Preguntas Teóricas Final PLP | Jano

- ¿Qué es el paradigma funcional?
La programación funcional consiste en definir funciones y aplicarlas para procesar información.

Las “funciones” son verdaderamente funciones (parciales):
- Aplicar una función no tiene efectos secundarios.
- A una misma entrada corresponde siempre la misma salida (son determinísticas).
- Las estructuras de datos son inmutables.

Las funciones son datos como cualquier otro:
- Se pueden pasar como parámetros.
- <u>Su significado es su valor (si está definido). Y su valor depende del valor de sus subexpresiones.</u>
- Se pueden devolver como resultados.
- Pueden formar parte de estructuras de datos.

(Ej. árbol binario en cuyos nodos hay funciones).

- ¿Qué es el paradigma orientado a objetos?

Un programa basado en este paradigma es un conjunto de objetos que interactúan entre sí en un ambiente mandándose mensajes para lograr un objetivo determinado.

- ¿Qué sistemas de polimorfismo vimos y cuáles son las diferencias entre ellos?

- Sistema F
- Polimorfismo Let

- ¿Qué es la semántica denotacional?

La semántica denotacional es un enfoque formal para definir el significado de los programas en lenguajes de programación. Se basa en asociar a cada construcción del lenguaje un valor matemático (su denotación), lo cual permite analizar y razonar sobre los programas de manera rigurosa y abstracta.

<u>Conceptos Básicos</u>

- Denotación: En la semántica denotacional, cada construcción sintáctica de un lenguaje de programación se mapea a un objeto matemático, como funciones, conjuntos o dominios. La denotación de una expresión es el valor matemático que representa su significado.
- Dominios: Son estructuras matemáticas que se utilizan para representar los valores denotacionales. Pueden ser conjuntos, espacios de funciones, órdenes parciales, entre otros.
- Funcionales Semánticos: Son funciones que describen cómo las construcciones del lenguaje se mapean a sus denotaciones. Estos funcionales son definidos inductivamente sobre la estructura de las construcciones sintácticas.

Semántica denotacional (en programas deterministas). Para cada programa P, la relación entre las entradas y las salidas de P es una función que escribimos $[\![P]\!]$. La relación se define entonces como

$$P, E \hookrightarrow S \iff [\![P]\!]E = S$$

La pregunta es, obviamente, cómo definir $[\![P]\!]$.

- ¿Qué es la inferencia de tipos?

- ¿Cuál es la motivación de agregarle tipos al cálculo lambda?

- Si agregamos tipos, y efectivamente demostramos conservación de tipos, podemos conocer algunas propiedades de nuestro resultado antes de computarlo.

- Por otro lado, dependiendo la extensión y si lo podemos efectivamente probar, se puede llegar a demostrar que si un término está bien tipado entonces es fuertemente normalizable. Es decir, termina su evaluación.
- También permite analizar estáticamente tu código si está bien formado en una etapa pre compilación.

- ¿Cómo funciona Prolog?

Para verificar una fórmula, Prolog intenta refutar su negación usando el método de Resolución SLD. Unifica las variables recorriendo los predicados en forma DFS (Depth-First Search) de izquierda a derecha, y realizando unificación en cada uno.

- ¿Por qué el tipado de cálculo lambda se parece al de deducción natural?

Son isomórficos: las reglas de tipado simple del cálculo lambda se corresponden con alguna regla de juicio del sistema de deducción natural que vimos. Por ejemplo, introducir la flecha sigma -> sigma en los tipos es igual que la implicación en deducción natural.

$$\frac{A \qquad B}{A \,\&\, B}\;(\&\text{-I}) \qquad\qquad \frac{A::M \qquad B::N}{A\times B::(M,N)}\;(\times\text{-I})$$

$$\frac{A \,\&\, B}{A}\;(\&\text{-}E_1) \qquad\qquad \frac{A\times B::(M,N)}{A::M}\;(\pi_1)$$

$$\frac{A \,\&\, B}{B}\;(\&\text{-}E_2) \qquad\qquad \frac{A\times B::(M,N)}{B::N}\;(\pi_2)$$

$$\frac{A \to B \qquad A}{B}\;(\to\text{-}E) \qquad\qquad \frac{\lambda A.B::M\to N \qquad A::M}{B::N}\;(\lambda\text{-}E)$$

$$\begin{array}{c} [A]^x \\ \vdots \\ B \\ \hline A\to B \end{array}\;(\to\text{-}I^x) \qquad\qquad \begin{array}{c} [A::M]^x \\ \vdots \\ B::N \\ \hline \lambda A.B :: M \to N \end{array}\;(\lambda\text{-}I^x)$$

El isomorfismo va más allá de las reglas formales. La evaluación de programas en el cálculo lambda se corresponde directamente con la simplificación y normalización de pruebas que usa el cálculo natural.

- ¿Qué es el método de resolución general y el SLD?

- ¿Por qué en un lenguaje débilmente tipado te puede tirar type error?

Eso es posible mediante los algoritmos de inferencia de tipos. La inferencia de tipos es la capacidad de deducir automáticamente, ya sea de manera parcial o completa, el tipo de una expresión en tiempo de compilación. El compilador a menudo es capaz de inferir el tipo de una variable o una función, sin que se hayan dado anotaciones de tipo explícitas.

- ¿Qué es el teorema de normalización fuerte?

Teorema de Tait o normalización fuerte: Todo término tipado en este sistema, termina

El teorema de normalización fuerte establece que en ciertos sistemas formales, como el cálculo lambda tipado y algunos sistemas de tipos de lógica, para todo término existe una secuencia finita de reducciones que termina en una forma normal. Una forma normal es un término que no puede ser reducido o simplificado más.

Este teorema es particularmente relevante en los siguientes contextos:

- Cálculo Lambda Tipado: Es un formalismo fundamental en la teoría de tipos y el diseño de lenguajes de programación funcionales.
- Lógica Intuicionista: La lógica intuicionista es una variante de la lógica clásica que rechaza el principio del tercero excluido y tiene aplicaciones en la teoría de la demostración.

Importancia del Teorema de Normalización Fuerte:

- Consistencia: Garantiza que no se pueden derivar contradicciones en el sistema, porque todas las secuencias de transformaciones (reducciones) de términos finalizan en un término normal, evitando bucles infinitos y términos indefinidos.
- Terminación: Asegura que cualquier programa o prueba en el sistema eventualmente termina (se evalúa por completo), lo cual es crucial para la decidibilidad de los problemas y la confianza en los resultados producidos por el sistema.
- Constructividad: En la lógica intuicionista y la teoría de tipos, asegura que las construcciones son efectivas, es decir, cualquier término que se puede escribir en el sistema representa una construcción finita y completa.

- ¿Cómo es el algoritmo de inferencia?

El algoritmo de Martelli-Montanari es guiado por sintaxis. Por cada nodo del árbol que se desprende de un término, se aplica cada una de las reglas que se establecen. El algoritmo siempre termina y sea G un conjunto de pares, si G tiene un unificador, el algoritmo termina exitosamente y retorna un MGU y si G no tiene unificador, el algoritmo termina con falla.

- Explicar la relación entre deducción natural y valuación

Ambos enfoques buscan establecer la validez de fórmulas lógicas. La deducción natural lo hace a través de reglas de inferencia y manipulaciones sintácticas, mientras que la valuación lo hace a través de la asignación de valores de verdad y evaluaciones semánticas.

Correlación de Correctitud y Completitud:

- Correctitud: Si una fórmula puede ser derivada utilizando deducción natural a partir de un conjunto de premisas, entonces la fórmula es válida en todas las valuaciones que hacen verdaderas a las premisas. Esto significa que las deducciones realizadas a través de deducción natural son semánticamente correctas.
- Completitud: Si una fórmula es válida en todas las valuaciones posibles (es una tautología), entonces puede ser derivada utilizando deducción natural. Esto implica que el sistema de deducción natural es completo con respecto a la lógica proposicional.

Demostraciones y Verificación:

- En la deducción natural, se demuestra la validez de una fórmula construyendo una prueba formal paso a paso siguiendo reglas de inferencia.
- En el enfoque de valuación, se verifica la validez de una fórmula comprobando que sea verdadera bajo todas las posibles asignaciones de valores de verdad (valuaciones) de las proposiciones básicas.