I N T E R A K T I V N Í G R A F O V É S Y S T É M Y

RNDr.Zdeněk Botek, UJEP Brno

1. Úvod do problematiky

Problematika gramatik pracujících nad řetězci patří ke klasickým partiím matematické informatiky, chomskyho hierarchie jazyků je notoricky známa. Teoretické výsledky výzkumu v této oblasti jsou počítačovou praxí běžně využívány. V poslední době se vyskytlo několik problémů, jejichž řešení si vyžádalo zavedení gramatik nad složitějšími strukturami - nad grafy. Uveďme si několik příkladů, u nichž lze problém jednoduchým a přitom dostatečně formálním způsobem popsat pomocí pojmenovaného grafu. Všechny tyto příklady vyžadují aparát umožňující (obvykle interaktivní) modifikaci grafu podle přesně formulovaných pravidel.

INKREMENTÁLNÍ KOMPILACE: Při ladění programů v systémech s konvenčním program. jazykem a generačním způsobem překladu je po lokalizaci chyby nutná rekompilace celého programu. K řešení tohoto problému se někdy užívá inkrementální způsob překladu. Podrobněji viz /3/,/5/,/7/,/9/. Inkrementální kompilátor nejdříve transformuje zdrojový text do tzv.skeletonu, jenž zachycuje logickou strukturu programu a z něj se teprve generuje výsledný kód. Modifikace zdrojového textu způsobí změny ve skeletonu a rekompilaci pouze odpovídající části výsledného kódu. Pozornost zaměříme na skeleton, neboť musí representovat zdrojový program takovým způsobem, aby jeho modifikace byla jednoduchá, ale přitom jednoznačná. Navíc musí být zachován požadavek korektnosti skeletonu a tedy syntakt. správnosti programu po provedené modifikaci.

Jedna možnost pro representaci programu je popsána v /7/. Skeleton je tvořen inkrementy obsahujícími mimo informaci o významu inkrementu také množinu ukazatelů. Tyto strukturální ukazatele vytvářejí vnořené lineární seznamy representující seznamy příkazů na jednotlivých úrovních. Modifikace programu spočívá

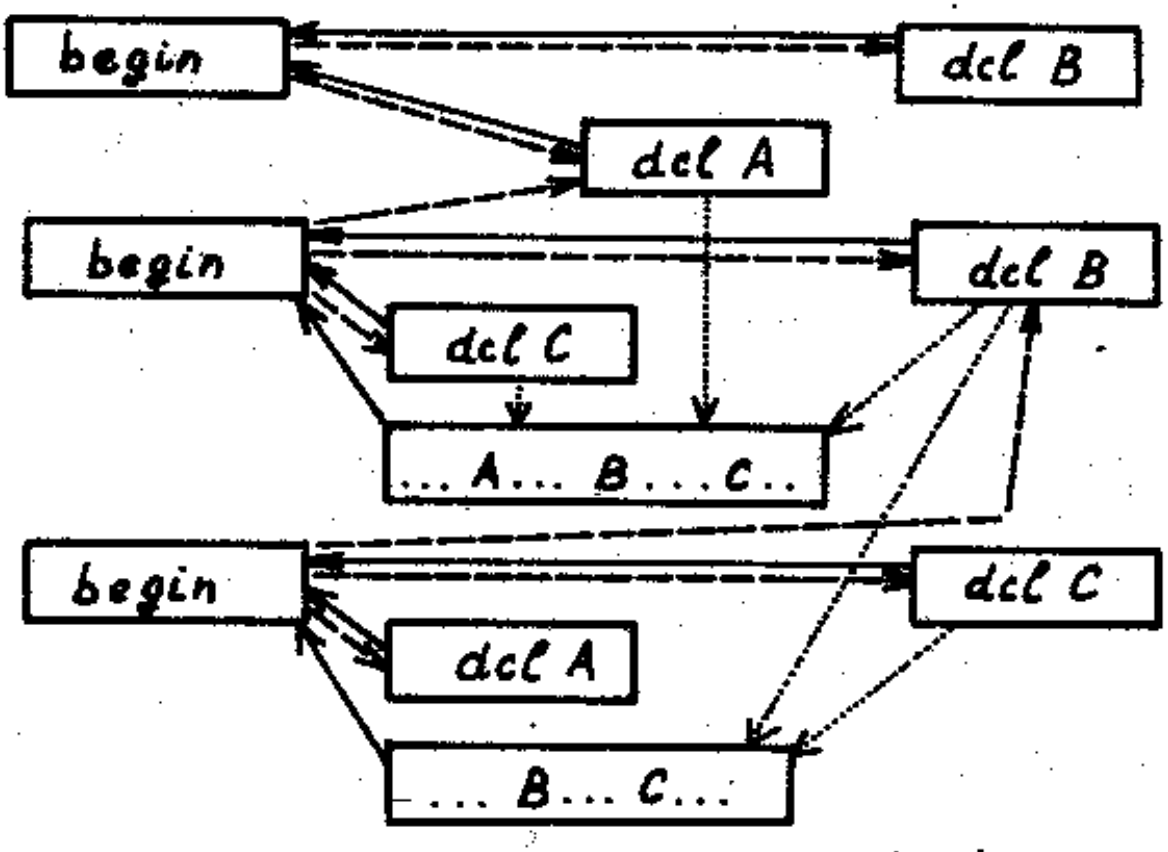
v přerušení lineárního seznamu a vynechánim (vložením, příp. modifikací) některých inkrementů a opětovné spojení seznamu.

Druhá varianta vychází z representace programu pojmenovaným grafem. Jednotlivé usly grafu představují typy inkrementů, pojmenované hrany vyjadřují různé typy relací mezi inkrementy. Uzly jsou tedy pojmenovány např. begin, dcl A, end, stat,.. . Typy hran jsou např: - aplikovaný výskyt deklarované proměnné
- první inkrement bloku
- následující inkrement atd.

Příklad bude pro názornost obsahovat pouze deklarace proměnných a jejich aplikovaný výskyt:

```
begin dcl B;  dcl A; ...
                begin dcl C;  dcl B; ...
                        ...A...B...C...
                        begin dcl A;  dcl C; ..
                                ...B...C...
                        end
                end
end
```

Representaci programu pojmenovaným grafem uvádí obr.1.



Obr. 1

Různé typy hran popisují následující vztahy:

a ——————→ b inkrement a je prvkem bloku b

a - - - - - - - →b deklarace b platí v bloku a

a ..... →b deklarovaný identifikátor je použit v inkrementu b

Jednoduchá modifikace zdrojového textu způsobí netriviální změnu grafové representace. Jestliže např. do třetího bloku vložíme deklaraci proměnné b, změna grafu bude realizována následující výměnou subgrafů:

Ani takovou jednoduchou změnu nelze popsat jednoznačně levou a pravou stranou pravidla, které změnu realizuje. Naše řešení např. nebere v úvahu možnost, že proměnná b je již v bloku deklarována, neřeší případné modifikace dalších odkazů.

Problém inkrementální modifikace grafové representace programu vyžaduje zavedení formálního aparátu popisujícího grafovou strukturu a umožňujícího její netriviální změny. S tímto se seznámíme ve 2.kap. Kapitola 3. naznačí možný způsob aplikace formálního aparátu grafových gramatik při řešení problému inkrementální kompilace.

NÁVRH ALGORITMU: Od doby, co jsou používány interaktivní grafické vstupní a výstupní zařízení, je na místě otázka, zda lineární notace algoritmů (tj. program v některém z obecných program. jazyků) je struktura adekvátní myšlení programátora. Není snadnější vytvářet tu stěžejní část procesu zpracování úlohy na počítači -návrh algoritmu- pomocí dvoudimenzionálních prostředků jako je vývojový diagram, síťová mapa atd.? Pro velkou část úloh (speciálně pro nenumerické aplikace se složitou strukturou dat) bude tento způsob výhodnější. Takto navržený program i datová struktura jsou opět určitým druhem kódu, ovšem kódu bližšího představivosti programátora. Jeho formalizací se dostaneme k pojmenovanému grafu, kde hrany představují tok řízení či přístup. cestu, uzly označují význam inkrementu. Příslušná grafová

gramatika může být použita na přímý generační proces, což zajistí vytváření syntakticky správných programů. Viz /12/.

GRAF PROGRAMU: Je-li program zapsán obvyklým způsobem (tedy v program. jazyce), může být velmi užitečná representace jeho logické struktury. Tento tzv. graf programu lze použít v několika aspektech teorie, aplikace a implementace:
- techniky optimalizace kódu /10/
- vyhodnocování nezávislých (paralelních) částí /10/
- specifikace operačních sémantik /13/
- formulace problémů analýzy toku dat /16/ a jiné

BÁZE DAT: Grafová representace se zde používá pro popis tzv. konceptuálního schématu datové báze - tj. specifikace přípustných struktur dat na logické úrovni. Grafová gramatika pak vyjadřuje operace nad bází dat /14/.

ABSTRAKTNÍ TYPY DAT: Datové struktury a příslušné operace nad nimi jsou chápány jako nedělitelná jednota také v teorii abstraktních datových typů. Grafový přepisovací systém je zde využíván jako prostředek ke specifikaci operací na abstraktní úrovni/15/.

Pokusme se shrnout společné požadavky aplikačních oblastí a současně naznačit možnosti aparátu grafových gramatik. Ve všech uvedených aplikacích je nutné representovat strukturu problému pojmenovaným grafem, nad nímž je definována množina pravidel (produkcí). Stejně jako v gramatikách nad řetězci, i zde obsahuje produkce levou a pravou stranu - obě ve tvaru pojmenovaného grafu. Navíc zde vzniká zásadní problém - jak začlenit pravou stranu produkce do původního grafu ochuzeného o levou stranu produkce. Přístupy ke grafovým gramatikám jsou rozdílné především v právě popsaném problému, liší se tedy v řešení otázky transformace vložení. Obvykle je definován nějaký typ zobrazení, který "předává" vlastnosti uzlů levé strany produkce uzlům pravé strany. Vstupuje-li např. v původním grafu do uzlu A hrana p a vystupuje z něj hrana q, bude mít ve výsledném grafu tuto vlastnost uzel pravé strany produkce, který je obrazem uzlu A. Přístup, který popíšeme ve 2.kap., se snaží být

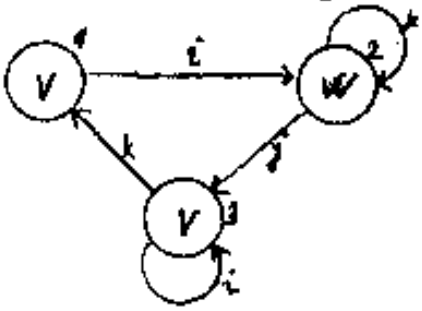
natolik obecný, aby umožnil co nejširší prostor pro manipulaci s hranami spojujícími pravou stranu produkce se zbytkem původního grafu. Grafová gramatika s takto definovanou transformací vložení je schopna klasifikovat všechny předchozí přístupy.

Uvedené příklady aplikací také naznačují, že středem našeho zájmu nebude generativní síla gramatiky, která začíná jedním počátečním grafem a generuje nekonečně dlouhou derivační posloupnost. Pro naše aplikace je závažnější problém grafového přepisovacího mechanismu, který realizuje přepis pojmenovaného grafu podle formálně přesně definované produkce na graf jiný. Proto také dáváme přednost pojmu grafový přepisovací systém před pojmem grafová gramatika.

2. Grafové gramatiky

Definice 2.1: Pojmenovaným grafem nad V je (n+2)-tice
$d_n(V)=(K, \varsigma_1 \ldots \varsigma_n, \beta)$, kde
a) K je konečná abeceda pro označení uzlů, obvykle přirozená čísla 1,...k.
b) $\varsigma_i \subseteq K \times K$ - hrany grafu. Čísla 1..n representují pořadové číslo jména hrany.
c) β: K→V - funkce přiřazující uzlu jeho jméno

Poznámka : Obrázku odpovídá vpravo zapsaný formální popis.



$d_3(V)$: K = (1,2,3)
$\varsigma_i = \{(1,2);(3,3)\}$
$\varsigma_j = \{(2,3)\}$
$\varsigma_k = \{(2,2);(3,1)\}$
β: 1→v; 2→w; 3→v

Dále budeme pod pojmem graf rozumět pojmenovaný graf ve smyslu definice 2.1.

Definice 2.2: $d'_n(V)$ je podgraf grafu $d_n(V)$, právě když:
a) $K' \subseteq K$
b) $\varsigma'_i = \varsigma_i \cap (K' \times K')$
c) $\beta' = \beta/K'$

Poznámka : Podgraf s množinou uzlů K' označujeme také $d_n(K')$.

Nechť $d_n(K') \leq d_n(V)$. Podgraf $d_n(K-K')$ se nazývá komplementární ke grafu $d_n(K')$.

Definice 2.3: Množina hran $EM(d',d) := \{IN_1(d',d), OUT_1(d',d), \ldots IN_n(d',d), OUT_n(d',d)\}$ se nazývá vložení grafu $d'_n(V)$ do grafu $d_n(V)$.

Množiny $IN_i(d',d)$ a $OUT_i(d',d)$ representující i-hrany vstupující (resp. vystupující) do grafu (resp. z grafu) $d'_n(V)$ jsou definovány následovně:

$IN_i(d',d) := \rho_i \cap ((K-K') \times K')$

$OUT_i(d',d) := \rho_i \cap (K' \times (K-K'))$

Poznámka: Nyní přichází na řadu otázka přímého odvození. Máme-li k dispozici produkci, která definuje záměnu grafu d_l za d_r (oba jsou ve tvaru $d_n(V)$ grafu), jak tuto produkci aplikovat, abychom z počátečního grafu d_A dostali výsledný graf d_B ? Otazník se týká především problému správné manipulace s vložením $EM(d_l,d_A)$ tak, abychom dostali správné vložení $EM(d_r,d_B)$. Tato manipulace by měla být nezávislá na hostitelských grafech d_A, d_B. Znamená to, že transformace vložení bude součástí produkce stejně jako levá a pravá strana.

Náš přístup nejdříve ve zbytku grafu $d(K_A-K_l)$ určí uzly, které se budou podílet na spojení zbytku $d(K_A-K_l)$ s pravou stranou produkce d_r. K tomu budou sloužit operátory, které po aplikaci na uzly grafu d_l obsaženého v d_A určí uzly grafu $d(K_A-K_l)$ podílející se na spojení. Následuje operace $d_A - d_l$, jejímž výsledkem bude graf $d(K_A-K_l)$. Nový graf ztrácí uzly K_l, hrany mezi uzly K_l a také hrany mezi uzly K_l a K_A-K_l. Uvedená operace tedy zrušila všechna spojení zbytku grafu s měněným podgrafem. A nyní můžeme přistoupit ke spojení zbytku grafu $d(K_A-K_l)$ s pravou stranou d_r. Spojení bude realizováno hranami mezi dříve specifikovanými uzly zbytku $d(K_A-K_l)$ a uzly grafu d_r. Jméno i orientace těchto hran je součástí produkce.

Definice 2.4: Operátor je slovo nad abecedou $V_l = V \cup \{L_l, R_l, C, \cap, \cup, (,)\}$, které aplikováno na uzel grafu d_l určí množinu uzlů grafu $d(K_A-K_l)$.

Tyto uzly dále slouží jako počáteční pro hrany vstupující do grafu d_r v grafu d_B nebo jako cílové pro hrany vystupující z grafu d_r. Transformace vložení musí pro uvedené hrany dále určit cílové (resp. počáteční) uzly grafu d_r.

Základní operátory jsou definovány následovně:

$L_i^{d',d}(k) := \{\underline{k} \mid \underline{k} \in K-K' \wedge (\underline{k},k) \in \varrho_i\}$

- určí uzly, z nichž vycházejí i-hrany do k

$R_i^{d',d}(k) := \{\underline{k} \mid \underline{k} \in K-K' \wedge (k,\underline{k}) \in \varrho_i\}$

- určí uzly, do nichž vcházejí i-hrany z k

$(CA)^{d',d}(k) := \{\underline{k} \mid \underline{k} \in K-K' \wedge \underline{k} \in K - A^{d',d}(k)\}$

- komplement v K-K' k množině uzlů $A^{d',d}(k)$

$v_1 \ldots v_m A^{d',d}(k) := \{\underline{k} \mid \underline{k} \in A^{d',d}(k) \wedge \beta(\underline{k}) \in (v_1 \ldots v_m)\}$

- podmnožina $A^{d',d}(k)$ uzlů, jejichž jméno je obsaženo v množině $v_1 \ldots v_m$

Se základními operátory je možné provádět operace skládání, průniku a sjednocení.

$AB^{d',d}(k) := \{\underline{k} \mid \exists \underline{\underline{k}}(\underline{\underline{k}} \in B^{d',d}(k) \wedge \underline{k} \in A^{d',d}(\underline{\underline{k}}))$

$(A \cap B)^{d',d}(k) := A^{d',d}(k) \cap B^{d',d}(k)$

$(A \cup B)^{d',d}(k) := A^{d',d}(k) \cup B^{d',d}(k)$

Komplement prázdného operátoru budeme dále označovat I, to jest $I := C\emptyset$. Operátor I tedy určí všechny uzly $K-K_1$. Pro ilustraci si uveďme několik příkladů operátorů:

1) $(CL_i)^{d',d}(k)$ - všechny uzly $\underline{k} \notin K'$, ze kterých nevede i-hrana do k

2) $L_i R_j^{d',d}(k)$ - množina všech uzlů $\underline{k} \notin K'$ spojených s k řetězcem následujícího tvaru:

3) $((v_1 L_i) \cup (v_2 R_j))^{d',d}(k)$ - množina uzlů $k \notin K'$, které mají jméno v_1 a vede z nich i-hrana do k nebo mají jméno v_2 a vede do nich j-hrana z k

Definice 2.5: Produkce je trojice $p=(d_l, d_r, E)$, kde $d_l, d_r \subseteq d_n(V)$ a $E = (l_1, r_1, \ldots, l_n, r_n)$ je transformace vložení.

$$l_i = \bigcup_{j=1}^{p} A_j(k_j') \times \{k_j''\} \qquad r_i = \bigcup_{j=1}^{q} \{k_j''\} \times A_j(k_j')$$

kde $k_j' \in K_l$, $k_j'' \in K_r$, $A_j \in op(V_1)$, $p, q \geq 1, 1 \leq i \leq n$

Poznámka: Hrany l_i, r_i představují spojení grafu $d(K-K_l)$ s pravou stranou produkce d_r, hrany l_i jsou vstupní, hrany r_i výstupní vzhledem k d_r. Index j naznačuje možnost, že lze hrany se stejným jménem definovat více způsoby.

Definice 2.6: Graf $d' \in d_n(V)$ je přímou derivací grafu d podle produkce $p=(d_l, d_r, E)$, právě když:

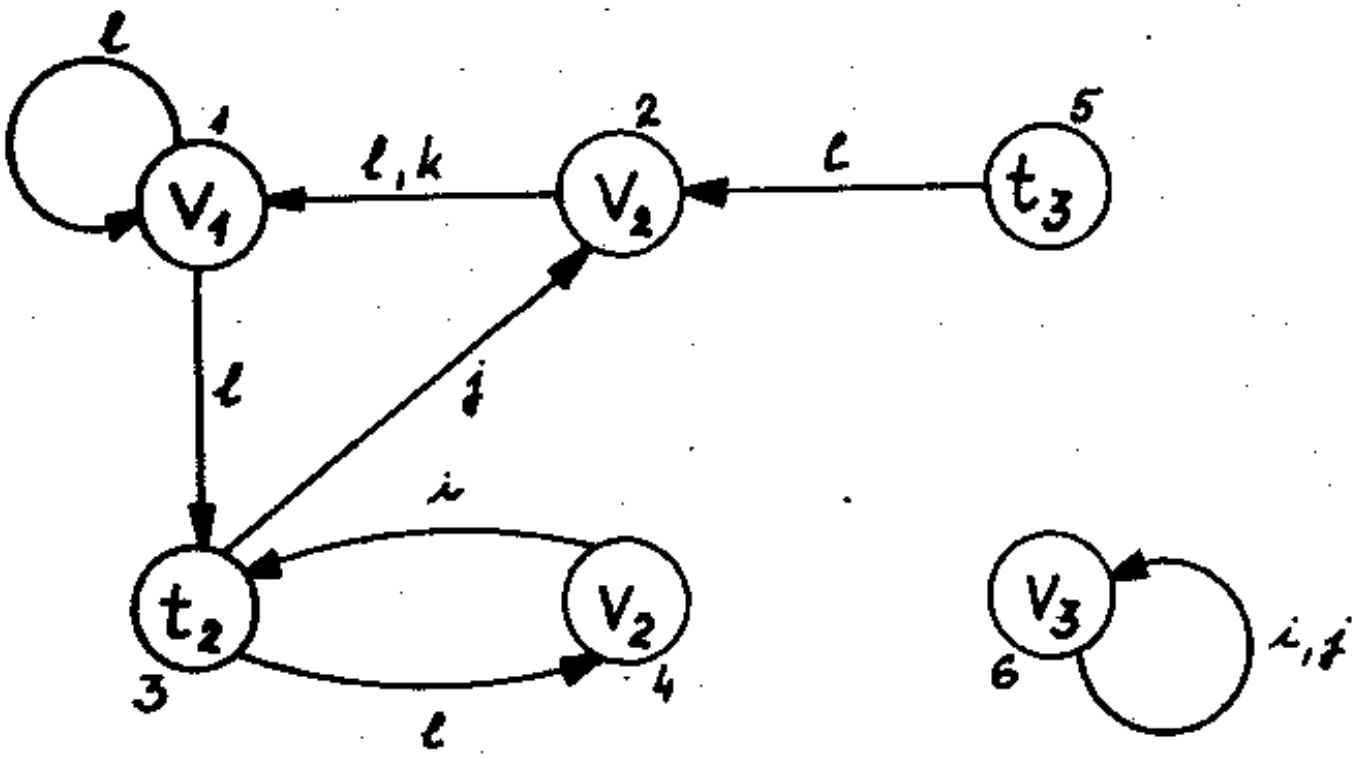
a) $d_l \subseteq d, \quad d_r \subseteq d'$

b) $d - d_l = d' - d_r$

c) $IN_i(d_r, d') = l_i \qquad OUT_i(d_r, d') = r_i$

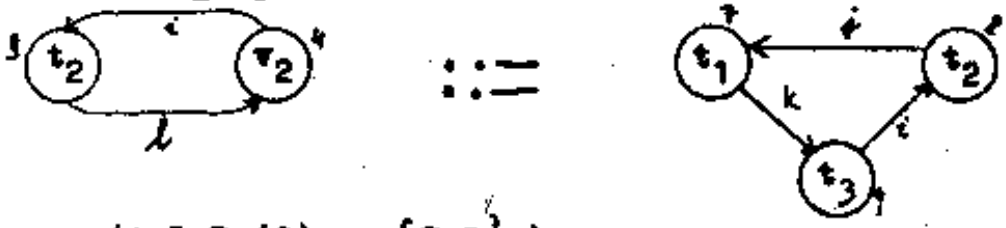
Zapisujeme $d \xrightarrow{r} d'$.

Příklad : Obrázek 2 představuje pojmenovaný graf $d_n(V)$.



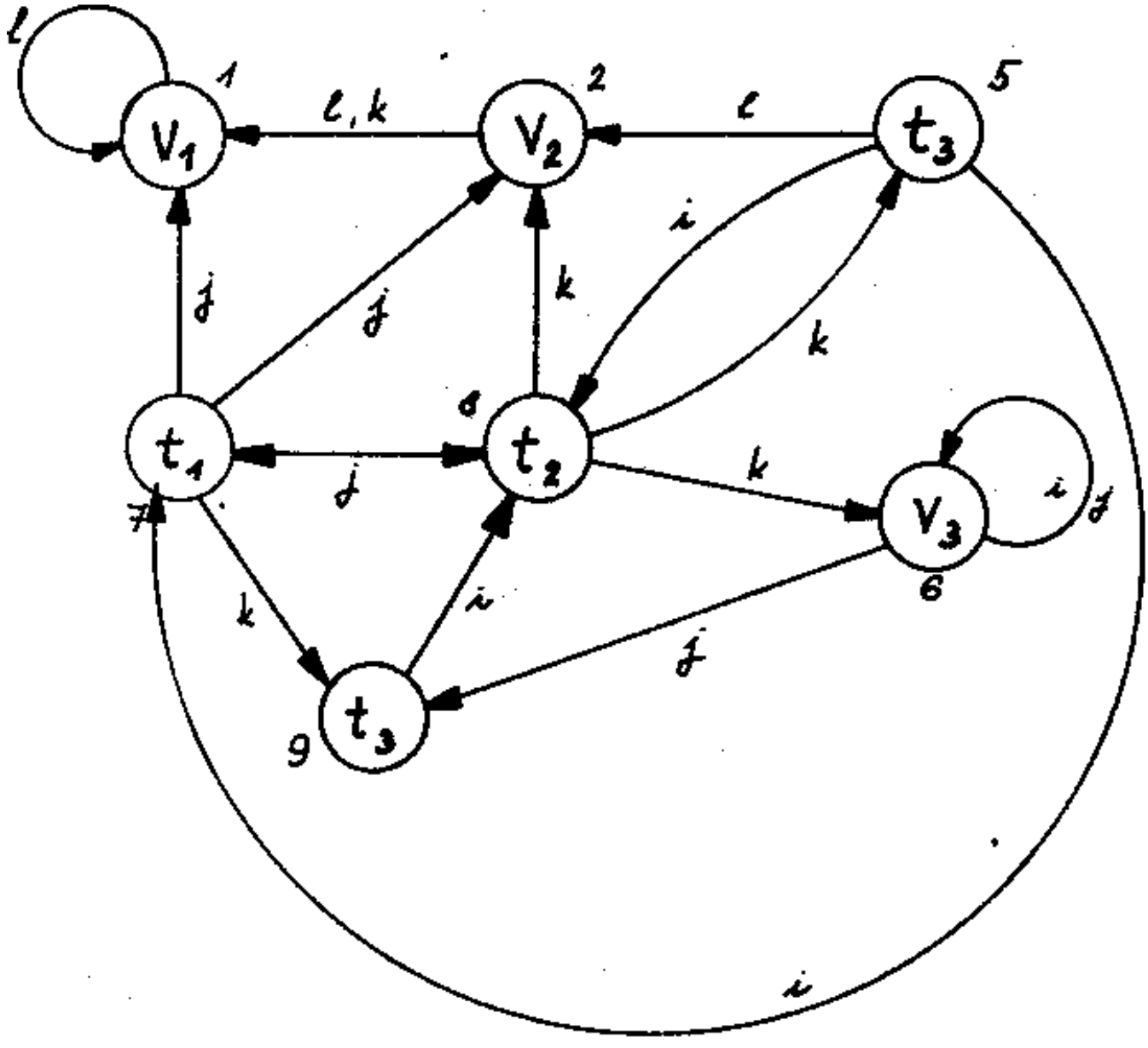
Obr. č. 2

Produkce $p=(d_l, d_r, E)$ je definována následovně:



E: $l_1 = (t_3 L_1 R_j(3) \times \{7,8\})$
$l_j = (v_3 t_1 I(4) \times \{9\})$
$r_j = (\{7\} \times R_j \cup L_1 L_1(4,3))$
$r_k = (\{8\} \times CR_k R_j(3))$

Aplikací produkce p na původní graf dostáváme graf z obr.3



Obr. č.3

Náhrada levé strany pravou a její spojení se zbytkem grafu probíhá podle dříve popsaného postupu. Například prvek l_1 transformace vložení vytváří hrany jména 1 vycházející z uzlu $t_3L_1R_j(3)$ a končící na uzlech 7 a 8 pravé strany. Operátor $t_3L_1R_j$ aplikovaný na uzel 3 původního grafu postupně určí uzly označené čísly 2(aplikace R_j) a 5(aplikace L_1 na uzel 2). Výsledný uzel má jméno t_3 shodné se jménem určeným operátorem a proto z něj povede hrana na uzel 7 a hrana na uzel 8 pravé strany produkce.

Poznámka: Příklad naznačil široké možnosti přepisovacího mechanismu používajícího dříve definované operátory. Popsaný přístup umožňuje štěpení i slučování hran, změnu orientace i jména hrany, generování hran závisejících na jménu uzlu, specifikaci uzlu řetězcem jednoduchých operátorů, generování komplementárních nebo úplně nových hran. Jsme tedy schopni pomocí určitých omezení definovat jednodušší přístupy, získali jsme základnu pro vzájemné srovnávání a výměnu výsledků.

Definice 2.7: Grafová gramatika je pětice $G = (V,T,k,d_0,P)$, kde

a) V je konečná abeceda jmen uzlů.

b) $T \subseteq V$ je terminální abeceda uzlů

c) $\varphi_1 \dots \varphi_k$ pro $k \in n$ určuje relace terminál. uzlů, $\varphi_{k+1} \dots \varphi_n$ specifikuje relace neterm. uzlů

d) $d_0 \in d_n(V) - (d_n(T) \cup \{d_\varepsilon\})$ je počáteční graf, $\{d_\varepsilon\}$ je prázdný graf

e) P je konečná množina produkcí $p = (d_l, d_r, E)$ výše popsaného tvaru s tím, že $d_l, d_r \in d_n(V)$ a $d_l \notin d_n(T) \cup \{d_\varepsilon\}$

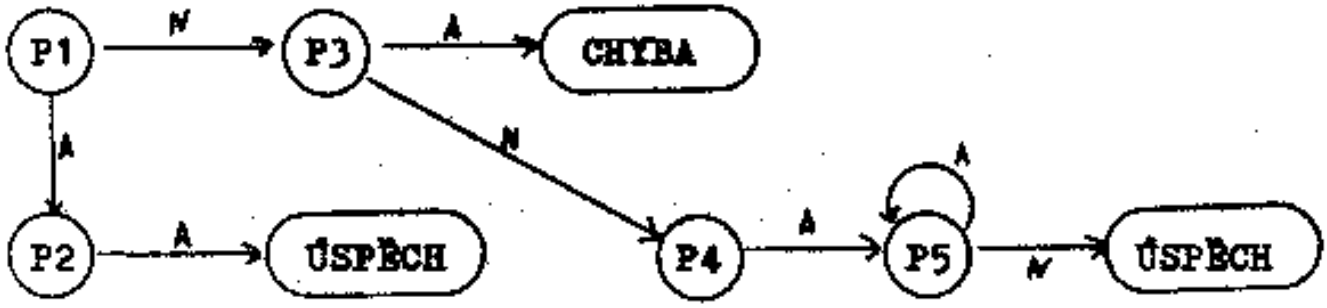
Definice 2.8: Jazyk grafové gramatiky G je

$$L(G) = \{d \mid d \in d_n(T) \wedge d_0 \xrightarrow[G]{*} d\}$$

3. Aplikace grafových gramatik

Formální aparát grafových gramatik je dostatečně mohutným základem pro řešení problémů uvedených v kap.1. Předvedeme řešení naznačeného problému inkrementální kompilace.

Grafová gramatika je zde definována jako pětice $G = (A_U, A_H, P, S, C)$. Konečné množiny A_U, A_H obsahují jména uzlů a hran, S je počáteční graf. P je konečná množina primitivních akcí, z nichž některé jsou produkce ve smyslu kap.2, ostatní jsou například dotazy na stav grafu apod. Navíc je definován pro každý typ modifikace grafu tzv. řídící diagram, který zajistí realizaci příslušného typu modifikace.Tvar i aplikace řídícího diagramu odpovídá funkci diagramu vývojového. V jeho uzlech jsou primitivní akce z množiny P, jejich zpracování lze označit názvem podmíněný operační blok. Je-li primitivní akce aplikovatelná, provede se a pokračuje se ve zpracování diagramu větví A. Když nelze aplikovat primitivní akci, pokračuje se po větvi N. Například pro typ modifikace grafu způsobené vynecháním deklarace platí následující řídící diagram:



Primitivní akce představují následující operace:

P1 - existuje deklarace identifikátoru v nadřazeném bloku?
P2 - vymaž deklaraci a předej odkazy
P3 - existuje aplikovaný výskyt identifikátoru?
P4 - vymaž deklaraci
P5 - vymaž hrany - - - - - - - - → z vnořených bloků

Vezmeme-li v úvahu graf z obrázku 1, snadno projdeme řídící diagram při vykonání následujících úloh:

1. Z vnějšího bloku vynechej deklaraci A
   (po cestě P1,P3 narazíme na CHYBA; úlohu nelze realizovat,neboť by se porušila korektnost grafu)

2. Z vnitřního bloku vynechej deklaraci C
   (po cestě P1,P2 se dostaví ÚSPĚCH)

3. Z vnitřního bloku vynechej deklaraci A
   (po cestě P1,P3,P4,P5 se dostaví ÚSPĚCH)

L i t e r a t u r a :

1. NAGL,M: Formal Languages of Labelled Graphs, Computing 16 (1976) ,pp.113-137

2. SCHNEIDER,H,J: Syntax-directed description of incremental compilers. Lect.Notes Comp.Sci 26,pp.192-201,(1975)

3. NAGL,M: An Incremental Compiler as Component of a System for Software Development. Informatik-Fachberichte 25, pp.29-44,(1980)

4. RAJLICH,V: Úvod do teorie počítačů. SNTL 1979

5. BOTEK,TOMAN,ŠTĚTINA: Interaktivní systémy. SOFSEM 79, str.249-282, Labaká bouda 1979

6. BOTEK,Z: Konverzační (strukturované) programování. PROGRAMOVÁNÍ 80, str.264-276, Havířov 1980

7. BOTEK,Z: Incremental compilation languages with nested statement structure. SCRIPTA Fac.Sci.Nat.Univ. Purk.Brun., vol.10(1980), No.8,pp.377-387

8. PADAWITZ,P: Graph grammars and operational semantics. Theoret.Computer Science vol.19(1982),pp.117-141

9. MEDINA-MORA,R FEILER,P.H: An Incremental Programming Environment, IEEE Trans.Software Eng.,Sept.1981, 472-481

10. NAGL,M: Application of Graph Rewriting to Optimization and Parallelization of Programs,Computing 1981,p.105

11. ENGELS ...:Software specification by Graph Grammars,preprint

12. DELLA VIGNA,P.L: PLAN2d , Lect.Notes Comp.Sci 26(1975)

13. CULIK,K II: Linked Forest manipulation system. Techn.Rep 77

14. EHRIG,H: Algebraic theory of Graph Grammars, Workshop WG 78

15. RAJLICH,V: Theory of Data Structures by Graph Grammars,LNCS52

16. FARROW,R: Graph Grammars and Global Program Data Flow Analysis. Techn. report 1979