RNDr. Josef Želenka
VSŽ n.p. Košice

PROGRAMOVÉ OVEROVANIE SPRÁVNOSTI RT - PROGRAMOV

Chyby v programoch možno rozdeliť do dvoch skupín:

a/ "gramatické" chyby, t.j. prečiny proti syntaxi príslušného programovacieho jazyka. Odstránia sa veľmi ľahko na základe diagnostických oznamov kompilátora, ďalej sa nimi nebudeme zaoberať;

b/ "logické" chyby, zahrňujúce všetko to, čo z ladenia programov robí zdĺhavú záležitosť: gramaticky správny program sa síce rozbehne, ale skončí s neočakávanými výsledkami - - zlyhá pre určité dáta, zacyklí sa, vystúpia z neho nesprávne výsledky, atď.

Pre bližšie štúdium logických chýb je účelné zaviesť si ich klasifikáciu. Podľa dopadu, ktorý majú na funkciu programu ich môžeme rozstriedit opäť do dvoch skupín:

Prvú skupinu logických chýb predstavujú chyby spočívajúce vo vadnej štruktúre programu, chybnom prenose riadenia, v jeho nesprávnej inicializácii a iných elementárnych algoritmických omyloch. Tieto chyby sú vlastné vnútornej stavbe príslušného programu; budeme hovoriť, že sú k nemu <u>interné</u>. Prejavia sa tým, že program jednoducho zlyhá, nefunguje ani sám osebe /"pokazia" ho dáta, zacyklí sa, prepíše si časť vlastného kódu a pod./.

Do druhej skupiny logických chýb patria chyby spočíva-

júce v nesprávnom chovaní sa programu voči okoliu, sú to jeho externé chyby. Program je síce odolný voči hocijakým vstupným dátam, zbehne za každých okolností, ale nesplní predpísanú úlohu. Tieto chyby sú spravidla ťažšie zvládnuteľné ako interné chyby. Prejavia sa nesprávnym fungovaním programu v systéme - program nesprávne reaguje na vstupy, vydáva vadné výstupy a pod. Externé chyby spôsobia, že zlyhá buď celý systém, alebo niektorá jeho časť.

Pre programy zapísané pomocou bežne používaných algoritmických prostriedkov a jazykov /vývojový diagram, PL/I a pod./ neexistuje žiaden prakticky použiteľný formálny postup na detekovanie logických chýb v programe. Takéto chyby sa odhalia až pri ladení programu so skúšobnými dátami, pri sledovaní systému, alebo - čo je omnoho nepríjemnejšie - až pri jeho prevádzke. Verifikovanie správnosti programu automatickými prostriedkami je možné len testovaním jeho dynamického chovania sa /pomocou sledovacích programov a pod./.

Budeme sa snažiť ukázať, že je možné vytvoriť prekladač rozhodovacích tabuliek /ďalej len RT/, ktorý môže testovať nielen syntaktickú, ale aj logickú správnosť prekladanej RT. Toto umožňuje, aby sa už počas gramatického ladenia programu /t.j. programu v jeho statickom tvare/ odladila - aspoň sčasti - aj jeho logika. RT takto predstavujú veľký krok vpred pri riešení problému programového dokazovania správnosti programov: umožňujú, aby počítač odhaľoval nielen chyby kódovania algoritmu, ale aj chyby algoritmu samého.

V ďalšom sa budeme zaoberať dvoma skupinami problémov:
- rôznymi zvláštnosťami a nedostatkami publikovaných techník pre dokazovanie nesprávnosti RT, t.j. v podstate pre hľadanie interných chýb v RT-programe /slepé vetvy, zacyklenie, nikdy nevykonávané časky kódu a pod/;
- možnosťami použitia prekladača RT na analýzu vnútornej povahy riešeného problému a chovania sa RT-programu, t.j. pre účely správnosti v RT zapísaného algoritmu, resp. pre hľadanie jeho externých chýb.

Dokazovanie nesprávnosti RT

sa opiera o analýzu vzťahov medzi pravidlami v RT /[1] kap.2.4, [2] kap.6 a 10/. Predovšetkým je možné testovať úplnosť RT, t.j. či v RT sú zahrnuté všetky možné kombinácie podmienok, aby sa nestalo, že pri vyhodnocovaní RT pre určitú kombináciu podmienok nevyhovie ani jedno pravidlo. Systematickým prešetrením kombinácií podmienok /3. kvadrantu RT/, resp. určitými operáciami s binárnymi maticami sa dajú identifikovať nejednoznačné pravidlá, t.j. také, ktoré sú pre tú istú kombináciu podmienok splnené súčasne. Porovnaním kombinácií činností možno pre takéto prípady zistiť, či ide o nadbytočné pravidlá - ak činnosti v nich sú rovnaké, alebo o rozporné pravidlá, ktoré pre tú istú kombináciu podmienok predpisujú prevedenie rôznych kombinácií činností. V prípade výskytu závislých podmienok sa v RT môžu vyskytnúť iracionálne pravidlá, ktorým zmysluplne nikdy vyhovieť lebo napr. predpisujú, aby

vek < 18 rokov ..... Y ...
vek > 50 rokov ..... Y ...

Ak sa RT použije aj na popis premenu riadenia v algoritme /pripojením rekurzívneho odvolania sa na seba samú pomocou akcie "opakuj RT" a pripojením počiatočného pravidla/ je za istých predpokladov možné otestovať, či pri vykonávaní RT-programu nemôže vzniknúť nekonečný cyklus.

Algoritmus pre zistenie úplnosti obmedzenej RT je všeobecne známy. Vychádza sa z toho, že ak RT obsahuje c podmienok, potom počet všetkých možných kombinácií [x] dvoch stavov Y, N týchto podmienok je 2^c. Ak v i-tom pravidle má n_i podmienok prázdny vstup /"-"/, potom toto pravidlo zahrňuje

$$2^{n_i}$$

kombinácií podmienok, nakoľko každý prázdny stav "-" repre-

[x] V skutočnosti ide o variácie s opakovaním, ale v literatúre sa ustálil tento praktickejší termín.

zentuje oba možné stavy Y aj N príslušnej podmienky. Podmienka úplnosti RT teda je

$$2^c = \sum_{i=1}^{r} 2^{n_i}$$

kde r je počet pravidiel. Zrejme ak $2^c < \sum 2^{n_i}$ je RT nejednoznačná, ak $2^c > \sum 2^{n_i}$ je RT neúplná.

Toto kritérium však ani zďaleka nemožno aplikovať úplne mechanicky. Treba si uvedomiť, že je síce nutnou, ale nie postačujúcou podmienkou pre úplnosť tabuľky. Uvažujme o RT

	1	2	3	4
C1	Y	Y	N	-
C2	-	N	-	Y
C3	N	-	Y	N
A1	X	-	-	X
A2	X	-	X	X
A3	-	X	-	-

Aplikácia kritéria hovorí, že $2^3 = 2 + 2 + 2 + 2$ a teda naša RT sa zdá byť úplnou. Po trochu experimentovaní však ľahko nahliadneme, že napr. kombinácii podmienok

$$\begin{bmatrix} C1 \\ C2 \\ C3 \end{bmatrix} = \begin{bmatrix} Y \\ Y \\ N \end{bmatrix}$$

vyhovuje pravidlo 1 aj 4; nakoľko pravidlá majú rovnaké akcie, v našej "úplnej" tabuľke sa objavila nadbytočnosť. Kombinácii

$$\begin{bmatrix} Y \\ N \\ N \end{bmatrix}$$

pritom vyhovujú súčasne pravidlá 1 aj 2 s rôznymi stavmi a máme teda do činenia aj s rozpornosťou.

Je teda zrejmé, že kritérium úplnosti treba používať nanajvýš opatrne a jedinou schodnou cestou sa zdá byť odstraňovanie nejednoznačností z tabuľky ešte pred uplatnením kritéria úplnosti: Ak prijmeme metódu vyhodnocovania RT "zľava doprava", potom nejednoznačnú kombináciu priradíme najľavejšiemu pravidlu, ktorému vyhovuje /v našom prípade oba nejednoznačné kombinácie priradíme pravidlu 1/ a z ostatných pravidiel ju "odčítame". Takto v našom prípade dostaneme

	1	2	3	4
C1	Y	Y	N	N
C2	-	N	-	Y
C3	N	Y	Y	N
A1	X	-	-	X
A2	X	-	X	X
A3	-	X	-	-

čo je zjavne neúplná tabuľka $2^3 > 2 + 1 + 2 + 1$; chýbajú pravidlá pre $\begin{bmatrix} Y \\ X \\ Y \end{bmatrix}$ a $\begin{bmatrix} N \\ X \\ N \end{bmatrix}$.

Nejednoznačný zápis, ktorý robí problémy pre test úplnosti je však pre zostavovateľa RT výhodný: môže ho použiť na zostručnenie RT zavedením akýchsi lokálnych else-pravidiel. Ak sa má dodržať konvencia o vyhodnocovaní RT zľava doprava /podľa pravidiel/, potom bude asi potrebné, aby prekladač nejednoznačnosť v RT nepovažoval za chybu, ale sa vyrovnal s týmto problémom napr. horeuvedeným spôsobom. Takýto prístup - odstraňovanie nejednoznačností - by potom namiesto nejednoznačnosti zaviedol a detekoval ďalší druh logickej chyby v RT - pravidlo, ktoré sa nikdy nevykoná.

Uvažujme o RT /nezaujímame sa o 2. a 4. kvadrant/

	1	2	3
C1	Y	Y	Y
C2	N	-	Y
a?b	>	*	<

ktorá je na prvý pohľad neúplná ale inak snáď v poriadku. Ak si však pravidlo 2 rozpíšeme na dve pravidlá nahradením prázdneho vstupu "-" pre C2 explicitne obomi možnosťami Y a N, dostaneme

	1	2a	2b	3
C1	Y	Y	Y	Y
C2	N	Y	N	Y
a?b	>	*	*	<

kde vidíme, že nakoľko symbol * znamená v podstate "väčší alebo menší":

a/ "odčítaním" pravidla 1 od 2b sa toto zmení na $\begin{bmatrix} Y \\ N \\ < \end{bmatrix}$;

b/ pravidlo 3 je plne obsažené v pravidle 2a a teda sa nik-

dy nevykoná, jeho uvedenie v tabuľke je logickou chybou.

Ďalším možným praktickým dôsledkom uvedeného postupu je zisťovanie kombinácií, ktoré sú zhrnuté v else-pravidle. Na to postačí postupne odčítať všetky pravidlá RT od else pravidla, pozostávajúceho v tomto poňatí zo samých prázdnych vstupov podmienok. Táto informácia môže byť pre zostavovateľa RT často veľmi užitočná.

Iným problémom, veľmi sťažujúcim kontrolu úplnosti RT, je výskyt závislých podmienok. Uvažujme o RT /upravené z [3] /:

	DEPUTÁT	1	2	3	4	5	6	7
C1	pracovník	Y	Y	N	N	N	N	N
C2	dôchodca	N	N	Y	Y	Y	Y	N
C3	cudzí	N	N	N	N	N	N	Y
C4	ženatý	Y	N	-	-	-	-	-
C5	v ban.odprac. < 5 rokov	-	-	Y	N	N	N	-
C6	v ban.odprac.5-10rokov	-	-	N	Y	N	N	-
C7	v ban.odprac.11-15 r.	-	-	N	N	Y	N	-
C8	v ban.odprac. >15 r.	-	-	N	N	N	Y	-
	prídel ? t uhlia	5.4	2.7	0	1.2	1.8	2.7	0
	prídel ? plm dreva	2	1	0	0	0	0	0

Na prvý pohľad je zrejmé, že táto RT neobsahuje žiadne dvojznačnosti; kritérium úplnosti tvrdí

$$2^8 \ (= 256) > 2^4 + 2^4 + 2^1 + 2^1 + 2^1 + 2^1 + 2^5 (= 72)$$

že tabuľka je neúplná. Ľahko nahliadneme, že skutočne neobsahuje napr. pravidlo pre kombináciu

$$\begin{bmatrix} Y \\ N \\ Y \\ N \\ Y \\ N \\ Y \\ N \end{bmatrix}$$

ktoré však popisuje iracionálnu situáciu: žiadateľa o deputát, ktorý má byť súčasne vlastným aj cudzím zamestnancom a v baníctve odpracoval menej ako 5 rokov ale súčasne aj 11-15 rokov.

Prepis tejto RT do rozšíreného tvaru /píšeme len 1. a 3. kvadrant/

DEPUTÁT-1		1	2	3	4	5	6	7
P1	žiadateľ ?	prac.	prac.	dôch.	dôch.	dôch.	dôch.	cudzí
P2	ženatý	X	X	-	-	-	-	-
P3	v ban.odprac. ? rokov	-	-	<5	5-10	11-15	>15	-

má až dve výhody

1. je samozrejme výhodnejší pre zostavovateľa;
2. ukážeme, že napriek bežnému tvrdeniu literatúry, že obmedzené RT sú pre automatický preklad vhodnejšie ako rozšírené RT, je - z nami sledovaného hľadiska - pre prekladač vhodnejší rozšírený tvar. Špeciálne ukážeme, že je možné dokázať úplnosť rozšírenej RT.

Predovšetkým si všimnime, že prepisom RT DEPUTÁT do RT DEPUTÁT-1 boli tri binárne závislé podmienky C1, C2 a C3 zhrnuté do jedinej rozšírenej podmienky P1. Táto podmienka má tri možné stavy /"prac.", "dôch." a "cudzí"/, ktoré sú vzájomne exkluzívne. Podobne zo štyroch závislých podmienok C5 - - C8 vznikla jedna 4-hodnotová rozšírená podmienka P3, ktorej všetky štyri možné stavy sú exkluzívne /rozsahy rokov nimi predpísané sa vzájomne neprekrývajú a spolu zahrňujú celý rozsah hodnôt odpracovaných rokov/.

Ak teraz "hodnotovosťou" h_i rozšírenej podmienky P_i nazveme počet všetkých možných jej exkluzívnych stavov, úvahou obdobnou ako pre prípad obmedzenej RT zistíme, že kritérium úplnosti rozšírenej RT je

$$\prod_{i=1}^{c} h_i = \sum_{j=1}^{r} \left(\prod_{\substack{i=1 \\ v_{ij} \neq -}}^{c} h_i \right)$$

kde c je počet podmienok,

r je počet pravidiel a

v súčine na pravej strane sa berú h_i len pre prípad, že príslušný vstup podmienky v_{ij} v i-tom riadku a j-tom stĺpci 3. kvadrantu je prázdny / ≠ "-"/.

Ľahko nahliadneme, že toto kritérium pre prípad obmedzenej tabuľky, kde všetky $h_i = 2$ prejde v pôvodné kritérium ú-

plnosti. Pre našu rozšírenú tabuľku DEPUTÁT-1 platí

$$3.2.4\ (=24) = 4 + 4 + 2 + 2 + 2 + 2 + 2.4\ (=24)$$

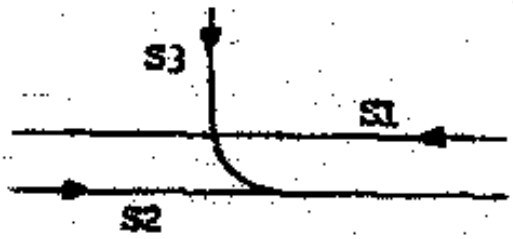
a teda táto RT je úplná.

Z povedaného je zrejmý záver, že je potrebné pracovať zásadne s rozšírenými/zmiešanými RT, pričom

- zostavovateľ RT je zodpovedný za zlúčenie všetkých závislých podmienok do jednej rozšírenej. Do zostavovanej RT sa mu povoľuje zavádzať ľubovoľné nejednoznačnosti; napr. aj kvôli tomu, aby nemusel explicitne vypísať všetky hodnoty viachodnotových podmienok, lebo ich trebárs ani nepoznú;
- prekladač RT je zodpovedný za odhalenie aspoň tu diskutovaných logických chýb v takto zostavenej tabuľke. Za týmto účelom bude pravdepodobne nutné pri spracovaní RT opustiť od booloveskej algebry a pracovať s aparátom viachodnotových logík.

Dokazovanie správnosti RT

Ďalšie možnosti, ktoré RT poskytujú pri analýze logiky algoritmu budú ilustrované na príklade analýzy bezpečnosti premávky na železničnej križovatke troch jednosmerných tratí podľa schémy



v ktorej šípky vyznačujú smer premávky. Premávka je riadená tromi semaformi S1, S2 a S3 s dvoma stavmi: Y - prepúšťa /zelená/, N - neprepúšťa /červená/. U takéhoto jednoduchého systému ľahko nahliadneme, že stav všetkých troch semaforov nie je pre bezpečnosť premávky rovnako významný. Zatiaľ čo trate 1 a 2 sú úplne nezávislé, premávka na nich sa vzájomne nechrozuje, stav semaforu S3 je kritický. Ak je S3 otvorený, môže ohroziť premávku na oboch zbývajúcich tratiach; len v tomto prípade záleží na stave semaforov S1 a S2 /oba musia byť zavrené, ak premávka má byť bezpečná/.

Ukážeme, že takúto analýzu vnútornej povahy riešeného systému, určenie kritičnosti jednotlivých jeho prvkov /semaforov/ pre cieľové chovanie sa systému /bezpečnosť premávky/ je možné previesť čisto formálnou analýzou RT popisujúcej tento systém.

Popíšme si teda tento problém pomocou RT /príklad je prevzatý z [2], str. 106, 174/, napr. tak, že popíšeme všetky možné kombinácie stavov semaforov:

BEZP

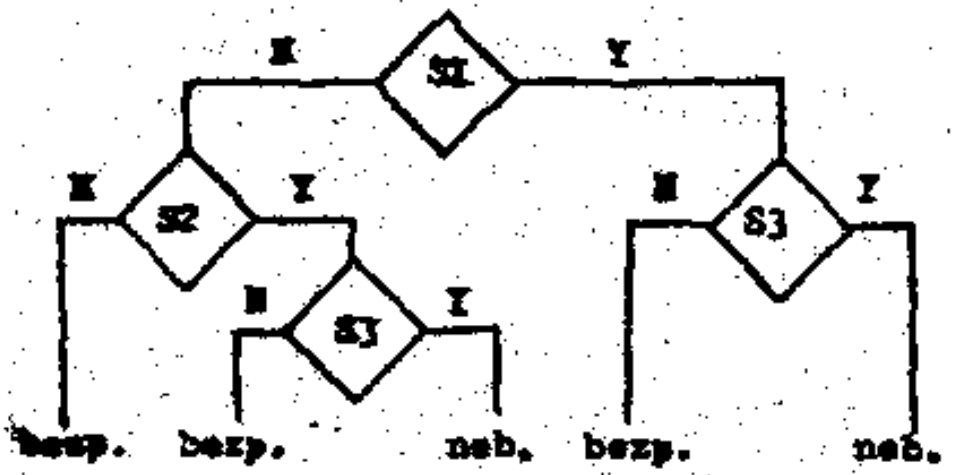
	1	2	3	4	5	6	7	8
S1	Y	Y	Y	Y	N	N	N	N
S2	Y	Y	N	N	Y	Y	N	N
S3	Y	N	Y	N	Y	N	Y	N
premávka bezpečná	-	X	-	X	-	X	X	X
premávka nebezpečná	X	-	X	-	X	-	-	-

Túto RT možno redukovať zlúčením pravidiel, v ktorých sú rovnaké stĺpce akcií a v ktorých sa kombinácie podmienok líšia len v jednej podmienke /Y, N/; pre ňu v zlúčenom pravidle vznikne prázdny vstup /-/. /O zlučovaní pravidiel viď bližšie napr. [1], kap. 2.4.1; [2], kap. 6/ V našom prípade možno zlúčiť napr. pravidlá 1 a 3, 2 a 4, 7 a 8 čím dostaneme

BEZP1

	1+3	2+4	5	6	7+8
S1	Y	Y	N	N	N
S2	-	-	Y	Y	N
S3	Y	N	Y	N	-
bezpečná	-	X	-	X	X
nebezpečná	X	-	X	-	-

Táto tabuľka obsahuje už len 5 pravidiel a ďalej je neredukovateľná. Čitateľ oboznámený s problematikou prevodu RT na vývojový diagram /viď napr. [1], kap.3/ vidí okamžite, že východzou podmienkou pre testovanie, ktorá vedie k optimálnemu vývojovému diagramu je S1; takto dostávame kód obsahujúci 4 testy



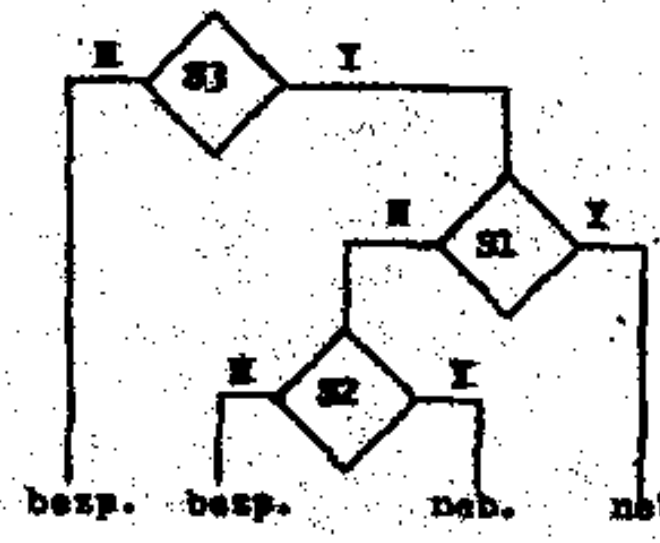
Keby sme však pri redukovaní RT BEZP miesto pravidiel 7 a 8 boli zlúčili pravidlá 6 a 8, dostali by sme pravidlo

-
N
Y
-

ktoré je ďalej možné zlúčiť s kombináciou pravidiel 2÷4, takže výsledkom by bola RT BEZP2, obsahujúca už len 4 pravidlá

BEZP2	1÷3	(2÷4)÷(6÷8)	5	7
S1	Y	-	N	N
S2	-	-	Y	N
S3	Y	N	Y	Y
bezpečná	-	X	-	X
nebezpečná	X	-	X	-

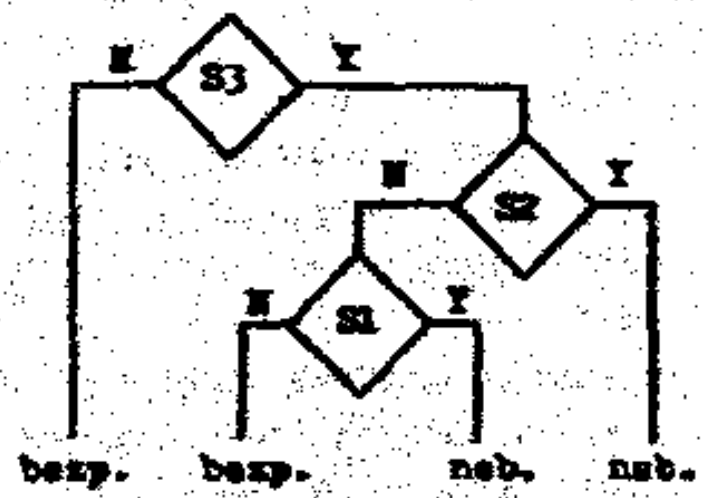
v ktorej kritickou podmienkou je S3 a, ktorá rezultuje vo výhodnejší vývojový diagram s len tromi testami:



Vidíme, že tento diagram už vystihuje kritičnosť semaforu S3, nakoľko jeho stav testuje ako prvý. Za menej kritický považuje S1 a najmenej kritický S2. Pozorný čitateľ však ľahko nahliadne, že zo všetkých možných spôsobov zlučovania pravidiel existuje ešte jeden /a už len tento/, ktorý taktiež vedie k optimálnemu výsledku

BEZP3	1+5	(2+4)+(6+8)	3	7
S1	-	-	Y	N
S2	Y	-	N	N
S3	Y	N	Y	Y
bezpečná	-	X	-	X
nebezpečná	X	-	X	-

t.j. k diagramu



ktorý sa od rovnaké optimálneho BEZP2 líši len poradím testovania S1 a S2, čo naznačuje, že kritičnosť semaforov S1 a S2 je rovnaká a menšia ako semaforu S3. Čisto formálnymi postupmi nad pôvodnou RT sme teda dospeli k rovnakému výsledku, ako rozumovou úvahou.

Snaha po optimalizácii programu má zjavne okrem svojho primárneho žiaduceho efektu ešte aj druhý dôsledok, ktorý je na žiadnie tesne viazaný: správnosť programu. Len tan človek /alebo prekladač/, ktorý problém pochopí ho môže optimalizovať: ide hneď po jadre problému, aby rýchle dospel k výsledku. Alebo naopak: z programu v jeho minimálnom tvare môže pochopiť naprogramovaný problém, môže usúdiť na rozhodujúce

faktory ovplyvňujúce chovanie sa programu, teda na správnosť programu.

Zistená skutočnosť má dva dôsledky. Predovšetkým sa zdá, že obecný princíp šetrnosti, ktorý je známy napr. vo fyzike /"sústava sa realizuje v stave s minimálnou energiou"/, ale aj v estetike /"v jednoduchosti je krása"/, platí aj pre programovanie, kde by sme ho mohli formulovať trebárs takto: chovanie sa systému je určené minimálnym programom. Pre programátorskú prax je však viac než tento ontologický aspekt dôležitý druhý, utilitárny dôsledok doteraz urobených úvah:

Zdá sa byť reálne, aby v dohľadnej budúcnosti riešenie úloh na počítači prebiehalo asi takto:
1/ úloha sa sformuluje, algoritmizuje a naprogramuje pomocou RT;
2/ prekladač na základe RT
   a/ vygeneruje program pre počítač,
   b/ analyzuje a popíše vlastnosti naprogramovanej úlohy
      /t.j. chovanie sa systému programom reprezentovaného/;
3/ človek na základe tohoto popisu usúdi, či program je správny, či realizuje ním formulovanú úlohu.
Otázka, akou formou by sa dali popísať vlastnosti algoritmu /mohol by to byť napr. kvantitatívne ohodnotený význam tej-ktorej podmienky pre tú-ktorú činnosť/ zostáva samozrejme otvorená; dôležitý je samotný fakt, že takýto popis je principiálne možný.

Záver

Na základe jednoduchých príkladov je ilustrované, ako môžu prekladače RT prispieť ku automatizácii nielen kódovania, ale aj ladenia programov. Úvahy boli orientované na predkompilátor RT; možnosti kompilátora RT sú pravdepodobne väčšie.

Literatúra

[1] Chvalovský, V.: Rozhodovací tabulky, SNTL Praha 1974.
[2] Košner, J.: Rozhodovací tabulky, ČKVŠ Praha 1972.
[3] Standard č. 12 /Rozhodovací tabulky/, OKR Automatizace řízení Ostrava 1977.