- Avaliação por demanda: o que é

A avaliação por demanda é uma estratégia utilizada para que seja possível expandir as capacidades de um computador quando lidamos com procedimentos que exigem muita memória ou processamento. O princípio se resume à frase: "Não calcule o valor de uma expressão até que ele seja necessário". Assim, o programador ganha mais flexibilidade e expressabilidade porque usando essa estratégia somos capazes de representar qualquer estrutura infinita através de listas. Foi visto em aula, por exemplo: A construção de uma lista que contém todos os números primos; uma lista que tem todos os números; etc.

- Quando algo é calculado? Quando as computações são suspensas?

Relembrando a frase que representa o princípio da avaliação por demanda: "Não calcule o valor de uma expressão até que ele seja necessário". Tendo isso em mente, é intuitivo que só serão feitos os cálculos quando eles forem necessários. Em scheme, por exemplo, uma parte desse princípio é violado porque calcula o valor dos argumentos de uma função antes que esta seja chamada pelo usuário, que pode ser observado no uso do 'cons'. Isso não deveria ocorrer porque as computações deveriam estar suspensas para postergar a complexidade do procedimento e só serem realizadas quando acessadas, de tal modo que uma mesma suspensão já calculada apenas retorna seu valor.

- Qual é a estrutura de implementação utilizada? Como ela funciona?

A linguagem utilizada em aula foi o Scheme, porém ela viola alguns dos princípios da avaliação por demanda porque calcula o valor dos argumentos de uma função antes que esta seja chamada pelo usuário, que pode ser observado no uso do 'cons': Os argumentos não deveriam ser calculados até que fosse feito uma chamada de 'car' ou 'cdr'.
Além disso, para a implementação da avaliação por demanda são usadas 'suspensões'. Uma estrutura semelhante a um fechamento, por conter um ambiente e uma expressão. Esses fechamentos só são calculados quando são necessários.

- Se não tiver a estrutura acima, como posso usar apenas fechamentos para implementar avaliação por demanda? Dê um exemplo

Dado que podemos usar fechamentos, então podemos criar as 'suspensões'. Então basta que sejam construídos fechamentos com expressões e ambiente, mas estes sejam calculados quando necessários. Um exemplo dessa situação pode ser o seguinte:

```
'((expressão ambiente) (expressão ambiente) ... (expressão ambiente))
```

Usando apenas uma lista de listas já somos capazes de fazer implementações simples da avaliação por demanda. Nesse caso, cada

elemento de 1° nível da lista representa um fechamento e quando for feita uma chamada de car ou cdr é necessário que este seja calculado. Além disso, se usarmos o seguinte código, podemos representar uma lista infinita como '((expressão ambiente))

```
(set ints-de(lambda (x) (cons x( lambda () (inst-de (+ x 1))))))
(set inteiro (ints-de 0))
(0 . #<procedure>)
```

- O que são Streams? Como funcionam?

As Streams são promessas, isto é, listas infinitas que são utilizadas pela avaliação por demanda para representar um conjunto de dados muito grande. Elas funcionam através do uso inerente das suspensões para não exigir muito processamento e memória. Portanto, em cada célula há a informação necessária para que ela seja calculada quando necessária.

- O que são esquemas de recursão? Explique e mostre um exemplo para uma série onde f(i) = g(f(i-1), f(i2), i)

São técnicas de codificação usadas em linguagens semelhantes à SASL porque é criada uma sequência de valores infinita. A ideia consiste no uso de uma função recursiva que se baseia em valores anteriores, como o Fibonacci visto em aula.
Se temos uma lista de valores *list_val* e uma lista de índices *list_indices* podemos fazer:

```
(set mapcar (lambda g list_val list_indices)
  (cons (g (car list_val) (cadr list_val) (caddr list_indices))
    (mapcar g (cdr list_val) (cdr list_indices)))
```

- Porque avaliação por demanda é perigosa se temos efeitos colaterais?

Por causa da 'temporalidade', isto é, como os valores só são calculados quando utilizados, os resultados podem ser afetados pelo fluxo de uso, tornando muito difícil a depuração do código.

- Qual a utilidade de avaliação por demanda? Como ela pode fazer nossa programação mais simples? Dê um exemplo.

A sua principal utilidade é a expressividade que ela dá ao programador por permitir listas infinitas, tornando a programação mais simples por não obrigar o código a ser o mais otimizado possível ou limitar a quantidade de dados processados. Por exemplo, para calcular uma lista com todos os fatoriais, a pilha de execução quebraria em uma linguagem que não usa a avaliação por demanda.

- Como foi feita a implementação por demanda no ep? Em que lugares foram introduzidas as suspensões? Como funcionam as promessas e as suspensões?

A implementação da avaliação por demanda no EP usou a seguinte estratégia: Após a criação da estrutura SuspV e Promise, foi definido que dentro de cada promessa há uma suspensão contendo uma expressão e ambiente ou um valor, e para obter o resultado foi usada uma função chamada query-promise. Então, bastou que algumas das funções do interpretador fossem alteradas, como a appC, carC, consC, ifC, interp, etc.