Call-by-Value in a Basic Logic for Interaction

Ulrich Schöpp
University of Munich

June 16, 2014

Introduction

Resource bounded compilation

- Logarithmic Space [Dal Lago, S]
- Hardware synthesis [Ghica, Smith]

Semantic approach

- Organise low-level programs into game semantic models: *"Low-level programs implement game semantic strategies."*
- The resulting structure can interpret higher-order languages, but is also suitable for fine-grained resource control.

Connections to standard compilation techniques

- Interpretation in one such model is related to call-by-name CPS-translation and defunctionalization [TLCA13].

Introduction

Are such semantic approaches useful for general compilation?

- Can we compile existing languages?
- Would we obtain efficient compilation methods?
- How would they relate to existing methods?
- Would we gain anything from the semantic approach?
  - Proving correctness of low-level programs?
  - Specification of low-level program behaviour?
  - Resource analysis?

Simple Source Language

Source Types	$X, Y ::= \mathbb{N} \mid X \to Y$
Source Values	$V, W ::= x \mid \lambda x{:}X.\, M \mid n$
Source Terms	$M, N ::= V \mid M\ N \mid \Omega$
	$\mid \mathsf{add}(V, W) \mid \mathsf{if0}\ V\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2$

Different evaluation strategies: call-by-value, call-by-name

Target Language

LLVM IR

```
entry:
  %a = extractvalue <{ <{}> }> %packed_arg, 0
  br label %L466
L466:                                             ; preds = %entry
  switch i1 false, label %case0 [
    i1 true, label %case1
  ]
case1:                                            ; preds = %L466
  br label %L278
case0:                                            ; preds = %L466
  br label %L280
L280:                                             ; preds = %case011, %case0
  %g = phi i64 [ 12, %case0 ], [ %g6, %case011 ]
  %x = phi i1 [ false, %case0 ], [ %slt, %case011 ]
  %i = call i64 @printf(i8* getelementptr inbounds ([3 x i8]* @format, i64 0, i64 0), i64 %g)
  %i3 = call i64 @printf(i8* getelementptr inbounds ([3 x i8]* @format1, i64 0, i64 0),
                         i64 ptrtoint ([2 x i8]* @s to i64))
  %sub = sub i64 %g, 1
  switch i1 %x, label %case05 [
    i1 true, label %case14
  ]
L278:                                             ; preds = %case110, %case1
  %g1 = phi i64 [ 12, %case1 ], [ %g6, %case110 ]
  %x2 = phi i1 [ false, %case1 ], [ %slt, %case110 ]
  ret <{ <{}> }> undef
```

Low-Level Computation

Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Values $v, w ::= \langle\rangle \mid n \mid \langle v, w \rangle \mid \mathsf{inl}(v) \mid \mathsf{inr}(v)$

A **program** is a set of blocks

$$\mathtt{f}(x : \mathsf{A})\ \{\ \mathit{body}\ \}$$

with a choice of entry and exit labels.

$$\begin{aligned}
\mathit{body} ::= {} & \mathsf{let}\ x{=}\mathit{primop}(v)\ \mathsf{in}\ \mathit{body} \\
\mid {} & \mathsf{let}\ \langle x, y \rangle{=}v\ \mathsf{in}\ \mathit{body} \\
\mid {} & \mathsf{case}\ v\ \mathsf{of}\ \mathsf{inl}(x) \Rightarrow \mathit{body}_1 \\
& \qquad\qquad ;\ \mathsf{inr}(y) \Rightarrow \mathit{body}_2 \\
\mid {} & \mathtt{g}(v)
\end{aligned}$$

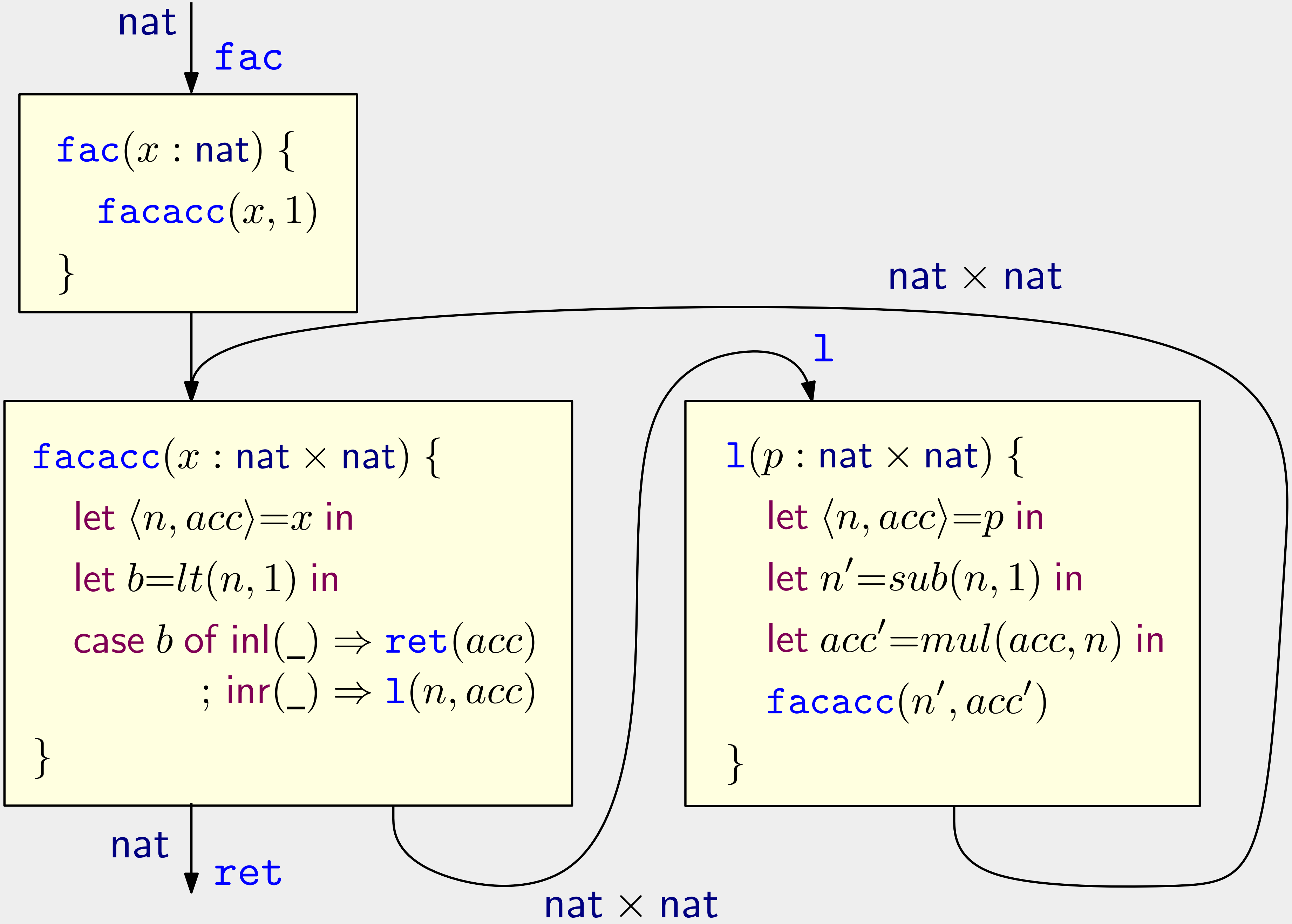
Low-Level Computation

```
$\texttt{fac}(x : \mathsf{nat})\ \{$
    $\texttt{facacc}(x, 1)$
$\}$
```

```
$\texttt{facacc}(x : \mathsf{nat} \times \mathsf{nat})\ \{$
    let $\langle n, acc \rangle = x$ in
    let $b = lt(n, 1)$ in
    case $b$ of inl(_) $\Rightarrow \texttt{ret}(acc)$
              ; inr(_) $\Rightarrow \texttt{l}(n, acc)$
$\}$
```

```
$\texttt{l}(p : \mathsf{nat} \times \mathsf{nat})\ \{$
    let $\langle n, acc \rangle = p$ in
    let $n' = sub(n, 1)$ in
    let $acc' = mul(acc, n)$ in
    $\texttt{facacc}(n', acc')$
$\}$
```

Low-Level Computation



nat

fac

```
fac($x$ : nat) {
   facacc($x, 1$)
}
```

nat × nat

l

```
facacc($x$ : nat × nat) {
   let $\langle n, acc\rangle = x$ in
   let $b = lt(n, 1)$ in
   case $b$ of inl(_) ⇒ ret($acc$)
            ; inr(_) ⇒ l($n, acc$)
}
```

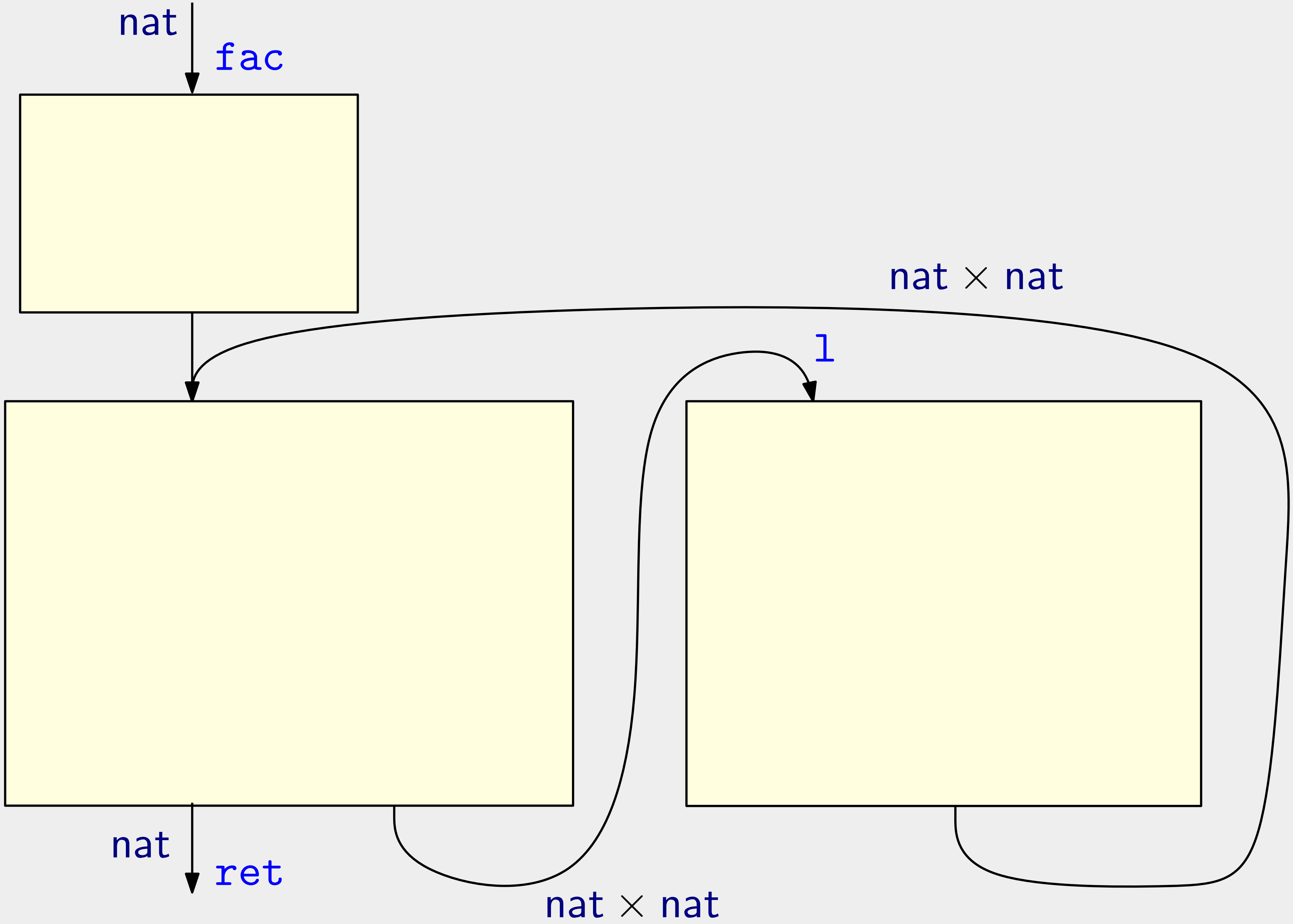
```
l($p$ : nat × nat) {
   let $\langle n, acc\rangle = p$ in
   let $n' = sub(n, 1)$ in
   let $acc' = mul(acc, n)$ in
   facacc($n', acc'$)
}
```

nat

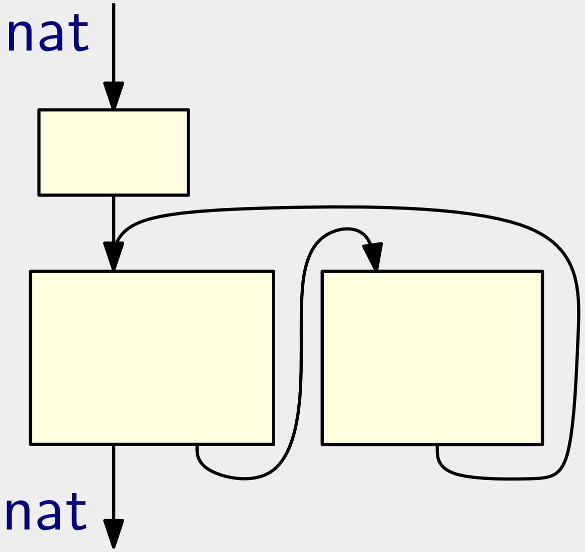
ret

nat × nat

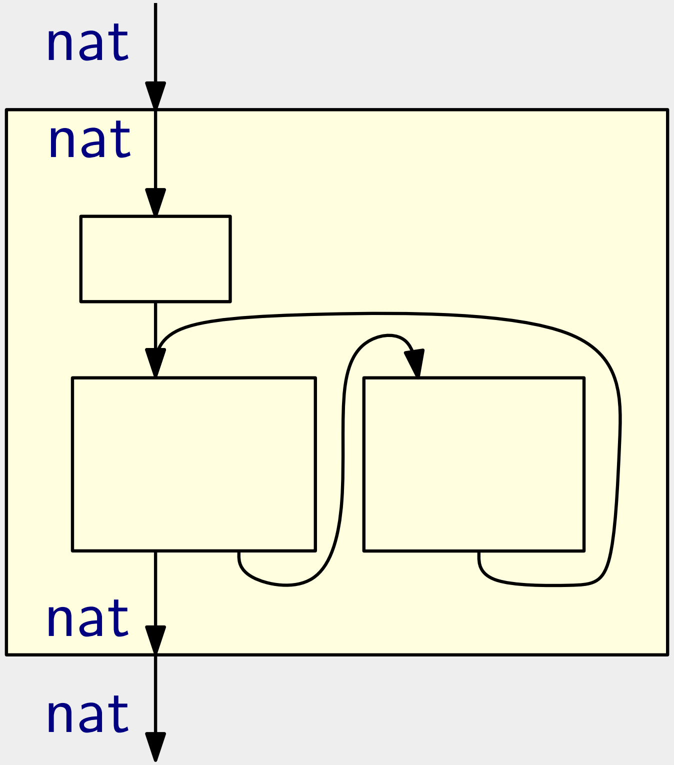
Low-Level Computation



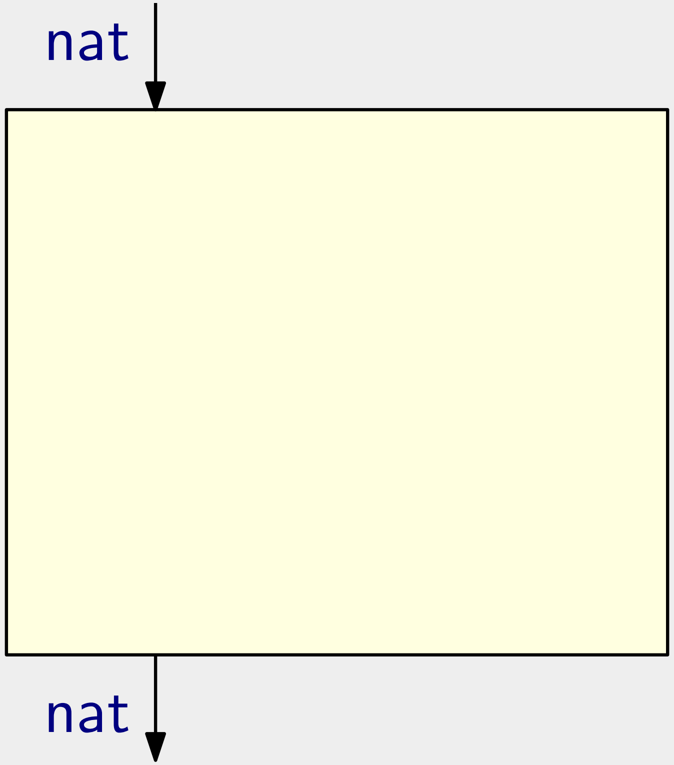
Low-Level Computation



Low-Level Computation



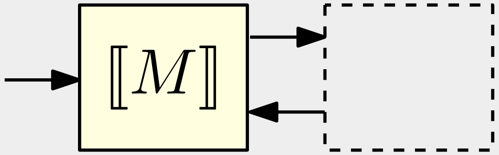
Low-Level Computation



Organizing Low-Level Computation

Some issues in compilation

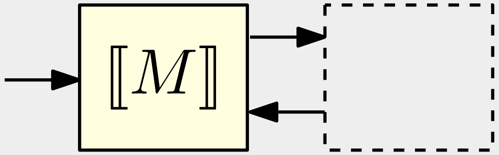
- Parameterisation



Organizing Low-Level Computation

Some issues in compilation

- Parameterisation

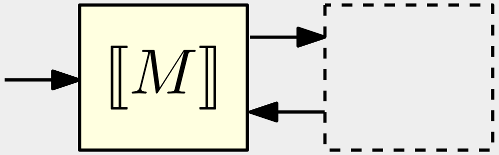


- Interface specification

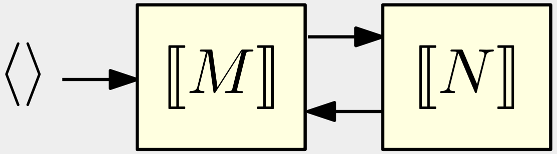
Organizing Low-Level Computation

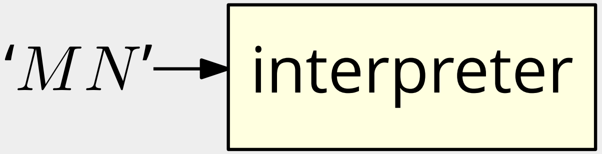
Some issues in compilation

- Parameterisation



- Interface specification
- Values vs. Computation

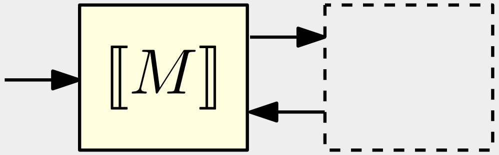




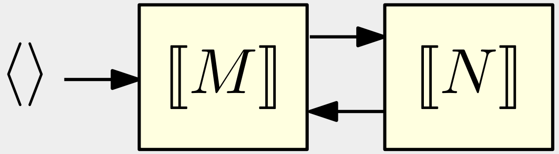
Organizing Low-Level Computation

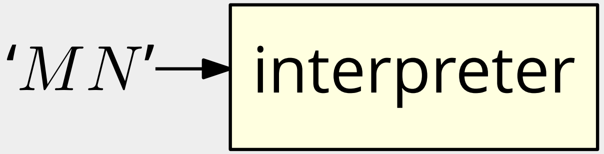
Some issues in compilation

- Parameterisation



- Interface specification
- Values vs. Computation



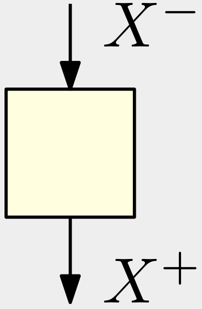


- Value management
  - encoding
  - tail calls
  - stack management
  - space usage

Organizing Low-Level Computation

Value Types $A, B \ ::= \ \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y \ ::= \ TA \mid A \to X \mid A \cdot X \multimap Y \mid \forall \alpha.\, X$

$t\colon X$ represents 

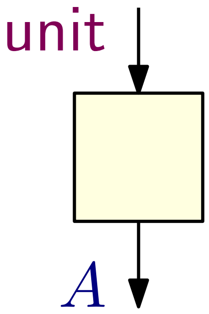
Effect PCF [Filinski], Call by Push Value [Levy],
Enriched Effect Calculus [Møgelberg & Simpson]

Organizing Low-Level Computation

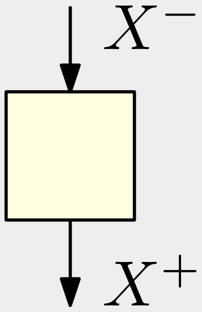
Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y ::= TA \mid A \to X \mid A \cdot X \multimap Y \mid \forall \alpha . X$

$$TA^- = \mathsf{unit}$$
$$TA^+ = A$$



ents



Effect PCF [Filinski], Call by Push Value [Levy],
Enriched Effect Calculus [Møgelberg & Simpson]

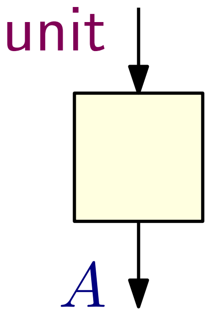
Organizing Low-Level Computation

Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

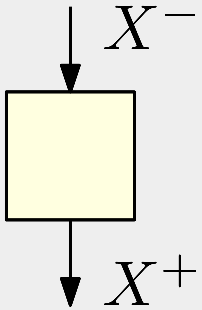
Computation Types $X, Y ::= TA \mid A \to X \mid A \cdot X \multimap Y \mid \forall \alpha.\, X$

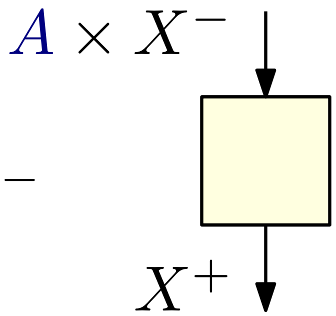
$TA^- = \mathsf{unit}$

$TA^+ = A$



ents





$(A \to X)^- = A \times X^-$

$(A \to X)^+ = X^+$

lue [Levy],

g & Simpson]

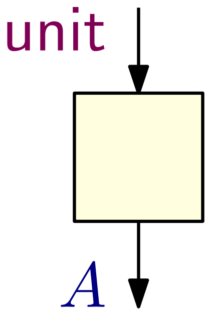
Organizing Low-Level Computation

Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

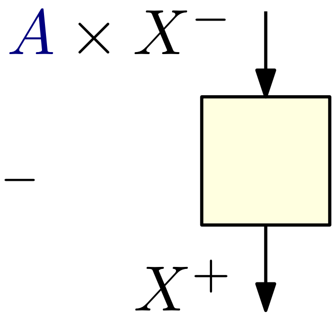
Computation Types $X, Y ::= TA \mid A \to X \mid A \cdot X \multimap Y \mid \forall \alpha. X$

$$TA^- = \mathsf{unit}$$

$$TA^+ = A$$



ent



$$(A \to X)^- = A \times X^-$$

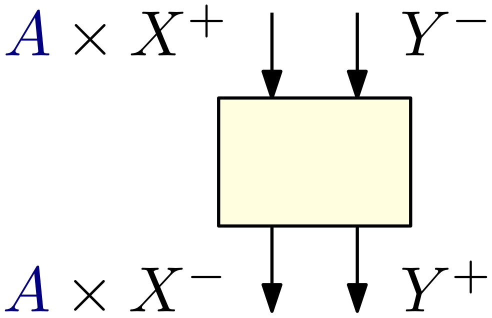
$$(A \to X)^+ = X^+$$

lue

g &

$$(A \cdot X \multimap Y)^- = A \times X^+ + Y^-$$

$$(A \cdot X \multimap Y)^+ = A \times X^- + Y^+$$

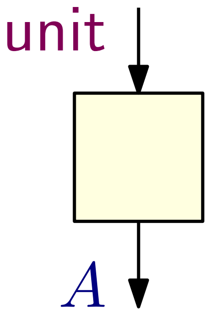


Organizing Low-Level Computation

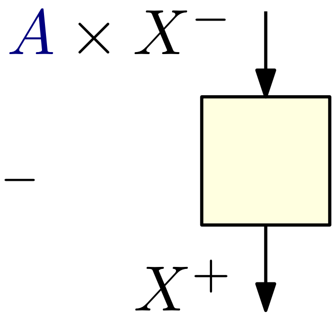
Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y ::= TA \mid A \to X \mid A \cdot X \multimap Y \mid \forall \alpha. X$

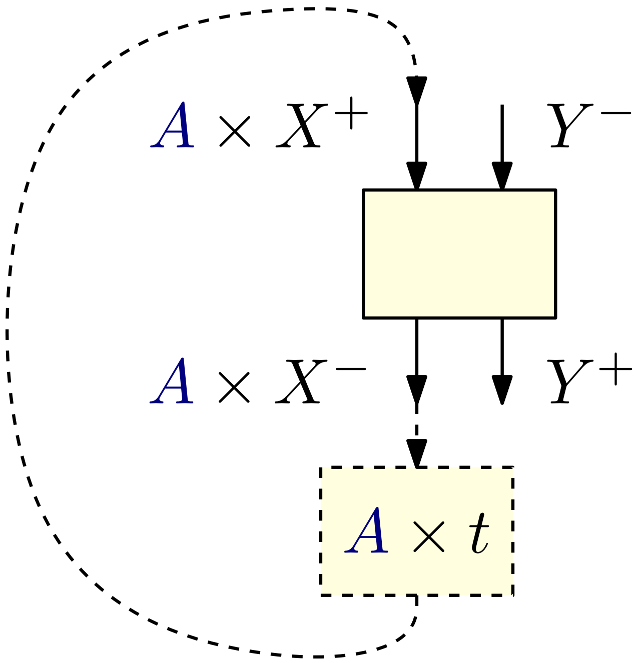
$$TA^- = \mathsf{unit}$$
$$TA^+ = A$$



$$(A \to X)^- = A \times X^-$$
$$(A \to X)^+ = X^+$$



$$(A \cdot X \multimap Y)^- = A \times X^+ + Y^-$$
$$(A \cdot X \multimap Y)^+ = A \times X^- + Y^+$$



en

ue

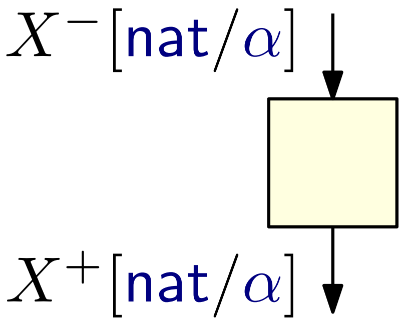
g &

ation

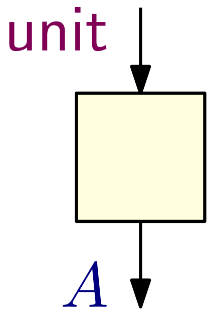
unit | $A \times B$ | 0 | $A + B$

$\to X$ | $A \cdot X \multimap Y$ | $\forall \alpha.\, X$

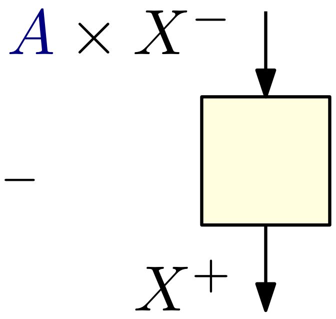
$$(\forall \alpha. X)^- = X^-[\mathsf{nat}/\alpha]$$
$$(\forall \alpha. X)^+ = X^+[\mathsf{nat}/\alpha]$$



$$TA^- = \mathsf{unit}$$
$$TA^+ = A$$



ent

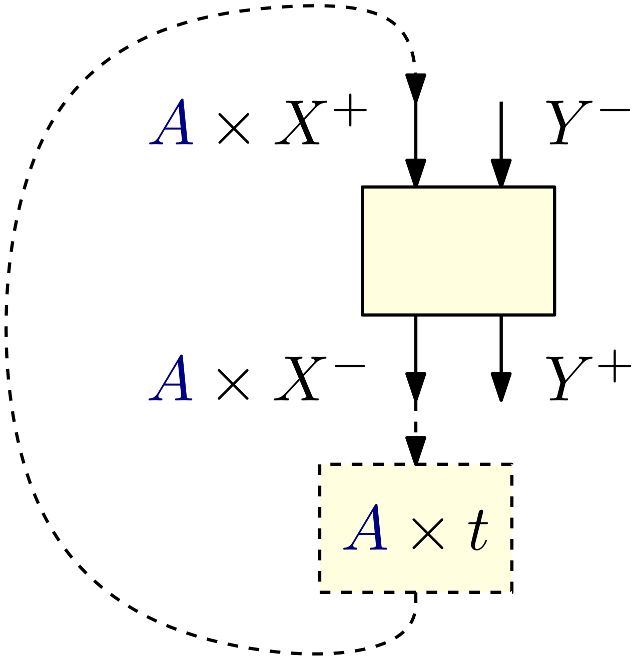


$$(A \to X)^- = A \times X^-$$
$$(A \to X)^+ = X^+$$

lue

g &

$$(A \cdot X \multimap Y)^- = A \times X^+ + Y^-$$
$$(A \cdot X \multimap Y)^+ = A \times X^- + Y^+$$



Call-by-Name

Handled nicely in the fragment

$$
\begin{aligned}
X, Y \; &::= \; T A \mid A \cdot X \multimap Y \\
s, t \; &::= \; \mathsf{return}(v) \mid \mathsf{let}\ x{=}s\ \mathsf{in}\ t \mid \lambda x{:}X.\, t \mid s\, t
\end{aligned}
$$

(used by [Dal Lago, S.], [Ghica, Smith])

Easily extended to compile PCF or Idealized Algol

- C-like stack management
- efficient compilation
  (related to CPS-translation and defunctionalization [TLCA13])
- separate compilation
- stack shape inference
- soundness proofs

Call-by-Value?

Not immediate

- How to represent function values?
- Computations are not values (as in other effect calculi).

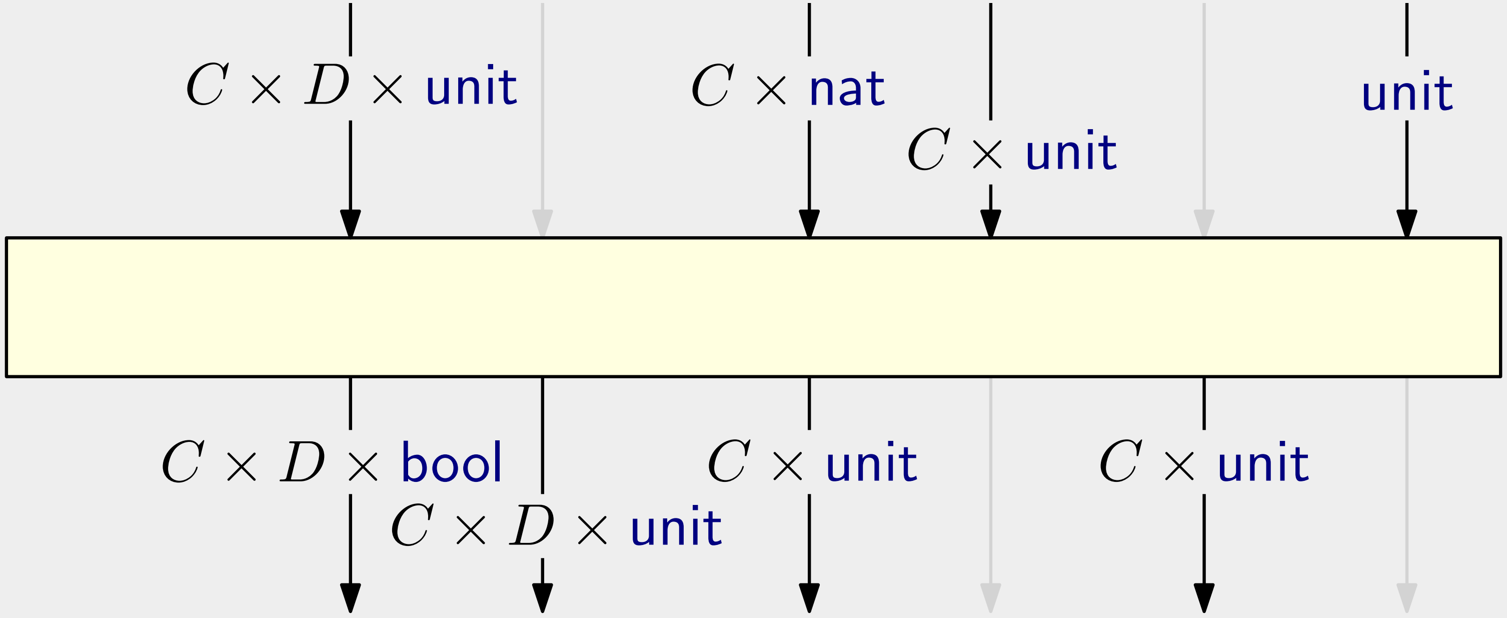
Call-by-value CPS-translation [Plotkin 1975]

$$
\begin{aligned}
\mathsf{cps}(x) &= \lambda k.\, k\ x \\
\mathsf{cps}(n) &= \lambda k.\, k\ n \\
\mathsf{cps}(\lambda x.\, M) &= \lambda k.\, k\ (\lambda k_1.\, \lambda x.\, \mathsf{cps}(M)\ k_1) \\
\mathsf{cps}(M\ \ N) &= \lambda k.\, \mathsf{cps}(M)\ (\lambda f.\, \mathsf{cps}(N)\ (\lambda x.\, f\ \ k\ \ x)) \\
\mathsf{cps}(\mathsf{add}(V, W)) &= \lambda k.\, \mathsf{cps}(V)\ (\lambda x.\, \mathsf{cps}(W)\ (\lambda y.\, k\ (x + y)))
\end{aligned}
$$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot) \big) \multimap \bot \Big) \multimap \bot$$

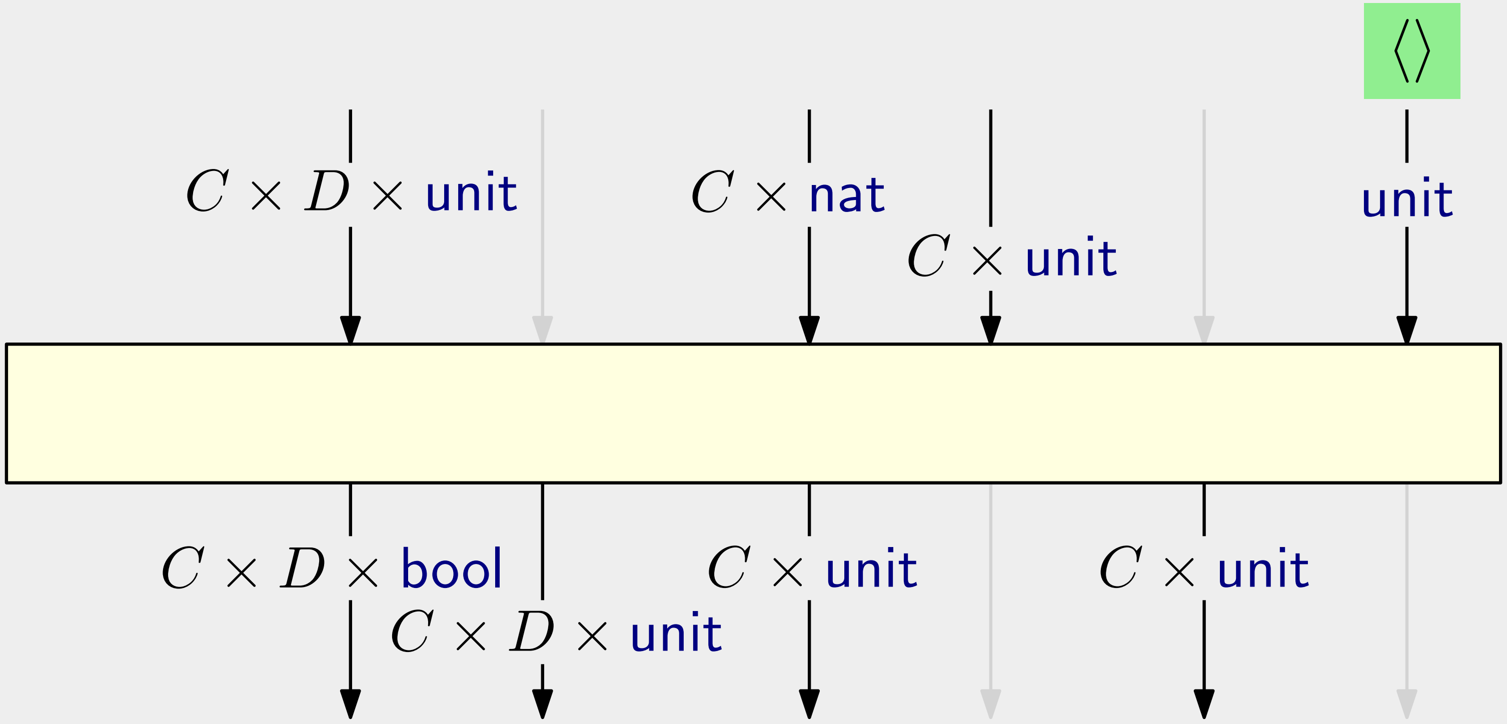


$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot) \big) \multimap \bot \Big) \multimap \bot$$

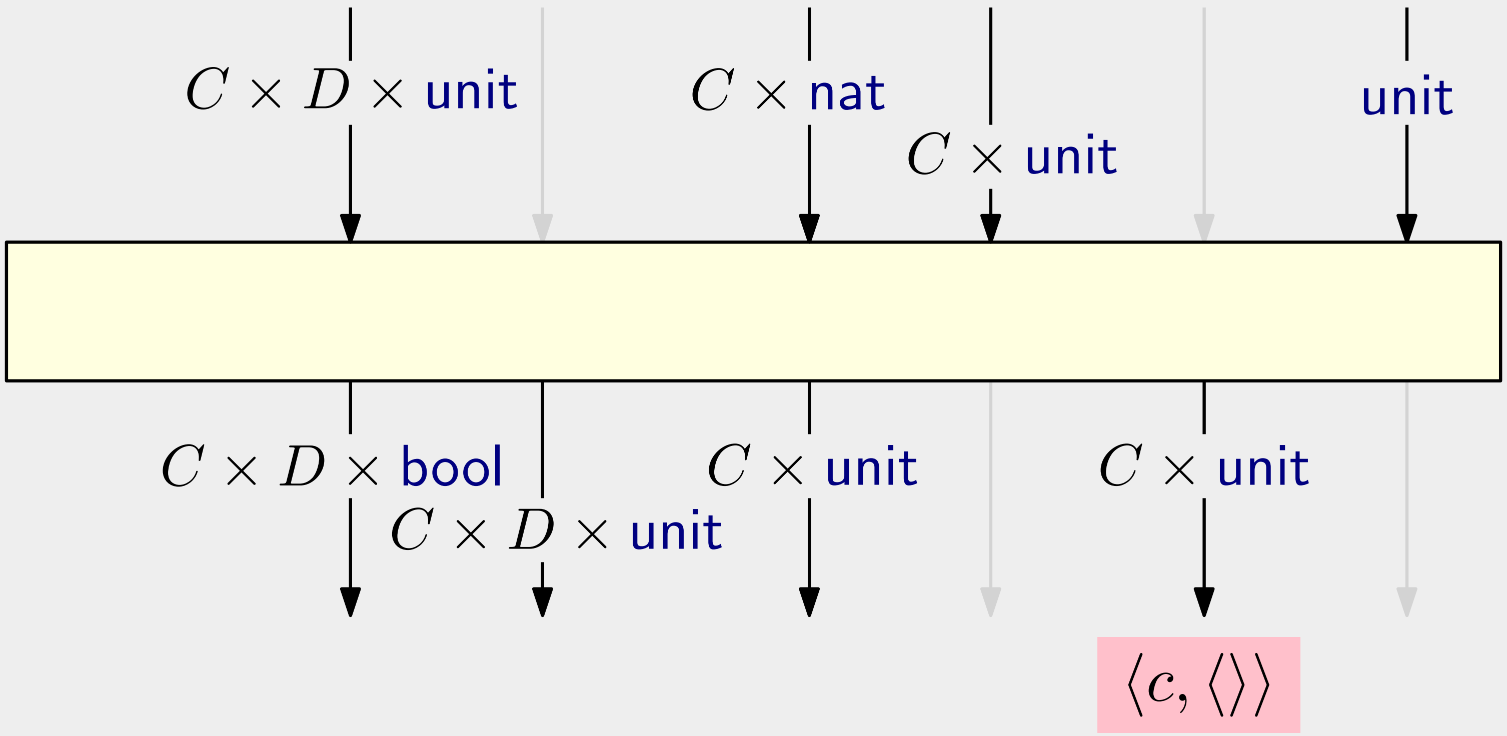


$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot)\big) \multimap \bot\Big) \multimap \bot$$

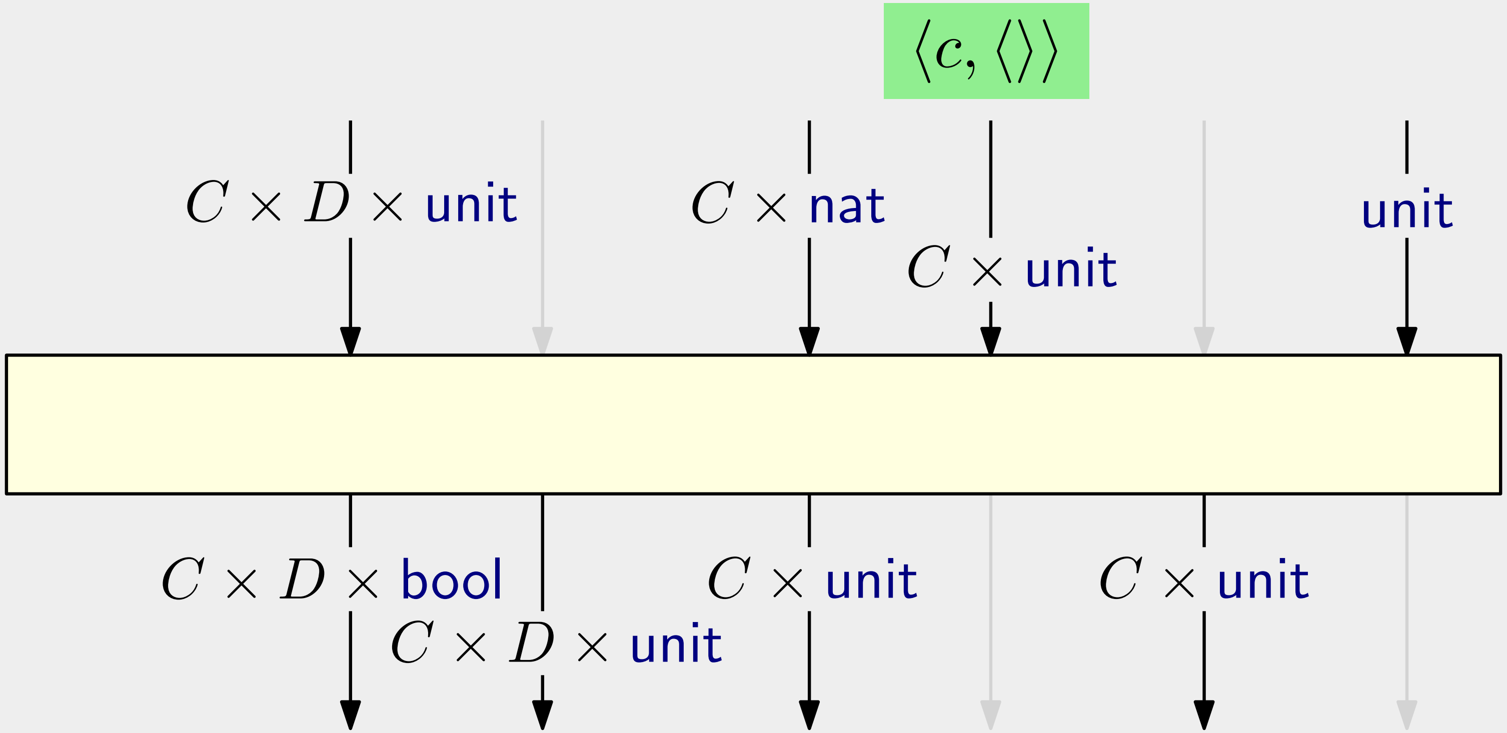


$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot) \big) \multimap \bot \Big) \multimap \bot$$

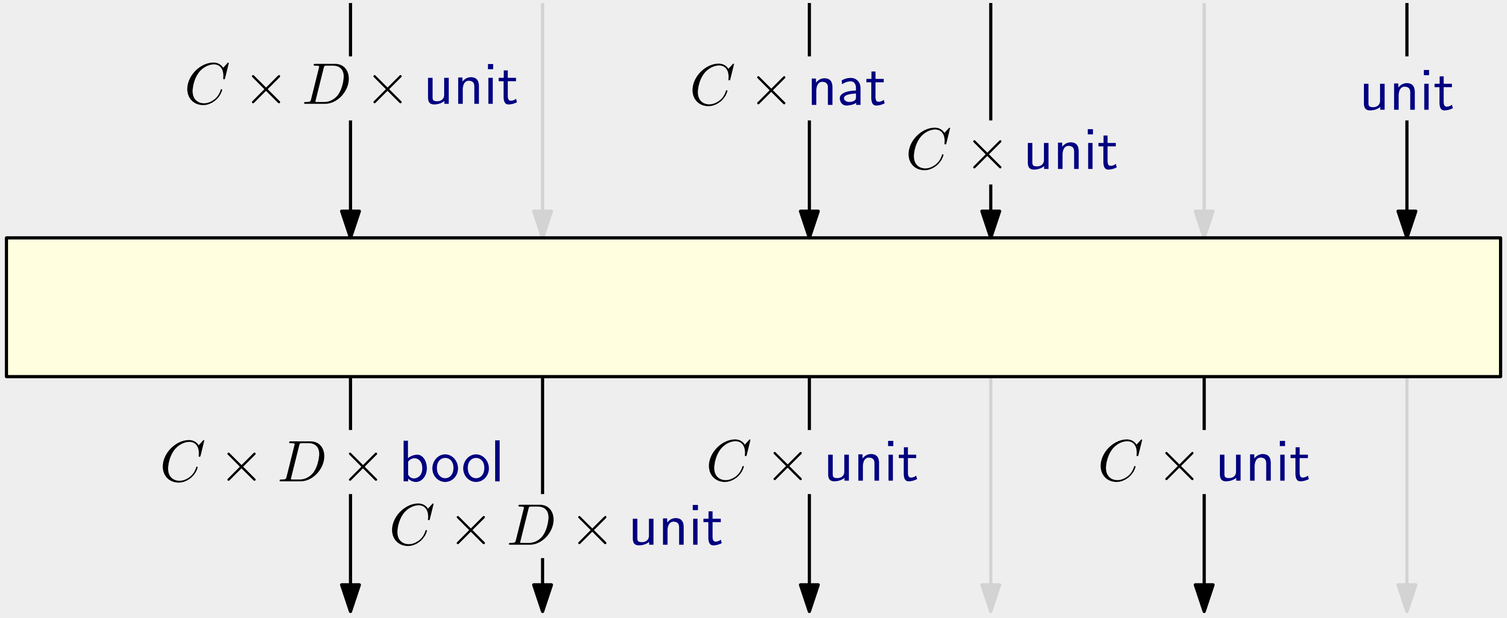

$\langle c, \langle\rangle\rangle$
$C \times D \times \mathsf{unit}$
$C \times \mathsf{nat}$
$C \times \mathsf{unit}$
unit
$C \times D \times \mathsf{bool}$
$C \times D \times \mathsf{unit}$
$C \times \mathsf{unit}$
$C \times \mathsf{unit}$

$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot) \big) \multimap \bot \Big) \multimap \bot$$



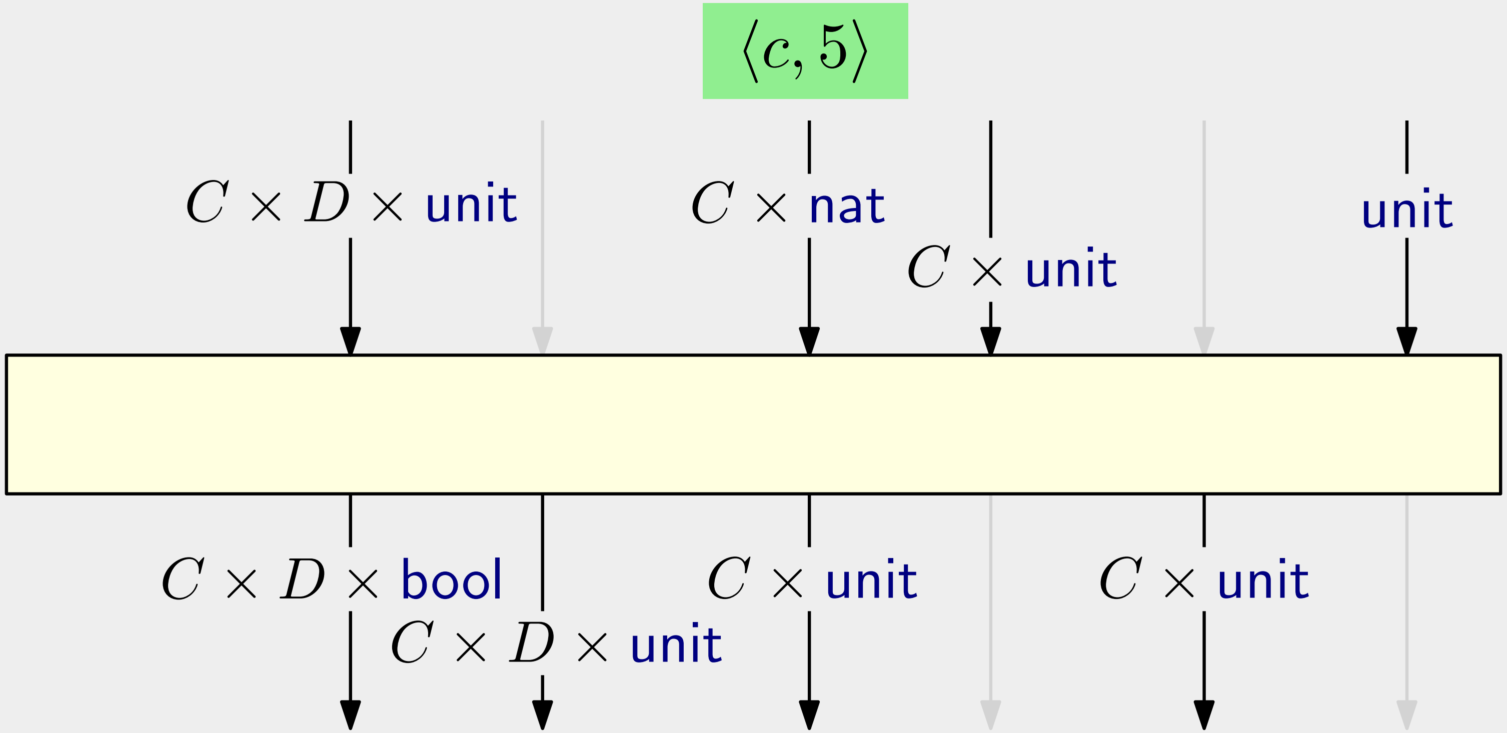
$\langle c, \langle\rangle \rangle$

$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot)\big) \multimap \bot\Big) \multimap \bot$$



$\langle c, 5 \rangle$

$C \times D \times \mathsf{unit}$

$C \times \mathsf{nat}$

$C \times \mathsf{unit}$

unit

$C \times D \times \mathsf{bool}$

$C \times D \times \mathsf{unit}$

$C \times \mathsf{unit}$

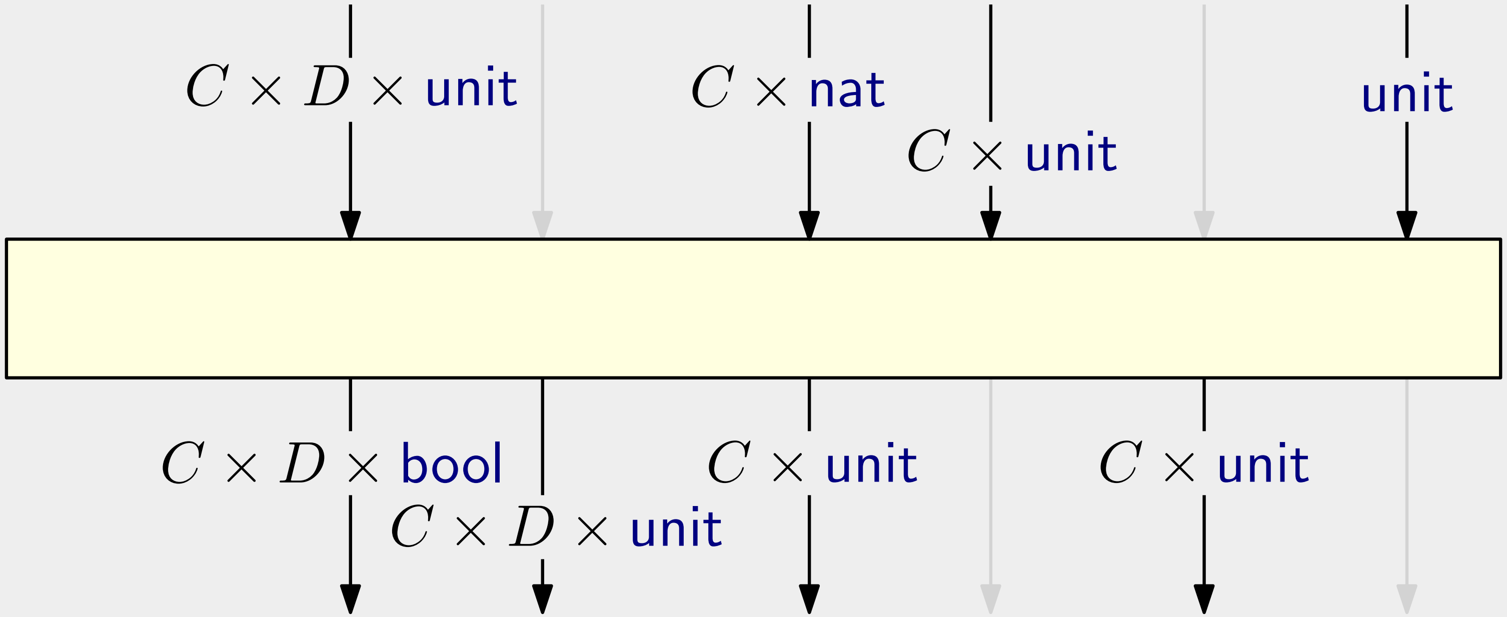
$C \times \mathsf{unit}$

$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot)\big) \multimap \bot\Big) \multimap \bot$$



$C \times D \times \mathsf{unit}$

$C \times \mathsf{nat}$

$C \times \mathsf{unit}$

unit

$C \times D \times \mathsf{bool}$

$C \times D \times \mathsf{unit}$

$C \times \mathsf{unit}$

$C \times \mathsf{unit}$

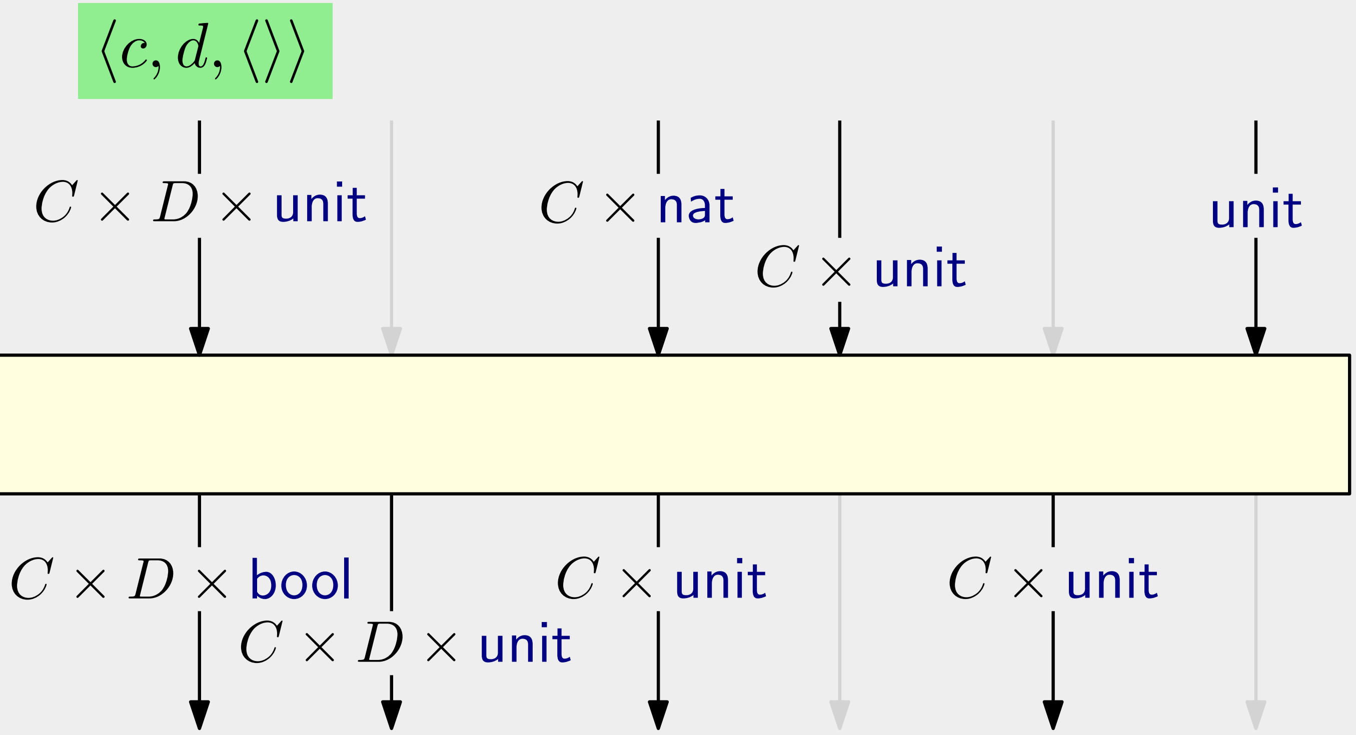
$\langle c, d, \langle\rangle\rangle$

$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot)\big) \multimap \bot\Big) \multimap \bot$$

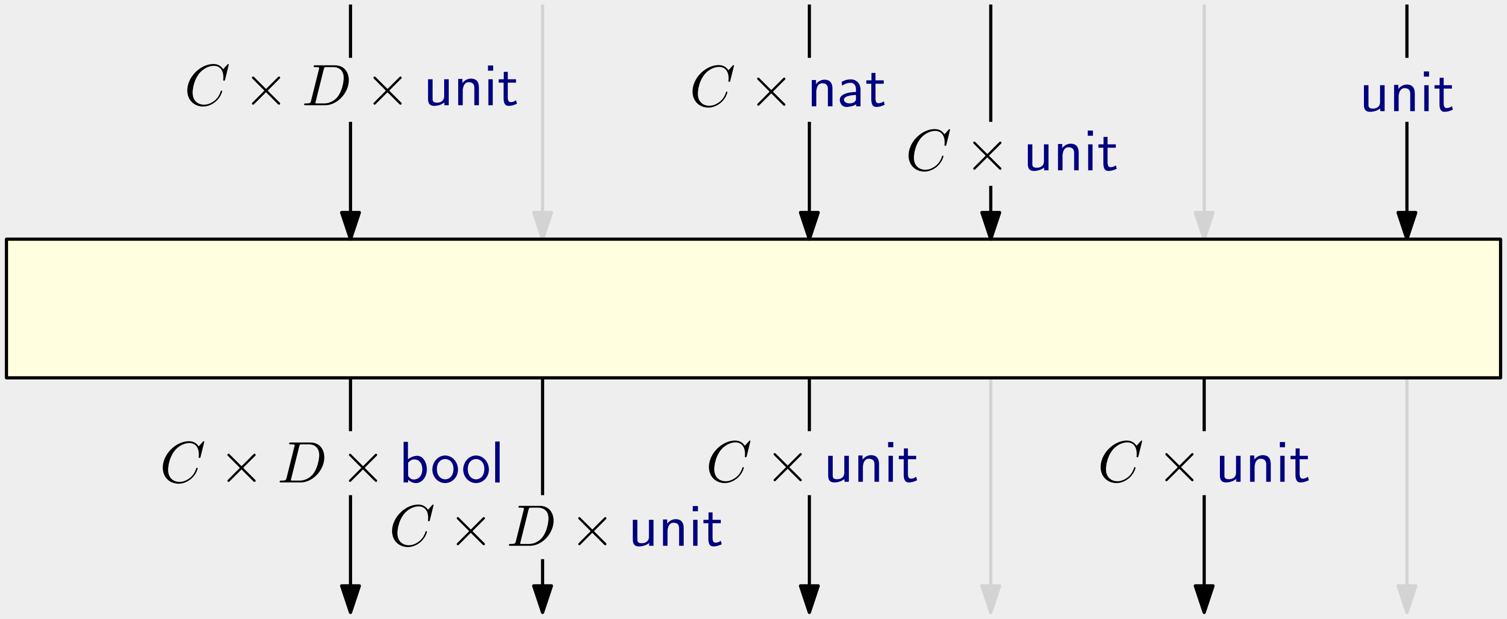


$\bot = T0$

Call-by-Value?

Example: Translation of a function $\vdash M : \mathbb{N} \to \mathbb{B}$.

$$C \cdot \Big(\big(D \cdot (T\mathsf{bool} \multimap \bot) \multimap (T\mathsf{nat} \multimap \bot)\big) \multimap \bot\Big) \multimap \bot$$



$C \times D \times$ unit

$C \times$ nat

$C \times$ unit

unit

$C \times D \times$ bool

$C \times D \times$ unit

$C \times$ unit

$C \times$ unit

$\langle c, d, \mathsf{true} \rangle$

$\bot = T0$

Call-by-Value

Problem: Translation is not *safe for space* [Appel].

$$\begin{aligned} \text{let } x_1 &= M_1 \text{ in} \\ \text{let } x_2 &= M_2 \text{ in} \\ \ldots \\ \text{let } x_k &= M_k \text{ in } N \end{aligned}$$

$$(C_1 \times \cdots \times C_k) \cdot (X \multimap \bot) \multimap \bot$$

Values are deallocated only when a continuation returns (= never).

Organizing Low-Level Computation

Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y ::= T A \mid A \to X \mid A \cdot X \multimap Y \mid \forall \alpha. X$

Organizing Low-Level Computation

Value Types $A, B \ ::= \ \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y \ ::= \ \bot^{A} \mid X \multimap Y \mid \forall \alpha.\, X$

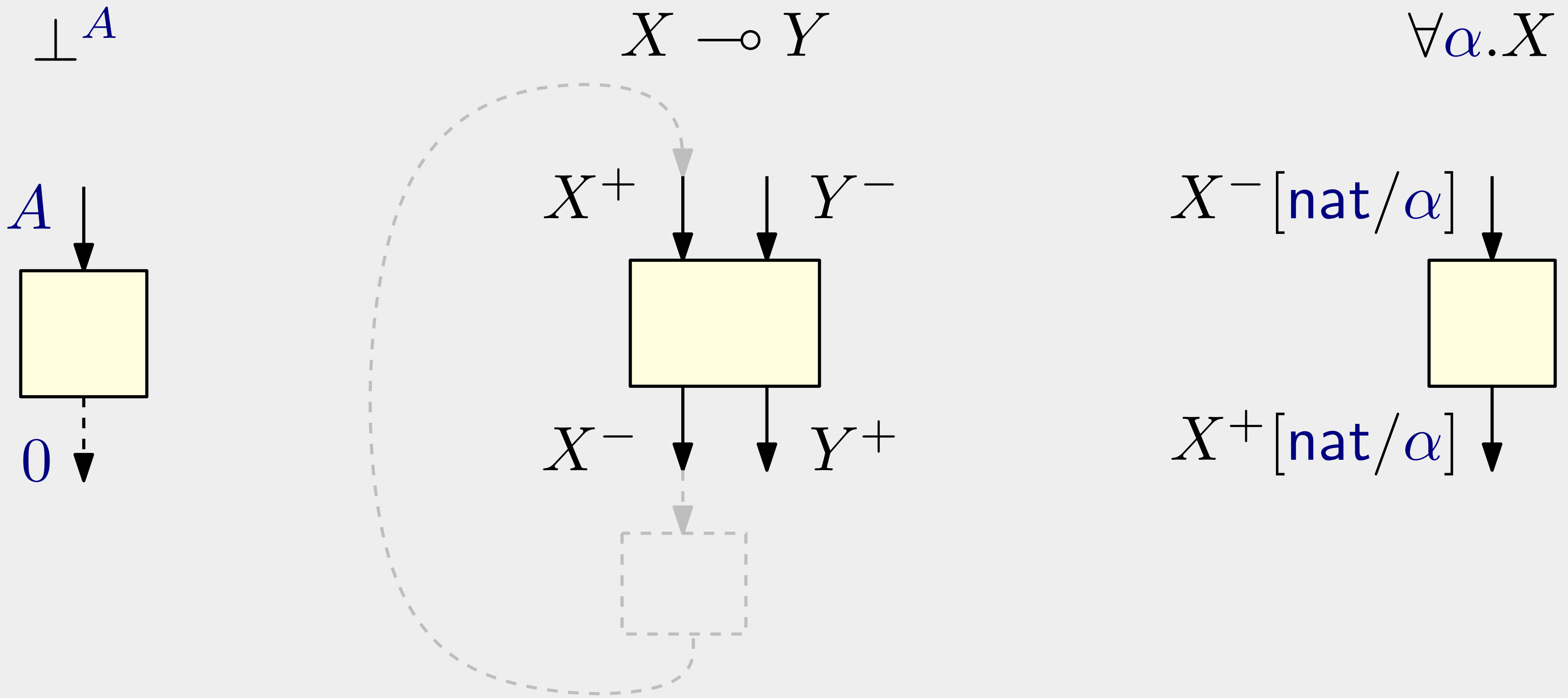
$$(\bot^{A} \cong A \to T0)$$

Organizing Low-Level Computation

Value Types $A, B ::= \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y ::= \bot^{A} \mid X \multimap Y \mid \forall \alpha.\, X$

$$(\bot^{A} \cong A \to T0)$$



Study the *linear* source language first.

Organizing Low-Level Computation

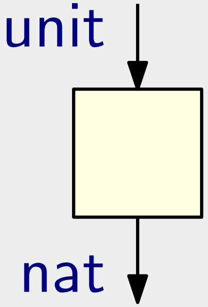
Rules (selection)

$$\text{0}\ \frac{}{\vdash \star\colon \bot^{0}} \qquad \text{ACT}\ \frac{x\colon A \vdash v\colon B \qquad \Gamma \vdash t\colon \bot^{B}}{\Gamma \vdash (x \mapsto v)^{*} t\colon \bot^{A}}$$

$$\multimap\text{I}\ \frac{\Gamma, x\colon X \vdash t\colon Y}{\Gamma \vdash \lambda x{:}X.\, t\colon X \multimap Y} \qquad \multimap\text{E}\ \frac{\Gamma \vdash s\colon X \multimap Y \qquad \Delta \vdash t\colon X}{\Gamma, \Delta \vdash s\, t\colon Y}$$

$$\forall\text{I}\ \frac{\Gamma \vdash t\colon X}{\Gamma \vdash \Lambda\alpha.\, t\colon \forall\alpha.\, X}\ \alpha \text{ not in } \Gamma \qquad \forall\text{E}\ \frac{\Gamma \vdash t\colon \forall\alpha.\, X}{\Gamma \vdash t\, A\colon X[A/\alpha]}$$

Call-by-Name



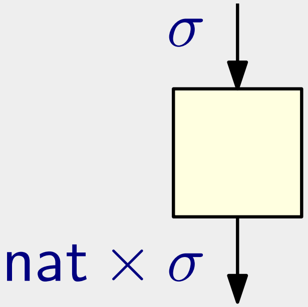
$$\begin{aligned} [\![\mathbb{N}]\!] &= \bot^{\mathsf{nat}} \multimap \bot \\ [\![X \to Y]\!] &= [\![X]\!] \multimap [\![Y]\!] \end{aligned}$$

$$\begin{aligned} \mathsf{cps}(n) &= \lambda k.\, (\langle\rangle \mapsto n)^{*} k \\ \mathsf{cps}(\lambda x{:}X.\, M) &= \lambda x.\mathsf{cps}(M) \\ \mathsf{cps}(M\ N) &= \mathsf{cps}(M)\ \mathsf{cps}(N) \\ \mathsf{cps}(\mathsf{add}(M, N)) &= ? \\ &\dots \end{aligned}$$

Call-by-Name

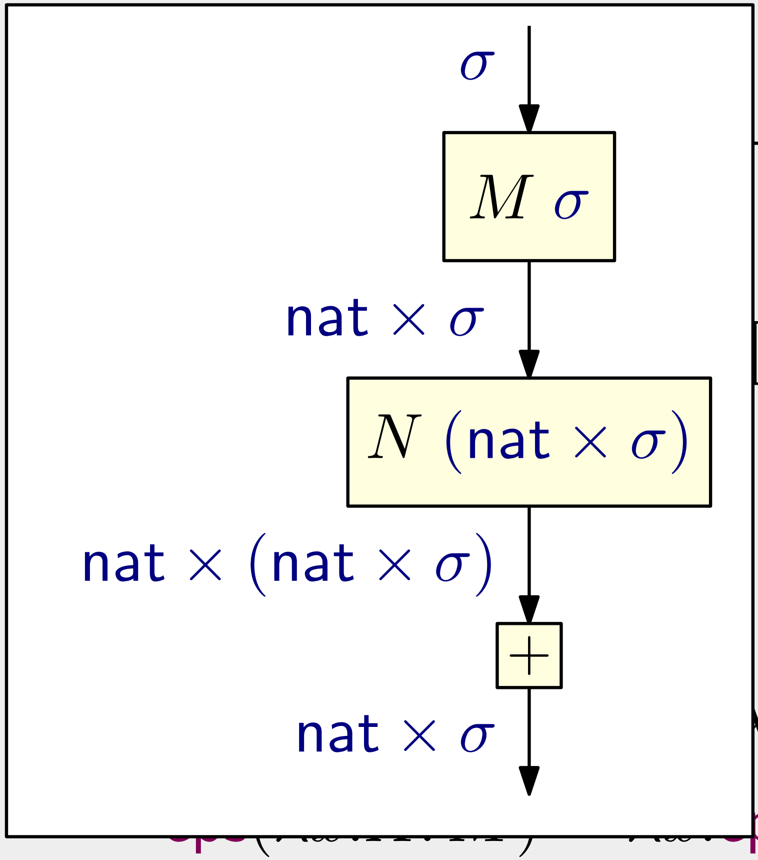
$$[\![\mathbb{N}]\!] = \forall\sigma.\, \bot^{(\mathsf{nat}\times\sigma)} \multimap \bot^{\sigma}$$

$$[\![X \to Y]\!] = [\![X]\!] \multimap [\![Y]\!]$$



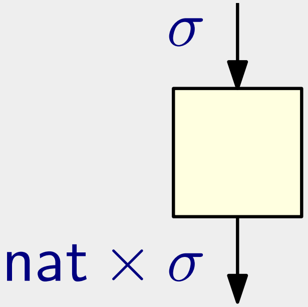
$$\begin{aligned}
\mathsf{cps}(n) &= \Lambda\sigma.\, \lambda k.\, (s \mapsto \langle s, n\rangle)^{*} k \\
\mathsf{cps}(\lambda x{:}X.\, M) &= \lambda x.\mathsf{cps}(M) \\
\mathsf{cps}(M\ N) &= \mathsf{cps}(M)\ \mathsf{cps}(N) \\
\mathsf{cps}(\mathsf{add}(M, N)) &= \Lambda\sigma.\, \lambda k.\, M\ \sigma\ (N\ \ (\mathsf{nat}\times\sigma) \\
&\qquad\qquad\qquad\qquad (((\langle m, \langle n, s\rangle\rangle \mapsto \langle m+n, s\rangle)^{*} k) \\
&\ldots
\end{aligned}$$

$$(TA := \forall\sigma.\, \bot^{(A\times\sigma)} \multimap \bot^{\sigma})$$



$$\mathbb{N}]\!] = \forall\sigma.\,\bot^{(\mathsf{nat}\times\sigma)} \multimap \bot^{\sigma}$$

$$Y]\!] = [\![X]\!] \multimap [\![Y]\!]$$



$$\lambda k.\,(s \mapsto \langle s, n\rangle)^{*} k$$

$$\mathsf{ps}(M)$$

$$\mathsf{cps}(M\ N) = \mathsf{cps}(M)\ \mathsf{cps}(N)$$

$$\mathsf{cps}(\mathsf{add}(M,N)) = \Lambda\sigma.\,\lambda k.\,M\ \sigma\ (N\ \ (\mathsf{nat}\times\sigma)$$
$$((\langle m, \langle n, s\rangle\rangle \mapsto \langle m+n, s\rangle)^{*} k)$$

$$\ldots$$

$(TA := \forall\sigma.\,\bot^{(A\times\sigma)} \multimap \bot^{\sigma})$

Call-by-Value

Standard CPS-translation [Plotkin 1975]

$$
\begin{aligned}
\mathsf{cps}(x) &= \lambda k.\, k\ x \\
\mathsf{cps}(n) &= \lambda k.\, k\ n \\
\mathsf{cps}(\lambda x.\, M) &= \lambda k.\, k\ (\lambda k_1.\, \lambda x.\, \mathsf{cps}(M)\ k_1) \\
\mathsf{cps}(M\ \ N) &= \lambda k.\, \mathsf{cps}(M)\ (\lambda f.\, \mathsf{cps}(N)\ (\lambda x.\, f\ \ k\ x)) \\
\mathsf{cps}(\mathsf{add}(V, W)) &= \lambda k.\, \mathsf{cps}(V)\ (\lambda x.\, \mathsf{cps}(W)\ (\lambda y.\, k\ (x + y)))
\end{aligned}
$$

$$
\begin{gathered}
x_1 \colon X_1, \ldots, x_n \colon X_n \vdash M \colon X \\
\Longrightarrow \\
x_1 \colon \mathcal{A}(X_1), \ldots, x_n \colon \mathcal{A}(X_n) \vdash \mathsf{cps}(M) \colon \mathcal{T}(X)
\end{gathered}
$$

$$
\begin{aligned}
\mathcal{T}(X) &= \mathcal{K}(X) \to \bot & \mathcal{A}(\mathbb{N}) &= \mathbb{N} \\
\mathcal{K}(X) &= \mathcal{A}(X) \to \bot & \mathcal{A}(X \to Y) &= \mathcal{K}(Y) \to \mathcal{K}(X)
\end{aligned}
$$

Refining the Translation

CPS-Translation

$$x_1 : \mathcal{A}(X_1), \ldots, x_n : \mathcal{A}(X_n) \vdash \mathsf{cps}(M) : \mathcal{T}(X)$$

Refinement

$$x_1 : \mathcal{A}_{\varphi_1}(X_1), \ldots, x_n : \mathcal{A}_{\varphi_n}(X_n) \vdash \mathsf{cps}(M) : \mathcal{T}_{\mathcal{C}_{\varphi_1}(X_1) \times \cdots \times \mathcal{C}_{\varphi_n}(X_n)}(X)$$

Refining the Translation

CPS-Translation

$$x_1 : \mathcal{A}(X_1), \ldots, x_n : \mathcal{A}(X_n) \vdash \mathsf{cps}(M) : \mathcal{T}(X)$$

$$\mathcal{T}(X) = \mathcal{K}(X) \to \bot$$
$$\mathcal{K}(X) = \mathcal{A}(X) \to \bot$$

Refinement

$$x_1 : \mathcal{A}_{\varphi_1}(X_1), \ldots, x_n : \mathcal{A}_{\varphi_n}(X_n) \vdash \mathsf{cps}(M) : \mathcal{T}_{\mathcal{C}_{\varphi_1}(X_1) \times \cdots \times \mathcal{C}_{\varphi_n}(X_n)}(X)$$

$$\mathcal{T}_\gamma(X) = \forall \sigma.\, \mathcal{K}_\sigma(X) \multimap \bot^{(\gamma \times \sigma)}$$
$$\mathcal{K}_\sigma(X) = \forall \varphi.\, \mathcal{A}_\varphi(X) \multimap \bot^{(\mathcal{C}_\varphi(X) \times \sigma)}$$

Refining the Translation

CPS-Translation

$$x_1 \colon \mathcal{A}(X_1), \ldots, x_n \colon \mathcal{A}(X_n) \vdash \mathsf{cps}(M) \colon \mathcal{T}(X)$$

$$\mathcal{T}(X) = \mathcal{K}(X) \to \bot \qquad\qquad \mathcal{A}(\mathbb{N}) = \mathbb{N}$$

$$\mathcal{K}(X) = \mathcal{A}(X) \to \bot \qquad\qquad \mathcal{A}(X \to Y) = \mathcal{K}(Y) \to \mathcal{K}(X)$$

Refinement

$$x_1 \colon \mathcal{A}_{\varphi_1}(X_1), \ldots, x_n \colon \mathcal{A}_{\varphi_n}(X_n) \vdash \mathsf{cps}(M) \colon \mathcal{T}_{\mathcal{C}_{\varphi_1}(X_1) \times \cdots \times \mathcal{C}_{\varphi_n}(X_n)}(X)$$

$$\mathcal{T}_{\gamma}(X) = \forall \sigma.\, \mathcal{K}_{\sigma}(X) \multimap \bot^{(\gamma \times \sigma)}$$

$$\mathcal{K}_{\sigma}(X) = \forall \varphi.\, \mathcal{A}_{\varphi}(X) \multimap \bot^{(\mathcal{C}_{\varphi}(X) \times \sigma)}$$

$$\mathcal{C}_{\varphi}(\mathbb{N}) = \mathsf{nat} \qquad\qquad \mathcal{A}_{\varphi}(\mathbb{N}) = \bot^{0}$$

$$\mathcal{C}_{\varphi}(X \to Y) = \varphi \qquad\qquad \mathcal{A}_{\varphi}(X \to Y) = \forall \tau.\, \mathcal{K}_{\tau}(Y) \multimap \mathcal{K}_{\varphi \times \tau}(X)$$

Code Values, Access Programs, Continuations

Idea: Encode (source) values of type X by values of type

$$\exists\varphi.\,\mathcal{C}_\varphi(X) \times \mathcal{A}_\varphi(X)$$

Code types (value types)

$$\begin{aligned}\mathcal{C}_\varphi(\mathbb{N}) &= \mathsf{nat}\\ \mathcal{C}_\varphi(X \to Y) &= \varphi\end{aligned}$$

Access types (computation types)

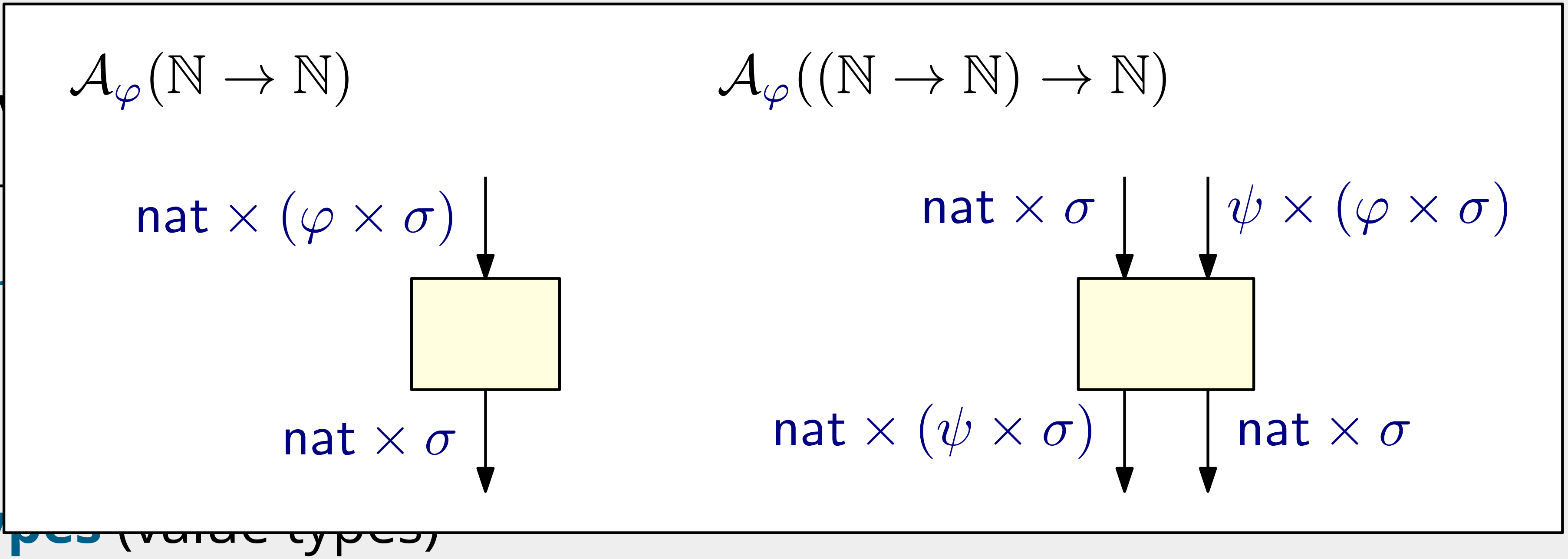
$$\begin{aligned}\mathcal{A}_\varphi(\mathbb{N}) &= \bot^{0}\\ \mathcal{A}_\varphi(X \to Y) &= \forall\sigma.\,\mathcal{K}_\sigma(Y) \multimap \mathcal{K}_{\varphi\times\sigma}(X)\end{aligned}$$

Continuations (computation types)

$$\mathcal{K}_\sigma(X) = \forall\varphi.\,\mathcal{A}_\varphi(X) \multimap \bot^{(\mathcal{C}_\varphi(X)\times\sigma)}$$

Code

Idea: Er


$\mathcal{A}_\varphi(\mathbb{N} \to \mathbb{N})$
$\mathsf{nat} \times (\varphi \times \sigma)$
$\mathsf{nat} \times \sigma$
$\mathcal{A}_\varphi((\mathbb{N} \to \mathbb{N}) \to \mathbb{N})$
$\mathsf{nat} \times \sigma$
$\psi \times (\varphi \times \sigma)$
$\mathsf{nat} \times (\psi \times \sigma)$
$\mathsf{nat} \times \sigma$

Code types (value types)

$$\mathcal{C}_\varphi(\mathbb{N}) = \mathsf{nat}$$
$$\mathcal{C}_\varphi(X \to Y) = \varphi$$

Access types (computation types)

$$\mathcal{A}_\varphi(\mathbb{N}) = \bot^0$$
$$\mathcal{A}_\varphi(X \to Y) = \forall \sigma. \mathcal{K}_\sigma(Y) \multimap \mathcal{K}_{\varphi \times \sigma}(X)$$

Continuations (computation types)

$$\mathcal{K}_\sigma(X) = \forall \varphi. \mathcal{A}_\varphi(X) \multimap \bot^{(\mathcal{C}_\varphi(X) \times \sigma)}$$

Refined Translation

$$x_1 \colon \mathcal{A}_{\varphi_1}(X_1), \ldots, x_n \colon \mathcal{A}_{\varphi_n}(X_n) \vdash \mathsf{cps}(M) \colon \mathcal{T}_{\mathcal{C}_{\varphi_1}(X_1) \times \cdots \times \mathcal{C}_{\varphi_n}(X_n)}(X)$$

$$\mathcal{T}_{\gamma}(X) = \forall \sigma . \mathcal{K}_{\sigma}(X) \multimap \bot^{(\gamma \times \sigma)}$$

Example: $\mathcal{T}_1(\mathbb{N} \to \mathbb{N})$

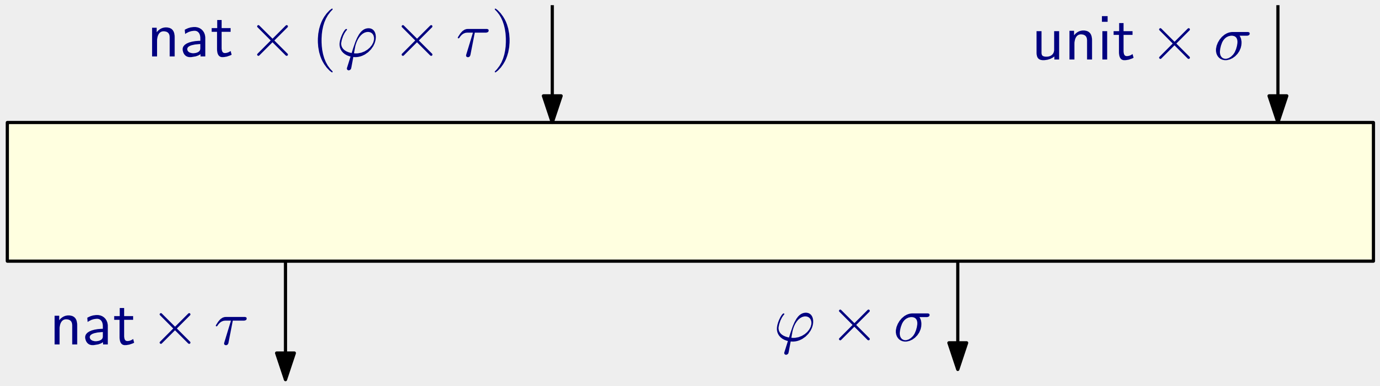
$$\mathcal{T}_1(\mathbb{N} \to \mathbb{N})$$

$$\cong$$

$$\forall \sigma.\, \Big(\forall \varphi.\, \mathcal{A}_\varphi(\mathbb{N} \to \mathbb{N}) \multimap \bot^{(\mathcal{C}_\varphi(\mathbb{N}\to\mathbb{N}) \times \sigma)}\Big) \multimap \bot^{(\mathsf{unit} \times \sigma)}$$

$$\cong$$

$$\forall \sigma.\, \Big(\forall \varphi.\, (\forall \tau.\, \bot^{(\mathsf{nat} \times \tau)} \multimap \bot^{(\mathsf{nat} \times (\varphi \times \tau))}) \multimap \bot^{(\varphi \times \sigma)}\Big) \multimap \bot^{(\mathsf{unit} \times \sigma)}$$


$\mathsf{nat} \times (\varphi \times \tau)$
$\mathsf{unit} \times \sigma$
$\mathsf{nat} \times \tau$
$\varphi \times \sigma$

$$\lambda x{:}\mathbb{N}.\, \mathsf{add}(x, y) \quad \Longrightarrow \quad \begin{array}{r} \texttt{eval\_term}(\langle\rangle, s)\{\, \texttt{ret\_funval}(\ulcorner\langle\rangle\urcorner, s)\ \} \\ \texttt{apply\_fun}(x, \langle c, t\rangle)\{\, \texttt{ret\_natval}(x + 5, t)\ \} \end{array}$$

Refined Call-by-Value Translation

If Γ declares the variables $\vec{z}$ and these all appear free in M, then define $\mathsf{cps}(\Gamma \vdash M)$ by:

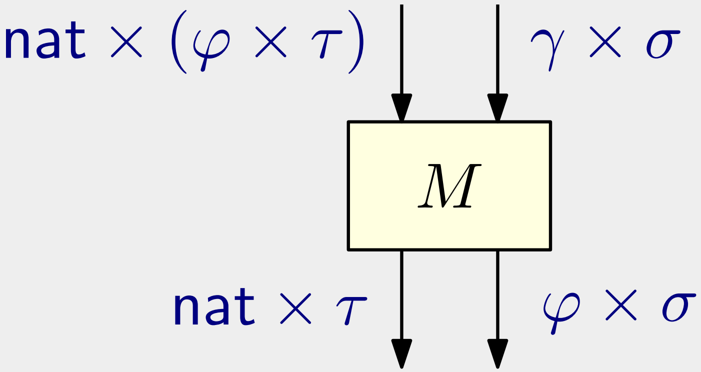
$$
\begin{aligned}
\mathsf{cps}(x\colon X \vdash x) &= \Lambda\sigma.\, \lambda k.\, (\langle\langle\langle\rangle, x\rangle, s\rangle \mapsto \langle x, s\rangle)^* (k\, \mathcal{C}(x{:}X)\; x) \\
\mathsf{cps}(\vdash n) &= \Lambda\sigma.\, \lambda k.\, (\langle\langle\rangle, s\rangle \mapsto \langle n, s\rangle)^* (k\; \mathsf{unit}\; \star) \\
\mathsf{cps}(\Gamma \vdash \lambda x{:}X.\, M) &= \Lambda\sigma.\, \lambda k.\, k\; \mathcal{C}(\Gamma)\; (\Lambda\tau.\, \lambda k_1.\, \Lambda\varphi_x.\, \lambda x.\; (\langle a, \langle\vec{z}, t\rangle\rangle \mapsto \langle\langle\vec{z}, a\rangle, t\rangle)^* \\
&\qquad\qquad (\mathsf{cps}(\Gamma, x\colon X \vdash M)\; \tau\; k_1)) \\
\mathsf{cps}(\Gamma \vdash M\; N) &= \Lambda\sigma.\, \lambda k.\, (\langle\vec{z}, s\rangle \mapsto \langle\vec{z}, \langle\vec{z}, s\rangle\rangle)^* \mathsf{cps}(\Gamma \vdash M)\; (\mathcal{C}(\Gamma) \times \sigma)\; t \\
&\quad \text{where } t = (\Lambda\varphi.\, \lambda f.\; (\langle\varphi, \langle\vec{z}, s\rangle\rangle \mapsto \langle\vec{z}, \langle\varphi, s\rangle\rangle)^* \\
&\qquad\qquad \mathsf{cps}(\Gamma \vdash N)\; (\varphi \times \sigma)\; (\Lambda\tau.\, \lambda x.\, f\; \sigma\; k\; \tau\; x)) \\
&\ldots
\end{aligned}
$$

If Γ declares more than the free variables of M, then define

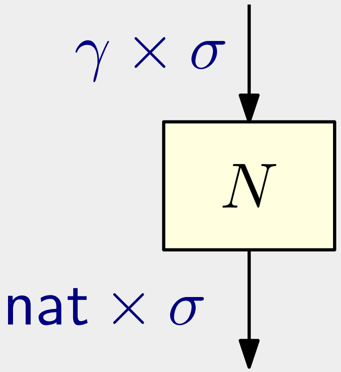
$$
\mathsf{cps}(\Gamma \vdash M) = \Lambda\sigma.\, \lambda k.\, (\langle\vec{z}, s\rangle \mapsto \langle\vec{y}, s\rangle)^* (\mathsf{cps}(\Delta \vdash M)\; \sigma\; k) \quad .
$$

Application of $\Gamma \vdash M : \mathbb{N} \to \mathbb{N}$ to $\Gamma \vdash N : \mathbb{N}$

$\forall \sigma. \exists \varphi. \forall \tau.$

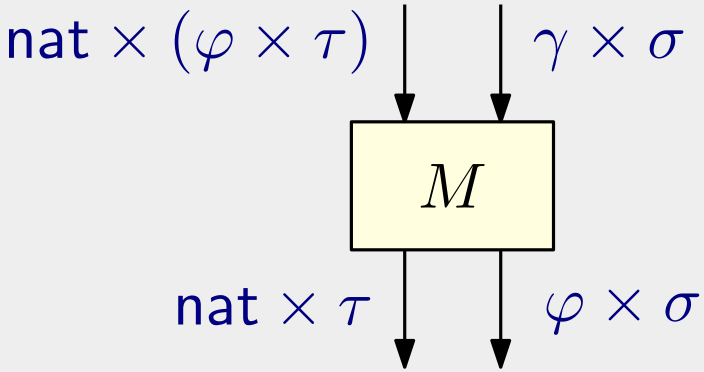


$\forall \sigma.$



Application of $\Gamma \vdash M : \mathbb{N} \to \mathbb{N}$ to $\Gamma \vdash N : \mathbb{N}$

$\forall \sigma . \exists \varphi . \forall \tau .$


nat $\times (\varphi \times \tau)$

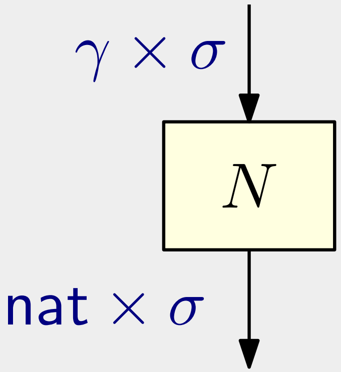
$\gamma \times \sigma$

M

nat $\times \tau$

$\varphi \times \sigma$

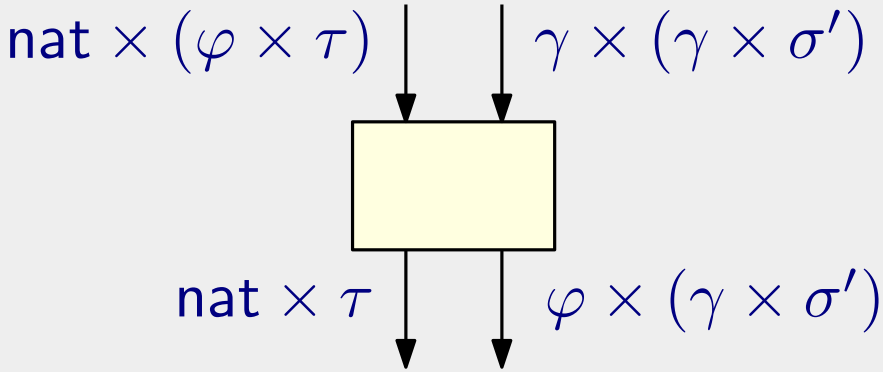
$\forall \sigma .$


$\gamma \times \sigma$

N

nat $\times \sigma$

$\forall \tau .$


nat $\times (\varphi \times \tau)$

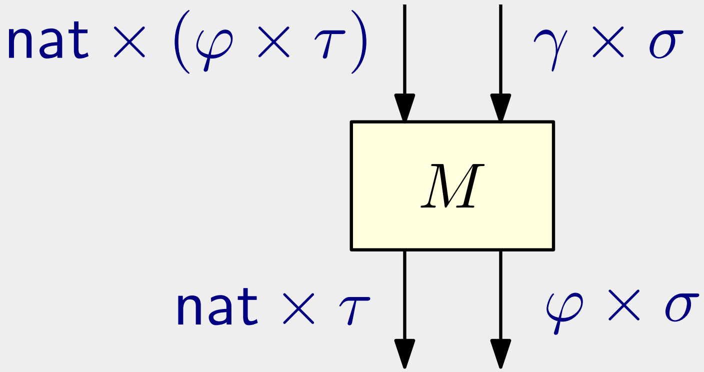
$\gamma \times (\gamma \times \sigma')$

nat $\times \tau$

$\varphi \times (\gamma \times \sigma')$

Application of $\Gamma \vdash M : \mathbb{N} \to \mathbb{N}$ to $\Gamma \vdash N : \mathbb{N}$

$\forall\sigma.\,\exists\varphi.\,\forall\tau.$



$\mathsf{nat} \times (\varphi \times \tau)$

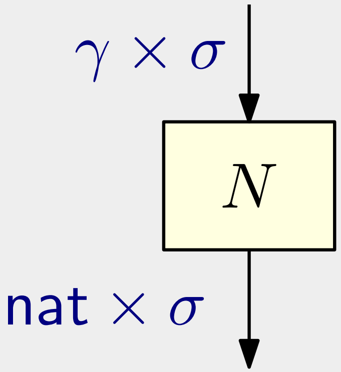
$\gamma \times \sigma$

M

$\mathsf{nat} \times \tau$

$\varphi \times \sigma$

$\forall\sigma.$

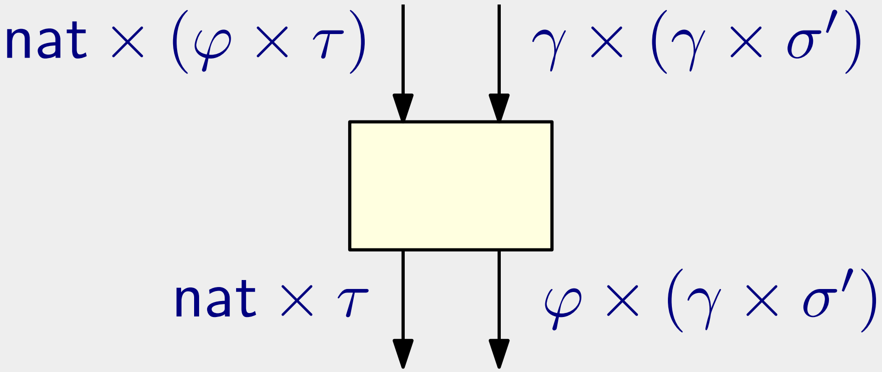


$\gamma \times \sigma$

N

$\mathsf{nat} \times \sigma$

$\forall\tau.$

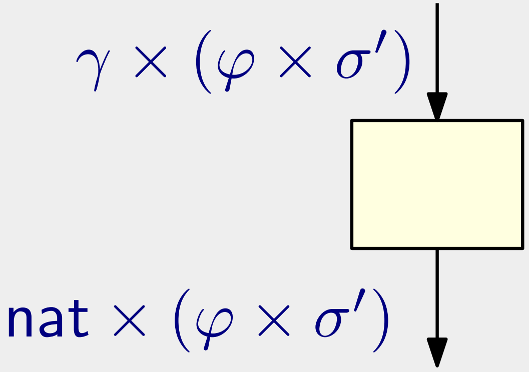


$\mathsf{nat} \times (\varphi \times \tau)$

$\gamma \times (\gamma \times \sigma')$

$\mathsf{nat} \times \tau$

$\varphi \times (\gamma \times \sigma')$

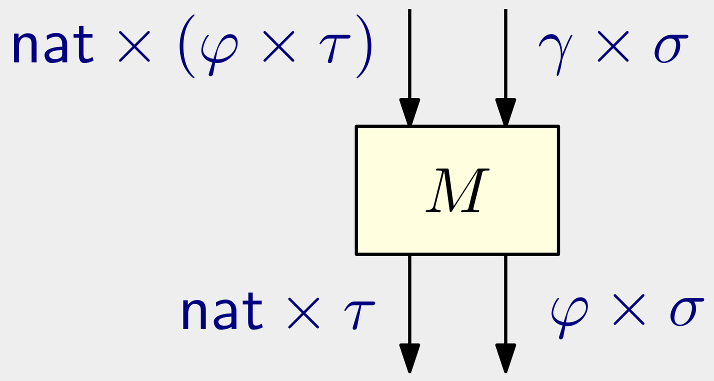


$\gamma \times (\varphi \times \sigma')$

$\mathsf{nat} \times (\varphi \times \sigma')$

Application of $\Gamma \vdash M : \mathbb{N} \to \mathbb{N}$ to $\Gamma \vdash N : \mathbb{N}$

$\forall \sigma . \exists \varphi . \forall \tau .$



$\mathsf{nat} \times (\varphi \times \tau)$

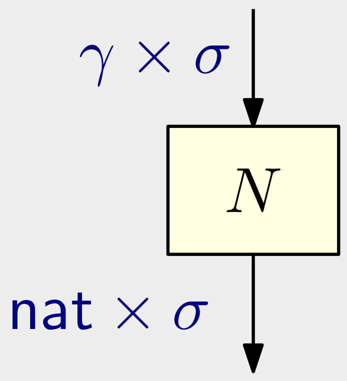
$\gamma \times \sigma$

M

$\mathsf{nat} \times \tau$

$\varphi \times \sigma$

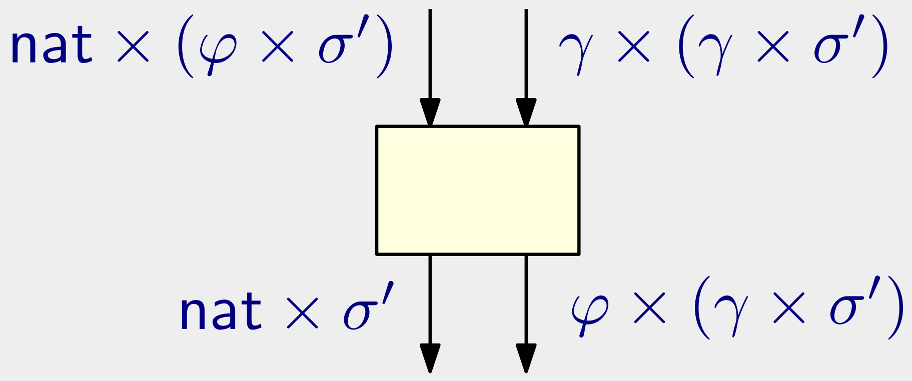
$\forall \sigma .$



$\gamma \times \sigma$

N

$\mathsf{nat} \times \sigma$

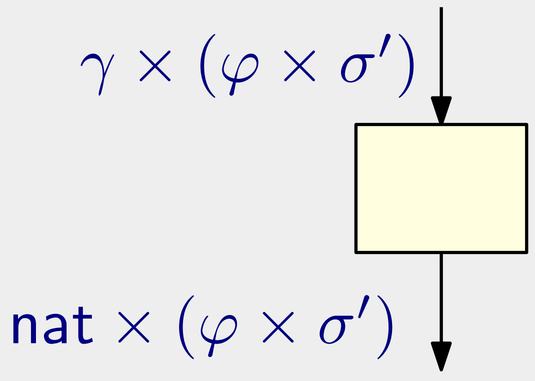


$\mathsf{nat} \times (\varphi \times \sigma')$

$\gamma \times (\gamma \times \sigma')$

$\mathsf{nat} \times \sigma'$

$\varphi \times (\gamma \times \sigma')$

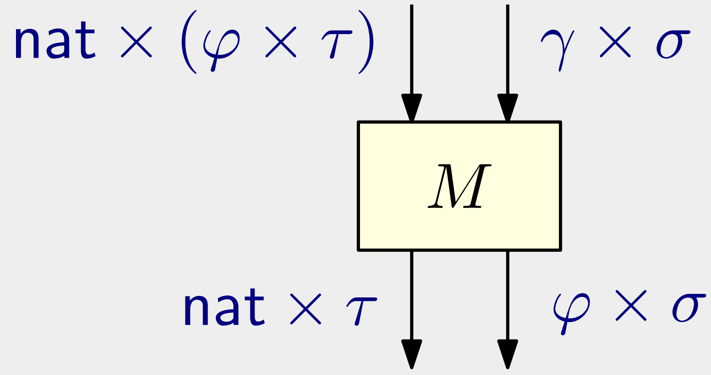


$\gamma \times (\varphi \times \sigma')$

$\mathsf{nat} \times (\varphi \times \sigma')$

Application of $\Gamma \vdash M : \mathbb{N} \to \mathbb{N}$ to $\Gamma \vdash N : \mathbb{N}$

$\forall \sigma. \exists \varphi. \forall \tau.$



$\mathsf{nat} \times (\varphi \times \tau)$

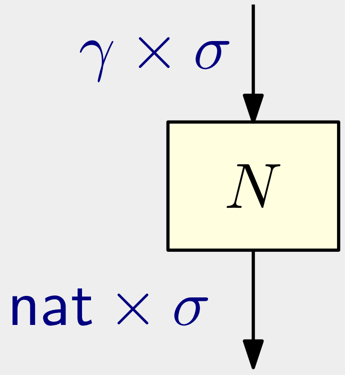
$\gamma \times \sigma$

M

$\mathsf{nat} \times \tau$

$\varphi \times \sigma$

$\forall \sigma.$



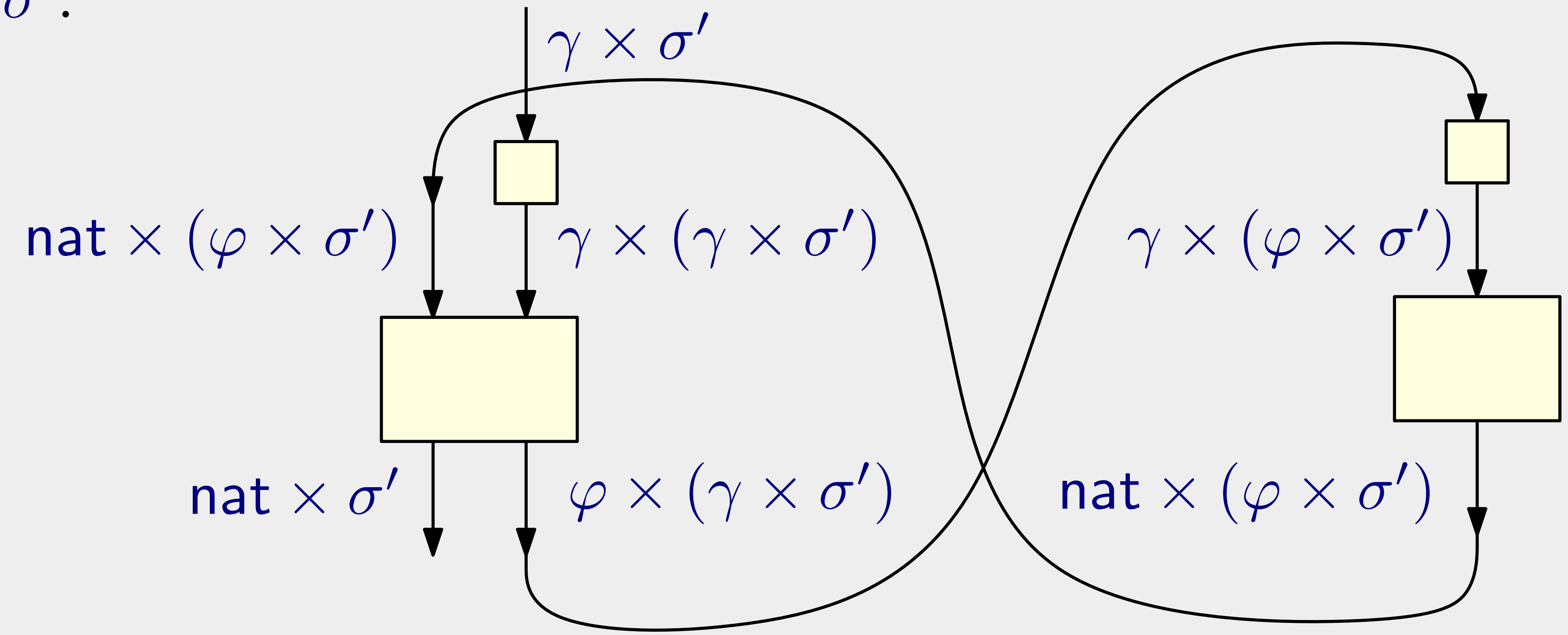
$\gamma \times \sigma$

N

$\mathsf{nat} \times \sigma$

$\forall \sigma'.$

$M\ N$



$\gamma \times \sigma'$

$\mathsf{nat} \times (\varphi \times \sigma')$

$\gamma \times (\gamma \times \sigma')$

$\gamma \times (\varphi \times \sigma')$

$\mathsf{nat} \times \sigma'$

$\varphi \times (\gamma \times \sigma')$

$\mathsf{nat} \times (\varphi \times \sigma')$

Application of $\Gamma \vdash M : \mathbb{N} \to \mathbb{N}$ to $\Gamma \vdash N : \mathbb{N}$

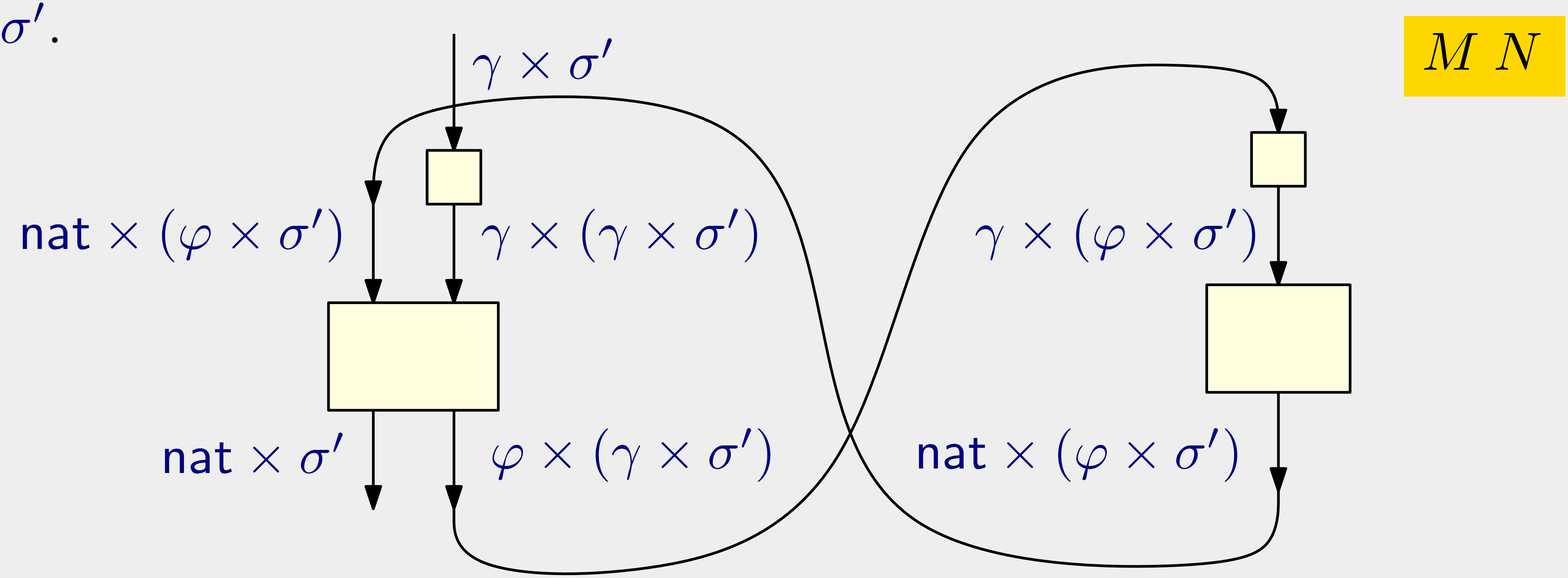
$\forall \sigma. \exists \varphi. \forall \tau.$ $\forall \sigma.$

nat $\times (\varphi \times \tau)$ $\gamma \times \sigma$

$$\mathsf{cps}(\Gamma \vdash M\ N) = \Lambda \sigma'.\, \lambda k.\, (\langle \vec{z}, s\rangle \mapsto \langle \vec{z}, \langle \vec{z}, s\rangle\rangle)^* \mathsf{cps}(\Gamma \vdash M)\ (\mathcal{C}(\Gamma) \times \sigma')\ t$$

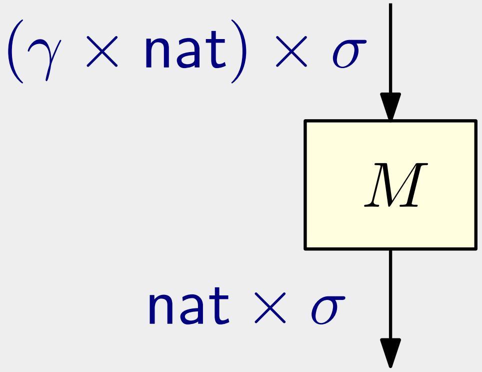
$$\text{where } t = (\Lambda \varphi.\, \lambda f.\, (\langle \varphi, \langle \vec{z}, s\rangle\rangle \mapsto \langle \vec{z}, \langle \varphi, s\rangle\rangle)^*$$
$$\mathsf{cps}(\Gamma \vdash N)\ (\varphi \times \sigma')\ (\Lambda \tau.\, \lambda x.\, f\ \sigma'\ k\ \tau\ x))$$

$\forall \sigma'.$


$M\ N$
$\gamma \times \sigma'$
nat $\times (\varphi \times \sigma')$
$\gamma \times (\gamma \times \sigma')$
$\gamma \times (\varphi \times \sigma')$
nat $\times \sigma'$
$\varphi \times (\gamma \times \sigma')$
nat $\times (\varphi \times \sigma')$

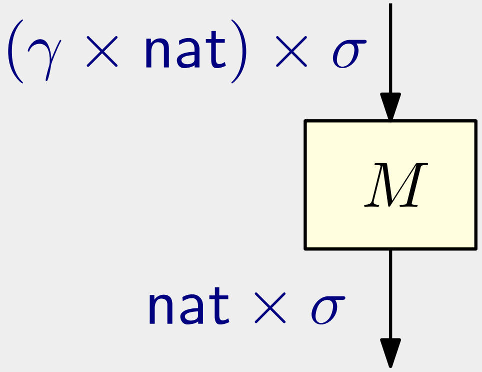
Abstraction of $\Gamma, x : \mathbb{N} \vdash M : \mathbb{N}$

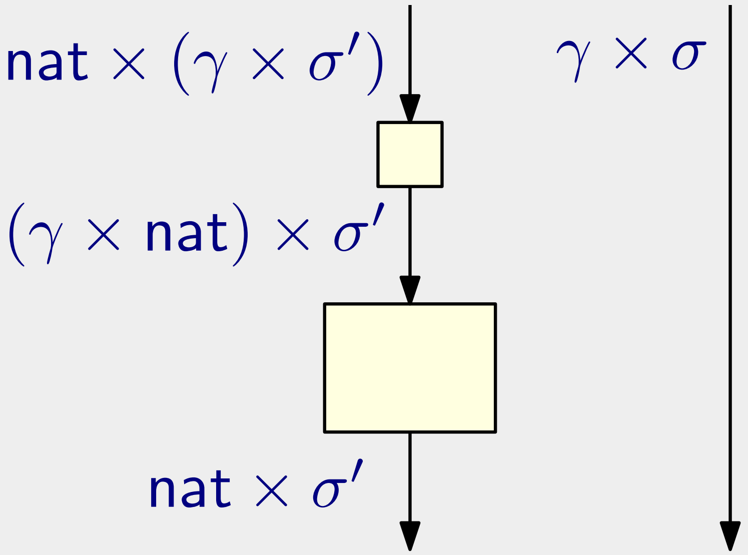
$\forall \sigma.$


$(\gamma \times \mathsf{nat}) \times \sigma$
M
$\mathsf{nat} \times \sigma$

Abstraction of $\Gamma, x : \mathbb{N} \vdash M : \mathbb{N}$

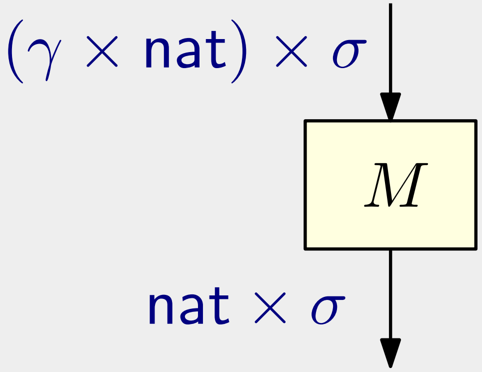
$\forall \sigma.$


$(\gamma \times \mathsf{nat}) \times \sigma$
M
$\mathsf{nat} \times \sigma$


$\mathsf{nat} \times (\gamma \times \sigma')$
$\gamma \times \sigma$
$(\gamma \times \mathsf{nat}) \times \sigma'$
$\mathsf{nat} \times \sigma'$

Abstraction of $\Gamma, x\colon \mathbb{N} \vdash M\colon \mathbb{N}$

$\forall \sigma.$



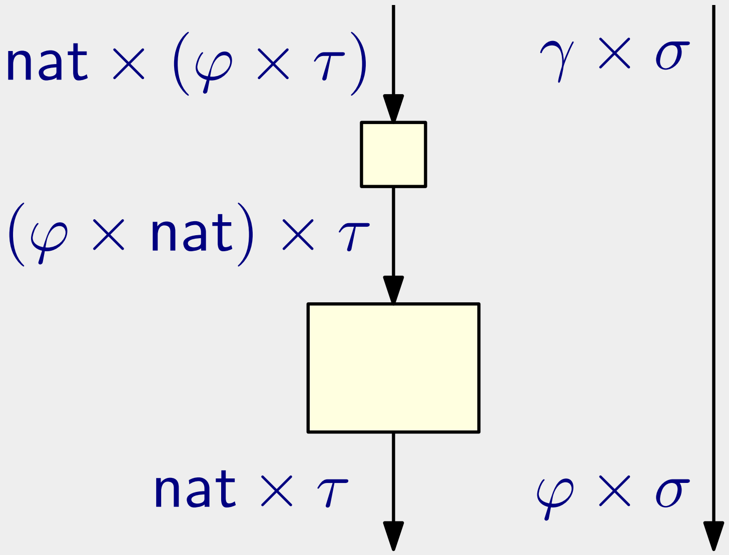
$(\gamma \times \mathsf{nat}) \times \sigma$

M

$\mathsf{nat} \times \sigma$

$\lambda x{:}X.\, M$

$\forall \sigma.\, \exists \varphi.\, \forall \tau.$



$\mathsf{nat} \times (\varphi \times \tau)$

$\gamma \times \sigma$

$(\varphi \times \mathsf{nat}) \times \tau$

$\mathsf{nat} \times \tau$

$\varphi \times \sigma$

Abstraction of $\Gamma, x\colon \mathbb{N} \vdash M : \mathbb{N}$

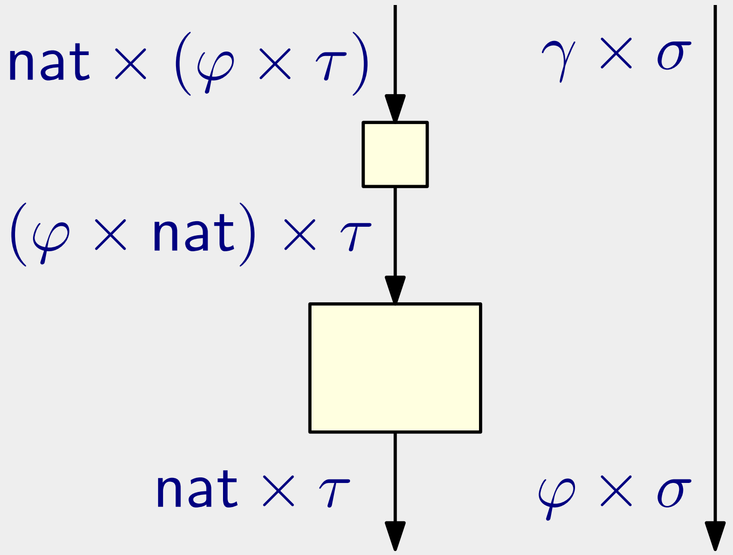
$\forall\sigma.$

$(\gamma \times \mathsf{nat}) \times \sigma$

$$\mathsf{cps}(\Gamma \vdash \lambda x{:}X.\, M) = \Lambda\sigma.\, \lambda k.\, k\, \mathcal{C}(\Gamma)\, (\Lambda\tau.\, \lambda k_1.\, \Lambda\varphi_x.\, \lambda x.\, (\langle a, \langle \vec{z}, t\rangle\rangle \mapsto \langle\langle \vec{z}, a\rangle, t\rangle)^*$$
$$(\mathsf{cps}(\Gamma, x\colon X \vdash M)\ \tau\ k_1))$$

$\lambda x{:}X.\, M$

$\forall\sigma.\, \exists\varphi.\, \forall\tau.$

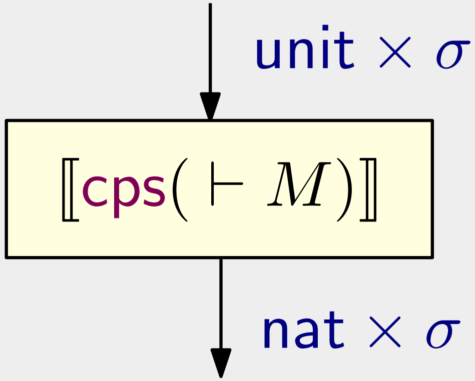

$\mathsf{nat} \times (\varphi \times \tau)$
$\gamma \times \sigma$
$(\varphi \times \mathsf{nat}) \times \tau$
$\mathsf{nat} \times \tau$
$\varphi \times \sigma$

Correctness

Theorem. Suppose $\vdash M : \mathbb{N}$ and $M \longrightarrow^{*}_{\mathsf{cbv}} n$, where n is a value. Then

$$\mathsf{cps}(\vdash M)\ \mathsf{unit}\ K = (\langle \vec{z}, s\rangle \mapsto \langle n, s\rangle)^{*}(K\ \mathsf{unit}\ \star)$$

for any closed continuation K of type $\mathcal{K}_{\mathsf{unit}}(\mathbb{N})$.



$\mathsf{unit} \times \sigma$

$[\![\mathsf{cps}(\vdash M)]\!]$

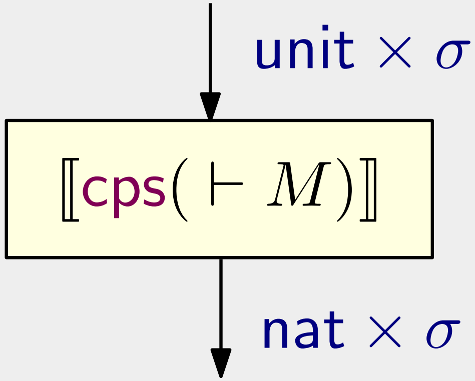
$\mathsf{nat} \times \sigma$

Correctness

Theorem. Suppose $\vdash M : \mathbb{N}$ and $M \longrightarrow^{*}_{\mathsf{cbv}} n$, where n is a value. Then

$$\mathsf{cps}(\vdash M)\ \mathsf{unit}\ K = (\langle \vec{z}, s\rangle \mapsto \langle n, s\rangle)^{*}(K\ \mathsf{unit}\ \star)$$

for any closed continuation K of type $\mathcal{K}_{\mathsf{unit}}(\mathbb{N})$.



$\mathsf{unit} \times \sigma$

$[\![\mathsf{cps}(\vdash M)]\!]$

$\mathsf{nat} \times \sigma$

Lemma. Let M be a source term well-typed in context Γ. Then, for all σ and all closed K such that $\mathsf{cps}(\Gamma \vdash M)\ \sigma\ K$ is well-typed, we have $\mathsf{cps}(\Gamma \vdash M)\ \sigma\ K = M :^{\Gamma}_{\sigma} K$.

Lemma. If $M \longrightarrow_{\mathsf{cbv}} N$ then $M :^{\Gamma}_{\sigma} K = N :^{\Gamma}_{\sigma} K$ for any σ and closed K of the appropriate type.

Contraction

To translate the full source calculus, we need contraction on variables of type $\mathcal{A}_{\varphi}(X)$.

Value Types $A, B \ ::= \ \alpha \mid \mathsf{nat} \mid \mathsf{unit} \mid A \times B \mid 0 \mid A + B$

Computation Types $X, Y \ ::= \ \bot^{A} \mid A \cdot X \multimap Y \mid \forall \alpha.\, X$

Conclusion

CPS translations for call-by-name and call-by-value can be refined to target a low-level computation calculus.

- fully specified translation to low-level language
- interface specification
- separate compilation
- exposes low-level details, e.g. closure conversion
- soundness proof managable
- value/computation-separation à la defunctionalization

Further work

- space bounds / optimisation using $\forall\alpha \lhd A.\, X$
- understand control flow data
- relation to call-by-value games
- fully abstract compilation