Acta Universitatis Carolinae. Mathematica et Physica

Ladislav Koubek

Algoritmus kompilace nepodmíněných výrazů, dosazovacích příkazů a příkazů skoku v překladači jazyka Algol 60

Acta Universitatis Carolinae. Mathematica et Physica, Vol. 10 (1969), No. 1-2, 57--69

Persistent URL: http://dml.cz/dmlcz/142233

1969 ACTA UNIVERSITATIS CAROLINAE MATHEMATICA ET PHYSICA VOL.10

ALGORITMUS KOMPILACE NEPODMÍNĚNÝCH VÝRAZŮ, DOSAZOVACÍCH PŘÍKAZŮ A PŘÍKAZŮ SKOKU V PŘEKLADAČI Z JAZYKA ALGOL 60

L. KOUBEK
Centrum numerické matematiky UK, Praha

АЛГОРИТМ ТРАНСЛЯЦИИ БЕЗУСЛОВНЫХ ВЫРАЖЕНИЙ ОПЕРАТОРОВ ПРИСВАИВАНИЯ И ОПЕРАТОРОВ ПЕРЕХОДА В ТРАНСЛЯТОРЕ С ЯЗЫКА АЛГОЛ 60. В работе приводится полная система правил, по которым в трансляторе с языка АЛГОЛ 60 проводится трансляция безусловных операторов и выражений. Эти правила основываются на нами введенном понятии микропрограммы.

Kompilování aritmetických výrazů je pravděpodobně nejlépe propracovanou částí teorie i praxe tvorby kompilátorů. Velmi úplný přehled různých metod je uveden v /3/.

V této práci jsme se však pokusili navrhnout nový algoritmus a to tak, aby příslušná část překladače mohla být použita ve všech případech, v nichž se v ALGOLu 60 výrazy mohou vyskytnout. Mohou to být části dosazovacích příkazů, indexové výrazy, skutečné parametry procedur atd. Algoritmus je upraven tak, aby nebylo nutné striktně rozlišovat výrazy aritmetické, boolovské a cílové a současně abychom ho mohli použít při kompilaci nejdůležitějších základních syntaktických jednotek, tj. dosazovacích příkazů a příkazu skoku.

V dalším předpokládáme, že všechny potřebné seznamy obsahující údaje o všech identifikátorech jsou již vytvořeny a že všechny

symboly operátorů a některé další symboly (identifikátory elementárních funkcí) jsou zařazeny do operační tabulky. V této tabulce je u každého symbolu uvedena posloupnost strojových instrukcí (rozumí se v kódu použitého počítače), která je mu přiřazena a tabulkový stupeň operátoru. Je samozřejmé, že zobrazení množiny operátorů na množinu posloupností strojových instrukcí nemusí být vzájemně jednoznačné, neboť několika různým operátorům může být přiřazena táž posloupnost instrukcí. Stupně operátorů vyjadřují v podstatě jejich prioritu (viz /1/, odst. S.3.5. a 3.3.6.).

Vhodnou volbou posloupností instrukcí a tabulkových stupňů formálně snadno odstraníme rozdíly mezi operátory aritmetickými, relačními a boolskými, takže v dalším můžeme mluvit jen o kompilaci výrazů.

Operační tabulka v uvedeném tvaru byla volena proto, že práce s ní je velmi jednoduchá a současně umožňuje jednoduše provádět změny významu operátorů i eventuální rozšíření o další operátory např. pro práci s řetězci nebo komplexními proměnnými. Konkrétní volba operační tabulky pro operátory definované v ALGOLu 60 a počítač ODRA 1003 je uvedena v /2/.

Protože posloupnosti instrukcí závisejí podstatně na použitém počítači, nebudeme je explicitně uvádět a pro jednoduchost budeme stejnými symboly označovat operátory v textu i posloupnosti instrukcí strojového programu, které jsou jim přiřazeny. Identifikátory budeme používat jak pro označení veličin v algolském textu tak adres, které jsou jim kompilátorem přiřazeny.

Algoritmus nejsnáze popíšeme pro jednoadresový počítač, na kterém byl také ověřen. Z toho plyne, že všechny symboly operátorů použité ve významu posloupností instrukcí chápeme jako kódy jednoadresového počítače s adresou označenou příslušným identifikátorem nebo konstantou. Protože druhým operandem většiny aritmetických instrukcí je u jednoadresového počítače obsah střádače, musíme zavést ještě označení pro některé strojové instrukce, které se vyskytují jen ve strojovém programu, především pro instrukci přenosu informace z paměti do střádače, kterou označíme B a instrukci opačnou, kterou označíme H. Označení některých dalších strojových instrukcí zavedeme později.

Text psaný v programovacím jazyce postupně načítáme a každý symbol ihned zpracujeme na základě jeho lokálních a semilokálních vlastností bez studia kontextu, v němž se vyskytuje. Textu přiřazujeme strojový program vždy po úsecích, kterým říkáme jednotky programu, a které jsme definovali v /4/.

V tomto článku se omezíme jen na výrazy nebo příkazy, které jsou základními syntaktickými jednotkami, tj. popíšeme jen kompilaci programovacích jednotek, které v sobě již neobsahují další jednotky programu. Tím automaticky vylučujeme podmíněné příkazy a výrazy. Nejde nám ani o to, jak máme pokračovat v práci po sestavení programu jednotky programu a proto budeme předpokládat, že jejím ukončujícím symbolem je vždy ; .

Algoritmus kompilace rozdělíme na dvě fáze. V prvé fázi, která začíná přečtením počátečního symbolu jednotky, přiřazujeme symbolům textu podle níže uvedených pravidel mikroprogramy. V druhé fázi, která začíná po nalezení ukončujícího symbolu, sestavujeme z těchto mikroprogramů úsek strojového programu a podle indexu základního přepínače /4/ přejdeme k dalším částem algoritmu.

Abychom v dalším nekomplikovali výklad, umluvíme se, že v první fázi chápeme stupně operátorů dynamicky, tj. jako funkci, jejíž hodnota závisí na tabulkovém stupni operátoru a na rozložení závorek v textu. Je-li právě načten operátor φ a je-li z počet dosud neuzavřených závorek a k jistá pevně zvolená konstanta větší než maximální tabulkový stupeň operátorů, definujeme hodnotu této funkce vztahem

$$s(\varphi) = \varphi_s + k.z, \qquad (2)$$

kde φ_s je tabulkový stupeň operátoru. Pokud v dalším budeme mluvit o stupni operátoru, budeme mít na mysli hodnotu této funkce.

Podle syntaxe ALGOLu 60 je výraz posloupností symbolů, v níž jsou identifikátory, konstanty, závorky a operátory. Výraz začíná buď identifikátorem (konstantou) nebo operátorem typu sčítání. Odstraníme-li formálně závorky tím, že každému operátoru přiřadíme jeho stupeň s (φ) a na místa, kde chybí levý operand doplníme nulu, lze říci, že výraz je posloupnost symbolů začínající identifikátorem (konstantou) a střídají se v něm pravidelně identifikátory s operátory. Výraz končí ukončujícím symbolem (v této práci předpokládáme, že je to ;). Identifikátor funkce se sou-

hrnem skutečných parametrů a identifikátor pole se seznamem indexů pokládáme za jediný symbol.

Úkolem translátoru je přiřadit této posloupnosti symbolů posloupnost instrukcí, jejíž provedení vede k vyčíslení hodnoty výrazu.

Termínem posloupnost operátorů výrazu budeme označovat posloupnost operátorů vybranou z takto upravené posloupnosti symbolů, tvořících výraz (jednotku programu), přičemž uspořádání je podstatné. Vyslovíme definici:

Definice 1: Mikroprogram je posloupnost instrukcí, jejíž provedení vede k výpočtu hodnoty části výrazu. Začíná instrukcí B a patří do ní instrukce přiřazené posloupnosti operátorů výrazu, jejichž stupně nerostou.

Definice 2: Stupeň prvého operátoru posloupnosti nazveme stupněm mikroprogramu.

Mezivýsledky nutné pro výpočet výrazu, tj. hodnoty, které jsou výsledkem (některých) mikroprogramů budeme ukládat do pole pracovních buněk p_i, kde i je index tohoto pole.

Definice 3: Indexem mikroprogramu rozumíme index pracovní buňky, do které mikroprogram ukládá (alespoň formálně) výsledek své činnosti.

Mikroprogram stupně k a indexu i budeme označovat symbolem M_i^k. Budeme říkat, že zapisujeme do mikroprogramu, když do prvé dosud neobsazené buňky pole, které je vyhrazeno v paměti počítače pro tento mikroprogram, zapisujeme některou instrukci, a že otevíráme nový mikroprogram, když instrukci B zapisujeme do prvé buňky jeho pole.

Mějme výraz, tj. posloupnost symbolů

$$a_1 \varphi_1 a_2 \varphi_2 \cdots \varphi_m,$$

kde a_i, $i = 1,\ldots, m$ jsou identifikátory nebo konstanty, φ_i, $i = 1,\ldots, m-1$ jsou operátory a φ_m je ukončující symbol.

Výraz zpracováváme podle těchto pravidel:

Pravidlo A 0: Symboly načítáme zleva.

A 0 1: Načteme-li a_i, najdeme v seznamech odpovídající adresu a další údaje. Adresu uložíme do pracov-

vní buňky A a načítáme další symbol.

A 0.2.: Načteme-li φ_i, $i < m$, určíme podle (2) jeho stupeň, vyhledáme v operační tabulce posloupnost instrukcí, která je mu přiřazena a pokračujeme podle pravidla A 1.

A 0.3.: Načteme-li symbol, který není uveden v operační tabulce, pokračujeme v práci podle odpovídajícího pravidla C.

Zatím uvedeme jen dvě z těchto pravidel:

Pravidlo C 1.: Načteme-li symbol (, zvětšíme index z (pro výpočet stupně podle (2)) a pokračujeme podle A 0.

Pravidlo C 1': Načteme-li) , zmenšíme index z.

Poznámka: Symbol) může patřit mezi ukončující symboly. V této práci ho však jako ukončující symbol neuvažujeme a proto neformulujeme odpovídající pravidlo D.

Poznámka: V dalším předpokládáme, že na počátku práce algoritmu je v buňce A adresa nuly. Kromě buňky A budeme ještě používat pracovní buňku S, jejíž počáteční stav je 0.

Předpokládejme, že jsou již otevřeny mikroprogramy $M_0^k o$, $M_1^k 1 \ldots$ $M_t^k t$, že právě zapisujeme do $M_j^k i$, kde $o \leq j \leq t$ a že načítáme symbol φ_r, $r < m$. Označíme-li, jak je obvyklé, lomenými závorkami obsah buňky, je

$$\langle A \rangle = a_r,$$

a jak uvidíme později

$$\langle S \rangle = s(\varphi_{r-1}).$$

Pravidlo A 1.:

A 1.1.: Je-li $\langle S \rangle < s(\varphi_r)$, zapíšeme do mikroprogramu $M_j^i j$ instrukci

$$\varphi_{r-1}\, p_{t+1}$$

a otevřeme nový mikroprogram $M_{t+1}^{k_{t+1}}$, kde $k_{t+1} = s(\varphi_r)$, instrukcí B $\langle A \rangle$.

Do S zapíšeme hodnotu $s(\varphi_r)$, do A adresu nuly a pokračujeme podle A 0.

Poznámka: Je-li $r = 1$, je $s(\varphi_1) > 1$ a není otevřen žádný mikroprogram s nezáporným indexem. Abychom mohli používat pravidlo

A 1.1. bez vyjímek, položíme definitoricky φ_0 = B, ⟨A⟩ = adresa 0, a zapisujeme do fiktivního mikroprogramu M_{-1}^{0}. Současně otevíráme mikroprogram $M_0^{s(\varphi_r)}$ instrukcí B ⟨A⟩.

A 1.2.: Je-li S ≥ $s(\varphi_r)$, zapíšeme do M_j^kj instrukci φ_{r-1} ⟨A⟩ a vyhledáme největší nezáporný index i, pro který $k_i \geq s(\varphi_r)$. Neexistuje-li takový index, položíme i = 0. Další zápis provádíme do M_i^ki.
Do S zapíšeme $s(\varphi_r)$, A adresu nuly a pokračujeme podle A 0.

A 1.3.: Je-li r = m, tj. načítáme-li koncový symbol, zapíšeme odpovídající hodnotu do indexu základního přepínače, do M_j^ki zařadíme instrukci φ_{m-1} ⟨A⟩ a pokračujeme podle pravidla B.

Pravidlo B se ovšem týká už druhé fáze, ve které vytváříme z mikroprogramů program výpočtu hodnoty výrazu. V této práci ho vyslovíme ve zjednodušeném tvaru, v němž by bylo vhodné i pro překladače z jednodušších jazyků a teprve v /5/ ho vyslovíme v definitivním tvaru tak, jak je to třeba pro překlad ALGOLu 60.

Pravidlo B: Do programu zařazujeme mikroprogramy sestupně podle indexů i, přičemž pro i ≥ 1 zařazujeme do zapsání mikroprogramu instrukci Hp_i.
Práci podle B ukončíme po zapsání mikroprogramu s indexem 0 a podle stavu základního přepínače pokračujeme v práci podle pravidla D (t), kde t je index tohoto přepínače.

Pravidla D (t) vyslovíme v dalších pracech. V této práci jen předpokládáme, že ukončíme práci, neboť program výpočtu hodnoty výrazu (příkazu), který nás zajímá je sestaven.

Uveďme příklad kompilace nepodmíněného výrazu

-a+bx (c+dx(e+fxh)+l)xm+nx(-o+q);

nejprve odstraníme závorky a přiřadíme operátorům jejich stupně (konstantu k ve vzorci (2) volíme k = 12) a doplníme chybějící operandy:

1	1	2	13	14	25	26	13	2	1	2	13	13
0	-	a	+	b	x	c	+	d	x	e	+	f

x	h	+	l	x	m	+	n	x	0	-	o	+	q	;

Poznámka 1: Tuto předběžnou úpravu provádíme ovšem jen pro názornost výkladu. Při realizaci algoritmu na počítači podle našich

pravidel jsou provedeny automaticky (viz pravidlo A 1.). Hodnotu stupně počítáme podle (2) po vyhledání údajů o operátoru v operační tabulce; hodnotu m zvětšujeme o k při nalezení každé otevírající závorky a zmenšujeme při nalezení závorky uzavírající.

Poznámka 2: Chceme-li standardní funkce zpracovávat jako operátory, musíme naše pravidla poněkud modifikovat. V tomto případě se totiž mohou v posloupnosti symbolů textu vyskytovat vedle sebe dva operátory, např. = sin (a). Abychom mohli i v tomto případě použít náš algoritmus, musíme zavést prázdný symbol, který zařazujeme mezi + a "operátor" sin. Pravidla A 1.1. a A 1.2. doplníme větou: "Je-li obsahem A prázdný symbol, nezapisujeme instrukci B".

Posloupnost instrukcí přiřazená "operátoru" standardní funkce musí mít ovšem tvar

B ...

podprogram,

kde slovem podprogram označujeme instrukci vyvolání příslušného podprogramu. Tabulkový stupeň tohoto "operátoru" musí být větší než tabulkové stupně všech ostatních operátorů. V této práci však standardní funkce jako operátory nepoužíváme, neboť bychom museli zavést další omezení proti ALGOLu 60. Identifikátory standardních funkcí by totiž nemohly být skutečnými parametry procedur.

Pro přehled sestavíme dvě tabulky. V prvé uvedeme údaje o postupně načítaných symbolech, stav buněk A a S, a uvedeme do kterých mikroprogramů zapisujeme. Ve druhé tabulce jsou strojové instrukce patřící do jednotlivých mikroprogramů. Nesmíme ovšem zapomenout na to, že doplněné nuly načítány nebudou.

Tabulka 1

pořadí načítání	symbol	stav A		stav S		mikroprogram		příští zápis
		počáteční	konečný	počáteční	konečný	zapisujeme	otevíráme	
0	-	0	0	0	0	-		M^0_{-1}
1	-	0	0	0	1	M^0_{-1}	M^1_0	M^1_0
2	a	0	a					

pořadí načítání	symbol	stav A		stav S		mikroprogram		příští zápis
		počáteční	konečný	počáteční	konečný	zapisujeme	otevíráme	
3	+	a	0	1	1	M_0^1		M_0^1
4	b	0	b					
5	x	b	0	1	2	M_0^1	M_1^2	M_1^2
6	c	0	c					
7	+	c	0	2	13	M_1^2	M_2^{13}	M_2^{13}
8	d	0	d					
9	x	d	0	13	14	M_2^{13}	M_3^{14}	M_3^{14}
10	e	0	e					
11	+	e	0	14	25	M_3^{14}	M_4^{25}	M_4^{25}
12	f	0	f					
13	x	f	0	25	26	M_4^{25}	M_5^{26}	M_5^{26}
14	h	0	h					
15	+	h	0	26	13	M_5^{26}	-	M_2^{13}
16	l	0	l					
17	x	l	0	13	2	M_2^{13}		M_1^2
18	m	0	m					
19	+	m	0	2	1	M_1^2	-	M_0^1
20	n	0	n					
21	x	n	0	1	2	M_0^1		M_6^2
22	-	0	0	2	13	M_6^2	M_7^{13}	M_7^{13}
23	o	0	o					
24	+	o	0	13	13	M_7^{13}	-	M_7^{13}
25	q	0	q					
26	;					M_7^{13}		

Tabulka 2

mikroprogram		M_0^1	M_1^2	M_2^{13}	M_3^{14}	M_4^{25}	M_5^{26}	M_6^2	M_7^{13}
instrukce	1	B 0	B b	B x	B d	B e	B f	B n	B o
	2	-a	xp2	+p3	xp4	+p5	x h	xp7	-o
	3	+p1	x m	+1					+q
	4	+p6							

Po nalezení ukončujícího symbolu sestavíme z mikroprogramů podle pravidla B program, který uvádíme v tabulce 3.

Tabulka 3

α + 0	B 0
+ 1	- o
+ 2	+ q
+ 3	H p7
+ 4	B n
+ 5	x p7
+ 6	H p6
+ 7	B f
+ 8	x h
+ 9	H p5
α + 10	B e
+ 11	x p5
+ 12	H p4
+ 13	B d

+ 14	x p4
+ 15	H p3
+ 16	B c
+ 17	+ p3
+ 18	+ 1
+ 19	H p2
α + 20	B b
+ 21	x p2
+ 22	x m
+ 23	H p1
+ 24	B 0
+ 25	- a
+ 26	+ p1
+ 27	+ p6

Snadno ověříme, že program, který jsme vytvořili, vede k výpočtu hodnoty výrazu. Není obtížné nahlédnout, že podle pravidel A a B vytvoříme správný program pro výpočet hodnoty každého nepodmíněného výrazu. V těchto programech je však ještě mnoho zbytečných instrukcí. Změnou pravidla A 1 a pravidla B tyto zbytečné instrukce odstraníme.

Pravidlo A 1.1'.: Je-li $\langle S \rangle < s(\varphi_r)$, a je-li volná právě druhá buňka pole mikroprogramu M^k_jj, zapíšeme do ní instrukci

$$\varphi_{r-1}\omega ,$$

kde ω je adresa z instrukce zapsané v první buňce mikroprogramu. První buňku mikroprogramu vymažeme.
Je-li druhá buňka mikroprogramu obsazena, zapíšeme do prvé volné buňky instrukci $\varphi_{r-1}\ p_{t+1}$ a v obou případech otevřeme nový mikroprogram M^{kt+1}_{t+1} instrukcí B ⟨A⟩.

Poznámka: V tomto případě je ovšem j = t a kromě toho $\omega = a_{r-1}$. Toho však nemůžeme využít, protože adresa identifikátoru a_{r-1} byla již nahražena adresou a_r.

Pravidlo B': Do programu zařazujeme mikroprogramy sestupně podle indexů i. Je-li i ≥ 1 a první instrukce mikroprogramu s indexem i-1 je nenulová, zařadíme po zapsání mikroprogramu instrukci Hp_i.
Zařazování mikroprogramů ukončíme po zapsání mikroprogramu s indexem nula a pokračujeme podle pravidla D(t).

Použijeme-li obě pozměněná pravidla na náš příklad, změní se mikroprogramy tak, jak je uvedeno v tabulce 4 a po jejich zařazení do programu podle pravidla B' dostaneme program uvedený v tabulce 5.

Tabulka 4

	K_0^1	K_1^2	K_2^{13}	K_3^{14}	K_4^{25}	K_5^{26}	K_6^2	K_7^{13}
1	B 0	0	0	0	0	B f	0	B 0
2	-a	x b	+c	xd	+e	xh	xm	-o
3	+p1	x m	+1					+q
4	+p6							

Tabulka 5

α + 0	B 0
+ 1	- o
+ 2	+ q
+ 3	x n
+ 4	H p6
+ 5	B f
+ 6	x h
+ 7	+ e
+ 8	x d

+ 9	+ c
α + 10	+ 1
+ 11	x b
+ 12	x n
+ 13	H p1
+ 14	B 0
+ 15	- a
+ 16	+ p1
+ 17	+ p6

Tento program je ve skutečnosti programem výpočtu hodnoty výrazu

-a + ((f x h + e) x d + c + e) x b x m + (-6 + q)x n;

vzhledem k tomu, že levým operandem každé aritmetické instrukce je obsah střádače.

Neuvažujeme-li některé mezní případy, kdy v důsledku velkého rozdílu v řádu hodnot nemusejí být strojové operace v pohyblivé řádové čárce komutativní, je hodnota tohoto výrazu identická s hodnotou výrazu zadaného.

Poslední program je již ve značném počtu případů optimálním programem výpočtu hodnoty výrazu. Neoptimální zůstane např. program výpočtu hodnot výrazu.

a + b + c x d;,

který bude mít tvar

Bc
x d
Hp_1
B a
+ b
+ p_1

nebo výrazu

a x b + a x b;,

který bude

B a
x b
Hp_1
B a
x b
+ p_1

Naším algoritmem lze kompilovat nejen výrazy, ale také příkazy skoku. Stačí k tomu zařadit do operační tabulky operátor postupu go to a přiřadit mu stupeň -1. Přiřazená posloupnost instrukcí se skládá z jediné instrukce nepodmíněného předání řízení.

Operátor go to načítáme ovšem vždy jako první symbol. Podle pravidla A 1.2. je ⟨S⟩ = 0 > s(go to)= -1, takže do M^0_{-1} zapíšeme H 0, a protože neexistuje žádný mikroprogram s nezáporným indexem, bude příští zápis do M^{-1}_0. Dalším symbolem je návěští, např. s a ukončující symbol, takže do M^{-1}_0 zapíšeme instrukci go to s a podle B' vytvoříme program téhož tvaru.

Abychom mohli programovat také dosazovací příkazy, musíme připojit další z pravidel C a to:

Pravidlo C 2.: Je-li načten oddělovač :=, zapíšeme do zvláštního mikroprogramu M (který nemá ani stupeň ani index) do prvé volné buňky instrukci H ⟨A⟩, do A dáme adresu nuly, za φ_r vezmeme instrukci B (stupně nula) a pokračujeme podle A 0. Kromě toho je nutno pozměnit předposlední větu pravidla B resp. B'. Pro jistotu vyslovíme pravidlo B znovu.

Pravidlo B 1: Do programu zařazujeme mikroprogramy sestupně podle indexů i. Je-li $i \geq 1$ a první buňka mikroprogramu s indexem $i - 1$ je nenulová, zařadíme po zapsání mikroprogramu instrukci Hp_i. Zařazování mikroprogramů ukončíme po zapsání mikroprogramu s indexem 0. Je-li alespoň jedna buňka mikroprogramu M nenulová, zapíšeme všechny instrukce tohoto mikroprogramu do programu. Po té pole mikroprogramu M vymažeme a pokračujeme podle jednoho z pravidel D (t).

Podotkněme na závěr, že zřejmou slabinou popsaného algoritmu je nemožnost úplné kontroly textu po stránce syntaktické. Nelze např. dobře kontrolovat, zda argumentem příkazu skoku je cílový výraz nebo veličina jiného druhu apod.

Literatura

/1/ Backus J. W. - Programování v jazyku ALGOL 60, SNTL - Praha 1963

/2/ Koubek L. - Programující program PHEN-ALGOL pro počítače ODRA 1003 a 1013, Závěrečná zpráva číslo 10/69 VÚZORT - Praha

/3/ Randell B. a Russel L. J. - ALGOL 60 Implementation, Ac.Press - London 1964

/4/ Koubek L. - Algoritmus překladače z jazyka ALGOL 60 vhodný pro malý počítač, AUC, Math.-Phys., Vol.10, 47-56, (1969)

/5/ Koubek L. - Programování podmíněných výrazů a příkazů v translátoru z jazyka ALGOL 60, AUC, Math.-Phys., Vol. 10, 77-85 (1969)