Paradigmi di Programmazione – CORSI A e B
C.d.L. in Informatica – A.A. 2025-2026

Traccia degli argomenti per l'orale

Prof.ssa Chiara Bodei, Prof. Paolo Milazzo

Questo documento contiene un elenco dettagliato degli argomenti che potranno essere oggetto di domande durante la prova orale dell'esame del corso.

Attenzione: questo **non** è un elenco di domande. Le domande potranno entrare più nel dettaglio dei singoli argomenti, potranno richiedere di fare collegamenti tra i vari argomenti, di formulare esempi dei vari concetti, oppure di spiegare un concetto sulla base di un esempio dato dal docente.

Lambda-calcolo

1. Cenni storici:
   - origini del λ-calcolo
   - rapporto con Macchine di Turing
   - nascita dei paradigmi imperativo e funzionale
2. Aspetti sintattici:
   - sintassi delle λ-espressioni
   - variabili libere e legate
   - α-equivalenza
3. Aspetti semantici:
   - valutazione di λ-espressioni
   - capture avoiding substitution
   - β-riduzione
   - β-equivalenza
   - confluenza: il Teorema di Church-Rosser
   - non terminazione: la λ-espressione Ω
4. λ-calcolo e programmazione funzionale:
   - funzioni Curryed e applicazione parziale di funzioni
   - la ricorsione tramite il combinatore Y
   - strategie di riduzione deterministiche ispirate dai linguaggi di programmazione: *call-by-name* e *call-by-value*

Type Systems

1. Motivazioni per il controllo dei tipi
2. Concetti di:
    - sistema di tipi
    - type safety
    - controllo dei tipi statico (e suoi vantaggi/svantaggi)
    - controllo dei tipi dinamico (e suoi vantaggi/svantaggi)
3. Sistema di tipi per le espressioni:
    - funzionamento generale
    - concetti di approssimazione e composizionalità
    - proprietà di progresso
    - proprietà di conservazione
4. Lambda-calcolo tipato semplice:
    - funzionamento generale
    - ambiente dei tipi e suo ruolo nei giudizi di tipo
    - proprietà di progresso
    - proprietà di conservazione

Sviluppo di interpreti (e compilatori)

1. Compilazione, interpretazione e approccio misto
2. Il concetto di macchina astratta, le sue componenti e tipiche implementazioni (in hardware e in software)
3. Il Run-Time Support
4. Struttura di un compilatore e le varie fasi della compilazione
5. Introduzione allo sviluppo di interpreti:
    - la grammatica del linguaggio come base per lo sviluppo di Scanner e Parser
    - la semantica SOS come base dell'implementazione dell'interprete

Paradigma Object-Oriented

1. Definizione di oggetto e sue caratteristiche
2. Concetti di OOP:
    - incapsulamento
    - astrazione
    - interfaccia

- ereditarietà
- sostituzione
- polimorfismo

3. Object-based vs class-based, structural-(sub)typing vs nominal-(sub)typing
4. OOP in OCaml (a grandi linee)
5. OOP in Java:
   - basi del linguaggio Java (a grandi linee),
   - le interfacce
   - tipo apparente e tipo effettivo di un oggetto
   - modello della memoria della Java Virtual Machine (JVM): suo funzionamento e collocazione dei membri statici e d'istanza
   - ereditarietà in Java,
   - overloading e overriding di metodi
   - dynamic dispatch con le tabelle dei metodi e lo sharing strutturale
   - generici in Java: funzionamento generale, motivazioni per la non covarianza e type erasure
   - Java Collection Framework: struttura generale, concetto di iteratore e suo funzionamento
6. Ereditarietà multipla e mixin:
   - il problema dell'ambiguità nell'accesso ai membri ereditati
   - il problema del diamante (diamond problem)
   - le soluzioni di C++ basate su disambiguazione ed ereditarietà "virtual"
   - il funzionamento delle interfacce multiple in Java tramite iTable
   - l'ereditarietà multipla in linguaggi interpretati: la soluzione di Python
   - Il concetto di composizione tramite mixin
7. Il principio di sostituzione di Liskov:
   - definire le specifiche di una classe
   - precondizioni e postcondizioni dei metodi
   - il principio di sostituzione di Liskov come relazione tra i comportamenti di supertipo e sottotipo
   - indecidibilità del principio di sostituzione di Liskov
   - regole indotte dal principio di sostituzione di Liskov: della segnatura, dei metodi e delle proprietà

Garbage collection

1. Gestione della memoria dinamica (heap):
   - dangling pointers e garbage
   - il modello a grafo dell'heap: root set e concetto di raggiungibilità
   - heap con blocchi a dimensione fissa o variabile
   - la frammentazione interna ed esterna
   - obiettivi di un garbage collector
2. Principali approcci di garbage collection:
   - reference counting
   - mark and sweep
   - copy collection
   - approcci generazionali
3. Gestione della memoria dinamica in Rust:
   - caratteristiche generali del linguaggio Rust
   - i concetti di ownership e lifetime
   - il trasferimento dell'ownership
   - il concetto di borrowing

Programmazione concorrente

1. Concetti introduttivi:
   - esecuzione non sequenziale: concorrenza e parallelismo
   - interleaving e motivazioni per la sua analisi a livello del linguaggio macchina
   - thread vs processi
   - shared-memory vs message-passing
2. La concorrenza spiegata tramite un linguaggio modello:
   - definizione del linguaggio concorrente minimale: sintassi, sistema di tipi e semantica
   - problematiche di concorrenza in assenza di meccanismi di sincronizzazione
   - estensione del linguaggio con primitive di mutua esclusione lock/unlock: scopo e funzionamento
   - locking coarse-grained vs fine-grained
   - deadlock e modi per risolverlo (ad es. 2PL)
3. La concorrenza nei linguaggi di alto livello:
   - thread Java: funzionamento generale (a grandi linee)
   - primitive di sincronizzazione per i thread java: metodi e blocchi sincronizzati
   - cenni su primitive di concorrenza e sincronizzazione in altri linguaggi