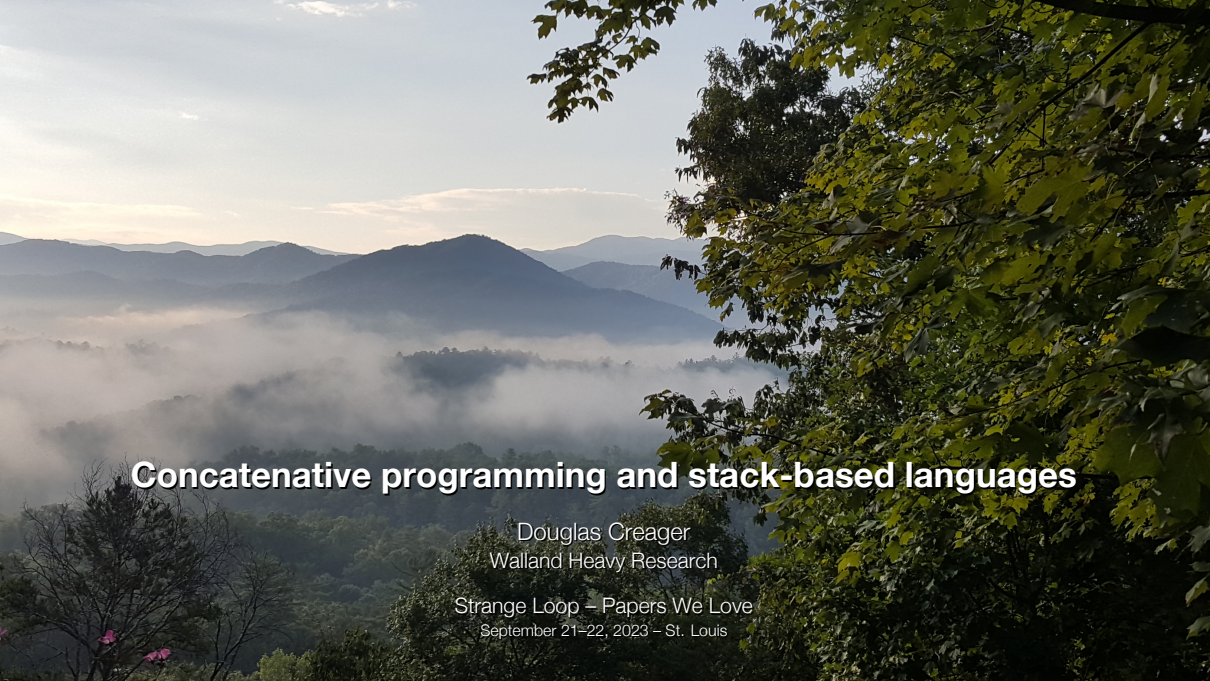
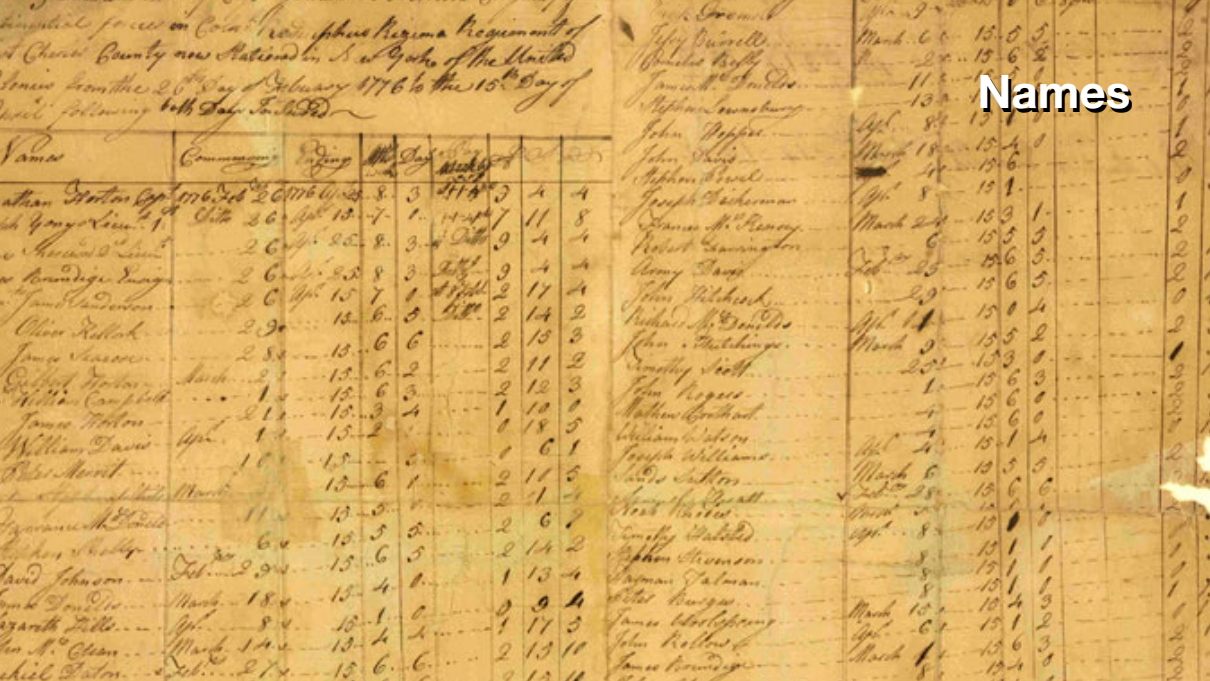


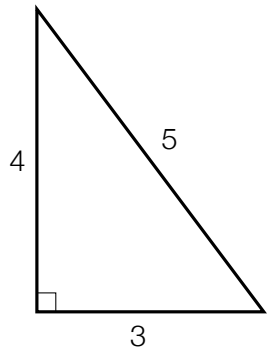
Concatenative programming and stack-based languages

Douglas Creager
Walland Heavy Research

Strange Loop – Papers We Love
September 21–22, 2023 – St. Louis

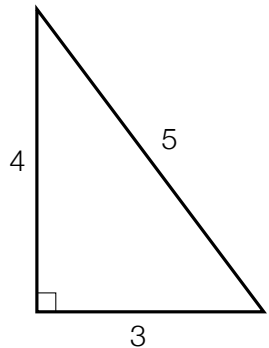

Names

Pythagoras




```
import math

leg1 = 3
leg2 = 4

l1sq = leg1 * leg1
l2sq = leg2 * leg2
hypotsq = l1sq + l2sq
result = math.sqrt(hypotsq)

print(result)
```

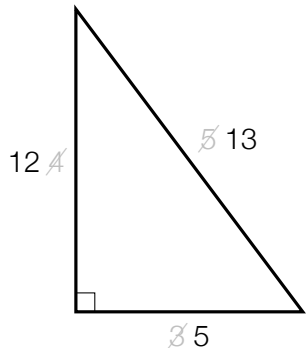
Pythagoras




```
import math

leg1 = 3
leg2 = 4

l1sq = leg1 * leg1
l2sq = leg2 * leg2
hypotsq = l1sq + l2sq
result = math.sqrt(hypotsq)

print(result)
```

```
5
```

Pythagoras

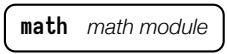



```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras

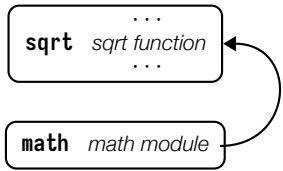
```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



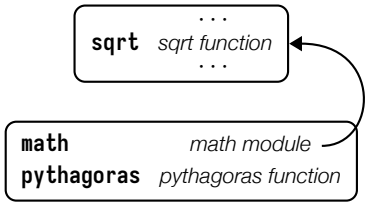

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



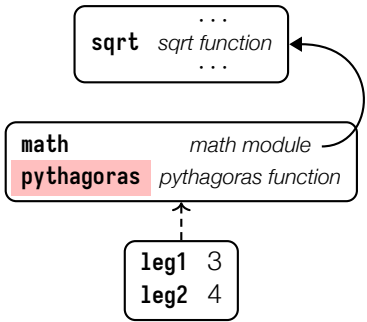

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



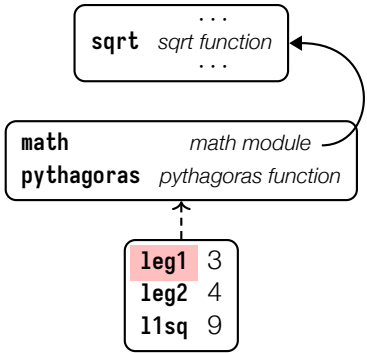

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



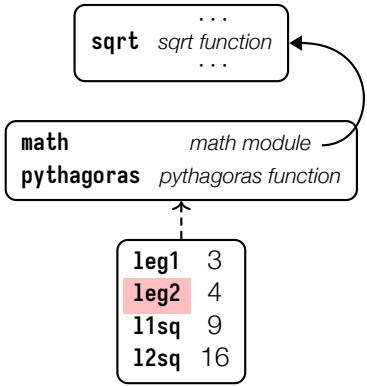

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



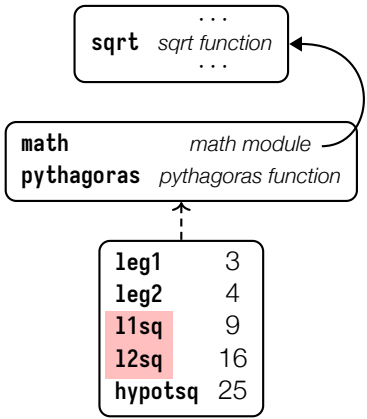

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



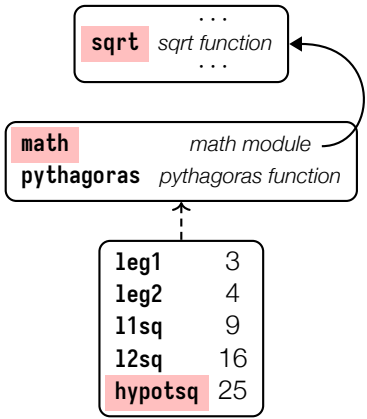

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras



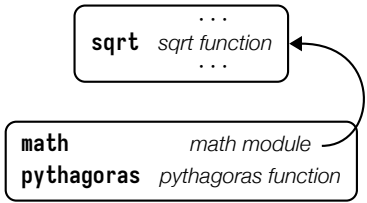

```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

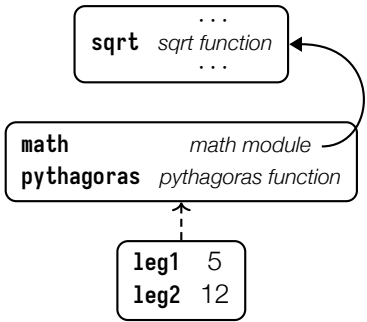
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
```

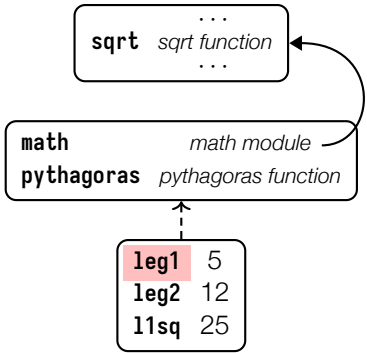
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
```

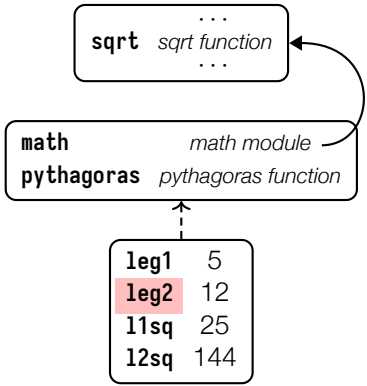
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
```

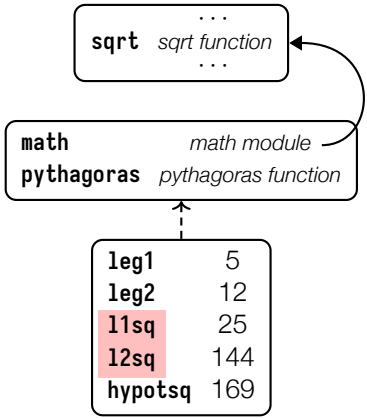
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
```

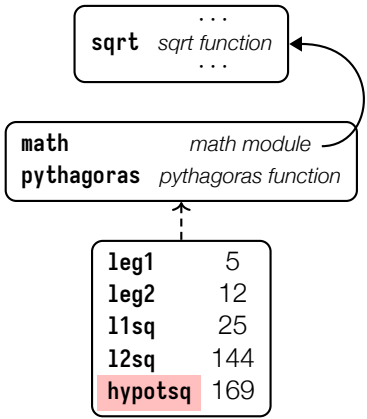
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
```

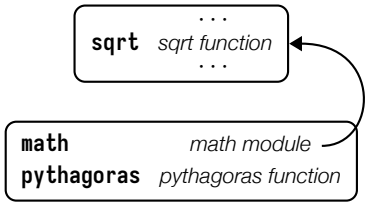
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
```

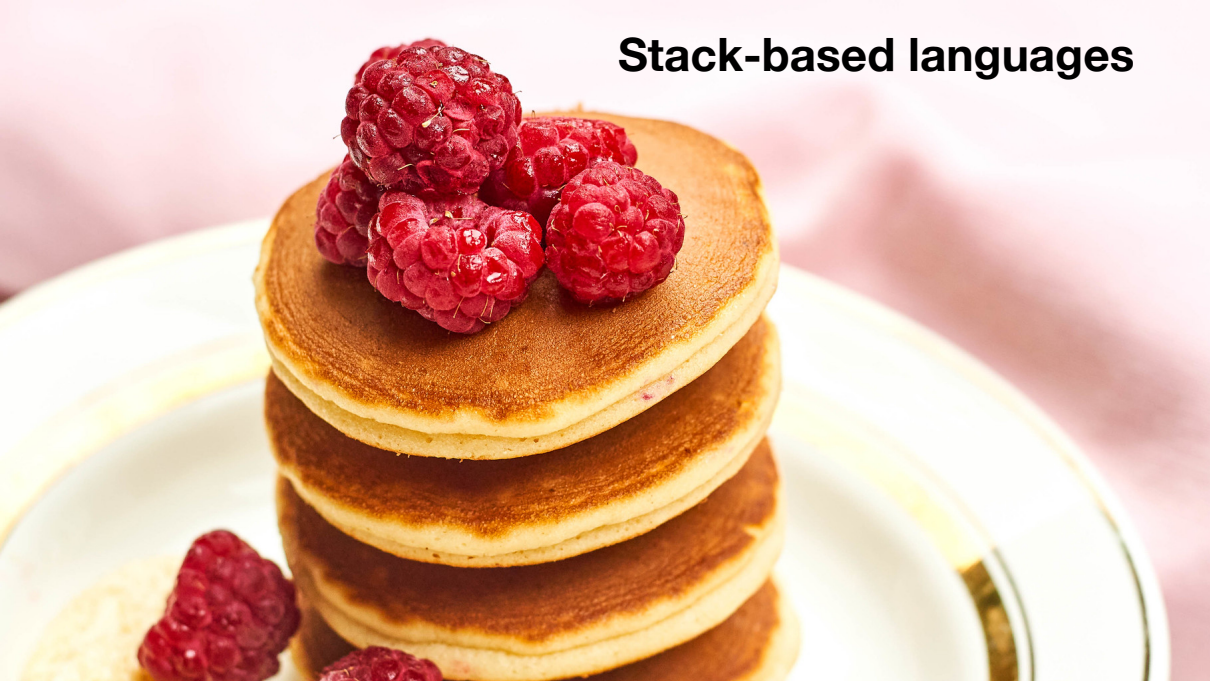
Pythagoras




```
import math

def pythogoras(leg1, leg2):
    l1sq = leg1 * leg1
    l2sq = leg2 * leg2
    hypotsq = l1sq + l2sq
    return math.sqrt(hypotsq)

print(pythagoras(3, 4))
print(pythagoras(5, 12))
```

```
5
13
```

Stack-based languages



Programs operate on a stack

Programs operate on a stack

1 2 +

Programs operate on a stack

2 +

Programs operate on a stack

Programs operate on a stack

Pythagoras

```
3 3 * 4 4 * + sqrt
```

Pythagoras


```
3 * 4 4 * + sqrt
```

Pythagoras


```
* 4 4 * + sqrt
```

Pythagoras


```
4 4 * + sqrt
```

Pythagoras


```
4 * + sqrt
```

Pythagoras

```
9 4 4
```

```
* + sqrt
```

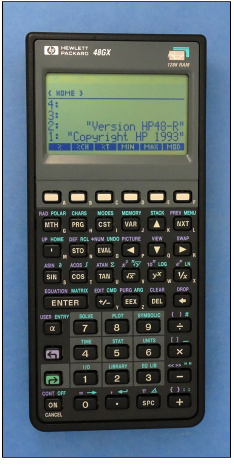
Pythagoras

Pythagoras

Pythagoras

Pythagoras

5



Manipulating the stack

```
3 3 * 4 4 * + sqrt
```

Manipulating the stack

```
3 4 dup * swap dup * + sqrt
```

Manipulating the stack


```
4 dup * swap dup * + sqrt
```

Manipulating the stack


```
dup * swap dup * + sqrt
```

Manipulating the stack


```
* swap dup * + sqrt
```

Manipulating the stack


```
swap dup * + sqrt
```

Manipulating the stack


```
dup * + sqrt
```

Manipulating the stack


```
* + sqrt
```

Manipulating the stack


```
+ sqrt
```

Manipulating the stack

Manipulating the stack

Abstraction

```
3 4 dup * swap dup * + sqrt
```

Abstraction

```
3 4 [dup * swap dup * + sqrt] apply
```

Abstraction


```
4 [dup * swap dup * + sqrt] apply
```

Abstraction

Abstraction

```
3 4 [dup * swap dup * + sqrt]    apply
```

Abstraction


```
dup * swap dup * + sqrt
```

Abstraction


```
* swap dup * + sqrt
```

Abstraction


```
swap dup * + sqrt
```

Abstraction

Abstraction


```
* + sqrt
```

Abstraction

Abstraction

Abstraction

Digging for dirt

```
3 4 [dup * swap dup * + sqrt] apply
```

Digging for dirt

```
[dup * swap dup * + sqrt] 3 4 ??? ap
```

Digging for dirt

```
[dup * swap dup * + sqrt]
```

```
3 4 ??? apply
```

Digging for dirt

```
[dup * swap dup * + sqrt] 3
```

```
4 ??? apply
```

Digging for dirt

```
[dup * swap dup * + sqrt] 3 4
```

```
??? apply
```

Digging for dirt

```
[dup * swap dup * + sqrt] 3 4    dig2 apply
```

Digging for dirt

```
3 4 [dup * swap dup * + sqrt]    apply
```

Digging for dirt


```
dup * swap dup * + sqrt
```

Digging for dirt


```
* swap dup * + sqrt
```

Digging for dirt


```
swap dup * + sqrt
```

Digging for dirt


```
dup * + sqrt
```

Digging for dirt


```
* + sqrt
```

Digging for dirt

Digging for dirt

Digging for dirt

Applying a quotation more than once

```
[dup * swap dup * + sqrt] dup 3 4 dig2 apply drop | 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] dup 3 4 dig2 apply drop | 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] dup 3 4 dig2 apply drop 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] dup 3 4 dig2 apply drop 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] dup 3 4 d
```

Applying a quotation more than once

Applying a quotation more than once

```
rt] [dup * swap dup * + sqrt]
```

```
3 4 dig2 apply drop 5 12 dig2 apply
```

Applying a quotation more than once

```
] [dup * swap dup * + sqrt] 3
```

```
4 dig2 apply drop 5 12 dig2 apply dı
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] 3 4
```

```
dig2 apply drop 5 12 dig2 apply drop
```

Applying a quotation more than once

```
3 4 [dup * swap dup * + sqrt]    apply drop 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] 3 4
```

```
dup * swap dup * + sqrt drop 5 12 d
```

Applying a quotation more than once

Applying a quotation more than once

Applying a quotation more than once

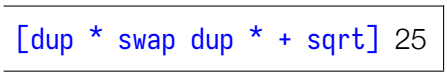
Applying a quotation more than once

```
] * swap dup * + sqrt] 16 3 3
```

```
* + sqrt drop 5 12 dig2 apply drop
```

Applying a quotation more than once

Applying a quotation more than once




```
sqrt drop 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] 5
```

```
drop 5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt]
```

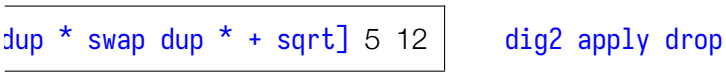
```
5 12 dig2 apply drop
```

Applying a quotation more than once

```
[dup * swap dup * + sqrt] 5
```

```
12 dig2 apply drop
```

Applying a quotation more than once



Applying a quotation more than once

```
12 [dup * swap dup * + sqrt]    apply drop
```

Applying a quotation more than once


```
dup * swap dup * + sqrt drop
```

Applying a quotation more than once


```
* swap dup * + sqrt drop
```

Applying a quotation more than once


```
swap dup * + sqrt drop
```

Applying a quotation more than once


```
dup * + sqrt drop
```

Applying a quotation more than once


```
* + sqrt drop
```

Applying a quotation more than once

Applying a quotation more than once

Applying a quotation more than once

Applying a quotation more than once

Meaning

What is the meaning of a stack program?

What is the meaning of a stack program?

1 2 +

What is the meaning of a stack program?

2 +

What is the meaning of a stack program?

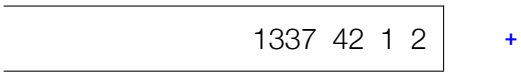
What is the meaning of a stack program?

The stack doesn’t have to start empty

1 2 +

The stack doesn’t have to start empty

The stack doesn’t have to start empty



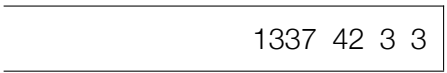
The stack doesn’t have to start empty

1 2 + $\Longrightarrow$ *Push* 3 *onto the stack*

Programs can consume existing values


```
dup *
```

Programs can consume existing values



*

Programs can consume existing values

`dup *` $\Longrightarrow$ *Pop one value from the stack, square it, and push the result onto the stack*

Stack effects

```
-- 3
1 2 +
```

Slava Pestov, Daniel Ehrenberg, Joe Groff. “Factor: A Dynamic Stack-based Programming Language”. ACM SIGPLAN Notices 45:12. December 2010.

Stack effects

```
x -- x*x
dup *
```

Slava Pestov, Daniel Ehrenberg, Joe Groff. “Factor: A Dynamic Stack-based Programming Language”. ACM SIGPLAN Notices 45:12. December 2010.

Stack effects

```
x -- result
dup *
```

Slava Pestov, Daniel Ehrenberg, Joe Groff. “Factor: A Dynamic Stack-based Programming Language”. ACM SIGPLAN Notices 45:12. December 2010.

More mathier

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$\mathit{program} \triangleq \mathit{instruction} \mid \mathit{program}\ \mathit{program}$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$
\begin{aligned}
\textit{instruction} &\triangleq \textit{literal} \mid \textit{builtin} \mid [\textit{program}] \\
\textit{program} &\triangleq \textit{instruction} \mid \textit{program}\ \textit{program}
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$
\begin{array}{rcl}
\textit{literal} & \ni & \texttt{0}, \texttt{1}, \texttt{42}, \ldots \\
\\
\textit{instruction} & \triangleq & \textit{literal} \mid \textit{builtin} \mid [\textit{program}] \\
\textit{program} & \triangleq & \textit{instruction} \mid \textit{program program}
\end{array}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$
\begin{array}{rcl}
\textit{literal} & \ni & \texttt{0}, \texttt{1}, \texttt{42}, \ldots \\
\textit{builtin} & \triangleq & \texttt{+} \mid \texttt{*} \mid \texttt{sqrt} \mid \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\textit{instruction} & \triangleq & \textit{literal} \mid \textit{builtin} \mid [\textit{program}] \\
\textit{program} & \triangleq & \textit{instruction} \mid \textit{program program}
\end{array}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$
\begin{array}{rcl}
\textit{literal} & \ni & \texttt{0}, \texttt{1}, \texttt{42}, \ldots \\
\textit{builtin} & \triangleq & \texttt{+} \mid \texttt{*} \mid \texttt{sqrt} \mid \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\textit{instruction} & \triangleq & \textit{literal} \mid \textit{builtin} \mid [\textit{program}] \\
\textit{program} & \triangleq & \textit{instruction} \mid \textit{program program}
\end{array}
$$

$$
[\![\textit{program}]\!] \in \hat{v} \to \hat{v}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$
\begin{array}{rcl}
\hat{v} & \triangleq & \epsilon \mid \hat{v} \cdot v \\
\\
\textit{literal} & \ni & \texttt{0}, \texttt{1}, \texttt{42}, \ldots \\
\textit{builtin} & \triangleq & \texttt{+} \mid \texttt{*} \mid \texttt{sqrt} \mid \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\textit{instruction} & \triangleq & \textit{literal} \mid \textit{builtin} \mid [\textit{program}] \\
\textit{program} & \triangleq & \textit{instruction} \mid \textit{program program} \\
\\
[\![\textit{program}]\!] & \in & \hat{v} \to \hat{v}
\end{array}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

More mathier

$$
\begin{array}{rcl}
i & \in & \mathbb{Z} \\
v & \triangleq & i \mid \texttt{[}\mathit{program}\texttt{]} \\
\hat{v} & \triangleq & \epsilon \mid \hat{v} \cdot v
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{literal} & \ni & \texttt{0}, \texttt{1}, \texttt{42}, \ldots \\
\mathit{builtin} & \triangleq & \texttt{+} \mid \texttt{*} \mid \texttt{sqrt} \mid \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\mathit{instruction} & \triangleq & \mathit{literal} \mid \mathit{builtin} \mid \texttt{[}\mathit{program}\texttt{]} \\
\mathit{program} & \triangleq & \mathit{instruction} \mid \mathit{program}\ \mathit{program}
\end{array}
$$

$$
[\![\mathit{program}]\!] \in \hat{v} \to \hat{v}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{+}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 + i_2) \\
[\![\texttt{*}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 \times i_2) \\
[\![\texttt{sqrt}]\!]\ \hat{v} \cdot i &\triangleq \hat{v} \cdot \sqrt{i}
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
[\![[p]]\!]\ \hat{v} &\triangleq \hat{v} \cdot [p] \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{+}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 + i_2) \\
[\![\texttt{*}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 \times i_2) \\
[\![\texttt{sqrt}]\!]\ \hat{v} \cdot i &\triangleq \hat{v} \cdot \sqrt{i}
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
[\![[p]]\!]\ \hat{v} &\triangleq \hat{v} \cdot [p] \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{+}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 + i_2) \\
[\![\texttt{*}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 \times i_2) \\
[\![\texttt{sqrt}]\!]\ \hat{v} \cdot i &\triangleq \hat{v} \cdot \sqrt{i}
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{+}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 + i_2) \\
[\![\texttt{*}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 \times i_2) \\
[\![\texttt{sqrt}]\!]\ \hat{v} \cdot i &\triangleq \hat{v} \cdot \sqrt{i}
\end{aligned}
$$

$$
\begin{aligned}
[\![[p]]\!]\ \hat{v} &\triangleq \hat{v} \cdot [p] \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

$$
[\![p_1\ p_2]\!]\ \hat{v} \triangleq \textbf{?}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{+}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 + i_2) \\
[\![\texttt{*}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 \times i_2) \\
[\![\texttt{sqrt}]\!]\ \hat{v} \cdot i &\triangleq \hat{v} \cdot \sqrt{i}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{[}p\texttt{]}]\!]\ \hat{v} &\triangleq \hat{v} \cdot \texttt{[}p\texttt{]} \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot \texttt{[}p\texttt{]} &\triangleq [\![p]\!]\ \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

$$
[\![p_1\ p_2]\!]\ \hat{v} \triangleq \qquad [\![p_1]\!]\ \hat{v}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating stack programs

$$
\begin{aligned}
[\![\texttt{0}]\!]\ \hat{v} &= \hat{v} \cdot 0 \\
[\![\texttt{1}]\!]\ \hat{v} &= \hat{v} \cdot 1 \\
[\![\texttt{42}]\!]\ \hat{v} &= \hat{v} \cdot 42 \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{+}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 + i_2) \\
[\![\texttt{*}]\!]\ \hat{v} \cdot i_1 \cdot i_2 &\triangleq \hat{v} \cdot (i_1 \times i_2) \\
[\![\texttt{sqrt}]\!]\ \hat{v} \cdot i &\triangleq \hat{v} \cdot \sqrt{i}
\end{aligned}
$$

$$
\begin{aligned}
[\![[p]]\!]\ \hat{v} &\triangleq \hat{v} \cdot [p] \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

$$
[\![p_1\ p_2]\!]\ \hat{v} \triangleq [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v})
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![1\ \ 2\ \ +]\!]\,\hat{v} = \ ?$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![1\ \ 2\ \ +]\!]\,\hat{v} = \ ?$$

Push 3 *onto the stack*

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![1\ \ 2\ \ +]\!]\ \hat{v} \stackrel{?}{=} \hat{v} \cdot 3$$

Push 3 *onto the stack*

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![1\ \ 2\ \ +]\!]\ \hat{v} \stackrel{?}{=} \hat{v} \cdot 3$$

$$\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ \ +]\!]\ \hat{v} &\stackrel{?}{=} \hat{v} \cdot 3 \\
&= [\![2\ \ +]\!]\ ([\![1]\!]\ \hat{v}) \qquad p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ \ +]\!]\ \hat{v} &\stackrel{?}{=} \hat{v} \cdot 3 \\
&= [\![2\ \ +]\!]\ ([\![1]\!]\ \hat{v}) && p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v} \\
&= [\![2\ \ +]\!]\ (\hat{v} \cdot 1) && \hat{v}_1{=}\hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ +]\!]\ \hat{v} &\stackrel{?}{=} \hat{v} \cdot 3 \\
&= [\![2\ +]\!]\ ([\![1]\!]\ \hat{v}) && p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v} \\
&= [\![2\ +]\!]\ (\hat{v} \cdot 1) && \hat{v}_1{=}\hat{v} \\
&= [\![+]\!]\ ([\![2]\!]\ (\hat{v} \cdot 1)) && p_1{=}2,\ p_2{=}+,\ \hat{v}_4{=}\hat{v}{\cdot}1
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ \ +]\!]\ \hat{v} &\stackrel{?}{=} \hat{v} \cdot 3 \\
&= [\![2\ \ +]\!]\ ([\![1]\!]\ \hat{v}) && p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v} \\
&= [\![2\ \ +]\!]\ (\hat{v} \cdot 1) && \hat{v}_1{=}\hat{v} \\
&= [\![+]\!]\ ([\![2]\!]\ (\hat{v} \cdot 1)) && p_1{=}2,\ p_2{=}+,\ \hat{v}_4{=}\hat{v}{\cdot}1 \\
&= [\![+]\!]\ (\hat{v} \cdot 1 \cdot 2) && \hat{v}_2{=}\hat{v}{\cdot}1
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ \ +]\!]\ \hat{v} &\stackrel{?}{=} \hat{v} \cdot 3 \\
&= [\![2\ \ +]\!]\ ([\![1]\!]\ \hat{v}) && p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v} \\
&= [\![2\ \ +]\!]\ (\hat{v} \cdot 1) && \hat{v}_1{=}\hat{v} \\
&= [\![+]\!]\ ([\![2]\!]\ (\hat{v} \cdot 1)) && p_1{=}2,\ p_2{=}+,\ \hat{v}_4{=}\hat{v}{\cdot}1 \\
&= [\![+]\!]\ (\hat{v} \cdot 1 \cdot 2) && \hat{v}_2{=}\hat{v}{\cdot}1 \\
&= \hat{v} \cdot (1 + 2) && \hat{v}_3{=}\hat{v},\ i_1{=}1,\ i_2{=}2
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ \ +]\!]\ \hat{v} &\stackrel{?}{=} \hat{v} \cdot 3 \\
&= [\![2\ \ +]\!]\ ([\![1]\!]\ \hat{v}) && p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v} \\
&= [\![2\ \ +]\!]\ (\hat{v} \cdot 1) && \hat{v}_1{=}\hat{v} \\
&= [\![+]\!]\ ([\![2]\!]\ (\hat{v} \cdot 1)) && p_1{=}2,\ p_2{=}+,\ \hat{v}_4{=}\hat{v}{\cdot}1 \\
&= [\![+]\!]\ (\hat{v} \cdot 1 \cdot 2) && \hat{v}_2{=}\hat{v}{\cdot}1 \\
&= \hat{v} \cdot (1 + 2) && \hat{v}_3{=}\hat{v},\ i_1{=}1,\ i_2{=}2 \\
&= \hat{v} \cdot 3
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![1\ \ 2\ \ +]\!]\ \hat{v} &\overset{\checkmark}{=} \hat{v} \cdot 3 \\
&= [\![2\ \ +]\!]\ ([\![1]\!]\ \hat{v}) && p_1{=}1,\ p_2{=}2\ +,\ \hat{v}_4{=}\hat{v} \\
&= [\![2\ \ +]\!]\ (\hat{v} \cdot 1) && \hat{v}_1{=}\hat{v} \\
&= [\![+]\!]\ ([\![2]\!]\ (\hat{v} \cdot 1)) && p_1{=}2,\ p_2{=}+,\ \hat{v}_4{=}\hat{v}{\cdot}1 \\
&= [\![+]\!]\ (\hat{v} \cdot 1 \cdot 2) && \hat{v}_2{=}\hat{v}{\cdot}1 \\
&= \hat{v} \cdot (1 + 2) && \hat{v}_3{=}\hat{v},\ i_1{=}1,\ i_2{=}2 \\
&= \hat{v} \cdot 3
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![\texttt{dup *}]\!]\ \hat{v} = \ ?$$

$$\begin{aligned}
[\![\texttt{1}]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![\texttt{2}]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![\texttt{+}]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![\texttt{dup *}]\!]\ \hat{v} = \ ?$$

Pop one value from the stack, square it, and push the result onto the stack

$$
\begin{aligned}
[\![\texttt{1}]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![\texttt{2}]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![\texttt{+}]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![\texttt{dup *}]\!]\ \hat{v} \cdot i \stackrel{?}{=} \hat{v} \cdot (i \times i)$$

Pop one value from the stack, square it, and push the result onto the stack

$$\begin{aligned}
[\![\texttt{1}]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![\texttt{2}]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![\texttt{+}]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4)
\end{aligned}$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$[\![\text{dup } *]\!]\ \hat{v} \cdot i \stackrel{?}{=} \hat{v} \cdot (i \times i)$$

$$\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4) \\
[\![*]\!]\ \hat{v}_5 \cdot i_1 \cdot i_2 &= \hat{v}_5 \cdot (i_1 \times i_2) \\
[\![\text{dup}]\!]\ \hat{v}_6 \cdot v &= \hat{v}_6 \cdot v \cdot v
\end{aligned}$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![\text{dup } *]\!]\ \hat{v} \cdot i &\stackrel{?}{=} \hat{v} \cdot (i \times i) \\
&= [\![*]\!]\ ([\![\text{dup}]\!]\ \hat{v} \cdot i) \qquad p_1{=}\text{dup},\ p_2{=}*,\ \hat{v}_4{=}\hat{v}{\cdot}i
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4) \\
[\![*]\!]\ \hat{v}_5 \cdot i_1 \cdot i_2 &= \hat{v}_5 \cdot (i_1 \times i_2) \\
[\![\text{dup}]\!]\ \hat{v}_6 \cdot v &= \hat{v}_6 \cdot v \cdot v
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![\text{dup } *]\!]\ \hat{v} \cdot i &\stackrel{?}{=} \hat{v} \cdot (i \times i) \\
&= [\![*]\!]\ ([\![\text{dup}]\!]\ \hat{v} \cdot i) && p_1{=}\text{dup},\ p_2{=}*,\ \hat{v}_4{=}\hat{v}{\cdot}i \\
&= [\![*]\!]\ (\hat{v} \cdot i \cdot i) && \hat{v}_6{=}\hat{v},\ v{=}i
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4) \\
[\![*]\!]\ \hat{v}_5 \cdot i_1 \cdot i_2 &= \hat{v}_5 \cdot (i_1 \times i_2) \\
[\![\text{dup}]\!]\ \hat{v}_6 \cdot v &= \hat{v}_6 \cdot v \cdot v
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![\text{dup } *]\!]\ \hat{v} \cdot i &\stackrel{?}{=} \hat{v} \cdot (i \times i) \\
&= [\![*]\!]\ ([\![\text{dup}]\!]\ \hat{v} \cdot i) && p_1{=}\text{dup},\ p_2{=}*,\ \hat{v}_4{=}\hat{v}{\cdot}i \\
&= [\![*]\!]\ (\hat{v} \cdot i \cdot i) && \hat{v}_6{=}\hat{v},\ v{=}i \\
&= \hat{v} \cdot (i \times i) && \hat{v}_5{=}\hat{v},\ i_1{=}i,\ i_2{=}i
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4) \\
[\![*]\!]\ \hat{v}_5 \cdot i_1 \cdot i_2 &= \hat{v}_5 \cdot (i_1 \times i_2) \\
[\![\text{dup}]\!]\ \hat{v}_6 \cdot v &= \hat{v}_6 \cdot v \cdot v
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Evaluating an example program

$$
\begin{aligned}
[\![\text{dup } *]\!]\ \hat{v} \cdot i &\overset{\checkmark}{=} \hat{v} \cdot (i \times i) \\
&= [\![*]\!]\ ([\![\text{dup}]\!]\ \hat{v} \cdot i) && p_1{=}\text{dup},\ p_2{=}*,\ \hat{v}_4{=}\hat{v}{\cdot}i \\
&= [\![*]\!]\ (\hat{v} \cdot i \cdot i) && \hat{v}_6{=}\hat{v},\ v{=}i \\
&= \hat{v} \cdot (i \times i) && \hat{v}_5{=}\hat{v},\ i_1{=}i,\ i_2{=}i
\end{aligned}
$$

$$
\begin{aligned}
[\![1]\!]\ \hat{v}_1 &= \hat{v}_1 \cdot 1 \\
[\![2]\!]\ \hat{v}_2 &= \hat{v}_2 \cdot 2 \\
[\![+]\!]\ \hat{v}_3 \cdot i_1 \cdot i_2 &= \hat{v}_3 \cdot (i_1 + i_2) \\
[\![p_1\ p_2]\!]\ \hat{v}_4 &= [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v}_4) \\
[\![*]\!]\ \hat{v}_5 \cdot i_1 \cdot i_2 &= \hat{v}_5 \cdot (i_1 \times i_2) \\
[\![\text{dup}]\!]\ \hat{v}_6 \cdot v &= \hat{v}_6 \cdot v \cdot v
\end{aligned}
$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html


Completeness

The lambda calculus is Turing-complete

$$\lambda x \,.\, \lambda y \,.\, y\ x$$

The lambda calculus is Turing-complete

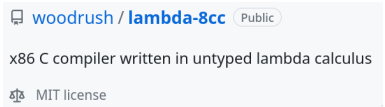
$$\lambda x\,.\,\lambda y\,.\,y\;x$$

$$\begin{aligned} 0 &\triangleq \lambda f\,.\,\lambda x\,.\,x \\ 1 &\triangleq \lambda f\,.\,\lambda x\,.\,f\;x \\ 2 &\triangleq \lambda f\,.\,\lambda x\,.\,f\;(f\;x) \\ &\vdots \end{aligned}$$

$$\mathrm{succ} \triangleq \lambda n\,.(\lambda f\,.\,\lambda x\,.\,f\;(n\;f\;x))$$

Yes, really

`woodrush/lambda-8cc`



> *It has long been known in computer science that lambda calculus is Turing-complete. lambda-8cc demonstrates this in a rather straightforward way by showing that C programs can directly be compiled into lambda calculus terms.*

Yes, really

```c
// rot13.c: Encodes/decodes standard input to/from the ROT13 cipher

#define EOF -1

int putchar(int c);
char getchar(void);

char c;
int offset;

int main (void) {
    for (;;) {
        c = getchar();
        if (c == EOF) {
            break;
        }

        offset = 0;
        if (('a' ≤ c && c < 'n') || ('A' ≤ c && c < 'N')) {
            offset = 13;
        } else if (('n' ≤ c && c ≤ 'z') || ('N' ≤ c && c ≤ 'Z')) {
            offset = -13;
        }
        putchar(c + offset);
    }
    return 0;
}
```

Yes, really

```
((\x.\y.\z.\a.\b.((\c.((\d.((\e.((\f.((\g.((\h.(a ((\i.(i (d ( \j.\k.(k (\l.\m.\n.\o.(o k (j m))) k)) a) (\j.(i z (d (\k.\l.\m.\n.\o.\p.((\q.((\r.((\s.(n (\t
(\f.(f((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x)x))(\x.\y.(y(\x.\a.a)x)))
(\f.(f((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x)x))(\x.\y.(y(\x.\a.a)x)))
(\f.(f((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(y(y(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x)x))(\x.\y.(y(\x.\a.a)x)))(\x.\y.y)))))))
(\f.(f
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.a)(\x.\y.x)((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(y(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
(\f.(f
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.b)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.d)(\x.\y.x)((\x.\y.(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.c)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.b)(\
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.b)(\
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.b)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.b)(\
(\f.(f
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.x)(\x.\y.y)(\x.(x(\y.\z.y)(\y.\z.z)))(\x.\y.\z.x)))
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.y)(\x.\y.x)((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
(\f.(f
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.x)((\x.\y.(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
(\f.(f
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.x)((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.c)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.y)(\y.\z.z)))))
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.x)((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.b)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.y)(\y.\z.z)))))
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.b)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.d)(\x.\y.x)((\x.\y.(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.c)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.y)(\y.\z.z)))))
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.b)(\
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.x)((\x.\y.(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(x(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.e)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.z)(\x.(x(\a.\b.b)(\a.\b.b)))))))(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.d)(\x.\y.x)((\x.\y.(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(y(\x.\y.y))))))))))))))))))))))))))(\x.\y.(y(\x.\a.x
  (\f.(f(\f.(f(\x.\y.\z.\a.\b.\c.\d.\e.c)(\x.\y.y)(\x.(x(\y.\z.z)(\x.(x(\z.\a.a)(\x.(x(\a.\b.a)(\a.\b.b)))))))(\x.(x(\y.\z.y)(\y.\z.z)))))
```

Those pesky names

$$\lambda x \,.\, \lambda y \,.\, y\ x$$

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\ z\ (y\ z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\ x$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = \underline{\lambda x\,.\,\lambda y\,.\,y\,x}$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = \underline{\lambda x\,.\,\underline{\lambda y\,.\,y\,x}}$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = \underline{\lambda x\,.(\mathbf{S}\ \underline{\lambda y\,.\,y}\ \underline{\lambda y\,.\,x})}$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = \underline{\lambda x\,.(\mathbf{S}\,\mathbf{I}\,\underline{\lambda y\,.\,x})}$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\;z\;(y\;z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\;x = \underline{\lambda x\,.(\mathbf{S}\;\mathbf{I}\;(\mathbf{K}\;\underline{x}))}$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \lambda x\,.\,x$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = \underline{\lambda x\,.(\mathbf{S}\,\mathbf{I}\,(\mathbf{K}\,x))}$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\begin{aligned}\mathbf{S} &\triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)\\ \mathbf{K} &\triangleq \lambda x\,.\,\lambda y\,.\,x\\ \mathbf{I} &\triangleq \lambda x\,.\,x\end{aligned}$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = (\mathbf{S}\ \underline{\lambda x\,.(\mathbf{S}\ \mathbf{I})}\ \underline{\lambda x\,.(\mathbf{K}\ x)})$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\begin{aligned}\mathbf{S} &\triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\;z\;(y\;z)\\ \mathbf{K} &\triangleq \lambda x\,.\,\lambda y\,.\,x\\ \mathbf{I} &\triangleq \lambda x\,.\,x\end{aligned}$$

$$\lambda x\,.\,\lambda y\,.\,y\;x = (\mathbf{S}\;(\mathbf{K}\;(\mathbf{S}\;\mathbf{I}))\;\underline{\lambda x\,.(\mathbf{K}\;x)})$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\begin{aligned}\mathbf{S} &\triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)\\ \mathbf{K} &\triangleq \lambda x\,.\,\lambda y\,.\,x\\ \mathbf{I} &\triangleq \lambda x\,.\,x\end{aligned}$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = (\mathbf{S}\,(\mathbf{K}\,(\mathbf{S}\,\mathbf{I}))\,(\mathbf{S}\,\underline{\lambda x\,.\,\mathbf{K}}\,\underline{\lambda x\,.\,x}))$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\begin{aligned}
\mathbf{S} &\triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\;z\;(y\;z)\\
\mathbf{K} &\triangleq \lambda x\,.\,\lambda y\,.\,x\\
\mathbf{I} &\triangleq \lambda x\,.\,x
\end{aligned}$$

$$\lambda x\,.\,\lambda y\,.\,y\;x = (\mathbf{S}\;(\mathbf{K}\;(\mathbf{S}\;\mathbf{I}))\;(\mathbf{S}\;(\mathbf{K}\;\mathbf{K})\;\underline{\lambda x\,.\,x}))$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\begin{aligned}
\mathbf{S} &\triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\;z\;(y\;z)\\
\mathbf{K} &\triangleq \lambda x\,.\,\lambda y\,.\,x\\
\mathbf{I} &\triangleq \lambda x\,.\,x
\end{aligned}$$

$$\lambda x\,.\,\lambda y\,.\,y\;x = (\mathbf{S}\;(\mathbf{K}\;(\mathbf{S}\;\mathbf{I}))\;(\mathbf{S}\;(\mathbf{K}\;\mathbf{K})\;\mathbf{I}))$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \cancel{\lambda x\,.\,x} \quad \mathbf{S}\,\mathbf{K}\,\mathbf{K}$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = (\mathbf{S}\,(\mathbf{K}\,(\mathbf{S}\,\mathbf{I}))\,(\mathbf{S}\,(\mathbf{K}\,\mathbf{K})\,\mathbf{I}))$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

Those pesky names

$$\mathbf{S} \triangleq \lambda x\,.\,\lambda y\,.\,\lambda z\,.\,x\,z\,(y\,z)$$
$$\mathbf{K} \triangleq \lambda x\,.\,\lambda y\,.\,x$$
$$\mathbf{I} \triangleq \cancel{\lambda x\,.\,x} \quad \mathbf{S}\,\mathbf{K}\,\mathbf{K}$$

$$\lambda x\,.\,\lambda y\,.\,y\,x = (\mathbf{S}\,(\mathbf{K}\,(\mathbf{S}\,(\mathbf{S}\,\mathbf{K}\,\mathbf{K})))\,(\mathbf{S}\,(\mathbf{K}\,\mathbf{K})\,(\mathbf{S}\,\mathbf{K}\,\mathbf{K})))$$

Smullyan, Raymond. *To Mock a Mockingbird.* Alfred A. Knopf: New York, 1985.

The stack language is Turing-complete

The stack language is (almost) Turing-complete

The stack language is (almost) Turing-complete

$$
\begin{array}{rcl}
\textit{literal} & \ni & \texttt{0}, \texttt{1}, \texttt{42}, \ldots \\
\textit{builtin} & \triangleq & \texttt{+} \mid \texttt{*} \mid \texttt{sqrt} \mid \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\textit{instruction} & \triangleq & \textit{literal} \mid \textit{builtin} \mid \texttt{[}\textit{program}\texttt{]} \\
\textit{program} & \triangleq & \textit{instruction} \mid \textit{program}\ \textit{program}
\end{array}
$$

The stack language is (almost) Turing-complete

$$
\begin{aligned}
\textit{builtin} &\triangleq \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\textit{instruction} &\triangleq \textit{builtin} \mid \texttt{[}\textit{program}\texttt{]} \\
\textit{program} &\triangleq \textit{instruction} \mid \textit{program program}
\end{aligned}
$$

The stack language is (almost) Turing-complete

$$
\begin{aligned}
\textit{builtin} &\triangleq \texttt{apply} \mid \texttt{dig2} \mid \texttt{drop} \mid \texttt{dup} \mid \texttt{swap} \mid \ldots \\
\textit{instruction} &\triangleq \textit{builtin} \mid \texttt{[}\textit{program}\texttt{]} \\
\textit{program} &\triangleq \textit{instruction} \mid \textit{program program}
\end{aligned}
$$

$$
\begin{aligned}
0 &\triangleq \texttt{[drop]} \\
\text{succ} &\triangleq \texttt{quote [apply] compose [[dup]] swap dup [[compose]]} \\
&\quad\ \texttt{swap [apply] compose compose compose compose}
\end{aligned}
$$

Scott J Maddox. “Foundations of Dawn: The Untyped Concatenative Calculus”.
https://www.dawn-lang.org/posts/foundations-ucc/

The stack language is (almost) Turing-complete

$$
\begin{aligned}
[\![\texttt{[}p\texttt{]}]\!]\ \hat{v} &\triangleq \hat{v} \cdot \texttt{[}p\texttt{]} \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot \texttt{[}p\texttt{]} &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

The stack language is Turing-complete

$$
\begin{aligned}
[\![\texttt{[}p\texttt{]}]\!]\ \hat{v} &\triangleq \hat{v} \cdot \texttt{[}p\texttt{]} \\
[\![\texttt{apply}]\!]\ \hat{v} \cdot \texttt{[}p\texttt{]} &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{compose}]\!]\ \hat{v} \cdot \texttt{[}p_1\texttt{]} \cdot \texttt{[}p_2\texttt{]} &\triangleq \hat{v} \cdot \texttt{[}p_1\ \ p_2\texttt{]} \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot \texttt{[}p\texttt{]} &\triangleq \hat{v} \cdot \texttt{[}v\ \ p\texttt{]} \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot \texttt{[}p\texttt{]} &\triangleq ([\![p\ ]\!]\ \hat{v}) \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot \texttt{[}v\texttt{]}
\end{aligned}
$$

No, really

$$
\begin{aligned}
\mathbf{S} &\triangleq \texttt{swap dig2 dup dig2 cons swap dig2 apply} \\
\mathbf{K} &\triangleq \texttt{swap drop apply} \\
(M\,N) &\triangleq [N]\,M
\end{aligned}
$$

No, really

$$
\begin{aligned}
\mathbf{S} &\triangleq \texttt{swap dig2 dup dig2 cons swap dig2 apply} \\
\mathbf{K} &\triangleq \texttt{swap drop apply} \\
(M\,N) &\triangleq [N]\,M
\end{aligned}
$$

$$\lambda x\,.\,\lambda y\,.\,y\;x \;=\; (\mathbf{S}\;(\mathbf{K}\;(\mathbf{S}\;(\mathbf{S}\;\mathbf{K}\;\mathbf{K})))\;(\mathbf{S}\;(\mathbf{K}\;\mathbf{K})\;(\mathbf{S}\;\mathbf{K}\;\mathbf{K})))$$

No, really

$$
\begin{aligned}
\mathbf{S} &\triangleq \texttt{swap dig2 dup dig2 cons swap dig2 apply} \\
\mathbf{K} &\triangleq \texttt{swap drop apply} \\
(M\,N) &\triangleq [N]\,M
\end{aligned}
$$

$$\lambda x\,.\,\lambda y\,.\,y\;x = [[[[\mathbf{K}]\;\mathbf{K}]\;\mathbf{S}]\;[[\mathbf{K}]\;[\mathbf{K}]\;\mathbf{S}]]\;[[[[\mathbf{K}]\;[\mathbf{K}]\;\mathbf{S}]\;\mathbf{S}]\;\mathbf{K}]\;\mathbf{S}$$

No, really

$$\begin{aligned}
\mathbf{S} &\triangleq \texttt{swap dig2 dup dig2 cons swap dig2 apply} \\
\mathbf{K} &\triangleq \texttt{swap drop apply} \\
(M\,N) &\triangleq \texttt{[}N\texttt{]}\,M
\end{aligned}$$

$$\begin{aligned}
\lambda x\,.\,\lambda y\,.\,y\,x = {} & \texttt{[[[[swap drop apply] swap drop apply] swap dig2 dup dig2} \\
& \texttt{cons swap dig2 apply] [[swap drop apply] [swap drop apply]} \\
& \texttt{swap dig2 dup dig2 cons swap dig2 apply]] [[[[swap drop apply]} \\
& \texttt{[swap drop apply] swap dig2 dup dig2 cons swap dig2 apply]} \\
& \texttt{swap dig2 dup dig2 cons swap dig2 apply] swap drop apply]} \\
& \texttt{swap dig2 dup dig2 cons swap dig2 apply}
\end{aligned}$$

THE C PROGRAMMING LANGUAGE → λ → SK →

Minimalism

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ \ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ \ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

unquote

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{array}{ll}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] & \triangleq\ [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] & \triangleq\ \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq\ \hat{v} \cdot [v\ \ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 & \triangleq\ \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq\ ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v & \triangleq\ \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v & \triangleq\ \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v & \triangleq\ \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 & \triangleq\ \hat{v} \cdot v_2 \cdot v_1
\end{array}
$$

quote

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

concatenate

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{array}{ll}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] & \triangleq\ [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] & \triangleq\ \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq\ \hat{v} \cdot [v\ \ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 & \triangleq\ \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq\ ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v & \triangleq\ \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v & \triangleq\ \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v & \triangleq\ \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 & \triangleq\ \hat{v} \cdot v_2 \cdot v_1
\end{array}
$$

reorder

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ \ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

destroy

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ \ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ \ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \\
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1
\end{aligned}
$$

copy

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{cake}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \cdot [p\ v] \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{k}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{s}]\!]\ \hat{v} \cdot v \cdot [p_1] \cdot [p_2] &\triangleq [\![p_2]\!]\ \hat{v} \cdot [v\ p_1] \cdot v \\
[\![\texttt{sip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v} \cdot v) \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1 \\
[\![\texttt{take}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [p\ v]
\end{aligned}
$$

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{array}{lll}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] & \triangleq & [\![p]\!]\ \hat{v} \\
[\![\texttt{cake}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq & \hat{v} \cdot [v\ p] \cdot [p\ v] \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] & \triangleq & \hat{v} \cdot [p_1\ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq & \hat{v} \cdot [v\ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 & \triangleq & \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq & ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v & \triangleq & \hat{v}
\end{array}
$$

$$
\begin{array}{lll}
[\![\texttt{dup}]\!]\ \hat{v} \cdot v & \triangleq & \hat{v} \cdot v \cdot v \\
[\![\texttt{k}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq & [\![p]\!]\ \hat{v} \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v & \triangleq & \hat{v} \cdot [v] \\
[\![\texttt{s}]\!]\ \hat{v} \cdot v \cdot [p_1] \cdot [p_2] & \triangleq & [\![p_2]\!]\ \hat{v} \cdot [v\ p_1] \cdot v \\
[\![\texttt{sip}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq & ([\![p]\!]\ \hat{v} \cdot v) \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 & \triangleq & \hat{v} \cdot v_2 \cdot v_1 \\
[\![\texttt{take}]\!]\ \hat{v} \cdot v \cdot [p] & \triangleq & \hat{v} \cdot [p\ v]
\end{array}
$$

destroy

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{cake}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \cdot [p\ v] \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{k}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{s}]\!]\ \hat{v} \cdot v \cdot [p_1] \cdot [p_2] &\triangleq [\![p_2]\!]\ \hat{v} \cdot [v\ p_1] \cdot v \\
[\![\texttt{sip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v} \cdot v) \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1 \\
[\![\texttt{take}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [p\ v]
\end{aligned}
$$

copy

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

$$
\begin{aligned}
[\![\texttt{apply}]\!]\ \hat{v} \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{cake}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \cdot [p\ v] \\
[\![\texttt{compose}]\!]\ \hat{v} \cdot [p_1] \cdot [p_2] &\triangleq \hat{v} \cdot [p_1\ p_2] \\
[\![\texttt{cons}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [v\ p] \\
[\![\texttt{dig2}]\!]\ \hat{v} \cdot v_1 \cdot v_2 \cdot v_3 &\triangleq \hat{v} \cdot v_2 \cdot v_3 \cdot v_1 \\
[\![\texttt{dip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v}) \cdot v \\
[\![\texttt{drop}]\!]\ \hat{v} \cdot v &\triangleq \hat{v}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{dup}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot v \cdot v \\
[\![\texttt{k}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq [\![p]\!]\ \hat{v} \\
[\![\texttt{quote}]\!]\ \hat{v} \cdot v &\triangleq \hat{v} \cdot [v] \\
[\![\texttt{s}]\!]\ \hat{v} \cdot v \cdot [p_1] \cdot [p_2] &\triangleq [\![p_2]\!]\ \hat{v} \cdot [v\ p_1] \cdot v \\
[\![\texttt{sip}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq ([\![p]\!]\ \hat{v} \cdot v) \cdot v \\
[\![\texttt{swap}]\!]\ \hat{v} \cdot v_1 \cdot v_2 &\triangleq \hat{v} \cdot v_2 \cdot v_1 \\
[\![\texttt{take}]\!]\ \hat{v} \cdot v \cdot [p] &\triangleq \hat{v} \cdot [p\ v]
\end{aligned}
$$

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

```
apply   = [[]] dip k
cake    = cons [cons sip] dup apply take
compose = [[apply] dip apply] cons cons
cons    = cake [] k
dig2    = [] cons cons dip
dip     = cake k
drop    = [] k
dup     = [] cake dip dip
k       = [drop] dip apply
quote   = [] cons
s       = dig2 [dig2 cons] sip dig2 apply
sip     = [quote compose] s apply
swap    = quote dip
take    = [dip] cons cons
```

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

```
apply      unquote
cake    = cons [cons sip] dup apply take
compose    concatenate
cons    = cake [] k
dig2    = [] cons cons dip
dip     = cake k
drop       destroy
dup        copy
k       = [drop] dip apply
quote      quote
s       = dig2 [dig2 cons] sip dig2 apply
sip     = [quote compose] s apply
swap       reorder
take    = [dip] cons cons
```

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

```
apply   = [[]] dip k
cake    = cons [cons sip] dup apply take
compose = [[apply] dip apply] cons cons
cons      concatenate, quote
dig2    = [] cons cons dip
dip     = cake k
drop    = [] k
dup     = [] cake dip dip
k         destroy, unquote
quote   = [] cons
s       = dig2 [dig2 cons] sip dig2 apply
sip       copy, reorder, unquote
swap    = quote dip
take    = [dip] cons cons
```

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Minimalism

```
apply   = [[]] dip k
cake      concatenate, copy, reorder, quote
compose = [[apply] dip apply] cons cons
cons    = cake [] k
dig2    = [] cons cons dip
dip     = cake k
drop    = [] k
dup     = [] cake dip dip
k         destroy, unquote
quote   = [] cons
s       = dig2 [dig2 cons] sip dig2 apply
sip     = [quote compose] s apply
swap    = quote dip
take    = [dip] cons cons
```

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Apologies in advance

$\lambda x\,.\,\lambda y\,.\,y\;x\;=$ `[[[[swap drop apply] swap drop apply] swap dig2 dup dig2 cons swap dig2 apply] [[swap drop apply] [swap drop apply] swap dig2 dup dig2 cons swap dig2 apply]] [[[[swap drop apply] [swap drop apply] swap dig2 dup dig2 cons swap dig2 apply] swap dig2 dup dig2 cons swap dig2 apply] swap drop apply] swap dig2 dup dig2 cons swap dig2 apply`

Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html

Apologies in advance

$\lambda x\,.\,\lambda y\,.\,y\;x\;=$

```
[[[[k] k] [] cake [] k cake k [] cake [] k cake [] k cake k [] cake cake k
cake k [] cake [] k cake [] k cake k cake [] k [] cake [] k cake k [] cake
[] k cake [] k cake k [[]] cake k k] [[k] [k] [] cake [] k cake k [] cake
[] k cake [] k cake k [] cake cake k cake k [] cake [] k cake [] k cake k
cake [] k [] cake [] k cake k [] cake [] k cake [] k cake k [[]] cake k k]]
[[[[k] [k] [] cake [] k cake k [] cake [] k cake [] k cake k [] cake cake
k cake k [] cake [] k cake [] k cake k cake [] k [] cake [] k cake k []
cake [] k cake [] k cake k [[]] cake k k] [] cake [] k cake k [] cake [] k
cake [] k cake k [] cake cake k cake k [] cake [] k cake [] k cake k cake
[] k [] cake [] k cake k [] cake [] k cake [] k cake k [[]] cake k k] k]
[] cake [] k cake k [] cake [] k cake [] k cake k [] cake cake k cake k []
cake [] k cake [] k cake k cake [] k [] cake [] k cake k [] cake [] k cake
[] k cake k [[]] cake k k
```


Brent Kirby. “The theory of concatenative combinators”.
http://tunes.org/~iepos/joy.html


THE
C
PROGRAMMING
LANGUAGE
λ
SK


Associativity

Concatenation revisited

$$[\![p_1\ p_2]\!]\ \hat{v} \triangleq [\![p_2]\!]\ ([\![p_1]\!]\ \hat{v})$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Concatenation revisited

$$[\![p_1\ p_2]\!]\ \hat{v} \triangleq ([\![p_2]\!] \circ [\![p_1]\!])\ \hat{v}$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Concatenation revisited

$$[\![p_1\ p_2]\!] \triangleq [\![p_2]\!] \circ [\![p_1]\!]$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Concatenation revisited

$$[\![p_1\ p_2]\!] \triangleq [\![p_2]\!] \circ [\![p_1]\!]$$

Syntactic concatenation is semantic composition

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Concatenation revisited

$$[\![p_1\ p_2]\!] \triangleq [\![p_2]\!] \circ [\![p_1]\!]$$

Manfred von Thun. “The mathematical foundations of Joy”. 1994.
https://hypercubed.github.io/joy/html/j02maf.html

Associativity of stack transformers

$$[\![a\ b\ c\ d]\!] = \textbf{?}$$

Associativity of stack transformers

$$[\![(a\ b)\ (c\ d))]\!] = \textbf{?}$$

Associativity of stack transformers

$$[\![(a\ b)\ (c\ d))]\!] = [\![c\ d]\!] \circ [\![a\ b]\!]$$

Associativity of stack transformers

$$[\![(a\ b)\ (c\ d))]\!] = ([\![d]\!] \circ [\![c]\!]) \circ ([\![b]\!] \circ [\![a]\!])$$

Associativity of stack transformers

$$[\![(a\ b)\ (c\ d))]\!] = ([\![d]\!] \circ [\![c]\!]) \circ ([\![b]\!] \circ [\![a]\!])$$
$$[\![a\ (b\ (c\ d))]\!] = (([\![d]\!] \circ [\![c]\!]) \circ [\![b]\!]) \circ [\![a]\!]$$

Associativity of stack transformers

$$[\![(a\ b)\ (c\ d))]\!] = ([\![d]\!] \circ [\![c]\!]) \circ ([\![b]\!] \circ [\![a]\!])$$
$$[\![a\ (b\ (c\ d))]\!] = (([\![d]\!] \circ [\![c]\!]) \circ [\![b]\!]) \circ [\![a]\!]$$
$$[\![((a\ b)\ c)\ d)]\!] = [\![d]\!] \circ ([\![c]\!] \circ ([\![b]\!] \circ [\![a]\!]))$$

Associativity of stack transformers

$$
\begin{aligned}
[\![(a\ b)\ (c\ d))]\!] &= ([\![d]\!] \circ [\![c]\!]) \circ ([\![b]\!] \circ [\![a]\!]) \\
[\![a\ (b\ (c\ d))]\!] &= (([\![d]\!] \circ [\![c]\!]) \circ [\![b]\!]) \circ [\![a]\!] \\
[\![((a\ b)\ c)\ d)]\!] &= [\![d]\!] \circ ([\![c]\!] \circ ([\![b]\!] \circ [\![a]\!])) \\
[\![(a\ (b\ c))\ d]\!] &= [\![d]\!] \circ (([\![c]\!] \circ [\![b]\!]) \circ [\![a]\!])
\end{aligned}
$$

Associativity of stack transformers

$$[\![a\ b\ c\ d]\!] = \left\{ \begin{array}{l} [\![(a\ b)\ (c\ d))]\!] = ([\![d]\!] \circ [\![c]\!]) \circ ([\![b]\!] \circ [\![a]\!]) \\ [\![a\ (b\ (c\ d))]\!] = (([\![d]\!] \circ [\![c]\!]) \circ [\![b]\!]) \circ [\![a]\!] \\ [\![((a\ b)\ c)\ d)]\!] = [\![d]\!] \circ ([\![c]\!] \circ ([\![b]\!] \circ [\![a]\!])) \\ [\![(a\ (b\ c))\ d]\!] = [\![d]\!] \circ (([\![c]\!] \circ [\![b]\!]) \circ [\![a]\!]) \end{array} \right\} = [\![d]\!] \circ [\![c]\!] \circ [\![b]\!] \circ [\![a]\!]$$

Associativity of stack transformers

$$[\![a\ b\ c\ d]\!] = \left\{ \begin{array}{l} [\![(a\ b)\ (c\ d))]\!] = ([\![d]\!] \circ [\![c]\!]) \circ ([\![b]\!] \circ [\![a]\!]) \\ [\![a\ (b\ (c\ d))]\!] = (([\![d]\!] \circ [\![c]\!]) \circ [\![b]\!]) \circ [\![a]\!] \\ [\![((a\ b)\ c)\ d)]\!] = [\![d]\!] \circ ([\![c]\!] \circ ([\![b]\!] \circ [\![a]\!])) \\ [\![(a\ (b\ c))\ d]\!] = [\![d]\!] \circ (([\![c]\!] \circ [\![b]\!]) \circ [\![a]\!]) \end{array} \right\} = [\![d]\!] \circ [\![c]\!] \circ [\![b]\!] \circ [\![a]\!]$$

This is *not* changing the order that we execute instructions!
This is *not* the associativity of $+$ and $\times$!

Jeremy Gibbons. "Continuation-Passing Style, Defunctionalization, Accumulations, and Associativity". The Art, Science, and Engineering of Programming, 2022, Vol. 6, Issue 2, Article 7.


Associativity ⇒ parallelism

Compiling stack programs

$\langle 0 \rangle \triangleq$
```
mov %ax, 0
push %ax
```

$\langle 1 \rangle \triangleq$
```
mov %ax, 1
push %ax
```

$\langle 42 \rangle \triangleq$
```
mov %ax, 42
push %ax
```

$\vdots$

$\langle + \rangle \triangleq$
```
pop %ax
add %ax, [%sp]
mov [%sp], %ax
```

$\langle * \rangle \triangleq$
```
pop %ax
mul [%sp]
mov [%sp], %ax
```

$\langle sqrt \rangle \triangleq$
```
fild [%sp]
fsqrt
fisstp [%sp]
```

$\langle dup \rangle \triangleq$
```
mov %ax, [%sp]
push %ax
```

$\langle drop \rangle \triangleq$
```
pop %ax
```

$\langle swap \rangle \triangleq$
```
pop %ax
pop %bx
push %ax
push %bx
```

Compiling stack programs

$\langle \texttt{0} \rangle \triangleq$
```
mov %ax, 0
push %ax
```

$\langle \texttt{1} \rangle \triangleq$
```
mov %ax, 1
push %ax
```

$\langle \texttt{42} \rangle \triangleq$
```
mov %ax, 42
push %ax
```

$\vdots$

$\langle \texttt{+} \rangle \triangleq$
```
pop %ax
add %ax, [%sp]
mov [%sp], %ax
```

$\langle \texttt{*} \rangle \triangleq$
```
pop %ax
mul [%sp]
mov [%sp], %ax
```

$\langle \texttt{sqrt} \rangle \triangleq$
```
fild [%sp]
fsqrt
fisstp [%sp]
```

$\langle \texttt{dup} \rangle \triangleq$
```
mov %ax, [%sp]
push %ax
```

$\langle \texttt{drop} \rangle \triangleq$
```
pop %ax
```

$\langle \texttt{swap} \rangle \triangleq$
```
pop %ax
pop %bx
push %ax
push %bx
```

$$\langle p_1\ p_2 \rangle \triangleq \langle p_1 \rangle +\!\!+ \langle p_2 \rangle$$

Compiling our example expression

```
3 4 dup * swap dup * + sqrt
```

Compiling our example expression

```
3 4 dup * swap dup * + sqrt
```

Compiling our example expression

```
1 2 + 2 2 * dup * swap dup * + sqrt
```

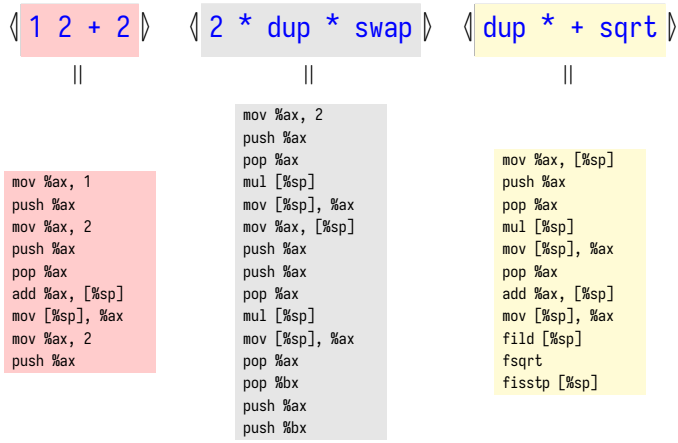
Compiling our example expression

```
1 2 + 2 | 2 * dup * swap dup * + sqrt
```

Compiling our example expression

```
1 2 + 2 2 * dup * swap dup * + sqrt
```

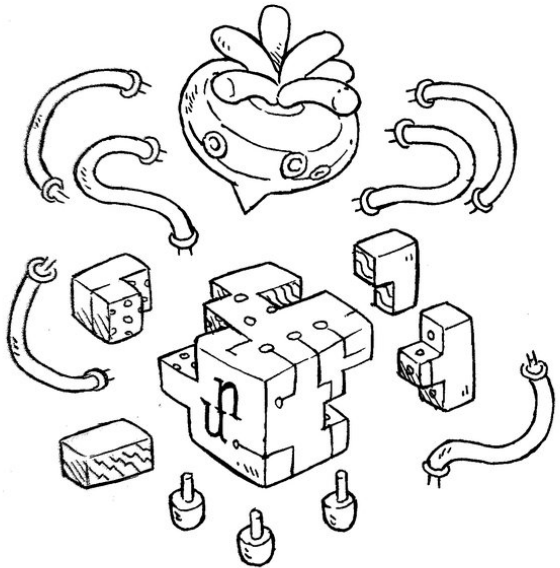
Compiling our example expression



Compiling our example expression

⟨ 1 2 + 2 2 * dup * swap dup * + sqrt ⟩ =

```
mov %ax, 1
push %ax
mov %ax, 2
push %ax
pop %ax
pop %bx
add %ax, %bx
push %ax
mov %ax, 2
push %ax
mov %ax, 2
push %ax
pop %ax
pop %bx
mul %bx
push %ax
pop %ax
push %ax
push %ax
pop %ax
pop %bx
mul %bx
push %ax
pop %ax
pop %bx
push %ax
push %bx
pop %ax
push %ax
push %ax
pop %ax
pop %bx
mul %bx
push %ax
pop %ax
pop %bx
add %ax, %bx
push %ax
fild [%sp]
fsqrt
fisstp [%sp]
```



uxn

https://100r.co/site/uxn.html

Picture credits

Slide 1	Douglas Creager, “Laurel Valley” All rights reserved
Slide 2	peagreengirl, “Very Old payroll Journal” CC-BY-2.0, https://flic.kr/p/B2YPU
Slide 5	Marco Verch, “Stack of pancakes with berries on a plate” CC-BY-2.0, https://flic.kr/p/2jYUh8M
Slide 7	striegel, “HP-48GX calculator” CC-BY-2.0, https://flic.kr/p/BWttzE
Slide 12	Mustang Joe, “The Thinker” Public domain, https://flic.kr/p/a6gTRj
Slide 23	Seattle Municipal Archives, “West Seattle Bridge under construction, circa 1983” CC-BY-2.0, https://flic.kr/p/7jKWYi
Slide 30	shmai, “Pancakes icon” Flaticon Free License (with attribution), https://www.flaticon.com/free-icon/pancakes_6785822
Slide 31	Tom Driggers, “Two Planters in Puerto Vallarta” CC-BY-2.0, https://flic.kr/p/NeHcZ6
Slide 35	monkik, “Cake slice icon” Flaticon Free License (with attribution), https://www.flaticon.com/free-icon/cake-slice_992754
Slide 36	Sentinel Hub, “Parts of the Irrawaddy River Delta, Myanmar – 25 April 2023” CC-BY-2.0, https://flic.kr/p/2ovU8m9
Slide 39	10 Cats.+, “10 Cats Eat Family Meals Together Every Day” Screenshot, https://www.youtube.com/watch?v=bs-2uVsKJr0
Slide 42	Hundredrabbits, “uxn explode” CC-BY-NC-SA-2.0, https://100r.co/media/content/projects/uxnexplode.jpg