[Fede] - Cos'è un future, cos'è una monade, e come è possibile vedere un future come una monade

La semantica fondamentale che preserva un future è la memoization, la make\_future corrisponde alla return (data una computazione ne ritorna un future), la bind corrisponde alla yield grossomodo:

bind(x : F<a>, f: a -> F<B>) : F<B>

make\_future (fun () ->

y : a= yield(x)

f(y) )

(fare la yield su un make future deve essere come la funzione identità)

[Fede] - Dispatching statico e dispatching dinamico, differenza ed esempi di costrutti visti nel corso per entrambi

Rust solitamente ha dispatch statico, ma con un trait object si può fare dispatching dinamico [...]

[Stach] - Descrivere brevemente il concetto di ownership in Rust

[... da manuale ...]

[Stach] - Come si può inquadrare la ownership tramite una monade

La return significa prendere un dato non posseduto e metterci la ownership, ad esempio una copia del dato

La bind bisogna prendere un dato posseduto (monade) e una funzione, quello che basta fare è f x:

bind(x : m a, f : a -> m b) : m b {

f x

}

In Rust: bind(x a, f a -> b) b { f(x) } fa la stessa cosa, ma la funzione dovrebbe prenderla per reference, lasciandola la legittimo proprietario, quindi: bind(x a, f &(a -> b)) b { f(x) }

[Stach] - Ricordi qualche caso in cui abbiamo osservato che l'uso di costrutti imperativi portava a soluzioni più costose

La memoria e l'ipotesi generazionale nella garbage collection

[Vainigli] - Cos'è una type class e quali costrutti generalizzano

...

[Vainigli] - Comparazione tra Traits e Type-classes

Le type classes sono predicati sui tipi mentre i trait di go sono predicati unari sui tipi, altre comparazioni [...]

[Vainigli] - Cos'hanno i traits di Rust in più rispetto ai traits di Go

[hint: associated types] quando si vuole predicare in un tipo l'esistenza di certi metodi si ha il problema che tutto sia un metodo di una classe sola, quando invece spesso si hanno metodi che agiscono su più classi. Es. il grafo, si ha tipo dei nodi, degli archi e si vuole avere la libertà di non dover definire nodi e archi per poter usare grafi, quindi il predicato deve essere ternario non unario (come in go), altri esempi sono le tabelle hash, c'è il tipo chiave, valore e tabella, anche qui predicato ternario (non disponibile in go). Rust definisce un trait con predicato su tipo, dentro definisco campi ecc [...]

[Vainigli] - Cosa si intende per riga nel polimorfismo di riga

[Vainigli] - Se faccio una get di un tipo, che tipo dà lei al tipo?

[Guerra] - Parli delle chiusure, degli oggetti, dei traits e il loro confronto a runtime

[Guerra] - all'interno di un metodo scrivi una chiusura, ha accesso alle variabili dello scope esterno compresa this, la domanda è cosa succede quando si usa this all'interno di una chiusura?

Ad esempio i linguaggi a oggetti che hanno aggiunto le chiusure successivamente, una chiusura altro non è che un oggetto di una classe che ha un solo metodo, il codice scritto nella chiusura è il codice di un oggetto con un unico metodo, this in quel contesto si riferisce all'oggetto chiusura, non quello iniziale (questo succede ad esempio in JavaScript).

[Bartolini] - gli atomi, strategie di implementazione e relative problematiche

[Bartolini] - quali sono le feature di Haskell che rendono semplice definire le monadi in Haskell senza ricorrere a modifiche

due features: type-class e polimorfismo sui costruttori di tipo

(i traits di go e rust sarebbero sufficienti ma non si riesce, perchè rust non ha polimorfismo sui costruttori di tipo e go non ha polimorfismo generico)

[Bartolini] - definire una monade per il lancio di una moneta

d a = list (float \* a) [una lista di coppie prob e elemento di tipo a]

la return prende un elemento di tipo a e deve restituire una lista di probabilità con probabilità 100%

la bind ((a,b):t) >>= f = [ (a\*x, o) | (x,o) <- f b ] ++ t >>= f

[Adamo] - implementazione di Kotlin per gli ADT

[Adamo] - I GADT

[Adamo] - lifetime in Rust e che problema vogliono risolvere

[Adamo] - Quali sono le strategie che il compilatore utilizza per compilare codice che usa polimorfismo parametrico (templates, generics, ecc), e i loro vantaggi e svantaggi

Monomorfizzazione (+ pro e contro)

Rappresentazione uniforme dei dati (se a runtime tutti i dati hanno la stessa rappresentazione)

Polimorfismo alla C (una funzione è polimorfa quando prende in input un puntatore arbitrario in memoria e tratta il dato come se fosse una collezione di bytes)

[Minervino] - Nullable types in Kotlin