Situační programování a produkční systémy

Radomír Jeliga

1. Situační programování

Termín situační programování se objevuje v souvislosti se známou Minského charakteristikou [1] vývoje programovacích jazyků. Ve škále dané příkazy "dělej co máš přikázáno", "dělej co můžeš" a "dělej něco smysluplného", které určují kompetence počítače, odpovídá situační programování druhému stupni. Ono "dělej co můžeš" se zde chápe jako "vyhodnoť aktuální situaci a reaguj na ni".

Situační programování je charakteristické základní změnou přístupu k řešenému problému: programátor se soustředí především na popis funkčních a logických vztahů, platících v existujícím, či vytvářeném systému, a práci s těmito vztahy přenechává vnitřnímu mechanismu použitého programovacího prostředku. Označení "situační" se objevuje díky skutečnosti, že především ony vzpomínané funkční vztahy bývají obvykle vyjádřeny ve formě popisu nějaké situace a odpovídající reakce na ni.

Možnost převedení části práce (v některých případech většiny) na vnitřní mechanismus programovacího prostředku již sama o sobě výrazně zvyšuje produktivitu práce programátora. V kontextu současných trendů ve výpočetní technice však nabývají na významu i další vlastnosti situačního programování. V tomto příspěvku se pokusím nastínit alespoň nejdůležitější z nich.

2. Produkční programování a produkční systémy

Místo dalších vágních charakteristik si ukážeme, jak jsou principy a postupy situačního programování realizovány v tzv. produkčním programování, t.j. v programování založeném na použití produkčních jazyků a pomocí nich vytvořených produkčních systémů (PS).

Pro úplnost je třeba podotknout, že následující charakteristiky neplatí vždy pro všechny PS. Principům situačního programování odpovídají spíše daty řízené PS, a těch si budeme všímat především. Hlavními nástroji pro tvorbu PS tohoto typu jsou jazyky OPS4 a OPS5 [4] a z nových produktů pak jazyk OPS83 [5]. Jediným domácím zástupcem této větve je produkční jazyk TOPS [6], jehož syntaxe budeme v následujících příkladech používat.

PS se objevují již v počátcích využívání výpočetní techniky. Jejich myšlenka je velmi prostá a analogie PS můžeme nalézt dokonce i v živé přírodě.

PS je tvořen neuspořádanou množinou produkčních pravidel, daty soustředěnými v pracovní pracovní paměti a tzv. inferenčním mechanismem.

Data jsou organizována ve formě objektů. Každý objekt patří do nějaké třídy (class), jejíž deklarace určuje strukturu objektu. Ta je dána počtem a typy tzv. atributů. Objekty mají globální povahu jsou "viditelné" pro všechna pravidla.

Produkční pravidla se skládají z tzv. levé (situační) a pravé (akční) strany. Levá strana specifikuje situaci, při jejímž výskytu má být pravidlo aktivováno (t.j. připraveno k provedení), pravá strana určuje akce, které budou provedeny při použití pravidla.

Specifikace levé strany mají formu testu na existenci, či neexistenci nějakých objektů v pracovní paměti. Mezi atributy těchto objektů mohou být specifikovány různé vztahy.

Akce pravé strany umožňují měnit výchozí situaci modifikací, vytvářením a rušením objektů pracovní paměti. Kromě toho mohou být na pravé straně pravidla prováděny vstupy a výstupy dat, volány externí funkce atd.

Pravidla jsou vzájemně nezávislá. Jedno pravidlo může způsobit aktivaci jiného pravidla pouze prostřednictvím zásahu do pracovní paměti (např. vytvoření objektu), tedy vytvořením takové situace, na niž je druhé pravidlo citlivé. Tato zdánlivá neohrabanost má velmi příznivý vliv na modulárnost a modifikovatelnost PS.

Inferenční mechanismus - hnací motor celého PS - pracuje v cyklu, který se skládá ze tří fází. V první fázi je situace v pracovní paměti porovnána se specifikacemi levých stran všech pravidel. Ta pravidla, jejichž specifikace odpovídají aktuální situaci, jsou zařazena do tzv. konfliktní množiny. V druhé fázi se na základě určité strategie z této množiny vybere jedno pravidlo. Toto pravidlo (akce pravé strany) se v rámci třetí fáze provede. Provedením pravidla se může změnit situace v pracovní paměti, a cyklus pokračuje první fází. Pokud změna nezpůsobí aktivaci nějakých pravidel (nebo deaktivaci pravidel z konfliktní množiny), vybere se k provedení další z pravidel připravených v konfliktní množině.

Vzhledem k omezenému rozsahu tohoto příspěvku není možné podrobně popisovat syntaxi jazyka TOPS. V následujícím příkladu se proto musíme spokojit spíše s intuitivním přístupem.

Specifikujme pravidlo, které zajistí naplnění poptávky na základě nabídky:

```
RULE nabídka_poptávka      ; uspokojení poptávky
FOR _p požadavek           ; existuje objekt třídy požadavek
    @typ        = _t       ; na výrobek určitého typu
```

```
    @max_cena     = _mc     ; s danou max. přípustnou cenou
    @zákazník     = _iz     ; identifikace zákazníka
FOR _n nabídka              ; existuje objekt třídy nabídka,
    @typ          = _t      ; kde typ výrobku
    @cena         <= _mc    ; a cena odpovídají požadavku
    @výrobce      = _iv     ; identifikace výrobce
DO                          ; oddělovač situační a akční části
MAKE objednávka             ; vytvoř objekt třídy objednávka
    @zákazník     = _iz     ; identifikace zákazníka
    @výrobce      = _iv     ; identifikace výrobce
    @zboží        = _t      ; typ zboží
REMOVE _p                   ; odstraň uspokojený požadavek
REMOVE _n                   ; zruš nabídku
```

Toto pravidlo vyhledá všechny dvojice navzájem si odpovídajících nabídek a požadavků, vytvoří objednávku na požadovaný výrobek a odstraní objekty představující uspokojený požadavek a uplatněnou nabídku z pracovní paměti. Strukturu použitých tříd objektů zde nebudeme rozebírat - podrobnosti lze nalézt v [6].

3. Použití produkčních systémů

Produkční jazyky (a pomocí nich napsané produkční systémy) jsou používány především v oblasti umělé inteligence, a to hlavně pro tvorbu expertních systémů [7]. Zde se využívá schopnosti produkčních pravidel přímočarým způsobem zachytit expertovy znalosti. Snadnost, s níž je možno specifikovat i poměrně složité situace, činí z produkčních jazyků velmi silný nástroj pro psaní různých monitorovacích a řídících systémů. Produkční jazyky OPS83 a TOPS jsou navrženy tak, aby mohly spolupracovat s procedurálním okolím. Oba mají implementovány běžně používané datové typy a mohou volat procedury a funkce napsané v procedurálních programovacích jazycích. PS napsané v jazyku TOPS mohou být dokonce samy volány jako procedury a mohou volat podřízené PS. Tyto vlastnosti výrazně rozšiřují oblast nasazení obou prostředků. Kombinace procedurálního a produkčního programování umožňuje použití PS i v těch úlohách, v nichž je značná část problému řešena procedurálně. PS pak řeší právě tu část, která je pro jeho nasazení vhodná.

Všimněme si zde ještě dalších vlastností, které jsou nesporně velmi užitečné při zpracování úloh, u kterých je nutno počítat s častými změnami, rozšiřováním a úpravami řešení. Typickými příklady jsou návrhové a plánovací systémy, oblast řízení a managementu atd.

V případě nutnosti zásahů do hotového ogramu velmi zjednušuje situaci především:

- soustředění souvisejících akcí na dobře ohraničitelné místo (pravidlo, případně skupina pravidel),
- odolnost proti chybám způsobeným modifikací - přidáním pravidla postihujícího specifickou situaci není chování systému (s výjimkou oné specifické situace) nijak ovlivněno,
- schopnost samodokumentace - při vhodně zvolených názvech pravidel, tříd a atributů je produkční program velmi dobře čitelný,
- možnost ladění na velmi obecné úrovni, kdy není většinou nutné uvažovat v elementárnějších pojmech než pravidlo a objekt. Tato možnost se pochopitelně uplatní i při samotném vývoji systému.

Ilustrujme si na úpravě pravidla nabídka_poptávka alespoň první dva body našeho výčtu. Doplňme pravidlo tak, aby se pro určitý požadavek vybrala nabídka s nejnižší cenou:

Kladné specifikace (FOR ..) situační části pravidla doplníme jednou zápornou specifikací, která nepřipustí vytvoření objednávky pro určitý výrobek, jestliže v pracovní paměti existuje nabídka s výhodnější cenou. Výsledné pravidlo bude mít tvar

```
RULE nabídka_poptávka2      ; uspokojení poptávky
FOR _p požadavek            ; existuje objekt třídy požadavek
  @typ            = _t      ; na výrobek určitého typu
  @max_cena       = _mc     ; s danou max. přípustnou cenou
  @zákazník       = _iz     ; identifikace zákazníka
FOR _n nabídka              ; existuje objekt třídy nabídka,
  @typ            = _t      ; kde typ výrobku
  @cena     {_cn,<=_mc}     ; a cena odpovídají požadavku
  @výrobce        = _iv     ; identifikace výrobce
NOT nabídka                 ; neexistuje nabídka
  @typ            = _t      ; stejného výrobku
  @cena           < _cn     ; s nižší cenou
DO                          ; oddělovač situační a akční části
MAKE objednávka             ; vytvoř objekt třídy objednávka
  @zákazník       = _iz     ; identifikace zákazníka
  @výrobce        = _iv     ; identifikace výrobce
  @zboží          = _t      ; typ zboží
REMOVE _p                   ; odstraň uspokojený požadavek
```

REMOVE _n ; zruš nabídku
Konstrukce {_cn,< =_mc} znamená kombinaci podmínky a přiřazení proměnné.

Stejného efektu, jako úpravou pravidla nabídka_poptávka, bychom dosáhli i přidáním pravidla nabídka_poptávka2, při ponechání původního pravidla. Pravidlo nabídka_poptávka2 je specifičtější (má bohatší situační část) a taková pravidla mají při výběru přednost.

4. Závěr

V současné době jsme svědky pronikání výpočetní techniky do oblastí, které jsou typické obtížnou algoritmizací, složitostí řízení, častými modifikacemi řešení a značným rozsahem výsledného programu. Situační programování v prostředí produkčních systémů má - díky svým vlastnostem - dobré předpoklady stát se významným nástrojem pro řešení úloh tohoto typu.

Literatura:

[1] Minski, M.: Why people think computers can't., in M.I.T. Technology review, 86, 1983, No. 3.

[2] Nillson,N.,J.: Principles of artificial intelligence. Tioga Publishing Co., Palo Alto 1980.

[3] Popper, M., Kelemen, J.: Expertné systémy. Alfa Bratislava, 1989.

[4] Forgy,C.L.: OPS5 User's Manual. Výzk. zpráva CMU-CS-81-135. Carnegie-Mellon 1981.

[5] Forgy,C.L.: The OPS83 Report. Výzk. zpráva CMU-CS-84-133. Carnegie-Mellon 1984.

[6] Drápal,A., Jeliga,R., Bohuslav,Z., Vosátka,K.: Produkční jazyk TOPS/PC, Praha 1991.

[7] Brownston,L., Farrel,R., Kant,E., Martin,N.: Programming Expert Systems in OPS5. Addison-Wesley 1985.

Autor: RNDr. Radomír J e l i g a
VÚMS, Lužná 2, 160 00 Praha 6
tel. 02 - 366 251 kl. 301, 359