Vrije Universiteit Brussel

The essence of meta-tracing JIT compilers

Maarten Vandercammen
Advisors: Theo D'Hondt, Coen De Roover,
Joeri De Koster, Stefan Marr and Jens Nicolay

What is tracing JIT compilation?

```
(define (fac x)
   (if (< x 2)
     1
     (* x
        (fac (- x 1)))))
```

$\longrightarrow$

```
(label 'fac-loop)
(literal-value 2)
(save-val)
(lookup-var 'x)
(save-val)
(lookup-var '<)
(apply-native 2)
(guard-false)
(literal-value 1)
(save-val)
(lookup-var 'x)
(save-val)
(lookup-var '-)
(apply-native 2)
...
(goto 'fac-loop)
```

What is tracing JIT compilation?

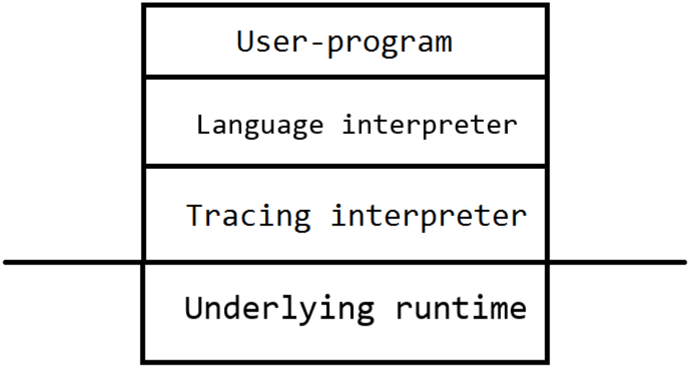
```
(define (fac x)
   (if (< x 2)
     1
     (* x
        (fac (- x 1)))))
```

$\longrightarrow$

```
(label 'fac-loop)
(literal-value 2)
(save-val)
(lookup-var 'x)
(save-val)
(lookup-var '<)
(apply-native 2)
(guard-false)
(literal-value 1)
(save-val)
(lookup-var 'x)
(save-val)
(lookup-var '-)
(apply-native 2)
...
(goto 'fac-loop)
```

Traces are always linear!
Use guards to protect against changes in control-flow

Meta-tracing



Meta-tracing

- Advantage: language implementer can get tracing for free

Meta-tracing

- Advantage: language implementer can get tracing for free

```
(define (close parameters expressions)
      (define lexical-environment environment)
      (define (closure . arguments)
        (define dynamic-environment environment)
        (can-start-loop expressions)
        (set! environment lexical-environment)
        (bind-parameters parameters arguments)
        (let* ((value (evaluate-sequence expressions)))
          (set! environment dynamic-environment)
          value))
      closure)
```

Meta-tracing

- Advantage: language implementer can get tracing for free

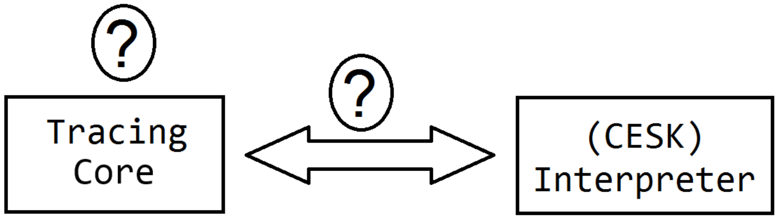
```
(define (close parameters expressions)
      (define lexical-environment environment)
      (define (closure . arguments)
        (define dynamic-environment environment)
        (can-start-loop expressions)
        (set! environment lexical-environment)
        (bind-parameters parameters arguments)
        (let* ((value (evaluate-sequence expressions)))
          (set! environment dynamic-environment)
          value))
      closure)
```

- PyPy and RPython project [1]

[1] Bolz, C. F. (2012)

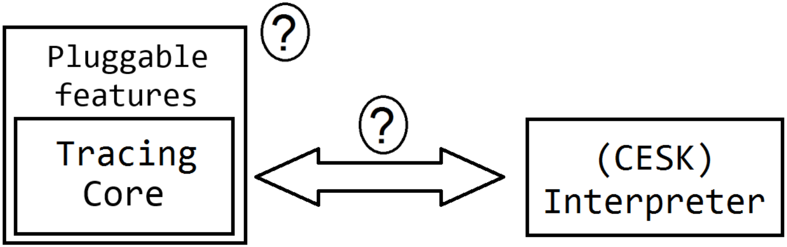
Goal

Construct minimalistic meta-tracing JIT

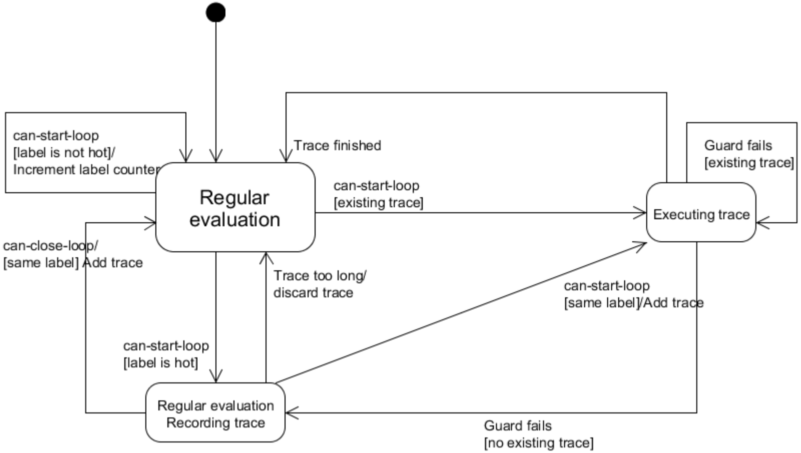


Goal

Grow meta-tracing JIT



Tracing JIT as state-machine


Regular evaluation
Executing trace
Regular evaluation Recording trace
can-start-loop [label is not hot]/ Increment label counter
Trace finished
Guard fails [existing trace]
can-start-loop [existing trace]
can-close-loop/ [same label] Add trace
Trace too long/ discard trace
can-start-loop [same label]/Add trace
can-start-loop [label is hot]
Guard fails [no existing trace]

Specify formal semantics

Tracing framework

$$\Big\downarrow$$

$$\frac{CESK \to \; <CESK', \tau_2>}{<x_1, x_2, \ldots, \tau_1, CESK> \to <x_1, x_2, \ldots, \tau_1\tau_2, CESK'>}$$

Roadmap

- Literature study ✓
- Tracing core ✓
- Additional features
  - Trace jumping ✓
  - Trace merging ✓
  - True vs. false loops ✓

Roadmap

- Transform state-machine into working operational semantics
- Specify formal semantics
- Write thesis