Universidade Federal de Alagoas
Instituto de Computação
Curso de Ciência da Computação

Curry
Paradigmas de Linguagens de Programação

João Victor Ribeiro Ferro
Lucas Albuquerque Lisboa

Maceió
2021

Introdução

Curry é uma linguagem declarativa de propósito geral que mescla programação funcional com programação lógica, com forte influência em Haskell e Prolog. Seu nome é em homenagem ao matemático Haskell Curry, em especial pela sua contribuição em lógica combinatória. Foi implementada por Michael Hanus e Sergio Antoy.

Programa

Um programa em Curry especifica a semântica das expressões e ao executar significa simplificar uma expressão até que um único valor (junto com ligações de variáveis livres) seja calculado, ou seja, consistem em um conjunto de declarações de tipo e função. As declarações de tipo definem os domínios computacionais (construtores) e as declarações de função as operações nesses domínios. Predicados no sentido de programação lógica podem ser considerados como funções com tipo de resultado Bool.

Curry é fortemente tipada. Caso a variável seja declarada sem especificar o tipo, o sistema de inferência de tipos determinará o tipo da variável a partir do sistema de tipos de Hindley/Milner, o qual implementa o polimorfismo paramétrico, em que realiza o lambda calculus para determinar o melhor tipo de dado na expressão e atribuir às variáveis com tipos omissos (STAGNI, 2011). Como o nome sugere, esse sistema decorre do trabalho de Roger Hindley e, posteriormente, de Robin Milner e é essencial para grande parte das linguagens funcionais estaticamente tipadas.

A linguagem utiliza a técnica de avaliação preguiçosa, isto é, a avaliação de uma expressão é retardada até o momento em que o valor é necessário. Isso evita que o programa faça avaliações desnecessárias.

Tipos de Dados

Boolean

Os valores booleanos são definidos pela declaração do tipo de dados

Ex.:

```
data Bool = True| False
```

A conjunção(sequencial) é associativa a esquerda pelo operador &&:

EX.:

```
True && y = y
False && x = False
```

Da mesma forma, a disjunção (sequencial) "||" e a negação **not** são definidos como de usual. Sendo mais utilizados em condicionais (if_then_else).

Ex.:

```
if_then_else :: Bool → a → a → a
if_then_else b t f = case b of True → t
                               False → f
```

Uma função com tipo de resultado Bool é frequentemente chamado de predicado, tem como padrão pré-definidos "==" e "=:=". Além disso, há também predicados integrados para comprar objetos, como "<". Os dados definidos pelo usuário são comparados na ordem de sua definição nas declarações de tipo de dados e recursivamente nos argumentos.

EX.:

```
data Coin = Head | Tail

compare Head Head = EQ
compare Head Tail = LT
compare Tail Head = GT
compare Tail Tail = EQ
```

Pode também comparar expressões contendo variáveis livres. Por exemplo a avaliação **"X < [Trail]"** retornando True para as ligações **{x = []}** e **{x = (Head:_)}** . Para números ou caracteres, pode haver suspensão ao comprar valores desconhecidos.

Função

As funções como do tipo t1 → t2, é o tipo que produz um valor do tipo t2 para cada argumento do tipo t1, ou seja, uma função f é aplicada a um argumento por escrito "f x", é associativa à direita.

EX.:

t1 → t2 … → tn+1

Uma função f para n argumentos é uma expressão, com associatividade à esquerda.

Ex.:

fe1 e2 … en

Além disso, o prelúdio define um operador associativo à direita "$", muito útil para evitar a utilização de colchetes, o $ tem baixa precedência de associação à direita.

EX.:

"f $ g $ 3+4", é equivalente "f (g(3+4))"

Inteiro

Os valores inteiros comuns, como “14” ou “-14”, são considerados construtores (constantes) do tipo int. Os operadores usuais, como + ou *, são funções pré-definidas que são avaliadas apenas se ambos os argumentos forem valores inteiros.

Float

Assim como para inteiros, valores como “3.1423” ou “5.0e-4” são considerados construtores do tipo float.

List

O tipo [t] denota todas a lista cujos elementos são valores do tipo t. O tipo de listas pode ser considerado como pré-definido pela declaração:

data [a] = [ ] | a : [a]

Logo, [ ] representa uma lista vazia e x : xs é uma lista não vazia, na qual consiste o primeiro elemento x e restante xs. Também é comum representar a lista por colchetes, como mostra no exemplo a seguir:

[e_1,e_2,...,e_n]

Observa-se que a lista e_1: e_2: … e_n: [ ] (é equivalente e_1:(e_2:(...:(e_n: [ ]) …)) sendo ‘:’ é associativo à direita).

Characters

Valores como ‘a’ ou ‘0’ são constantes do tipo char. Caracteres especiais podem ser denotados com uma barra invertida, por exemplo, ‘\n’ ou ‘\228’ para o caractere correspondente na tabela ASCII de posição 228.

Tuples

Se t1,t2,..., tn são tipos e n>= 2, então (t1,t2,...,tn) denotam os tipos de n-tuplas. O tipo unitário ( ) tem um único elemento ( ) e pode ser definido por,

data ( ) = ( )

que também pode ser interpretado como uma 0-tupla.

Expressões

As expressões são parte essencial do funcionamento do Curry. Há dois princípios que regem o funcionamento do Curry (HANUS, 2000):

1. Se existe uma solução, ela (ou uma mais genérica) será computada;
2. Se a expressão pode ser reduzida a um valor, Curry computa esse valor;

Para garantir o primeiro ponto, Curry utiliza o mecanismo de avaliação preguiçosa. Isso possibilita que, caso não haja um valor possível para atribuir a uma expressão, será retornada a expressão em uma forma genérica. Outro mecanismo utilizado é o *backtracking* para busca de uma solução, similar ao Prolog, o que pode gerar o problema da incompletude da expressão. No entanto, caso o programador prefira outras estratégias para avaliação das expressões, ele pode especificar em seu programa restrições a serem seguidas pelo compilador na execução do programa.

Referências Bibliográficas

HANUS, Michael. **Curry**: An Integrated Functional Logic Language. [*S. l.*: *s. n.*], 2016. Disponível em: https://www-ps.informatik.uni-kiel.de/currywiki/_media/documentation/report.pdf. Acesso em: 31 mar. 2021.

HANUS, Michael; KUCHEN, Herbert; AACHEN, Rwth; LI, Informatik. **Curry: A Truly Functional Logic Language**. [*S. l.*: *s. n.*], 2000. Disponível em: https://www.researchgate.net/publication/2463383_Curry_A_Truly_Functional_Logic_Language. Acesso em: 31 mar. 2021.

STAGNI, Henrique. **XBA: Uma linguagem orientada a objetos comtipagem estrutural**. 2011. 27 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade de São Paulo (USP), São Paulo, 2011. Disponível em: https://bcc.ime.usp.br/tccs/2011/henrique/monografia_pre.pdf. Acesso em: 31 mar. 2021.