8- Sistema de Tipos

- O projeto de um verificador de tipos de uma LP é baseado nas informações sobre as construções sintáticas da linguagem, a noção de tipos e as regras de atribuição de tipos às construções da linguagem
- Um sistema de tipo é uma coleção de regras para atribuição de tipos a várias partes de um programa
- Um verificador de tipos implementa um sistema de tipo

8- Sistema de Tipos...

- Monomorfismo
  - Um sistema de tipo é dito ser monomórfico, se toda constante, variável, resultado de função e parâmetro formal deve ser declarado com um tipo específico
  - Vantagem: verificação de tipo é simples e direta
  - Desvantagem: dificulta a criação de programas reutilizáveis, já que muitos algoritmos e estruturas de dados são genéricos
  - Nenhuma LP é totalmente monomórfica. Por exemplo, algumas funções e procedimentos pré-definidos da linguagem Pascal têm tipos que não podem ser expressos no sistema de tipos da linguagem:
    - Procedimento *write* – o efeito de uma chamada de procedimento *write(E)* irá depender do tipo de *E*. Na verdade, o identificador *write* denota simultaneamente vários procedimentos distintos $\Rightarrow$ sobrecarga
    - Função eof – o tipo da função é *File(*τ*)*$\rightarrow$ *Lógico*, sendo τ qualquer tipo $\Rightarrow$ polimorfismo
    - A herança também pode ser encontrada de uma forma bem simples: um tipo intervalo herda todas as operações de seu pai

$\Downarrow$

Pascal é inconsistente: todas as abstrações explicitamente definidas pelo programador são monomórficas, mas muitas abstrações pré-definidas são sobrecarregadas ou polimórficas

8- Sistema de Tipos...

- Sobrecarga
  - Capacidade de amarrar a um único identificador ou operador várias entidades (amarrações) simultaneamente
  - Também conhecido por *polimorfismo ad-hoc*
  - Em geral, sobrecarga só é aceitável onde cada chamada de função não é ambígua
  - Considere um identificador ou operador I que denote uma função f_1 do tipo $S_1 \rightarrow T_1$ e uma função f_2 do tipo $S_2 \rightarrow T_2$. Há dois tipos de sobrecarga:
    - *Sobrecarga independente de contexto*: requer que S_1 e S_2 sejam distintos. Para identificar a função a ser chamada, basta saber o tipo do parâmetro real

8- Sistema de Tipos...

- *Sobrecarga dependente de contexto*: requer que S_1 e S_2 sejam distintos ou que T_1 e T_2 sejam distintos. Se S_1 e S_2 são distintos, a função a ser chamada pode ser identificada pelo tipo do parâmetro. Caso contrário, se T_1 e T_2 são distintos, deve-se considerar o contexto para poder identificar a função a ser chamada. Com esse tipo de sobrecarga, torna-se possível formular expressões cujas chamadas de funções sejam ambíguas. Caberá à LP proibir expressões ambíguas.
  - Exemplo: suponhamos que o operador */* denote três funções distintas:
    - » Inteiro x Inteiro $\rightarrow$ Inteiro
    - » Real x Real $\rightarrow$ Real
    - » Inteiro x Inteiro $\rightarrow$ Real

    Considerando n uma variável do tipo Inteiro e x, do tipo Real, qual função será chamada nas expressões abaixo?

    x := 7 / 2 $\Rightarrow$ x = 3.5

    n := 7 / 2 $\Rightarrow$ n = 3

    n := (7 / 2) / (5 / 2) $\Rightarrow$ n = 3 / 2 = 1

    x := (7 / 2) / (5 / 2) $\Rightarrow$ x = 3 / 2 = 1.5 **OU** $\Rightarrow$ x = 3.5 / 2.5 = 1.4

8- Sistema de Tipos...

- Polimorfismo
  - Relaciona-se a abstrações que operam de maneira uniforme sobre valores de diferentes tipos
  - Também conhecido por *polimorfismo paramétrico*
  - ML foi a primeira LP a fornecer um sistema de tipo realmente polimórfico

8- Sistema de Tipos...

Abstrações polimórficas

- Ao invés de definir tipos específicos para o tipo de uma função, utiliza-se variáveis de tipo
- Exemplo: ML

  fun segundo (x: σ, y: τ) = y

  A função **segundo** é do tipo σ x $\tau \rightarrow \tau$, sendo que σ e τ representam qualquer tipo. Assim, a chamada de função **segundo (13, true)** retorna **true**. A chamada de função **segundo ("José", "Silva")** retorna "**Silva**". Mas a chamada de função **segundo (10, 9, 1990)** é inválida

8- Sistema de Tipos...

Tipos parametrizados

- Um tipo parametrizado é um tipo que tem outro(s) tipo(s) como parâmetro(s)
- Exemplo: os tipos **file**, **set** e **array** do Pascal
- Em uma LP monomórfica, somente existem tipos parametrizados pré-definidos. Por exemplo, em Pascal, o programador **não** pode escrever:

  type par (τ) = record primeiro, segundo: τ end;
  ParInt = par (integer);
  ParReal = par (real);

- Exemplo: ML

  type τ par = τ * τ

  Podemos utilizar essa definição de tipo para criar um par de inteiros **int par** ou um par de reais **real par**, por exemplo

8- Sistema de Tipos...

Politipos

- Um politipo é um tipo que contém uma ou mais variáveis de tipo. Por exemplo, σ x $\tau \rightarrow \tau$, σ e τ são variáveis de tipo, ou seja, representam um tipo desconhecido
- Um politipo deriva uma família completa de tipos, obtida através da substituição sistemática de um tipo real por cada variável de tipo
- Um tipo que não possui variáveis de tipo é chamado de monotipo. Em linguagens monomórficas, todos os tipos são monotipos
- Um politipo também é um tipo. Em geral, o conjunto de valores de um politipo é a interseção de todos os tipos que podem ser derivados a partir dele
- Exemplo 1: considerando o politipo Lista (τ), a lista vazia é a interseção de todos os tipos Lista
- Exemplo 2: O tipo $\tau \rightarrow \tau$ é a interseção de todos os tipos Inteiro $\rightarrow$ Inteiro, String $\rightarrow$ String, Boolean $\rightarrow$ Boolean, etc

8- Sistema de Tipos...

- Inferência de tipo
  - Inferência de tipo $\Rightarrow$ o tipo de uma entidade declarada é inferido, ao invés de ser explicitamente determinado

Inferência de tipo monomórfico

- Em ML, podemos escrever:
  fun par(n) = (n mod 2 = 0)

  Supondo que o operador **mod** tenha tipo **integer x integer** $\rightarrow$ **integer**, torna-se possível inferir que **n** tem que ser do tipo **integer** e que o tipo do retorno da função é "lógico" $\Rightarrow$ o tipo da função **par** é **integer** $\rightarrow$ **boolean**

- ML adota uma atitude *laissez-faire* com relação a tipos: o programador pode determinar ou não o tipo de uma entidade
- Desvantagem: um pequeno erro de programação pode confundir o compilador, que irá produzir mensagens de erro obscuras, ou inferir um tipo diferente daquele desejado pelo programador

$\Downarrow$

é boa prática de programação determinar tipos explicitamente, mesmo que isso seja redundante

8- Sistema de Tipos...

Inferência de tipo polimórfico

- Nem sempre é possível que uma inferência de tipo produza um monotipo. Em outras palavras, às vezes será produzido um politipo
- Exemplo (ML):

```
fun length (l) =
  case l of
     nil   ⇒  0
   | cons (h, t)  ⇒  1 + length (t)
```

O resultado dessa função é **integer** e **l** é do tipo **Lista (τ)**. Assim, **length** é do tipo **Lista (τ) → integer**

8- Sistema de Tipos...

- Coerção
  - Coerção é um mapeamento implícito de valores de um tipo para valores de um tipo diferente. Ela é executada automaticamente, sempre que necessário
  - Considere um contexto no qual se espera um operando do tipo *T*, mas é fornecido um operando do tipo *T'* (não equivalente a *T*). A LP pode permitir uma coerção nesse contexto, desde que a linguagem defina um mapeamento único entre o tipo *T'* e o tipo *T*
  - LPs modernas tendem a minimizar ou eliminar a coerção, pois ela não funciona bem com sobrecarga e polimorfismo. Por exemplo, no caso de Ada, o mapeamento é feito explicitamente *Float(x)*

8- Sistema de Tipos...

- Subtipos e herança
  - Relacionada à capacidade de permitir que subtipos herdem operações de seus supertipos
  - Também conhecido por *polimorfismo de inclusão*
  - Pascal reconhece uma forma restrita de subtipo, ao permitir a definição de intervalos de qualquer tipo primitivo discreto T
  - Cada LP reconhece alguns subconjuntos de tipos como subtipos, mas não subconjuntos arbitrários. Por exemplo, nenhuma LP permite declarar uma variável que varie somente sobre os números inteiros primos

8- Sistema de Tipos...

- Uma condição necessária para S ser um subtipo de T é que $S \subseteq T$. Assim, um valor de S pode ser usado com segurança onde um valor do tipo T é esperado
- Seja U um tipo que não seja um subtipo de T, mas que tenha elementos comuns com T. Então, um valor do tipo de U pode ser usado onde se espera um valor do tipo de T, desde que o verificador de tipo, em tempo de execução, reconheça que o valor pertence também a T
- Associado a cada tipo T, há um subconjunto de operações que podem ser aplicadas a valores do tipo T. Essas operações também poderão ser aplicadas a valores de qualquer tipo S que seja subtipo de T. Podemos dizer então que S herda todas as operações associadas a T
- O termo herança vem da Programação Orientada a Objetos