Towards static compilation of dynamic code generation

Morten Rhiger

Roskilde University, Denmark

IFIP 2.11 WG Meeting 12

June 3–6, 2013, Minneapolis, USA

Multi-stage programs ...

construct, combine, and execute

code fragments

dynamically at run time.

Multi-stage programs ...

construct, combine, and execute

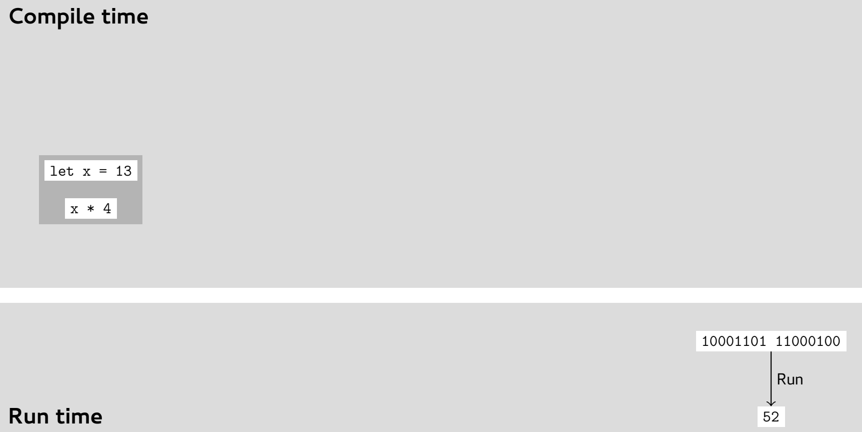
code fragments

dynamically at run time.

Representing code fragments?

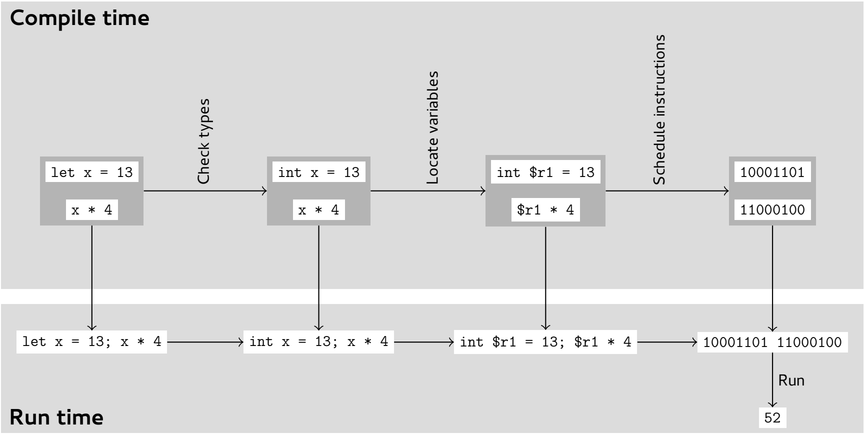
Dynamic code generation

A spectrum of compilers



Dynamic code generation

A spectrum of compilers



This talk

Compiling away variable names in code fragments

1. A high-level, multi-stage source language $\lambda^{[]}$
   - Type system (New)
2. A low-level, multi-stage target language
   - Nameless de Bruijn terms (Known)
   - Semantics (Known)
3. Extending type system of 1 with translation into 2
   - The code type of 1 precisely characterizes the memory layout of code fragments.

Source language

Source language ($\lambda^{[]}$)

Syntax

$$x \in \textbf{Names}$$

$$s ::= x_1, \cdots, x_k$$

$$t ::= t_1 \to t_2 \mid [s]t$$

$$e ::= x \mid \lambda x{:}t\,.\,e \mid e_1\ e_2 \mid \uparrow e \mid \downarrow e \mid \mathsf{run}\ e$$

Source language ($\lambda^{[]}$)

Lexically scoped code type

$[\ x_1, \cdots, x_n\ ]\ t$ — Type of value produced by code fragment.

Variables allowed free in code fragment.

Each variable must be in scope!

Source language ($\lambda^{[]}$)

Environments

$$\Gamma ::= \emptyset \mid x^{n}{:}t, \Gamma$$

$$\Gamma|^{n} = \{x \mid x^{n}{:}t \in \Gamma\}$$

Source language ($\lambda^{[]}$)

Typing (functional part)

$$\frac{\Gamma[x^{n}] = t}{\Gamma \vdash^{n} x : t} \quad \text{(T-Var)}$$

$$\frac{\Gamma \vdash^{n} t \qquad x^{n}{:}t, \Gamma \vdash^{n} e : u}{\Gamma \vdash^{n} \lambda x{:}t.\, e : t \to u} \quad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash^{n} e : t' \to u \qquad \Gamma \vdash^{n} e' : t'}{\Gamma \vdash^{n} e\, e' : u} \quad \text{(T-App)}$$

Source language ($\lambda^{[]}$)

Typing (multi-stage part)

$$\frac{\Gamma \vdash^{n+1} e : t}{\Gamma \vdash^{n} \uparrow e : [\Gamma|^{n+1}]t} \quad \text{(T-Up)}$$

$$\frac{\Gamma \vdash^{n} e : [\Gamma|^{n+1}]t}{\Gamma \vdash^{n+1} \downarrow e : t} \quad \text{(T-Down)}$$

$$\frac{\Gamma \vdash^{n} e : [\,]t}{\Gamma \vdash^{n} \text{run}\ e : t} \quad \text{(T-Run)}$$

Source language ($\lambda^{[]}$)

Typing (subtyping part)

$$\frac{\Gamma \vdash^{n} e : t' \quad t' \leq t}{\Gamma \vdash^{n} e : t} \quad \text{(T-Sub-}t\text{)}$$

$$\frac{\Gamma' \vdash^{n} e : t \quad \Gamma' \subseteq \Gamma}{\Gamma \vdash^{n} e : t} \quad \text{(T-Sub-}\Gamma\text{)}$$

Source language ($\lambda^{[]}$)

Subtyping

$$\frac{t_2 \leq t_1 \quad u_1 \rightarrow u_2}{t_1 \rightarrow u_1 \leq t_2 \rightarrow u_2} \quad \text{(S-Arr)}$$

$$\frac{s_1 \subseteq s_2 \quad t_1 \leq t_2}{[s_1]t_1 \leq [s_2]t_2} \quad \text{(S-Box)}$$

Source language ($\lambda^{[]}$)

Subtyping (intuition)

Subtyping *extend* the expressiveness of the type system.

Subtyping **enables** the following:

1. $\texttt{let x : []int = } \uparrow\texttt{1 in } \uparrow(\lambda\texttt{x:bool. } \downarrow\texttt{c})$

2. $\uparrow(\lambda\texttt{x:bool. } \downarrow(\cdots \texttt{ run}(\uparrow\texttt{2}) \cdots))$

Source language ($\lambda^{[]}$)

Kinding

$$\frac{\Gamma \vdash^{n} t \quad \Gamma \vdash^{n} u}{\Gamma \vdash^{n} t \to u} \quad \text{(K-Arr)}$$

$$\frac{s \subseteq \Gamma|^{n+1} \quad \Gamma \vdash^{n+1} t}{\Gamma \vdash^{n} [s]t} \quad \text{(K-Box)}$$

Source language ($\lambda^{[]}$)

Kinding (intuition)

Kinding *guarantees* that types are lexically scoped and correctly staged.

Kinding (and typing) **prevents** the following (attempts at scope extrusion):

1. ```
   let c : []int ref = ref(↑1)
   in  ↑(λx:int. ···↓(c := ↑x)···)
   ```
2. ```
   let c : [x]int ref = ref(↑1)
   in  ↑(λx:int. ···↓(c := ↑x)···)
   ```
3. ```
   ↑(λx:int.
       let c : [x]int ref = ref(↑1)
       in  ↑(λx:int. ···↓(c := ↑x)···))
   ```
   ```

Observation

The code type dictates variable locations

$$[\ x_1, \cdots, x_n\ ]\ t$$

Code fragments of this type expect

$$x_1, \cdots, x_n$$

to be at the top n positions of the execution stack.

Target language

A prototypical nameless language

Target language

Syntax

$i \in$ **Indices**

$E ::= \#i \mid \lambda.\, E \mid E_1\ @\ E_2 \mid \uparrow E \mid \downarrow E \mid \mathsf{run}\ E$

Target language

De Bruijn indices (intuition)

Named term	Stack	De Bruijn term
$\lambda x.$	$x, \cdots$	$\lambda.$
$\quad \lambda y.$	$y, x, \cdots$	$\quad \lambda.$
$\quad\quad x + y$		$\quad\quad \#1 + \#0$

Target language

Evaluation (functional static part)

$$
\begin{aligned}
[\![\#i]\!]^0\ st &= v_i, \\
&\quad \text{where } st = (v_0, \cdots, v_k) \\
[\![\lambda.\ E]\!]^0\ st &= \langle E, st \rangle \\
[\![E_1 \mathbin{@} E_2]\!]^0\ st &= [\![E]\!]^0\ (v, st'), \\
&\quad \text{where } \langle E, st' \rangle = [\![E_1]\!]^0\ st \\
&\quad\quad \text{and } v = [\![E_2]\!]^0\ st
\end{aligned}
$$

Target language

Evaluation (multi-stage static part)

$$[\![\uparrow E]\!]^0 \; st = [\![E]\!]^1 \; st$$

$$[\![\downarrow E]\!]^1 \; st = [\![E]\!]^0 \; st$$

$$[\![\mathsf{run}\; E]\!]^0 \; st = [\![E']\!]^0 \; (\,), \quad \text{where } E' = [\![E]\!]^0 \; st$$

Target language

Evaluation (dynamic part)

$$
\begin{aligned}
[\![\#i]\!]^{n+1}\ st &= \#i \\
[\![\uparrow E]\!]^{n+1}\ st &= \uparrow([\![E]\!]^{n+1}\ st) \\
[\![\downarrow E]\!]^{n+2}\ st &= \downarrow([\![E]\!]^{n+1}\ st) \\
[\![\lambda.\ E]\!]^{n+1}\ st &= \lambda.\ [\![E]\!]^{n+1}\ st \\
[\![E_1 \mathbin{@} E_2]\!]^{n+1}\ st &= ([\![E_1]\!]^{n+1}\ st) \mathbin{@} ([\![E_2]\!]^{n+1}\ st) \\
[\![\mathsf{run}\ E]\!]^{n+1}\ st &= \mathsf{run}\,([\![E]\!]^{n+1}\ st)
\end{aligned}
$$

Translation

Translation

Machinery

$$\Gamma \vdash^{n} e : t \;/\; E$$ (Typing with translation)

Translation

Machinery

$$\Gamma \vdash^{n} e : t \;/\; E$$ (Typing with translation)

with support from

$$x \in s \;/\; i$$ (Membership with evidence)

$$s' \subseteq s \;/\; F$$ (Scope subset with evidence)

$$\Gamma[x^{n}] = t \;/\; i$$ (Variable lookup with evidence)

$$\Gamma' \subseteq \Gamma \;/\; F$$ (Environment subset with evidence)

$$t \leq t' \;/\; F$$ (Subtyping with evidence, coercion)

Translation

Typing (functional part)

$$\frac{\Gamma[x^n] = t \;/\; i}{\Gamma \vdash^n x : t \;/\; \#i} \quad \text{(Tx-Var)}$$

$$\frac{\Gamma \vdash^n t \qquad x^n{:}t, \Gamma \vdash^n e : u \;/\; E}{\Gamma \vdash^n \lambda x{:}t.\, e : t \to u \;/\; \lambda.\, E} \quad \text{(Tx-Abs)}$$

$$\frac{\Gamma \vdash^n e : t' \to u \;/\; E \qquad \Gamma \vdash^n e' : t' \;/\; E'}{\Gamma \vdash^n e\, e' : u \;/\; E \,@\, E'} \quad \text{(Tx-App)}$$

Translation

Typing (multi-stage part)

$$\frac{\Gamma \vdash^{n+1} e : t \ / \ E}{\Gamma \vdash^{n} \uparrow e : [\Gamma|^{n+1}]t \ / \ \uparrow E} \quad \text{(Tx-Up)}$$

$$\frac{\Gamma \vdash^{n} e : [\Gamma|^{n+1}]t \ / \ E}{\Gamma \vdash^{n+1} \downarrow e : t \ / \ \downarrow E} \quad \text{(Tx-Down)}$$

$$\frac{\Gamma \vdash^{n} e : [\,]t \ / \ E}{\Gamma \vdash^{n} \text{run}\, e : t \ / \ \text{run}\, E} \quad \text{(Tx-Run)}$$

Translation

Typing (subtyping part)

$$\frac{\Gamma \vdash^n e : t' \;/\; E \quad t' \leq t \;/\; F}{\Gamma \vdash^n e : t \;/\; F(E)} \qquad \text{(Tx-Sub-}t\text{)}$$

$$\frac{\Gamma' \vdash^n e : t \;/\; E \quad \Gamma' \subseteq \Gamma \;/\; F}{\Gamma \vdash^n e : t \;/\; F(E)} \qquad \text{(Tx-Sub-}\Gamma\text{)}$$

Related work

- DCG (Engler&Proebsting, ASPLOS'94)
- Fabius (Leone&Lee, PLDI'96)
- VCODE (Engler, PLDI'96)
- 'C (Engler et al, POPL'96)
- DyC (Grant et al, 1997)
- Tempo (Consel&Noël, POPL'96; Noël et al, ICCL'98)
- tcc (Poletto et al, PLDI'97; Poletto et at, TOPLAS'99)
- MetaOCaml (Taha, Kiselyov, etc.)

Wrapping up

- Towards compiling code fragments statically, before manipulating them dynamically.
- Register allocation?
- Administrative redeces in target code?
- Explicit shuffling of variables in target code?

Appendix

Target language

Lifting de Bruijn indices

$$\lfloor \#i \rfloor_j^n = \begin{cases} \#(i+1), & \text{if } n = 0 \text{ and } i \geq j \\ \#i, & \text{otherwise} \end{cases}$$

$$\lfloor \lambda.\ E \rfloor_j^0 = \lambda.\ \lfloor E \rfloor_{j+1}^0$$

$$\lfloor \lambda.\ E \rfloor_j^{n+1} = \lambda.\ \lfloor E \rfloor_j^{n+1}$$

$$\lfloor E_1 \mathbin{@} E_2 \rfloor_j^n = \lfloor E_1 \rfloor_j^n \mathbin{@} \lfloor E_2 \rfloor_j^n$$

$$\lfloor \uparrow E \rfloor_j^n = \uparrow \lfloor E \rfloor_j^{n+1}$$

$$\lfloor \downarrow E \rfloor_j^n = \downarrow \lfloor E \rfloor_j^{n-1}$$

$$\lfloor \text{run}\ E \rfloor_j^n = \text{run}\ \lfloor E \rfloor_j^n$$

Translation

Membership with evidence

$$\frac{}{x \in x, s \;/\; 0}$$

$$\frac{x \neq y \quad x \in s' \;/\; i}{x \in y, s' \;/\; i + 1}$$

Translation

Scope subset with evidence

$$\frac{}{\emptyset \subseteq s \;/\; \lambda E.\, E}$$

$$\frac{x \in s \;/\; i \qquad s' \subseteq x, s \;/\; F}{s', x \subseteq s \;/\; \lambda E.\, \textsf{let}\ \#i\ \textsf{in}\ F(E)}$$

Translation

Variable lookup with evidence

$$(x^n{:}t, \Gamma)[x^n] = t \;/\; 0$$

$$(y^n{:}u, \Gamma)[x^n] = t \;/\; i + 1, \qquad \text{when } \Gamma[x^n] = t \;/\; i \text{ and } x \neq y$$

$$(y^m{:}u, \Gamma)[x^n] = t \;/\; i, \qquad \text{when } \Gamma[x^n] = t \;/\; i \text{ and } m \neq n$$

Translation

Environment subset with evidence

$$\frac{}{\emptyset \subseteq \Gamma \;/\; \lambda E \,.\, E}$$

$$\frac{\Gamma[x^0] = t \;/\; i \quad \Gamma' \subseteq x^0{:}t, \Gamma \;/\; F}{\Gamma', x \subseteq \Gamma \;/\; \lambda E \,.\, \text{let}\ \#i\ \text{in}\ F(E)}$$

$$\frac{\Gamma[x^{n+1}] = t \;/\; i \quad \Gamma' \subseteq \Gamma \;/\; F}{\Gamma', x \subseteq \Gamma \;/\; F}$$

Translation

Subtyping with evidence

$$\frac{}{\mathsf{B} \leq \mathsf{B} \;/\; \lambda E\,.\,E}$$

$$\frac{t_2 \leq t_1 \;/\; F_t \qquad u_1 \leq u_2 \;/\; F_u}{t_1 \to u_1 \leq t_2 \to u_2 \;/\; \lambda E\,.\,\lambda.\, F_u(\lfloor E \rfloor_0^0 \,@\, (F_t(\#0)))}$$

$$\frac{s_1 \subseteq s_2 \;/\; F_s \qquad t_1 \leq t_2 \;/\; F_t}{[s_1]t_1 \leq [s_2]t_2 \;/\; \lambda E\,.\uparrow(F_s(\downarrow(F_t(E))))}$$