Expressões e Sentenças de Atribuição

Eduardo Piveta

Expressões

- Para entender a avaliação de expressões é necessário entender
  - A ordem de avaliação dos operadores
  - A avaliação dos operandos
- A ordem de avaliação dos operadores é ditada pela
  - associatividade e
  - pelas regras de precedência da LP
- A ordem de avaliação dos operandos normalmente é deixada em aberto no projeto de LPs
  - Isso repassa aos implementadores da LP escolherem a ordem...o que pode gerar a resultados diferentes (ef. colaterais)

Atribuições

- Atribuições são a essência das LPs imperativas
- As sentenças de atribuição são usadas para modificar os valores das variáveis
  - Modificando assim os seus valores em tempo de execução
- Sentenças de atribuição especificam uma expressão a ser avaliada e uma localização para armazenar o resultado da expressão
  - Com algumas variações

Expressões Aritméticas

- A avaliação automática de expressões aritméticas foi um dos objetivos das primeiras LPs de alto nível (ex. FORTRAN)
- São compostas de operadores, operandos, parênteses e chamadas de funções
  - Um operador pode ser unário, binário ou ternário
    - (um, dois ou três operandos)
  - Os operadores normalmente são do tipo infix
    - Entre os operandos (a + b)
      - Exceções incluem Perl, que possui alguns operadores prefix

Expressões Aritméticas

- A implementação de uma expressão normalmente causa duas ações:
  - Obtenção dos operandos em memória
  - Execução das expressões aritméticas
- Questões relacionadas às expressões aritméticas:
  - Quais são as regras de precedência de operadores?
  - Quais são as regras de associatividade de operadores?
  - Qual é a ordem da avaliação dos operandos?
  - Existem restrições na avaliação dos operandos em termos de efeitos colaterais?
  - A linguagem possibilita sobrecarga de operadores definida pelo usuário?
  - Que misto de tipos é permitido nas expressões?

Expressões > Ordem de Avaliação de Operadores > Precedência

- O valor de uma expressão depende em parte da ordem de avaliação dos operadores.
- Exemplo:

  a + b * c
- Avaliar esta expressão da esquerda para a direita e da direita para a esquerda pode gerar resultados diferentes
- As LPs devem definir uma ordem de precedência entre os operadores
  - Isto é feito através de regras de precedência de operadores

Expressões > Ordem de Avaliação de Operadores > Precedência

- Nas linguagens imperativas, a ordem de precedência normalmente é a mesma (baseada na matemática)
  - A mais alta precedência é o operador de exponenciação...
  - ...Multiplicação e Divisão
  - ...Subtração e Adição
  - ...

Expressões > Ordem de Avaliação de Operadores > Precedência

- Podem existir também versões unárias de adição e subtração
- Normalmente aparecem entre parênteses:

  A + (-B) * C // Legal

  A + -B * C // Ilegal
- Dependendo da LP, as regras podem ser um pouco diferentes

Expressões > Ordem de Avaliação de Operadores > Precedência

Precedência/ LP	Ruby	LPs baseadas em C	Ada
Mais alta	**	++ e -- posfixados	**, abs
	+ e – unários	++ e – préfixados, + e – unários	*, /, mod, rem
	*, / e %	*, / e %	+ e – unários
Mais baixa	+ e – binários	+ e – binários	+ e – binários

Em APL todos os operadores possuem a mesma precedência...

Expressões > Ordem de Avaliação de Operadores > Associatividade

- Considere a expressão:

  a – b + c – d

- Quando uma exp. possui dois operadores adjacentes com a mesma precedência, a ordem de avaliação é feita através das regras de associatividade da LP
  - Mais à direita
  - Mais à esquerda
- Normalmente os operadores tem associatividade à esquerda, com exceção da exponenciação (à direita)
  - Exceção: Em Ada, a exponenciação não é associativa....

Expressões > Ordem de Avaliação de Operadores > Associatividade

Linguagem	Regra de associatividade
Ruby	Esquerda: *, /, +, - Direita: **
LPs baseadas em C	Esquerda: *, /, %, + e – binários Direita: ++, --, + e – unários
Ada	Esquerda: todos, exceto ** Não associativo: **
APL	Direita: todos os operadores

Expressões > Ordem de Avaliação de Operadores > Parênteses

- Parenteses podem ser usados para forçar uma ordem de avaliação

  (A + B) + (C + D)

- É mais simples (em termos de implementação de LPs) dispensar as regras de precedência, associar todos os operadores da esq. p/ a dir (ou vice-versa) e usar parênteses
  - No entanto, isso pode comprometer a legibilidade
  - E pode fazer com que escrever expressões seja mais tedioso

Expressões > ... > Expressões em Ruby

- Como Ruby é uma linguagem puramente OO, os operadores são métodos...
  - a + b é uma chamada ao método +, da classe do objeto referenciado por a, passando b como parâmetro...
- Os operadores podem ser redefinidos pelos usuários
  - Isso é util para tipos definidos pelo usuário

Expressões > Ordem de Avaliação de Operadores > Expressões Condicionais

- Operadores ternários estão disponíveis em LPs baseadas em C, Perl, JavaScript, Ruby...

```
int var;
if (expr)
    var = expressao1;
else
    var = expressao2;
```

- é igual a

```
var = (expr) ? expressao1 : expressao 2;
```

Expressões > Ordem de Avaliação de Operandos

- A ordem de avaliação dos operandos é irrelevante....
  - ... a menos que existam efeitos colaterais.
- Um efeito colateral de uma função ocorre quando a função muda um de seus parâmetros ou uma variável externa (global).
  - b = a + fun(a);

Expressões > ... > Efeitos Colaterais

- Em LPs funcionais puras e na matemática não existe tal problema (não existem variáveis)
- Em LPs imperativas existem 2 soluções p/ o problema:
  - Desabilitar efeitos colaterais
    - Difícil, tem problemas de desempenho (var. globais)
  - Especificar no projeto da linguagem a ordem de avaliação dos operandos (e forçar implementadores a isso)
    - Dificulta a otimização do compilador (reordenar operandos)
    - Ex. Java (esq. para a direita)

Sobrecarga de Operadores

- É o uso de um mesmo operador para diferentes tipos de operandos
  - adição (+) para inteiros, floats, strings etc.
- Problemas
  - Ex.: x = &y (endereço de y) e x = z&y (operador bit-a-bit)
  - avg = sum / count; // avg é um float, sum e count são ints.
    - Pascal diferencia as divisões: div para inteiros, / para floats. Ex.: avg := sum / count;
    - JavaScript evita esse problema ao não ter aritmética para inteiros;
    - Em PHP, se a divisão de dois inteiros não é inteira, é produzido um valor do tipo float

Sobrecarga de Operadores

- Ada, C++, Fortran95, C# permitem que símbolos sejam sobrecarregados
  - Ex.: MatrixAdd(MatrixMult(A,B), MatrixMult(C, D)) poderia ser escrito como A * B + C * D
- No entanto, tais sobrecargas devem ser usadas com cuidado...

Sentenças de Atribuição

- São uma das principais construções das LPs imperativas
- Normalmente são construídas através da seguinte forma:

  <variavel_alvo> <operador_atribuicao> <expressao>
- A maioria das LPs usa = para atribuição e == para testar igualdade
  - Algumas LPs derivadas do ALGOL 60 usam := e = respectivamente (pascal, ada etc)

Operadores de Atribuição Compostos

- O ALGOL 68 introduziu operadores de atribuição compostos
- São normalmente usados para expressões do tipo

  a = a operador b
- E são substituidas por algo como

  a operador= b
- Ex.:

  sum += valor;

  sum = sum + valor;

Operadores de Atribuição Unários

- LPs baseadas em C, Perl e JavaScript fornecem dois operadores aritméticos unários que são na verdade sentenças de atribuição
- Exemplo:
  - sum = ++ cont;
- É o mesmo que:
  - count = count + 1;
  - sum = count;
- E sum = cont ++ ???

Atribuições como uma Expressão

- LPs baseadas em C, Perl e JavaScript possibilitam que atribuições produzam um resultado, usado em expressões:
  - while ((ch = getchar()) != EOF) {...}
- Ou então:

```
int a, b, c, d;
a = b = c = d = 10;
printf("%d, %d, %d, %d\n",a, b, c , d);
a = b + (c = d / b++) - 1;
printf("%d, %d, %d, %d\n",a, b, c , d);
```

- Problemas podem ocorrer também ao fazer:
  - if (x = y) ... // Em C e C++

Atribuições em Listas

- LPs recentes, como Perl e Ruby permitem sentenças de atribuição com múltiplos alvos e múltiplas fontes, tais como:

```
// Em Perl
($primeiro, $segundo, $terceiro) = (20, 40, 60)
Ou
($primeiro, $segundo) = ($segundo, $primeiro)
// Troca sem temp
```

Considerações Finais

- Expressões são compostas de constantes, variáveis, parênteses, chamadas a funções e operadores.
- A semântica de uma expressão é dada normalmente pela ordem de avaliação dos operadores
  - Para isso são utilizadas regras de associatividade e precedência.
  - Regras de precedência de operandos são necessárias para tratar de efeitos colaterais
- Sentenças de atribuição incluem variáveis alvo, operadores de atribuição e expressões.
  - Possuem formas variadas, incluindo alvos condicionais, operadores de atribuição e atribuições em listas.

Expressões e Sentenças de Atribuição

Eduardo Piveta