NOVÁ TECHNOLOGIE PRO INTERAKTIVNÍ NÁVRH PROGRAMŮ

HNDr Ladislav Dvořák,CSc., VÚMS Praha

Úvod

R-technologie je programovací technologie pro interaktivní návrh a ladění programů vytvořená Ústavem kybernetiky kyjevské Akademie věd. Na její podporu byl vybudován programový systém RTK, který byl realizován prakticky pro všechny typy počítačů a operačních systémů dostupných v SSSR. V současné době jej v SSSR provozují řádově stovky výpočetních středisek. VÚMS ve spolupráci s SSSR vyvíjí verzi R-technologie pod DOS4. V současné době již existuje i provozovatelná verze RTK pro personální počítače. Obě tyto verze by měly být v ČSSR dostupné koncem roku 1989.

Návrh programu pomocí R-technologie je založený na silné programové podpoře návrhu shora dolů a na použití tzv. R-grafů pro popis programových struktur. Přitom je pro návrh programu pomocí R-technologie charakteristické, že pro zápis programu používá dvourozměrnou plochu místo lineárního zápisu, takže program má kratší délku a je přehlednější /je-li vhodně navržen/.

Práce s RTK

RTK je interaktivní systém orientovaný na uživatelský přístup podobným způsobem jako programové systémy pro personální počítače, ačkoliv byl původně vyvíjen pro "klasický provoz" /pod OSem, později pod CMS/. Pro zobrazení informací používá panely a menu, které uživatel většinou ovládá pomocí PF kláves, i když může použít i klasické povelové příkazy. K dispozici je též rozsáhlý systém help.

Základní komponenta systému RTK, kterou při komunikaci s počítačem uživatel používá, je tzv. archiv, což je soubor, který obsahuje přehled skupiny programů, s nimiž uživatel pracuje.

Archiv je pomocí panelu zobrazován na terminál. Uživatel může do archivu přidávat nebo ubírat odkazy na programy, přejít na zpracování programu ap. Programy navržené pomocí RTK se nazývají práce. Jejich zápis je uchovávám v souboru uspořádaném určitým způsobem. S pracemi lze pracovat pomocí menu, které nabízí editování, překládání, ladění, tisk dokumentace ap.

Mezi pracemi může existovat hierarchická závislost volání - systém RTK při překladu a ladění automaticky provede spojení všech potřebných programů.

Při návrhu programu pomocí R-technologie je důležitý pojem abstrakce - míněný ve smyslu navrženém Dijkstrou v /1/. R-technologie umožňuje zapsat do textu programu volání abstrakce jako libovolný identifikátor /resp. text prakticky neomezené délky/ a označit jej jako abstrakci, jejíž realizace bude popsána odděleně. Při překladu bude text abstrakce vkopírován na příslušné místo. Abstrakce jsou tedy vlastně jednoduché makropříkazy bez parametrů. Jejich význam nespočívá v jejich síle, ale v mechanismu jejich zabudování do RTK, který umožňuje jejich velmi pružné a pohodlné zpracování. Jejich použití však vyžaduje, aby zápis práce byl organisován určitým způsobem.

Zápis práce je realizován pomocí souboru rozděleného na několik částí nazývaných pole. První pole je pole specifikací, které obsahuje slovní popis programu. Druhé je pracovní pole, které obsahuje kořen programu. Třetí a další jsou pole popisující jednotlivé abstrakce - přitom uvnitř abstrakce lze volat opět další, hierarchicky nižší abstrakci. Větší abstrakce lze realizovat jako samostatné práce. Tím vzniká hierarchická struktura programu navrženého pomocí RTK.

Soubor polí odpovídající konkrétní práci je přístupný pomocí panelu, zobrazujícího seznam polí a umožňujícího práci s nimi /listování v jejich seznamu, přidávání a ubírání polí, tisk vybraného pole, přechod k editaci zvoleného pole a další/.

Návrh programu pomocí abstrakcí má řadu výhod, neboť umožňuje navrhovat program krok za krokem po malých přehledných částech. V případě, že se budeme držet zásad strukturovaného programování, které jsou dnes dostatečně rozšířené, zajišťuje i přehledné ladění a snadnou údržbu programu. Na druhé straně, jestliže tyto zásady porušíme, můžeme získat program daleko nepřehlednější než pomocí dosud běžných návrhových prostředků.

Ideální je navrhnout program tak, aby vazby pomocí abstrakcí mezi jednotlivými částmi programu byly co nejprůzračnější a aby každá samostatná část programu /ať jde o kořen nebo abstrakci/ odpovídala přibližně velikosti obrazovky terminálu.

Zápis programu pomocí R-grafů

Program je možné napsat jako libovolnou kombinaci textu v podobě

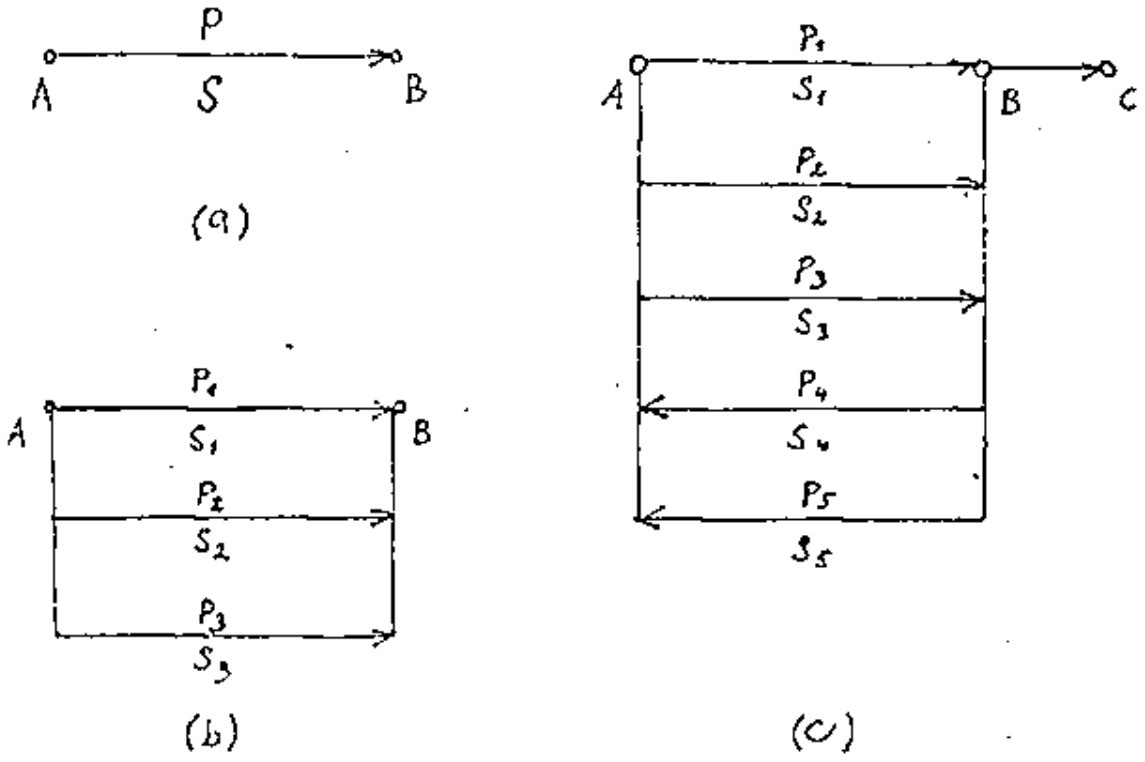
programových příkazů a R-grafů.

R-grafy /R znamená ruské/ jsou grafy, jejichž vrcholy /podobně jako u grafu toku řízení/ odpovídají stavům programu /resp. jejich výpočtu/. Významnou informaci nesou hrany grafu. Každé hraně může být přiřazena skupina podmínek a blok příkazů. Přechod z jednoho vrcholu do druhého může být realizován, existuje-li mezi nimi hrana, pro kterou platí, že jsou splněny podmínky přiřazené této hraně /zapisující se nad ní/. Při přechodu je vyplněn blok příkazů, přiřazený této hraně /zapisující se pod ní/. Přitom pro zápis libovolné podmínky nebo programového příkazu můžeme použít abstrakci.

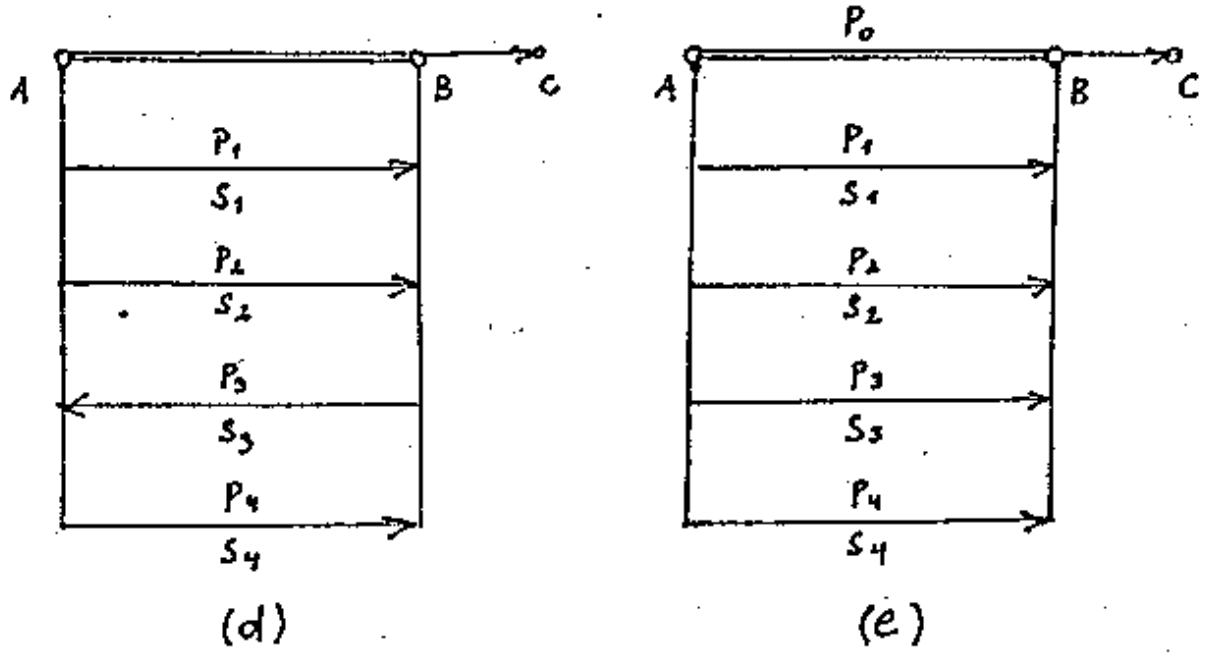
Tím, že je možné jedné hraně přiřazovat několik podmínek a zejména několik příkazů, dochází k prostorově efektivnímu a přitom přehlednému zápisu programu.

R-grafy jsou dobře uzpůsobeny pro zobrazení na terminálu /i na alfanumerické tiskárně/ díky "pravoúhlému" vedení hran grafu. U personálních počítačů je pochopitelně obraz programu díky lepším terminálům značně kvalitnější.

Používané typy grafických struktur jsou uvedeny na obr.1.



Obr. 1



obr. 1

Základním prvkem všech grafových struktur je jednoduchá selekce /obr. a/. Z vrcholu A do vrcholu B přejdeme, jestliže je splněna skupina podmínek, přiřazená této hraně /P/. Při přechodu se realizuje blok příkazů /S/. Nepodmíněný příkaz /resp. blok příkazů/ z ní vznikne vynecháním skupiny podmínek /P/.

Vodorovné hrany grafu můžeme sdružovat pomocí svislých hran, které mají pouze pomocný význam. Na obr. b/ je znázorněna několikanásobná selekce. Všechny vodorovné hrany znamenají možný přechod z vrcholu A do B. Vyhodnocují se postupně shora dolů. Je-li skupina podmínek přiřazená některé z nich splněna, provede se blok příkazů, přiřazený příslušné hraně a výpočet přejde do vrcholu B.

Směr šipky některých vodorovných hran v příkazu několikanásobné selekce můžeme obrátit - tím vznikne cyklus podobný typu repeat /obr. c/. Nacházíme-li se ve vrcholu A, provede se vyhodnocení šipek vedoucích z A do B stejným způsobem jako u několikanásobné selekce. Po přechodu do vrcholu B se nejprve zkoumá, zda není splněna některá z podmínek patřící hraně vedoucí z B do A. Jestliže ano, vrátíme se do vrcholu A. Jestliže nikoliv, pokračujeme /je-li splněna příslušná podmínka/ do vrcholu C.

Posloupnost hran vedoucí z A do B /resp. z B do A/ se přitom vyhodnocuje postupně, shora dolů. Z toho vyplývá, že obě posloupnosti mohou být promíchané, aniž by to měnilo význam programové struktury. Pro větší přehlednost je však lepší jejich hrany oddělit.

Na obrázku d/ je uveden další typ cyklu. Dvojitá hrana není "plnoprávnou" hranou grafu, je to takzvaná technologická hrana, pomocí níž realizujeme přechod do toho samého vrcholu /např. z A do A/. Používá se, aby bylo možné udržet "hranatý" zápis grafu. K technologické hraně nesmí být u příkazu cyklu přiřazena žádná podmínka. Významné hrany, které jsou k ní přiřazeny, mohou vést libovolným směrem /u hran typu z A do A to nemá význam/. Vyhodnocují se opět shora dolů - je-li podmínka přiřazená některé z nich splněna, provede se blok příkazů přiřazený příslušné hraně a pokračuje se ve vyhodnocování podmínek od první hrany cyklu /byl uskutečněn přechod z A do A/. Teprve v okamžiku, kdy není ani jedna z podmínek u hran cyklu splněna, pokračujeme /je-li splněna příslušná podmínka/ do bodu B.

Poslední používaná struktura se nazývá klíč /obr.e/. Je uvedená opět technologickou hranou, které je však přiřazena podmínka nutná k tomu, aby ke zpracování klíče došlo. Tato struktura je v různých programovacích jazycích využita různým způsobem. /Pro Pascal je použita k implementaci příkazu case, zatímco v jazyce C slouží k realizaci příkazu cyklu typu for/.

V neodladěném programu nebo programu, který zpracovává nekorektní data se při průchodu grafem snadno můžeme dostat do vrcholu, z něhož nebudeme mít žádné korektní pokračování. V takovém případě se automaticky použije tzv. systémový východ. Je mu přirazen standardní význam: goto M, kde M je předem zadané návěští pro ošetření možné chyby. Toto návěští je možné průběžně speciálním příkazem měnit.

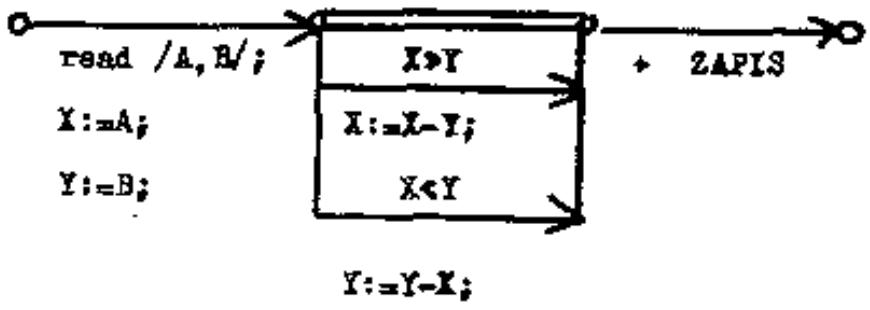
Programová realizace R-grafů /řídící se filosofií "raději více než méně"/ umožňuje použít ještě další prostředky. Můžeme do grafů vkládat návěští a skoky na ně. Dále je možné do grafu vložit libovolný text. To umožňuje např. v PL/1 vynutit zadání záhlaví cyklu typu for. Použití těchto rozšíření je značně diskutabilní, protože vede ke značné nepřehlednosti a ztrátě kontroly správnosti zápisu.

Příklad použití R-grafu pro zápis programu je uveden na obr.2. Jde o příklad hledání největšího společného dělitele. Na obr. a/ je uvedena verze v Pascalu, na obr. b/ je použito R-grafu, kde pro realizaci zápisu výsledku je použita abstrakce.

```
program NSD  /input, out:  :/;
var A, B, X, Y: integer;
begin
       read  /A, B/;
       X:=A;
       Y:=B;
       while  X<>Y do
           if  X>Y  then X:=X-Y else Y:=Y-X;
       writeln  /'NSD CISEL', A, 'A ', B,  'JE ', X/
end.
```

obr. a/

```
program NSD  /input, output/;
var A, B, X, Y: integer;
```



/abstrakce/ ZAPIS

```
writeln  / 'NSD  CISEL', A, 'A ', B 'JE ', X/
```

obr. b/

Obrázek č. 2

Způsob překladu R-grafů do jednotlivých programovacích jazyků se případ od případu liší. Zdá se mi, že je veden dvěma protichůdnými idejemi :

a/ snahou o jednoduchost realizace /která převládá např. u PL/1/,
b/ snahou o podporu využití co největšího množství jazykových konstrukcí pomocí grafů /převládající u jazyka C/.

Tato implementační rozkolísanost /nebo pružnost, chcete-li/ vede k tomu, že by uživatel měl znát způsob realizace grafických prostředků v programovacím jazyce, který používá. Současné systémy k tomu neposkytují přílišnou oporu /částečně je přehled realizace

R-grafů uveden v manuálech RTK/. Proto do systému RTK pod DOSem bude u typu práce /která určuje volbu programovacího jazyka/ rozšířen Help o přehled realizace grafických struktur.

Ladění na grafech

Jednou z nejsilnějších stránek R-technologie je možnost ladění programů na grafech, kterou systém RTK implementovaný pod CMS poskytuje v jazyce C. Při tomto způsobu ladění se nám na obrazovce zobrazuje zápis programu a my se během výpočtu můžeme zastavit v libovolném vrcholu grafu /resp. na vrcholu mezi dvěma textovými příkazy/ a můžeme prohlížet a měnit obsah jednotlivých proměnných. Jestliže při ladění přejdeme na zpracování abstrakce, automaticky se nám obrazovka přepne na zobrazení příslušné abstrakce /pokud volba jejího zobrazení není potlačena/. Systém ladění na grafech je velmi efektivní. Díky tomu lze při ladění počítat se snížením pracnosti a značnou časovou úsporou.

Systém ladění na grafech v RTK pro CMS funguje pouze pro jazyk C, protože pro jazyk C tvůrci RTK dělali vlastní překladač. DOS4 má tu výhodu, že pro řadu vyšších programovacích jazyků používá standardní ladicí systém HULDA. Ten bude ve verzi RTK pod DOSem použit, takže bude možno ladit na grafech pomocí prostředků HILDy prakticky ve všech důležitějších programovacích jazycích.

Realizace systému RTK v DOSu

Realizace RTK v CMS připomíná programy napsané pro personální počítače nejen moderním přístupem k uživateli, ale bohužel i v tom, že některé části systému nejsou zabezpečeny před chybou uživatele, takže například při volbě nesprávné akce může dojít k vypadnutí ze systému nebo jeho zablokování spojenému se ztrátou dat. V některých případech dokonce musíme provést znovuzavedení operačního systému CMS, abychom zabránili uložení nesprávných dat /a tím přemazání korektních/. Těmto nedostatkům se chceme ve verzi RTK pod DOSem vyhnout.

Použití programového systému RTK v DOSu je pro uživatele spojeno s jednou podstatnou výhodou. Operační systémy typu CMS nebo MSDOS dávají k dispozici relativně stálé vlastní výpočetní prostředí /ať virtuální nebo reálné/, což pro DOS4 tak zcela neplatí. Systém RTK však takové prostředí vytváří - prakticky dává uživateli

k dispozici plně soběstačný virtuální počítač RTK, aniž by se uživatel musel zabývat příkazy DOSu.

Závěr

Srovnáme-li R-technologii s existujícími programovacími technologiemi, je vidět značná síla, ale i jistá primitivnost a nevyváženost R-grafů při realizaci řídicích programových struktur. Tato nevýhoda je vyvážena tím, že R-grafy - na rozdíl od ostatních typů grafů, používaných jinými programovacími technologiemi - jsou nejlépe přizpůsobeny zobrazení na terminálu.

Pragmatický přístup k jazykové realizaci souvisí s tím, že R-technologii neodpovídá žádná teorie, není pro ni vypracována žádná programovací metoda, pouze užívá základní myšlenky strukturovaného programování. To na jedné straně představuje výhodu - před začínajícím uživatelem nestojí žádná "metodologická bariéra", která je například u Jacsonovy programovací technologie poměrně vysoká, na druhé straně uživatel při používání poměrně silných programových nástrojů nemá metodologickou podporu a není-li zkušeným programátorem, může produkovat velmi zapeklité a těžko odladitelné programy. Seriózní používání RTK však zřejmě vyžaduje použít jistou metodiku přístupu k návrhu programu - nejlépe podle svyklostí té které instituce, uzpůsobenou podmínkám interaktivního přístupu R-technologie.

Vcelku je možno říci, že jde o dobře použitelnou programovací technologii, která při dobrém přístupu programátora zajišťuje kvalitní návrh, ladění a údržbu programu. Zejména při ladění a údržbě lze počítat se značnými časovými úsporami programátorů.

Literatura

/1/ O.J.Dahl, E.W.Dijkstra, C.A.R.Hoare:Structured programming, New York, 1972

/2/ I.V.Velbickij:Technologija programirovanija,Kijev, 1984

/3/ manuály a dokumentace pro RTK v CMS, Kijev, 1989