Linguagens de Domínio Específico

Fabio Mascarenhas – 2017.1

http://www.dcc.ufrj.br/~fabiom/dsl

Subtipagem

- O conjunto de valores de um tipo pode ser um subconjunto do conjunto de valores de outro tipo

- Podemos querer expressar isso no sistema de tipos através de uma *relação de subtipagem* ≤

- Em uma linguagem com subtipagem nominal essa relação é declarada pelo programador; em Java ela é dada pelas cláusulas *extends* e *implements*

- A relação de subtipagem é *simétrica* (t ≤ t) e *transitiva* (r ≤ s e s ≤ t implica r ≤ t)

Tipagem nominal vs estrutural

- Se um tipo é identificado unicamente pelo seu nome, independente de quaisquer outros atributos, esse tipo é *nominal*
- Um exemplo são todos os tipos de Java exceto vetores e tipos genéricos
- Se um tipo é identificado unicamente pela sua estrutura, independente de ter um nome associado, ele é *estrutural*
- Tipos nominais são mais simples de entender e implementar, tipos estruturais são mais flexíveis

Subtipagem nominal

- Determinar se um tipo nominal é subtipo de outro é fácil com um algoritmo recursivo (contanto que se tenha cuidado de eliminar ciclos)
- Todo tipo é subtipo dele mesmo
- Consultamos todos os supertipos já registrados do tipo em um *mapa de subtipagem direta*, caso um deles seja o outro tipo então já terminamos
- Caso não seja, pegamos cada supertipo direto do tipo e verificamos recursivamente se ele é subtipo do outro tipo
- As hierarquias de tipos não costumam ser profundas, então a recursão não é problemática

Subtipagem estrutural

- A subtipagem entre dois tipos estruturais depende do que eles representam
- Exemplo simples: tipos de funções em diversas linguagens que têm funções como valores
- Exemplo intermediário: tipos de registros com subtipagem por *largura* e *profundidade*
- Exemplo complexo: tipos genéricos de Java com *coringas*
- Na subtipagem estrutural, *igualdade* de tipos é subtipagem mútua: t1 = t2 se somente se t1 <= t2 e t2 <= t1

Tipos com subtipagem – blocos e structs

- Uma extensão da linguagem de blocos com tipos simples e structs nominais

```
prog   -> struct* bloco
bloco  -> stat*
struct -> "struct" NAME ("<:" NAME)? campo+ "end"
campo  -> NAME ":" NAME
stat   -> "while" exp "do" bloco "end" |
          NAME (":" NAME)? "=" exp | NAME ("." NAME)+ "=" exp
exp    -> aexp ">" aexp | aexp
aexp   -> termo (aop termo)*
termo  -> fator (mop fator)*
fator  -> NUM | "new" NAME | NAME ("." NAME)* | "(" exp ")"
aop    -> "+" | "-"
mop    -> "*" | "/"
```

- Tipos pré-definidos são int, real, bool, com coerções entre int e real

- O escopo dos tipos é global, e um tipo pode ser referenciado mesmo antes de sua definição