CHARAKTERISTIKA JAZYKA LISP

Ľudovít Molnár

1. Úvod

Jazyk lisp patrí medzi prvé "vyššie" programovacie jazyky. Vznikol už v roku 1960 a skutočnosť, že prežil, ba že sa stal jedným zo základných jazykov počítačov 5. generácie, možno považovať za jednu z "charakteristík" tohoto jazyka. Jeho názov je odvodený z anglického LISt Processor - spracovateľ zoznamov, z čoho vyplýva jeho ďalšia charakteristika : zoznam je základnou údajovou štruktúrou, ktorú jazyk lisp používa, a to nielen na reprezentáciu údajov, ale aj na zápis programu. Ďalšou charakteristikou jazyka lisp je spôsob opisu výpočtového procesu : výpočtový proces opisujeme pomocou kompozície funkcií. Tento spôsob programovania sa často nazýva funkcionálne programovanie a jazyk lisp funkcionálny jazyk. Oblasť použitia jazyka lisp je predovšetkým v nenumerickej oblasti. Považuje sa za základný jazyk v oblasti umelej inteligencie. Lenže práve "cez umelú inteligenciu", čoraz širšie využívanie jej prostriedkov a metód, zvyšuje sa záujem o jazyk lisp aj v širšej počítačiarskej verejnosti.

To je aj jeden z dôvodov, prečo sa ním zaoberáme aj na terajšom seminári.

Charakteristík by sa pochopiteľne dalo napísať aj viac. Nebudeme ich ďalej vymenúvať, pozrime sa na ne, cez opis základných konštrukcií jazyka lisp. Jednou z negatívnych "charakteristík" jazyka lisp je skutočnosť, že napriek jeho "veku" a napriek viacerým pokusom, neexistuje zatiaľ jeho všeobecne uznávaná norma. Odchýlky sú v syntaxe, sémantike, ale predovšetkým v bohatstve ponúkaných prostriedkov. Princípy sú však rovnaké a na tie sa v ďalšom zameriame.

2. Údajové objekty jazyka lisp

2.1 S-výraz

Jazyk lisp pracuje s jediným údajovým typom, a to je symbolický výraz, ktorý sa zvyčajne označuje skrítene ako s-výraz. Strukturálne môže byť organizovaný ako atóm, zoznam alebo bodka dvojica.

2.2 Atóm

Základným údajovým objektom jazyka lisp je atóm. Ide o objekt, ktorý je prostriedkami jazyka lisp nedeliteľný. Rozorňáváme dva druhy atómov : numerické a nenumerické. Numerický atóm označuje hodnotu, reprezentovanú samotným atómom. V základnej verzii je to postupnosť desiatkových číslic, pred ktorou môže byť znamienko, napr. 5, 123, -55. Nenumerický atóm pozostáva z postupnosti písmen a číslic, ktorá začína písmenom, napr. PRIEMER, TEPLOTA, R1. O hodnotách a sposobe ich viazania s atómom si povieme neskor.

2.3 Zoznam

Zoznam je štruktúrovaný údajový objekt jazyka lisp, ktorý má tvar

(sv1 sv2 ... svn),

kde sv1, sv2 ... svn sú (zatiaľ) atómy alebo zoznamy. Zoznam je teda postupnosť atomov alebo zoznamov, oddelených medzerou a uzatvotená v okrúhlých zátvorkách. Ako príklady možno uviesť : (A), (JANO VELKY), (A (B (C) D)).

Počet prvkov zoznamu udáva jeho dĺžku. Speciálnym zoznamom je zoznam s nulovou dĺžkou, tzv. prázdny zoznam, ktorý má tvar (). Na označenie prázdneho zoznamu je v lispe vyhradený špeciálny atóm NIL. NIL je teda atómom a súčasne zoznamom (špeciálnym - prázdnym).

2.4 Bodka = dvojica

Bodka dvojica je symbolický výraz tvaru

(sv1.sv2),

kde sv1, sv2 sú symbolické výrazy jazyka lisp.

Ak sa pozrieme na zápisy zoznamu a bodka dvojica, už na prvý pohľad vidieť, že sa jedná o "príbuzné" zápisy jazyka lisp. Vzťah medzi zoznamom a bodka dvojicou vyjadruje následujúci rekurentný vzorec :

f() = NIL
f (sv1 sv2 ... svn) =)sv1.f (sv2 ... svn)

Tedn napr. zoznam (A B) možno na zápis pomocou bodka dvojice upraviť takto :

f (A B) = (A.f (B) = (A.(B.f())) =)A.(B.NIL))

Špeciálne, úpravou jednoprvkového zoznamu, napr. s prvkom A dostaneme ekvivalenciu zápisov (A) a (A.NIL).

3. Základné operácie s s-výrazmi

Všetky operácie v jazyku lisp sa uskutočňujú pomocou funkcií. Každá verzia jazyka lisp poskytuje používateľovi určitú množinu funkcií i spôsob, ako vytvárať ďalšie funkcie. Na zápis funkcií používame zoznam. Napr. funkciu f, aplikovanú na argumenty a1, a2, ... an zapíšeme v tvare

(F A1 A2 ... AN)

Takýto zápis nazývame v jazyku lisp forma. Pretože celý výpočtový proces opisujeme v jazyku lisp ako kompozíciu funkcií, spôsob vyhodnotenia funkcií má v lispe svoju osobitosť v tom, že ak to nie je špeciálne povedané, funkcie začíname vyhodnocovať tak, že najprv sa vyhodnotia argumenty.

Špeciálnou formou je atóm. S každým atómom je asociovaná hodnota : s numerickým atómom je to hodnota, ktorú sám reprezentuje, atómy s NIL reprezentujú logické hodnoty pravda a nepravda, s nenumerickými atómami sú viazané hodnoty špeciálnym

spôsobom. Atómy, spolu so svojimi asociovanými hodnotami, tvoria kontext danej formy, v rámci ktorého sa daná forma vyhodnocuje.

3.1 Sprístupňovanie časti s-výrazov

Na sprístupnenie prvého prvku zoznamu používame v lispe funkciu CAR. Na sprístupnenie zvyšku zoznamu bez prvého prvku používame funkciu CDR. Ide o jednoargumentové funkcie a hodnotou argumentu je zoznam. Funkcia CAR dáva ako výsledok prvý prvok hodnoty argumentu, funkcia CDR zvyšok hodnoty argumentu bez prvého prvku, pričom hodnota argumentu musí byť zoznam. Ako príklady možno uviesť :

X	(CAR X)	(CDR X)
(A)	A	() alebo NIL
(A B)	A	(B)
((A B) C)	(A B)	(C)
()	nedefinované	nedefinované
A	nedefinované	nedefinované

3.2 Vytváranie s-výrazov

Základnou funkciou na vytváranie zoznamov je funkcia CONS. Je to dvojargumentová funkcia, ktorá dáva ako výsledok bodku dvojicu z hodnot argumentov. Ako príklady možno uviesť :

X	Y	(CONS X Y)
A	B	(A.B)
A	NIL	(A.NIL) = (A)
A	(B)	(A.(B)) = (A B)

3.3 Predikáty ATOM a EQ

Predikáty ATOM a EQ umožňujú zisťovať určité vlastnosti symbolických výrazov. Predikát ATOM je jednoargumentovou funkciou, ktorá dáva výsledok T, teda logickú hodnotu PRAVDA, ak hodnota argumentu je atóm a hodnotu NIL, teda logickú hodnotu NEPRAVDA, v opačnou prípade. Špeciálnym

prípadom je prázdny zoznam, ktorý, ako už vieme, môže byť reprezentovaný atómom NIL a práve túto reprezentáciu akceptuje aj funkcia ATOM. V prípade, že hodnotou argumentu je prázdny zoznam alebo atóm NIL, funkcia ATOM dáva ako výsledok hodnotu T. Ako príklady možno uviesť :

X	(ATOM X)
A	T
(A)	NIL
25	T
NIL	T

Predikát EQ umožňuje zistiť, či sa dva atómy rovnajú. Ide o dvojargumentovú funkciu, ktorá dáva ako výsledok hodnotu T, ak hodnoty argumentov sú rovnaké atómy. Ako príklady možno uviesť :

X	Y	(EQ X Y)
A	A	T
A	10	NIL
A	B	NIL
A	(B)	nedefinované

3.4 Kompozícia funkcií

Ako sme už povedali, výpočtový proces opisujeme v jazyku lisp pomocou kompozície afunkcií. Ako príklady kompozície už zavedených funkcií možno uviesť :

Nech s atómami X a Y sú asociované následujúce hodnoty : (A B) s X, (C D (E F)) s Y.

forma	vyhodnotenie
(ATOM (CAR X))	T
(CONS (CAR X) (CDR Y))	(A D (E F))
(CAR (CDR Y))	D
(CDR (CDR X))	NIL
(EQ (CAR X) (CAR Y))	NIL

Aj z uvedených príkladov vidieť, že vyhodnotenie formy sa robí tak, že sa najprv vyhodnotia argumenty, pričom toto pravidlo platí pre každú aplikáciu funkcie, teda pre ľubovoľnú "hĺbku" kompozície.

4. Lambda výraz

Až doteraz sme predpokladali, že s nenumerickým atómom je asociovaná hodnota. Lambda-výraz nám udáva spôsob, ako z formy, obsahujúcej nenumerické atomy, vytvoriť funkciu práve tých premenných, ktoré sa vyskytujú v danej forme ako nenumerické atómy a tiež spôsob, ako spojiť s týmito atómami hodnotu. Ak sa pozrieme napr. na formu z predchádzajúceho príkladu (EQ (CAR X) (CAR Y)), v ktorej sa nachádzajú dva nenumerické atómy X, Y, tak možno na prvý pohľad túto formu chápať ako funkciu dvoch premenncý X, Y. Pravda, potom už nebude platiť náš predpoklad, že s atómami X, Y sú asociované hodnoty spôsobom, ako sme to uviedli, ale spôsob asociovania hodnot s atómami (premennými) funkcie treba definovať. Vytvorenie funkcie z formy robíme pomocou lambda-výrazu, ktorý má pre uvedenú formu tvar

(LAMBDA (X Y) (EQ (CAR X) (CAR Y))),

v ktorom premenné X, Y nazývame lambda-premenné, niekedy tiež viazané premenné. V programátorskom prostredí je snáď najpriliehavenším názvom uvedených objektov formálne parametre funkcie.

Vo všeobecnosti má funkcia, vytvorená z formy FO, tvar

(LAMBDA (X1 X2 ... Xn) FO),

kde zoznam (X1 X2 ... Xn) nazývame zoznam lambda premenných, pričom zoznam premenných je usporiadaný.

Abysme funkciu, vytvorenú pomocou lambda-výrazu, mohli používať rovnakým spôsobom ako ktorúkoľvek inú funkciu, potrebujeme určiť spôsob viazania premenných funkcie (lambda-premenných) s argumentami. Ten je definovaný na základe pora-

dia premenných v zozname lambda premenných a poradia argumentov v zozname argumentov takto :

Ak máme funkciu, vytvorenú pomocou lambda výrazu so zoznamom lambda premenných (X1 X2 ... Xn), aplikovať na argumenty A1, A2, ... An, vytvoríme z nich zoznam argumentov tak, aby zodpovedajúce si premenné a argumenty boli na rovnakom mieste v zozname premenných a v zozname argumentov. Aplikácia funkcie má tvar :

((LAMBDA (X1 X2 ... Xn) FO) (A1 A2 ... An)),

kde zoznam (A1 A2 ... An) je zoznam argumentov.

Vyhodnotenia aplikácie funkcie, reprezentovanej lambda výrazom, sa uskutočňuje takto :

- vyhodnotia sa argumenty
- urobí sa väzba alebo asociácia lambda premenných a argumentov
- aplikuje sa daná funkcia.

Nech sú s atómami U, V spojené nasledujúce hodnoty :

(A B) s U, (C D (E F)) s V.

Potom vyhodnotenie aplikácie funkcie

(((LAMBDA (X Y) (EQ (CAR X) (CDR Y)) U V)

prebieha takto :

Najprv sa vyhodnotia argumenty, to znamená U na (A B), V na (C D (E F)). Vzhľadom na tieto hodnoty sa urobí asociácia premenných X na hodnotu argumentu U, to znamená na (A B), Y na hodnotu argumentu V, to znamená (C D (E F)) a vyhodnotí sa forma v lambda výraze, v našom prípade forma

(EQ (CAR X) (CDR Y))

Opäť ide o aplikáciu funkcie, preto najprv vyhodnotíme argumenty (CAR X) a (CDR Y), čo sme už urobili, ako aj vyhodnotenie celej formy.

5. Program

Programom v jazyku lisp rozumieme jeden alebo viac aplikácií funkcií. Realizáciu programu zabezpečuje programový systém, ktorý nazývame lispovský systém alebo systém jazyka lisp. Systém jazyka lisp zabezpečuje postupné vyhodnotenie aplikácií funkcí tak, ako sú v programe napísané (sekvencia v bežných programovacích jazykoch). Keďže niektoré základné funkcie jazyka lisp poznáme, mohli by sme už písať programy na riešenie jednoduchých úloh. Potrebujeme však určiť spôsob zadávania "prvotných" hodnôt, pre ktoré sa má program realizovať. Problém, vzhľadom na to, čo sme si doteraz povedali, spočíva v tom, že pred aplikáciou funkcie sa vyhodnocujú argumenty. Potrebujeme teda určiť spôsob, ako zabrániť vyhodnocovaniu prvotných argumentov. Keďže prvotné argumenty sa nachádzajú na najvyššej úrovni aplikácie funkcie, tzv. nultá úroveň, potrebujeme zabrániť vyhodnocovaniu argumentov funkcie na nultej úrovni. Na riešenie tohoto problému existuje a používa sa viac spôsobov, ktoré si stručne napíšeme.

5.1 Funkcie eval, evalquote, quote

Najprirodzenejší spôsob, ako zabrániť vyhodnocovaniu argumentov na nultej úrovni aplikácie funkcie, je zavedenie špeciálnej funkcie s vlastnosťami :

- argument sa nevyhodnocuje
- funkcia vracia ako výsledok samotný argument.

Takúto funkciu jazyk lisp poskytuje a nazýva sa QUOTE. Pre ľubovoľný argument A dáva (QUOTE A) hodnotu A, pričom argument A sa nevyhodnocuje.

Uvedený spôsob zápisu funkcií sa používa v tzv. eval-lispe, pretože na realizáciu programu používa systém jazyka lisp špeciálnu systémovú funkciu EVAL.

Zabrániť vyhodnocovaniu argumentov na nultej úrovni aplikácie funkcie možno aj špeciálnym zápisom tak, že aplikácii

ciu funkcie na nultej úrovni zapisujeme ako dvojicu symbolických výrazov - bežný prefixový zápis, ako ho poznáme z matematiky, pričom prvým výrazom je funkcia, druhým zoznam argumentov, napr. CAR ((A.B)), EQ (A B).

Uvedený spôsob zápisu funkcií sa používa v tzv. eval-quote lispe.

5.2 Pomenovanie funkcií

Jazyk lisp umožňuje funkcie reprezentované pomocou lambda výrazu pomenovať. Používame na to špeciálnu systémovú funkciu DEFINE, čím sa vyjadruje skutočnosť, že funkciu s daným menom vlastne definujeme. Funkcia DEFINE je jednoargumentová funkcia, ktorej argumentom je zoznam definicií funkcií, pozostávajúci z dvojprvkového zoznamu, pričom prvým prvkom je meno definovanej funkcie, druhým definovaná funkcia, reprezentovaná lambda výrazom. Funkcia DEFINE vracia ako výsledok zoznam atómov - mien definovaných funkcií. Všeobecný zápis má tvar :

```
(DEFINE (QUOTE (
      (meno1 F1)
      (meno2 F2)
      ...
)))   (meno Fk)
```

Ako príklad možno uviesť :

```
(DEFINE (QUOTE (
     (PRVY (LAMBDA (Z) (CAR Z)))
     (DRUHY (LAMBDA (Z)(CAR (CDR Z))))
)))
```

čím sme definovali dve funkcie PRVY a DRUHY s jasným významom : funkcia PRVY dává ako výsledok prvý prvok hodnoty argumentu, funkcia DRUHY, druhý prvok. Teda napr. aplikáciou (PRVY (QUOTE (A B C))) dostaneme A, aplikáciou (DRUHY (QUOTE A B C))) dostaneme B.

5.3 Vetvenie a opakovanie

Vetvenie programu možno v jazyku lisp realizovať pomocou špeciálnej funkcie COND. Funkcia COND má premenný počet algoritmov, z ktorých každý má tvar (podm forma), kde podm je forma, predstavujúca podmienku, pri splnení ktorej sa má vyhodnotiť forma forma. Všeobecný tvar formy COND je

```
(COND (podm1 forma1)
      (podm2 forma2)
      ...
      (podmk formak))
```

a jej vyhodnocovanie prebieha takto : vyhodnotí sa forma podm1. Ak jej hodnotou nie je NIL, znamená to, že podmienka, ktorú táto forma reprezentuje, je splnená a vyhodnotí sa forma forma1, ktorej hodnota je hodnotou celej formy COND. Ak je hodnotou formy podm1 NIL, pokračuje sa postupne vo vyhodnocovaní ďalších podmienok podm2, podm3, ..., podmk, kým nenájdeme podmienku s hodnotou rôznou od NIL. Ak žiadna z podmienok podm1, ... podmk nie je vyhodnotená na hodnotu rôznu od NIL, hodnota funkcie COND nie je definovaná. Ak sa podmienka podm1 vyhodnotí na hodnotu rôznu od NIL, hodnotou celej formy COND je hodnota formy forma1.

Ako príklad použitia funkcie COND si ukážeme na definovaní funkcie - programu na zistenie, či daný argument je prázdny zoznam. Na definovanie takejto funkcie využijeme skutočnosť, že prázdny zoznam možno reprezentovať atómom, a to špeciálnym atómom NIL. Preto jadro programu možno písať takto :

```
(COND ((ATOM X) (EQ X NIL))
      (T        NIL))
```

Ako vidieť, funkcia bude vracať ako výsledok atóm T, ak hodnotou argumentu je prázdny zoznam, NIL inak.

Typickým prostriedkom jazyka lisp na predpísanie opakova-

nia výpočtového procesu je rekurzia. Ako príklad možno uviesť definovanie funkcie, ktorá zistí, či hodnota argumentu je zoznam. Pri definovaní tejto funkcie využijeme nasledujúce skutočnosti :

Hodnota argumentu bude zoznam vtedy, ak je to atóm NIL, teda prázdny zoznam alebo prázdny zoznam možno dostať postupným odstraňovaním prvého prvku zoznamu (zostane vždy časť CDR). Funkciu, nazvime ju JEZOZNÁM, možno písať takto :

```
(JEZOZNAM (LAMBDA (X)
     (COND ((ATOM X) (EQ X NIL))
           (T        (JEZOZNAM (CDR X)))
```

6. Záver

Každý systém jazyka lisp má definovaný celý rad systémových funkcií, ktoré možno nájsť v príslušnom manuáli. Programovanie v lispe je jasné a elegantné, aj keď sa pochopiteľne líši od programovania v "klasických" programovacích jazykoch.

Literatúra

/1/ P.Henderson : Functional Programming: Application and Implementation. Englewod Cliffs, Prentoce Hall 1980

/2/ McCarthy ; P.W.Abrahams, D.J.Edwards, T.P.Hart, M.I.Levin: The Lisp 1.5 Programmer ś Manual. Cambridge, MIT Press 1962

/3/ L. Molanr, P. Návrat: Programovanie v jazyku lisp, Bratislava, Alfa 1988

Ľudovít Molnár
Katedra počítačov EF SVST
Bratislava