1. Quem foi o idealizador do cálculo lambda e quando ele foi proposto (aproximadamente)?

O cálculo lambda foi idealizado por Alonzo Church, um matemático e lógico norte-americano. Ele introduziu o cálculo lambda na década de 1930 como parte de seus trabalhos em lógica matemática e teoria da computação. A formulação inicial do cálculo lambda ocorreu por volta de 1932, com desenvolvimentos posteriores nos anos seguintes. O cálculo lambda é uma importante ferramenta na teoria da computabilidade e serviu como base para muitos desenvolvimentos subsequentes na ciência da computação.

2. Qual a relação entre cálculo lambda e máquina de Turing?

A relação entre o cálculo lambda e a máquina de Turing está no fato de que ambos os modelos têm o mesmo poder expressivo. A "tese de Church-Turing" postula que qualquer função computável pode ser expressa tanto através do cálculo lambda quanto por uma máquina de Turing. Isso implica que, em termos de capacidade de calcular algoritmicamente, o cálculo lambda e a máquina de Turing são equivalentes. Essa equivalência é fundamental para estabelecer o que é computável e para entender as limitações da computação.

3. O que é o cálculo lambda? Em que ele foi útil na computação?

O cálculo lambda é um formalismo matemático desenvolvido por Alonzo Church na década de 1930 como uma notação para expressar e manipular funções matemáticas. Ele descreve a computação em termos de substituição e abstração de funções, permitindo a representação de cálculos complexos em termos de operações mais simples.

Foi fundamental para Teoria da Computação, linguagens de programação, semântica formal, estudos sobre recursão, prova de teoremas, programação funcional, etc.

4. O que são variáveis livres (independentes)? E variáveis vinculadas (dependentes)? Cite exemplos de variáveis livres e vinculadas em uma expressão lambda.

Variáveis Livres (Variáveis Independentes): São variáveis que aparecem em uma expressão lambda, mas não estão vinculadas por uma abstração. Em outras palavras, são variáveis que não estão dentro do escopo de uma função abstrata que a define. Elas podem ser substituídas por valores concretos ou outras expressões.

Na expressão lambda λx.(x+y), a variável y é uma variável livre, porque ela não é definida dentro do escopo da função abstrata. Ela é uma variável externa à função e pode assumir qualquer valor.

Variáveis Vinculadas (Variáveis Dependentes): São variáveis que aparecem em uma expressão lambda dentro do escopo de uma abstração, ou seja, são os argumentos da função definida pela abstração.

Na expressão lambda λx.(x+y) a variável x é uma variável vinculada, pois é o argumento da função definida pela abstração λx. Qualquer ocorrência de xx dentro do corpo da função se refere a esse argumento específico.

5. O que significa currying em cálculo lambda? Exemplifique.

Currying é um conceito importante no cálculo lambda e na programação funcional que se refere à técnica de transformar uma função que aceita vários argumentos em uma sequência de funções que aceitam um único argumento cada. Essa técnica é nomeada em homenagem ao matemático e lógico Haskell Curry, que contribuiu significativamente para a teoria da computação.

6. O que significa uma expressão ser um "combinador", em cálculo lambda? Cite um exemplo de expressão que é um combinador.

Em cálculo lambda, um "combinador" é uma expressão lambda que não possui variáveis livres, ou seja, todas as variáveis presentes na expressão são vinculadas por abstrações. Isso significa que um combinador é uma função autônoma que não depende de nenhuma variável externa.

Um exemplo clássico de um combinador é o "combinador de ponto fixo", frequentemente denotado como Y. Este combinador é usado para definir recursão em termos do cálculo lambda, permitindo que as funções se refiram a si mesmas.

7. O que é aplicação e o que é abstração numa expressão lambda? Exemplifique.

1. **Aplicação:** Aplicação no cálculo lambda envolve a aplicação de uma função a um argumento. A ideia é substituir todas as ocorrências da variável vinculada no corpo da função pelo argumento fornecido. Essa operação é similar a chamar uma função em programação convencional.
   Exemplo: Suponha que temos a função f(x)=x2 definida no cálculo lambda como λx.x∗x Se quisermos aplicar essa função ao valor 3, faríamos a seguinte operação: (λx.x∗x) 3 que resulta em 3∗3 ou 9.
2. **Abstração:** Abstração no cálculo lambda envolve a criação de uma função anônima. É como definir uma função com uma variável vinculada como argumento, permitindo a generalização do cálculo para um valor desconhecido. A abstração é denotada pela notação λx, onde x é a variável vinculada e a expressão após o ponto descreve o corpo da função.
   Exemplo: Vamos definir uma função que dobra um número. A expressão lambda para isso seria λx.2∗x, onde x é a variável vinculada e 2∗x é o corpo da função. Essa expressão representa uma função que aceita um argumento e retorna o dobro desse argumento.
   Então, (λx.2∗x) 4 resultaria em 2∗4 ou 8.

8. O que significa dizer que duas expressões lambda são α-equivalentes? Exemplifique.

Duas expressões lambda são consideradas α-equivalentes se elas diferem apenas no nome de suas variáveis vinculadas, mas são idênticas em termos de estrutura e comportamento. Em outras palavras, α-equivalência é uma relação de equivalência que ignora a nomenclatura das variáveis e foca na estrutura da expressão e nas relações de ligação entre as variáveis.

Quando duas expressões lambda são α-equivalentes, elas representam essencialmente a mesma função, apenas com nomes de variáveis diferentes. Isso ocorre porque a escolha dos nomes das variáveis não afeta o comportamento da função.

Exemplo:
Considere as seguintes expressões lambda:

1. \(λx.x + 1\)
2. \(λy.y + 1\)

Essas duas expressões são α-equivalentes, pois a única diferença entre elas é o nome da variável vinculada ((x) e (y)), mas a estrutura e o comportamento da função são os mesmos - ambas adicionam 1 ao argumento passado.

9. O que é a operação de α-conversão (α-renomeação)? Exemplifique.

A operação de α-conversão, também conhecida como α-renomeação, é uma operação no cálculo lambda que envolve a renomeação de variáveis vinculadas em uma expressão lambda para evitar conflitos de nomes. Ela é usada para garantir que as variáveis vinculadas em uma função tenham nomes únicos, sem alterar a estrutura ou o comportamento essencial da função.

A α-conversão é importante porque expressões lambda α-equivalentes (ou seja, expressões que têm a mesma estrutura, mas diferentes nomes de variáveis) podem ser consideradas equivalentes em termos de comportamento, mas podem parecer diferentes em termos de representação textual.

Vamos considerar um exemplo para ilustrar a α-conversão:

Suponha que temos a seguinte expressão lambda:
(λx.x + y)

Agora, suponha que queremos evitar conflitos de nomes e renomear a variável vinculada (x) para (z). Isso seria feito através de uma α-conversão:

(λz.z + y)

Nesse exemplo, a α-conversão foi aplicada para renomear (x) para (z), evitando qualquer conflito de nomes com outras variáveis. A expressão original (λx.x + y) e a expressão resultante (λz.z + y) são α-equivalentes, pois têm a mesma estrutura e o mesmo comportamento.

A α-conversão é especialmente útil quando você está manipulando expressões lambda em provas formais, otimização de código ou análises matemáticas, onde a escolha de nomes de variáveis pode ser arbitrária e a renomeação é usada para evitar ambiguidades ou erros.

10. O que é Redução-Beta (β-Redução)?

A redução-beta, muitas vezes chamada de β-redução, é uma operação fundamental no cálculo lambda. Ela descreve a aplicação de uma função a um argumento, substituindo todas as ocorrências da variável vinculada no corpo da função pelo argumento fornecido. A redução-beta é o mecanismo principal pelo qual as expressões lambda são avaliadas e simplificadas.

11. O que significa dizer que uma expressão lambda está em sua "forma normal"? Exemplifique com uma expressão lambda em sua forma normal.

Uma expressão lambda está em sua "forma normal" quando não é mais possível realizar reduções-beta nela. Isso significa que a expressão está completamente simplificada e não pode ser avaliada ainda mais usando as regras de redução do cálculo lambda. Considere a expressão (λx.x+2) 5. A β-redução desta expressão resulta em 5+2, que é 7. Portanto, essa expressão não está em sua forma normal, pois ainda pode ser reduzida.

Agora, considerando a expressão (λy.3), essa expressão não possui variáveis vinculadas que possam ser substituídas por argumentos. Não há abstrações que precisam ser aplicadas a argumentos. Portanto, essa expressão está em sua forma normal, pois não há mais reduções-beta possíveis.

12. Pesquise sobre o Combinador Y. O que é e o que ele faz? Descreva um pouco seu funcionamento.

O Combinador Y, também conhecido como Combinador de Ponto Fixo Y, é uma construção importante no cálculo lambda e na teoria da recursão. Ele permite a definição de funções recursivas dentro do cálculo lambda, onde a recursão é expressa sem a necessidade de se referir explicitamente à própria função.

O Combinador Y é uma expressão lambda que, quando aplicada a uma função \(f\), produz uma função que é o ponto fixo de \(f\), ou seja, uma função que tem o mesmo comportamento da função \(f\) quando aplicada a si mesma.

A expressão do Combinador Y é a seguinte:

\[Y = λf.(λx.f (x x)) (λx.f (x x)) \]

Vamos entender o funcionamento do Combinador Y com um exemplo. Suponha que desejamos definir uma função recursiva que calcule o fatorial de um número \(n\). A função fatorial \(fact\) poderia ser expressa em cálculo lambda como:

\[fact = λn.\, \text{if}\ (n = 0)\ \text{then}\ 1\ \text{else}\ n * (fact (n - 1)) \]

Agora, usando o Combinador Y, podemos definir essa função recursiva de maneira mais elegante:

\[fact = Y \, (\lambda f.\, λn.\, \text{if}\ (n = 0)\ \text{then}\ 1\ \text{else}\ n * (f (n - 1))) \]

Neste exemplo, o Combinador Y permite que a função \(fact\) seja definida sem se referir explicitamente a si mesma. A recursão é tratada pela aplicação da função \(f\) ao próprio Combinador Y.

O funcionamento do Combinador Y envolve a criação de um "loop" funcional no qual a função \(f\) é chamada com a expressão \((x x)\), onde \(x\) é a mesma expressão \((λx.f (x x))\). Isso cria uma recursão, onde a função \(f\) é aplicada a ela mesma, permitindo que ela seja chamada recursivamente sem a necessidade de referências diretas à própria função.

O Combinador Y é uma construção poderosa que demonstra a capacidade do cálculo lambda de expressar recursão e é uma ferramenta crucial para entender a relação entre o cálculo lambda e a recursão na teoria da computação.