbf{K} \wedge$$
$$P(Y_1)(M_1) = M_2 \tag{1.7}$$

$$\tag{1.8}$$

$$TRANSP(\ell, r) \quad \Leftrightarrow_{df} \quad \mathbf{M}_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}_1, a_{i+1}, ..., a_n\mathbf{K} \wedge$$
$$\mathbf{M}_2 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_{i+1}, \mathbf{Y}_1, a_i, ..., a_n\mathbf{K} \wedge$$
$$TRANSP(\ell, r)(M_1) = M_2 \tag{1.9}$$

$$\tag{1.10}$$

$$UST(Y_1, H) \quad \Leftrightarrow_{df} \quad \mathbf{M}_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}_1, a_{i+1}, ..., a_n\mathbf{K} \wedge$$
$$\mathbf{M}_2 = \mathbf{M}{:}\mathbf{H}\mathbf{Y}_1 a_1, a_2, ..., a_i, a_{i+1}, ..., a_n\mathbf{K} \wedge$$
$$UST(Y_1, H)(M_1) = M_2 \tag{1.11}$$

Operátor E definuje identickú funkciu, resp. ide o ´nič nerobiaci´ operátor. Operátor $P(Y_1)$ definuje funkciu, ktorá pre ľubovoľnú označenú postupnosť $\mathbf{M}_1$ vracia označenú postupnosť $\mathbf{M}_2$, ktorá sa od $\mathbf{M}_1$ líši aktuálnou pozíciou, ktorá je o jeden symbol napravo od aktuálnej pozície v $\mathbf{M}_1$. Inými slovami indikátor aktuálnej pozície $\mathbf{Y}_1$ v $\mathbf{M}_2$ je posunutý o 1 symbol doprava oproti $\mathbf{M}_1$. TRANSP(ℓ,r) definuje funkciu, ktorá pre ľubovoľnú označenú postupnosť $\mathbf{M}_1$ vracia označenú postupnosť $\mathbf{M}_2$, v ktorej sú navzájom vymenené ľavý (a_i) a pravý (a_{i+1}) prvok vzhľadom na aktuálnu pozíciu ukazovateľa $\mathbf{Y}_1$ v $\mathbf{M}_1$. Aktuálna pozícia ukazovateľa $\mathbf{Y}_1$ v $\mathbf{M}_2$ sa od $\mathbf{M}_1$ nelíši. Operátor **UST** definuje funkciu, ktorá pre ľubovoľnú

označenú postupnosť $\mathbf{M}_1$ vracia označenú postupnosť $\mathbf{M}_2$, s nezmeneným obsahom a usporiadaním s výnimkou pozície ukazovateľa $\mathbf{Y}_1$. Posledný sa v $\mathbf{M}_2$ nachádza pred 1. symbolom.

1.2 Logicko-funkcionálne modely a formulácia zadania úlohy.

V tejto časti uvedieme základné pojmy a výsledky týkajúce sa logicko-funkcionálnych modelov (LFM), ktoré slúžia k formalizovanému vyjadreniu špecifikácie úlohy, resp. zadania pre návrh (programového) systému . V týchto termínoch sa obyčajne realizuje formalizovaná definícia úloh, ktorá vlastne dáva odpoveď na otázku- ´Čo je treba urobiť?´.
Model $\widetilde{M}$ sa definuje ako systém $\widetilde{M} = (A; SIGN_\pi)$, kde A je báza (množina dát) a $SIGN_\pi = \{\pi_i | \in \mathbf{I}\}$, sa volá *signatúra* pozostávajúca z predikátov π na množine A a $\mathbf{I}$ je množina indexov. Všeobecná schéma použitia modelu $\widetilde{M}$ pri špecifikovaní problému (zadania, úlohy atp.) sa dá uzrieť, vychádzajúc z nasledujúcej úvahy: majme 2 množiny $V, R \subseteq A$, kde V predstavuje množinu vstupných a R množinu výstupných dát. Formulovanú úlohu je možno predstaviť ako niektoré zobrazenie (funkciu)

$$f^{(n)} : V^n \to R$$

Funkcia $f^{(n)}$ je n-árna funkcia, ktorá n-ici dát z V priraďuje údaj z R. Ako bolo uvedené v úvode paragrafu 1.1 s funkciou $f^{(n)}$ je asociovaná n+1-árna relácia $F^{(n+1)} \subseteq V^n \times R$ s vlastnosťou , že

$$(d_1, d_2, ..., d_n, r) \in F^{(n+1)} \Leftrightarrow f^{(n)}(d_1, d_2, ..., d_n = r$$

Zostáva si uvedomiť, že podľa $F^{(n+1)}$ je možno zostrojiť charakteristický predikát $J^{(n+1)}$ pre množinu $V^n \times R$ s vlastnosťou, že

$$J^{(n+1)}(d_1, d_2, ..., d_n, r) = 1 \Leftrightarrow (d_1, d_2, ..., d_n, r) \in F^{(n+1)} \tag{1.12}$$

Pre zadanú úlohu je možno zostrojiť predikát typu (1.12) pre každú zodpovedajúcu povahe úlohy funkciu $f_i^{(n_i}$. To završuje konštrukciu modelu ako formalizácie zadania úlohy.
Teraz uvedieme príklad.

Príklad 1.3

Zostrojíme LFM $\widetilde{M_1} = (SEQ, SIGN_1)$, ktorý bude definovaný pre úlohu triedenia postupností $M \in SEQ$.
Začneme s konštrukciou elementárnych predikátov vytvárajúcich signatúru $SIGN_1$, ktoré poslúžia k vyjadreniu 2-miestneho charakteristického predikátu *S-O-R-T*.

- unárny predikát UM (viď 1.4), pre ktorý platí, že $UM(M) = 1 \Leftrightarrow M \in USQ$
- binárny predikát $D : SEQ^2 \to \{0, 1\}$ a taký , že $D(M, M^{'}) = 1 \Leftrightarrow |M| = |M^{'}|$, kde $|Z|$ je dĺžka ľubovoľnej postupnosti $Z \in SEQ$.
- binárny predikát $R : SEQ^2 \to \{0, 1\}$ a taký , že $R(M, M^{'}) = 1 \Leftrightarrow ||M|| = ||M^{'}||$, kde $||Z||$ je množina prvkov v ľubovoľnej postupnosti $Z \in SEQ$.
- binárny predikát $B : SEQ^2 \to \{0, 1\}$ a taký , že $B(M, M^{'}) = 1 \Leftrightarrow \mathbf{I}(q, M) = \mathbf{I}(q, M^{'})$, pre každý prvok $q \in ||M|| \cap ||M^{'}||$, kde $\mathbf{I}$(q,Z) je počet výskytov prvku q v postupnosti $Z \in SEQ$. Inými slovami, predikát $B(M, M^{'}) = 1$ práve vtedy, ak počty výskytov spoločných prvkov v postupnostiach M a $M^{'}$ sú rovnaké.

Potom

$$SIGN_1 = \{UM, D, R, B\}$$

Predikát S-O-R-T sa dá v LFM $\widetilde{M_1} = (SEQ, SIGN_1)$ vyjadriť nasledujúcou konjunkciou :

$$S - O - R - T(M, M^{'}) = D(M, M^{'}) \wedge R(M, M^{'}) \wedge B(M, M^{'}) \wedge U(M^{'}) \tag{1.13}$$

pričom $M \in SEQ, M^{'} \in USQ$.
Teda môžeme konštatovať, že v LFM $\widetilde{M_1} = (SEQ, SIGN_1)$ je opísaný formulou (1.13) predikát S-O-R-T charakterizujúci funkcionálnu reláciu SORT, ktorá je spriahnutá s úlohou triedenia postupností definovaných na množine **U**.

□
(Koniec príkladu)

1.3 Algebry.

V tejto časti sú stručne uvedené základné pojmy a definície algebry a s ňou spojených pojmov, akými sú :signatúry algebry, systém vytvárajúcich prvkov (generátorov), bázy, superpozície, axiomatický systém a ďalšie.
Majme ľubovoľnú množinu A a funkciu $f(x_1, x_2, ..., x_n)$ definovanú na A s hodnotami taktiež v množine A. Formálne

$$f : A^n \to A$$

Funkciu $f(x_1, x_2, ..., x_n)$ voláme *operáciou* definovanou na množine A. Ako príklad uvedieme množinu prirodzených čísel $N = \{1, 2, ..., n, ...\}$ a funkciu $f(x, y) = x + y$ dvoch argumentov x,y. Funkcia $f(x, y) = x + y$ je definovaná na N a priraďuje každej dvojici prirodzených čísel x,y jednoznačne dané prirodzené číslo x+y. Funkcia x+y predstavuje operáciu sčítania prirodzených čísel.
Majme na množine A definovanú celú množinu operácií $\Omega = \{F_i(x_1, x_2, ..., x_n)|i - 1, 2, ..., k\}$. *Univerzálnou algebrou* nazveme systém $\widetilde{A} = (A; \Omega)$, kde A sa volá *osnova (základňa)* algebry, a Ω je *signatúra* algebry. Ako príklad uvedieme algebru boolovských funkcií $\widetilde{ABF} = (BF(n); \Omega_{ABF})$, kde osnovou je množina všetkých boolovských funkcií (b.f.) n premenných BF(n) a signatúra Ω_{ABF} pozostáva z 3 operácií: $x \wedge y$, $x \vee y$ a negácie $\overline{x}$.
Teraz pristúpime k definícii *systému vytvárajúcich prvkov (generátorov)* a *bázy* algebry.

Generátory algebry a báza algebry.

Vzniká otázka, či pre danú algebru $\widetilde{A} = (A; \Omega)$ sa nájde v A taká podmnožina prvkov, povedzme $S \subseteq A$, že každý prvok základnej množiny A je možno vytvoriť z prvkov množiny S s využitím operátorov zo signatúry Ω. Ak taká S existuje budeme ju volať *množinou vytvárajúcich prvkov*, alebo *množinou generátorov* algebry $\widetilde{A}$.

Príklad 1.4

Majme algebru $\widetilde{A_N} = (\mathbf{N}; \{+\})$, ktorá je daná množinou **N**- prirodzených čísel-osnovou algebry a signatúrou pozostávajúcou z jedinej operácie $x+y$ definovanej na **N**. Nie je ťažké sa presvedčiť, že interval prirodzených čísel $[1, k]$, $k \in$ **N**je systémom generátorov algebry $\widetilde{A_N}$. Skutočne z prvkov intervalu $[1, k]$ je možné vytvoriť ľubovoľné prirodzené číslo s použitím schémy ich generovania (k+1),(k+1)+1,((k+1)+1)+1,... . Týmto spôsobom je možné vytvoriť ľubovoľné prirodzené číslo mimo $[1, k]$. Prirodzené čísla z intervalu $[1, k]$ sú dané samotným intervalom.

□
(Koniec príkladu)

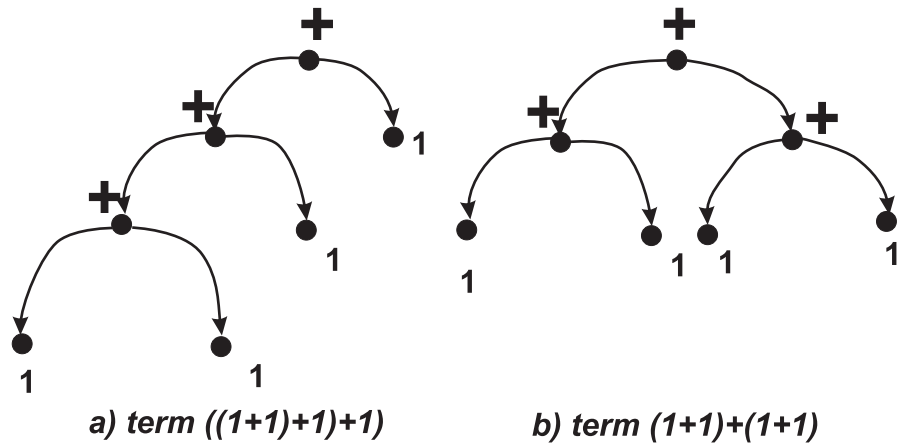
Ukazuje sa , že algebra môže mať viac než jeden systém generátorov. Tak v prípade algebry $\widetilde{A_N}$ sme videli, že interval $[1, k]$ vytvára systém jej generátorov. Ak si vezmeme interval $[1, k - 1]$ môžeme sa presvedčiť, že aj tento interval je systémom generátorov algebry $\widetilde{A_N}$. Analogicky aj každý interval $[1, k - 2]$, $[1, k - 3]$,..., [1] je systémom generátorov algebry $\widetilde{A_N}$. Všimnime si, že posledný systém [1] neumožňuje pokračovať vo vytváraní ďalších systémov generátorov algebry $\widetilde{A_N}$.

Systém S generátorov algebry $\widetilde{A} = (A; \Omega)$ z ktorého nie je možné vylúčiť žiadny prvok $a \in S$ z S, aby sa pritom nenarušila vlastnosť $S - a$ byť systémom generátorov algebry $\widetilde{A}$ sa volá *bázou* algebry $\widetilde{A}$.
To znamená, že 1-prvkový interval [1] je bázou algebry $\widetilde{A_N}$.
Spôsob tvorby nových prvkov algebry, tak ako to vyplýva z definície systému vytvárajúcich prvkov (generátorov) a bolo ilustrované na príklade tvorby prvkov algebry prirodzených čísel $\widetilde{A_N}$, sa deje pomocou aplikácie operácií signatúry algebry na generátory.
Vo všeobecnosti môžeme definovať pojem *termu* algebry $\widetilde{A} = (A; \Omega)$ s množinou generátorov $G \subseteq A$.

- každý prvok $x_i \in G$ je termom algebry $\widetilde{A}$;
- pre každú n-árnu operáciu $f_i^{(n)}$ výraz $t = f_i^{(n)}(x_1, x_2, ..., x_n)$, kde pre každé j $1 \leq j \leq n$ $x_j \in G$, je termom algebry $\widetilde{A}$. Budeme používať označenie $t(x_1, x_2, ..., x_n)$, aby sme zvýraznili , že ide o term závislý od n argumentov (n-árny term)
- pre každú n-árnu operáciu $f_i^{(n)}$ a termy $t_1, t_2, ..., t_n$ je $t = f_i^{(n)}(t_1, t_2, ..., t_n)$ termom algebry $\widetilde{A}$;
- dva termy $t = t(x_1, x_2, ..., x_n)$ a $t' = t'(x_1, x_2, ..., x_n)$ budeme považovať za ekvivalentné, ak platí, že pre každú n-icu argumentov $x_1, x_2, ..., x_n$ $t(x_1, x_2, ..., x_n)$ a $t'(x_1, x_2, ..., x_n)$ budú vytvárať (generovať) rovnaký prvok algebry $\widetilde{A}$. Také vzťahy medzi termami sa volajú *vzťahmi totožnosti*, alebo jednoducho *totožnosti*.

Príklad 1.5

Vrátime sa k algebre $\widetilde{A_N} = (\mathbf{N}; \{+\})$ z príkladu 1.4. Prvok 4 môže byť vytvorený na báze [1] pomocou termu t=((1+1)+1)+1. Taký term, vo všeobecnosti, vznikne viacnásobnou superpozíciou substitúcie jedných operácií zo signatúry Ω namiesto argumentov iných operácií danej algebry. V danom konkrétnom prípade pôjde o substitúciu argumenta x v operácii *x+1* termom (1+1), čo nám dá prvok 3=(1+1)+1. Ak to zopakujeme znovu, že nahradíme v operácii *x+1* x termom ((1+1)+1) dostaneme prvok 4=((1+1)+1)+1.
Na druhej strane prvok 4 je možné vytvoriť aj pomocou termu 4=((1+1)+(1+1)). To znamená, že termy (((1+1)+1)+1) a ((1+1)+(1+1)) sú ekvivalentné, alebo jednoducho hovoríme, že sa rovnajú. Situáciu ilustruje obr. 1.1.



Obrázok 1.1: Grafová reprezentácia termov

□
(Koniec príkladu)

Pomocou vzťahu totožnosti sa dajú charakterizovať vlastnosti operácií algebry $\widetilde{A}$. Základné z týchto vlastností sa formulujú ako *axiómy*, zatiaľ čo ostatné vlastnosti sa dajú odvodiť z axióm. Týmto spôsobom prichádzame ku axiomatickej charakteristike algebry. Ako príklad uvádzame nám dobre známu

axiomatickú charakteristiku boolovej algebry $\widetilde{B} = (B; \{+, \cdot, ^- \})$, operácie ktorej vyhovujú týmto axiómam:

zákon asociatívnosti	$(x \circ y) \circ z = x \circ (y \circ z) = x \circ y \circ z$
	kde symbol $\circ$ je symbolom operácie $+$, alebo $\cdot$
zákon komutatívnosti	$x \circ y = y \circ x$
zákon idempotentnosti	$x \circ x = x$
zákon distributívnosti $\cdot$ nad $+$	$(x + y) \circ z = (x \circ z) + (y \circ z)$
zákon distributívnosti $+$ nad $\cdot$	$(x \circ y) + z = (x + z) \circ (y + z)$
zákon negácie negácie	$\overline{\overline{x}} = x$
zákon pohltenia (absorbcie)	$x + x \cdot y = x$
	$x \cdot (x + y) = x$
pravidlá de Morgana	$\overline{x \cdot y} = \overline{x} + \overline{y}$
zákon vylúčenia tretieho	$x + \overline{\overline{x}} = 1$
zákon protirečenia	$x \cdot \overline{\overline{x}} = 0$
	kde 0 a 1 sú konštanty b.a.
axiómy pre konštanty	$\overline{1} = 0, \overline{0} = 1$
	$1 \cdot x = x, 0 + x = x$
	$1 + x = 1, 0 \cdot x = 0$

1.4 Mnoho-druhové algebraické systémy.

obr. **??** Mnoho-druhové algebry zohrávajú významné miesto v programovaní, napríklad pri algebraických špecifikáciach abstraktných dátových typov (ADT). Sú zovšeobecnením pojmov *model* a *algebra*. Ich význačnou črtou je, že operácie a predikáty sú *polymorfnej* povahy, čo znamená, že definičné obory pozostávajú z množín prvkov rôznej povahy. Povedané podložíme príkladom.

Príklad 1.6

Ako ilustrácia nám poslúžia tri množiny *SEQ(qq, q=1,2,3* na ktorých budeme skúmať signatúru predikátov a operácií.

$$\begin{aligned} SEQ(1) &= \{M(1)_i \,|i \in I\} \\ SEQ(2) &= \{M(2)_j \,|j \in J\} \\ SEQ(3) &= \{M(3)_k \,|k \in K\} \end{aligned}$$

kde

- $M(1)_i : Ha_1a_2...a_\ell Y(1)a_{\ell+1}...a_nK$ je (označená) číselná postupnosť;
- $M(2)_j : Ht_1t_2...t_iY(2)a_{i+1}...a_mK$ je (označená) postupnosť symbolov;
- $M(3)_k : Hz_1z_2...z_rY(3)z_{r+1}...z_pK$ je (označená) postupnosť záznamov;

Na množinách *SEQ(q), q=1,2,3* sme v odseku 1.1 definovali niektoré elementárne predikáty a operátory. Tu použijeme ten istý prístup pre naše tri množiny.

Predikáty

Všetky nižšie uvádzané predikáty sú definované na množinách *SEQ(q)* a symbolom *Y(q)* bude označený ukazovateľ aktuálnej pozície v aktuálnej $M(q)_r$, pre príslušné *q=1,2,3* a $r \in \{i, j, k\}$ označenej postupnosti .

1) $\ell > r | Y(q) = 1 \quad \Leftrightarrow_{df} \quad \mathbf{M}(\mathrm{q})_r = \mathbf{M}' : \mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}(q), a_{i+1}, ..., a_n\mathbf{K} \wedge a_i > a_{i+1}$
2) $d(Y(q), K) = 1 \quad \Leftrightarrow_{df} \quad \mathbf{M}(\mathrm{q})_r = \mathbf{M}' : \mathbf{H}a_1, a_2, ..., a_n, \mathbf{Y}(q)\mathbf{K}$
3) $UM = 1 \quad \Leftrightarrow_{df} \quad \mathbf{M}(\mathrm{q})_r = \mathbf{M}' : \mathbf{H}\mathbf{Y}(q), a_1, a_2, ..., a_n\mathbf{K} \wedge a_i < a_{i+1}, i = 1, 2, ..., n-1$

Predpokladá sa, že na množinách *SEQ(q)* sú definované binárne relácie lineárneho usporiadania: čísel, symbolov, reťazcov (lexikografické usporiadanie).

Operátory

Podobne ako v predošlom prípade definujeme si operátory, na množinách *SEQ(q)*.

4) $E \quad \Leftrightarrow_{df} \quad E(\mathbf{M}(q)_r) = \mathbf{M}(q)_r$

5) $P(Y(q)) \quad \Leftrightarrow_{df} \quad \mathbf{M}(q)_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}(q), a_{i+1}, ..., a_n\mathbf{K}\wedge$
$\mathbf{M}(q)_2 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, a_{i+1}, \mathbf{Y}(q), a_{i+2}, ..., a_n\mathbf{K}\wedge$
$P(Y_1)(M(q)_1) = M(q)_2$

6) $L(Y(q)) \quad \Leftrightarrow_{df} \quad \mathbf{M}(q)_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}(q), a_{i+1}, ..., a_n\mathbf{K}\wedge$
$\mathbf{M}(q)_2 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ...a_{i-1}, \mathbf{Y}(q), a_i, , a_{i+1}, ..., a_n\mathbf{K}\wedge$
$P(Y_1)(M(q)_1) = M(q)_2$

7) $TRANSP(\ell, r) \quad \Leftrightarrow_{df} \quad \mathbf{M}(q)_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}(q), a_{i+1}, ..., a_n\mathbf{K}\wedge$
$\mathbf{M}(q)_2 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_{i+1}, \mathbf{Y}(q), a_i, ..., a_n\mathbf{K}\wedge$
$TRANSP(\ell, r)(M(q)_1) = M(q)_2$

8) $UST(Y(q), H) \quad \Leftrightarrow_{df} \quad \mathbf{M}(q)_1 = \mathbf{M}{:}\mathbf{H}a_1, a_2, ..., a_i, \mathbf{Y}(q), a_{i+1}, ..., a_n\mathbf{K}\wedge$
$\mathbf{M}(q)_2 = \mathbf{M}{:}\mathbf{HY}(q)a_1, a_2, ..., a_i, a_{i+1}, ..., a_n\mathbf{K}\wedge$
$UST(Y(q), H)(M(q)_1) = M(q)_2$

□
(Koniec príkladu)

Zavedené predikáty a operácie , ako ukazuje príklad, sú polymorfné-sú definované na každej množine SEQ(q), q=1,2,3. Teda na množinách SEQ(q) je definovaná signatúra pozostávajúca z polymorfných predikátov a operácií 1)-8).

Definícia 1.1

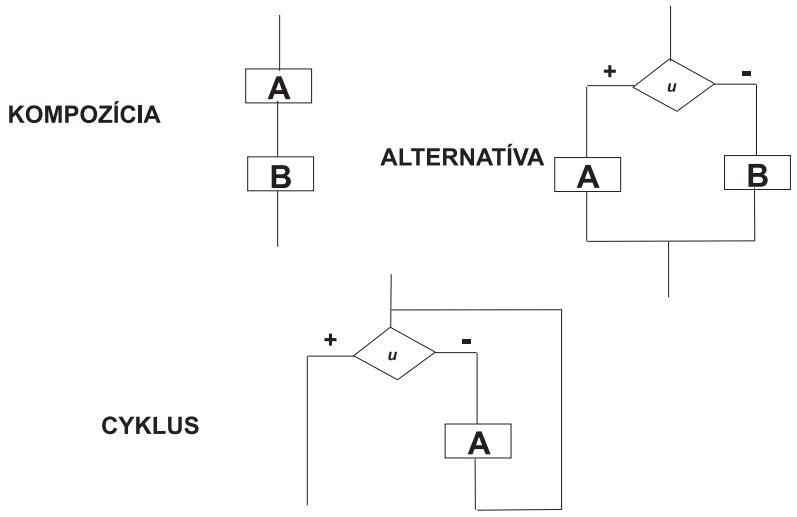
Mnoho-druhovým algebraickým systémom voláme sytém $AS = (Osnovy;$ Signatúra $)$, kde $Osnovy = \{A_i \,|i \in \mathbf{I}\}$ a Signatúra $= \{SIGN_{\Pi} \cup SIGN_o\}$, ktorá je daná zjednotením predikátov a operácií definovaných na množine osnov.

□

Mnoho-druhové algebraické systémy, ako boli definované v Def. 1.1, sú zovšeobecnemím pojmov model a algebry, tak ako boli uvedené v odseku 1.2resp. 1.3. Skutočne 1-druhový, alebo m-druhový model sú mnoho-druhové algebraické systémy v ktorých $SIGN_o = \Phi$. Ďalším vážnym prípadom mnoho-druhového algebraického systému je 1-druhová, alebo m-druhová algebra. Ide o algebraický systém v ktorom $SIGN_{\Pi} = \Phi$.
Teoretický fundament algoritmiky tvoria algebry algoritmov. Na ilustráciu teraz uvedieme jednoduchú algebru algoritmov, ktorá je orientovaná na analytické vyjadrenie algoritmov, teda na vyjadrenie algoritmu ako niektorej formuly v takej algebre.
Definujeme si 2-druhovú algebru $DA = (\{YC, OP\}; SIGN)$, v ktorej YC,OP sú osnovy algebry DA, pričom YC je množina predikátov (podmienok) definovaných na spracovaných dátach, OP je množina operátorov (operácií) na spracovanie dát. Signatúra operácií SIGN je definovaná na osnovách YC,OP. Do SIGN patria logické (boolovské) operácie: *dizjunkcia* ($\vee$), *konjunkcia* ($\wedge$) a *negácia* ($\neg$)- tieto sú definované na osnove YC. Na osnove OP sú definované operácie: kompozícia ($*$), alternatíva a cyklus. Pre tieto operácie budeme používať tieto označenia :

kompozícia	$A * B$
alternatíva	$([u]\, A, B)$
cyklus	$\{[u]\, A\}$

Význam $A * B$ je, že najprv sa vykoná A a po jej skončení sa začne vykonávať B.
Alternatíva $([u]\, A, B)$ sa dá vyjadriť ako **if** u **then** A **else** B, význam ktorej je v programátorskej komunite jasný.
Cyklus $\{[u]\, A\}$ sa dá vyjadriť v Pascal-ovskej notácii ako **while** $\neg u$ **do** A. Na obr. 1.2 je význam uvedených operácií uvedený v grafickej forme.



Obrázok 1.2: Grafická reprezentácia operáciií algebry DA

Signatúra SIGN zahŕňa známe programové konštrukcie zavedené E.W.Dijkstrom [47].
Orientácia na určitú triedu úloh si žiada aby sme pre algebru AD zvolili vhodný systém generátorov algebry. To znamená výber množiny elementárnych (pre danú úroveň úvah) operátorov a predikátov, pomocou ktorých operáciou superpozície je možno vytvárať zložitejšie operácie a predikáty, ktoré sú prvkami osnov YC a OP algebry DA.
Nech $S = S_u \cup S_o$ je systém generátorov algebry DA, kde $S_u \subset YC$ a $S_o \subset OP$. Analytické vyjadrenie zloženého operátora $F \in OP$ v tvare algebraického termu, vytvoreného superpozíciou prvkov z S a operácií zo SIGN voláme *štruktúrnou schémou* (jednoduchšie schémou) opertátora F v danej algebre.

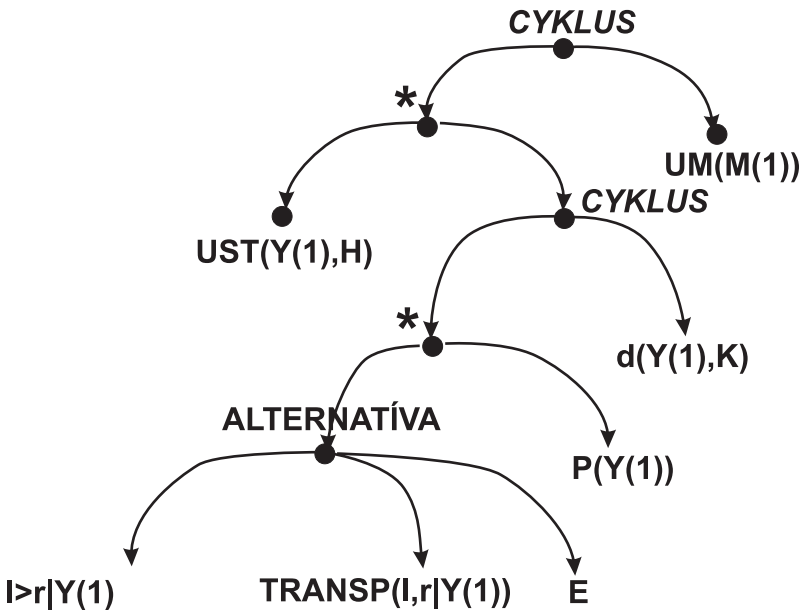
Príklad 1.7

Zvolíme si za systém generátorov algebry DA súbor 8 operácií a predikátov navrhnutých v odsekoch 1.4 a 1.4. Potom známy algoritmus bublinkového triedenia *BubbleSort* sa vyjadruje termom $Bubble ::= \left\{[UM(M(1))]\, \{[d(Y(1), K)]\, ([\ell > r | Y(1)]\, TRANSP\,(\ell, r, |Y(1))), E)^*\, P\,(Y(1)\}^*\, UST(Y(1), H)\right\}$

Grafová reprezentácia operácií vytvárajúcich term Bubble je na obr. 1.3.

□
(Koniec príkladu)

Aparát algebier algoritmov je zameraný na transformácie analytických reprezentácií algoritmov s cieľom zvýšenia efektívnosti ich funkčnosti z hľadiska časovej a pamäťovej zložitosti, a ďalších kritérií.



Obrázok 1.3: Grafová reprezentácia termu Bubble

Kapitola 2

Algebry algoritmov, logiky a schematológie

V tejto kapitole sa venujeme tvorbe algebier algoritmov orientovaných na formálny opis štruktúrnych a neštruktúrnych schém, ktoré budú reprezentované v analytickej, lingvistickej, ako aj v grafovej forme. Signatúra každej z týchto algebier pozostáva z opertátorových a predikátových konštrukcií zodpovedajúcich zvolenej metóde konštruovania algoritmov.

2.1 Formalizované projektovanie algoritmov.

V algoritmike sú hlavným objektom štúdia a skúmania schémy algoritmov a ich superpozície-vkladanie jednej schémy namiesto prvkov v druhej schéme. So superpozíciou je úzko spätý proces *rozvinutia* schémy (evolúcia), ktorý spočíva v projektovaní algoritmu zhora-nadol (top-down), ako aj proces *zvinutia*schémy (involúcia)- prechod k formalizovanej špecifikácii vyššej úrovne. Používa sa aj kombinovaný prístup, pri ktorom sa vyu žívajú tak zvinutie ako aj rozvinutie schém.
V úlohe elementárnych stavebných prvkov pre návrh schém algoritmov budeme používať schémy, ktoré sme uviedli v **??**, ktoré zodpovedajú základným konštrukciam štrukturovaného programovania. Začneme príkladom. Tu aj ďalej budeme budeme používať na označenie operátorov veľké písmena A,B,C,... a na označenie predikátov používame písmeno u, resp. u_i s indexom. Ako obyčajne zložené logické podmienky budeme vytvárať pomocou boolovských operácií: dizjunkcia ($u \vee u^{'}$), konjunkcia ($u \wedge u^{'}$), negácia ($\neg u$), alebo ($\overline{u}$).

Príklad 2.1

Navrhneme pomocou uvedených elementárnych koštrukcií netriviálnu zloženú schému Π, ktorá je špecifikáciou štruktúry algoritmov určitej triedy.

$$\begin{array}{lll} \Pi & ::= & \{[u_1]\, A_1\}\,, \\ A_1 & ::= & \{[u_2]\, A_2 * D\}\,, \\ A_2 & ::= & A_3 * C, \\ A_3 & ::= & ([u]\, A, B)\,, u ::= \overline{u}_2 \wedge u_3. \end{array}$$

Vykonáme teraz superpozíciu - zvinutie procesu Π, ktorý je nateraz uvedený ako postupnosť evolučných krokov (špecifikáciou A_1, A_2, A_3). Po podtupnom nahradení premenných A_1, A_2, A_3 dostaneme formulu

$$\Pi ::= \{[u_1]\, \{[u_2]\, ([\overline{u}_2 \wedge u_3]\, A, B) * C * D\}\}$$

□
(Koniec príkladu)

Schéma Π je príklad *neiterpretovanej* schémy. Prechod od neiterpretovaných schém k algoritmom je spojený s iterpretáciou operátorových a logických (predikátových) premenných v neiterpretovanej schéme. Tvorba interpretácií je podmienená špeciálnym označením spracovávaných dát tak, ako to bolo v prípade označených postupností na množinách SEQ, resp. USQ (odsek 1.1). Tak v uvedenom prípade sa inerpretácia operátoroových a logických premenných realizuje na označených postupnostiach.

Príklad 2.2

Majme označenú postupnosť

$$\mathbf{M} : \mathbf{H}a_1, a_2, ..., a_i, Y_1, a_{i+1}, ..., a_n\mathbf{K}$$

Zavedieme si teraz čiastočnú interpretáciu premenných schémy Π

$$u_2 \to d(Y_1, K),\ C \Rightarrow P(Y_1),\ D \Rightarrow UST(Y_1, H)$$

kde symboly $\to$ a $\Rightarrow$ označujú interpretáciu zodpovedajúco logických a operátorových premenných v schémach. S touto čiastočnou interpretáciou schémy Π dostávame čiastočne interpretovanú schému $C\Pi$

$$C\Pi ::= \left\{[u_1]\left\{[d(Y_1, K)]\left(\left[\overline{d(Y_1, K)} \wedge u_3\right] A, B\right) * P(Y_1) * UST(Y_1, H)\right\}\right\}$$

□

(Koniec príkladu)

Zmysel schémy $C\Pi$ pri zodpovedajúcej interpretácii jej logických a operátorových premenných , môže spočívať v cyklickom prehliadaní (skanovaní) postupnosti $\mathbf{M}$ v smere zľava doprava až do pravdivosti predikátu (podmienky) $d(Y_1, K)$ s nasledujúcim návratom indikátora Y_1 k značke $\mathbf{H}$ a výstupom na opakovanie vonkajšieho cyklu. Opísaný proces bude pokračovať do doby , kedy začne platiť podmienka u_1. V priebehu takéhoto mnohofázového skanovania sa realizuje spracovanie aktuálnych symbolov postupnosti operátormi A alebo B vnorenej alternatívy (v závislosti od hodnoty podmienky u_3).
Čiastočne interpretované schémy sa volajú *stratégiami* spracovania symbolov. V príklade (Príklad 2.2) uvedená stratégia $C\Pi$ je stratégiou mnoho-fázového spracovania postupnosti $\mathbf{M}$ metódou BubbleSort. Poznamenávame, že v závislosti od výberu interpretácie logických a operátorových premenných schémy $C\Pi$ môžu byť vytvorené algoritmy riešenia rôznych aplikačných úloh. To ilustruje tento príklad.

Príklad 2.3

Majme $\mathbf{M}$ číselnú postupnosť; zavedieme si interpretáciu logických a operátorových premenných schémy $C\Pi$.

$$u_1 \to UM,\ u_3 \to l > r|Y_1,\ A \Rightarrow TRANSP(l, r),\ B \Rightarrow E$$

S touto interpretáciou logických a operátorových premenných schémy $C\Pi$ získavame algoritmus

$$BUBBLE ::= \left\{[UM]\left\{[d(Y_1, K)]\left(\left[\overline{d(Y_1, K)} \wedge l > r|Y_1\right] TRANSP(l, r), E\right) * P(Y_1) * UST(Y_1, H)\right\}\right\}$$

□

(Koniec príkladu)

Záverom poznamenávame, že uvedený postup spočívajúci v postupnej interpretácii logických a operátorových premenných schém , resp. stratégií privádza ku konštrukcii konkrétnych algoritmov riešenia úloh danej konkrétnej problémovej oblasti.

2.2 Dijkstrova algebra.

V tejto časti sa zaoberáme tvorbou algebry algoritmov, ktorá sa volá Dijkstrova algebra (AD). Algebra je pomenovaná podľa holandského informatika E.W.Dijkstru. Dijkstra je autorom koncepcie tzv. štrukturovaného programovania a už uvádzaných konštrukcií:sekvencia, alternatíva a cyklus. Dijkstra ešte v r.1968 vo svojom dopise Akadémii vied USA navrhoval vytvorenie algebry algoritmov založenej na týchto konštrukciach.

Definícia 2.1 *Dijkstrova algebra je 2-druhový algebraický systém* $AD = (ACC, L(2); SIGN)$ *osnovami ktorej sú: množina operátorov ACC, pozostávajúca zo štruktúrnych schém a množina boolovských funkcií L(2). Signatúru SIGN vytvárajú operácie sekvencia, alternatíva a cyklus (ktoré nadobúdajú hodnoty patriace do ACC) a boolovské operácie konjunkcia (∧), dizjunkcia (∨), negácia (¬)(ktoré nadobúdajú hodnoty patriace do L(2)). Premenné* $A = \{A_1, A_2, ..., A_n\}$ *a* $U = \{u_1, u_2, ..., u_m\}$ *sa využívajú na označenie zodpovedajúco elementárnych (bázových) operátorov a logických podmienok.*

Príkladom zloženej operátorovej schémy v algebre AD je schéma Π z príkladu 4.4.
Pre AD bol G.E.Cejtlinom [15] rozpracovaný systém transformácií, ktorý charakterizuje povahu vlastností operácií vystupujúcich v SIGN. Tak napríklad spojitosť medzi alternatívou a cyklom vyjadruje rovnosť

$$\{[u]\, A\} = ([u]\, E, A * \{[u]\, A\}) \tag{2.1}$$

Okrem základných operácií vyskytujúcich sa v SIGN sa v AD využívajú aj ďalšie , z nich odvodené, operácie, ktoré sa získajú ako výsledok superpozície základných operácií a konštánt danej algebry . Tak v algebre čísel je operácia násobenia odvodená z operácie sčítania, zatiaľ čo v boolovej algebre je operácia dizjunkcie odvoditeľná z konjunkcie a negácie.
V AD k odvodeným operáciam patrí operácia *filtrácie*:$\Phi(u) = ([u]\, E, N)$, ktorá je získaná ako výsledok superpozície alternatívy, operátora E a neznámeho operátora N (ktoré v AD vystupujú ako konštanty). Operátor N preruší výpočet podľa vetvy *false*, takže operátor - filter Φ dovoľuje pokračovanie výpočtu iba ak je u=*true*.
Ďalšou odvodenou operáciou je *zovšeobecnený cyklus* (operátor **DO-WHILE-DO**), ktorý je daný superpozíciou operácií kompozície a cyklu a ktorý realizuje výstup z cyklu podľa podmienky nachádzajúcej sa v strede cyklu:

$$\{A\,[u]\,B\} = A * \{[u]\, B * A\}$$

Ak A=E dostávame základný cyklus

$$\{E\,[u]\,B\} = \{[u]\, B\}$$

a pri B=E dostaneme cyklus typu **DO-WHILE**, v ktorom sa test podmienky vykoná po prechode telom cykla

$$\{A\,[u]\,E\} = \{A\,[u]\}$$

Operácie alternatívy a cyklu majú tieto vlastnosti:

$$\{[u]\, A\} = \{[u]\, \Phi(\overline{u}) * A\} \tag{2.2}$$

$$\left\{[u]\, \Phi(\overline{u}) * \left(\left[\overline{u} \wedge u'\right] A, B\right) * C\right\} = \left\{[u]\, \Phi(\overline{u}) * \left(\left[u'\right] A, B\right) * C\right\} \tag{2.3}$$

Príklad 2.4

Uvedieme teraz proces transformácie schémy Π z príkladu 4.4 do kompaktnejšieho tvaru s použitím rovností (2.2) a (2.3). Proces transformácie bude sprevádzaný komentármi, ktoré sú uvedené v komentárových zátvorkach (/* a */).

$$\Pi ::= \{[u_1]\, \{[u_2]\, ([\overline{u}_2 \wedge u_3]\, A, B) * C * D\}\}$$

/* Použijeme rovnosť (2.2), k formovaniu filtra $\Phi(\overline{u})$ pred vloženou alternatívou*/

$$= \{[u_1]\, \{[u_2]\, \Phi(\overline{u}_2)\, ([\overline{u}_2 \wedge u_3]\, A, B) * C * D\}\} =$$

/* Na základe rovnosti (2.3) dochádza k absorbcii (pohlteniu) prvého konjunktívneho súčiniteľa v podmienke vloženej alternatívy */

$$= \{[u_1]\, \{[u_2]\, \Phi(\overline{u}_2)\, ([u_3]\, A, B) * C * D\}\} =$$

/* Použijeme rovnosť (2.2) v opačnom smere */

$$= \{[u_1]\, \{[u_2])\, ([u_3]\, A, B) * C * D\}\}\,.$$

Realizovanú postupnosť transformačných krokov môžeme považovať za formálny dôkaz (odvodenie) rovnosti:

$$\left\{[u]\, (\left[\overline{u} \wedge u'\right] A, B) * C\right\} = \left\{[u]\, (\left[u'\right] A, B) * C\right\}.$$

□
(Koniec príkladu)

Ak teraz aplikujeme interpretácie premenných schémy Π ako v príkladoch 2.2 a 2.3 dostaneme kompaktnejšie vyjadrenie algoritmu BubbleSort:

$$BUBBLE ::= \{[UM]\,\{[d(Y_1, K)]\,([l > r|Y_1]\,TRANSP(l, r), E) * P(Y_1)\} * UST(Y_1, H)\}$$

Žiada sa poznamenať, že predvedený proces transformácie predstavuje silný nástroj tvorby a získavania nových poznatkov algoritmickej povahy. O tom sa čitateľ presvedčí v ďalšom.

2.3 Algebra schém Janova.

V tejto časti sa zaoberáme tvorbou algebry algoritmov, ktorá sa volá algebra Janova(AJ). Algebra je pomenovaná podľa ruského informatika J.Janova, ktorý je autorom operátorových schém, ktoré sa volajú *schémy Janova* [37],[**?**].
Prv než pristúpime k samotnej definícii algebry Janova uvedieme niekoľko poznámok:

1. Operátorové schémy algoritmov Janova (v ďalšom iba schémy Janova) sú založené na špeciálne označených postupnostiach (viď ďalej)
2. Schémy Janova patria do kategórie *neštrukturovaných* schém algoritmov;
3. Signatúra AJ si vyžaduje, vychádzajúc z povahy operátorových schém Janova, zavedenie dvoch operátorov: kompozícia $A * B$ a podmienený prechod $\Pi(u)\downarrow_m$, ktorý, pri pravdivej podmienke u, odovzdáva riadenie na značku (marker) m. V opačnom prípade sa realizuje, v realizovanej(spracovávanej) postupnosti operátorov, napravo od aktuálnej pozície stojaci operátor. Operátor podmieneného prechodu $\Pi(u)\downarrow_m$ zodpovedá príkazy *GO TO* v programovacích jazykoch.
4. Nech **F** je neštrukturovaná logická schéma (v ktorej sú použité operácie kompozície a podmieneného prechodu), na ktorú možno nazerať ako na označenú symbolovú postupnosť, ktorá je označená dvomi markermi:**I** a m: . Marker **I** môže byť umiestnený buď na začiatku schémy **F**, alebo hneď za symbolom $*$, alebo $\downarrow$. Marker *m:* označuje ľubovoľný výskyt niektorého operátorového podvýrazu v schéme **F**. Podotýkame, že zavedenie takých označení v schéme **F** nenarúša poradie realizácie operátorov v nej. V tom prípade podmienený operátor možno interpretovať ako niektorú binárnu operáciu $\Pi(u, F)$, ktorá závisí od podmienky u a označenej schémy **F**. Výsledok aplikácie takej operácie je nová schéma $\mathbf{F}'$, v ktorej je marker **I** v schéme **F** nahradený symbolom $\Pi(u)\downarrow_m$.

Príklad 2.5

Nech je daná schéma $F_1 ::= A * B * C$. Vytvoríme označenú schému $F_1 ::= A * \mathbf{m:}B * \mathbf{I}\ C$. Aplikácia operácie $\Pi(u, F_1)$ na F_1 nám dá schému

$$F_1' = \Pi(u, F_1) ::= A * \mathbf{m:}B * \Pi(u)\downarrow_m C$$

Všimnime si, že aplikácia operácie $\Pi(u, F_1)$ na sekvenčnú schému F_1 priviedla k cyklickej schéme F_1'. Špeciálnym prípadom operácie $\Pi(u, F_1)$ pri u=1 je operácia bezpodmienkového prechodu $\Pi\downarrow_m$, ktorá závisí len od označenej schémy F_1.

□
(Koniec príkladu)

Definícia 2.2 *Janovova algebra je 2-druhový algebraický systém* $AJ = (\{AHC, L(2)\}; SIGN')$ *osnovami ktorej sú: množina AHC neštrukturovaných (označených i neoznačených) schém a množina boolovských funkcií L(2). Signatúru* $SIGN'$ *vytvárajú operácie kompozície* $A * B$*, neštrukturovaného prechodu* $\Pi(u, F)$ *a boolovské operácie konjunkcia (*$\wedge$*), dizjunkcia (*$\vee$*), negácia (*$\neg$*)(ktoré nadobúdajú hodnoty patriace do L(2)).*

Medzi odvodené operácie AJ patria *alternatíva* a *cyklus*, ktoré sú v signatúre algebry AD - SIGN, ako aj cyklus **DO-WHILE**. Nasleduje formálna reprezentácia uvedených operácií, ktorá je zároveň aj formálnym dôkazom platnosti uvedeného tvrdenia.

$$\begin{aligned}
([u]A, B) &= \Pi(u) \downarrow_m B * \Pi \downarrow_{m'} m : A * m^{'} : E, \\
\{[u]A\} &= m : \Pi(u) \downarrow_{m'} A * \Pi \downarrow_m * m^{'} : E, \\
\{A[u]\} &= m : A * \Pi(\overline{u}) \downarrow_m E, \qquad (2.4) \\
\{A[u]\} &= A * \left\{ [u]\, A^{'} \right\}, \qquad (2.5)
\end{aligned}$$

kde $A = A^{'}$, alebo A sa líši od $A^{'}$ prítomnosťou značiek.
Definícia algebry AJ a jej porovnanie s algebrou AD umožňuje urobiť tento záver.

Theorem 2.1 *V algebre AJ je vyjadriteľná ľubovoľná štruktúrna schéma, ktorá je vyjadriteľná v algebre AD;inými slovami platí, že ACC* $\subset$ *AHC a teda vyjadrovacia sila AJ je väčšia ako AD. Tu ACC je množina schém vyjadriteľných v AD a AHC je množina schém vyjadriteľných v zodpovedajúcej algebre Dijkstru AD algebre Janova AJ.*

Príklad 2.6

Ukážeme ako je možno pretransformovať štrukturovanú schému na neštrukturovanú pomocou aplikácie uvedených rovností.
Ako príklad si vezmeme algoritmus opisujúci v AD fungovanie abstraktnej tlačiarne:

PRINT ::=
TLAČ PRVÉHO RIADKU∗
$\{[eof]$ TLAČ AKTUÁLNEHO RIADKU$\}$;

TLAČ PRVÉHO RIADKU ::=
$P(Y_1) *$ TLAČ∗
$\{[eol]\, P(Y_1) *$ TLAČ$\} *$
CRL;

TLAČ AKTUÁLNEHO RIADKU ::=
TLAČ $* \{[eol]\, P(Y_1) *$ TLAČ$\} *$ CRL;

kde:

- *eof* je predikát označujúci koniec súboru (**e**nd **o**f **f**ile);
- *CRL* je operácia zabezpečujúca návrat hlavy a posun o 1 riadok (**c**arriage **r**eturn and **l**ine);
- *eol* je predikát označujúci koniec riadku (**e**nd **o**f **l**ine).

Predpokladá sa, že v počiatočnom stave indikátor Y_1 sa nachádza naľavo od 1. symbolu súboru, ktorý bude vytlačený a pri prechode na nový riadok sa indikátor Y_1 nastavuje na 1. symbol nového riadku.
Prejdeme teraz k neinterpretovanej schéme S(PRINT), ktorá odráža štruktúru schémy PRINT:

$$S(PRINT) ::= A * B\, \{[u]\, A * B\} * C * \left\{ \left[u^{'} \right] B * \{[u]\, A * B\} * C \right\}$$

kde¿ A::= $P(Y_1)$, B::= TLAČ, u::= eol, C ::= CRL, $u^{'}$ ::= eof.
Pretransformujeme teraz schému S(PRINT) na jej neštrukturovaný ekvivalent s použitím vyššie uvedených rovností:
S(PRINT) = /*pretože ľavý kontext základného cyklu B $\{[u]\, A * B\} * C$ je totožný s jeho telom, použijeme rovnosť (2.5) v smere zprava doľava */

$$= A * B \left\{ B * \{[u]\, A * B\} * C \left[u^{'} \right] \right\} =$$

/*Použijeme rovnosť (2.4) */

$$= A * m^{'} : B * \{[u]\, A * B\} * C * \Pi(u^{'}) \downarrow_{m'} =$$

/*Znovu použijeme rovnosť (2.5) */

$$= \left\{A * m^{'} : B\,[u]\right\} * C * \Pi(u^{'}) \downarrow_{m'} =$$

/*Napokon znovu použijeme rovnosť (2.4) */

$$= m : A * m^{'} : B * \Pi(u) \downarrow_{m} C * \Pi(u^{'}) \downarrow_{m'} .$$

Ak zodpovedajúcim spôsobom interpretujeme získanú neštrukturovanú schému, získame ekvivalentnú reprezentáciu algoritmu PRINT v AJ.

□
(Koniec príkladu)

Je treba zdôrazniť, že ekvivalentnosť použitých krokov zaručujje korektnosť neštrukturovanej reprezentácie a znamená aj bezpečnosť jej použitia. Podotýkame, že štrukturované a neštrukturované schémy sa navzájom dopĺňajú. Pre prvé je príznačná transparentnosť zatiaľ čo druhým je vlastná kompaktnosť a univerzalizmus.

2.4 Graf-schémy algoritmov.

V r. 1957 ruský (vtedy sovietsky) informatik L.A.Kalužnin navrhol vytvorenie algebry algoritmov založenej na grafickej ich reprezentácii pomocou tzv. *graf-schém*, ktoré sa u nás volali *vývojové diagramy.* Pre jednoduchosť budeme ďalej používať pojem graf-schéma, ako synonymum pre *vývojový diagram.*
Uvedieme teraz formálnu definíciu graf-schém algoritmov:

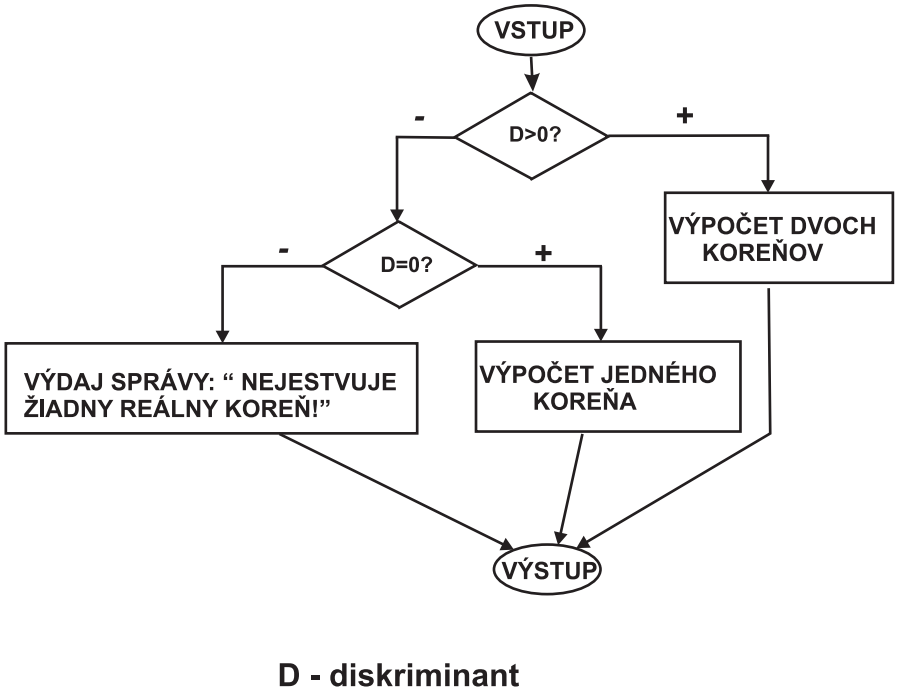
Definícia 2.3

Zvolíme si pevne dve množiny premenných $\underline{A} = \{A_i \,|i = 1, 2, ..., n\}$ a $\underline{U} = \{U_j \,|j = 1, 2, ..., m\}$, kde A_i sú operátorové a U_j sú logické premenné. Nech $G = (V, \vec{E})$ je orientovaný graf, kde V je množina vrcholov a $\vec{E}$ je množina orientovaných hrán, ktoré sú vlastne binárnou reláciou $R \subseteq V \times V$ na V, $e = (v_1, v_2) \in R \Leftrightarrow_d f v_1, v_2 \in \vec{E}$ sú spojené orientovanou hranou, t.j. $e = (v_1, v_2) \in \vec{E}$.
Definuje sa určité zobrazenie $\ell : V \to \underline{A} \cup \underline{U}$, ktoré každý vrchol grafu $v \in V$ označuje symbolom - $\ell(v)$ z množiny $(\underline{A} \cup \underline{U})$. Vrchol označený premennou $A_i \in \underline{A}$ sa volá operátorovým vrcholom (A-vrcholom) a označený premennou $U_i \in \underline{U}$ sa volá rozpoznávajúcim (testovacím) vrcholom (niekedy U-vrcholom). Hrany grafu sú 2 typov: obyčajné a logické; logické sú označené + (plusom), alebo - (mínusom). Z každého A-vrchola vychádza jedna obyčajná hrana, ktorá vedie do iného vrchola grafu G, zatiaľ čo z U-vrchola vedú dve hrany:plusová a mínusová. Rôzne A-vrcholy(U-vrcholy) môžu byť označené rovnakou operátorovou (logickou) premennou.
Graf G má dva doplňujúce vrcholy: $VSTUP$ do ktorého nevedie žiadna hrana a $VÝSTUP$z ktorého nevedie žiadna hrana.
Takto definovaný *súvislý* graf G sa volá *graf-schémou* algoritmu.

□

Na obr. 2.1je príklad graf-schémy algoritmu riešenia kvadratickej rovnice.

Prejdime teraz k problematike návrhu *algebry graf-schém.*
Nech sú dané dve graf-schémy G a $G^{'}$ na rovnakej rodine množín operátorových a predikátových premenných $\underline{V} = \underline{A} \cup \underline{U}$. Zafixujeme si v graf-schéme G všetky výskyty niektorého operátora $A_j \in \underline{A}$ a nahradíme všetky také výskyty A_j v G graf-schémou $G^{'}$. Výsledkom bude nová graf-schéma $G^{''} = G(A_j \Rightarrow G^{'})$ ako



Obrázok 2.1: Graf-schéma algoritmu riešenia kvadratickej rovnice.

superpozícia graf-schém G a $G^{'}$. V prípade, že graf-schéma $G^{'}$ nahradzuje testovací vrchol (so značkou u) vzniká vedľajší efekt- v procese výpočtu nahradzujúci algoritmus "kazí" dáta, ktoré sú vstupom pre testovací vrchol. Tým sa vylučuje možnosť prechodu po plusovej, alebo mínusovej hrane tým dátam, na ktorých sa mala, v súlade s povahou rozpoznávateľa, realizovať previerka predikátu u. Tento problém bol vyriešený vďaka zovšeobecneniu pojmu graf-schémy, aby bola zaručená korektnosť superpozície $G^{''} = G(u \Rightarrow G^{'})$.
Zavádza sa 2-druhová algebra AK=*({OP, YC};SIGN)*, kde OP - množina operátorových , YC-množina logických podmienok reprezentovaných v grafovej forme (viď ďalej); SIGN-signatúra operácií zahrňujúca boolovské operácie, operáciu kompozície reprezentovanej prostredníctvom orientovanej hrany spájajúcej dva operátorové vrcholy graf-schémy; unárnu operáciu *reštrukturalizácie* grafu - prepnutie jednej z jeho orientovaných hrán vychádzajúcej z niektorého operátorového vrchola, alebo rozpoznávača a jej napojenie na niektorý iný vrchol graf-schémy (g-s) algoritmu s následným odstránením ´visiaxich´- nedosiahnuteľných z koreňa g-s vrcholov. Operácia reštrukturalizácie zodpovedá operácii prechodu známej zo schém Janova a vchodiacej do jej signatúry (viď časť 2.3).
Poznamenávame, že zavedenie 2 osnov a doplňujúceho obmedzenia, ktoré vylučuje výskyt operátora v g-s, ktorá predstavuje niektorú logickú podmienku, zaručuje korektnosť superpozície g-s.
Takto vytvorená 2-druhová algebra sa volá *algebrou Kalužnina*(AK).

2.5 Systémy algoritmických algebier Gluškova.

Podobne algebraickým špecifikáciam sú systémy algoritmických algebier Gluškova (SAA) orientované na analytickú formu reprezentácie algoritmov a akokoľvek hlbokú formalizovanú transformáciu takej reprezentácie, špeciálne s cieľom optimalizovať algoritmy podľa zvolených kritérií.
SAA sú získané z AD rozšírením jej signatúry o operáciu *prognozovnia.*
Prognozovanie v procese konštruovania predstavuje efektívny prostriedok jeho riadenia, podľa známeho hesla Napoleona:riadiť -znamená predvídať!
Signatúra SAA obsahuje všetky operácie zo SIGN AD (viď časť 2.2); (ďalej) k takým operáciam patria boolovské operácie, zovšeobecnené na 3-hodnotový prípad; operácia prognozovania, ktorá je z formálne-

ho hľadiska operáciou ľavého násobenia podmienky u operátorom A: $u = A \bullet u'$. Táto operácia definuje predikát $u \in YC$ s vlastnosťou, že $u(m) = u'(m')$, kde $m' = A(m)$). $A \in OP$, $u' \in YC$ a $m, m' \in IM$‘ tu OP a YC je zodpovedajúco množina operátorov a množina podmienok a IM je informačná množina spracovávaných dát, na ktorej sú definované operátory z OP a podmienky z YC. Z toho plynie, že operácia prognozovania pozostáva z previerky (testu) podmienky u' po vykonaní operácie A. Táto previerka slúži ako prognóza o pokračovaní výpočtového procesu, ktorá sa realizuje prostredníctvom priradenia podmienke u v stave m(do vykonania operátora A) hodnoty u', ktorá sa vypočíta v stave m' do ktorého systém prejde po realizácii operátora A.

Príklad 2.7

Vysvetlíme si podstatu operácie prognozovania pri výpočte podmienky UM v príklade algoritmu BUBBLE$'$ (viď (Príklad 2.3)).Takú previerku pravdivosti predikátu UM je treba uskutočniť v procese triedenia postupnosti, bez nutnosti zavedenia ďalších ukazovateľov. Použijeme k tomu operáciu prognozovania:

$$UM ::= SKAN \bullet [d(Y_1, K)]\,,$$
$$\text{kde SKAN} ::= \{[d(Y_1, K) \vee (l > r | Y_1)]\, P(Y_1)\}. \tag{2.6}$$

Pri ´vstupe´ postupnosti $M \in SEQ$ na vstup predikátu UM realizuje sa prehliadanie (skanovanie) prvkov prostredníctvom premiestňovania ukazovateľa Y_1 doprava až do dosiahnutia značky **K**, alebo zafixovaní neusporiadanej dvojice (l,r). Po vykonaní tohto cyklu sa uskutoční previerka hodnoty predikátu $d(Y_1, K)$, ktorá sa priradí predikátu UM v súlade so sémantikou operácie prognozovania a Y_1 sa vracia k značke **H**.

□
(Koniec príkladu)

SAA Gluškova (označme ju AG) je definovaná ako

$$AG = (\{OP, YC\}; SIGN'')$$

kde OP, YC sú zodpovedajúco množina operátorov a logických podmienok definovaných na informačnej množine IM.
$SIGN'' = SIGN \cup \{progn\}$, pričom SIGN je signatúra AD a *progn* je operácia prognozovania.
Zafixujeme si bázu $\mathbf{I} \in (OP \cup YC)$; *Interpretovanou regulárnou schémou*(PC) $F/I \in OP$ sa volá superpozícia operácií zo signatúry $SIGN''$ a prvkov bázy **I**, ktorá predstavuje zložený operátor (algoritmus) **F/I** v AG.
Ďalej uvádzané výsledky boli získané v rámci výskumov týkajúcich sa rozvoja formalizmu založeného na algebrách a formálnych gramatikách a ktoré sa vzťahujú k AG [35].

Veta 2.1 *[35] Každý algoritmus A (vrátane programu, lebo mikroprogramu) môže byť reprezentovný niektorou PC F/I v AG, t.j. A=F/I. To znamená, že algebry Gluškova z hľadiska svojej vyjadrovacej sily sú totožné so známymi lgoritmickými systémami (ako napríklad Turingove stroje).*

□

Zvolíme si bázu algebry algoritmov ako množinu V operátorových a logických premenných $V = A \cup U$, kde $A = \{A_1, A_2, ..., A_m\}$ a $U = \{u_1, u_2, ..., u_n\}$; takú algebru budeme volať *neinterpretovanou.* Platí táto

Veta 2.2 *Pre neinterpretované algebry Dijkstry, Janova a Gluškova platia nasledujúce vzťahy vlastnej inklúzie:ACC $\subset$ AHC $\subset$ OP, kde ACC , AHC a OP sú triedy operátorových schém vyjadriteľných v zodpovedajúcich algebrách.*

□

Dôkazy vyššieuvedených výsledkov sú založené na príslušnosti operácie *progn* (prognozovania) k signatúre $SIGN''$.
Na formalizáciu nedeterministických a paralelných výpočtov sú orientované modifikované SAA, na ktorých je založená tzv. *štruktúrna schematológia*[35].
SAA Gluškova poslúžili ako prototyp programových logík a súčasným výskumom v oblasti funkcionálnych, algebraických a algebraicko-gramatických formalizmov a metódam transformačnej syntézy programov [32].

Kapitola 3

Algebra logiky a problém funkcionálnej úplnosti.

Potreba zaoberať sa algebrou logiky je daná dvomi príčinami:

1. princíp tvorby boolovských funkcií je použitý aj pre tvorbu algebier algoritmov; je vytvorená algebra schematológie ako meta-algebra, ktorá zahŕňa rôzne algebry algoritmov, vrátane tých, s ktorými sme sa oboznámili vyššie;

2. algebra logiky je významnou súčasťou algebry schematológie.

3.1 Algebry boolovských funkcií.

V tejto kapitole sa predpokladá, že čitateľ je oboznámený so základmi teórie boolovských funkcií (b.f.)v rozsahu ako v [50]:definícia b.f. n-premenných $f(x_1, x_2, ..., x_n)$; formy reprezentácie b.f.;kanonické tvary b.f. (udnf, ucnf).
Majme b.f. $f(x_1, x_2, ..., x_n)$; o b.f. f budeme hovoriť, že *podstatne závisí* od x_i ak sa nájde aspon jedna n-1-ica premenných $(a_1, a_2, ..., a_{i-1}, a_{i+1}, ..., a_n)$ a taká, že

$$f(a_1, a_2, ..., a_{i-1}, 0, a_{i+1}, ..., a_n) \neq f(a_1, a_2, ..., a_{i-1}, 0, a_{i+1}, ..., a_n) \tag{3.1}$$

Ak premenná x_i nie je podstatná pre f potom ju voláme *fiktívnou* premennou pre *f*.
O dvoch b.f. n premenných $f(x_1, x_2, ..., x_n)$ a $g(x_1, x_2, ..., x_n)$ hovoríme, že sú *duálne*, ak platí,

$$f(x_1, x_2, ..., x_n) = \overline{g}(\overline{x}_1, \overline{x}_2, ..., \overline{x}_n) \tag{3.2}$$

Príklad 3.1

Tabulka:B.f.2-premenných podstatne závislé od 2-premenných.

x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Oznacenie		$\wedge$	$\vee$	—	$\downarrow$	$\oplus$	$\equiv$	$\Rightarrow$	$\not\Leftarrow$	$\Leftarrow$	$\not\Rightarrow$
0	0	0	0	1	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0	1	1	0	0
1	0	0	1	1	0	1	0	0	0	1	1
1	1	1	1	0	0	0	1	1	0	1	0

kde

$f_1(x,y)$	=	$x \wedge y$	−konjunkcia	$f_6(x,y)$	=	$x \equiv y$	−ekvivalencia
$f_2(x,y)$	=	$x \vee y$	−dizjunkcia	$f_7(x,y)$	=	$x \Rightarrow y$	−implikácia
$f_3(x,y)$	=	$x\|y$	−-Shaefferova ciarka	$f_8(x,y)$	=	$x \not\Leftarrow y$	−obrátená antiimplikácia
$f_4(x,y)$	=	$x \downarrow y$	−-Pearceova šípka	$f_9(x,y)$	=	$x \Leftarrow y$	−obrátená implikácia
$f_5(x,y)$	=	$x \oplus y$	−súcet modulo 2	$f_{10}(x,y)$	=	$x \not\Rightarrow y$	− antiimplikácia

□

(Koniec príkladu)

Tak v tab. 3.1 sú funkcie f_1, f_2; f_3, f_4; f_5, f_6; f_7, f_8; f_9, f_{10} navzájom duálne. Skutočne, ak vezmeme prvú dvojicu f_1, f_2 máme

$$x \wedge y = \overline{\overline{x} \vee \overline{y}} \tag{3.3}$$

$$x \vee y = \overline{\overline{x} \wedge \overline{y}} \tag{3.4}$$

V teórii b.f. platí tzv. *princíp duality*:ak v teórii b.f. je určitý pojem, rovnosť, alebo tvrdenie, ktoré sú sformulované v pojmoch určitého systému b.f., tak potom existuje pojem, rovnosť, alebo tvrdenie, ktoré sú sformulované v pojmoch duálneho systému b.f.. Príkladom duálnych rovností sú rovnosti (3.3) a (3.4). Rovnosti (3.3) a (3.4) sú potvrdením možnosti vyjadrenia jedných b.f. pomocou iných prostredníctvom substitúcie - superpozície. Skutočne, vezmime rovnost $x \wedge y = \overline{z}$, kde $z = u \vee v$ a $u = \overline{x}, v = \overline{y}$. Ak nahradíme premenné *z,u,v* ich hodnotami (pravé strany zodpovedajúcich rovností) dostaneme vyjadrenie konjunkcie prostredníctvom superpozície dizjunkcie a negácií tak ako je to vidieť z (3.3).
Okrem toho pri vytváraní b.f. sa v algebre logiky (boolovej algebre) používajú aj dalšie operácie: *premenovanie premenných, stotožnenie dvoch alebo viac premenných a pridanie, alebo odobratie fiktívnych premenných.* Tak napríklad, ak v b.f. $\overline{x \vee y}$ stotožníme premenné x a y dostaneme $\overline{x \vee x} = \overline{x}$.

Definícia 3.1 *Algebraický systém AL=(BF;SIGN$_L$), kde BF je množina všetkých b.f. a SIGN$_L$ pozostáva zo superpozície, premenovania a stotožnenia (premenných), ako aj pridávania a odobratia fiktívnych premenných, voláme (dvojznacnou) algebrou logiky.*

Prejdeme teraz k vytvoreniu systému generátorov a úplných systémov v

AL - k problému *funkcionálnej úplnosti.* Systém b.f.

$\Omega \subset BF$ sa volá *úplný* systém b.f. ak lubovolná b.f.

z BF sa dá vyjadrit ako superpozícia funkciií z Ω.
Príkladom *úplného* systému b.f. je systém pozostávajúci z 3

b.f.:$x \vee y$, $x \wedge y$ a $\overline{x}$. Toto sa dá preukázat

aj formálne. Najprv uvedieme, že v teórii b.f. sa ujalo používanie symbolov

$+, \cdot$ namiesto symbolov $x \vee y$, $x \wedge y$, co umožnuje

jednoduchší a zaužívanejší zápis b.f., ak navyše budeme písat

jednoducho xy namiesto $x \cdot y$.

Veta 3.1 *Systém b.f. :$x + y$, $x \cdot y$ a $\overline{x}$ je úplný.*

Dôkaz: Z teórie b.f. [50] je známe, že každá bf. $f(x_1, x_2, ..., x_n)$ sa

dá jednoznacne vyjadrit v tzv. *úplnej dizjunktívnej normálnej forme (udnf)* , alebo v tvare tzv. *úplnej konunktívnej normálnej forme (ucnf)* (duálny tvar k udnf).Uvedieme formálny zápis pre undf:

$$f(x_1, x_2, ..., x_n) = \sum_{i=0}^{2^{n-1}} r_i \cdot m_i(x_1, x_2, ..., x_n) \tag{3.5}$$

kde $m_i(x_1, x_2, ..., x_n) = \widetilde{x_1}\widetilde{x_2}...\widetilde{x_n}$ je tzv. *minterm* n-premenných a $\widetilde{x_j} \in \{x_j, \overline{x_j}\}, j = 1, 2, ..., n$ a r_i je boolovská premenná, ktorá nadobúda hodnotu 1 práve na takej n-ici hodnôt 0 a 1 $(a_1, a_2, ..., a_n)$, na ktorej $m_i(a_1, a_2, ..., a_n) = 1$,t.j.

$$r_i = 1 \Leftrightarrow m_i(a_1, a_2, ..., a_n) = 1 \wedge [i]_2 = a_1 a_2 ... a_n \tag{3.6}$$

Príklad mintermov dvoch premenných je uvedený v tab. 3.1.Všimnime si vzťah medzi indexami a algebraickým tvarom mintermov; spočíva v tom , že ak si vyjadríme index i v binárnom tvare $[i]_2 = b_1 b_2$, tak potom ako dosadíme za hodnoty boolovských premenných v $\widetilde{x_1}$ a $\widetilde{x_2}$ zodpovedajúco b_1, resp. b_2 nadobudne minterm hodnotu 1.
Majme minterm $m_1(x_1, x_2) = \overline{x_1}x_2$; binárna hodnota indexu *1*-$[1]_2 = 01$. Pri dosadení do termu $\overline{x_1}x_2$ hodnoty $x_1 = 0$ $(\overline{x_1} = 1)$ a $x_2 = 1$ dostaneme podľa pravidiel boolovej algebry hodnotu 1.
Duálnym tvarom k udnf je ucnf.

$$f(x_1, x_2, ..., x_n) = \prod_{i=0}^{2^{n-1}} (r_i^{'} + M_i(x_1, x_2, ..., x_n)) \tag{3.7}$$

kde $M_i(x_1, x_2, ..., x_n) = \widetilde{x_1} + \widetilde{x_2} + ... + \widetilde{x_n}$ je tzv. *maxterm* n-premenných a $\widetilde{x_j} \in \{x_j, \overline{x_j}\}, j = 1, 2, ..., n$ a a $r_i^{'}$ je boolovská premenná, ktorá nadobúda hodnotu 0 práve na takej n-ici hodnôt 0 a 1 $(a_1, a_2, ..., a_n)$, na ktorej $M_i(a_1, a_2, ..., a_n) = 0$,t.j.

$$r_i^{'} = 0 \Leftrightarrow M_i(a_1, a_2, ..., a_n) = 0 \wedge [i]_2 = a_1 a_2 ... a_n \tag{3.8}$$

Príklad maxtermov dvoch premenných je uvedený v tab. 3.1. Charakteristika povahy maxtermov zodpovedá princípu duálnosti; maxterm $M_i(x_1, x_2, ..., x_n) = \widetilde{x_1} + \widetilde{x_2} + ... + \widetilde{x_n}$ nadobudne hodnotu 0 jedine v prípade ak za výskyty premenných v ňom dosadíme zodpovedajúce hodnoty v binárnom vyjadrení indexu *i*. Tak maxterm $M_1(x_1, x_2) = x_1 + \overline{x_2}$ nadobudne hodnotu 0 práve vtedy ak $x_1 = 0$ a $x_2 = 1(\overline{x_2} = 0)$, pritom $[1]_2 = 01$

Tabuľka 3.1

Tabulka:Mintermy a maxtermy 2-premenných .

i	$m_i(x_1, x_2)$	$\widetilde{x_1}\widetilde{x_2}$	$M_i(x_1, x_2)$	$\widetilde{x_1} + \widetilde{x_2}$
0	$m_0(x_1, x_2)$	$\overline{x_1}\,\overline{x_2}$	$M_0(x_1, x_2)$	$x_1 + x_2$
1	$m_1(x_1, x_2)$	$\overline{x_1}x_2$	$M_1(x_1, x_2)$	$x_1 + \overline{x_2}$
2	$m_2(x_1, x_2)$	$x_1\overline{x_2}$	$M_2(x_1, x_2)$	$\overline{x_1} + x_2$
3	$m_3(x_1, x_2)$	$x_1 x_2$	$M_3(x_1, x_2)$	$\overline{x_1} + \overline{x_2}$

Pre každú b.f. $f(x_1, x_2, ..., x_n)$ jej undf a tvar mintermov $m_i(x_1, x_2, ..., x_n)$ (jej uncf a tvar maxtermov $M_i(x_1, x_2, ..., x_n)$)dokazuje platnosť tvrdenia vety.

□

V ďalšom budeme pre undf b.f. $f(x_1, x_2, ..., x_n)$ (3.5) používať označenie D_f a teda

$$D_f = f(x_1, x_2, ..., x_n) = \sum_{i=0}^{2^{n-1}} r_i \cdot m_i(x_1, x_2, ..., x_n) \tag{3.9}$$

Majme D_f pre b.f. $\overline{f \equiv 0}$; ak zameníme v (3.10) symboly + za symboly operácie $\oplus$-*súcet modulo 2* dostaneme novú reprezentáciu *f*, ktorá sa nazýva *úplnou bisumárnou normálnou formou* (ubnf) b.f. f a budeme ju označovať ako B_f.

$$B_f = f(x_1, x_2, ..., x_n) = \oplus_{i=0}^{2^{n-1}} r_i \cdot m_i(x_1, x_2, ..., x_n) \tag{3.10}$$

Nie je tažké vidieť, že zámena symbolov + za symboly operácie $\oplus$ nemení hodnotu reprezentovanej funkcie. To plynie z povahy mintermov (nadobúdajú hodnotu 1 práve na jednej n-ici boolovských hodnôt $(a_1, a_2, ..., a_n)$) a povahy operácií + a $\oplus$ (sú totožné na všetkých 2-iciach argumentov okrem (1,1)). Potom platí tento

Dôsledok 3.1

Každá b.f. $f(x_1, x_2, ..., x_n)$ je jednoznacne reprezentovatelná svojou ubnf.

□

Dôsledok 3.2

Systém operácií pozostávajúci z konjunkcie ($\wedge$), súctu modulo 2 ($\oplus$) a negácie je úplný.

□

Princíp duality umožnuje sformulovať tento

Dôsledok 3.3

Systém operácií pozostávajúci z dizjunkcie ($\vee$), ekvivalencie ($\equiv$) a negácie je úplný.

□

Ako príklad uvedieme pre b.f. f_2 z tab. 3.1 jednotlivé formy jej reprezentácie:

$$\begin{aligned} f_2(x_1,x_2) &= \overline{x_1}\cdot x_2 + x_1\cdot\overline{x_2} + x_1\cdot x_2 \qquad (udnf)\\ f_2(x_1,x_2) &= x_1 + x_2 \qquad (ucnf)\\ f_2(x_1,x_2) &= \overline{x_1}\cdot x_2 \oplus x_1\cdot\overline{x_2} \oplus x_1\cdot x_2 \qquad (ubnf) \end{aligned}$$

Úplné systémy operácií v AL možno považovať za signatúry operácií definovaných na množine b.f. BF. Prostredníctvom superpozície týchto operácií môže byť zostrojená ľubovoľná b.f. z BF.
Množina BF spolu s vybratým úplným systémom b.f. sa volá *algebrou boolovských funkciií ABF*. Najznámejšími algebrami b.f. sú:

- algebra Boola (AB) so signatúrou (úplným systémom) $\wedge, \vee, \neg$;
- algebra Žegalkina (AŽ) so signatúrou (úplným systémom) $\wedge, \oplus,$ konštanta1

Algebra Boola vyhovuje všetkým zákonom Boolovej algebry, ktoré sme uviedli v kapitole 1.3. Tieto zákony tvoria základ teórie úplných foriem reprezentácie b.f., na ich základe bola dokázaná algoritmická rozhodnutelnost problému ekvivalencie b.f. a vyriešený problém minimalizácie b.f. (nájdenie najjednoduchšej formy reprezentácie b.f.).
Algebra Žegalkina AŽ=(BF;{$\wedge, \oplus,$ konštanta1}) - presnejšie operácie jej signatúry vyhovujú nasledujúcim zákonom:

zákon komutatívnosti	$(x \bullet y) = (y \bullet x)$
zákon asociatívnosti	$(x \bullet y) \bullet z = x \bullet (y \bullet z) = x \bullet y \bullet z$
	kde symbol $\bullet$ je symbolom operácie $\wedge$, alebo $\oplus$
zákon distributívnosti	$(x \oplus y) \wedge z = (x \wedge z) \oplus (y \wedge z)$
zákon idempotentnosti konjunkcie	$x \wedge x = x$
zákon (pravidlo) krátenia	$x \oplus x = 0$

Vlastnosti konštánt:

$$x \wedge 1 = x;\ x \wedge 0 = 0;\ x \oplus 0 = x;\ x \oplus 1 = \overline{x} \tag{3.11}$$

Posledná z rovností (3.11)ukazuje, že negácia je odvodená operácia v AŽ.
Medzi AB a AŽ zaujímavú pozíciu zaujíma algebra so signatúrou: konjunkcia, suma modulo 2 a negácia. Bol rozpracovaný aparát tzv. *bisumárnych normálnych foriem*b.f. [30, 35] a študovaná problematika

minimalizácie b.f. v rámci tejto algebry.
Majme b.f. $f(x_1, x_2, ..., x_n) \neq 0$ a nech f je reprezentovaná v ubnf.

$$B_f = f(x_1, x_2, ..., x_n) = \oplus_{i=0}^{2^{n-1}} r_i \cdot m_i(x_1, x_2, ..., x_n) \tag{3.12}$$

Zameníme teraz v každom minterme v (3.12) každý výskyt negovanej premennej $\overline{x}$ xa term $x \oplus 1$ podľa poslednej rovnosti z ((3.11)). Po otvorení zátvoriek a komprimácie konjunkcií na základe zákona idempotentnosti konjunkcie a odstránenia premenných podľa zákona krátenia získame napokon polynomiálnu reprezentáciu b.f. f v tvare

$$f = \oplus_{k-1}^{m} U_k \tag{3.13}$$

kde U_k je konjunktívny term (bez negovaných premenných). Reprezentácia (3.13) nesie názov *polynóm Žegalkina*(PŽ).

Príklad 3.2

Zostrojíme teraz PŽ pre b.f. f_2 z príkladu 3.1; ubnf f_2 má tvar

$$B_{f_2} = \overline{x} \wedge y \oplus x \wedge \overline{y} \oplus x \wedge y \tag{3.14}$$

Nahradením negovaných výskytov podľa (3.11) dostaneme

$$B_{f_2} = (x \oplus 1) \wedge y \oplus x \wedge (y \oplus 1) \oplus x \wedge y \tag{3.15}$$

Uplatnením zákona distributívnosti máme

$$B_{f_2} = x \wedge y \oplus y \oplus x \wedge y \oplus x \oplus x \wedge y \tag{3.16}$$

Uplatnením zákona krátenia napokon dostávame

$$B_{f_2} = x \oplus y \oplus x \wedge y \tag{3.17}$$

Tvar pravej strany (3.17) je PŽ.

□
(Koniec príkladu)

Platí táto

Veta 3.2

Ľubovoľná b.f. $f(x_1, x_2, ..., x_n) \neq 0$ sa dá jednoznačným spôsobom (s presnosťou do usporiadania konjunkcií) vyjadriť ako polynóm Žegalkina. **Dôkaz:** Skutočne, predpokladajme , že pre b.f. $f(x_1, x_2, ..., x_n) \neq 0$ sa nájdu dva rôzne PZ, napríklad PZ_1 a PZ_2 a $PZ_1 \neq PZ_2$, ktoré reprezentujú *f*. Pozrime sa na množinu (konjunktívnych) termov, ktoré sa súčasne nevyskytujú v PZ_1 a PZ_2; taká množina je neprázdna, lebo $PZ_1 \neq PZ_2$.
Zvolíme si v tejto množine najkratšiu konjunkciu. Priradíme všetkým premenným tejto konjunkcie hodnotu 1 a všetkým ostatným premenným f hodnotu 0. Týme sme vytvorili n-icu $\widetilde{a}$ na ktorej $PZ_1(\widetilde{a}) \neq PZ_2(\widetilde{a})$, čo protirečí predpokladu o existencii dvoch rôznych polynómov PZ_1 a PZ_2 reprezentujúcich rovnakú funkciu.

□

Duálne tvrdenie môže buť sformulované aj pre algebru so signatúrou : dizjunkcia, ekvivalencia a konštanta 0.

V AL existuje viac algebier b.f. so signatúrou predstavujúcou úplný systém: algebra Schaefera (Schaeferova čiarka), algebra Piercea (Pierceova šípka). Dôkazy úplnosti systému operácií sú založené na technike vyjadrenia operácií známeho úplného systému prostredníctvom superpozícií testovaného na úplnosť systému . Také dôkazy budú námetom cvičení a samostatnej práce čitateľa.

3.2 Veta o funkcionálnej úplnosti.

Táto kapitola je venovaná známej vete Posta o funkcionálnej úplnosti, ktorá je základom kritéria úplnosti systému b.f. v algebre logiky. Získané výsledky môžu byť použité pri riešení problému úplnosti v celom rade dôležitých subalgebier metaalgebry (MA), pretože MA a najzaujímavejšie jej subalgebry obsahujú AL ako svoju súčasť (jednu z osnov). Medzi také subalgebry patrí AD (viď 1.3).
Riešenie problému úplnosti pre AL je spojené s výskumom tried b.f. ktoré majú určité dôležité vlastnosti. V ďalšom sa budeme venovať štúdiu takých vlastností a s nimi spriahnutých tried b.f., ktoré sú základom pri riešení problému funkcionálnej úplnosti v AL.
Teraz prejdeme k definícii spomínaných vlastností b.f.:

1. *Zachovanie konštanty 0.* Funkcia $f(x_1, x_2, ..., x_n)$ zachováva konštantu 0, ak platí, že $f(0, 0, ..., 0) = 0$; inými slovami funkcia f zachováva konštantu 0, ak jej hodnota je 0 pri dosadení za každú jej premennú $x_i = 0$ pre $i = 1, 2, ..., n$. Triedu b.f.- s touto vlastnosťou budeme označovať symbolom T_0.

2. *Zachovanie konštanty 1.* Funkcia $f(x_1, x_2, ..., x_n)$ zachováva konštantu 1, ak platí, že $f(1, 1, ..., 1) = 1$; inými slovami funkcia f zachováva konštantu 1, ak jej hodnota je 1 pri dosadení za každú jej premennú $x_i = 1$ pre $i = 1, 2, ..., n$. Triedu b.f.- s touto vlastnosťou budeme označovať symbolom T_1. Táto definícia je duálnou k T_0.

3. *Samodualita.* Funkcia $f(x_1, x_2, ..., x_n)$ je *samoduálnou* b.f. ak platí, že $f(x_1, x_2, ..., x_n) = \overline{f(\overline{x_1}, \overline{x_2}, ..., \overline{x_n})}$; inými slovami funkcia f je samoduálnou, , ak jej hodnota na protikladných (inverzných) n-iciach hodnôt je protikladná. Triedu b.f.- s touto vlastnosťou budeme označovať symbolom S.

4. *Monotonosť.* Majme boolovskú funkciu $f(x_1, x_2, ..., x_n)$; o dvoch n-iciach hodnôt boolovských premenných $(x_1, x_2, ..., x_n)$ $\widetilde{a} = (a_1, a_2, ..., a_n)$ a $\widetilde{b} = (b_1, b_2, ..., b_n)$ budeme hovoriť, že sú porovnateľné (symbolicky to označujeme symbolom $\widetilde{a} \leq \widetilde{b}$) ak platí, že $a_i \leq b_i, i = 1, 2, ..., n)$, ináč budeme považovať $\widetilde{a}$ a $\widetilde{b}$ za *neporovnateľné*.Boolovskú funkciu $f(x_1, x_2, ..., x_n)$ voláme *monotonnou* b.f. ak platí, že ak máme dve n-ice hodnôt jej premenných $\widetilde{a} \leq \widetilde{b}$, potom $f(\widetilde{a}) \leq f(\widetilde{b})$. ; inými slovami funkcia f je monotonná, ak pri narastaní n-íc jej argumentov hodnota f *neklesá*. Triedu monotonných b.f. budeme označovať symbolom M.

5. *Lineárnosť.* B.f. $f(x_1, x_2, ..., x_n)$ je *lineárna b.f.* , ak sa dá vyjadriť v tvare $f(x_1, x_2, ..., x_n) = c_0 \oplus c_1 \cdot x_1 \oplus c_2 \cdot x_2 \oplus, ..., c_n \cdot x_n$, kde c_i je konštanta 0, alebo 1 pre každé i=0,1,2,...,n . Inými slovami boolovská funkcia $f(x_1, x_2, ..., x_n)$ je lineárna ak sa dá vyjadriť polynómom Žegalkina bez konjunktívnych (nelineárnych) zložiek tvaru $x_1 \wedge x_2 \wedge ... \wedge x_k$ pre akékoľvek $k > 1$. Triedu monotonných b.f. budeme označovať symbolom L.

Uvedieme príklady na jednotlivé triedy definovaných b.f.

Príklad 3.3

1. *Zachovanie konštanty 0.* Z tab. **??** 2-argumentových b.f. sú tieto b.f. zachovávajúce 0:konjunkcia-$(0 \wedge 0 = 0)$; dizjunkcia-$(0 \vee 0 = 0)$; suma modulo 2-$(0 \oplus 0 = 0)$; antiimplikácia - $(0 \not\Rightarrow 0 = 0)$; opačná antiimplikácia - $(0 \not\Leftarrow 0 = 0)$;

2. *Zachovanie konštanty 1.* Z tab. **??** 2-argumentových b.f. sú tieto b.f. zachovávajúce 1:konjunkcia-$(1 \wedge 1 = 1)$; dizjunkcia-$(1 \vee 1 = 1)$; ekvivalencia-$(1 \equiv 1 = 1)$; implikácia - $(1 \Rightarrow 1 = 1)$; opačná implikácia - $(11 \Leftarrow 1 = 1)$;

3. *Samodualita.* Medzi samoduálne funkcie patrí *negácia*-$(x = \overline{\overline{x}})$; 3-argumentová b.f. $f(x, y, z) = x \wedge y \vee y \wedge z \vee x \wedge z$. Skutočne, presvedčíme sa o tom, keď vyhodnotíme výraz

$$\begin{aligned}
\overline{\overline{x} \wedge \overline{y} \vee \overline{y} \wedge \overline{z} \vee \overline{x} \wedge \overline{z}} &= \\
= (x \vee y) \wedge (y \vee z) \wedge (x \vee z) &= \\
& \text{/* použitím pravidla de Morgana*/} \\
= x \wedge y \vee y \wedge z \vee x \wedge z & \\
& \text{/* odstránením zátvoriek, použitím zákona idempotencie} \\
& \text{/*a použitím pravidla pohltenia*/}
\end{aligned}$$

4. *Monotonosť.* Z tab. **??** 2-argumentových b.f. sú tieto b.f. monotonné:konjunkcia-(f_1); dizjunkcia-(f_2); monotonné sú aj konštanty 0 a 1.

5. *Lineárnosť.* Medzi elementárnymi 2-argumentovými b.f. (viď tab. 3.1) sú lineárnymi b.f. f_5 (*suma mod 2*) a b.f. f_6 (*ekvivalencia*). Skutočne:

$$\begin{aligned}
& \text{Lineárnosť b.f. } f_5 \text{ (\textit{suma mod 2})} \\
f_5(x,y) &= \overline{x}y \oplus x\overline{y} \\
& \text{/*použitie rovnosti } \overline{A} = A \oplus 1 \text{ dáva, že*/} \\
f_5(x,y) &= (x \oplus 1)y \oplus x(y \oplus 1) \\
& \text{/*použitie zákona distributívnosti dáva, že */} \\
f_5(x,y) &= xy \oplus y \oplus xy \oplus x \\
& \text{/*použitie zákona krátenia } A \oplus A = 0 \text{ dáva, že */} \\
f_5(x,y) &= 0 \oplus x \oplus y \\
& \\
& \text{Lineárnosť b.f. } f_6 \text{ (\textit{ekvivalencia})} \\
f_6(x,y) &= \overline{xy} \oplus xy \\
& \text{/*použitie rovnosti } \overline{A} = A \oplus 1 \text{ dáva, že*/} \\
f_6(x,y) &= (x \oplus 1)(y \oplus 1) \oplus xy \\
& \text{/*použitie zákona distributívnosti dáva, že */} \\
f_6(x,y) &= xy \oplus x \oplus (y \oplus 1) \oplus xy \\
& \text{/*použitie zákona krátenia dáva, že*/} \\
f_6(x,y) &= 1 \oplus x \oplus y
\end{aligned}$$

Všetky ostatné 2-argumentové funkcie z tab. 3.1 nie sú lineárne b.f..

□
(Koniec príkladu)

Dve n-ice hodnôt boolovských premenných $x_1, x_2, ..., x_n$ $\widehat{a} = (a_1, a_2, ..., a_n)$ a $\widehat{b} = (b_1, b_2, ..., b_n)$ a také, že $\widehat{a} \leq \widehat{b}$, sa volajú *susedné* ak sa nájde jediná premenná x_i taká, že $a_i = 0$ a $b_i = 1$. Inými slovami na susedných n-iciach hodnoty všetkých premenných s výnimkou x_i sú rovnaké. Príkladmi susedných n-íc sú:(0,0) a (0,1);(0,1,0) a (1,1,0);(0,0,0) a (0,1,0).
O vlastnostiach monotonných b.f. hovorí táto

Lemma 3.1 *Ak b.f.* $f(x_1, x_2, ..., x_n)$ *nie je monotonná, potom sa nájdu také 2 susedné n-ice* $\widehat{a}$ *a* $\widehat{a}^{'}$ *také, že* $\widehat{a} \leq \widehat{a}^{'}$ *a* $f(\widehat{a}) > f(\widehat{a}^{'})$

Dôkaz: Pre nemonotonnú b.f. $f(x_1, x_2, ..., x_n)$ sa nájdu vždy také 2 n-ice $\widehat{c}$ a $\widehat{b}$, že $\widehat{c} \leq \widehat{b}$ a $f(\widehat{c}) > f(\widehat{b})$. Zafixujeme si tie premenné $x_{i_1}, x_{i_2}, ..., x_{i_k}$ na ktorých $c_{i_j} \neq b_{i_j}$; pretože $\widehat{c} \leq \widehat{b}$ máme, že $c_{i_j} = 0, b_{i_j=1,j=1,2,...,k}$. Vytvoríme teraz n-icu $\widehat{c}^{'}$ tak, že zameníme $c_{i_k} = 0$ na 1. Ak teraz $f(\widehat{c}^{'}) = 0$ našli sme n-icu, ktorá vyhovuje tvrdeniu lemy; ak nie, vytvoríme k $\widehat{c}^{'}$ n-icu $\widehat{c}^{''}$ tak, že zameníme (v $\widehat{c}^{'}$) $c_{i_{k-1}} = 0$ na 1 . Ak $f(\widehat{c}^{''}) = 0$ našli sme n-icu, ktorá vyhovuje tvrdeniu lemy; ak nie opakujeme tento postup až kým také dve susedné n-ice nenájdeme. Rezultatívnosť takého postupu je zaručená konečnosťou množiny premenných v ktorej taký postup uplatňujeme a existenciou takej dvojice susedných n-íc. Posledné súvisí s existenciou n-ice $\widehat{b}$, na ktorej $f(\widehat{b}) = 0$ a v povahe výberu susedných n-íc. V najhoršom prípade, ak v postupnosti vytváraných n-íc $\left\{\widehat{c}^{'}, \widehat{c}^{''}, ..., \widehat{c}^{j}\right\}, j = 1, 2, ..., k-1$ nebola nájdená žiadna vyhovujúca dvojica

n-íc (t.j. $f(\widehat{c}^j) = 1$ pre každé j=1,2,...,k-1), napokon dosiahneme dvojicu $(\widehat{c}^{k-1}, \widehat{c}^k)$, ktorá vyhovuje podmienkam lemy. Skutočne, stačí si všimnúť, že $\widehat{c}^k = \widehat{b}$ a $f(\widehat{c}^{k-1}) = 1$ a $f(\widehat{c}^k) = f(\widehat{b}) = 0$ podľa predpokladu lemy.

□

Rodina $\boldsymbol{K}$ b.f. vytvára *uzavretú triedu* (lebo hovoríme o *subalgebre AL*), ak pri ľubovoľných superpozíciach b.f. z triedy $\boldsymbol{K}$ a tiež premenovaniami a stotožneniami ich premenných vzniknú iba b.f. z $\boldsymbol{K}$ a žiadne iné. Príkladom uzavretej triedy môže slúžiť trieda b.f. závislá nie od viac ako jedného argumenta ($n \leq 1$). Takú triedu predstavuje spoločenstvo b.f. $K_1 = \{0, 1, \{x_i, \overline{x_i} | i = 1, 2, ..., n\}\}$. Nie je ťažko vidieť, že v tejto triede pri ľubovoľných superpozíciach b.f. z triedy K_1 a tiež premenovaniami a stotožneniami ich premenných vzniknú iba b.f. závislé nie od viac ako jedného argumenta ($n \leq 1$) a žiadne iné.
O doteraz definovaných triedach b.f. T_0, T_1, S, M, L sa dá dokázať toto veľmi vážne tvrdenie, ktoré demonštruje hĺbku zámeru, ktorý sa skrýva vo výbere a definícii týchto tried b.f..

Lemma 3.2 *Triedy T_0, T_1, S, M, L sú uzavreté triedy b.f.*

Dôkaz: Majme b.f. $f(x_1, x_2, ..., x_n)$, $g_1(x_{11}, x_{12}, ..., x_{1n_1})$, $g_2(x_{21}, x_{22}, ..., x_{2n_2})$,..., $g_n(x_{n1}, x_{n2}, ..., x_{nn_n})$, ako aj ich superpozíciu $h(x_{11}, x_{12}, ..., x_{1n_1}, x_{21}, x_{22}, ..., x_{2n_2}, ..., x_{n1}, x_{n2}, ..., x_{nn_n}) = f(g_1(x_{11}, x_{12}, ..., x_{1n_1}), g_2(x_{21}, x_{22}, ..., x_{2n_2}), ...$
Ukážeme, že ak funkcie $f, g_1, g_2, ..., g_n$ patria do jednej z uvedených tried, potom aj superpozícia h patrí do rovnakej triedy.
Poukazujeme na špeciálny prípad, keď funkcie g_i sú jednoducho premenné x_i; premenné x_i patria do každej z tried T_0, T_1, S, M, L. Skutočne: každá x_i zachováva 0, zachováva 1, je samoduálna ($x_i = \overline{\overline{x_i}}$), je monotonná a je aj lineárna. To znamená, že prostredníctvom superpozície funkcie f s rôznymi premennými možno realizovať ich stotožňovanie a premenovávanie.
Predpokladjme, že funkcie $f, g_1, g_2, ..., g_n$ zachovávajú 0; potom $f(0, 0, ..., 0) = 0$, $g_1(0, 0, ..., 0) = 0$, $g_2(0, 0, ..., 0) = 0$, , ..., $g_n(0, 0, ..., 0) = 0$. Z toho plynie, že aj superpozícia $h(0, 0, ..., 0) = f(g_1(0, 0, ..., 0), g_2(0, 0, ..., 0), ..., g_n(0, 0, ..., 0)) = 0$.
Duálnym spôsobom možno preukázať platnosť tvrdenia o zachovaní 1 superpozíciou h za predpokladu, že funkcie $f, g_1, g_2, ..., g_n$ zachovávajúí 1.
Ukážeme teraz, že superpozícia h je samoduálna, za predpokladu, že funkcie $f, g_1, g_2, ..., g_n$ sú samoduálne. Skutočne, pre ľubovoľnú n-icu $a_i = (a_{i1}, a_{i2}, ..., a_{in_i})$ nech platí, že $g_i(a_{i1}, a_{i2}, ..., a_{in_i}) = g_i(\overline{a_{i1}}, \overline{a_{i2}}, ..., \overline{a_{in_i}})$. To znamená, že ak si označíme $c_i = g_i(a_{i1}, a_{i2}, ..., a_{in_i})$ potom $\overline{c_i} = g_i(\overline{a_{i1}}, \overline{a_{i2}}, ..., \overline{a_{in_i}})$, potom platí (zo samoduálnosti f), že $f(c_1, c_2, ..., c_n) \neq f(\overline{c_1}, \overline{c_2}, ..., \overline{c_n})$ a superpozícia $h = f(c_1, c_2, ..., c_n)$ je samoduálna.
Analogicky sa dá preukázať aj monotonnosť a linearita superpozície h ak sú monotonné, resp. lineárne f a $g_i, i = 1, 2, ..., n$.

□

Teraz prejdeme k formulácii a dôkazu fundamentálneho výsledku- vety o funkcionálnej úplnosti.

Veta 3.3 *Majme SYST- systém b.f.; SYST je úplným systémom b.f. vtedy a len vtedy ak pre každú triedu T_0, T_1, S, M, L platí, že v SYST sa nachádza aspon jedna b.f., ktorá nepatrí do danej triedy.*

Dôkaz: ($\Leftarrow$)*Nevyhnutnosť* Nevyhnutná podmienka vyplýva z toho, že ak všetky funkcie v SYST patria aspoň do jednej z uvedených tried, tak potom systém b.f. SYST je neúplný v dôsledku lemy 3.2 o uzavretosti tried T_0, T_1, S, M, L.
($\Rightarrow$)*Dostatočnosť.* Podľa predpokladu SYST obsahuje b.f. $f_1 \notin T_0, f_2 \notin T_1, f_3 \notin S, f_4 \notin M, f_5 \notin L$, ktoré nemusia byť nutne rôzne. Plán je nasledovný:

a) Najprv vytvoríme pomocou b.f. f_1 až f_4 konštanty 0,1 a negáciu;

b) potom pomocou b.f. f_5 vytvoríme konjunkciu a týme preukážeme, že systém b.f.SYST je redukovteľný k úplnému systému b.f. a tým je SYST úplný systém.

Vezmime najprv b.f. $f_1 \notin T_0$, čo znamená, že $f_1(0,0,...,0) = 1$; označme si $t = f_1(1,1,...,1)$. Môžu nastať 2 prípady:

1. t=0.V takom prípade stotožnením premenných v f_1 dostávame, že

$$f_1(x,x,...,x) = \overline{x} \tag{3.18}$$

Ďalej si zvolíme b.f. $f_3(x_1,x_2,...,x_k) \notin S$; to znamená, že sa nájdu dve protikladné k-ice $\widetilde{a} = (a_1,a_2,...,a_k)$ a $\widetilde{a}' = (\overline{a_1},\overline{a_2},...,\overline{a_k})$ na ktorých funkcia f_3 nadobúda rovnakú hodnotu. Rozbijeme teraz premenné funkcie f_3 na dve skupiny:

- -prvá: $x_i \in skp1 \Leftrightarrow a_i = 0$;
- -druhá: $x_i \in skp2 \Leftrightarrow a_i = 1$.

Všetky premenné zo $skp1$ stotožníme s $\overline{x}$ a tie zo $skp2$ stotožníme s x; dostaneme funkciu $f_3(x)$ a takú, že $f_3(0) = f_3(1) = const$. Dosadením získnej konštanty $const$ do funkcie $f_1(x,x,...,x)$ v (3.18) dostaneme druhú konštantu $\overline{const}$.

2. t=1. Po stotožnení premenných v f_1 dostaneme, že $f_1(x,x,...,x) = 1$ a teda získavame konštantu 1. Zvolíme ľubovoľnú funkciu $f_2 \notin T_1$ (t.j. $f_2(1,1,...,1) = 0$), a dosadíme do nej namiesto 1 $f_1(x,x,...,x) = 1$, čím dostaneme konštantu 0 podľa formuly

$$f_2(f_1(x,x,...,x), f_1(x,x,...,x), ..., f_1(x,x,...,x)) = 0 \tag{3.19}$$

K vytvoreniu negácie v danom prípade si zvolíme niektorú b.f. $f_4(x_1,x_2,...,x_k) \notin M$. Podľa lemy 3.1 pre nemonotonnú funkciu f_4 sa vždy nájdu 2 susedné k-ice $\widetilde{a} = (a_1,a_2,...,a_k)$ a $\widetilde{a}' = (a_1',a_2',...,a_k')$, ktoré sa líšia iba v jednej položke (povedzme) $a_i = 0$ a $a_i' = 1$ (t.j.$a_i < a_i'$), na ktorých $f_4(\widetilde{a}) = 1$ a $f_4(\widetilde{a}') = 0$. Dosadíme v f_4 namiesto premennej x_i premennú x ; za všetky premenné x_ℓ, $\ell \neq i$ pre ktoré $a_\ell = 0$ dosadíme *konštantu 0* a za ostatné premenné *konštantu 1.* Dostaneme b.f. $f_4(x) = \overline{x}$.

Preukázali sme, že v oboch prípadoch sa dajú zostrojiť konštanty 0 a 1 a negácia.
Prejedeme teraz k zostrojeniu *konjunkcie.* Za tým účelom si vyberieme nelineárnu b.f. $f_5(x_1,x_2,...,x_k) \notin L$, ktorá je reprezentovaná polynómom Žegalkina PZ, t.j. $f_5 = PZ$. Zvolíme si pevne premenné x a y podľa ktorých je f_5 nelineárna. Po vhodnom premenovaní premenných dostaneme:

$$f_5(x,y,z_1,z_2,...,z_{r-2}) = x \wedge y \wedge q_1(z_1,z_2,...,z_{r-2}) \oplus x \wedge q_2(z_1,z_2,...,z_{r-2}) \oplus y \wedge q_3(z_1,z_2,...,z_{r-2}) \oplus q_4(z_1,z_2,...,z_{r-2}) \tag{3.2}$$

Podľa predpokladu o nelinearite f_5 podľa x a y je $q_1(z_1,z_2,...,z_{r-2}) \neq 0$, čo znamená, že sa nájde r-2-ica $\widetilde{t} = (t_1,t_2,...,t_{r-2})$ na ktorej $q_1(\widetilde{t}) = 1$. Vzhľadom k tomu, že disponujeme konštantami 0 a 1, dosadíme v (3.20) do f_5 za každú premennú z_i hodnotu t_i, čím získame toto vyjadrenie b.f. f_5 :

$$f_5(x,y,z_1,z_2,...,z_{r-2}) = x \wedge y \oplus x \wedge c_1 \oplus y \wedge c_2 \oplus c_3 \tag{3.21}$$

kde $c_i = q_i(t_1,t_2,...,t_{r-2})$, $i = 1,2,3$. Vzhľadom na jestvujúcu konštrukciu negácie položíme $c_3 = 0$.
Konštrukcia konjunkcie bude závisieť od kombinácie hodnôt c_1 a c_2:

1. $c_1 \neq c_2$. Konjunkciu z $f_5(x,y)$ vytvoríme dosadením do $f_5(x,y)$ negáciu jednej z jej premenných. Pri $c_1 = 1$ a $c_2 = 0$ $f_5(x,y,z_1,z_2,...,z_{r-2}) = x \wedge y \oplus x$. Ak dosadíme namiesto y jej negáciu dostaneme

$$\begin{aligned} f_5(x,\overline{y}) &= x \wedge \overline{y} \oplus x \\ &= x \wedge (y \oplus 1) \oplus x \\ &= x \wedge y \oplus x \oplus x \\ &= x \wedge y \end{aligned} \tag{3.22}$$

2. $c_1 = c_2$.Prípad $c_1 = c_2 = 0$ dáva triviálne $f_5(x, y) = x \wedge y$; v prípade, $c_1 = c_2 = 1$ ak dosadíme v $f_5(x, y)$ namiesto x a y ich negácie dostaneme

$$\begin{aligned} f_5(\overline{x}, \overline{y}) &= \overline{x} \wedge \overline{y} \oplus \overline{x} \oplus \overline{y} \qquad (3.23) \\ &= (x \oplus 1) \wedge (y \oplus 1) \oplus (x \oplus 1) \oplus (y \oplus 1) \\ &= ((x \oplus 1) \wedge y) \oplus (x \oplus 1) \oplus (x \oplus 1) \oplus (y \oplus 1) \\ &= ((x \oplus 1) \wedge y) \oplus (y \oplus 1) \\ &= (x \wedge y) \oplus y \oplus (y \oplus 1) \\ &= (x \wedge y) \oplus 1 \quad \text{a definitívne} \\ f_5(\overline{x}, \overline{y}) &= (x \wedge y) \oplus 1 \qquad (3.24) \end{aligned}$$

Pravá strana (3.24) je negáciou $x \wedge y$ a teda $\overline{f_5(\overline{x}, \overline{y})} = x \wedge y$.
Záverom môžeme konštatovať, že SYST je úplný systém b.f.

□

Plán dôkazu, ktorý sme uviedli je možno vyjadriť aj takto:

Dôkaz ::= $VBER(f_1) * ([f_1(\widetilde{1} = 0)]VYTVORENIE(\overline{x}) * VBER(f_3) * VYTVORENIE(0,1), VBER(f_2) * VYTVORENIE($

Uzavretú triedu $\boldsymbol{K}$ voláme *pre-úplnou*(alebo *maximálnou subalgebrou*) v algebre logiky (AL), ak pripojením akejkoľvek b.f. nepatriacej do $\boldsymbol{K}'$ privádza k úplnému systému b.f.

Dôsledok 3.4 *Triedy T_0, T_1, S, M, L sú pre-úplné (sú maximálne subalgebry.)*

Dôkaz: Skutočne, nech $\boldsymbol{K}'$ je niektorá z uvedených tried; potom v $\boldsymbol{K}'$ sa nájde aspon jedna b.f. $f \notin \boldsymbol{K}''$ pre každú z uvedených tried $\boldsymbol{K}'' \neq \boldsymbol{K}'$. *Zafixujeme si takto vybraté funkcie a pripojíme k nim b.f. $g \notin \boldsymbol{K}'$. Z vety 3.3 máme, že vytvorený systém b.f. je úplný.*

□

Platí aj tento

Dôsledok 3.5 *(zoslabené kritérium úplnosti) Systém SYST pozostávajúci z konštánt 0 ,1 je úplný práve vtedy ak SYST obsahuje aspon jednu nemonotonnú a jednu lineárnu b.f.*

Dôkaz: *Skutočne,* $0 \notin T_1$ a $1 \notin T_0$; a každá z konštánt nie je samoduálna. Potom SYST vyhovuje kritériu úplnosti založenému na vete 3.3.

□

Poznamenávame, že veta 3.3 dáva predstavu o hornrj úrovni hierarchie subalgebier AL, ktorá je tvorená spoločenstvom jej maximálnych subalgebier. Výskum štruktúry subalgebier je spriahnuté s riešením problému funkcionálnej úplnosti pre každú z týchto maximálnych subalgebier- pre-úplných tried:T_0, T_1, S, M, L. Je na mieste pripomenúť, že problematika tvorby hierarchie uzavretých tried bola vyriešená významným americkým matematikom E. Postom [10, 11].
V [10] je dokázaná táto

Veta 3.4 *Majme AL-2-značnú algebru logiky. potom :*

i) každá uzavretá trieda - subalgebra AL, má konečný systém generátorov a teda aj konečnú bázu;

ii) množina všetkých uzavretých tried AL je spočítateľná, jestvuje konečne veľa typov uzavretých tried v AL;

iii) je vytvorená hierarchia podradenosti tried AL, reprezentovaná grafom, v ktorom trieda $\boldsymbol{K}$*, zodpovedajúca niektorému vrcholu v hierarchii, je maximálnou subalgebrou vo vzťahu k triedam* $\boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_s$*, ktoré sa nachádzajú na predchádzajúcej (vyššej) úrovni hierarchie, od ktorých vedú hrany k vrcholu* $\boldsymbol{K}$*.*

Pre k-značné (k-hodnotové) algebry logiky (pri k¿3) bola dokázaná existencia uzavretých tried s nekonečnou (spočítateľnou) bázou [31, 35]. Y existencie podobných uzavretých tried plynie kontinuálnosť množiny subalgebier pre k-značné AL.
Je zaujímavé si všimnúť, že kontinuálnosť množiny subalgebier je aj vlastnosťou modifikovanej AL s obmedzenou superpozíciou b.f. [33, 31] typu:

$$h(x_1, x_2, ..., x_{m+n-1}) = F(G(x_1, x_2, ..., x_n), G(x_2, x_3, ..., x_{n+1}), ..., G(x_m, x_{m+1}, ..., x_{n+m-1}))$$

Algebry , ktoré majú kontinuum subalgebier sa volajú *algebry kontinuálneho typu*[6]. K takým algebrám patria medzi inými metaalgebra (viď ďalej) a celý rad jej subalgebier, ktoré sú spojené so známymi metódami konštruovania algoritmov.

Kapitola 4

Algebra algoritmiky a jej aplikácie

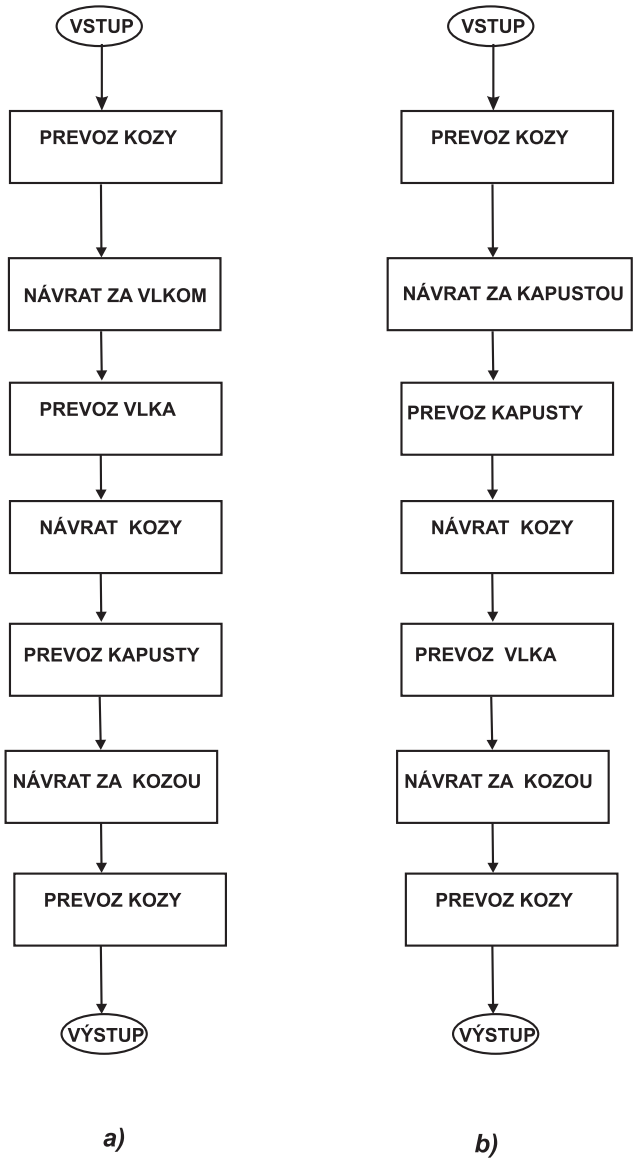
Základ každej algebry (a platí to aj o algebrách algoritmov z kapitoly 2) tvorí *superpozícia* operácií vytvárajúcich signatúru. Algebry algoritmov sú 2-druhové algebry v ktorých množina b.f. vystupuje ako jedna z osnov.
V tejto kapitole bude zostrojená algebra algoritmiky, ktorá zahŕňa ako svoju významnú zložku-algebru logiky. Budú preskúmanú vlastnosti štruktúry subalgebier algebry algoritmiky (AA). Zvláštna pozornosť je venovaná problému funkcionálnej úplnosti jej jednotlivých subalgebier, spojených s algebrami algoritmov z kapitoly 2). Tvorba algebier algoritmov v rámci zodpovedajúcich subalgebier AA sa realizuje analogicky s tvorbou algebier b.f. v rámci AL (viď kapitolu 3).

4.1 Algebra zovšeobecnených graf-schém.

?? Zovšeobecnená graf-schéma (zgs) sa získa úpravou definície g-s (viď Def. 2.3).Zovšeobecnená graf-schéma (zgs) $\widetilde{G}$ je označený graf ako v Def. 2.3 s tým rozdielom, že ak $\widetilde{G} = (V, \vec{E}, \ell)$, kde V-množina vrcholov, $\vec{E}$ -množina orientovaných hrán, potom každý vrchol zgs $\widetilde{G}$ (okrem vstupného a výstupného) je označený dvojicou *(A,u)*, kde A je operátorová premenná a u je predikátová premenná a okrem toho má dve vystupujúce hrany: plusovú (+) a mínusovú (-).
Ak pri interpretácii **I** zgs $\widetilde{G}$ premenná u(vo vrchole v so značkou *(A,u)*,t.j $\ell(v) = (A, u)$) má hodnotu u=*const*, potom sa daný vrchol považuje za *operátorový* a pri ´vstupe´ do takého v prvku $m \in IM$ výsledok spracovania $m' = A(m)$ vystupuje po tej hrane vrchola v, ktorý zodpovedá hodnote premennej u;t.j. ak u=*1* výstup bude na plusovú (+) hranu, v opačnom prípade na mínusovú (-) hranu.. Ak vrchol v má A=E(identický operátor), potom sa taký vrchol považuje za rozhodovcí (testovací) vrchol s podmienkou (predikátom) u. Vo všeobecnom prípade, ak $A \neq 0$ a $u \neq const$, to interpretujeme tak, že vo vrchole v s $\ell(v) = (A, u)$ sa realizuje operácia prognozovania $A \bullet [u]$ a to tak, že prvok $m \in IM$, ktorý vstupuje do v sa vytvorí kópia na ktorú sa aplikuje operátor A a v závislosti od hodnoty $u(m')$, kde $m' = A(m)$, pôvodný prvok m ´vystupuje´ z v po zodpovedajúcej hodnote $u(m')$ výstupnej hrane.
Bez narušenia všeobecnosti budeme predpokladať, že výstupný vrchol zgs $\widetilde{G}$ bude označený značkou $(E, const)$.
Z toho vyplýva, že g-s Klužnina sú parciálnym (špeciálnym) prípadom zgs.

Príklad 4.1

Ilustrujeme zavedené pojmy na príklade známeho hlavolamu o prevozníkovi, ktorý potrebuje prepraviť na druhý breh rieky vlka, zajac a kapustu. Loďka je taká malá, že sa do nej zmestí (okrem prevozníka) iba jeden z uvedených objektov. Prevozník nemôže, ak sa chce vyhnúť zjedeniu kapusty zajacom, alebo zajaca vlkom nechať na jednom brehu zajaca s kapustou, alebo vlka so zajacom. Na obr. 4.1 sú uvedené g-s reprezentujúce 2 riešenia hlavolamu a na obr. 4.2 je uvedená reprezentácia riešenia hlavolamu ako zgs.



Obrázok 4.1: Hlavolam prevozníka: graf-schéma

Vysvetlivky k obr. 4.1 vynecháme, pre dostatočnú jasnosť g-s. Na obr. 4.2 sú kvôli jednoduchosti reprezentácie vynechané mínusové hrany. Na obr. 4.3 je grafická reprezentácia operácie prognozovania .

□
(Koniec príkladu)

Hrubo vyznačená hrana znamená, že vstupujúci do vrchola A prvok m bude z vrchola u vystupovť po plusovej, alebo po mínusovej hrane, v závislosti od hodnoty m', kde $m' = A(m)$. V zgs obr. 4.3 sa predpokladá, že operátorovej premennej A logickej premennej u bude priradený niektorý konkrétny operátor $F \in OP$, resp. $\pi \in YC$ v súlade s interpretáciou zgs $\widetilde{G}$.
Na obr. 4.4 je uvedená zgs algoritmu triedenia BubbleSort, ktorá je získaná superpozíciou g-s BubbleSort z príkladu 1.7 a g-s previerky podmienky (predikátu) UM s použitím operácie prognozovania.

Pripomenieme, že

$$UM ::= SKAN \bullet [d(Y_1, K)]$$
$$\text{kde} \quad SKAN ::= \{[d(Y_1, K) \vee (l \leq r|Y_1)]P(Y_1)\}$$

Na obr. 4.4 je previerka realizovaná časťou gzs, ktorá je orámovaná prerušovanou čiarou. Nech G a $G^{'}$ s[gzs nad rovnakou rodinou V operátorových a logických premenných, $\mathbf{I} \subset (OP \cup YC)$ je interpretácia daných gzs. Budeme hovoriť, že dve schémy G a $G^{'}$ sú *ekvivalentné* a označovť to ako $G = G^{'}$, ak pre ľubovoľné $m \in IM$ bude platiť rovnosť

$$G/I(m) \quad = \quad G^{'}/I(m) \tag{4.1}$$

Inými slovami , dve ekvivalentné schémy pri zadannej interpretácii **I** budú dávať rovnaký výsledok na ľubovoľnom vybratom prvku $m \in IM$, alebo majú obidve schémy nedefinovaný výsledok na danom m. Problém ekvivlencie gzs je redukovateľný na problém ekvivalencie tkých reprezentácií algoritmov akými sú Turingove stroje, resp. iné algoritmické systémy. Je známe, že tento problém je pre tké systémy algoritmicky nerozhodnuteľný.
Zosilníme definíciu ekvivalencie zgs tak, že budeme dve schémy G a $G^{'}$ považovať za *ekvivalentné* a označovť to ako $G \equiv G^{'}$, ak rovnosť (4.1) platí pre ľubovoľnú interpretáciu **I**. Taká definícia privádza k *silnej ekvivalencii* gzs, nezávislej od interpretácie. Problém silnej ekvivlôencie pre gzs môže byť riešený analogicky ako je to v prípade schém Janova , pre ktoré je tento problém algoritmicky rozhodnuteľný [37].

Z definície gzs plynie, že ak máme dve gzs $\widetilde{G_1}$ a $\widetilde{G_2}$ nad množinou operátorových **A** a logických **U** premenných , potom výsledkom substitúcie schémy $\widetilde{G_2}$ do schémy $\widetilde{G_1}$ namiesto všetkých vrcholov označených dvojicou *(A,u)* bude nová gzs $\widetilde{G} = \widetilde{G_1}((A, u) \to \widetilde{G_2})$. To znamená, že množina gzs je *uzavretá* na operáciu superpozície .

Zostrojíme teraz algebru štrukturovaných g-s po analógii s SAA. Opereácie kompozície, alterntívy a cykla sú vyobrazené na obr. **??**. Konjunkcii zodpovedá sériové spojenie hrán označených ´+´ a ´-´, disjunkcii pralelné spojenie takých hrán, negácii komutácia (vzájomná výmena) značiek na vystupujúcich hranách. Grafová reprezentácia operácie prognozovania je na obr. 4.3. V takto vytvorenej algebre zgs - ZGS, algoritmy sú reprezentované v regulárnej (štruktúrnej) forme (PF)- superpozícii uvedených operácií, bázových operátorov predikátov príslušnej interpretácie **I**. Platí toto tvrdenie:

Veta 4.1 *K ľubovoľnej PC F(I) v SAA=(OP,YC;SIGN) môže byť vytvorená ekvivalentná zgs $\widetilde{G}/I$ v lgebre ZGS - $F(I) = \widetilde{G}/I$, naopak.*

□

Dôsledok 4.1 *Ľubovoľný algoritmus (alebo program) je vyjadriteľný v algebre ZGS prostredníctvom zodpovedajúcej PF.*

□

Tento dôsledok vyplýva z vety 4.1 a vety Gluškova o *regularizácii* ľubovoľných algoritmov programov v SAA.

Veta 4.2 *Nech AZG - algebra zgs; potom je AZG izomorfná s algebrou Gluškova AG.*

□

Na vytvorenú AZG môžu buť použité, resp. rozšírené už spomínané predtým v kapitole 2.5 výsledky štruktúrnej schemtológie. Koncepčná príbuznosť algebraických špecifikácií otvára možnosť spoločného používania grafových a analytických opisov (reprezentácií) algoritmov v procese mnohoúrovňového projektovnia tried algoritmov a programov [24, 25, 23]. Podobná interpretácia má svojim dôsledkom využitie

výhod a nivelizáciu nedostatkov prítomných tak v grafových ako aj v analytických špecifikáciach algoritmov, ak sú využívané oddelene. Tak bezpochyby medzi výhody analytických špecifikácií algoritmov patria: kompaktnosť, detailnosť, orientácia na transformáciu špecifikovaných objektov, neobmedzené možnosti na použitie komentárov orientovných na objasnenie niektorých stránok špecifikácie. Spolu s tým, pri opise relatívne jednoduchých algoritmov, grafové reprezentácie sú transparentnejšie v porovnaní s analytickými. Žiaľ táto výhod sa stáva nevýhodou pri prechode k projektovniu zložitých algoritmov, ktoré sú základom pri projektovaní aplikačných programových systémov rôznej orientácie. Tak, napríklad bolo konštatované , na základe skúseností získaných z projektovnia OS IBM 360 (Brooks [8]), že pri projektovaní veľkého systému máme čo robiť nie s jedným grfom, ale s celým albumom grafov. Práca s takým albumom pri značne obmedzených možnostiach pre umiestňovanie pojasňujúcich komentárov v grafoch, sa stáva mimoriadne zložitou a problematickou. Preto, je rozumné a výhodné využívať spolu grafové a analytické špecifikácie na každej úrovni propjektovania progrmového systému. Taký spôsob spočíva sprvidla v projektovní relatívne malých algoritmov a programov. Vzájomne prepojené poúrovňové projektovnie v termínoch grafových a analytických špecifikácií takých algoritmov a programov je založené na možnosti automatizovaného prechodu od jednej formy k druhej, stimuluje prehĺbenie analýzy projektovaných algoritmov, ich transformáciu a syntézu programov s využitím nástrtojov na ich generovanie.

4.2 Metaalgebra algoritmiky a štruktúra jej subalgebier.

Pristúpime teraz k vytvoreniu algebry spojenej s teóriou schém algoritmov a programov a k štúdiu vlastností štrtuktúry jej subalgebier. Podotýkame, že také štúdium má mimoriadny význam pretože subalgebry tejto algebry tvoria základ celého radu známych algoritmických systémov na najvyššej úrovni, akými sú SSA Gluškova, graf-schémy Kalužnina, logické schémy Ljpunov, Janova a ďalších.

Majme $A = \{A_1, A_2, ...\}$ a $U = \{u_1, u_2, ...\}$ rodiny operátorových a logických premenných a nech $V = A \cup U$ je ich zjednotením. Mechanizmus tvorby algoritmov je založený na zfixovaní elementárnych operátorových a logických konštrukcií (termov) a tvorbe , pomocou ich superpozícií, zložených (odvodených) termov, ktoré predstavujú schémy algoritmov v analytickom, grafovom tvare, alebo v tvare prirodzeného jazyka.

Zafixujme teraz niektorý výber T elementárnych termov.

$$T \quad = \quad \{A * B\text{-kompozícia}; ([u]A, B)\text{-alternatíva}; \{[u]A\}\text{-cyklus}\}$$

Superpozíciami uvedených elementárnych konštrukcií slúžia potom zložené štrukturované logické schémy. Ako príklad uvedieme uvedeným spôsobom vytvorenú štruktúrnu schému:

$$\Pi ::= \{[u_1]\,\{[u_2]([u_3]A, B) * C\} * D\}$$

Pri vhodnom výbere interpretácie **I**(nahradení (substitúcii) konkrétnych operátorov a predikátov namiesto operátorových a logických premenných) schémy Π môžu buť získané analytické špecifikácie rôznych algoritmov, konkrétne napríklad stratégie BUBBLE a príslušného algoritmu triedenia postupností (viď príklad 1.7)

Majme dva termy $t(v_1, v_2, ..., v_m)$ a $t(w_1, w_2, ..., w_n)$, kde $v_i, v_p) \in V$, i=1,2,...,m a j=1,2,...,n. Superpozícia $t(v \to t')$ označuje zámenu všetkých výskytov premennej v v terme t na term t').. Pri superpozícii sa logické premenné zamieňajú za logické termy a operátorové premenné za operátorové termy. Toto sa označuje ako *1. podmienka korektnosti superpozície.*

Pod *metaalgebrou algoritmiky* (MA) budeme rozumieť 2-druhovú algebru

$$MA = (\{AO, AY\}; SUPER)$$

kde AO,AY sú osnovy algebry:AO je množina všetkých logických schém algoritmov s premennými z V, AY je množina všetkých boolovských funkcií; SUPER je signatúra pozostávajúca z operácií superpozícií termov, ako aj operácií stotožnenia a premenovania premenných v termoch.

Algebru MA je možno vlastne považovať za systém pozostávajúci z dvoch algebier: *algebry operátorových termov*-AO a *algebry logiky* - L(2). To zapisujeme ako

$$MA = AO + L(2) \tag{4.2}$$

a signatúra SUPER je vztiahnutá primerane na obidve algebry. Symbol ´+´ použitý v (4.2) označuje tzv. *priamy súčet* algebier: $AL = AL^{'} + AL^{''}$ ak $AL = AL^{'} \cup AL^{''}$ a $AL^{'} \cap AL^{''} = \Phi$.

Majme niektorú algebru $\widetilde{A} = (A; W)$ a zvolíme si $B \subset A$ - podmnožinu osnovy A a takú, že B je uzavretá vzhľadom na všetky operácie signatúry W. Potom algebra $\widetilde{B} = (B; W)$ je *subalgebrou* algebry A. Výber zoskupenia elementárnych operátorových a logických termov a ich uzáveru (closure) vzhľadom na operácie signatúry SUPER vytvára niektorú subalgebru algebry A. Tak príkladmi subalgebier algebry logiky (AL) slúžia jej triedy T_0, T_1, S, M, L.

Subalgebra $\widetilde{B}$ algebry $\widetilde{A}$ je jej *maximálnou* subalgebrou , ak v dôsledku výberu ľubovoľného takého prvku $q \in \widetilde{A}$, ale $q \notin \widetilde{B}$ získame systém generátorov algebry $\widetilde{A}$. Príkladmi maximálnych subalgebier algebry logiky (AL) slúžia jej pre-úplné triedy T_0, T_1, S, M, L.

Zaujímavým sa javí problém preskúmania štruktúry subalgebier algebry MA, určiť mohutnosť danej štruktúry a sformulovať kritérium funkcionálnej úplnosti pre najzaujímavejšie subalgebry.

Platí táto

Veta 4.3 *Štruktúra subalgebier algebry MA zahŕňa (ako svoju súčasť) štruktúru subalgebier AL L(2).*

Dôkaz: Vyplýva z definície MA a fundamentálnej práce E.Posta, ktoré sú uvedené v kapitole 3 .

□

Dôsledok 4.2 *Mohutnosť štruktúry subalgebier algebry MA je prinajmenšom spočítateľná .*

Upresnenie tohoto tvrdenia dáva táto

Veta 4.4 *Zoskupenie všetkých subalgebier algebry MA má mohutnosť (kardinalitu)kontinua.*

Dôkaz: Dôkaz je založený na niekoľkých faktoch .

1. Je známe, že množina subalgebier AL je spočítateľná (viď veta);
2. Je známe, že množina s mohutnosťou kontinua (kontinuálna množina) je nespočítateľná ;

Idea dôkazu spočíva v zostrojení subalgebry algebry MA so spočítateľnou báizou a takou, že každej submnožine tejto bázy bude zodpovedať niektorá subalgebra algebry MA; pričom rôznym takým submnožinám budú zodpovedať rôzne subalgebry algebry MA. Je známe, že množina podmnožín spočítateľnej množiny je nespočítateľná.

Majme niektorú premennú $K \in A$ a na jej základe vytvoríme ďalšiu množinu

$$\underline{K} = \{K^n | n \in N\}\,, \text{kde} \quad K^n = \underbrace{K * K * ... * K}_{n} \text{kompozícia dĺžky n}$$

Nech teraz $t, t^{'} \in \underline{K}$ sú 2 kompozície, kde $|t| = r$ a $|t^{'}| = s$, a teda ide o kompozície dĺžky r resp. s. Ich superpozícia $t(K \rightarrow t^{'})$ bude opäť kompozícia dĺžky $r \cdot s$. To znamená, že trieda $\underline{K}$ je uzavretá na superpozíciu a teda máme, že

$$\widetilde{\underline{K}} = (K; SUPER), \text{je subalgebra algebry MA}$$

Subalgebra $\widetilde{\underline{K}}$ bude mať nekonečnú bázu

$$K(P) = \{K^p | p \in P\}, \text{kde } P \text{ je mnopžina prvočísel}$$

Skutočne, každé prirodzené číslo n sa dá jednoznačne vyjadriť ako súčin konečného počtu prvočísel, t.j. $n = p_1^{r_1} \cdot p_2^{r_2} \cdot ... \cdot p_k^{r_k}, \quad p_i \in P$. Ľubovoľná podmnožina množiny K(P) vytvára bázu niektorej subalgebry alhgebry MA; skutočne

$$K_i(P) = \{K^p | p \in P_i = \{p_1, p_2, ..., p_i\}, i \in N\},$$

Vytvoríme teraz množinu

$$A_{k_i} = \{K^n | n = p_{j_1} \cdot p_{j_2} \cdot ... \cdot p_{j_n}, p_{j_s} \in P_i, s = 1, 2, ..., n\},$$

Vezmime teraz $t, t^{'} \in A_{k_i}$, pričom $|t| = \ell, |t^{'}| = q$; dĺžka superpozície $|t(K \rightarrow t^{'})| = \ell \cdot q = p_{j_1} \cdot p_{j_2} \cdot ... \cdot p_{j_{\ell_q}}, p_{j_s} \in P_i, s = 1, 2, ..., \ell_q$. Máme teda, že $t(K \rightarrow t^{'}) \in A_{k_i}$. Je preto

$\widetilde{A}_{K_i} = (A_{K_i}; SUPER)$, subalgebra algebry MA a $\widetilde{A}_{K_i}$ bude jedinečná subalgebra algebry MA .

Je faktom, že množina podmnožín množiny prvočísel P (P obsahuje nekonečne veľa prvočísel) má mohutnosť kontinua. Štruktúra subalgebier algebry MA má taktiež mohutnosť kontinua. Veta je dokázaná.

□

Z hľadiska teoretického, ale aj praktického predstavuje zaujímavý problém štúdium subalgebier algebry MA, ktoré tvoria základ známych algebraických systémov najvyššej úrovne, pre ktoré je algoritmicky rozhodnuteľný problém silnej ekvivalencie. Medzi také systémy sa radia nám už známe 4 algebry algoritmov:

1. Dikstrova metaalgebra $\widetilde{AD} = ACC + L(2)$, kde ACC je algebra štruktúrnych schém vytvorená systémom generátorov totožných so signatúrou algebry AD (viď kapitolu 2 časť 2.2);

2. Kalužninova metaalgebra $\widetilde{AK} = HC + L(2)$, kde HC je algebra neštruktúrnych schém reprezentovaných v grafovej, alebo v analytickej forme [25];

3. Gluškovova metaalgebra $\widetilde{AG} = AO + AY$, kde AO a AY sú zodpovedajúco algebra operátorov a podmienok vytvorených systémom generátorov totožným so signatúrou operácií SAA.

Pretože systémy generátorov uvedených algebier obsahujú kompozíciu, platí nasledujúca inklúzia ($ACC \cap HC \cap AO) \supset K(P)$

Potom platí nasledujúce tvrdenie

Veta 4.5 *Každá zo subalgebier algebier $\widetilde{AD}, \widetilde{AK}, \widetilde{AG}$ obsahuje kontinuum subalgebier.*

□

V ďalšom sa budeme venovať štúdiu rodiny maximálnych subalgebier algebry $\widetilde{AD}$, ktorá podľ svojej konštrukcie je subalgebrou algebry $\widetilde{MA}$. Analogické výsledky boli dokázané pre algebry $\widetilde{AK}$ a $\widetilde{AG}$ [**?**].

4.3 Kritérium funkcionálnej úplnosti v metaalgebre Dijkstry.

V tejto časti sa budeme venovať riešeniu problému funkcionálnej úplnosti metaalgebry MA, ktorú budeme považovať za subalgebru algebry Dijkstry $\widetilde{AD}$, t.j. ako 2-druhová algebra so signatúrou pozostávajúcou z operácie superpozície.

V súvislosti s algebrami algoritmov, s ktorými sme sa oboznámili v kapitole 2 vznikajú dve základné otázky:

1. čo spoločné majú uvedené 4 algebry algoritmov?
2. ako sa dajú vytvárať nové algebry algoritmov?

Spoločné črty uvedených algebier sú v tom, že každá z nich je založená na superpozícii operácií zo signatúry týchto algebier. Práve pomocou takej superpozície operácií sa vytvárajú zložené operátory a predikáty, ktoré patria do osnov zodpovedajúcich algebier. Zároveň však výber operácií, ktoré tvoria signatúru algebry algoritmov je spojený s prechodom na ´dalšiu - vyššiu úroveň analýzy, na ktorej sa realizuje konštruovanie metaalgebry, ktorá zahŕňa algebry na najbližšej nižšej úrovni. Známou analógiou takej metaalgebry slúži algebra logiky AL (viď kapitolu 3). Konkrétne, v časti 3.2 je uvedený dôkaz o funkcionálnej úplnosti a sformulované zodpovedajúce kritérium v termínoch pre-úplných tried (maximálnych subalgebier) AL.
Pristúpme teraz k vytvoreniu maximálnych subalgebier v 2-druhovej algebre $\widetilde{AD}$ (viď kapitolu 4.4). Kritérium funkcionálnej úplnosti v tejto algebre sa definuje v pojmoch maximálnych subalgebier. Pripomeňme si, že $\widetilde{AD} = ACC + L(2)$, kde ACC je algebra štruktúrnych schém, determinovaná systémom generátorov D a L(2) je algebra logiky.

K maximálnym subalgebrám v $\widetilde{AD}$ budú patriť predovšetkým systémy typu $ACC + T_0$, $ACC + T_1$, $ACC + S$, $ACC + L$, $ACC + M$, kde T_0, T_1, S, L, M sú maximálne subalgebry AL. Prvky, ktoré nepatria k uvedeným systémom vyhovujú kritériu funkcionálnej úplnosti v AL a sú preto vyjadriteľné v ľubovoľnej báze L, napríklad pomocou konjunkcie a negácie. To znamená, že v procese ďalších konštrukcií môže byť použitá ľubovoľná b.f. vrátane konštánt 0 a 1.

Najprv ukážeme , že v systéme generátorov D je *alternatíva* odvoditeľná z *kompozície* a *cyklu*.

Skutočne, ak realizujeme v kompozícii $A * B$ superpozíciu

$$A * B(A \to \{[u]\, A\}\,; B \to \left\{\left[u^{'}\right] B\right\})$$

$$\text{dostaneme } \{[u]\, A\} * \left\{\left[u^{'}\right] B\right\}$$

$$/ * \text{ podľa rovnosti } \{[u]\, G\} * H = ([u]\, H, G * \{[u]\, G\} * H) * / \tag{4.3}$$

$$= ([u] \left\{\left[u^{'}\right] B\right\}, A * \{[u]\, A\} * \left\{\left[u^{'}\right] B\right\}) =$$

$$/ * \text{ podľa rovnosti } \{[u]\, G\} = ([u]\, E, G) * \{[u]\, G\} * / \tag{4.4}$$

$$= ([u]\, (\left[u^{'}\right] E, B * \left\{\left[u^{'}\right] B\right\}, A * \{[u]\, A\} * \left\{\left[u^{'}\right] B\right\})) \tag{4.5}$$

Podľa 2. podmienky korektnosti operácie *superpozícia* je možné dvom výskytom rovnakej logickej premennej, ktoré sú vo formule rozdelené výskytom niektorého operátora, priradiť odlišné logické hodnoty. Priradíme preto v (4.5) prvému výskytu logickej premennej $u^{'}$ hodnotu 0 a druhému a tretiemu výskytu logickej premennej $u^{'}$ a druhému výskytu u hodnotu 1. Ak si teraz uvedomíme, že

$$[0]\, G, H) = H,\ \{[1]\, G\} = G\ E * G = G * E = G$$

dostávame, že

$$([u]\, (\left[u^{'}\right] E, B * \left\{\left[u^{'}\right] B\right\}, A * \{[u]\, A\} * \left\{\left[u^{'}\right] B\right\})) = ([u]\, B, A)$$

To znamená, že kompozícia a báza tvoria bázu v algebre $\widetilde{ACC}$. Ukážeme aj *alternatíva* a *cyklus* tvoria bázu pre $\widetilde{ACC}$. Skutočne, ak použijeme k cyklu $\{[u]\,A\}$ rovnosť (4.4) dostaneme

$$([u]\,E, A * ([u]\,E, A * \{[u]\,A\}))$$

Ak teraz výskytu logickej premennej u v alternatívach priradíme hodnotu 0 a v cykle hodnotu 1 dostaneme

$$([u]\,E, A * ([u]\,E, A * \{[u]\,A\})) = A * A$$

Ak do získanej kompozície $A * A$ dosadíme za A alternatívu $([u]\,A, B)$ dostaneme $([u]\,A, B) * ([u]\,A, B)$. Dosadením prvému výskytu logickej premennej u hodnoty 1 a vdruhom hodnotu 0 dostaneme kompozíciu $A * B$.

Majme operátorovú schému $F \in ACC$ v algebre $\widetilde{AD}$; ak nahradíme výskyty niektorých výskytov logických premenných v alterenatívach a cykloch v schéme F logickými hodnotami 0, alebo 1 dostaneme niektorú schému, ktorú označíme ako $F^{'}$. Takú transformáciu voláme *1-konvolúciou* schémy F na $F^{'}$. Uvedieme

Príklad 4.2

Majme schému

$$G ::= ([u_1 \wedge u_2]\,A_1, A_2) * A_3 * \{[u_1]\,([\overline{u}_2])A_2, A_1\}$$

Dosadíme $u_1 = u_2 = 1$ v prvej alternatíve a $u_2 = 0$ vo vloženej alternatíve; dostaneme schému

$$G^{'} ::= A_1 * A_3 * \{[u_1]\,A_2\}$$

□
(Koniec príkladu)

Schéma F sa volá *NT-schémou* ak nie je transformovateľná pomocou 1-konvolúcie na *identický* operátor *E*.

Príklad 4.3

Schéma

$$G^{'} ::= A_1 * A_3 * \{[u_1]\,A_2\}$$

je NT schémou, avšak schéma

$$G^{''} ::= \{[u_1 \vee u_2]\,([u_3]\,A_1, A_3)\}$$

nie je NT schémou; ak dosadíme $u_1 = u_2 = 1$ dostaneme schému

$$G^{''} ::= E$$

□
(Koniec príkladu)

Označíme si množinu všetkých NT schém v ACC ako *NT*. Dokážeme platnoosť nasledujúceho tvrdenia.

Lemma 4.1 *Systém*

$$\widetilde{NT} = (\{NT, L(2)\}\,; SUPER)$$

je maximálnou subalgebrou algebry $\widetilde{ACC}$

Dôkaz: Systém $\widetilde{NT}$ je uzavretý na operácie signatúry SUPER a teda vytvára subalgebru algebry $\widetilde{MA}$. Nech **F** je ľubovoľne vybratá schéma a taká, že $F \notin NT$; potom z definície NT-schémy plynie, že bude existovať 1-konvolúcia schémy **F** vedúca na operátor **E**. Pridaním, v schéme $G^{'} \notin NT$ (viď (Príklad 4.3)), premenným A_1 a A_3 hodnoty **E** dostaneme cyklus $\{[u_1]\, A_2\}$. Zohľadňujúc fakt, že $A_1 * A_3 \in \widetilde{NT}$ prichádzame k záveru, že pripojením k subalgebre $\widetilde{NT}$ ľubovoľne zvolenej schémy $F \notin NT$ môže byť vytvorený systéme generátorov lgebry ACC. Máme, že subalgebra $\widetilde{NT}$ je maximálna.

□

Pod *C-konvolúciou* budeme chápať proces priradenia *všetkým*, alebo *niektorým* logickým a operátorovým premenným schémy **F** konštatntných hodnôt zodpovedajúco **0,1**, alebo **E**. Tak vyššie analyzovaný prípad procesu transformácie schémy $G^{'}$ na cyklus $\{[u_1]\, A_2\}$ možno považovať za príklad *C-konvolúcie* schémy $G^{'}$.

Budeme volať *degenerovanou alternatívou*- označovať ju budeme ako **ALT/Y**, konštrukciu

$$ALT/Y ::= ([u]\, A, E) = ([u]\, A)$$

Nech $F \in ACC$ je štruktúrna schéma vyjadriteľná v metaalgebre Dijkstry $\widetilde{AD}$. Operátorovú premennú A, od ktorej je závislá schéma F, budeme volať *zviazanou* premennou, ak v schéme F sú aspoň 2 výskyty premennej A. Majme schému F, ktorá obsahuje alternatívy a/alebo cykly. Budeme hovoriť, že schéma F vyhovuje podmienke *zviazanosti*, ak každá operátorová premenná , ktorá sa vyskytuje ne vetve alternatívy, alebo v tele cyklu je zviazaná. Takú schému F, ktorá vyhovuje podmienke *zviazanosti*, budeme volať *zviazanou*. Uvedieme príklady *zviazaných* schém. nasledujúce schémy sú zviazané schémy

$$G ::= ([u]\, A * \{[u]\, B\}\, , B * \{[u]\, A\}) \qquad \text{a} G^{'} ::= A * \{[u]\, A\}$$

Nech odteraz **NALT/Y** je zoskupenie takých schém, že každá $F \in NALT/Y$ vyhovuje aspoň jednej z týchto podmienok:

1. schéma F neobsahuje logické premenné, t.j. F=E, alebo F je operátorová premenná, alebo kompozícia takých premenných o konečnej dĺžke;
2. schéma F je zviazaná a naviac každá schéma $F^{'}$ získaná z F ako výsledok jej C-konvolúcie, bude patriť do **NALT/Y**.

To znamená, že zoskupenie **NALT/Y** obsahuje operátor **E**, kompozíciu typu $A * B * C$ ako aj schémy typu $A * ([u]\, A)$ a schémy typu $G^{'} ::= A * \{[u]\, A\}$. Na druhej strane však, schéma $G \notin NALT/Y$, pretože schéma $G^{''} ::= \{[u]\, B\}$ získaná z $G ::= ([u]\, A * \{[u]\, B\}\, , B * \{[u]\, A\})$ pomocou C-konvolúcie (prvý výskyt u=1 a A=E) nie je zviazanou schémou.

Lemma 4.2 *Systém* $\widetilde{NALT/Y} = (NALT/Y, L(2; SUPER))$ *je maximálnou subalgebrou* $\widetilde{ACC}$.

Dôkaz: Systém $\widetilde{NALT/Y}$ je uzavretý na operácie signatúry SUPER a z toho plynie, že tento systém vytvára subalgebru algebry $\widetilde{ACC}$. Skutočne, charakteristickou vlastnosťou podmienkových schém z NALT/Y (t.j. obsahujúcich alternatívy a/alebo cykly) je to, že vetvy alternatív a telá cyklov nie sú redukovateľné na operátor A, pretože obsahujú dva, alebo viac výskytov každej operátorovej premennej, od ktorých je závislá schéma. Superpozícia schém s takou vlastnosťou znovu privádza k schémam z NALT/Y . Máme teda, že $\widetilde{NALT/Y}$ je subalgebrou $\widetilde{ACC}$.
Vyberieme si ľubovoľne niektorú schému Q a takú, že $Q \notin NALT/Y$. Potom pomocou C-konvolúcie a invertovania premenných v Q dostaneme degenerovanú alternatívu ALT/Y. Zvolíme si ďalej schému $Q^{'} ::= \{[u]\, A * A\} \in NALT/Y$. Výsledkom superpozície $Q^{'}(A \to ALT/Y)$ získame schému $Q^{''} ::= \{[u]\, ([u]\, A) * ([u]\, A)\} \in NALT/Y$. Ďalej dosadíme namiesto druhej a tretej inštancie premennej u zodpovedajúco konštanty 0 a 1 a tým získame cyklus $\{[u]\, A\}$. Zohľadňujúc fakt, že kompozícia $A * B$ patrí do NALT/Y, prichádzame k záveru, že pridaním k subalgebre $NALT/Y$ ľubovoľnej schémy $Q \notin NALT/Y$ získame úplný systém generátorov algebry $\widetilde{ACC}$. Subalgebra $\widetilde{NALT/Y}$ je maximálna subalgebra.

□

Schéma F je *cyklická*, ak sa v nej vyskytuje aspoň jeden cyklus $\{[t]\, G\}$, pričom logický term $t \neq const$ a $G \neq E$. Také cykly budeme volať *podstatnými* cyklami. Schému F budeme volať *acyklickou* , ak neobsahuje podstatné cykly. Označíme si množinu acyklických schém $NC \subset ACC$; potom olatí nasledujúce tvrdenie.

Lemma 4.3 *Systém* $\widetilde{NC} = (NC, L(2; SUPER))$ *je maximálnou subalgebrou* $\widetilde{ACC}$.

Dôkaz: Systém $\widetilde{NC}$ je uzavretý na operácie signatúry SUPER a z toho plynie, že tento systém vytvára subalgebru algebry $\widetilde{ACC}$. Vyberieme si ľubovoľne niektorú schému F a takú, že $F \notin NC$. Potom pomocou 1-konvolúcie je možno F transformovať na schému $F^{'} ::= L_1 * IT * L_2$, kde $IT ::= \{[t]\, G_1\}$-podstatný cyklus. Aplikáciou C-konvolúcie schémy G_1 s nasledným stotožnením operátorových premenných dostaneme schému $F^{''} ::= A_n * \{[\widetilde{u}]\, A_k\} * A_m$, kde $\widetilde{u} \in \{u, \overline{u}\}$.
Ďalej vyberieme schému $ALT ::= ([u_1]\, B, C)$ na ktorú aplikujeme superpozíciu $F^{''}(A \to ALT)$. Výsledkom bude schéma $F^{'''} ::= ALT_n * \{[\widetilde{u}]\, ALT_k\} * ALT_m$.
Priradíme premennej C hodnotu E, všetkým inštanciám premennej u_1, s výnimkou tej ktorá sa nachádza v tele cykla, priradíme hodnotu 0 a tej v tele cyklu hodnotu 1.
Výsledkom je cyklus $\{[\widetilde{u}]\, B\}$. Pretože $A * B$ a ALT patria do NC, máme vytvorený systém generátorov algebry $\widetilde{ACC}$.

□

Majme množinu schém **I** , závislú len od jednej operátorovej premennej; $I \subset ACC$.

Lemma 4.4 *Systém* $\widetilde{\widetilde{U}(A)} = (I, L(2; SUPER))$ *je maximálnou subalgebrou* $\widetilde{ACC}$.

Dôkaz: Systém $\widetilde{U}(A)$ je uzavretý na operácie signatúry SUPER a z toho plynie, že tento systém vytvára subalgebru algebry $\widetilde{ACC}$. Vyberieme si ľubovoľne niektorú schému F a takú, že $F \notin I$.Posledné znamená, že F bude obsahovať prinajmenšom výskyty dvoch operátorových premenných A a B a takých, že $A \neq B$. Potom pomocou C-konvolúcie a známych rovností platných v algebre $\widetilde{AD}$ sa F transformuje na schému $F^{'}$, ktorá obsahuje v úlohe operátorových premenných iba A a B.
Možné sú dva prípady:

1. $F^{'} ::= A_n * B_m$
2. $F^{'} ::= ([u]\, A_n, B_m)$

Vyberieme si schémy ALT/Y a $ALT/Y^{'}$ patriace do **I** a také, že $ALT/Y ::= ([u]\, A)$ a $ALT/Y^{'} ::= (\left[u^{'}\right] B)$.

Aplikujeme superpozíciu $F(A \to ALT/Y), B \to ALT/Y^{'})$. Výsledkom bude schéma G, ktorá bude závisieť od operátorových premenných A a B a logických premenných u a $u^{'}$. Transformujeme teraz schému G prostredníctvom priradenia premenným u a $u^{'}$ nasledujúcich hodnôt: zafixujeme si po jednej inštancii schémy ALT/Y a $ALT/Y^{'}$ v G tak, že $u = u^{'} = 1$ v týchto inštanciach, zatiaľ čo v ostatných inštanciach ALT/Y a $ALT/Y^{'}$ budú hodnoty premenných $u = u^{'} = 0$. Pri tom, v závislosti od formy reprezentácie schémy F, schéma G sa transformuje na kompozíciu $A * B$, alebo alternatívu $([u]\, A, B)$.
Teda konštatujeme, že pridanie k subalgebre $\widetilde{U}(A)$ ľubovoľne zvolenej schémy $F \notin I$ privádza k báze v algebre $\widetilde{ACC}$. Subalgebra $\widetilde{U}(A)$ je maximálna subalgebra algebry $\widetilde{ACC}$.

□

Prejdeme teraz k algebre $\widetilde{AD} = ACC + L(2)$. Pre $\widetilde{AD}$ je L(2) izolovanou subalgebrou (viď [35]), čo znamená, že substitúcia boolovskej funkcie do ľubovoľnej schémy z ACC privádza opäť k schéme z ACC. Z vlastnosti izolovanosti plynie, že ak A_m je maximálna subalgebra algebry ACC, potom $A_m + L(2)$ je maximálne subalgebra pre $\widetilde{AD}$.

Veta 4.6 *(Kritérium funkcionálnej úplnosti pre $\widetilde{AD}$) K tomu, aby ľubovoľne vybraté zoskupenie prvkov*

$$SYST = SYST_1 \cup SYST_2, \quad kde SYST_1 \subset ACC, \quad SYST_2 \subset L(2)$$

bolo systémom generátorov algebry $\widetilde{AD}$ je nevyhnutné a postačujúce, aby:

1. *$SYST_1$ obsahoval:*
   - *aspoň jednu schému nepatriacu subalgebre $\widetilde{NT}$+L(2);*
   - *aspoň jednu schému nepatriacu subalgebre $\widetilde{NALT/Y}$+L(2);*
   - *aspoň jednu schému nepatriacu subalgebre $\widetilde{NC}$+L(2);*
   - *aspoň jednu schému nepatriacu subalgebre $\widetilde{U}(A)$+L(2);*
2. *$SYST_2$ vyhovoval kritériu funkcionálnej úplnosti pre algebru logiky.*

Dôkaz: Nevyhnutnosť plynie z uzavretosti týchto maximálnych subalgebier algebry $\widetilde{ACC}$: $\widetilde{NT}$, $\widetilde{NALT/Y}$, $\widetilde{NC}$, $\widetilde{U}(A)$ (Lemy **??**-4.4).
Postačujúca podmienka: prostredníctvom $SYST_2$-funkcionálne úplnej v L(2) môže byť zostrojená ľubovoľná b.f., vrátane konštánt 0 a 1, ako aj negácia.
Podľa Lemy **??** vyberieme schému $F \notin NT$ a pomocou 1-konvolúcie získame z F identický operátor E. Potom, na základe Lemy 4.2, zvoliac si schému $Q \notin NALT/Y$ a pomocou 1-konvolúcie a negácie logických premenných v schéme Q získame zo schémy Q redukovanú alternatívu ALT/Y. Ďalej aplikujeme C-konvolúciu zvolenej schémy $G \notin NC$, s využitím predtým zostrojených logických funkcií a operátorov, a získame cyklus $\{[u]\, A\}$(Lema 4.3). Napopkon pomocou C-konvolúcie a superpozície zvolenej schémy $H \notin U(A)$,s využitím predtým zostrojených logických funkcií a operátorov, na základe Lemy 4.4 zostrojíme kompozíciu, alebo alternatívu.
Tým sme dokázali, že ľubovoľne zvolený systém SYST, vyhovujúci podmienkam vety, sa dá previesť na systém generátorov algebry $\widetilde{AD}$. Veta je dokázaná.

□

Dôsledok 4.3

V algebre AD máme nasledujúce maximálne subalgebry:

$\widetilde{ACC}$+T_0, $\widetilde{ACC}$+T_1, $\widetilde{ACC}$+S, $\widetilde{ACC}$+M, $\widetilde{ACC}$+L, $\widetilde{NT}$+L(2), $\widetilde{NALT/Y}$+L(2), $\widetilde{NC}$+L(2), $\widetilde{U}(A)$+L(2).

□

Analogické výsledky je možno dokázať pre algebry $\widetilde{AG}$ a $\widetilde{AK}$ [29].

4.4 Algebra algoritmiky a aplikačné subalgebry

V tejto časti bude v úlohe matematického základu algoritmiky navrhnutý *2-úrovňový algebraický systém* ; horná úroveň bude tvorená *metaalgebrou algoritmiky* , ktorú sme vytvorili v kapitole a spodná úroveň bude tvorená *m-druhovými* algebraickými systémami ($m \geq 2$), ktoré sú formalizáciou koncepcie ADT a sú asociované s konkrétnymi predmetnými oblasťami.V úlohe posledných vystupujú triedy algoritmov, orientovaných na riešenie niektorých úloh ako je triedenie a vyhľadávanie informácií, jazykové procesory ap.

Pod m-druhovými algebraickými systémami (MAS) budeme chápať systém

$$S = (\{D_i | i \in I\}; SIGN_O, SIGN_\Pi)$$

kde D_i predstavujú osnovy (druhy) a $SIGN_O, SIGN_\Pi$ sú zodpovedajúco zoskupenia operácií a predikátov definovaných na druhoch D_i [35].

Ak $SIGN_O = \Phi$, ($SIGN_\Pi = \Phi$) prichádzame k definícii modelu (resp. algebry). Potom MAS predstavuje vlastne prirodzené zovšeobecnenie pojmov model a algebra (viď kapitolu 1.1). Ak druhy (osnovy) sa interpretujú ako množiny spracovávaných dát, potom MAS reprezentuje formalizáciu koncepcie ADT, ktorá nadobudla široké použitie v súvislosti s rozvojom metód objektovo-orientovaného programovania a s ním súvisiacich nástrojov.

MAS môžu buť použité v interpretácii logických schém algoritmov, ktoré sa na hornej úrovni reprezentované v MA, alebo v niektorej jej subalgebre.

Nech AD je Dikstrova algebra, generovaná systémom generátorov $SYST = SYST_O \cup SYST_Y$. Operátory a boolovské funkcie vystupujúce v SYST môžu byť považované za operácie definované na osnovách AO a L(2).

To znamená, že pripojením operácií, ktoré sa nachádzjú v SYST, k signatúre SUPER z algebry $\widetilde{AD}$ môže byť získaný systém SAA $SAD = (\{AO, L(2)\}; SUPER \cup SYST)$, ktorý je generovaný zoskupením $PER = PER_O \cup PER_Y$, kde

$$PER_O = \{A_i | i = 1, 2, ..., n\} \text{ operátorové premenné}$$

$$PER_Y = \{u_k | k = 1, 2, ..., m\} \text{ predikátové premenné}$$

Zadefinujeme si teraz zobrazenia

$$f : PER_O \to SIGN_O g : PER_Y \to SIGN_Y$$

kde $SIGN_O$ a $SIGN_Y$ sú zložky signatúry $SIGN$ niektorého

$$MAS = (\{D_i | i \in I\}; SIGN).$$

Zafixujeme si teraz 2-druhový algebraický systém

$$MASS = (\{mass, mum\}; SIGN_O \cup SIGN_Y)$$

kde $mass = \{M_q | q \in Q\}$ je zoskupenie označenných postupností typu $M_q : Ha_1a_2...a_iY_1a_{i+1}...a_nK$, $mum = \{R_t | t \in T\}$ je zoskupenie usporiadaných označenných postupností typu $R_t : HY_1s_1s_2...s_kK$; tu $s_i \leq s_{i+1}$ pre každé $i = 1, 2, ..., k-1$, $SIGN_0$ obsahuje operácie :$P(Y_1)$- posun ukazovateľa Y_1 o jeden prvok postupnosti $M_q \in mass$ doprava; $UST(Y_1, H)$- presun ukazovateľa Y_1 do pozície bezprostredne napravo od markera (značky) H- značky stojacej na začiatku postupnosti $M_q \in mass$. Signatúra $SIGN_Y$ bude obsahovať predikát $d(Y_1, K)$, ktorý je pravdivý ak Y_1 dosiahol marker (značku) K- značku, ktorá označuje koniec postupnosti $M_q \in mass$.

Vezmime si teraz schému Π z príkladu z kapitoly 2.1.

$$\Pi ::= \{[u_1] \{[u_2] ([u_3] A_1, A_2) * A_3\} * A_4\}$$

Schéma je vyjadrená v AD a závisí od operátorových premenných A_1, A_2, A_3, A_4 a logických premenných u_1, u_2, u_3.

Vytvoríme zobrazenia f_1 a g_1 tak, aby:

$$f_1(A_3) = P(Y_1),\ f_1(A_4) = UST(Y_1, H),\ g_1(u_2) = d(Y_1, K)$$

Ako výsledok získame schému

$$C\Pi ::= \{[u_1] \{[d(Y_1, K)] ([u_3] A_1, A_2) * P(Y_1)\} * UST(Y_1, H)\}$$

ktorá predstavuje mnohofázové spracovanie postupností $M \in mass$. Neinterpretované a čiastočne interpretované schémy sme nazvali *stratégiami spracovania dát*. Tieto stratégie sú formulované v termínoch označovania spracovávaných datových štruktúr, ktoré bolo zavedené v prácach [**?**].

Konštrukcia zobrazení f a g je realizovaná podľa jednotlivých úrovní plynule v súlade s koncepciou projektovania algoritmou a ich tried metódou zhora nadol (top-down). Takže , pokračujúc v konštrukcii interpretačných funkcií f_1 a g_1 podľa stratégie CII môže byť zostrojený známy algoritmus bublinkového triedenia postupnosti symbolov.

Do signatúry *mass*zavedieme operáciu $TRANSP(l, r|Y_1) \in SIGN_O$ permutácie (výmeny) prvkov l a r, ktoré sú susedné zľava, resp. zprava s ukazovateľom Y_1 v postupnosti M_q a predikáty $UM \in SIGN_Y$, ktorý bude pravdivý ak je postupnosť M_q utriedená a predikát $(l > r|Y_1) \in SIGN_Y$, ktorý bude pravdivý ak daný vzťah platí medzi uvedenými prvkami .

Doplníme definíciu zobrazení f a g tak, aby

$$f(A_1) = TRANSP(l, r|Y_1)\, f(A_2) = E$$
$$g(u_1) = UM\, g(u_3) = l > r|Y_1$$

a získavame algoritmus

$$BUBBLE ::= \{[UM]\,\{[d(Y_1, K)]\,([l > r|Y_1]\,TRANSP(l, r|Y_1), E) * P(Y_1)\} * UST(Y_1, H)\}$$

ktorý realizuje bublinkové triedenie postupnosti $M_q \in mass$.

Pri inej interpretácii stratégie CII môže byť získaný algoritmus polyfázového vyhľadávania záznamov v súboroch podľa postupnosti dopytov (queries, resp. inštrukcií ´Member´).

Vyššie sme ilustrovli proces postupnej poúrovňovej detailizácie logickej schémy II s cieľom získať ´bublinkovú´ stratégiu CII a nasledne algoritmus *BUBBLE*. Ďalej možno prejsť od jediného algoritmu k celej rodine algoritmov, ktorá bude asociovaná s konkrétnou predmetnou oblasťou (doménou), cestou postupného rozširovania systému generátorov tým, že budeme pridávať ďalšie operátory a predikáty ktoré sa vyskytujú v schémach , ktoré reprezentujú algoritmy v danej rodine .

V dôsledku tkého rozšíreni získame monotónne vzrstajúcu postupnosť subalgebier BA_i algebry $\widetilde{AD}$. S postupným rozširovaním systému generátorov musíme doplniž definíciu f a g, čím postupne fixujeme hodnoty operátorových a predikátových premenných, ktoré vytvárajú systém generátorov PER SAA, vytvorenú v rámci AD.

Napokon získame tak monotónne klesajúcu postupnosť subalgebier PD_i, pričom $PD_i \supseteq BA_i$, pre kždé i=1,2,...,k. Subalgebru PD_i budeme volať supertriedou a subalgebru BA_i budeme volať subtriedou pri výstavbe hľadaného mnohodruhového ADT. Platí táto veta.

Veta 4.7 *Majme mass mnohodruhový algebraický systém, ktorý reprezentuje vytvorený ADT a* $\{(PD_i, BA_i)\,|i = 1, 2, ..., k\}$ *je zodpovedajúca množina dvojíc (supertrieda, subtrieda) asociovaných s procesom tvorby hľadaného ADT; potom* $PD_1 \supset PD_2 \supset ... \supset PD_k$, $BA_1 \subset BA_2 \subset ... \subset BA_k$ *a* $mass =^k \bigcap_{i=1} PD_i =^k \bigcup_{i=1} BA_i$

□

Koncepcia 2-úrovňového algebraického systému zodpovedá procesu projektovania tried algoritmov a programov na 2 úrovniach. Tvoriaci jej základ aparát schém algoritmov a MAS sú polymorfné, čo umožňuje flexibilitu pri zmene vybratej problémovej oblasti.

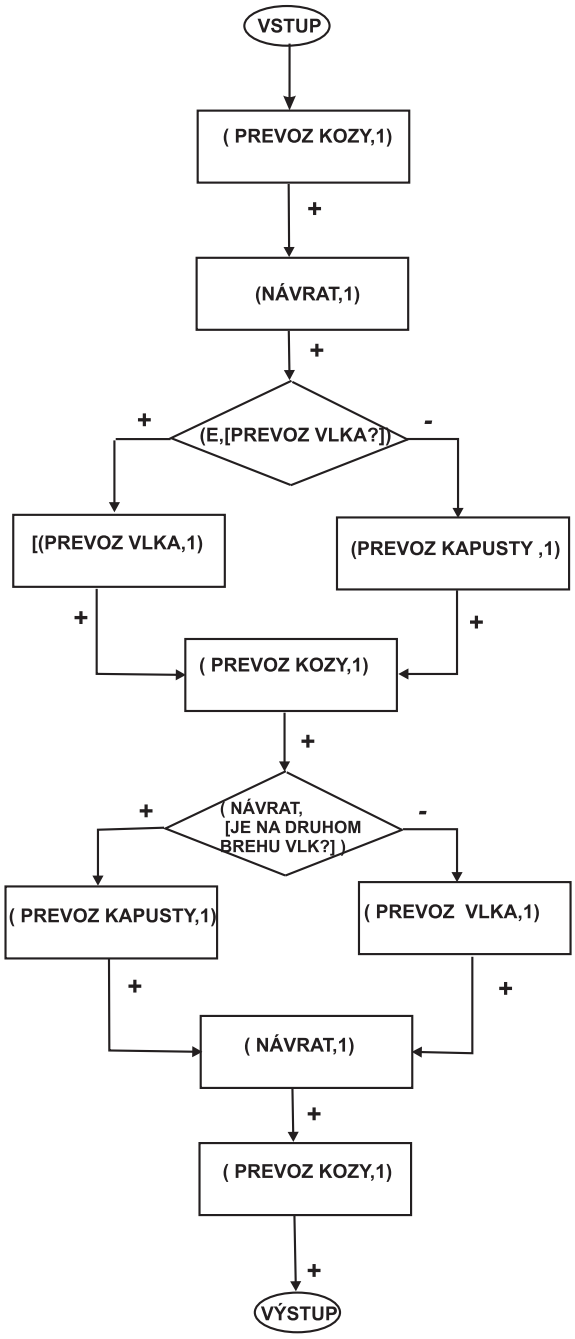
Prechod od supertriedy CK k subalgebre $PK \subset CK$ nižšej úrovne pri konštruovaní monotónne klesajúcich subalgebier pozostáva z dodefinovania interpretačných funkcií f a g; vďaka tomu v procese takého prechodu nastáva dedenie interpretácií supertriedy CK subtriedou PK.

Vďaka existencii transformačného aparátu, vyvinutého v rámci štruktúrnej schematológie, je možno transformovať projektované algoritmy a programy, konkrétne ich optimalizáciu podľa zvolených kritérií (viď kapitolu **??**).

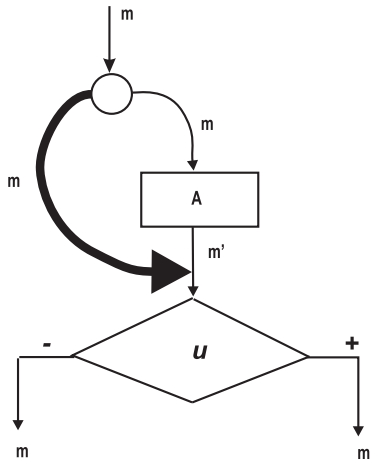
Možno povedať, že vyvíjaný aparát zodpovedá základným aspektom OO programovania . Získané výsledky odrážajú podstatu procesa reprezentácie algoritmických poznatkov, ktoré sú potvrdzované pri tvorbe klasifikačného grafu, ktorý charakterizuje vzájomné súvislosti medzi rôznymi stratégiami spracovania symbolickej (symbolovej) informácie a stým spojených algoritmov triedenia. Vrcholom takého grafu zodpovedajú jednotlivé triedy algoritmov spojených so známymi, ale aj novými metódmi triedenia a hranám prechody od jedných tried k druhým prostredníctvom rozpracovaných metapravidiel konštruovania algoritmov:transformácie, abstrakcia-konvolúcia a detailizácia - evolúcia (kapitola **??**). Pre algoritmy vnútorného triedenia sa dá sformulovať určitá analógia tézy Turinga-Churcha (tzv. lokálna téza Turinga-Churcha).

Lokálna téza Turinga-Churcha hlása, že pre ľubovoľný algoritmus vnútorného triedenia (sérioveho, alebo paralelného) existuje predstavujúca ho štruktúrna schéma v zodpovedajúcej subalgebre algebry Dijksty $\widetilde{AD}$.

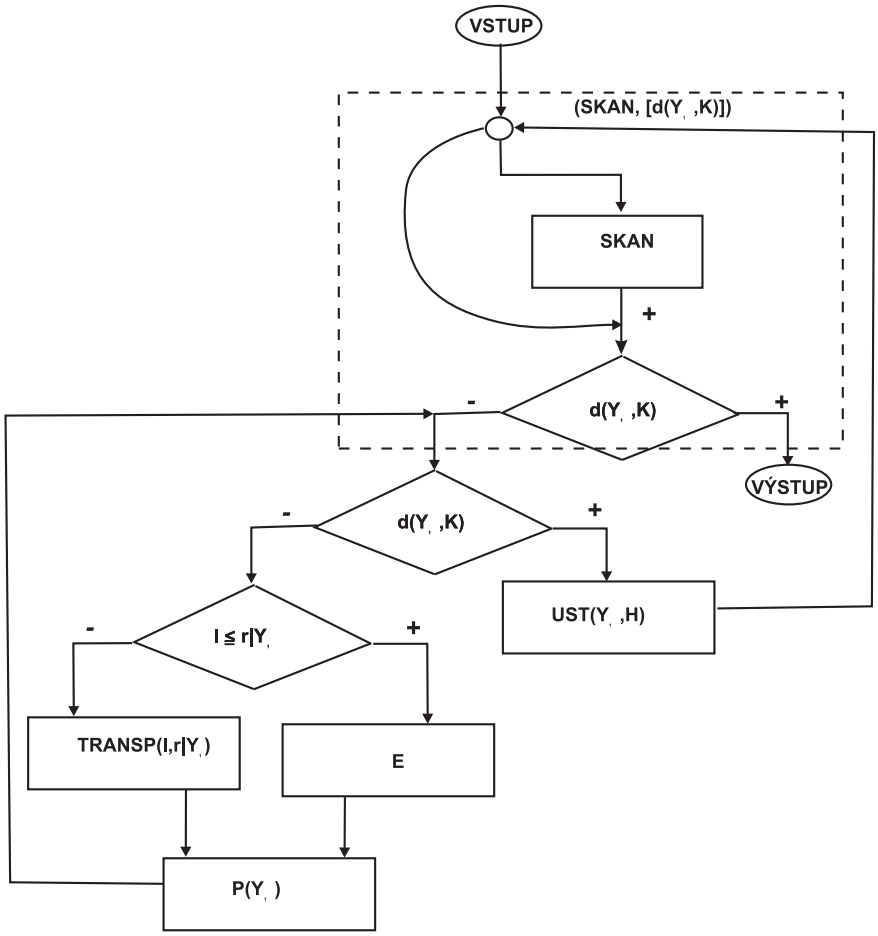
Praktické hľadisko realizovaných prác a výskumu spočíva v rozpracovaní perspektívnej technológie projektovania a syntézy tried algoritmov a programov, za základ ktorých je vzatá koncepcia objektovej orientácie a opakovaného použitia komponentov, ktoré sú reprezentované v tvare algoritmických termov, ktoré tvoria ooperátory a predikáty zodpovedajúcej aplikačnej algebry.



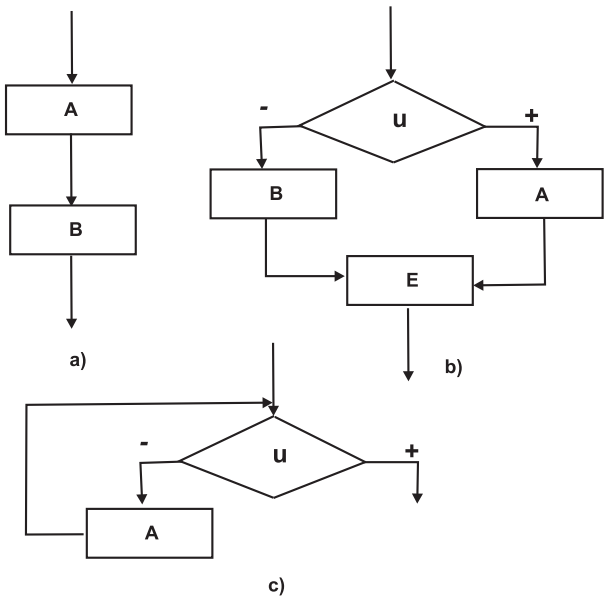
Obrázok 4.2: Hlavolam prevozníka:zovšobecnená g-s.



Obrázok 4.3: Zovšobecnená g-s operácie *prognoz*



Obrázok 4.4: Zovšobecnená g-s algoritmu BubbleSort



Obrázok 4.5: Operácie štrukturovaných g-s

Kapitola 5

Konštruovanie a klasifikácia algoritmov a stratégií spracovania dát.

V tejto kapitole sa budeme venovať konštruovaniu štruktúrnych schém, základom ktorých je aparát SAA (viď kapitolu 2.5). Budú zavedené špeciálne metapravidlá konštruovania schém:*konvolúcia* (zodpovedá abstrakcii); *evolúcia* (zodpovedá detailizácii (konkretizácii)); *preorientácia* (konvolúcia-evolúcia); a transformácia (transformácia schémy prostredníctvom rovností). Na základe uvedených metapravidiel sa realizujú optimalizačmé transformácie algoritmov triedenia za účelom zvýšenia ich adaptibility vzhľadom na povahu utrieďovanej postupnosti. V konečnom dôsledku je zostrojená klasifikácia adaptívnych algoritmov triedenia a stratégií spracovania dát, za pomoci ktorých je vytvorená rodina algoritmov vyhľadávania v hierarchických (m-úrovňových) súboroch.

5.1 Metapravidlá konštruovania schém a stratégií spracovania dát.

V tejto kapitole budú zavedené metapravidlá konvolúcie, evolúcie a ich kombinácií; tieto pravidlá patria medzi základné pravidlá pri konštruovaní schém algoritmov.

Pripomenieme si, že pod štruktúrnou (alebo *regulárnou*) schémou- PC sa rozumie *superpozícia* (alebo *term*), ktorá reprezentuje operátorový výraz v SAA.

Podobne, ako aj predtým, k identifikácii PC a ich subtermov budeme používať označenie

$$IDENTIFIKTOR ::= term, \text{alebo} \quad term =:: IDENTIFIKTOR;$$

operátorové a predikátové zložky (PC) sú sprevádzané opismi, resp. komentármi, ktoré budú vždy vymedzené zátvorkami $/*$ a $*/$.

Aparát SAA tvorí základ lingvistických prostriedkov jazyka SAA-schém, ktoré sú v ďalšom použité ku konštruovaniu formálnych algoritmických špecifikácií.

Na ilustráciu uvádzame teraz PC SH(k), ktorá predstavuje triedenie postupnosti podľa Shellovej metódy [12]. Podstata tejto metódy spočíva v rozdelení utrieďovanej postupnosti na disjunktné skupiny, každá o **k** prvkoch a postupnom utriedení prvkov postupnosti pomocou permutácií neusporiadaných dvojíc prvkov, ktoré sa nachádzajú od seba vo vzdialenosti k(k-vzdialené) . Potom sa parametru k priradí

hodnota rovná celej časti $\lfloor k/2 \rfloor$ a postupnosť sa rozdelí na nové skupiny o **k** prvkoch a postup sa opakuje až do úplného utriedenia celej postupnosti . Koncepcia označenia postupnosti podporuje realizáciu tejto metódy a to tak, že sa použijú dva indikátory Y_1 a Y_2, ktoré sú vo vzdialenosti k a usporiadaním prvkov, ktoré sú ´videné´ indikátormi Y_1 a Y_2 a po skončení jedného cyklu pridaním k novej hodnoty ako parametra pre nový cyklus. V súlade so Shellovou metódou je možno utrieďovanú postupnosť vnímať ako prepletenec k rozdistribuovaných subpostupností

$$PM_i ::= (a_i, a_{i+k}, a_{i+2k}, ...) \qquad \text{kde} \qquad i = 1, 2, ..., k;$$

pritom utrieďovanie prvkov daných subpostupností PM_i môže buť realizované jednou a tou istou procedúrou triedenia, alebo rôznymi procedúrami triedenia.

PC SH(k) triedenia postupnosti podľa metódy Shella má nasledujúci tvar.

Majme označenú postupnosť

$$M_0 : HY_0Y_1Y_2a_1a_2...a_nK$$

Uvedené označenie predstavuje počiatočné označenie postupnosti, kde Y_0, Y_1, Y_2 sú ukazovatele. ktoré sa budú presúvať po postupnosti v procese jej utrieďovania. Potom PC SH(k) má tvar:

Príklad 5.1

```
SH(k)  ::=
       begin
         k := n;
         while not Π do
           begin
             k := ⌊k/2⌋ ;
             while d(Y1, Y2) ≠ k do P(Y2);
             SORT(k) >;
             UST(Y0Y1Y2,H);
           end
             ELIM
       end
```

kde

- Π je predikát pravdivý vždy, ak je už postupnosť utriedená;
- $\lfloor k/2 \rfloor$ je celá časť podielu $k/2$;
- $d(Y_1, Y_2) = k$ je predikát pravdivý vždy, ak vzdialenosť medzi indikátormi $d(Y_1, Y_2)$ je k;
- $P(Y_2)$ je operátor posuvu indikátora Y_2 o 1 prvok doprava;
- $SORT(k) >$ je operátor usporiadania všetkých dvojíc prvkov rozmiestnených na vzdilenosť k;
- $UST(Y_0Y_1Y_2, H)$ je operátor umiestnenia indikátorov $Y_{0,1}$, Y_2 bezprostredne napravo od markera H;
- *ELIM* je operátor odstránenia indikátorov z utriedenej postupnosti.

Predikáty Π a $SORT(k) >$ sa dajú vyjadriť pomocou ďalších subtermov. Pritom v závislosti od výberu tej-ktorej stratégie spracovania dát, PC SH(k) bude zahrňovať celú rodinu algoritmov triedenia v rámci dnej metódy. Zvoľme si, napríklad, interpretáciu operátora $SORT(k) >$ stratégiu *BUBBLE* bublinkového triedenia postupnosti na vzdialenosť k rozdistribuovaných prvkov. Potom

Π ::=
begin
while not $[(l > r|Y_0) \vee d(Y_0, K)]$ **do** $P(Y_0)$;
if $d(Y_0, K)$**then** '*Postupnosť je utriedená*'
else '*Postupnosť nie je utriedená*';
$UST(Y_0, H)$;
end

$SORT(k >)$ ::= $BUBBLE(k) >$

$BUBBLE(k) >$ ::=
begin
while not $\Pi(k)$ **do**
begin
if $[l|Y_1 > l|Y_2]$ **then** $TRANSP(l|Y_1, l|Y_2)$
then E;
$P(Y_1, Y_2)$;
end
while not $[d(H, Y_1)]$ **do** $L(Y_1, Y_2)$;
end

kde

- $l > r|Y_0$ je predikát pravdivý vždy, ak ukázaná relácia platí medzi prvkami l a r nachádzajúcimi sa bezprostredne naľavo, resp. napravo od indikátora Y_0 ;
- $UST(Y_0, H)$ je operátor umiestnenia indikátora Y_0 bezprostredne napravo od markera H;
- ´TEXT´ - operátor výstupu textu medzi ´´ na displej;
- $\Pi(k)$ je predikát pravdivý vždy, ak sú v postupnocti utriedené všetky prvky, nachádzajúce sa na vzdialenosť k od seba;
- $l|Y_1, r|Y_2$ - ide o prvky postupnosti, nachádzajúce sa bezprostredne naľavo od Y_1, resp. od Y_2 ;
- TRANSP$(l|Y_1, r|Y_2)$ je operátor permutácie prvkov $l|Y_1$ a $r|Y_2$.

□
(Koniec príkladu)

V súlade s uvedenou PC, utrieďovanie každej podpostupnosti PM_i , na ktoré je rozvrstvená utrieďovaná postupnosť, sa realizuje metódou bublinkového triedenia. Z toho dôvodu uvedený algoritmus budeme označovať ako $BUBBLE(k) >$. V ďalšom sú použité aj iné interpretácie opoerátora $SORT(k) >$, ktoré sú orientované na riešenie problému zvýšenia stupň adaptability Shellovej metódy.

Nech v ďalšom pod $\widetilde{A} = \{A_i | i = 1, 2, ...\}$ budeme rozumieť množinu operátorových premenných, pod $\widetilde{U} = \{u_q | q = 1, 2, ...\}$ budeme rozumieť množinu predikátových premenných, a $V = \widetilde{A} \cup \widetilde{U}$.

Pripomewnieme si, že interpretovanou PC F voláme takú schému algoritmu, v ktorej sa nevyskytujú premenné z V. Schéma $F(v_1, v_2, ..., v_c)$ je *čiastočne interpretovanou* PC, alebo *stratégiou* spracovania dát, ak obsahuje premenné $v_1, v_2, ..., v_c \in V$.

Majme systém substitúcií I: $\{v_1 = G_1, v_2 = G_2, ..., v_r = G_r\}$, kde $v_i \in V$, $i = 1, 2, ..., r$ a G_i sú termy SAA $(i = 1, 2, ..., r)$.

5.1.1 Konvolúcia.

Vezmime si interpretovanú, alebo čiastočne interpretovanú schému F. Nech ďalej $B(G_1), B(G_2), ..., B(G_r)$ sú po dvojiciach nepretínjúce sa výskyty termov G_i v schéme F, $(i = 1, 2, ..., r)$. Výsledkom konvolúcie (zvinutia) schémy F podľa systému **I** budeme volať *stratégiou spracovania* $S = F \uparrow I$, ktorú získame zo schémy F prostredníctvom zámeny (nahradenia) výskytov $B(G_i)$ premennými v_i v súlade s **I**.

Príklad 5.2

Zavedieme si nasledujúce rovnosti

$$I_1 : \{v_{11} = \Pi, v_{12} = SORT(k) >\}$$

Stratégia $SP(k) = BUBBLE(k) \uparrow I_1$ bola získaná z PC $BUBBLE(k) >$, ktorá predstavuje prípad Shellovho triedenia , prostredníctvom systému I_1.

```
SP(k)  ::=
       begin
         k := n;
         while not v11 do
           begin
             k := ⌊k/2⌋ ;
             while d(Y1, Y2) ≠ k do P(Y2);
             v12;
             UST(Y0Y1Y2,H);
           end
             ELIM
       end
```

□
(Koniec príkladu)

5.1.2 Evolúcia (Rozvinutie).

Vezmime si čiastočne interpretovanú schému $F(v_1, v_2, ..., v_m)$, závislú od premenných $v_i \in V$ $(i = 1, 2, ..., m;\ r \leq m)$. Výsledkom rozvoja (evolúcie) schémy F podľa systému **I** nazveme schému $G = F \downarrow I$, ktorá je získaná prostredníctvom substitúcie v shéme F namiesto premenných v_i zodpovedajúcich termov G_i v súlade s rovnosťami **I**, kde $i = 1, 2, ..., r$. Systém **I** budeme volať *interpretáciou* schémy F, ak v dôsledku rozvinutia dostaneme schému $G = F \downarrow I$, ktorá neobsahuje žiadne premenné z V , alebo *čiastočnou interpretáciou* schémy F v opačnom prípade.

Nech ďalej $B(G_1), B(G_2), ..., B(G_r)$ sú po dvojiciach nepretínjúce sa výskyty termov G_i v schéme F, $(i = 1, 2, ..., r)$. Výsledkom konvolúcie (zvinutia) schémy F podľa systému **I** budeme volať *stratégiou spracovania* $S = F \uparrow I$, ktorú získame zo schémy F prostredníctvom zámeny (nahradenia) výskytov $B(G_i)$ premennými v_i v súlade s **I**.

Príklad 5.3

Nech $\Phi_0 : HY_1Y_2q_1q_2...q_nNIL$ je mnohoúrovňový (hierarchický) súbor a nech ďalej $M_3 : Y_3\pi_1\pi_2...\pi_sK$ bude postupnosť dopytov (queries), kde H a NIL sú markery označujúce (vynedzyjúce) súbor Φ_0 a K-marker konca postupnosti M_3 . Každý dopyt π_j je unárny predikát, definovaný na prvkoch súboru Φ_0 .

Spracovanie dopytu π_j , ktorý ´vidí´ Y_3 (ten, ktorý je bezprostredne napravo od Y_3) spočíva v nájdení všetkých prvkov súboru Φ_0, n ktorých je pravdivý predikát π_j. Úloha vyhľadávania v súbore predstavuje postupné spracovanie všetkých dopytov postupnosti M_3).

Zavedieme si nasledujúce rovnosti

$$I_2 : \{v_{11} = d(Y_3, K), v_{12} = SEARCH(k)\}$$

kde

- $d(Y_3, K)$ je predikát pravdivý vždy, ak indikátor Y_3 ´vidí´ marker K;
- SEARCH(k) je operátor vyhľadávania prvkov , ktoré vyhovujú predikátu aktuálneho dopytu, v podsúbore, ktorý je vymedzený indikátormi Y_1 Y_2 .

Výsledkom evolúcie stratégie SP(k) prostredníctvom systému I_2 dostaneme algoritmus Stratégia $SEARCH = SP(k) \downarrow I_2$, ktorý realizuje vyhľadávanie prvkov v súbore Φ_0 .

```
SEARCH ::=
        begin
          k := n;
          while not d(Y3, K) do
            begin
              k := ⌊k/2⌋ ;
              while d(Y1, Y2) ≠ k do P(Y2);
              SEARCH(k);
              UST(Y0Y1Y2,H);
            end
              ELIM
        end
```

□
(Koniec príkladu)

Odvodené metapravidlá sú vytvárané postupnosťou konvolúcií (zvinutí) a evolúcií (rozvinutí). Konkrétne preorientácia schémy F na schému G je takým metapravidlom pozostávajúcim z konvolúcie F podľa systému I_1 a evolúcie získanej stratégie podľa systému I_2. Máme teda, že $G = (F \uparrow I_1) \downarrow I_2$. Preorientáciu budeme tiež skrátene označovať ako $F \uparrow\downarrow G$. Potom v prípade nášho príkladu máme, že $SEARCH = (SH(k) \uparrow I_1) \downarrow I_2$, alebo zjednodušene $(SH(k) \uparrow\downarrow SEARCH$.

Špeciálnym prípadom preorientácie slúži *preinterpretácia*, spočívajúca v zámene bázy.

Metapravidlá konvolúcie a evolúcie a ich rôzne kombinácie sú orientované na konštruovanie tried algoritmov a stratégií spracovania dát (symbolov).

5.2 Metapravidlo transformácie schém a optimizácia triediacich algoritmov

Okrem konvolúcie a evolúcie zaraďujeme medzi základné metapravidlá konštruovania schém aj *transformáciu*- pretváranie (modifikácia) schém na základe aparátu identít (totožností) a rovností, ktoré boli rozpracované v rámci *štruktúrnej schematológie* [35, 15].

Táto kapitola je venovaná metapravidlu transformácie algoritmov, ktoré sú reprezentované v SAA. Transformácia tvorí základ analytických transformácií, ktoré sú typické pre algebraický prístup všeobecne a riešenia úloh (problémov) ´počítačovej´ algebry a logiky špeciálne.

Prv než pristúpime priamo k transformácii schém algoritmov uvádzame, že táto transformácia je založená na aparáte *identických*(totožnostných) modifikácii schém, ktoré boli vyvinuté v SAA. Metapravidlo

transformácie, spolu s konvolúciou a evolúciou sú silným nástrojom na ustanovenie vzájomných väzieb medzi rôznymi algoritmami a vytvárania nových algoritmov, ktoré sú efektívnejšie z hľadiska tých-ktorých kritérií.

V ďalšom uskutočníme transformačnú redukciu (prechod) algoritmu BUBBLE(k), ktorým sme sa zaoberali v kapitole 5.1, na algoritmus BUBBLE, ktorý bude prevedený na adaptívny algoritmus ČLNOK (SHUTTLE) založený na priamych vkladaniach, princípe, ktorý leží v základe tzv. člnkového triedenia (shuttle sorting) [12, 17]. Ďalej bude ralizovaná redukcia algoritmu SHUTTLE k algoritmu triedenie pomocou *alternatívnych vkladaní* CAB, ktorý má lepšiu časovú zložitosť v porovnaní s bublinkovým, či člnkovým triedením. Tieto prístupy možno považovať za proces opotimalizácie známych algoritmov triedenia, vďaka ktorým bol vytvorený veľmi efektívny algoritmus CAB [17, 18]. V tomto algoritme sa prvý raz objavil princíp, ktorý bol položený ako základ pre metódu SKIPSORT [19].

Identické (totožnostné) transformácie formúl-analytických reprezentantov objektov v niektorej algebre-sa realizujú pomocou použitia pravidiel nasledujúceho typu:$t_1(x_1, x_2, ..., x_s) = t_2(x_1, x_2, ..., x_s)$, kde t_1 a t_2 sú termy danel algebry závislé od premenných $x_1, x_2, ..., x_s$. Rovnosi, ktoré sú pravdivé pri ľubovoľných interpretáciach-hodnotách premenných, od ktorých t_1 a t_2 závisia , sa volajú *identitami* (totožnosťami) v danej algebre.

Príkladmi takých identít môžu slúžiť zákony boolovej algebry. Identity sa v algebre používajú k formulám v smere zľava doprava, alebo naopak. Uvedieme teraz príklad.

Príklad 5.4

Pozrime sa na nasledujúci fragment formuly, ktorá predstavuje algoritmus $BUBBLE >$ v SAA

$$ALT_1 ::= C.L.\left([l > r|Y_1]\, TRANSP\left(l, r|Y_1\right), E\right) * P\left(Y_1\right) C.R.$$

kde C.L. a C.R. predstavujú ľavý, resp. pravý kontext daného fragmentu schémy $BUBBLE >$.
Vlastnosť distributívnosti operácií kompozície a alternatívy vyjadrujú nasledujúca rovnosť:

$$T_1 : ([u]\, A, B) * C = ([u]\, A * C, B * C)$$

kde u je logická premenná, A,B,C sú operátorové premenné.

Rovnosť T_1 platí pri všetkých interpretáciach premenných v nej a preto táto rovnosť môže slúžiť príkladom identity (totožnosti) v SAA. Zavedieme si tieto interpretácie premenných v T_1:

$$u ::= l > r|Y_1;\ A ::= TRANSP(l, r|Y_1);\ B ::= E;\ C ::= P(Y_1).$$

Dosadíme si zavedené interpretácie ako hodnoty premenných v T_1 s následným použitím T_1 vo fragmente ALT_1 v smere zľava doprava (t.j. zámenou ľavej časti T_1, zhodnej so zafixovaným fragmentom ALT_1 na jeho pravú časť), po odstránení zátvoriek alternatívy dostaneme:

$$ALT_2 ::= C.L.\left([l > r|Y_1]\, TRANSP\left(l, r|Y_1\right) * P(Y_1), E * P(Y_1)\right) C.R.$$

Použitie identity T_1 na ALT_2 v smere zprava doľava privedie k vytknutiu operátora $P(Y_1)$ za zátvorku alternatívy tak, že dostaneme pôvodný fragment ALT_1.

□
(Koniec príkladu)

Spolu s identitami môžu byť pri transformácii algoritmov použité aj *kvaziidentity*- rovnosti, ktoré platia len pre určité interpretácie ich premenných.

Príklad 5.5

Majme formulu nasledujúceho fragmentu algoritmu reprezentovaného v SAA

$$ALT_3 ::= C.L.\left([l > r|Y_1]\, TRANSP\left(l, r|Y_1\right), E\right) * \{[l < r|Y_1]\, P(Y_1)\}\, C.R.$$

Vezmime si nasledujúca rovnosť:

$$T_2 : ([u]\, A, B) * \{[u^*]\, C\} = ([u]\, A, B)$$

Rovnosť T_2 platí len pri tých reprezentáciach , pri ktorých je podmienka u^* pravdivá po realizácii vetiev alternatívy. Vytvoríme takú interpretáciu, ktorá bude zodpovedať interpretácii z príkladu 5.4.

$$u ::= l > r|Y_1;\ A ::= TRANSP(l, r|Y_1);\ B ::= E;\ C ::= P(Y_1).\ u^* ::= l < r|Y_1$$

Použitie identity T_2 na ALT_3 v smere zľava doprava dostaneme:

$$ALT_4 ::= C.L.\left([l > r|Y_1]\, TRANSP\left(l, r|Y_1\right), E\right) C.R.$$

Použitie identity T_2 na ALT_4 v smere zprava doľava dostaneme pôvodný fragment ALT_3.

□
(Koniec príkladu)

Poznamenajme, že v procese transformácie algoritmov môžu byť použité vzťahy, ktoré odrážajú vlastnosti aktuálnej problémovej domény. Príkladmi takých vzťahov pre interpretácie používané pri konštruovaní algoritmov triedenia sú nasledujúce rovnosti:

$$P(Y) * L(Y) = E;\ L(Y) * P(Y) = E;\ P(Y) * UST(Y, H) = UST(Y, H)$$

Za pomoci aparátu identít (totožností), kvaziidentít (kvazitotožností) a ďalších vzťahov, vyvinutých v teórii SAA je možno realizovať transformáciu algoritmu SH(k) na algoritmus BUBBLE, ktorý bol uvedený predtým.

Systémy identít, kvaziidentít a vzťahov savyužívajú na charakteristiku vlastností operácií, ktoré vystupujú v signatúre SAA a vo vybratých predmetných oblastiach.. V každom kroku transformácie sa definuje zodpovedajúci cieľ a je daná postupnosť rovností použitie ktorých zabezpečí dosiahnutie postaveného cieľa. Podobne ako predtým, príslušná informácia bude uvedená v šoeciálnych zátvorkach ´∗/´ a ´/∗´.

Vrátime sa teraz k schéme SH(k), ktorá bola reprezentovaná v tvare pseudokódu v 5.1 v kapitole 5. Vyjadríme teraz SH(k) ako term SAA:

$$\begin{aligned}
\text{SH(k)} > \quad ::= & \\
& k := n* \\
& * \{[\Pi]\,(k := \lfloor k/2 \rfloor)* \\
& \quad * \{[d(Y_1, Y_2) = k]\, P(Y_2)\} * \\
& \quad * \{[\Pi(k)]\, \{[d(Y_2, K)]\, ([l|Y_1 > r|Y_2]\, TRANSP(l|Y_1, r|Y_2), E) * P(Y_1, Y_2)\} * \\
& \qquad * \{[d(H, Y_1)]\, L(Y_1, Y_2)\} \\
& \qquad \} \\
& \} * \\
& *ELIM
\end{aligned}$$

Transformácia schémy SH(k)> na schému BUBBLE> pozostáva z nasledujúcich krokov:

$$
\begin{aligned}
\mathrm{SH(k)} > \quad ::= {} & /* \text{ Cieľ 1: vyčistenie cyklu pri k=0; indikátory } Y_1 \text{ a } Y_2 \text{ sú zhodné} \\
& \text{a preto platia nasledujúce rovnosti:} \\
& \Pi(0) = \Pi; \\
& \{[d(Y_1, Y_2) = 0]\, P(Y_1)\} = E; \\
& [l|Y_1 > r|Y_2] = [l > r|Y_1]\,, TRANSP(l|Y_1, r|Y_2) = TRANSP(l, r) * / \\
& k := n* \\
& * \{[\Pi]\,(k := \lfloor k/2 \rfloor)* \\
& \quad * \{[\Pi]\,\{[d(Y_1, K)]\,([l > r|Y_1]\, TRANSP(l, r), E) * P(Y_1)\} * \\
& \qquad * \{[d(H, Y_1)]\, L(Y_1)\} \\
& \qquad \} \\
& \} * \\
& *ELIM = /* \text{ Cieľ 2: Odstránenie operátorov priradenia hodnôt parametru k,} \\
& \qquad \text{ktorý v získanej schéme chýba, ako aj nepodstatných operátorov} \\
& \qquad \text{(medzi ktoré patrí konkrétne operátor ELIM)} \\
& \qquad \text{prostredníctvom použitia kvazitotožnosti} \\
& \qquad \{[u]\, A\} * B = \{[u]\, A\}, \text{ kde požitie B po } \{[u]\, A\} \text{ nemení spracovávné dáta } */ \\
& = \{[\Pi]\,\{[\Pi]\,\{[d(Y_1, K)]\,([l > r|Y_1]\, TRANSP(l, r), E) * P(Y_1)\} * \\
& \qquad * \{[d(H, Y_1)]\, L(Y_1)\} \\
& \qquad /* \text{ Cieľ 3:Zníženie hĺbky vložených iterácií pomocou identity} \\
& \qquad \{[u]\,\{[u]\, A\}\} = \{[u]\, A\} */ \\
& = \{[\Pi]\,\{[d(Y_1, K)]\,([l > r|Y_1]\, TRANSP(l, r), E) * P(Y_1)\} * \\
& \qquad * \{[d(H, Y_1)]\, L(Y_1)\} \\
& \quad \} \\
& \qquad * \text{ Cieľ 4:Odstránenie vloženého cyklu pomocou použitia vzťahu} \\
& \qquad \{[d(H, Y_1)]\, L(Y_1)\} = UST(Y_1, H) \\
& \qquad \text{s následnou zámenou tejto časti formulou } * / \\
& = \{[\Pi]\,\{[d(Y_1, K)]\,([l > r|Y_1]\, TRANSP(l, r), E) * P(Y_1)\} * \\
& \qquad * \; UST(Y_1, H)\} \\
& \qquad =:: BUBBLE >
\end{aligned}
$$

Práve analyzovaný proces redukcie algoritmu A na algoritmus B prostredníctvom použitia metapravidla transformácie budeme v ďalšom volať *transformačnou redukovateľnosťou* A na B a označovať ako

$$A = * = B(T)$$

kde T označuje zoskupenie použitých pri redukcii rovností. Niekedy môžeme použiť aj jednoducho

$$A = * = B$$

ak je jasné o aké zoskupenie T ide. Jeden krok transformačnej redukcie bez komentára budeme označovať

$$A = (T_i) = B$$

kde T_i je rovnosť použitá v danom krotku; šípka po T_i bude poukazovať na smer použitia rovnosti T_i: ¿- zľava doprava a ¡ - zprava doľava.

Na základe realizovaných transformácií platí:

Lemma 5.1 *Schéma SH(k)> je transformačne redukovateľná na na schému BUBBLE> a*

$$SH(k) >= (T_1) = BUBBLE >$$

kde T_1 je zoskupenie rovností , ktoré boli použité v procese transformácie.

Vzhľadom na obojsmerné použitie pravidiel z T_1 platí aj nasledujúce tvrdenie:

Dôsledok 5.1 *Schéma BUBBLE> je transformačne redukovateľná na na schému SH(k)> a*

$$BUBBLE >= (T_1) = SH(k) >$$

Schéma BUBBLE> je mnohofázový algoritmus triedenia a je z hľadiska časovej zložitosti neefektívny ($O(n^2)$).

Veľmi aktuálnym sa javí problém adaptácie algoritmov triedenia podľa stupňa utriedenosti triedenej postupnosti. Uskutočníme teraz transformáciu schémy BUBBLE> na schému realizujúcu triedenie pomocou priamych vkladaní, tzv. člnkové triedenie (shuttle, resp. insertion sort). Najprv však opíšeme proces transformačnej redukcie schémy BUBBLE> na schému ROZTOK> , ktorý je predstaviteľom jednoduchých algoritmov 1-fázového triedenia.

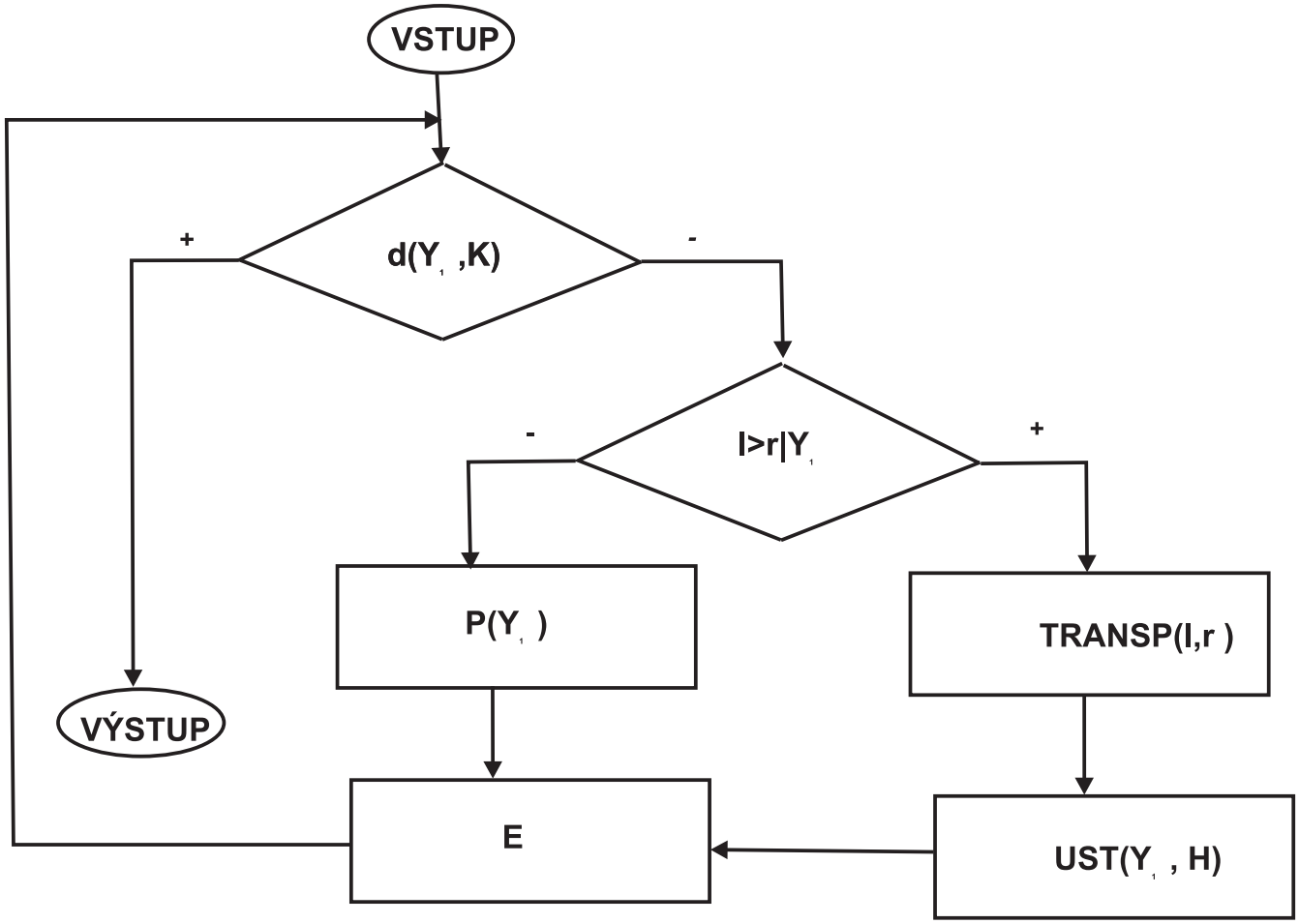
$$
\begin{aligned}
\text{BUBBLE} > \quad === \quad & /* \text{ Zníženie počtu zahniezdených cyklov:vzťahy:} \\
& \left\{[u]\left\{\left[u'\right] A\right\} * B\right\} = \\
& = \left\{[u]\left\{\left[u \vee u'\right] A\right\} * B\right\}, \\
& \left\{[u]\left\{\left[u \vee u'\right] A\right\} * B\right\} = \left\{[u] A * (\left[u'\right] B)\right\} * B \\
& \text{kde } u\text{- uzavretá podmienka, ktorá, ak raz začala platiť} \\
& \text{zostáva pravdivou až do zavŕšenia procesu} \\
& \text{spracovania dát (teória S-algebier- modifikovaná SAA} \\
& \text{s uzavretými logickými podmienkami, ktorá bola študovaná v [16]);} \\
& \text{ďalej pre stručnosť sa zavádza konštrukcia ´useknutého´}\mathbf{IF}\text{: } ([u]\,A) = ([u]\,A, E), \\
& \text{tu } E \text{ - je identický operátor } */ \\
= & \{[\Pi]\,([l > r|Y_1]\,TRANSP(l,r), E) * P(Y_1) * \\
& \quad *([d(Y_1,K)]\,UST(Y_1,H)) \\
& \quad) \\
& \} * \\
& *UST(Y_1,H) = \\
& \quad /* \text{ Odstránenie zátvoriek alternatívy } */ \\
= & \{[\Pi]\,([l > r|Y_1]\,TRANSP(l,r) * P(Y_1) * \\
& \quad *([d(Y_1,K)]\,UST(Y_1,H), P(Y_1) * \\
& \quad *([d(Y_1,K)]\,UST(Y_1,H)) \\
& \quad) \\
& \} * UST(Y_1,H) = \\
& \quad /* \text{ Prechod od m-fázového k 1-fázovému spracovaniu} \\
& \quad \text{postupnosti.} \\
& \quad \text{Zámena useknutého } \mathbf{IF} \text{ } ([d(Y_1,K)]\,UST(Y_1,H)) \text{ na operátor } UST(Y_1,H) \text{ po vetve } \textit{true}; \\
& \quad \text{odstránenie useknutého } \mathbf{IF} \text{ z vetvy } \textit{false} \text{ alternatívy; zámena predikátu } \Pi \text{ na } d(Y_1,K); \\
& \quad \text{odstránenie inštancie (výskytu) } UST(Y_1,H) \text{ po skončení základného cyklu. } */ \\
= & \{[d(Y_1,K)]\,([l > r|Y_1]\,TRANSP(l,r), E) * P(Y_1) * UST(Y_1,H), P(Y_1))\} = \\
& \quad /* \text{ odstránenie operátora posunu } P(Y_1) \text{ podľa vetvy } \textit{true} \text{ alternatívy} \\
& \quad \text{na základe vzťahu } P(Y_1) * UST(Y_1,H) = UST(Y_1,H) * / \\
= & \{[d(Y_1,K)]\,([l > r|Y_1]\,TRANSP(l,r) * UST(Y_1,H), P(Y_1))\} ==:: ROZTOK >
\end{aligned}
$$

Zvláštnosťou získaného algoritmu ROZTOK> je to, že jeho vykonanie sa realizuje v jednom jeho behu . V súlade s alternatívou, ktorá je vlastne telom základného cyklu, prostredníctvom skanovania indikátorom Y_1 v smere zľava doprava sa získa prvá neusporiadaná dvojica prvkov postupnosti. Potom, po ich transpozícii, Y_1 sa nastaví na značku H s opakovaným skanovaním tak, že objavený neusporiadaný prvok sa ´rozpúšťa´ v už utriedenej časti postupnosti atď. Proces sa končí po dosiahnutí indikátora Y_1 značky K. Graf-schéma algoitmu ROZTOK> je uvedená na obr. 5.1.

Platí

Lemma 5.2 *Schéma BUBBLE> je transformačne redukovateľná na schému ROZTOK>, t.j.*

$$BUBBLE >= (T_2) = ROZTOK >$$



Obrázok 5.1: Graf-schéma algoritmu ROZTOK>

, kde pod T_2 rozumieme množinu rovností, ktoré boli pri tejto transformácii použité.

□

Vzhľadom na reverzibilitu transformácií platí nasledujúce tvrdenie.

Dôsledok 5.2 *Schéma ROZTOK> je transformačne redukovateľná na schému BUBBLE>, t.j.*

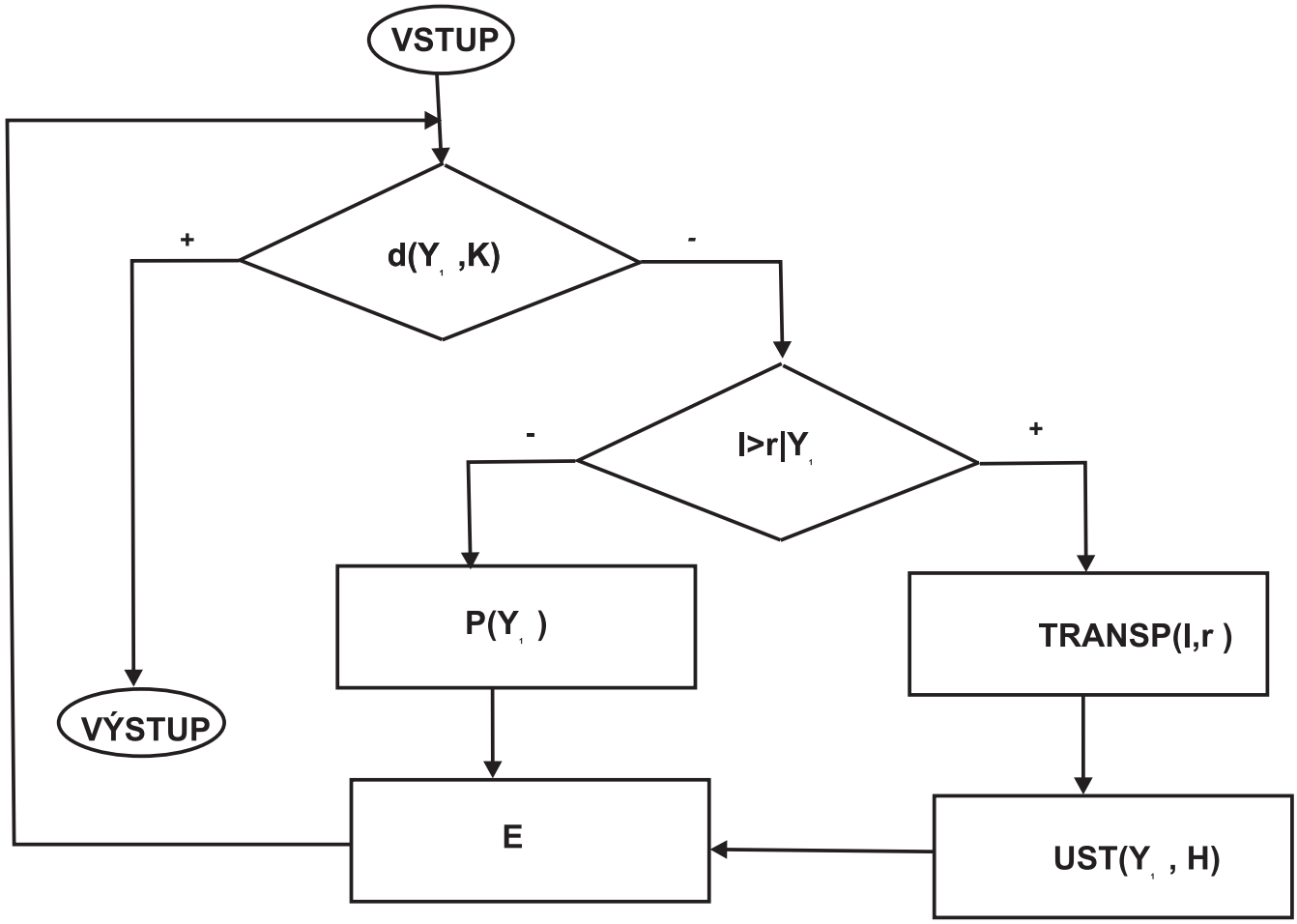
$$ROZTOK >= (T_2) = BUBBLE >$$

Okrem transformácií pri redukcii algoritmov je prípustné použitie aj ďalších pravidiel- konvolúcie, evolúcie, preorientácie, konkrétne preinterpretácie- zámeny bázových operátorov a/alebo predikátov schémy.

Analýza fungovnia algoritmu ROZTOK> odhaľuje možnosť prechodu k časovo efektívnejším (časovo menej náročným) modifikáciam algoritmu člnkového triedenia. Skutočne , ak namiesto umiestnenia indikátora Y_1 na marker **H** po každej transpozícii môže byť realizovaný posun $L(Y_1)$ za účelom priameho zaradenia neusporiadaného prvku na miesto. Teda preinterpretácia - spočívajúca v zámene operátora $UST(Y_1,H)$ na $L(Y_1)$ privádza k PC ČLNOKπ/a> (SHUTTLEπ/a>) v tvare:

$$\text{ROZTOK} >\uparrow\downarrow \text{ČLNOK}\pi/\text{a¿} \quad ::= \quad === \quad \{[d(Y_1,K)]\,([l > r|Y_1]\,TRANSP(l,r) * L(Y_1), P(Y_1))\}\,.$$

Graf-schéma algoritmu ČLNOKπ/a> je uvedená na obr. 5.2



Obrázok 5.2: Graf-schéma algoritmu ČLNOKπ/a>

Prostredníctvom metapravidla transformácie môže byť realizovaný prechod k PC ČLNOKπ/c> , pri ktorej sa v tele cyklu alternatíva pretvorí na kompozíciu cyklov. Prostredníctvom jedného cyklu sa vyhľadá prvá neusporiadaná dvojica prvkov a pomocou druhého sa vloží nájdený neusporidaný prvok na patričné (zodpovedajúce mu) miesto:

ČLNOKπ/a> === /* Transformácia sa realizuje pomocou rovnosti:
$\left\{[u]\,(\left[u'\right]A,B)\right\} =$
$= \left\{[u]\left\{\left[u \vee u'\right]B\right\} * \left\{\left[u \vee u'\right]A\right\}\right\}$, * /

$= \{[d(Y_1,K)]$ $\{[(l > r|Y_1) \vee d(Y_1,K)]\,P(Y_1)\} *$
$* \{[(l \leq r|Y_1) \vee d(Y_1,K)]\,TRANSP(l,r) * L(Y_1)\}$
$\} =$ / * Vylúčenie z podmienky druhého vnoreného cyklu s nulovou disjunktívnou zložkou
(pri posune doľava je d(Y_1,K)vždy nepravdivé) na základe rovnosti:
$u \vee 0 = u$ (pritom z pravdivosti d(Y_1,K) vyplýva pravdivosť $(l > r|Y_1)$), pretože $a < K$
pre každý prvok a utrieďovanej postupnosti);
vsunutie operátora E ako kompozičného súčiniteľa na koniec tela cyklu, na základe rovnosti $A = A * E$*/

$= \{[d(Y_1,K)]$ $\{[(l > r|Y_1) \vee d(Y_1,K)]\,P(Y_1)\} *$
$* \{[(l \leq r|Y_1)]\,TRANSP(l,r) * L(Y_1) * E\}$
$\} =::$ ČLNOKπ/c>

Lemma 5.3 *I. Schéma ROZTOK> sa dá zmenou interpretácie (preinterpretácia) previesť na ČLNOKπ/a>,*

a ROZTOK>↑↓ ČLNOKπ/a>;

II. Schéma ČLNOKπ/a> je transformačne redukovateľná na schému ČLNOKπ/c> , a ČLNOKπ/a>= (T_3) =ČLNOKπ/c>, kde T_3 je množina rovností, ktoré boli použité v priebehu transformácie.

Vzhľadom na reverzibilitu realizovaných transformácií platí aj nasledujúce tvrdenie .

Dôsledok 5.3 *I. Schéma ČLNOKπ/a> sa dá zmenou interpretácie (preinterpretácia) previesť na schému ROZTOK>, a ČLNOKπ/a>↑↓ ROZTOK> ;*

II. Schéma ČLNOKπ/c> je transformačne redukovateľná na schému ČLNOKπ/a> , a ČLNOKπ/c>= (T_3) =ČLNOKπ/a>, kde T_3 je množina rovností, ktoré boli použité v priebehu transformácie.

□

V procese redukcie jedného algoritmu na druhý algoritmus sa pripúšťa zavedenie do označenia postupnosti ďalších indikátorov (ukazovateľov, smerníkov) a/alebo markerov, vedúce k zodpovedajúcej modifikácii analyzovaných schém. Taká modifikácia je spojená so zovšeobecnením metapravidiel uvedených v kapitole 5.1. Také zovšeobecnenie je založené na použití konvolúcie a evolúcie pravidiel $F \rightarrow G$, alebo $p \Rightarrow p^{'}$, kde F,G a $p, p^{'}$ sú výskyty operátorových a predikátových termov v transformovanej schéme. Konkrétne to znamená, že v úlohe F a p môžu byť použité operátorové, resp. predikátové premenné (viď (Príklad **??**)). V rámci takého zovšeobecnenia je tiež možná modifikácia schém prostredníctvom zavedenia v nich doplňujúcich termov na základe evolúcie schém podľa pravidiel typu $E \rightarrow G$, kde E je identický operátor.

Zavedieme si teraz do pôvodného označenia postupnosti

$$M : HY_1a_1a_2...a_nK$$

doplňujúci indikátor Y_2 tak, že

$$M : HY_2Y_1a_1a_2...a_nK$$

Preinterpretácia algoritmu ČLNOKπ/c> pomocou nahradenia operátora E termom $\{[d(Y_2, Y_1)]\, P(Y_2)\}$ vyjadrujúcim stotožnenie pozícii indikátorov Y_2, Y_1 po zaradení ďalšieho neusporiadaného prvku privádza k nasledujúcej schéme:

$$\begin{aligned} \text{ČLNOK} > \quad &::= \\ &=== \quad \{[d(Y_1, K)]\, \{[(l > r | Y_2) \vee d(Y_1, K)]\, P(Y_2, Y_1)\} * \\ &\quad * \{[(l \leq r | Y_2)]\, TRANSP(l, r) * L(Y_2)\} \\ &\quad * \{[d(Y_2, Y_1)]\, P(Y_2)\} \\ \}\,. & \end{aligned}$$

Získaná schéma ČLNOK> sa odlišuje od schémy ČLNOKπ/c > v tom, že pomocou indikátora Y_1 sa fixuje miesto návratu indikátora Y_2 po zaradení na zodpovedajúce miesto ďalšieho v poradí neusporiadaného prvku, s pokračovaním procesu utriedovania na nasledujúcom behu vonkajšieho cyklu. Vyššia adaptívnosť schémy ČLNOK> v porovnaní so schémou ČLNOKπ/c > spočíva vo vylúčení porovnaní susedných prvkov pri návrate Y_2 k Y_1 po už utriedenému fragmentu postupnosti. Je potom platné toto tvrdenie.

Lemma 5.4 *Schéma ČLNOKπ/c > je redukovateľná na schému ČLNOK>, ČLNOKπ/c > $-* \rightarrow$ ČLNOK>, kde $A - -* \rightarrow A^{'}$ označuje redukovateľnosť schémy A na $A^{'}$ pomocou konvolúcie, evolúcie a ich zovšeobecnení.*

□

Ak vymeníme v procese redukcie konvolúciu s evolúciou prichádzame k tomuto tvrdeniu.

Dôsledok 5.4 *Schéma ČLNOK> je redukovateľná na schému ČLNOKπ/c > , ČLNOK> $-* \rightarrow$ ČLNOKπ/c >.*

□

Ak pokračujeme ďalej v analýze schémy ČLNOK zistíme, že počas spoločného prehliadania (skanovania) usporiadaného fragmentu postupnosti indikátormi Y_1, Y_2, s cieľom nájsť ďalší v poradí neusporiadaný prvok, pri každom prehliadanom prvku sa preverujú všetky tri podmienky schémy.

S cieľom zvýšiť adaptívnosť schémy ČLNOK> pokračujeme v transformácii tak, že minimalizujeme počet previerok podmienok na už usporiadaných fragmentoch postupnosti. Poznamenávame, že ak pri ďalšom kroku transformácie neuvádzame informáciu o používanej rovnosti, tak sa predpokladá, že takou rovnosťou je práve komentovaná rovnosť.

Zvýšenie adaptívnosti (adaptability) schémy ČLNOK> sa dosiahne pomocou nasledujúcich transformácií. Na začiatku realizujeme konvolúciu ČLNOK>$\uparrow I'$ na stratégiu Q, kde

$$I' : \{u_1 \Rightarrow d(Y_1, K), u_2 \Rightarrow l > r|Y_1, u_3 \Rightarrow d(Y_1, Y_2)\}$$

$$\begin{array}{lcl} A & \rightarrow & UST(HY_1, Y_2), \\ B & \rightarrow & P(Y_1, Y_2), \\ C & \rightarrow & TRANSP(l, r|Y_2) * L(Y_2), \\ D & \rightarrow & P(Y_2), \\ F & \rightarrow & ELIM(Y_1, Y_2). \end{array}$$

kde $UST(HY_1, Y_2)$ - nastavenie indikátorov Y_1, Y_2 bezprostredne napravo od markera **H** v utrieďovanej postupnosti **M**; $ELIM(Y_1, Y_2)$ - odstránenie indikátorov Y_1, Y_2 z **M**.

Napokon dostaneme stratégiu Q:

$$Q ::= A * \{[u_1]\,\{[u_1 \vee u_2]\,B\} * \{[\overline{u_2}]\,C\} * \{[u_3]\,D\}\} * F$$

Všimnime si, že v prosese vykonávania algoritmu ČLNOK> , ktorý zodpovedá danej formule, pri premiestňovaní indikátorov Y_1, Y_2 po už usporiadanej čsti postupnosti musia sa preverovať všetky tri podmienky pre každý prvok. V ďalšom je uvedená postupnosť rovností, po aplikácii ktorých k stratégii Q sa zníži počet previerok podmienok na utriedených fragmentoch utrieďovanej postupnosti:

$$\begin{array}{rcl} Q & = & /* \text{ V procese vykonávania prvého vnoreného cyklu môže byť dosiahnutý marker K.} \\ & & \text{Pre tento prípad zabezpečíme prechod na koniec cyklu} \\ & & \text{bez predbežnej previerky podmienok } u_2 \text{ a } u_3 * / \\ & = & A * \{[u_1]\,\{[u_1 \vee u_2]\,B\} * ([u_1]\,E, \{[\overline{u_2}]\,C * \{[u_3]\,D\}\})\} * F = \\ & = & / * \text{ Napokon, ak pri vykonávaní cyklu } \{[u_2 \vee u_1]\,B\} \text{ podmienka } u_2 \\ & & \text{sa stane pravdivou (pritom } u_1 \text{ je nepravdivá), potom možno hneď použiť operátor C} \\ & & \text{bez predbežnej previerky podmienky } u_2 * / \\ & = & A * \{[u_1]\,\{[u_1 \vee u_2]\,B\} * ([u_1]\,E, C * \{[\overline{u_2}]\,C\} * \{[u_3]\,D\})\} * F ::= \\ & ::= & Q' \end{array}$$

Ak teraz realizujeme evolúciu $Q' \downarrow I'$ dostaneme zdokonalenú schému ČLNOK/y> s vyšším stupňom adaptability v porovnaní so schémou ČLNOK>.

$$\begin{array}{rcl} Q' \downarrow I' & = & A * \{[d(Y_1, K)] \\ & & \quad \{[d(Y_1, K) \vee (l > r|Y_2)]\,P(Y_2, Y_1)\} * \\ & & \quad * ([d(Y_1, K)]\,E, \\ & & \qquad TRANSP(l, r|Y_2) * L(Y_1) * \\ & & \qquad * \{[l \leq r|Y_2]\,TRANSP(l, r|Y_2) * L(Y_2)\} * \\ & & \qquad * \{[d(Y_2, Y_1)]\,P(Y_2)\} \\ & & \quad) \\ \} * F & ::= & \text{ČLNOK/y} > \end{array}$$

Prichádzame k týmto tvrdeniam:

Lemma 5.5

Schéma ČLNOK> je transformačne redukovateľná na na schému ČLNOK/y> a

$$\check{C}LNOK > - - * \rightarrow \check{C}LNOK/y >$$

□

Ak zameníme v procese redukcie konvolúciu na evolúciu a naopak dostávame nasledujúce tvrdenie:

Dôsledok 5.5 *Schéma ČLNOK/y> je transformačne redukovateľná na na schému ČLNOK> a*

$$\check{C}LNOK/y > - - * \rightarrow \check{C}LNOK >$$

Týmto spôsobom prichádzame k vylepšenému algoritmu (z hľadiska počtu testov podmienok)ČLNOK/y>. V tomto algoritme pri prechádzaní (skanovaní) usporiadanými fragmentami postupnosti sa vykonávajú iba testy dvovh podmienok: $d(Y_1, K)$ a $l > r|Y_2$, zatiaľ čo v schéme ČLNOK> na týchto fragmenroch sa testujú všetky tri podmienky.

Pokračujúc v ceste zvyšovania adaptability procesu triedenia do úrovne usporiadanosti spracovávaných postupností na základe transformácií, ktoré boli vyvinuté v [17], dostávame schému triedenia CAB (triedenie systémom alternatívnych vkladaní), ktorá je efektívnejšia z hľadiska času (nižšia časová zložitosť) v porovnaní so schémou ČLNOK> a jej doteraz analyzovanými modifikáciami.

$$\begin{aligned}
CAB \quad ::= & \\
= \quad & A * \{[d(Y_1, K)] \\
& \quad \{[d(Y_1, K) \vee (l > r|Y_1)]\, P(Y_1)\} * \\
& \quad * ([(l < r|Y_1) \vee d(Y_1, K)] \\
& \qquad ([l|Y_2 < r|Y_1]\, \{[r|Y_2 > r|Y_1]\, P(Y_2)\}, \\
& \qquad \{[l|Y_2 < r|Y_1]\, L(Y_2\}) * \\
& \qquad * INSERTr|Y_1 POY_2 \\
& \quad \} \\
& \} * F.
\end{aligned}$$

V súlade s uvedenou schémou pomocou indikátora Y_1 sa postupne zafixujú neusporiadané prvky v postupnosti **M**, zatiaľ čo pri presune indikátora Y_2 po už usporiadanom fragmente postupnosti v požadovanom smere sa hľadá miesto, na ktoré bude umiestnený aktuálny nezaradený prvok $r|Y_1$.

K hlbšiemu pochopeniu stupňa efektívnosti schémy CAB napomôže prepojenie daného algoritmu s ďalšími známymi algoritmami vyhľadávania a triedenia, ktoré je založené na metóde preinterpretácie.

Príklad 5.6

Pozrime sa na problém vyhľadávania v slovníku D (konkrétne ide o anglicko-slovenský) hodnôt zoznamu C neznámych anglických slov, ktoré sa vyskytli pri preklade niektorého textu z angličtiny do slovenčiny. V úlohe označenej postupnosti dát budeme pracovať so súborom

$$\Phi : HY_2 s_1 s_2 ... s_n Y_1 w_1 w_2 ... w_s K$$

kde $s_1 s_2 ... s_n$ sú strany v slovníku D, ktoré obsahujú lexiko-graficky usporiadané anglické slová, spolu s ich slovenskými ekvivalentami; $C : w_1 w_2 ... w_s$ - zoznam anglických slov, ktoré je treba ešte preložiť. V počiatočnom stave Y_2 ´vidí´ prvé slovo na strane s_1 a Y_1 ´vidí´ prvé slovo w_1 v zozname C. Pomocou premiestňovania Y_2 v slovníku zľava doprava sa hľadá slovo w_1 , ktoré sa nachádza na strane s_i.

Potom sa Y_1 premiestni o jeden prvok doprava na slovo w_2. Vyhľadanie daného slova v slovníku D sa uskutoční listovaním strán, počínajúc stranou s_i, doľava, alebo doprava v závislosti od výsledku porovnania slova w_2 so slovom $l|Y_2$, ktoré bolo nájdené v predchádzajúcom kroku. Ďalej sa hľadanie prekladu ďalšieho v poradí slova w_3 realizuje analogicky v závislosti od polohy Y_2 v slovníku po skončení prekladu slova w_2 atď., až do momentu, kedy Y_1 dosiahne marker **K**, ktorý označuje koniec zoznamu C. Uvedený príklad potvrdzuje opodstatnenosť procedúry vyhľadania miesta , na ktoré bude umiestnený v poradí ďalší neusporiadaný (nezaradený) prvok ; na tom je založená schéma CAB>.

□
(Koniec príkladu)

Použitím inej preinterpretácie schémy CAB> môže byť získaný známy algoritmus binárneho triedenia BIS>, ktorý je založený na procedúre hľadania miesta vkladania ďalšieho v poradí zatrieďovaného prvku, prostredníctvom lokalizácie takého miesta podľa mediánu ďalšieho intervalu, obsahujúceho hľadané miesto vkladania. V tomto algoritme sa pohyby $L(Y_2)$ a $P(Y_2)$ interpretujú ako operátory polenia aktuálneho intervalu. Definovanie ďalšieho intervalu je výsledkom previerky podmienok nerovností, zodpovedajúcich vloenej alternatíve a jej cyklických vetiev.

Je zaujímavé, že v schéme CAB> bola použitá koncepcia, ktorá sa pozdejšie stala základom metódy adaptívneho triedenia SKIPSORT . Táto metóda sa realizuje na ADT zoznam (list), a je orientovaná na rýchle lokálne vkladania a je príbuzná s metódami na vyhľadávacích stromoch. ktoré majú vo vrcholoch informáciu o kľúčoch a prioritách. Bolo preukázané, že podľa základných kritérií adaptability je SKIPSORT efektívnejší v porovnaní s obyčajnými vkladaniami (inzerciami). Simulácie potvrdzujú, že SKIPSORT je 2-krát rýchlejšia metóda triedenia ako metódy ČLNOK. Metóda SKIPSORT a jej modifikácie , ktoré sú založené na vyhľadávacích stromoch, môžu byť získané zo schémy CAB> pomocou zodpovedajúcich interpretácií. Platia nasledujúce tvrdenia.

1. Schéma ČLNOK/y> je transformačne redukovateľná na schému CAB>,t.j. ČLNOK/y>$-*\rightarrow$CAB>.
2. Pri zmene smeru použitých rovností v uvedenej dostávame, že schéma CAB> je transformačne redukovateľná na schému ČLNOK/y>,t.j. ČLNOK/y> $-*\rightarrow$ ČLNOK/y>.
3. Na základe doteraz uvedených tvrdení máme, že schéma BUBBLE(k)> je transformačne redukovateľná na schému CAB>,t.j. BUBBLE(k)> $-*\rightarrow$ CAB>.
4. Pri zámene, v procese transformácie, smeru použitých rovností, metapravidiel konvolúcie na evolúciu a naopak, dostaneme , že schéma CAB> je transformačne redukovateľná na schému BUBBLE(k)> ,t.j. CAB> $-*\rightarrow$ BUBBLE(k)> .
5. Na základe uvedených tvrdení dostávame , že schéma BUBBLE> je transformačne redukovateľná na schému CAB> ,t.j. BUBBLE> $-*\rightarrow$ CAB> .
6. Na základe uvedených tvrdení dostávame , že schéma BUBBLE> je transformačne redukovateľná na schému CAB> ,t.j. BUBBLE> $-*\rightarrow$ CAB> .

5.3 Klasifikácia stratégií spracovania symbolov a reprezentácia algoritmických znalostí

V tejto časti privedieme klasifikáciu adaptívnych algoritmov sekvenšného triedenia a s nimi asociovaných stratégií spracovania symbolov. Tvorba takých stratégií sa uskutoční po analógii s konštrukciou stratégie CP(k) z kapitoly 5.1 pomocou metapravidla konvolúcie. Poznamenávame, že vo všeobecnosti sa s každým algoritmom asociuje niekoľko stratégií. Teda, tvorba interpretácie **I**, podľa ktorej sa realizuje konvolúcia, nie je jednoznačnou úlohou.

Algoritmy, ktoré boli uvedené v kapitole 5.2 patria medzi tzv. *ľavostranné*,t.j. že triedenie realizované podľa nich spočíva vo formovaní fragmentu utriedenej postupnosti v smere zľava doprava, od markera H

k markeru K.

Dá sa v tejto súvislosti sformulovať *princíp duality*, podľa ktorého je možné zostrojiť na základe algoritmu ľavostranného triedenia jeho pravostranný analóg a naopak.

Princíp duality. Nech $\overrightarrow{F}$ je schéma reprezentujúca ľavostranné triedenie. Duálna schéma $\overleftarrow{F}$ môže byť zo schémy $\overrightarrow{F}$ vytvorená v dôsledku zámeny:

1. počiatočného označenia postupnosti M_0 na duálne označenie, v ktorom indikátory susedné s markerom **H** sa umiestnia ako susedné k markeru **K** a naopak;
2. všetkýsh výskytov markera **H** v schéme $\overrightarrow{F}$ za marker **K** a naopak;
3. všetkýsh výskytov premenných l za r a naopak (okrem podmienok, v ktorých sa usporiadanosť prvkov preveruje podľa niektorého indikátora);
4. všetkýsh výskytov operátorov posuvu **L** na **R** a naopak .

Príklad 5.7

Vytvoríme s použitím princípu duality schému CAB< podľa schémy CAB> (viď časť 5.2). Počiatočné označenie postupnosti $M_0^{'}$, ktorá bude spracovávaná schémou CAB< má tvar:$M_0^{'}$: $Ha_1a_2...a_nY_1Y_2K$. Potom je schéma CAB< reprezentovaná takto:

$$
\begin{array}{ll}
\text{CAB} <::= & A^{'}* \\
& *\{[d(Y_1,H)]\,\{[d(Y_1,H) \vee (l > r|Y_1)]\,L(Y_1)\}* \\
& *\{[d(Y_1,H) \vee (l > r|Y_1)] \\
& ([r|Y_2 < l|Y_1]\,\{[l|Y_2 < l|Y_1]\,L(Y_2)\}\,, \\
& \{[r|Y_2 < l|Y_1]\,P(Y_2\})) * \\
& \text{INSERT } l|Y_1\text{PO}Y_2 \\
& \} \\
& \} * F
\end{array}
$$

□
(Koniec príkladu)

kde $A^{'}$ operátor počiatočného nastavenia indikátorov v súlade s označením $M_0^{'}$.

V súlade s uvedeným princípom duality a v dôsledku platnosti tvrdení v bodoch 3-5, platí táto veta:

Veta 5.1

Pre nasledujúce pravostranné schémy platia tieto redukcie:

1. Schéma BUBBLE(k)< je redukovateľná na schému CAB< , t.j. BUBBLE(k)< $-* \rightarrow$ CAB<;
2. Schéma BUBBLE< je redukovateľná na schému CAB< , t.j. BUBBLE< $-* \rightarrow$ CAB<;;
3. Schéma BUBBLE(k)< je redukovateľná na schému ČLNOK(k)< , t.j. BUBBLE(k)< $-* \rightarrow$ ČLNOK(k)<, a posledná je redukovateľná na schému CAB(k)<, t.j. BUBBLE(k)< $-* \rightarrow$ ČLNOK(k)<$-* \rightarrow$ CAB(k)<, kde schémy CAB(k)<,BUBBLE(k)< ,ČLNOK(k)< sú duálne k schémam CAB(k)>,BUBBLE(k)> ,ČLNOK(k)>

□

V súlade s vetou teda adaptívnym algoritmom ľavostranného triedenia zodpovedajú duálne pravostranné algoritmy a naopak. Kombinácia týchto dvoch prístupov je základom rodiny algoritmov triedenia **MK**, tzv. *kyvadlových* algoritmov triedenia, ktoré boli zavedené v [17]. Budeme rozlišovať tzv. *kmitavé* **KM** $\subset$**MK** a *ústretové* kyvadlá.

Príklad 5.8

Do triedy **KM** patrí schéma SHAKER, ktorá reprezentuje známy algoritmus triedenia pretriasaním (shaking) [12]:
v ďalšom **YM** je predikát pravdivý práve vtedy, keď je označená postupnosť

$$M_0 : HY_1a_1a_2...a_nK$$

utriedená;

$$\begin{aligned} \text{SHAKER} ::= &\quad \{[YM]\,T(BUBBLE>) * ([YM]\,E, T(BUBBLE<))\} \\ \text{T(BUBBLE>)} ::= &\quad \{[d(Y_1,K)]\,([l > r|Y_1]\,TRANSP(l,r|Y_1),E) * P(Y_1)\} \\ \text{T(BUBBLE<)} ::= &\quad \{[d(Y_1,H)]\,([l > r|Y_1]\,TRANSP(l,r|Y_1),E) * L(Y_1)\} \end{aligned}$$

□
(Koniec príkladu)

Poľa tejto schémy ľavostranné skanovanie (prehliadanie)T(BUBBLE>) postupnosti od markera H k markeru K sa strieda s pravostranným od markera K k markeru H. Pritom počas ľavostranného skanovania ´prebublávajú´ maximálne prvky a počas pravostranného skanovania sa zaraďujú na zodpovedajúce miesta (´rozpúšťajú sa´) minimálne prvky.

Pre algoritmus ústretového kyvadla je charakteristické striedanie krokov ľavostranného utrieďovania (zaradenie na zodpovedajúce miesto ďalšieho v poradí nezaradeného prvku v smere zľava doprava) s krokom pravostranného utrieďovania (zaradenie na zodpovedajúce miesto ďalšieho v poradí nezaradeného prvku v smere zprava doľava). Významnou podtriedou je tzv. *voľné*ústretové kyvadlo CBM $\subset$ BM, v ktorom ústretové kroky sú informačne nezávislé.

Príklad 5.9

Uvádzame teraz schému CAB><$\in CBM$:

$$CAB >< ::= \{[d(Y_1,Y_3)]\,T(CAB >) * ([d(Y_1,Y_3)]\,E, T(CAB <))\}$$

kde

- d(Y_1, Y_3) je podmienka pravdivá k došlo k splynutiu indikátorov Y_1, Y_3 v procese usporiadavania postupnosti;
- počiatočné označenie postupnosti má tvar M_0:$HY_2Y_1a_1a_2...a_nY_3Y_4K$;
- T(CAB>),T(CAB<) telá vonkajších cyklov duálnych schém CAB> a CAB<;

□
(Koniec príkladu)

Druhou podtriedou BM - spriahnuté ústretové kyvadlo BBM$\subset$ BM. Pri BBM sa ústretové kroky informaene závislé- kazdý krok sa môze zaea? az potom, ako získal informáciu o zavasení ústretového

kroku. Charakteristickým príkladom BBM je známy algoritmus triedenia tzv. QUICKSORT (autor C.A.R.Hoare).

Nech F∈ MK a F je schéma reprezentujúca kyvadlové triedenie a ktorá obsahuje v úlohe subschém komponenty, na ktoré sú aplikovateľné redukcie, ktoré sme uviedli vyššie; potom platí

Veta 5.2

Platí, že pre schému F sa uplatňuje redukovateľnosť na schému F/y ∈ MK, t.j. F$-*\rightarrow$F/y, kde F/y má vyšší stupeň adaptability ako F.

□

Z doteraz uvedených výsledkov a zo známeho prepojenia medzi algoritmami triedenia a vyhľadávania plynú tieto závery:

1. Pre ľubovoľne zvolený algoritmus triedenia AT∈ SORT existuje interpretácia **I**, podľa ktorej môže byť zostrojená uplatnením konvolúcie s AT asociovaná stratégia ST ∈ S a t.ž. ST=AT↑ I, kde S je rodina stratégií asociovaná s triedou SORT algoritmov triedenia.
2. Pre ľubovoľne zvolenú stratégiu ST ∈ S existuje interpretácia **I** taká, že AT=ST↓ I, kde AT∈ SORT je niektorý algoritmus triedenia.
3. Pre každú stratégiu ST ∈ S existuje interpretácia $\mathbf{I}'$ taká, že AV=ST↓ I', kde AV je algoritmus vyhľadávania asociovaný so stratégiou ST.
4. Pre ľubovoľne zvolený algoritmus triedenia AT∈ SORT môže byť vytvorený algoritmus vyhľadávania AV taký, že AS↑↓ AV a naopak.

5.4 Regulárne schémy a abstraktné typy pamäti

Predpokladajme, že sme zostrojili algoritmickú reprezentáciu (opis) vybratej problémovej oblasti v tvare algoritmickej algebry, ktorá má definovanú zodpovedajúcu bázu **Z**. Ďalším krokom je modelovanie (implementácia) takej reprezentácie . K realizácii takého kroku slúži (okrem koncepcie ADT, ktorá je súčasťou algoritmickej reprezentácie) koncepcia *abstraktného pamäťového typu*-APT.

APT sa definuje ako dvojica <***médium (nosič);metódy (prostriedky) prístupu k médiu (nosičovi)***>.

Nech teraz F(Z) je regulárna schéma (PC) nad bázou Z, ktorá pozostáva z operátorov a predikátov, ktoré odrážajú špecifiká príslušnej predmetnej oblasti (napr. triedenie). Tieto operátory a predikáty sú definované na zoskupení spracovávaných datových typov. Tak potom niektoré z týchto operátorov a predikátov budú patriť do signatúry niektorého MAS, ktorá sa používa k formálnemu vyjadreniu (reprezentácii) ADT. Je potrebné zdôrazniť, že prístup k jednotlivým prvkom (druhom) ADT a zmena ich hodnôt sa dá realizovať výlučne iba pomocou operátorov a predikátov, ktoré sú zo signatúry ADT. Toto jje vlastnosť tzv. *inkapsulácie*(zapúzdrenia) ADT a preto ADT je treba vnímať ako niektoré *púzdro*(kapsulu).

Na ADT možno nazerať ako na niektorý hypotetický (abstraktný) stroj -HM, ktorý funguje podľa známej schémy dvoch kooperujúcich strojov: riadiaceho- M_R a operačného- M_O [35]. Prístup k stavom stroja M_O a ich zmene sa realizuje prostredníctvom zložiek signatúry ADT, ktoré sa považujú za inštrukcie stroja HM. Tieto inštrukcie sa reprezentujú pomocou PC typu F(Z): v termínoch bázy týchto schém (termov) sa realizuje kooperácia M_R a M_O- riadiacej a operačnej štruktúry HM. Pritom PC F(Z) vystupujú v úlohe *mikroprogramov* a prvky bázy Z- v úlohe mikroinštrukcií.

Ďalšia formalizácia koncepcie HM sa realizuje prostredníctvom APT. Vyššie sme uviedli APT ako dvojzložkovú entitu-

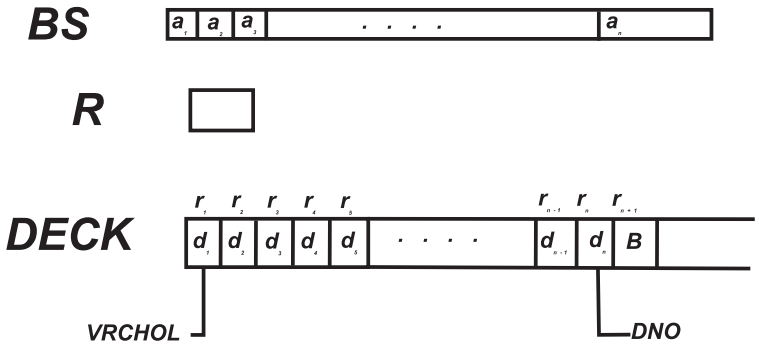
< médium (nosič);metódy (prostriedky) prístupu k médiu (nosičovi) >

Médium sa dá predstaviť vo všeobecnom prípade ako m-rozmerná štruktúra pozostávajúca z *buniek* spoplu s metódami prístupu k médiu [36]. Rozmiestnenie prvkov ADT na médiach APT zabezpečuje možnosť realizácie programu HM, reprezentovaného PC nad signatúrou ADT.

V termínoch APT možno reprezentovať rôzne štruktúry pamäti, napr. také, ako sú známe páskové (reťazcové) štruktúry - zoznam, zásobník, pole, záznam, počítadlo, front atp.

Teraz budeme ilustrovať proces detailizácie algoritmickej špecifikácie pomocou jej reprezentácie ako PC nad APT na príklade algoritmov triedenia ČLNOK> a CAB>.

V úlohe APT pre HM vyberieme štruktúru, ktorá je vyobrazená na obr. 5.3.



Obrázok 5.3: APT stroja HM

kde BS (bobsleigh) je vstupný front (FIFO pamäť); R - bunka (ako v T.s.); DECK je údajová štruktúra , ktorá má vlastnosti pd-zásobníka (pd-stack) a frontu; niektorí autori [9] hovoria o špeciálnom zozname, ktorý kombinuje vlastnosti frontu (queue) a stacku a dali mu názov *dequeue*, zjednodušene *deck*.

V BS je uložená utriedťovaná postupnosť. Deck možno interpretovať ako semiinfinitnú (nekonečnú doprava) lineárnu pamäť, ktorá obsahuje dáta súvisle uložené v bunkách $r_1, r_2, ..., r_n$. Prvá zľava bunka r_1 sa volá *vrchol decku*, zatiaľ čo prvá prázdna bunka nasledujúca za r_n sa volá *dno (decku)*. Deck umožňuje čítanie(R) a zápis (W) na obidvoch koncoch s nasledovným režimom:

- R/W v bunke r_1 sa dá vyjadriť pomocou nasledovných operácií:
  - **Zápis-W v bunke r_1 (vrchol decku)**: *Push(x,D)*- $\boldsymbol{x}$ sa stáva novým vrcholom decku D a starý i. prvok v D sa stáva i+1. prvkom v (novej inštancii) D ;
  - **Čítanie-R v bunke r_1 (vrchol decku)**:*Pop(D)*- odstraňuje a vracia vrchol D s tým, že starý i. prvok v D sa stáva i-1. prvkom v (novej inštancii) D (d_2 sa stáva novým vrcholom v (novej inštancii) decku D);
- R/W na pravej strane decku sa realizuje podobne ako v zásobníku; W sa realizuje do bunky r_{n+1} (1.prvá prázdna bunka zprava) ;R sa realizuje z bunky r_n , presnejšie odstraňuje a vracia prvok z r_n a posúva doľava pozíciu dna decku. dá sa to vyjadriť pomocou nasledovných operácií:
  - **Zápis-W na dno decku**: *Inject(x,D)*- pridáva $\boldsymbol{x}$ ako nový posledný neprázdny symbol-zapisuje x na dno decku D (r_{n+2} sa stáva novým dnom) ;

– **Čítanie-R z dna decku**:*Eject(D)*- odstraňuje a vracia posledný neprázdny prvok decku D; ak je deck D prázdny, potom ERROR.

Zavedieme si najprv niektoré označenia: nech

- P,Q sú premenné s oborom {**BS,R, DECK**};
- r- prvá zľava bunka v BS -vrchol (top) BS; ak $B \in$ **BS** potom r= B[1]=B.top;
- r_1, r_{n+1}- vrchol(top) resp. dno (bottom) decku. Ak D$\in$ **DECK**, potom budeme písať $D\,[r_1]$ (alebo alternatívne $D.top$), resp. $D\,[r_{n+1}]$ (alebo alternatívne $D.dno$);
- r_2- susedná bunka s r_1;
- ϕ- prázdny obsah bunky v ktorejkoľvek pamäti.

Definujeme ďalej operácie:

$$\begin{array}{rcl} PZ(P,Q) & \equiv & Q.top \leftarrow P.top \text{ kopírovanie, resp. prepis v pamäti;} \\ PZ_1(D) & \equiv & D.dno \leftarrow D.top \text{ kopírovanie, resp. prepis z vrcholu decku D na dno decku D;} \\ PZ_2(D) & \equiv & D.top \leftarrow\downarrow D.dno \text{ kopírovanie, resp. prepis z ´dna´ D na vrchol decku D.} \\ & & \text{Symbol } \downarrow D.dno \text{ označuje poslednú neprázdnu bunku decku D .} \end{array}$$

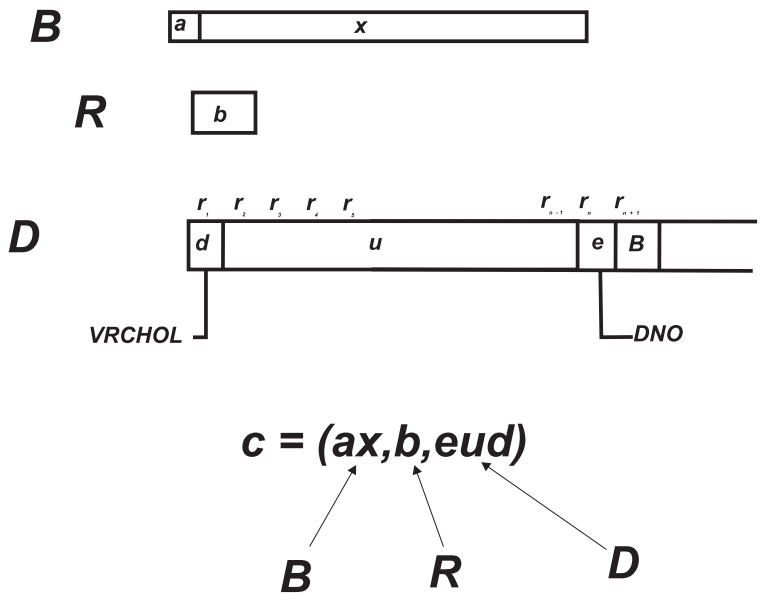
Operácie PZ a PZ_i, i=1,2, sú odvodené z operácií signatúr ADT typu APT.Skutočne, vyjadríme teraz operáciu PZ:nech B $\in$**BS**, D $\in$**DECK**, R $\in$ **R**.

$$\begin{array}{rcl} PZ(B,D) & = & (PUSH(B.top,D); DEQUE(B)) \\ PZ_1(D) & = & R \leftarrow POP(D); INJECT(R,D); \\ PZ_2(D) & = & R \leftarrow EJECT(D); PUSH(R,D). \end{array}$$

Charakterizujeme si teraz HM *konfiguráciou*, ktorú definujeme takto:

$$c = (ax, b, cud)$$

Situáciu v HM ilustruje obr. 5.4



Obrázok 5.4: Konfigurácia stroja HM

Ilustrujeme výpočtové kroky pre jednotlivé operácie v HM.

$$c = (ax, b, eud) \overset{PZ(B,D)}{\rightarrow} (x, b, euda)$$

Teraz sa pozrieme, čo sa stane po aplikácii sekvencie

$$PZ(D,R) * PZ_1(D) * PZ(R,D)$$

$$c = (ax, b, eufd) \overset{PZ(D,R)}{\rightarrow} (ax, d, euf) \overset{PZ_1(D)}{\rightarrow} (ax, d, feud) \overset{PZ(R,D)}{\rightarrow} (ax, \phi, feud)$$

Napokon uvedieme PC pre niektoré algoritmy realizovanú na APT a stroji HM.

$$\begin{array}{rl} \text{ČLNOK} > (B,D,R) & ::= \\ & \quad \{[B=\phi]\, PZ(B,D) * \\ & \qquad * \{[r_1 \le r_2]\, PZ(D,R) * PZ_1(D) * PZ(R,D) * \\ & \qquad * \{[\downarrow r = \#]\, PZ_2(D)\} \\ & \} \end{array}$$

Operátory PZ(B,D), PZ(D,R)* PZ_1(D)* PZ(R,D) a PZ_2(D) modelujú zodpovedajúco operátory $P(Y_1,Y_2)$, TRANSP(l,r|Y_2)*$L(Y_2)$, $P(Y_2)$ a podmienky $B = \phi, r_1 < r_2, r = \#$ zodpovedajúco podmienky $d(Y_1, K), l \le r|Y_2, d(Y_1, Y_2)$ v schéme ČLNOK>.

Uvedené operátory a podmienky možno vnímať ako príkazy (inštrukcie) stroja HM, ktorá realizuje triedu algoritmov triiedenia vrátane triedenia ČLNOK>.

Uvedieme teraz realizáciu algoritmu CAB> ako PC nad APT.

$$\begin{array}{rcl} \text{CAB} > (B,D,R) & ::= & \\ & = & PZ(B,R) * \{[KC]\, \{[KC \vee H\Pi]\, PZ(R,r_1) * PZ(B,R)\} * \\ & & \quad * \left\{[KC \vee Y\Pi]\, \left([H\Pi]\, \left\{\left[\overline{\Pi\Pi}\right]\, PZ_1(D)\right\}\right.\right. , \\ & & \qquad \{[Y\Pi]\, PZ_2(D)\} \\ & & \qquad\qquad) \\ & & \qquad * PZ(R,r_1) * PZ(B,R) \\ & & \qquad \} \\ & & \} . \end{array}$$

kde

$$\begin{array}{rcl} KC & = & (B = \phi) \wedge (R = \phi) \\ H\Pi & = & R < r_1 \\ Y\Pi & = & \overline{H\Pi} \wedge ((R \le \downarrow r) \vee (\downarrow r = \#)) \end{array}$$

Uvedená PC nad APT $CAB > (B,D,R)$ modeluje PC $CAB >$ takto:

$$\begin{array}{rcl} A & ::= & PZ(B,R); \\ d(Y_1,K) & ::= & KC = (B = \phi) \wedge (R = \phi); \\ l > r|Y_1 & ::= & H\Pi = R < r_1; \\ P(Y_1) & ::= & PZ(R,r_1) * PZ(B,R); \\ l \le r|Y_1 & ::= & Y\Pi = \overline{H\Pi} \wedge ((\downarrow r > R) \vee (\downarrow r = \#R)); \\ l|Y_2 < r|Y_1 & ::= & H\Pi; \\ r|Y_2 > r|Y_1 & ::= & \overline{H\Pi}; \\ P(Y_2) & ::= & PZ_1(D); \\ r|Y_2 < r|Y_1 & ::= & Y\Pi; \\ L(Y_2) & ::= & PZ_2(D); \\ INSERTr|y_1POY_2 & ::= & PZ(B,R) * PZ(R,r_1). \end{array}$$

Z hľadiska obsahového činnosť PC $CAB > (B,D,R)$ spočíva v nasledovnom:

1. Na začiatku sa realizuje prepis prvkov usporadúvanej postupnosti z B cez bunku R na vrchol r_1 decku až do doby, keď narazíme na prvú neusporiadanú dvojicu.

2. Po odhalení takej dvojice, po výstupe z prvého vnoreného cyklu (podmienka $KC \vee H\Pi$ platí)riadenie sa opdovzdá druhému vnorenému cyklu, ktorý pracuje s objavenými neusporiadanými prvkami a zaraďuje ich na vhodné miesta, pokým nenastane platnosť podmienky $KC \vee Y\Pi$

3. Ak je front neprázdny, do deku sa prenáša ďalší fragment postupnosti, pre ktorý je nájdené vhodné miesto, až do objavenia sa ďalšej neusporiadanej dvojice atď.

4. Schéma ukončí činnosť po vyprázdnení frontu B a bunky R ; všimnime si, prepis z prázdnej B do R spôsobí výmaz R.

Je dôležité vyzdvihnúť to, že detailizácia algoritmickej špecifikácie prostredníctvom PC nad APT je orientovaná tak na programovú, ako aj na hardvéropvú realizáciu navrhovaných algoritmov. Schémy PC $CAB > (B, D, R)$ a ČLNOK> (B, D, R) sú realizované ako HW zariadenia (HW triedičky), ktoré sú predmetom patentov [14] .

Literatúra

[1] CHYTIL, M.: Teórie automatů a formálních jazyků. SPN Praha, 1978.

[2] CHYTIL, M.: Automaty a gramatiky. SNTL Praha, 1981.

[3] MOLNÁR,Ľ.,ČEŠKA,M.,MELICHAR,B.: Gramatiky a jazyky. Alfa Bratislava, 1987.

[4] MINSKY,M.L.: Computation: Finite and Infinite Machines. Prentice Hall Inc. 1967.

[5] MANNA,Z.: Mathematical Theory of Computations. McGraw-Hill, 1974 (český preklad:MANNA,Z.: Matematická teórie programů. SNTL Praha 1981)

[6] GINSBURG,S.:The mathematical theory of context-free languages.1.vyd.New York, McGraw-Hill 1966.

[7] AHO,A.V.,HOPCROFT,J.E., ULLMAN,J.D.: The Design and Analysis of Computer Algorithms. Addison Wesley, 1974.

[8] BROOKS,F.P.:Ako s projektujú a vytvárajú programové komplexy (v ruštine). Moskva, Nuka, 1979, 150str.

[9] HEILEMAN,G.L.: Data Structures, Algorithms and Object-oriented Programming. McGraw-Hill, 1996.

[10] JABLONSKIJ, S.V.: Úvod do diskrétnej matematiky. Alfa Bratislava 1984.

[11] Post,E.: práca o úplnosti????

[12] KNUTH,D.E.: The Art of Computer Programming. vol. 1-3.Addison Wesley, 1969.

[13] STONE,H.: Discrete Mathematical Structures and Their Applications.SRA Science Research Associates Inc.1973.

[14] TSEYTLIN, G.E.: Úvod do algoritmiky (v ruštine).Medzinárodná Šalamúnova Univerzita, Kyjev, vydavateľstvo Sfera, Kyjev 1998, ISBN 960-7267-14-8.

[15] TSEYTLIN, G.E.: Problema tožděstvennych preobrazovanij schem strukturirovannych programm s zamknutymi logičeskimi uslovjami. Č.1-3. Kibernetika-1978, No.3,s.50-57, No.4,s.10-18, No.5,s.44-51.(v ruštine)

[16] TSEYTLIN, G.E.: Formálne aspekty štrukturovaného programovania s GO TO(v ruštine). Programmirovanije, 1984. No.1.

[17] TSEYTLIN, G.E.: Projektovanie sekvenčných algoritmov triedenia: klasifikácia, transformácia, syntéza (v ruštine). Programmirovanije, 1989. No.3,s.3-24.

[18] MANNILA,H : MEASURES OF PRESORTEDNESS AND OPTIMAL SORTING ALGORITHMS. IEEE Trans.Comput. vol.34,318-325.

[19] TSEYTLIN, G.E.: formal Transformation in three-valued logic of structural programming. Proc. of 12th Intern. Symp. on Multivalued Logic. Paris 1982. pp.78-89.

[20] TSEYTLIN, G.E.: Projektovanie algoritmov paralelného triedenia (v ruštine). Programmirovanije, 1989.No.6,s.4-19.

[21] TSEYTLIN, G.E.: Rozparalelňovanie algoritmov triedenia (v ruštine). Kibernetika, 1989.No.6,s.67-74.

[22] TSEYTLIN, G.E.: Vyhľadávanie a triedenie: klasifikácia, transformácia, syntéza.I,II . Avtomatika i telemechanika, 1992.No.4,s.147-154. No.5,s.156-165.

[23] JUŠČENKO,E.L.,TSEYTLIN, G.E.,GALUšKA, A.V.:Algebro-gramatické syntéza štrukturovaných schém programov (v ruštine). Kibernetika , 1989, No.6, str.5-16.

[24] TSEYTLIN, G.E.: Teoretické aplikačné aspekty algoritmiky(v ruštine).USiM,1995, No.1, str.3-13.

[25] TSEYTLIN, G.E.,JUŠČENKO,E.L.:Formalizované špecifikácie transformačná syntéza programov (v ruštine). Kibernetika i sistemnyj analiz, 1993, No.1, str.127-152.

[26] TSEYTLIN, G.E.:Kritérium funkcionálnej úplnosti v algebre Dijkstry (v ruštine). Kibernetika i sistemnyj analiz, 1995, No.5, str.28-39.

[27] TSEYTLIN, G.E.:Algebra algoritmiky a vlastnosti štruktúry jej subalgebier (v ruštine). Dopovidi Nacionaľnoj Akadeniji Nauk Ukrajiny, 1995, No.11, str.11-14.

[28] TSEYTLIN, G.E.:Čo je to algoritmika? (v ruštine). II. konferencia Sorosovskych učiteľov, 1996, str.182-212.

[29] TSEYTLIN, G.E.:Konštrukcia zväzu subalgebier algebry Dijkstra (v ruštine). Kibernetika i sistemnyj analiz, 1997, No.1, str.27-45.

[30] TSEYTLIN, G.E.: Algebra logiky a konštruovania programov(v ukrajinčine).Kyjev, Naukova dumka,1994, 90s.

[31] JUŠČENKO,E.L.,TSEYTLIN, G.E.,GRICAJ,V.P., TERZJAN,T.K.:Mnohoúrovňové štrujtúrne projektovanie programov:Teoretické základy, inštrumentárium (v ruštine).Moskva, Finansy i statistika, 1989,-208str.

[32] TSEYTLIN, G.E.,JUŠČENKO:Mnohoúrovňové štruktúrne projektovanie programov(v ruštine).Kibernetika,1988, No.4, str.34-46.

[33] BODNARČUK,V.G.,TSEYTLIN, G.E.:O algebrách periodicky definovaných transformáciach nekonečného registra (v ruštine). Kibernetika,1969, No.1, str.18-28.

[34] GLUŠKOV,V.M.: Sintez cifrovych avtomatov (v ruštine). Moskva, Fizmatgiz 1962, 476 str.

[35] GLUŠKOV,V.M.,TSEYTLIN, G.E.,Juščenko,E.L.: Algebra. Jazyky. Programovanie (v ruštine). Kyjev, Naukova dumka, 1989, 3. vydanie, 376 str.

[36] GLUŠKOV,V.M.,TSEYTLIN, G.E.,Juščenko,E.L.: Metody simvoľnoj multiobrabotki. (v ruštine). Kyjev, Naukova dumka, 1980, 252 str.

[37] JANOV,J.I.:O logických schémach algoritmov(v ruštine). Problemy kibernetiki, 195vyp.1., str.75-127.

[38] TRACHTENBROT,B.A., BARZDIŇ,J.M.: Konečnyje avtomaty: Povedenije i sintez. Izd.Nauka, Moskva 1970.

[39] SHANNON,C.E.: A symbolic analysis of relay and switching circuits. *Trans. AIEE* 57:713-723, 1938.

[40] ŠESTAKOV,V.I.: Algebra dvuchpoljusnych schem postrojenych isključiteľno iz dvuchpoljusnikov (Algebra schem). *Avtomatika i telemechanika* 2(6):15-24, 1941.

[41] MOORE,E.F.: Gedanken experiments on sequential machines. In *Automata Studies* ed.C.E.Shannon and J.McCarthy, pp.129-153. Princeton, N.J.: Princeton Universaity Press 1956.

[42] MEALY,G.H.: A Method for synthesizing sequential circuits. *Bell System Technical Journal* 34:1045-1079, 1955.

[43] McCLUSKEY,E.J.: Minimization of boolean functions. *Bell System Technical Journal* 35(5):1417-1444. 1956.

[44] KLEENE,S.C.: Representations of events in nerve nets and finite automata. In *Automata Studies* ed.C.E.Shannon and J.McCarthy, pp.3-42. Princeton, N.J.: Princeton Universaity Press 1956.

[45] TURING,A.M.: On computable numbers with an application to Entscheidungsproblem. Proc. London Math. Soc.,2, 1936, č.42, pp.230-265.

[46] DAVIS,M.: Computability and unsolvability. New York, McGraw-Hill, 1958.

[47] Dahl,W.,Dijkstra,E.W.,Hoare,C.A.R.: Structured Programming. ????

[48] Janov,J.I.: O logičeskich schemach algoritmov. Problemy kibernetiki, 1958. Vyp.1,s.75-127.(v ruštine)

[49] Hudák, Š.:Niektoré problémy syntézy automatov.Minimová práca, EF VŠT Košice, 1975.

[50] Hudák, Š.:Strojovo-orientované jazyky. FEI TU Košice, 2003.218s.

[51] HUFFMAN,D.A.: The synthesis of sequential switching circuits. *J. Franklin Institute* 257:161-190,275-303, 1954.

[52] CHOMSKY,N..: Formal properties of grammars. In *Handbook of Mathematicla Psychology*2, New York, Wiley , pp.323-418,1963.

[53] CHOMSKY,N..: On certain formal properties of grammars. *Inf. and Control*, 2, č.2, pp.137-167, 1959.

[54] KORN, A.G., KORN, T.M.: Mathematical Handbook for Scientists and Engineers. McGraw-Hill Book Company 1968. (ruský preklad Nauka Moskva 1977, 832 str.)