Frontend

• y2016p3q4 (a)

Grammar

- y2024q4p1 (a,b)
- y2004p4q1 (a-e)
 - Operator associativity / precedence (textbook)
- y2018p4q3 (b)
 - ambiguity
- y2002p4q2 (a)
 - CFG

Lexical (regex, FSA) & Syntax analysis/Parsing (CFG, PDA)

- y2018p4q3 (a)
- y2002p4q2 (b,c)
 - LEX / YACCtools
- y2023p4q1 (a)
- y2007p4q4 (a)
 - lexing

Recursive Descent

• y2004p4q1 (f)

LL(k)

- y2022p4q1
- y2024q4p1 (c,d)
- y2023p4q1 (b)
- y2020p4q4 (d)
 - left recursion not LL(1)

LR(k), SLR(k)

- y2024q4p1 (c,d)
- y2015p3q3
 - LR(0) items, shift / reduce
- y2020p4q4
 - DFA
- y2021p4q4

- y2023p4q1 (c)
 - SLR(1) ACTION and GOTO

Type Checking

- y2019p4q3 (a)
 - for user-defined data type

Simplification

remove type information, remove locations

- y2021p4q3 (b)
 - remove syntactic sugar
- y2020p4q3 (b)
 - let binding
- y2019p4q3 (d)
 - remove nested patterns

Backend

Translation

CPS, defunctionalise, etc

- y2017p23q4
- y2022p4q2
- y2023p4q2
 - memory, stack ↓ and heap↑
- y2016p3q3, y2014p3q5
 - tail recursion

JARGON

closure

- y2018p4q3 (c)
 - in functional programming

VM

stack-oriented intermediate code

- y2020p4q3
 - closure

```
(fun x -> e') e''
(i) evaluates e'' and pushes the result on the stack
(ii) creates a closure (c,k) for (fun x -> e') in the heap
and pushes a pointer to it on the stack.
> c = the address of the first instruction in e'
> k = the number of free variables of e' (excluding x)
[pop off k values from the stack and placed in the closure.]
(iii) apply the top of the stack to the argument below.
return
```

• y2019p4q3 (b,e)

LOOKUP STACK_LOCATION -2 # fetch the argument v from the stack CALL sum # sum(v) PUSH 3 ADD # sum(v) + 3 RETURN

- y2016p3q4 (b)
 - heap representation for pairs, machine instructions
- y2005p5q6
 - a parse tree of an expression
 - \rightarrow_i stack-oriented intermediate code
 - \rightarrow_{ii} machine code (register-oriented arch, e.g., RISCV)
 - remove push-pop pairs
 - efficiency

• y2000p3q3

- allocation and recovery of heap records, union type
- arrays with non-manifest bounds, labels and GOTO commands

Mixed topics

Garbage Collection

- Reference Counting
- Tracing Collection
 - Mark and Sweep
 - Copy Collection
 - Generational Collection

Past Papers

- y2024q4p2(c)
- y2006p5q6 (b)
 - Tracing Collection def
- y2018p4q3 (d)

- Reference Counting
- y2017p23q3 (a-c)
 - Garbage, reference count overflow

Opt

- y2023p4q1 (d)
- y2021p4q3 (a)
 - o (map f l1) @ (map f l2) = map f (l1 @ l2)
- y2017p23q3 (d)
 - \circ (map f) o (map g) = map (f o g)
- y2018p4q3 (g)
 - 0 × e
- y2016p3q4 (c)
 - pair
- y2013p3q4
 - tail recursive to iteration

Exception

stack-oriented code

- y2019p4q4
 - front-end, type safety, Optimise
- y2011p3q4 (c)
 - raise and handle

Linking

- y2024q4p2(b)
- y2001p6q6

.data	<-	.symtbl ->	.strtab
D0: 00 00 00 01		SØ: DØ NØ	N0: a
			N1: \0
		^	
		I	
.text	<-	.rela.text	
T8: LOAD R1, 0(R2) # 0	ג	R0: T8 S0	

Run-time Memory

Stack

- y2014p3q4
- y2018p4q4

- y2018p4q3 (f)
 - static link

Bootstrapping

• y2024q4p2(a)