Dynamic GoI machine

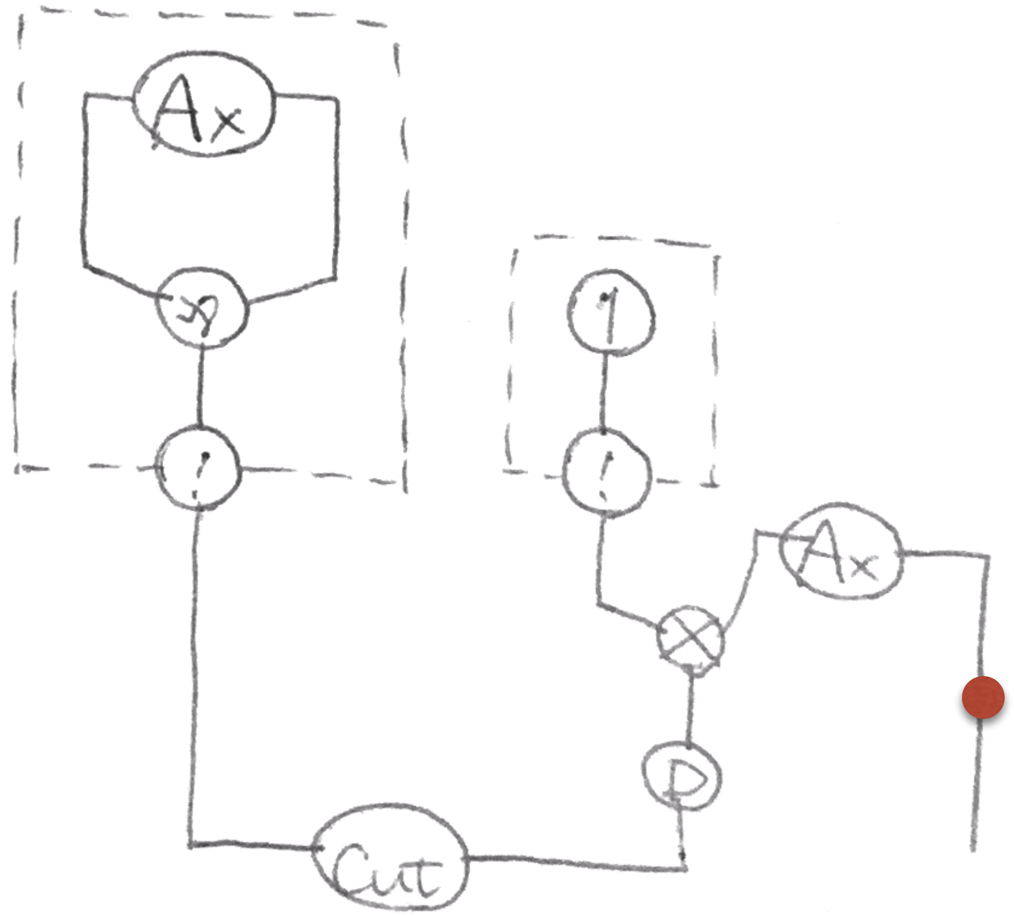
Call-by-need and call-by-value graph rewriter

Koko Muroya & Dan Ghica
(Univ. Birmingham)

CRECOGI (Marseille), 28 August 2016

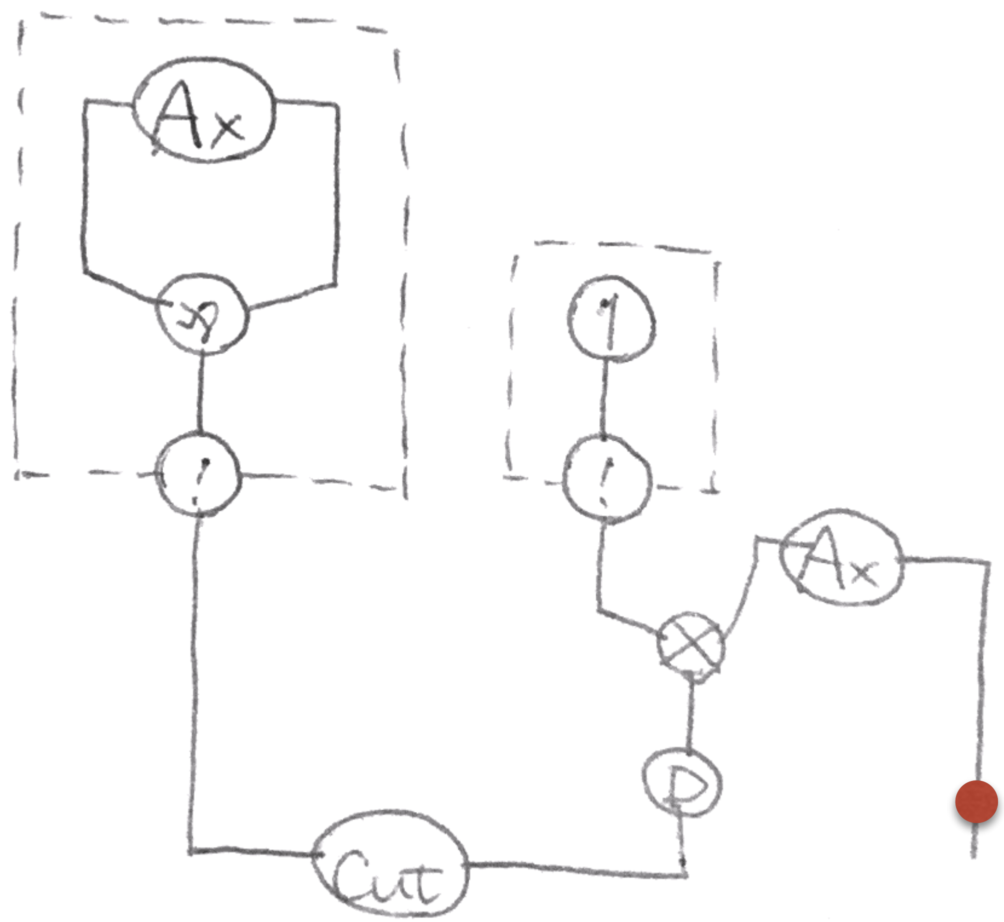
GoI machine [Danos & Regnier '99] [Mackie '95]

$$(\lambda x\,.\,x)\,1$$



GoI machine [Danos & Regnier '99] [Mackie '95]

$(\lambda x \,.\, x)\, 1$



GoI machine [Danos & Regnier '99] [Mackie '95]

call-by-name

call-by-value

effects

GoI machine [Danos & Regnier '99] [Mackie '95]

call-by-name

call-by-value

effects

	avoid re-evaluation	force evaluation of arguments	track each copy of terms
CPS transform.	?	✔	?
memory	△		✔
parallelism & sync.	△	✔	
dynamic jump	✔	✔	
dynamic rewrite	✔		☺
checkpoint		✔	

Schöpp

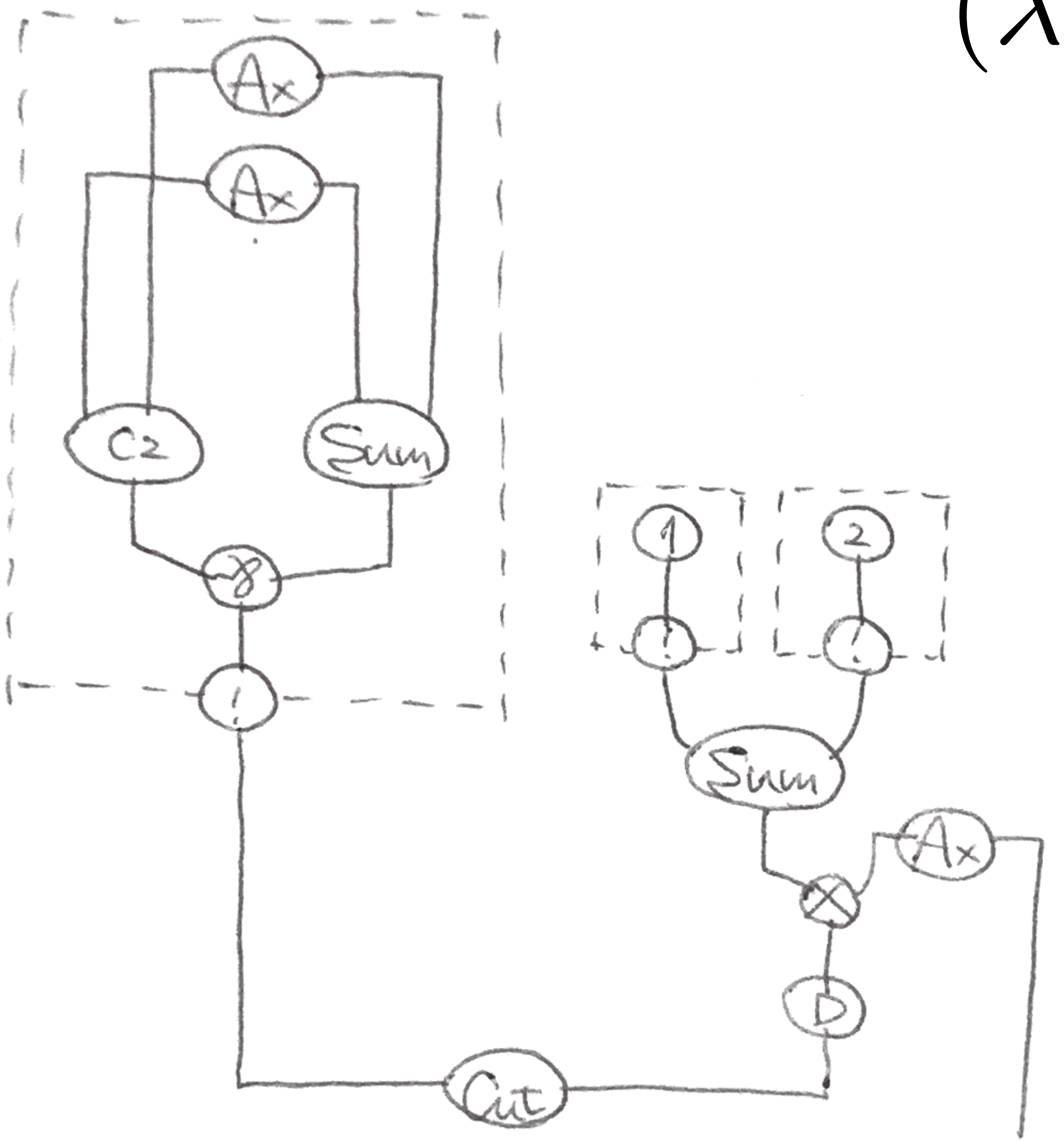
Hoshino+

Dal Lago+

Fernández+

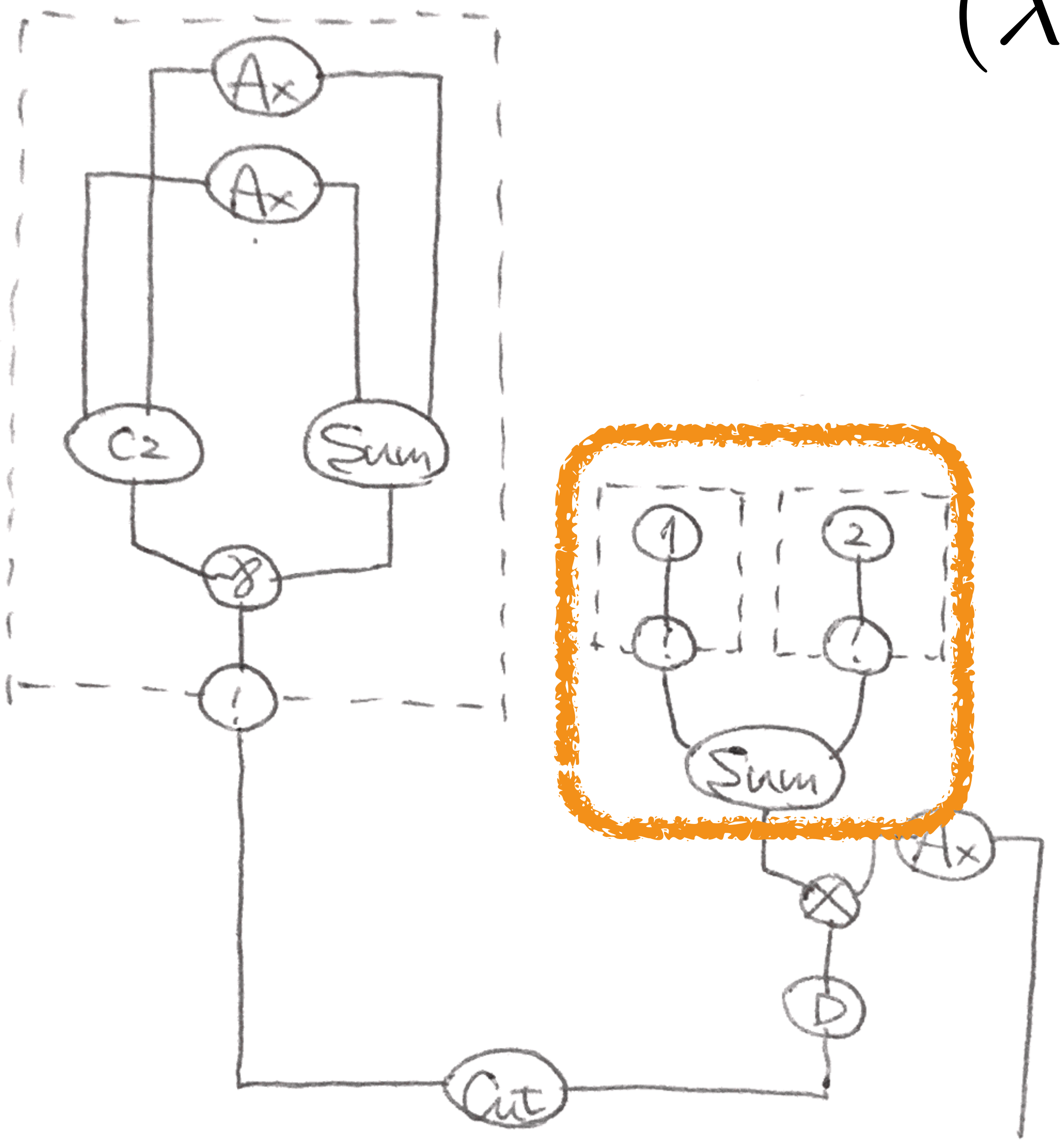
Avoid re-evaluation

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



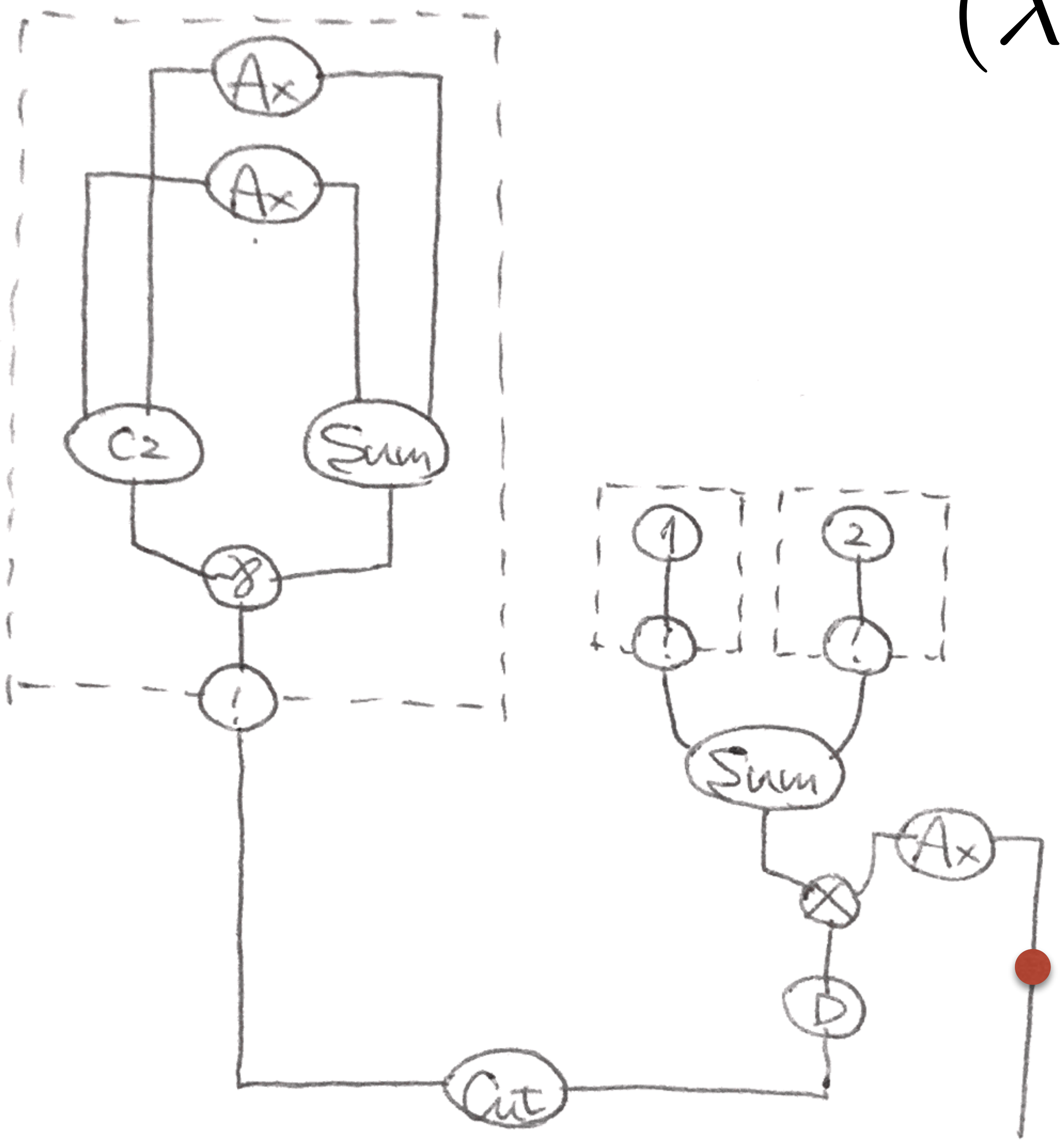
Avoid re-evaluation

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



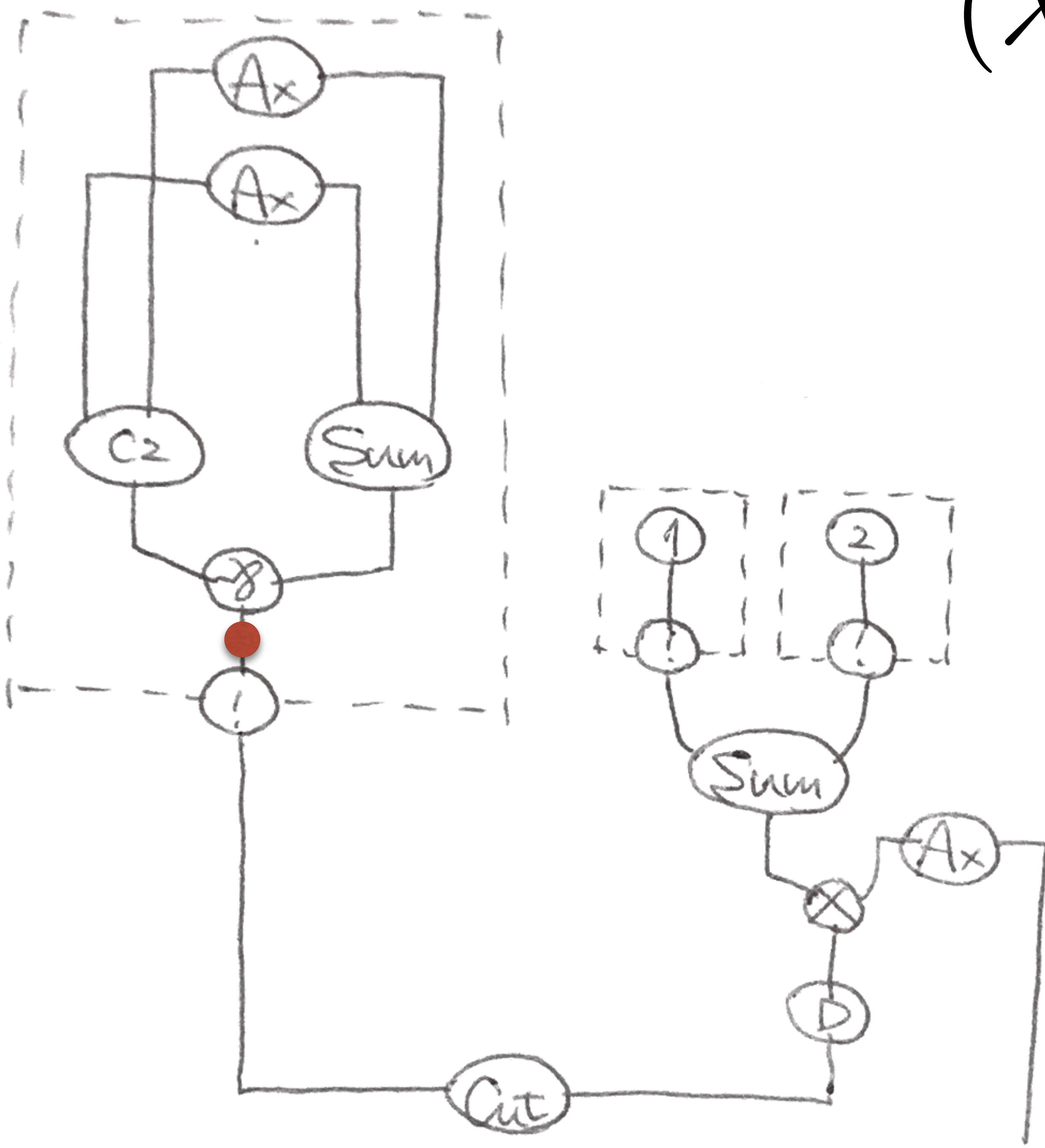
Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



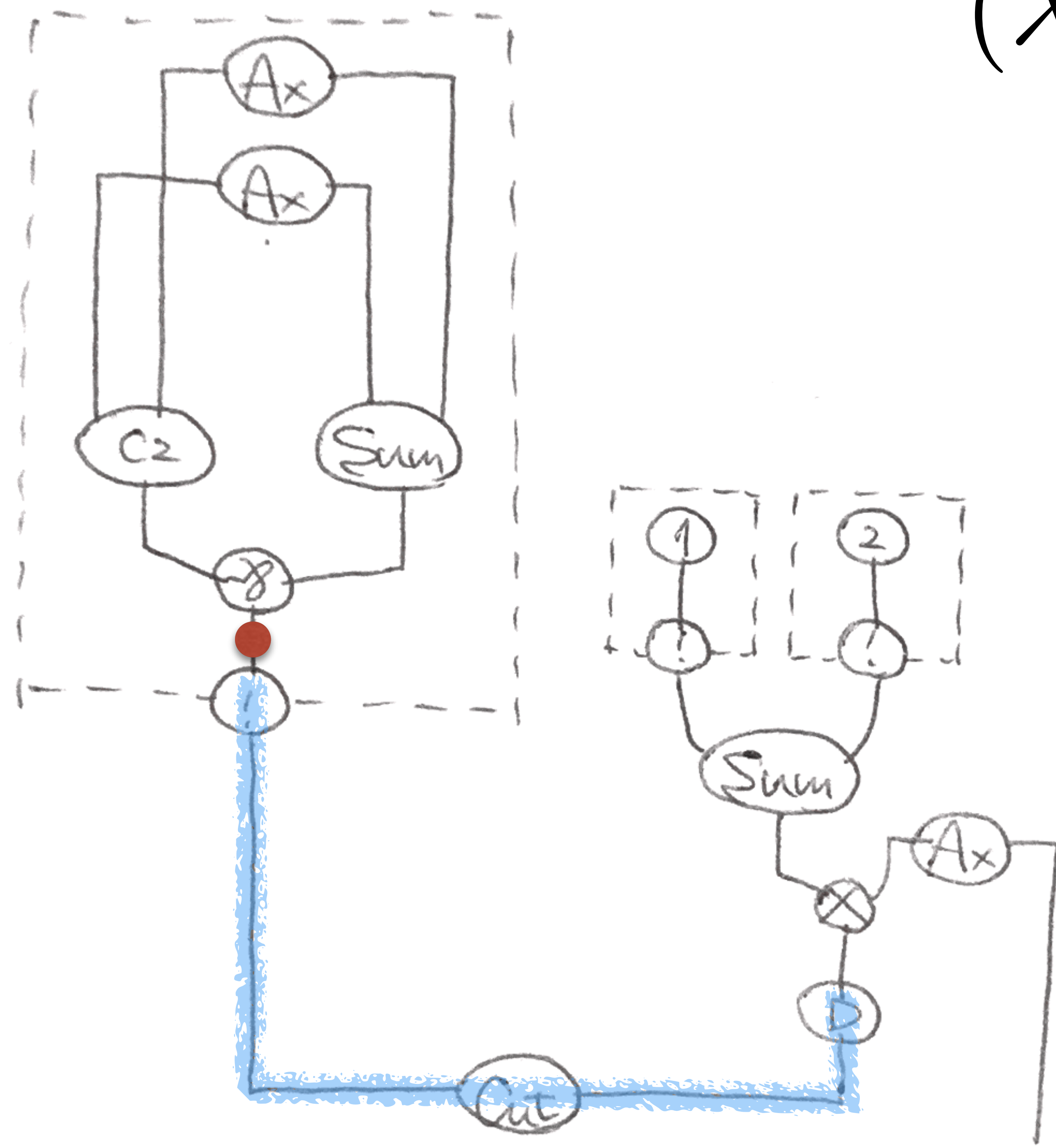
Avoid re-evaluation: dynamic rewrite

$$(\lambda x . x + x)\,(1 + 2)$$



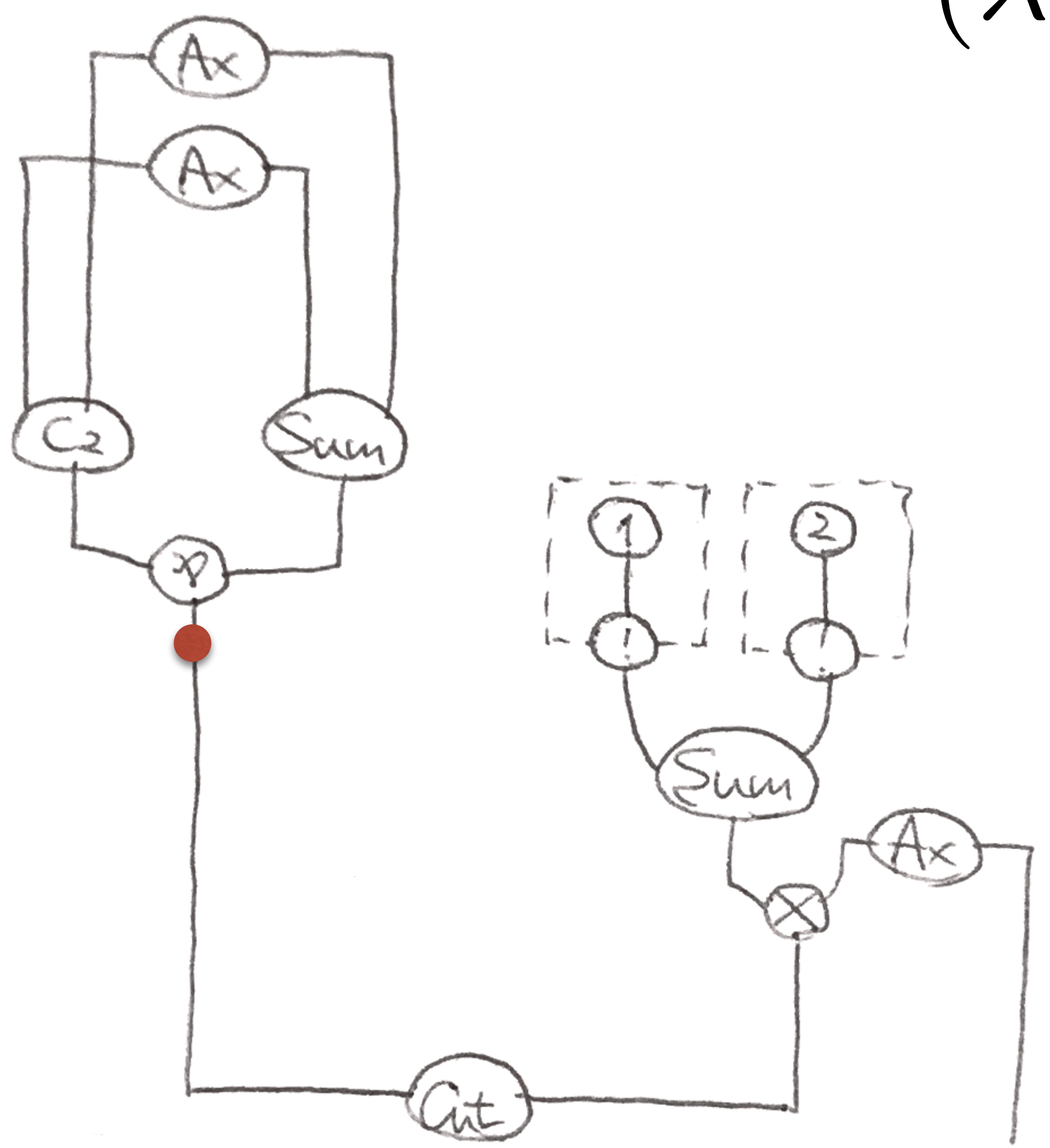
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\;(1 + 2)$$



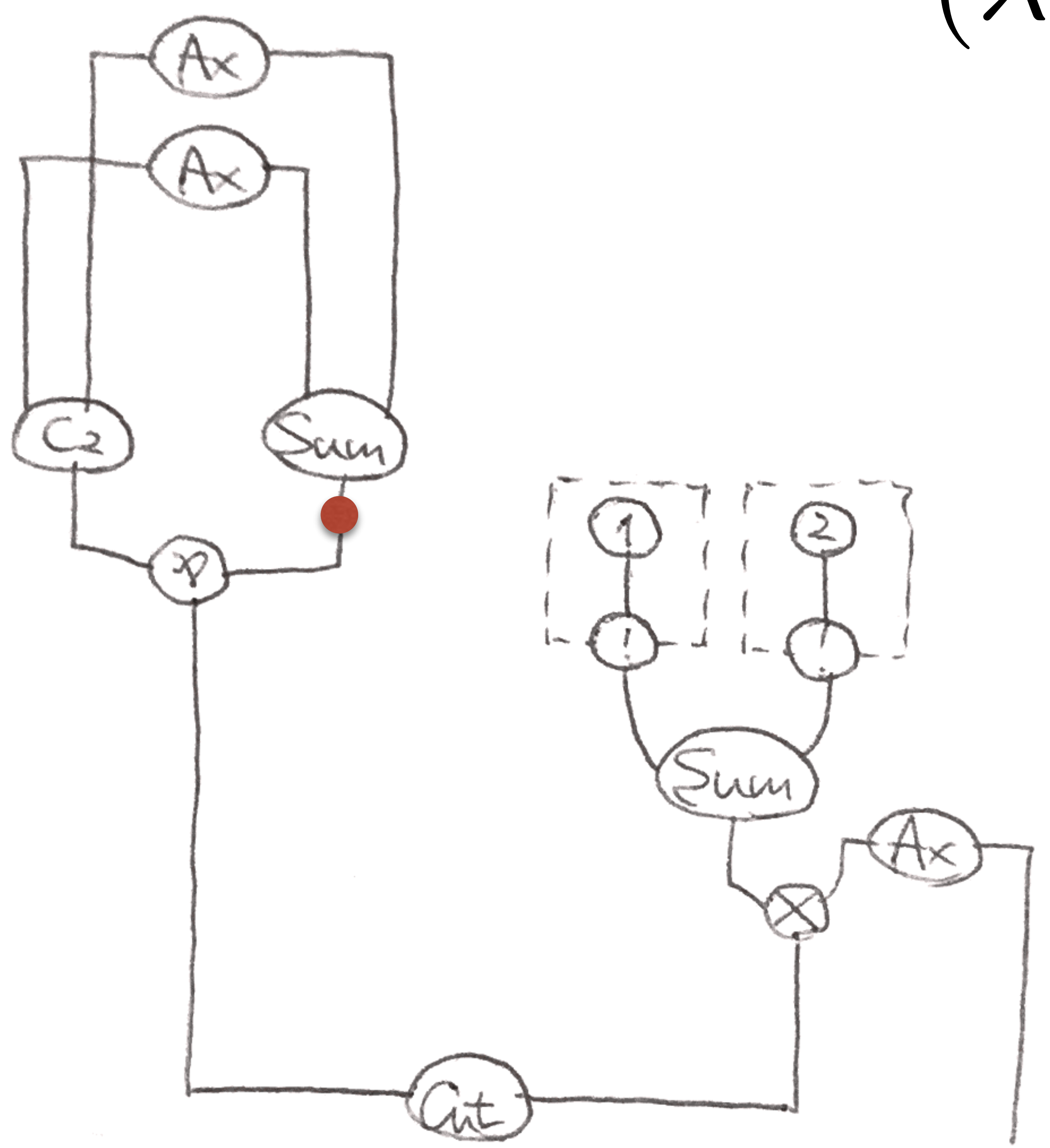
Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



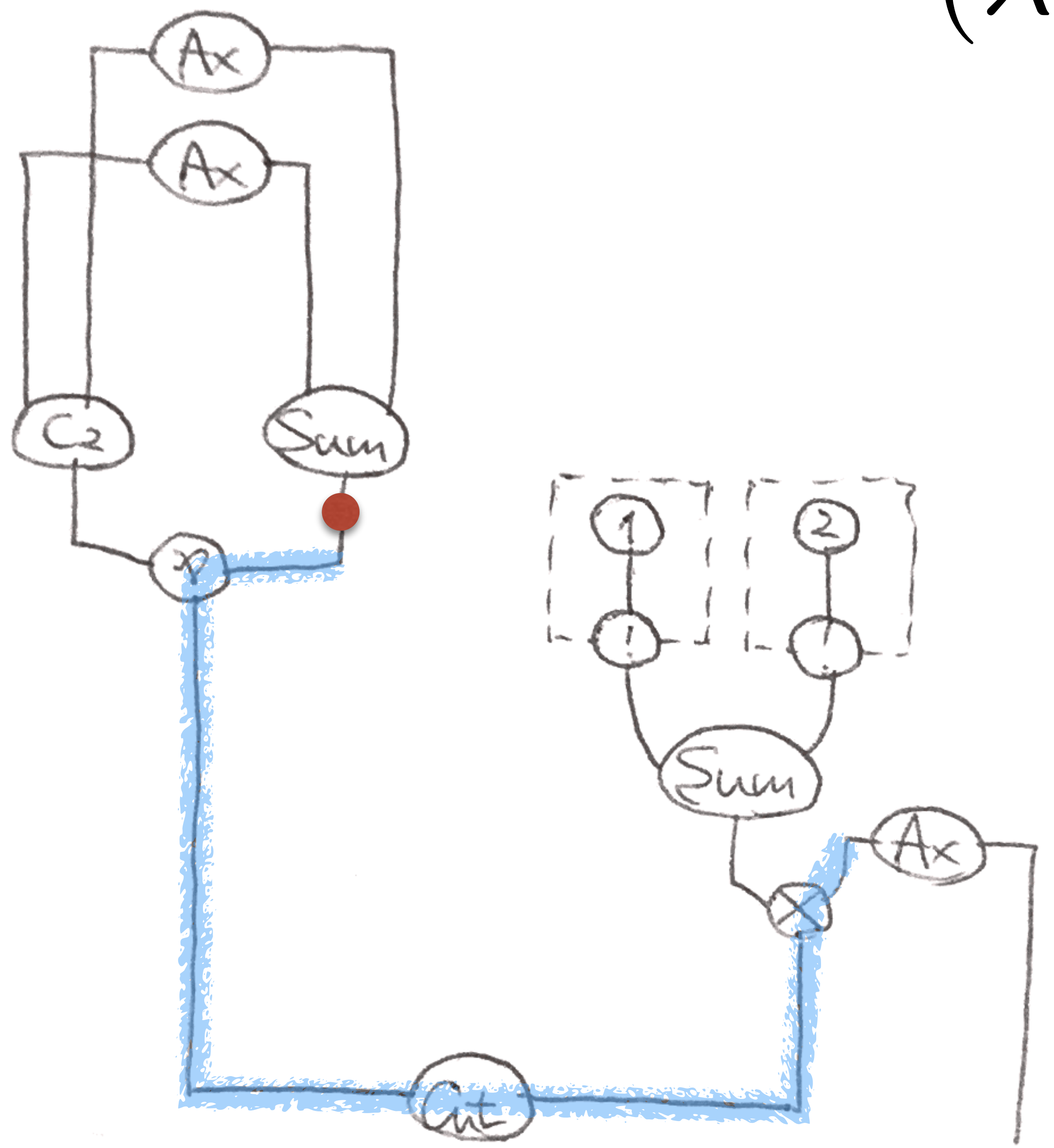
Avoid re-evaluation: dynamic rewrite

$$(\lambda x \,.\, x + x)\,(1 + 2)$$



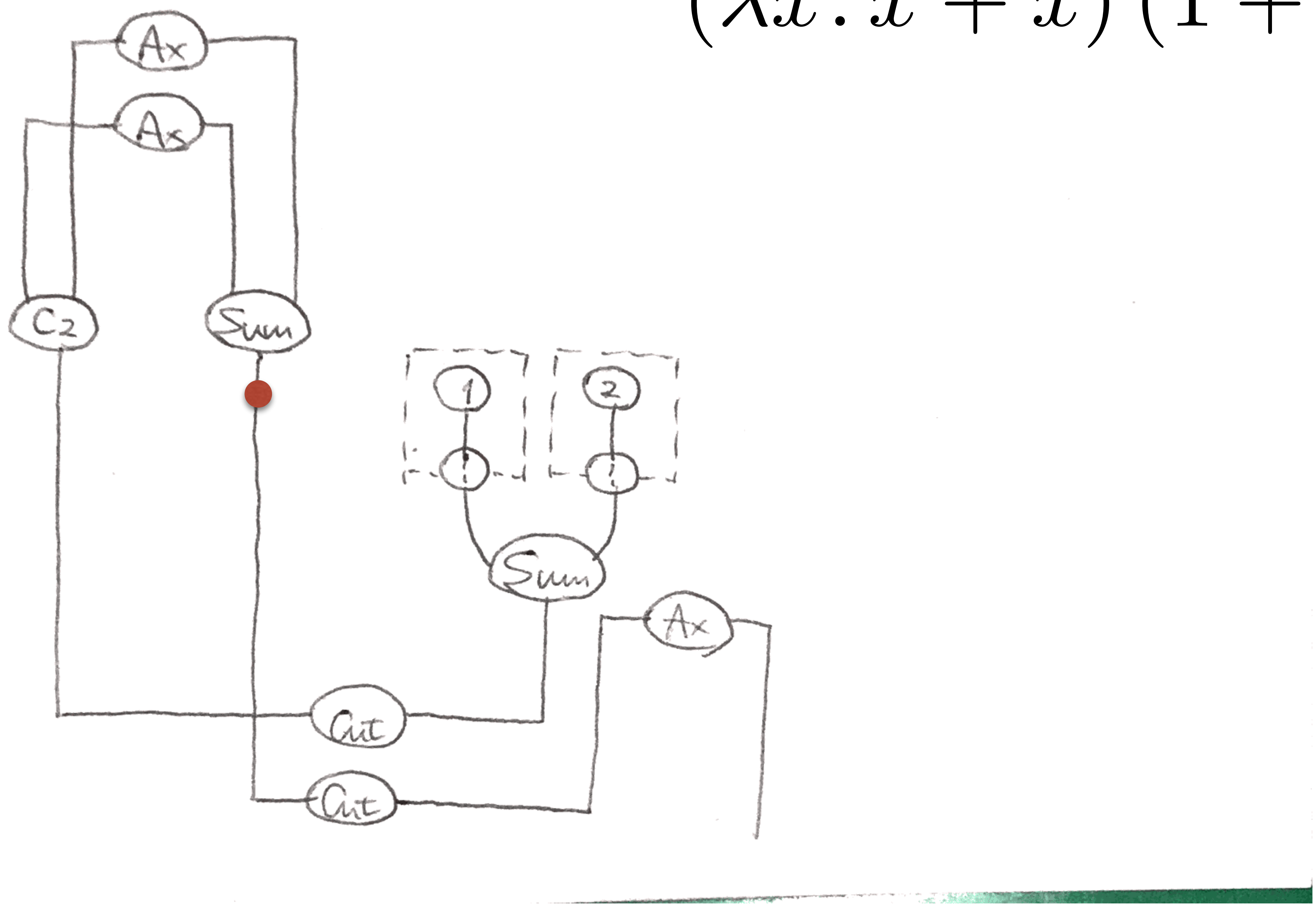
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



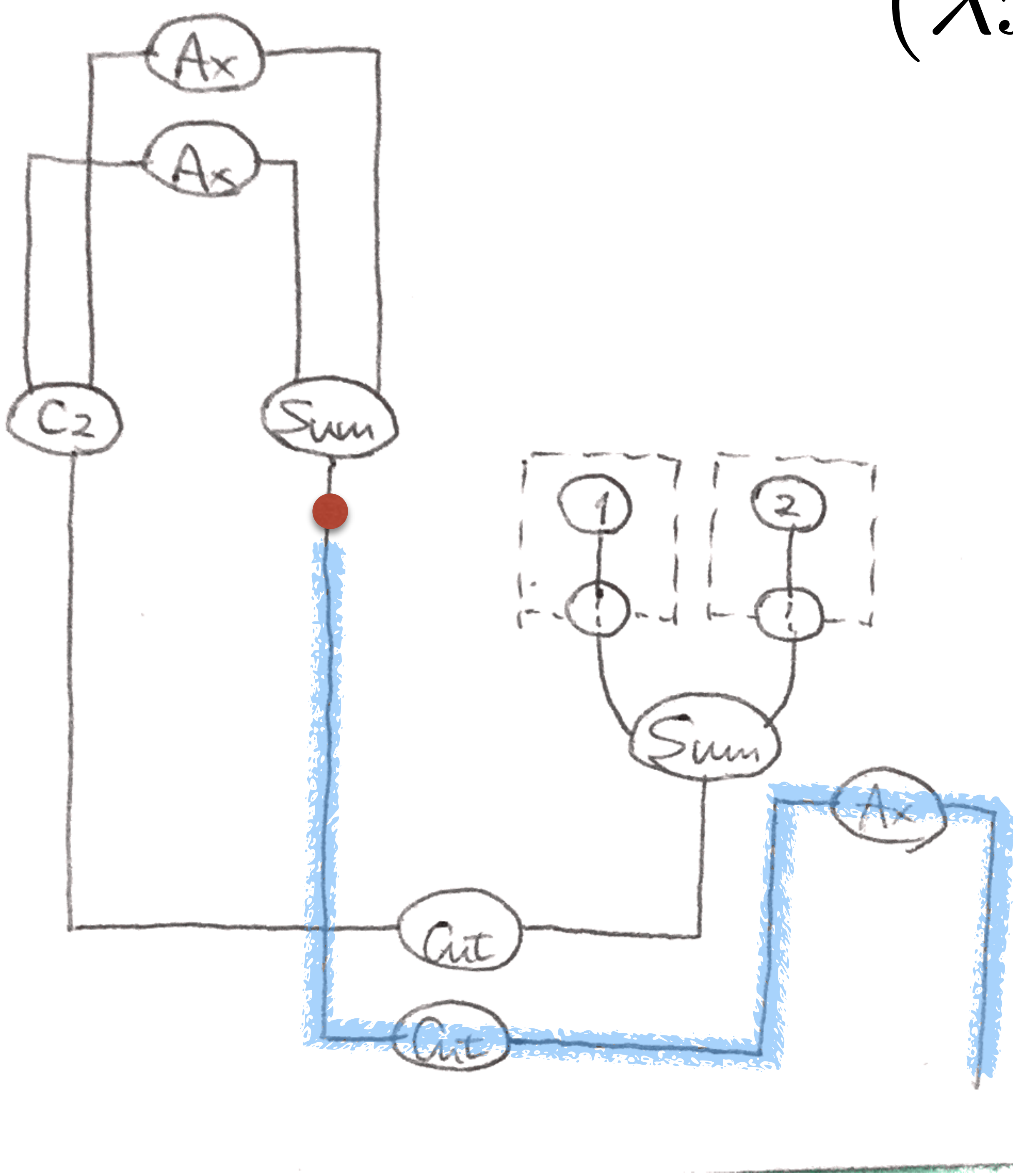
Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



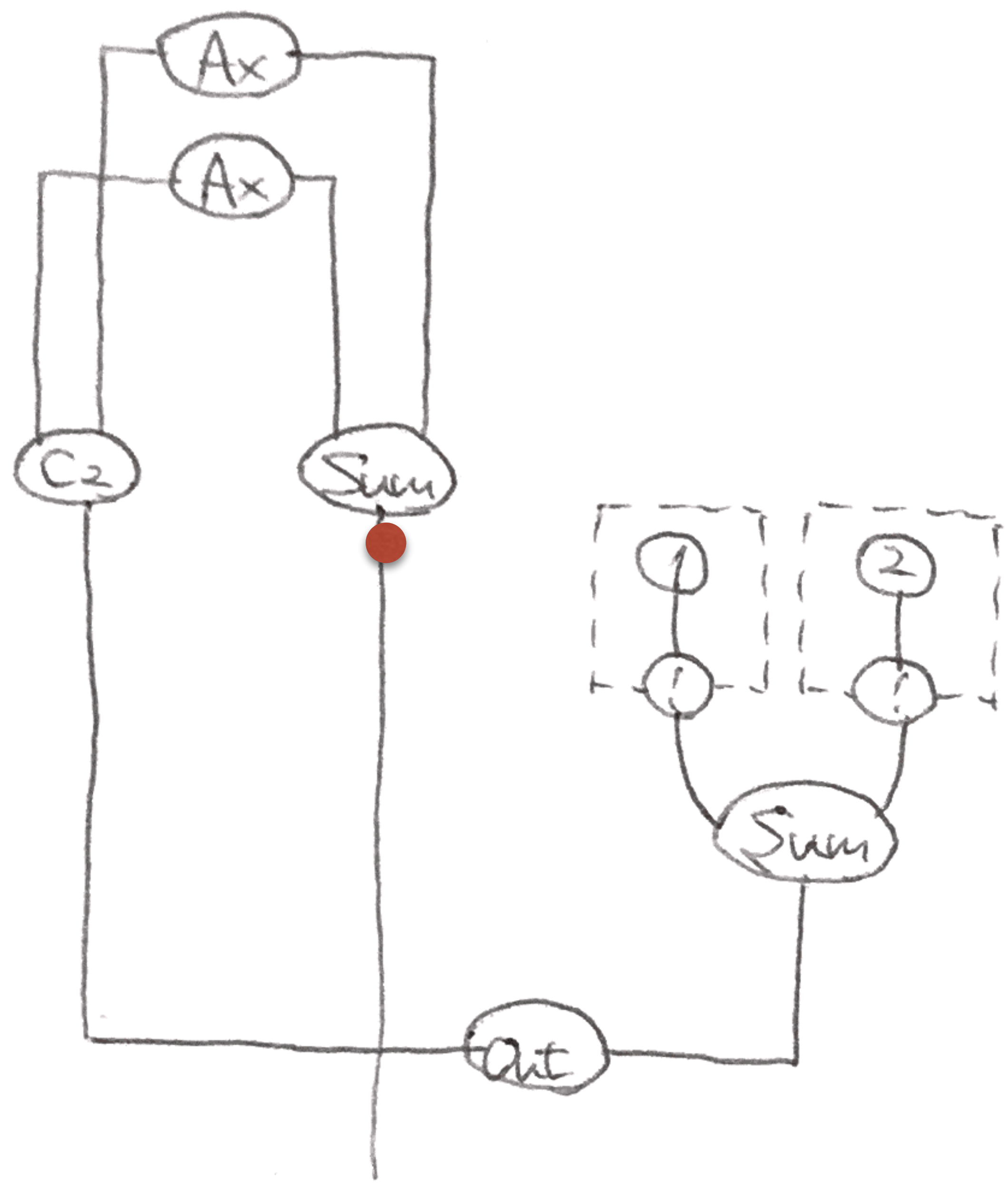
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



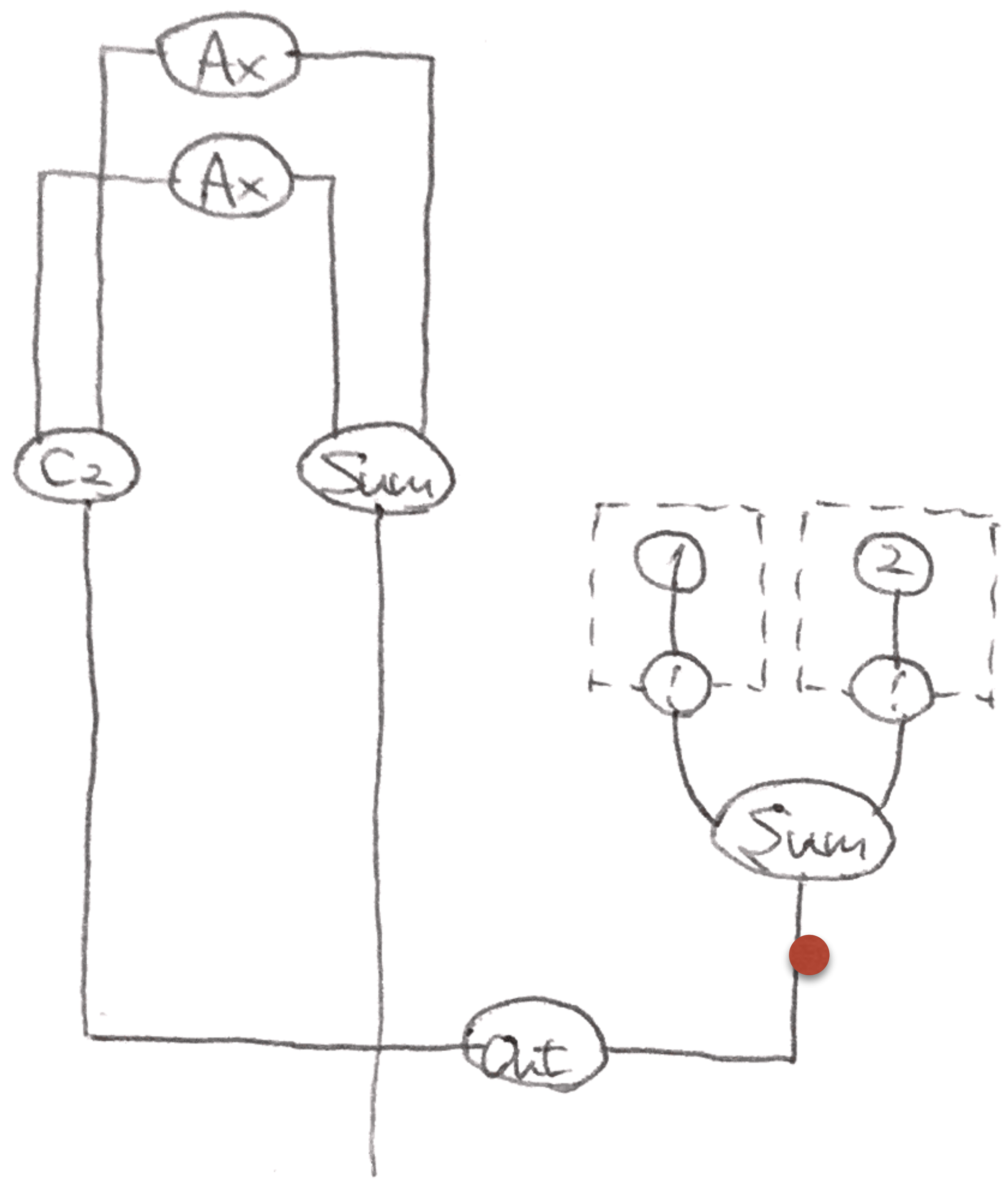
Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



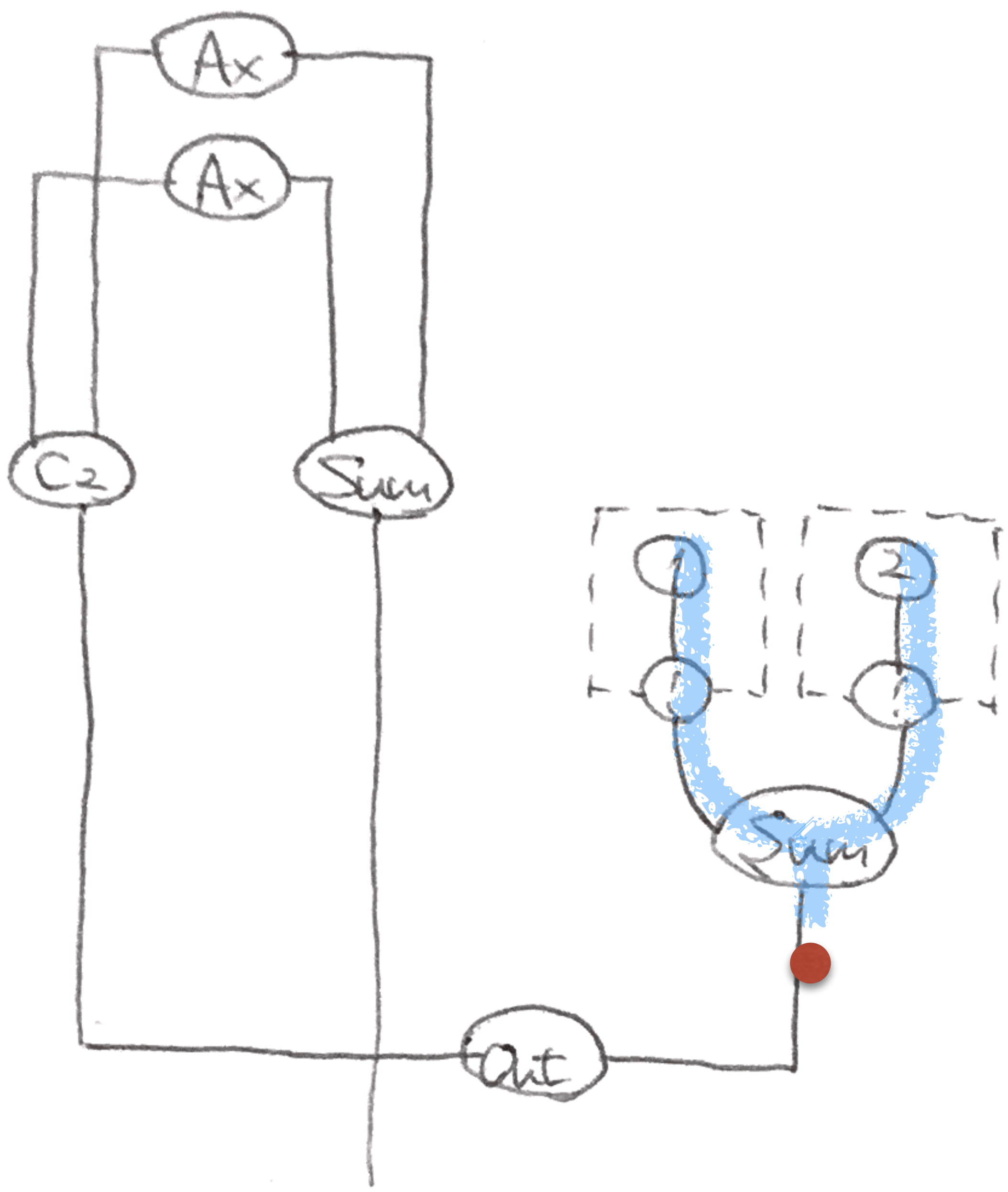
Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



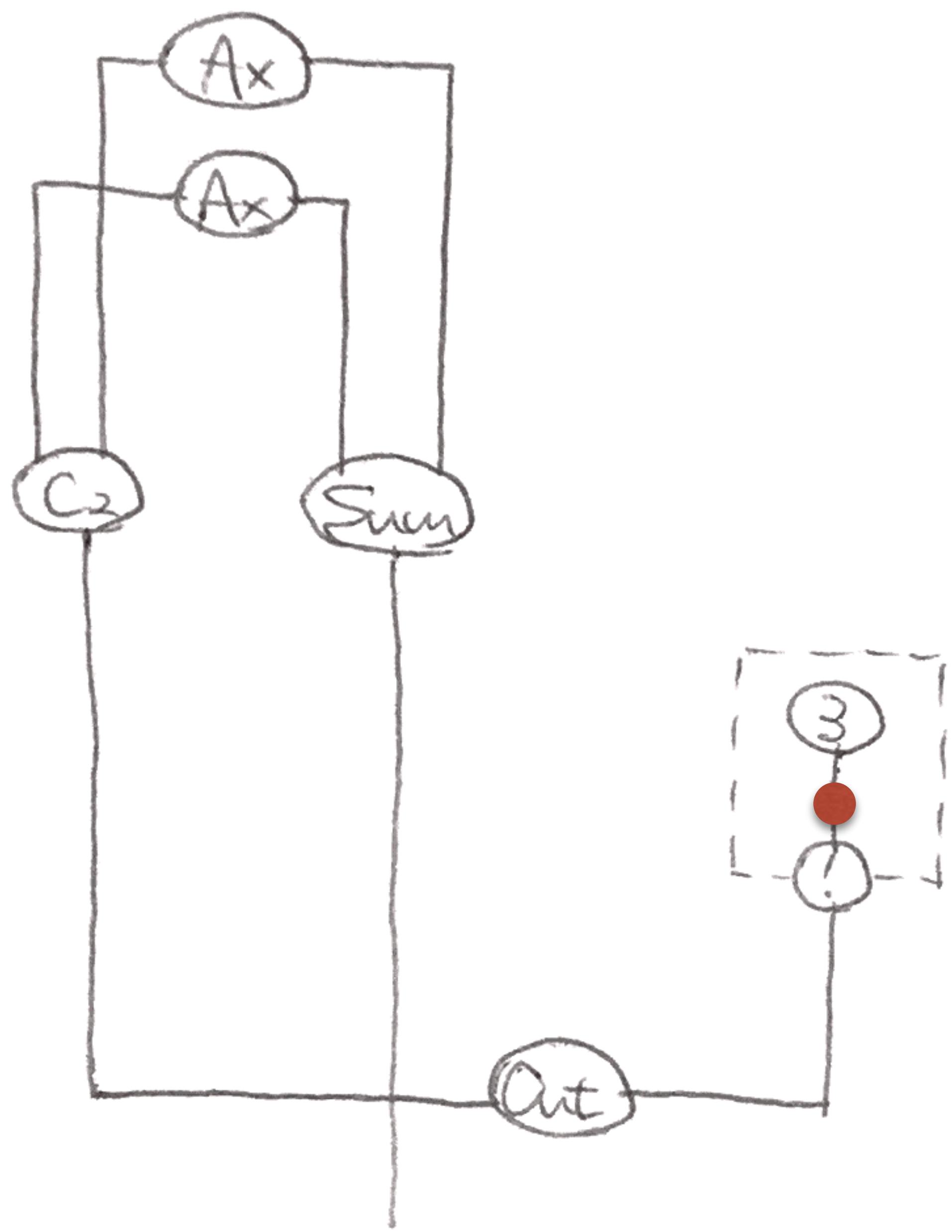
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



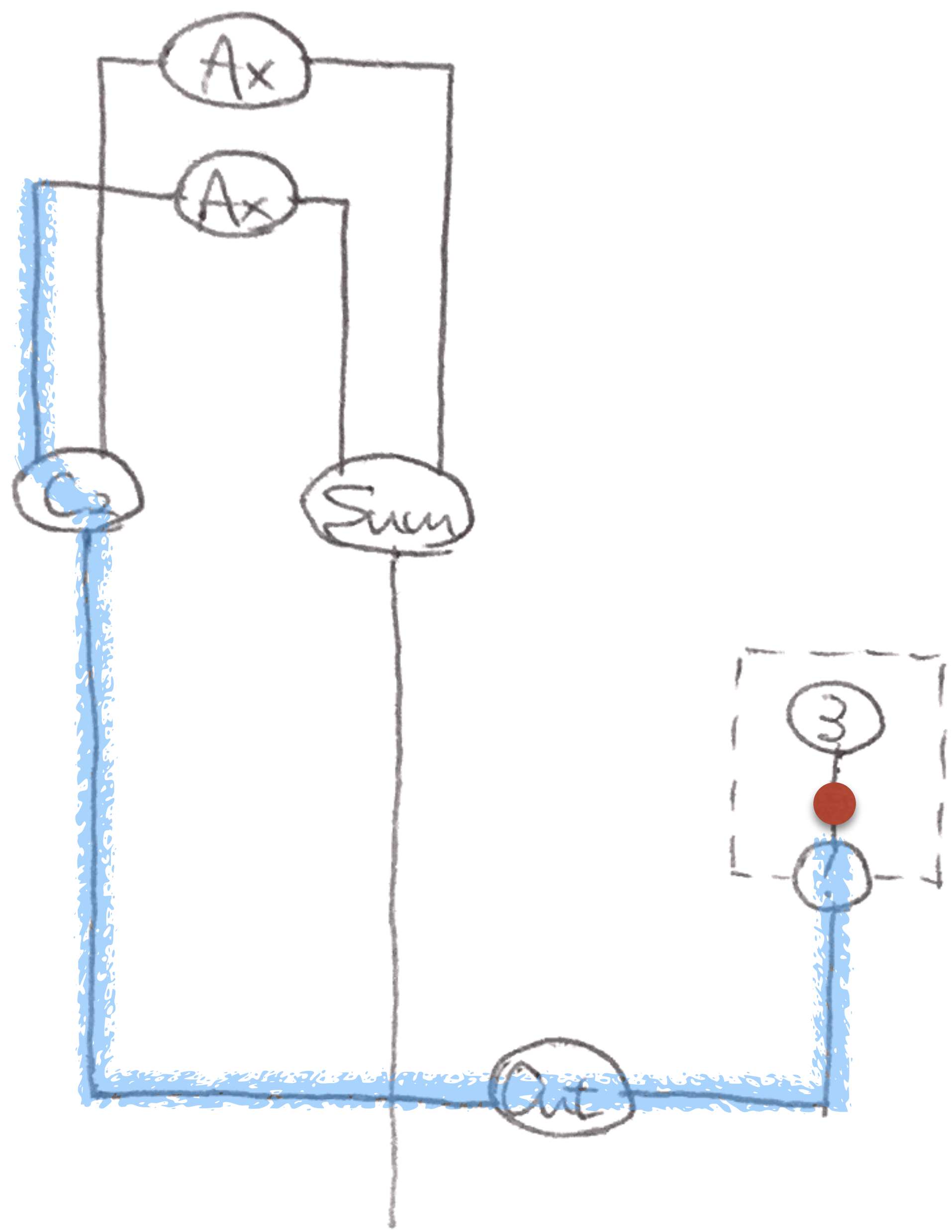
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



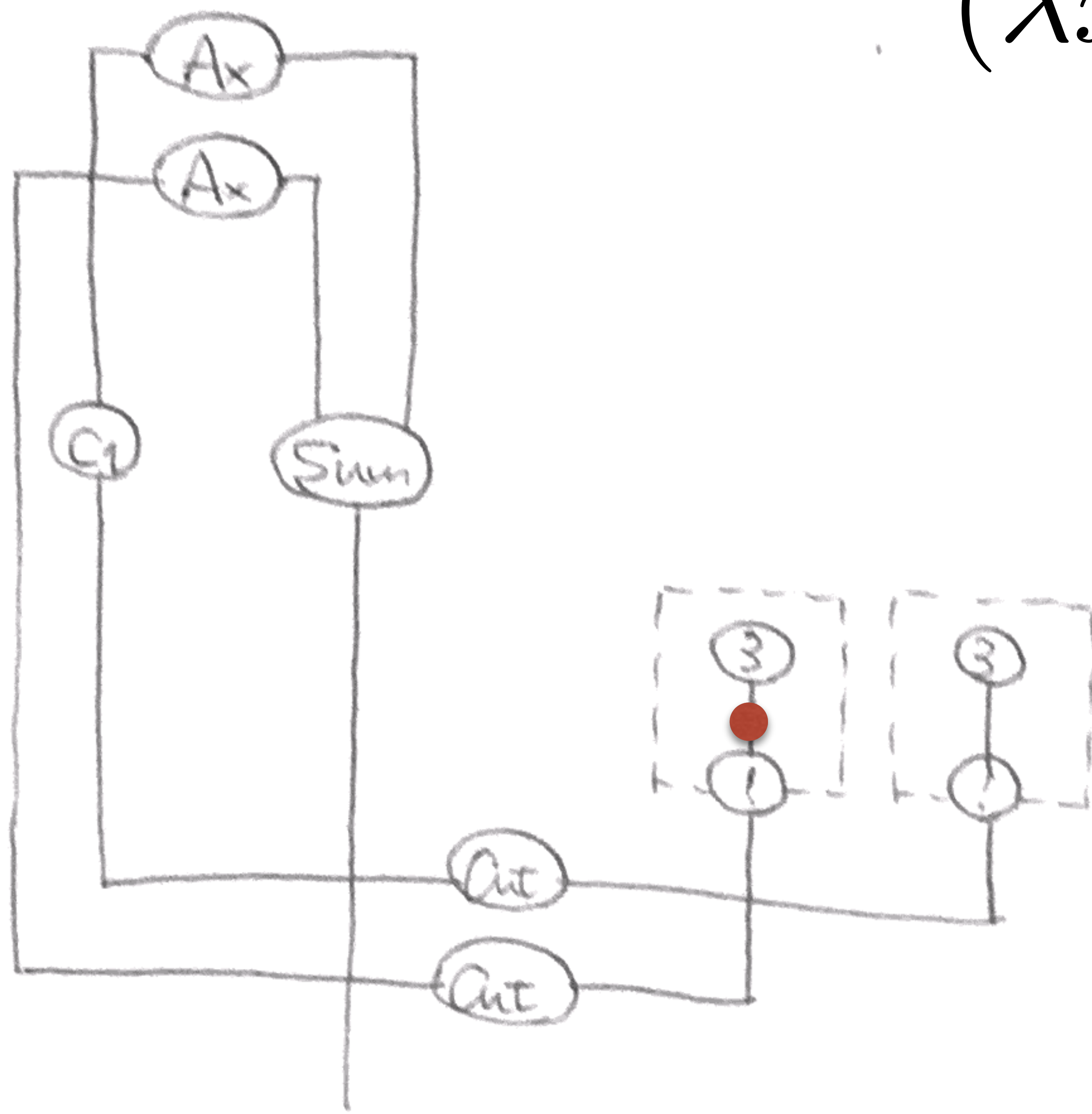
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



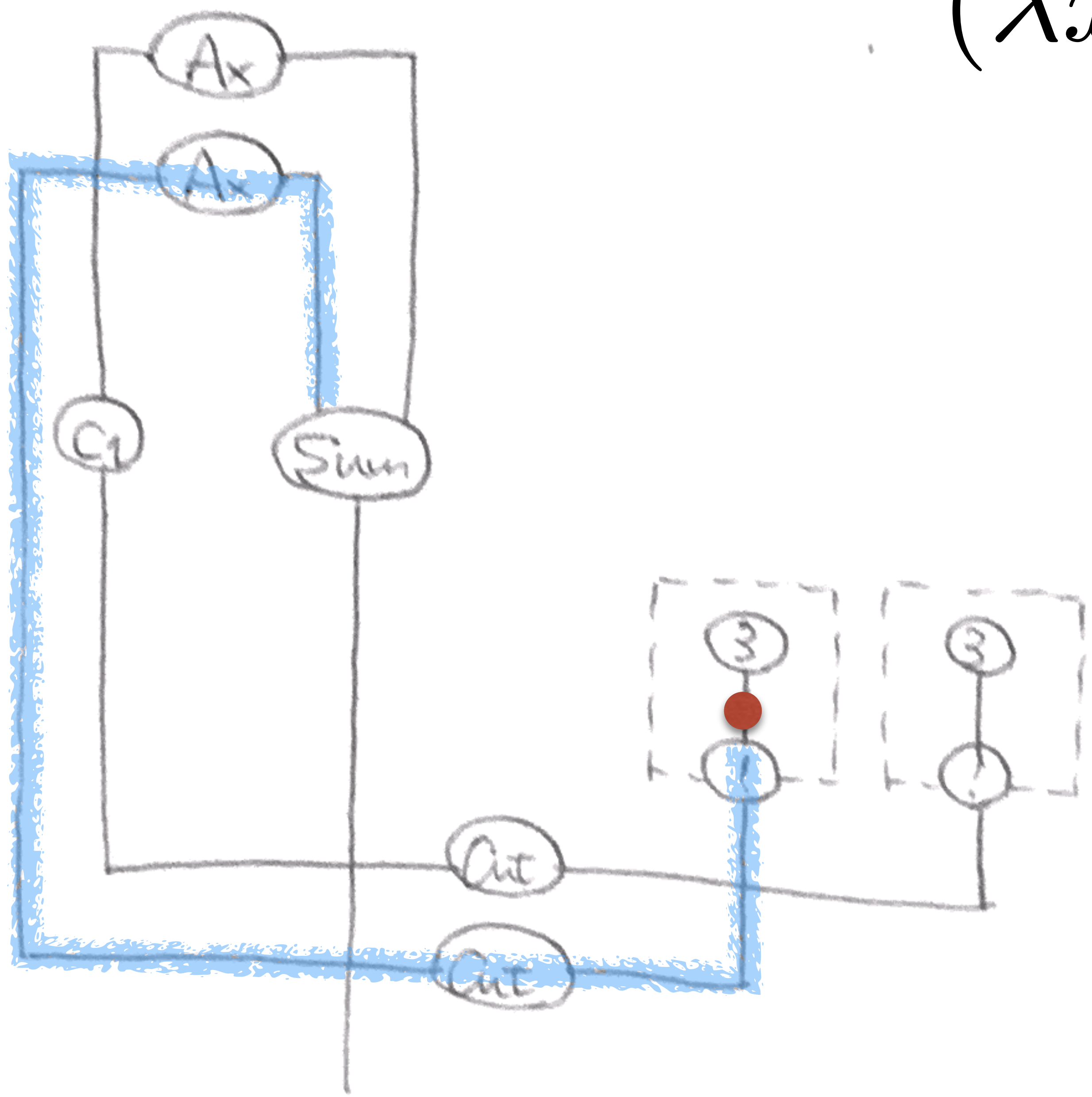
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



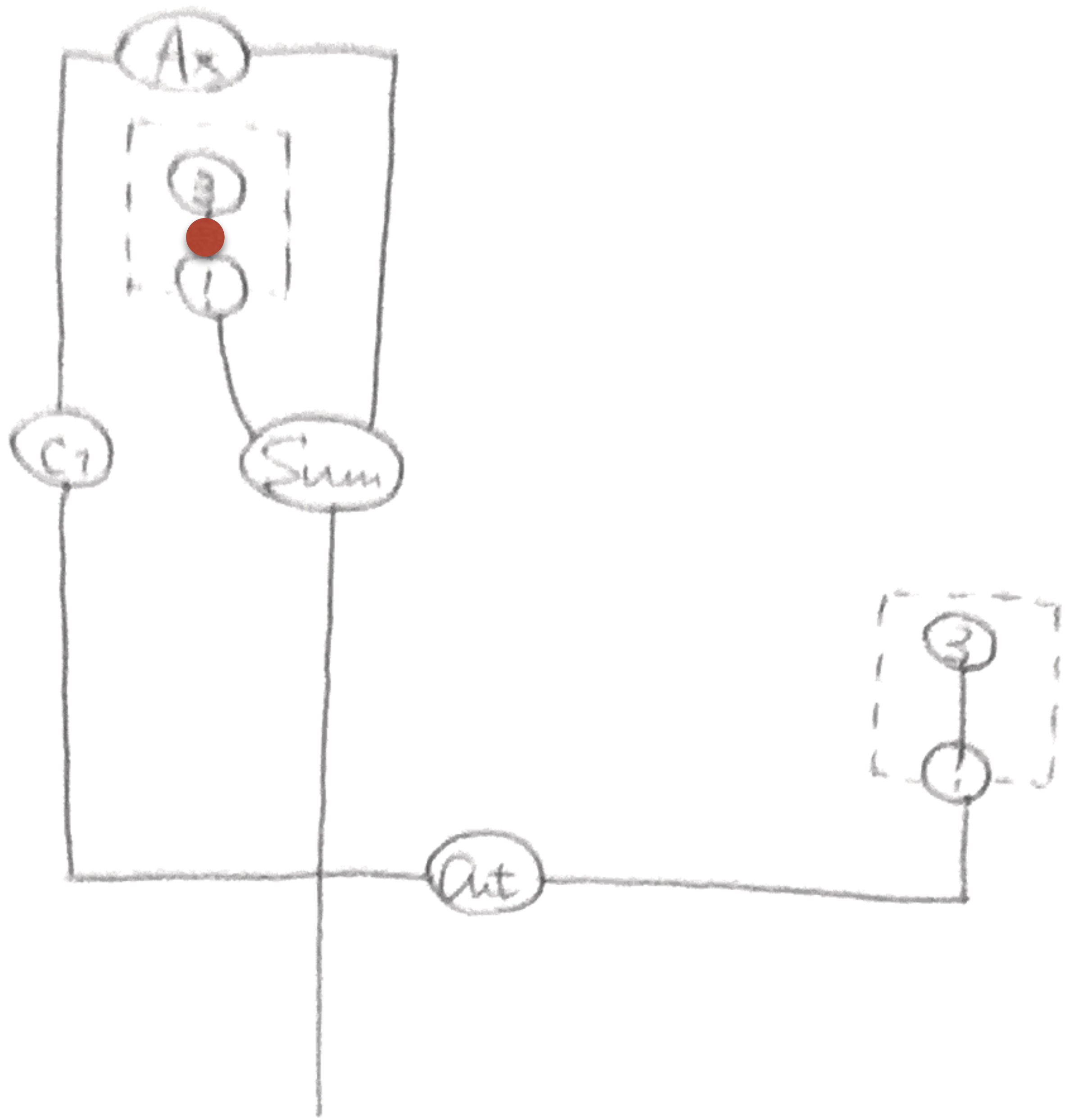
Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$



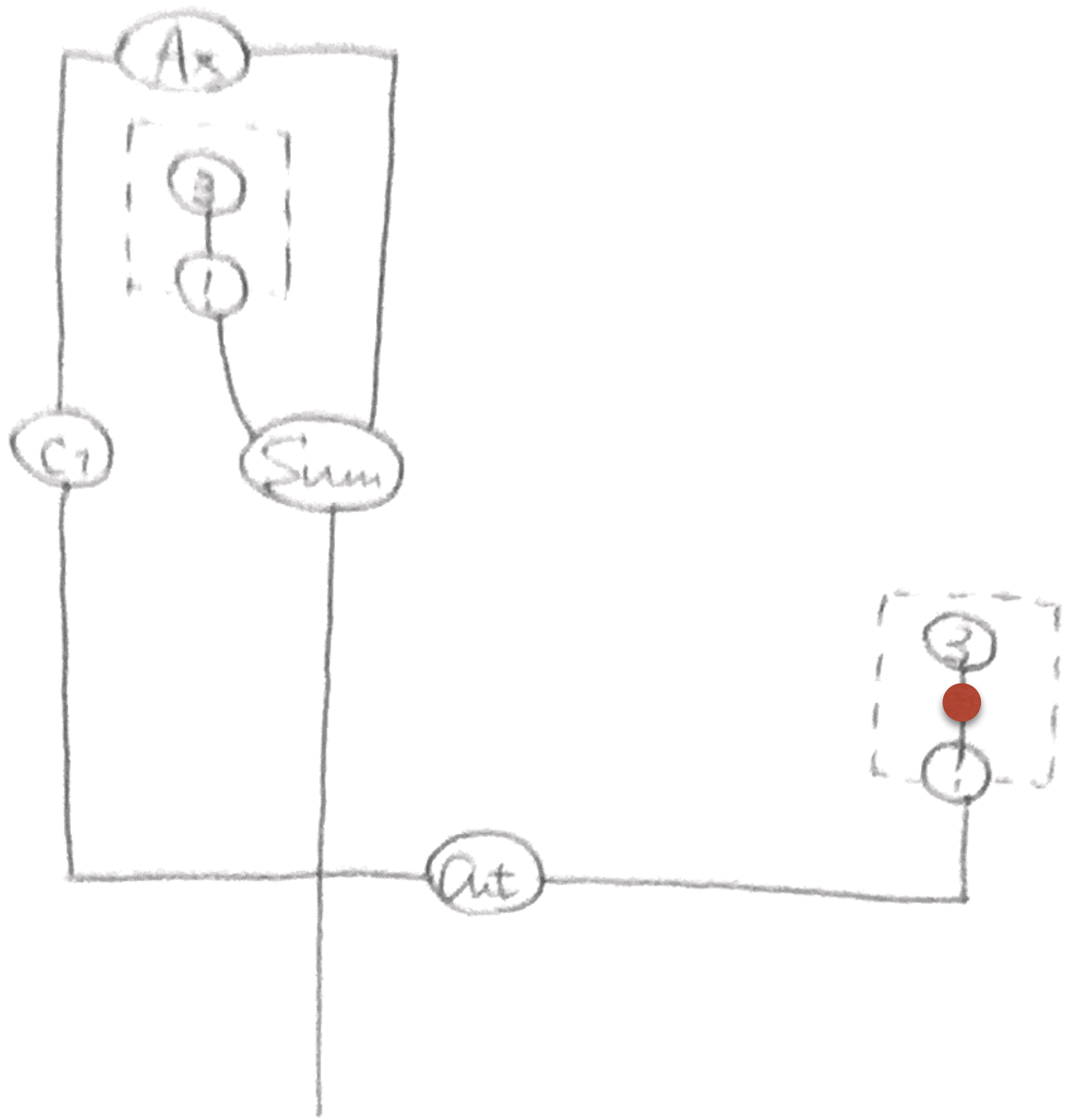
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.x + x)\,(1+2)$$



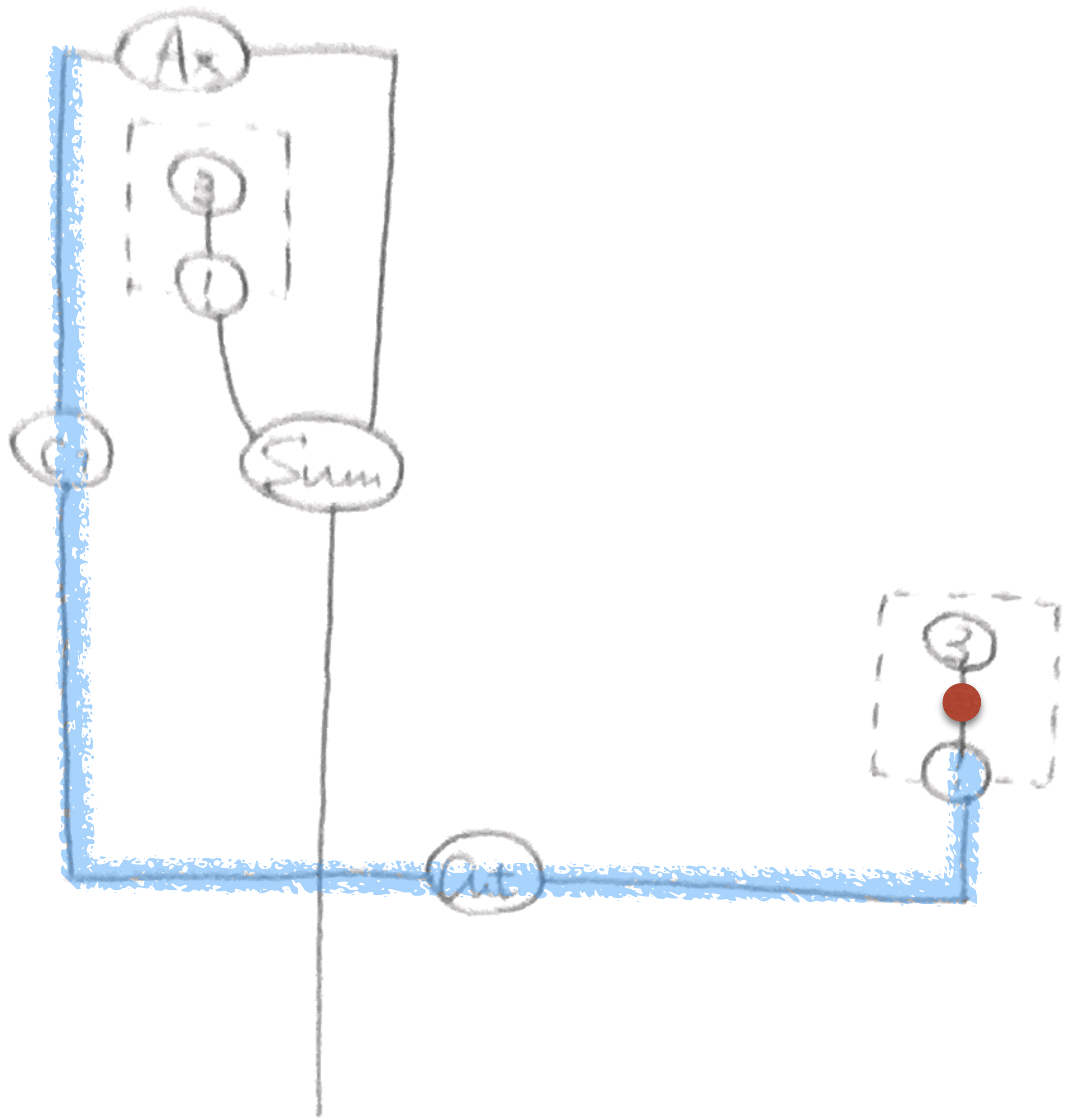
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



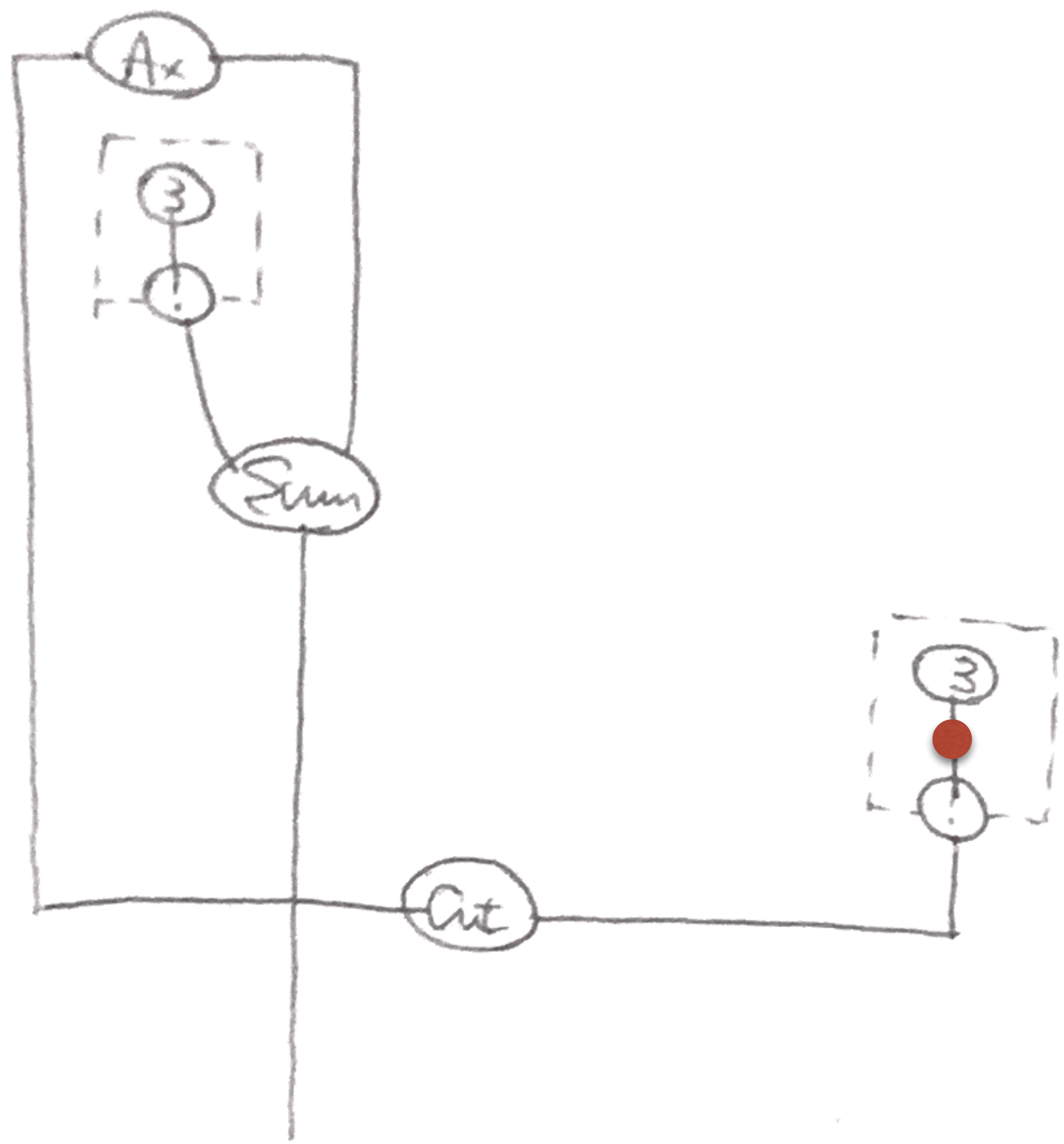
Avoid re-evaluation: dynamic rewrite

$$(\lambda x. x + x)\,(1 + 2)$$



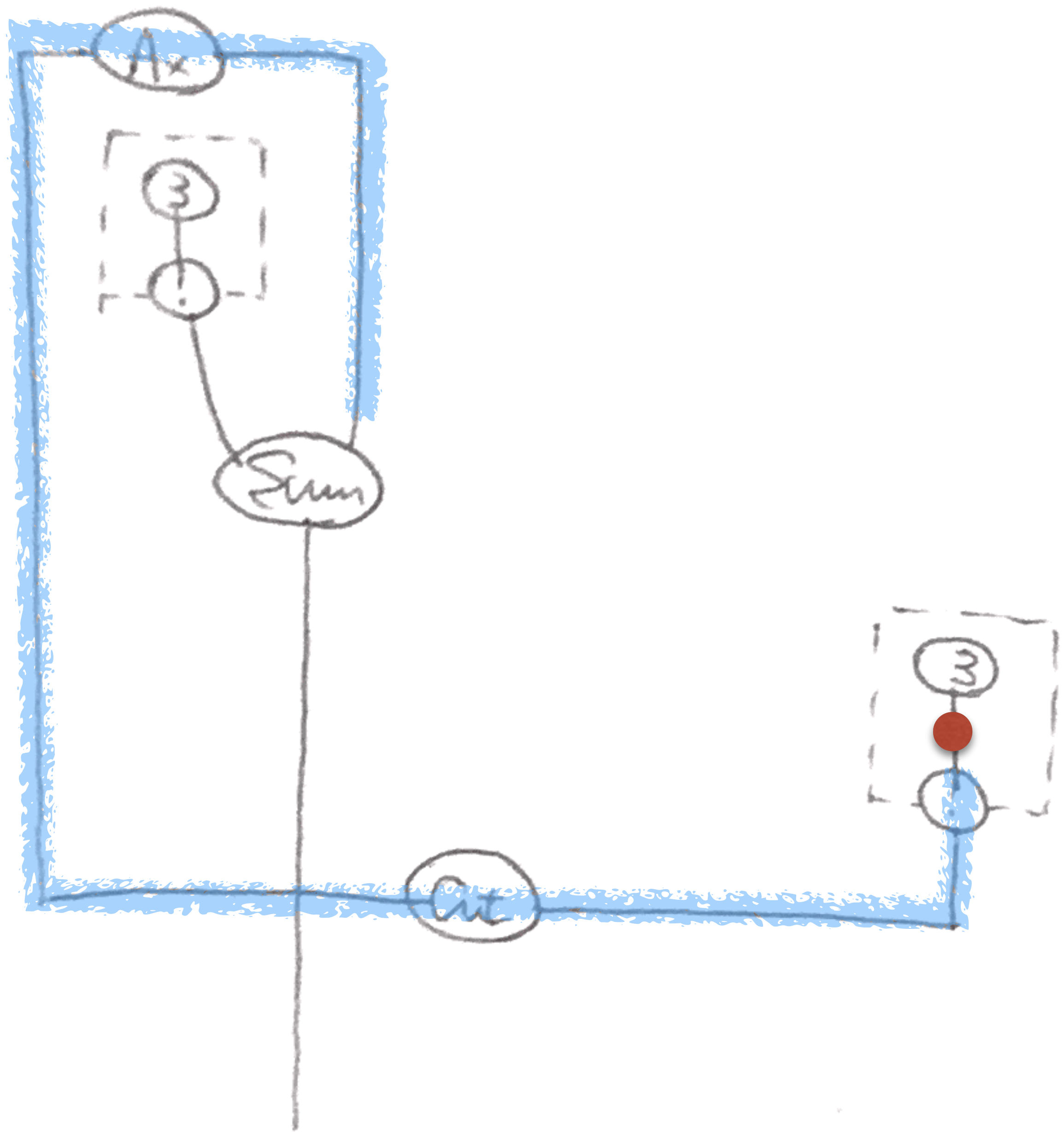
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



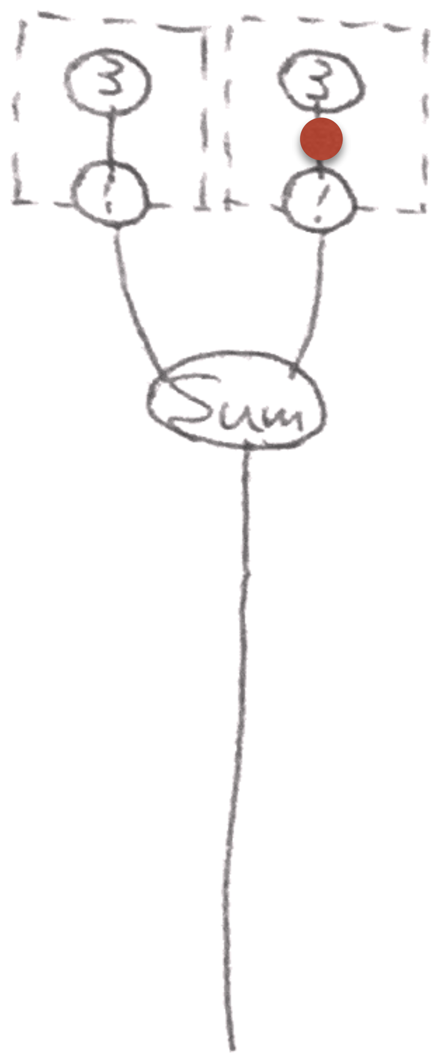
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



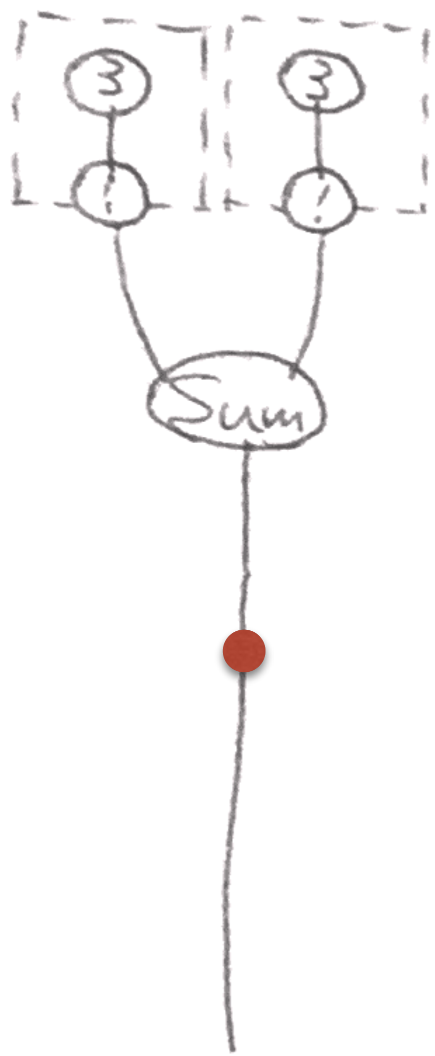
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



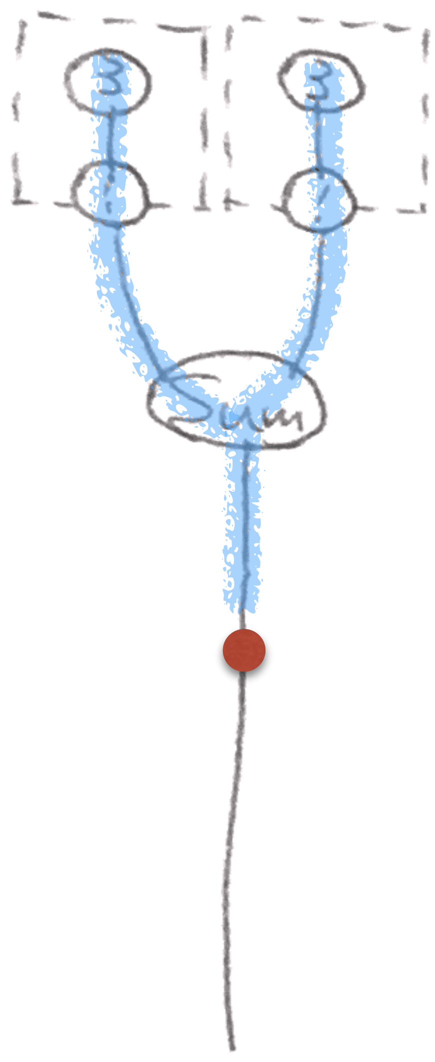
Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$



Avoid re-evaluation: dynamic rewrite

$$(\lambda x.\, x + x)\,(1 + 2)$$

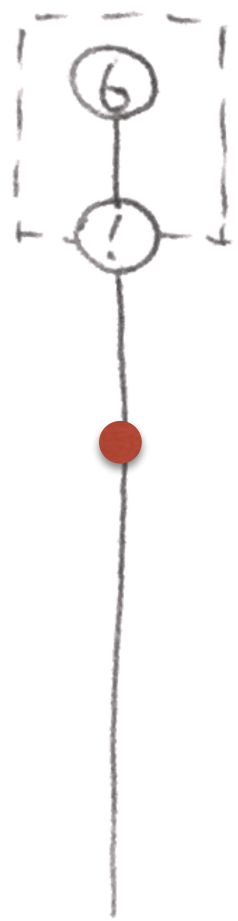


Avoid re-evaluation: dynamic rewrite

$$(\lambda x\,.\,x + x)\,(1 + 2)$$

Avoid re-evaluation: dynamic rewrite

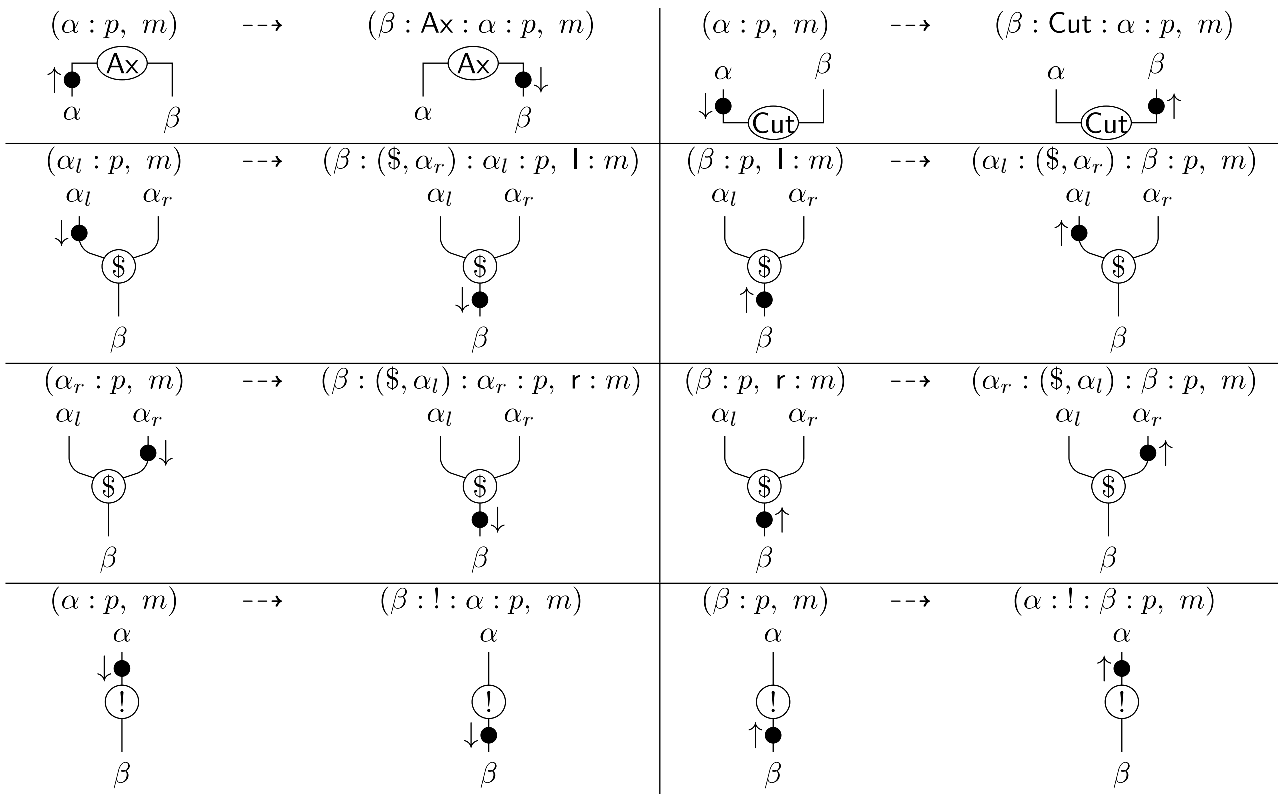
$$(\lambda x\,.\,x + x)\,(1 + 2)$$

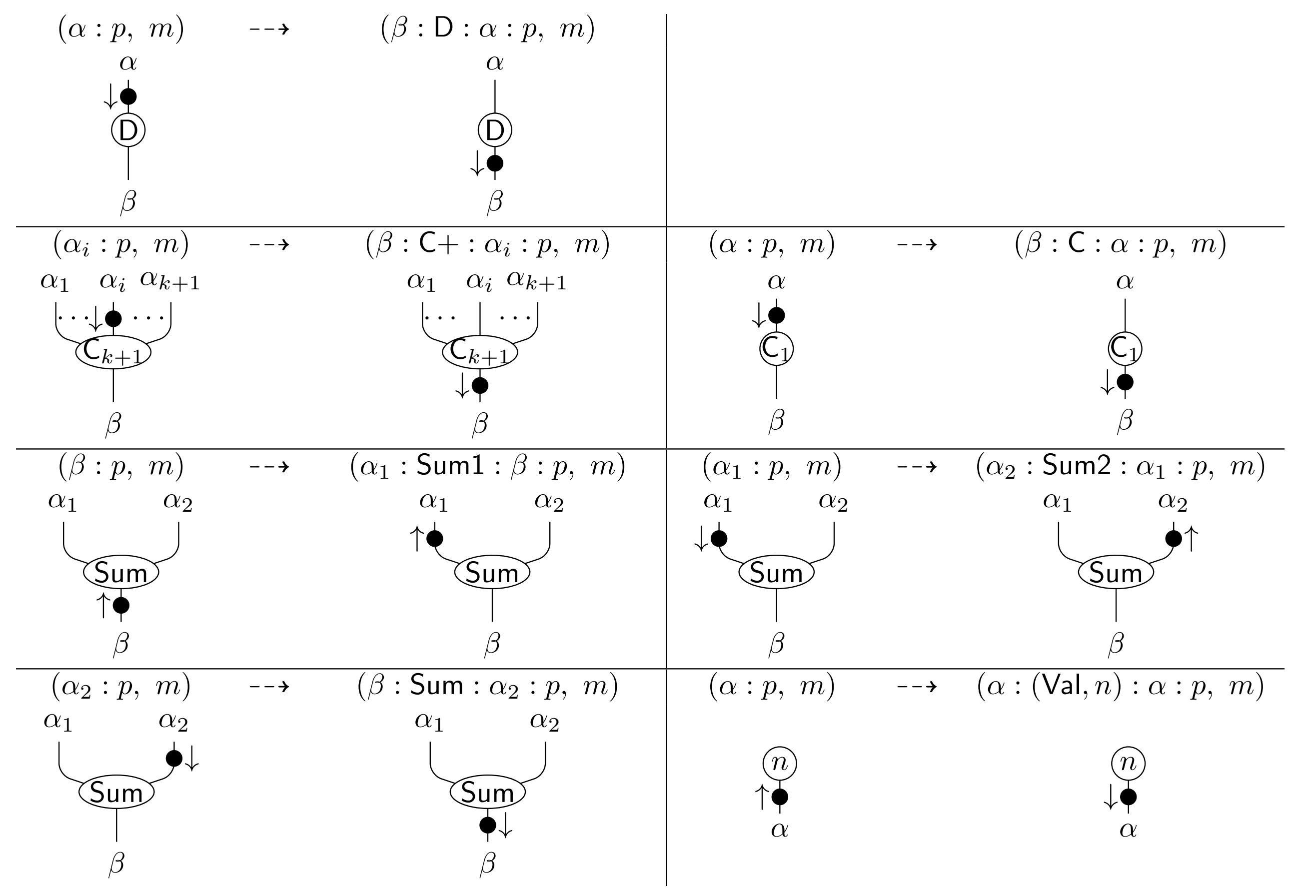


Muroya (U. Birmingham)

Dynamic GoI machine

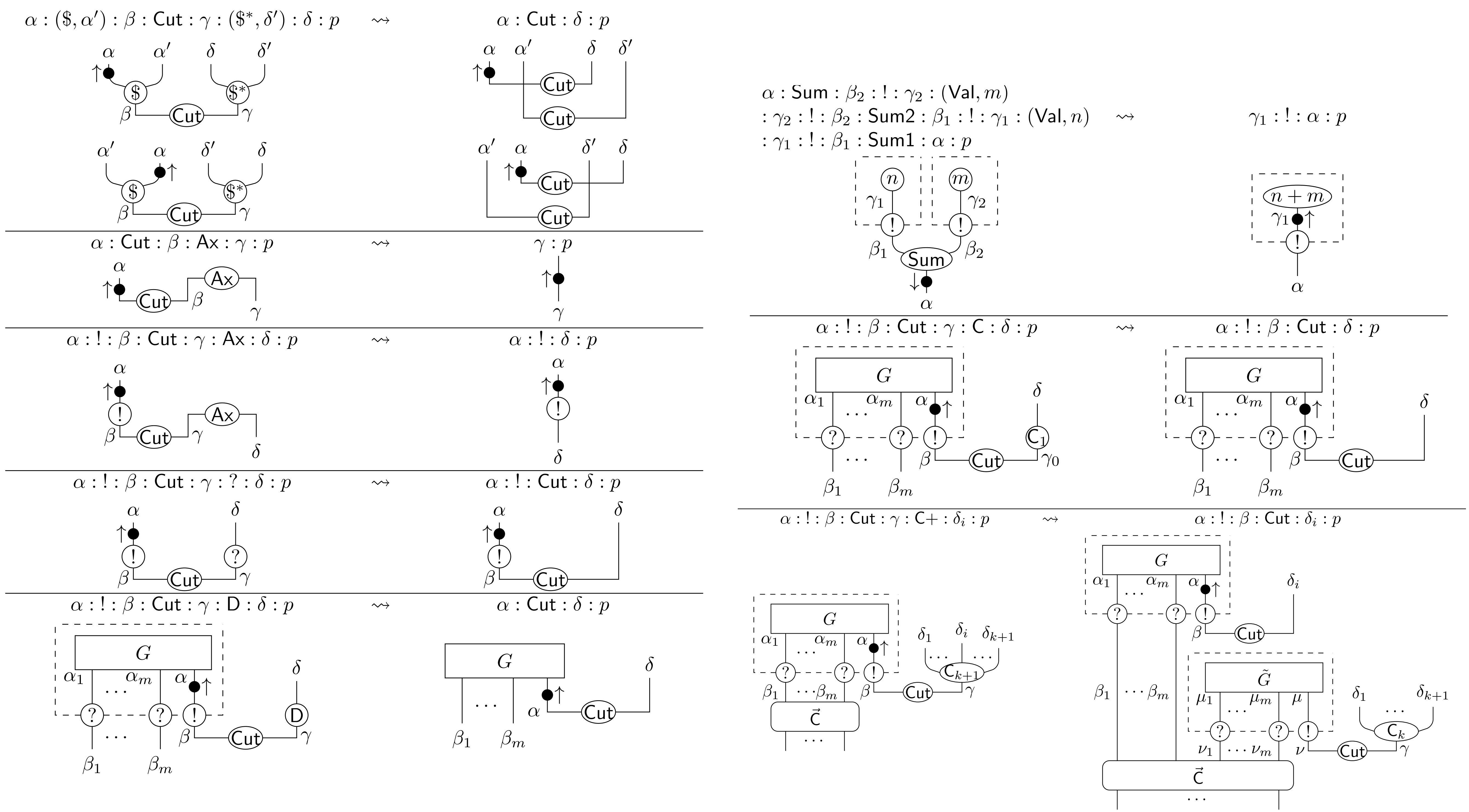
"move" transition $((G, \ell_e, \ell_b), p, d, m) \dashrightarrow ((G, \ell_e, \ell_b), p', d', m')$





Dynamic GoI machine

"rewrite" transition $((G, \ell_e, \ell_b), p, d, m) \rightsquigarrow ((G', \ell'_e, \ell'_b), p', d', m)$



Muroya (U. Birmingham)

“Linear-cost” simulation

- Dynamic GoI machine $\dashrightarrow \overset{\text{def.}}{\iff} \begin{cases} \rightsquigarrow & \text{if a rewrite is possible} \\ \dashrightarrow & \text{if no rewrite but a move is possible} \end{cases}$
- Call-by-need storeless abstract machine [Danvy & Zerny ’13]

$$\begin{aligned}
\langle V, E\rangle_t &\to_{\text{need}} \langle E, V\rangle_c \\
\langle M\,N, E\rangle_t &\to_{\text{need}} \langle M, E[[\,]\,N]\rangle_t \\
\langle M+N, E\rangle_t &\to_{\text{need}} \langle M, E[[\,]+N]\rangle_t \\
\langle x, E_1[\texttt{let}\ x = N\ \texttt{in}\ E_2]\rangle_t &\to_{\text{need}} \langle N, E_1[\texttt{let}\ x := [\,]\ \texttt{in}\ E_2[x]]\rangle_t
\end{aligned}$$

$$\begin{aligned}
\langle [\,], A[V]\rangle_c &\to_{\text{need}} \langle A[V]\rangle_a \\
\langle E[[\,]\,N], A[\lambda x.\,M]\rangle_c &\to_{\text{need}} \langle M[x'/x], E[A[\texttt{let}\ x' = N\ \texttt{in}\ [\,]]]\rangle_t \\
\langle E[[\,]+N], A[\underline{n}]\rangle_c &\to_{\text{need}} \langle N, E[A[\underline{n}+[\,]]]\rangle_t \\
\langle E[\underline{n}+[\,]], A[\underline{m}]\rangle_c &\to_{\text{need}} \langle E, A[\underline{n+m}]\rangle_c \\
\langle E[\texttt{let}\ x = N\ \texttt{in}\ [\,]], A[V]\rangle_c &\to_{\text{need}} \langle E, \texttt{let}\ x = N\ \texttt{in}\ A[V]\rangle_c \\
\langle E[\texttt{let}\ x := [\,]\ \texttt{in}\ E'[x]], A[V]\rangle_c &\to_{\text{need}} \langle E[A[\texttt{let}\ x = V\ \texttt{in}\ E']], V\rangle_c
\end{aligned}$$

Theorem A.1. There exists a binary relation $\ddagger$ that satisfies

$$c \xrightarrow{k}_{\text{need}} c' \ \wedge\ c \ddagger (G,p,d,m)$$
$$\implies\ (G,p,d,m) \to \overset{\mathcal{O}(k)}{\cdots} \to (G',p',d',m')\ \wedge\ c' \ddagger (G',p',d',m')\ .$$

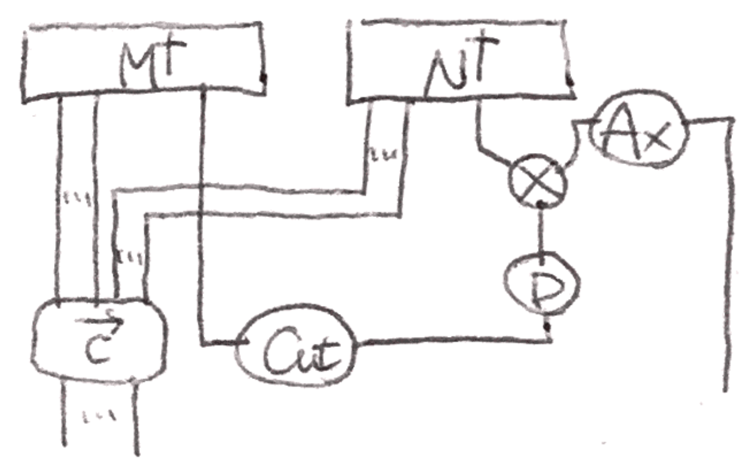
“Linear-cost” simulation

Theorem A.1. There exists a binary relation $\ddagger$ that satisfies

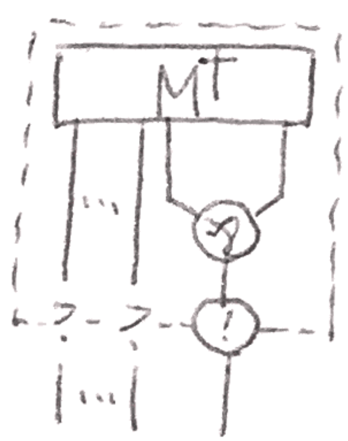
$$c \xrightarrow{k}_{\text{need}} c' \ \wedge \ c \ddagger (G, p, d, m)$$
$$\implies \ (G, p, d, m) \overset{\mathcal{O}(k)}{\twoheadrightarrow \cdots \twoheadrightarrow} (G', p', d', m') \ \wedge \ c' \ddagger (G', p', d', m') \ .$$

$x^{\dagger} :=$

$(M\,N)^{\dagger} :=$



$(\lambda x.\, M)^{\dagger} :=$



GoI machine [Danos & Regnier '99] [Mackie '95]

call-by-name | call-by-need | call-by-value | effects

	avoid re-evaluation	force evaluation of arguments	track each copy of terms
CPS transform.	?	✔	?
memory	△		✔
parallelism & sync.	△	✔	
dynamic jump	✔	✔	
dynamic rewrite	✔		☺
checkpoint		✔	

Schöpp

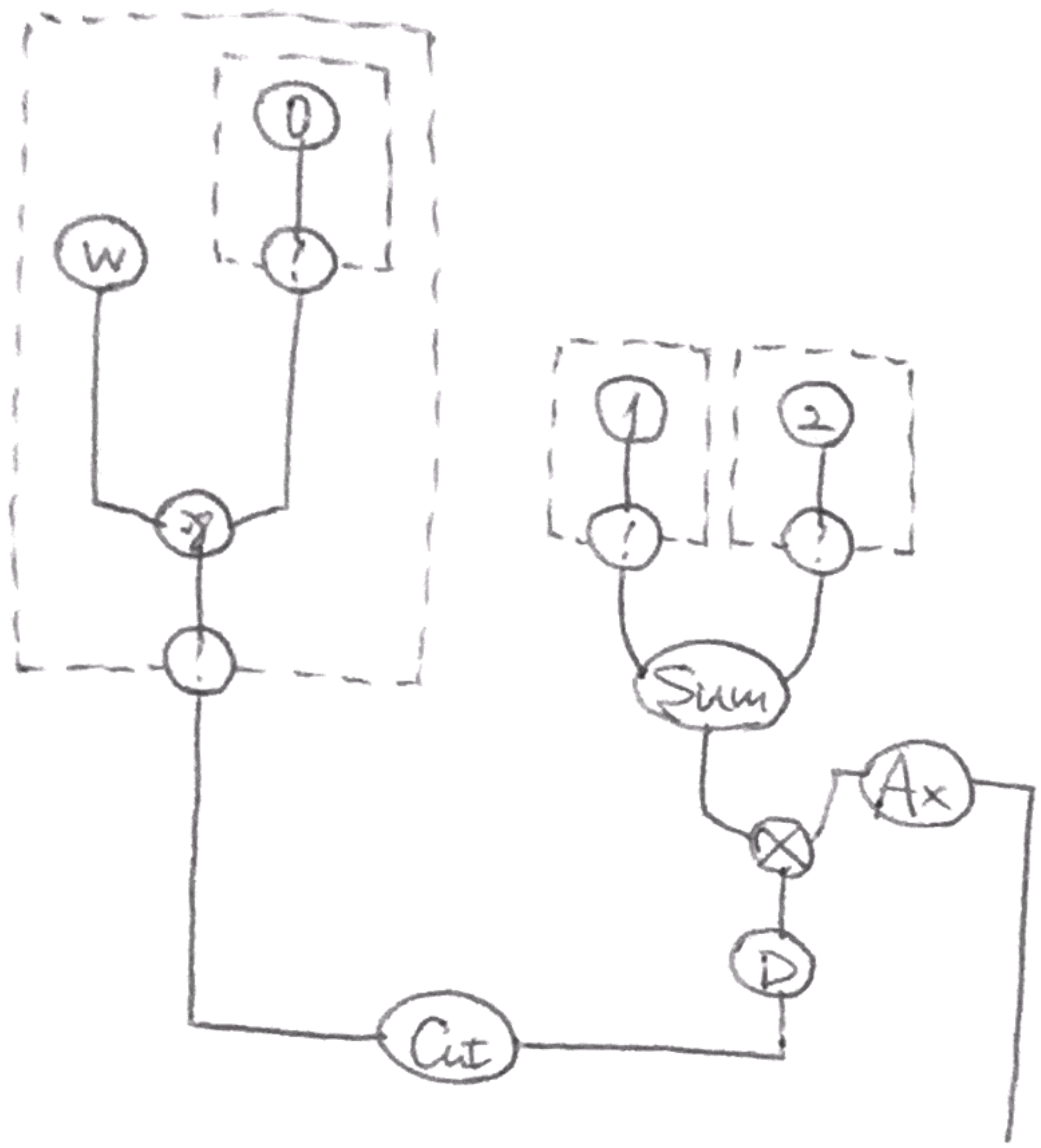
Hoshino+

Dal Lago+

Fernández+

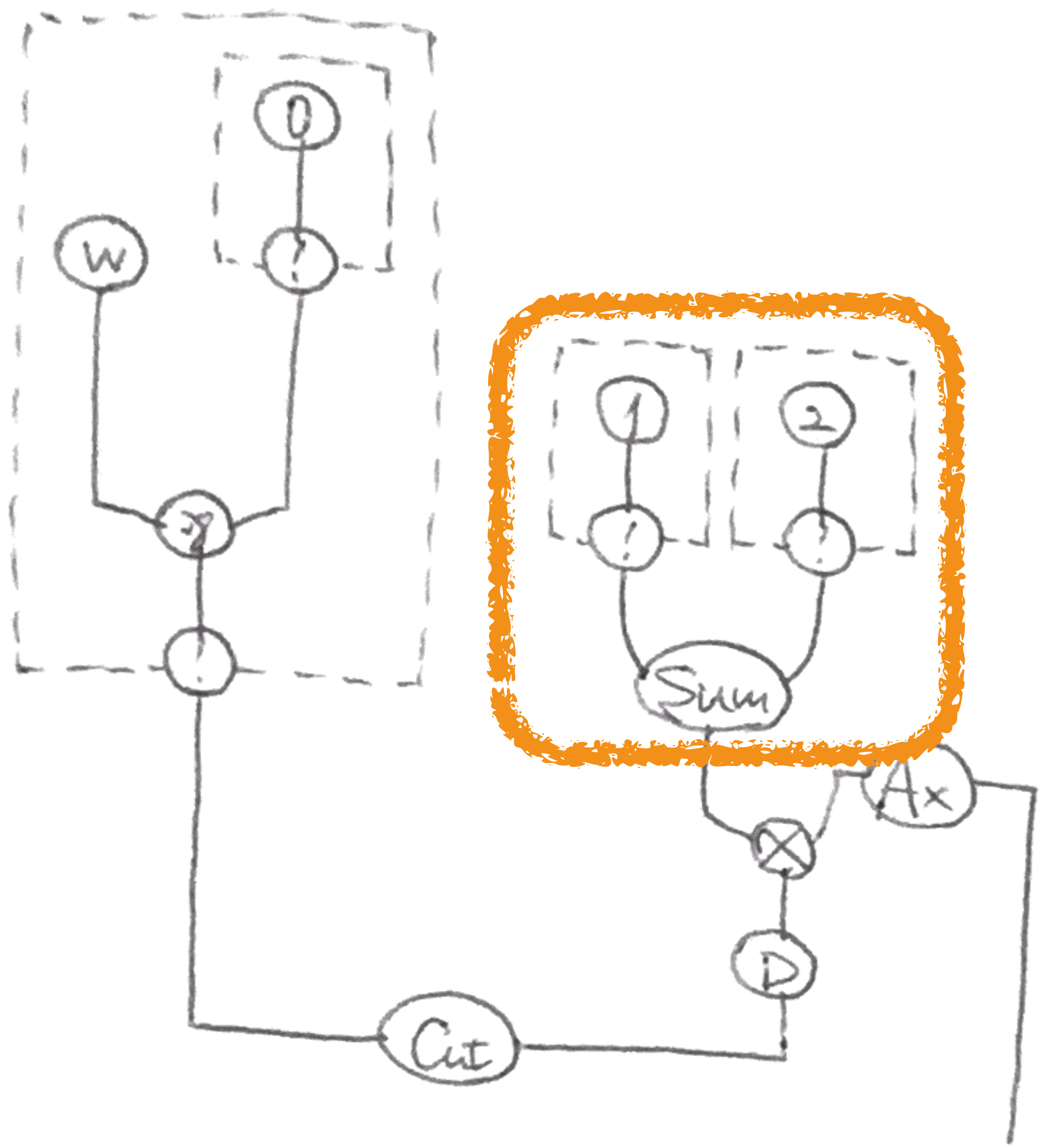
Force evaluation of arguments

$$(\lambda x.\,0)\,(1+2)$$



Force evaluation of arguments

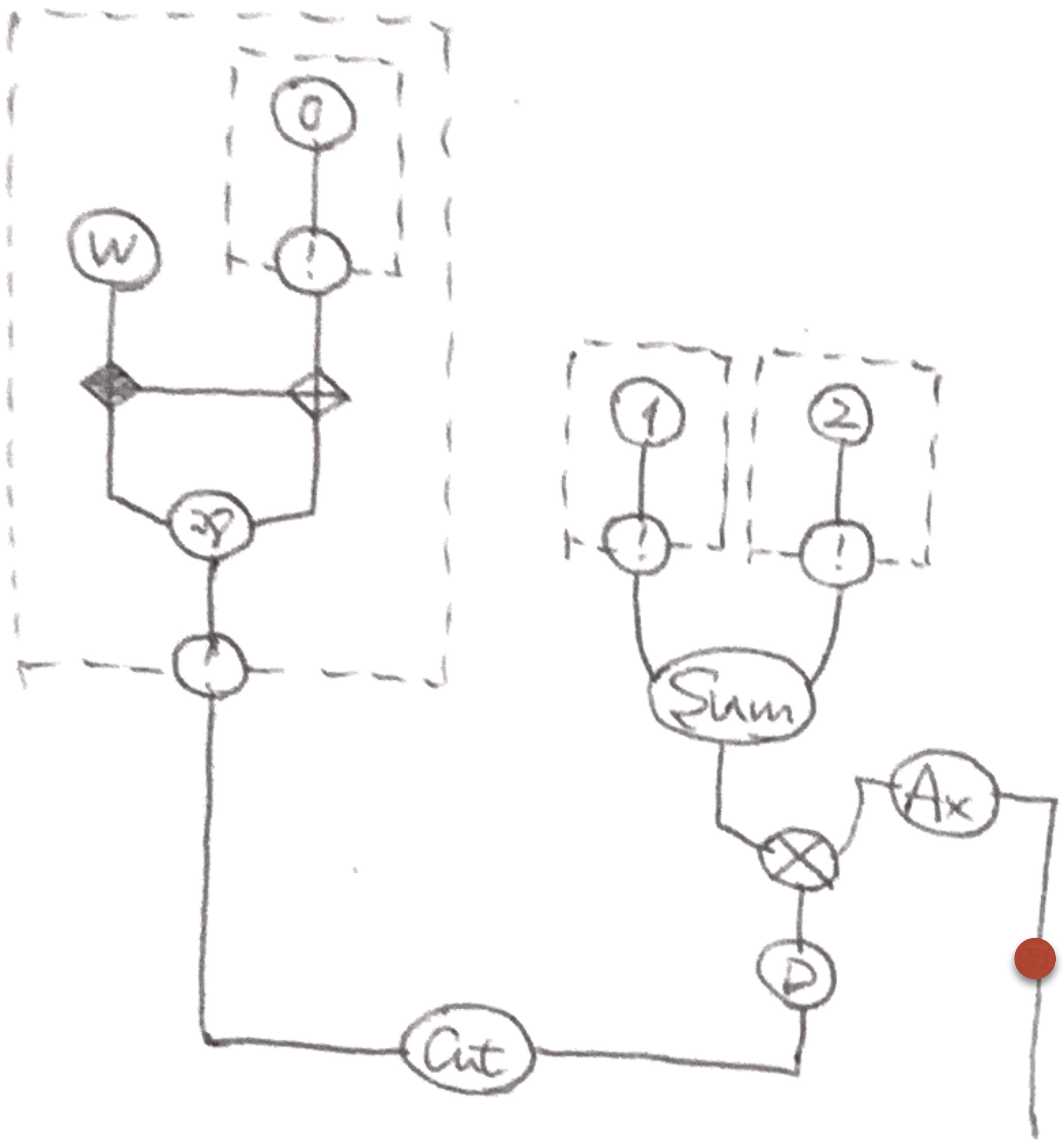
$$(\lambda x\,.\,0)\,(1+2)$$



Muroya (U. Birmingham)

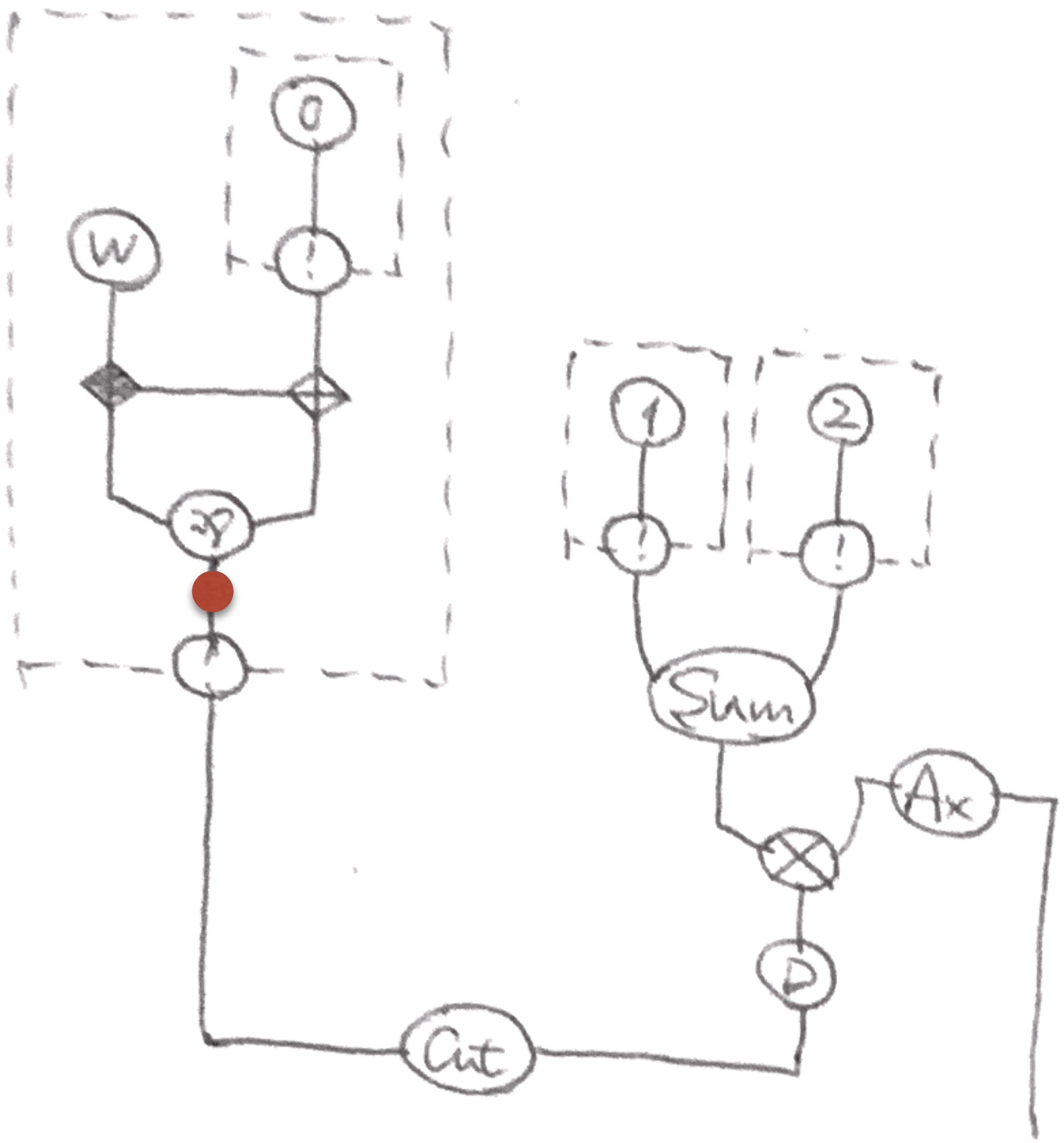
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1+2)$$



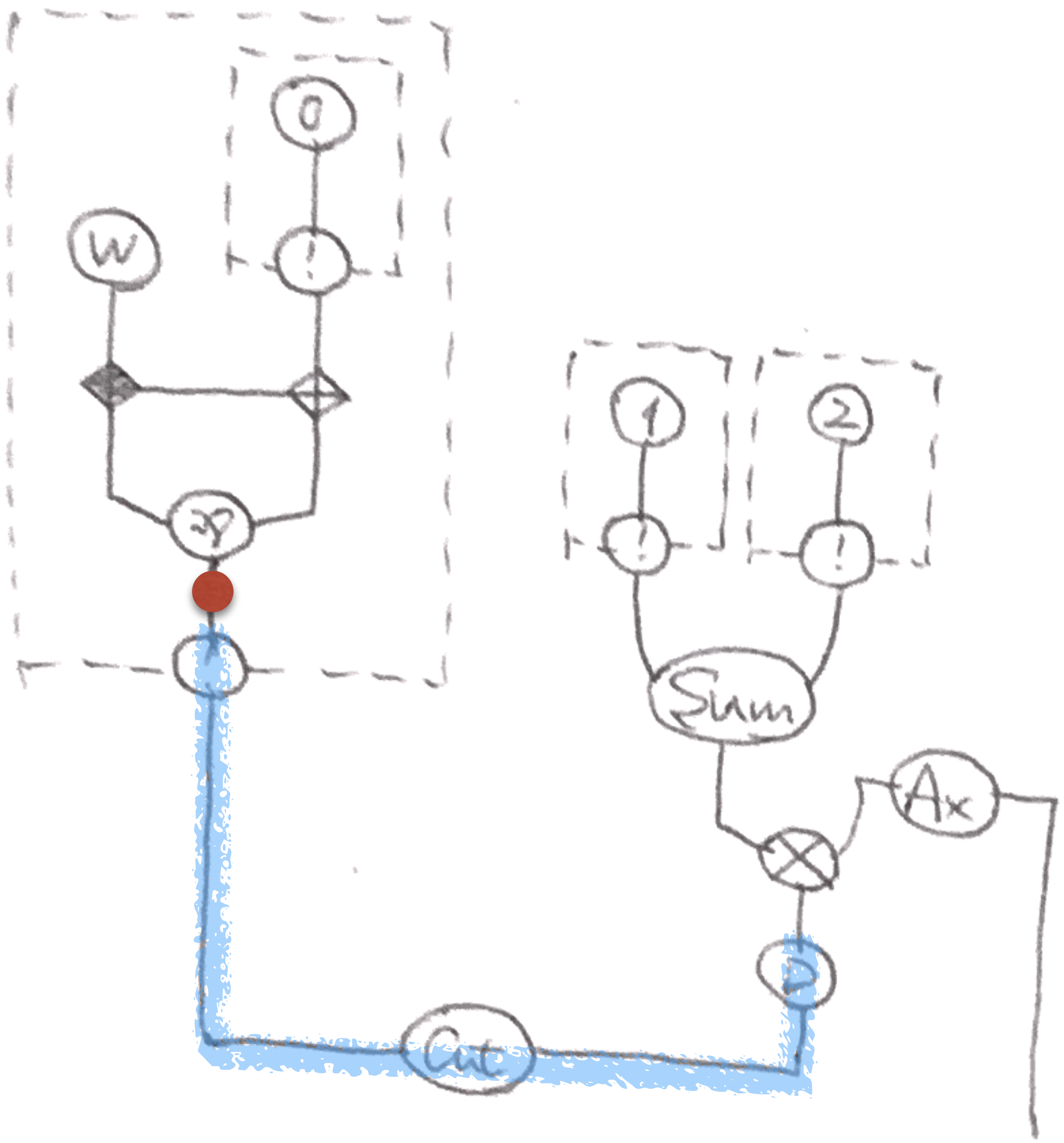
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1+2)$$



Force evaluation of arguments: checkpoint

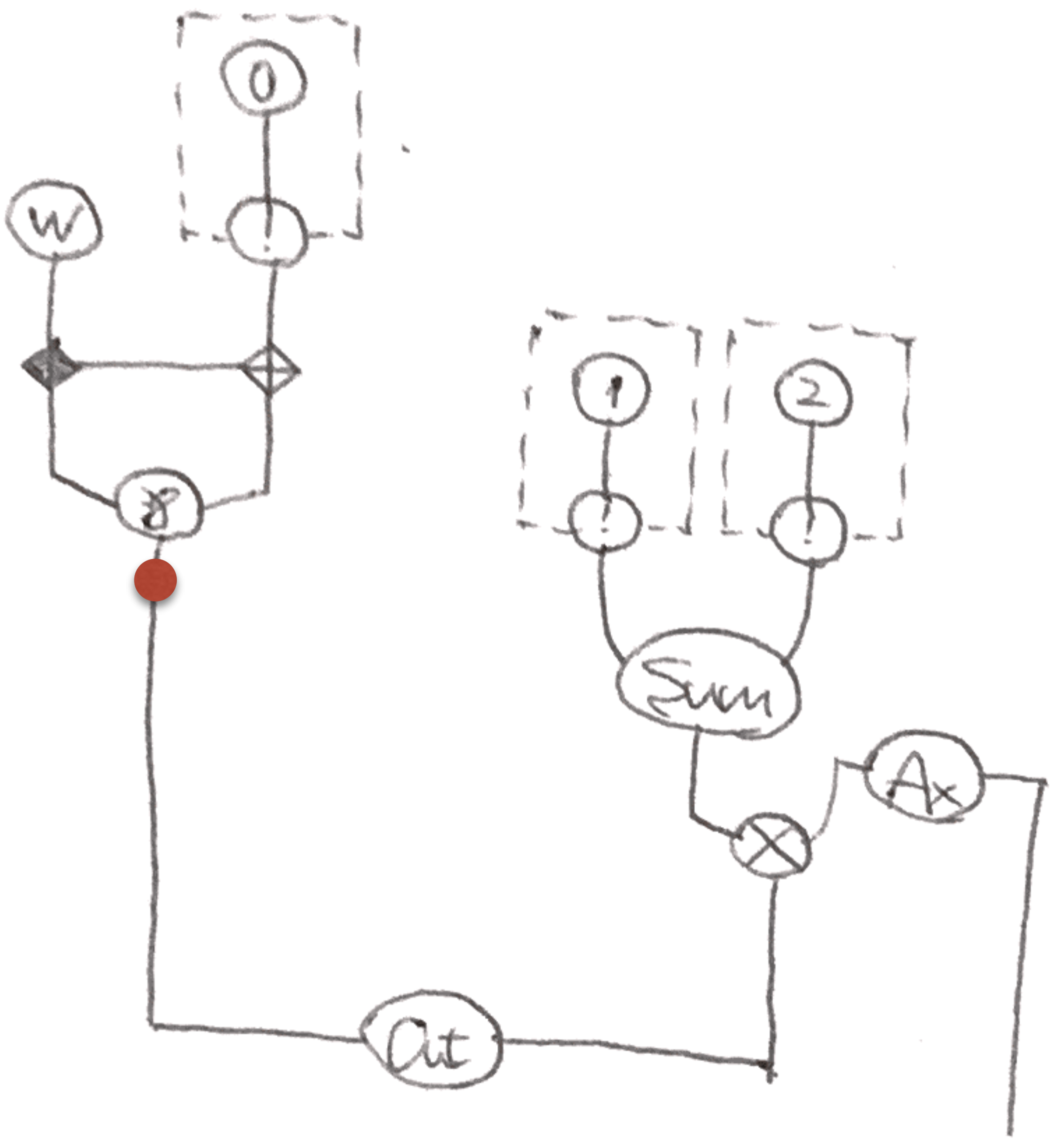
$$(\lambda x\,.\,0)\,(1+2)$$



Muroya (U. Birmingham)

Force evaluation of arguments: checkpoint

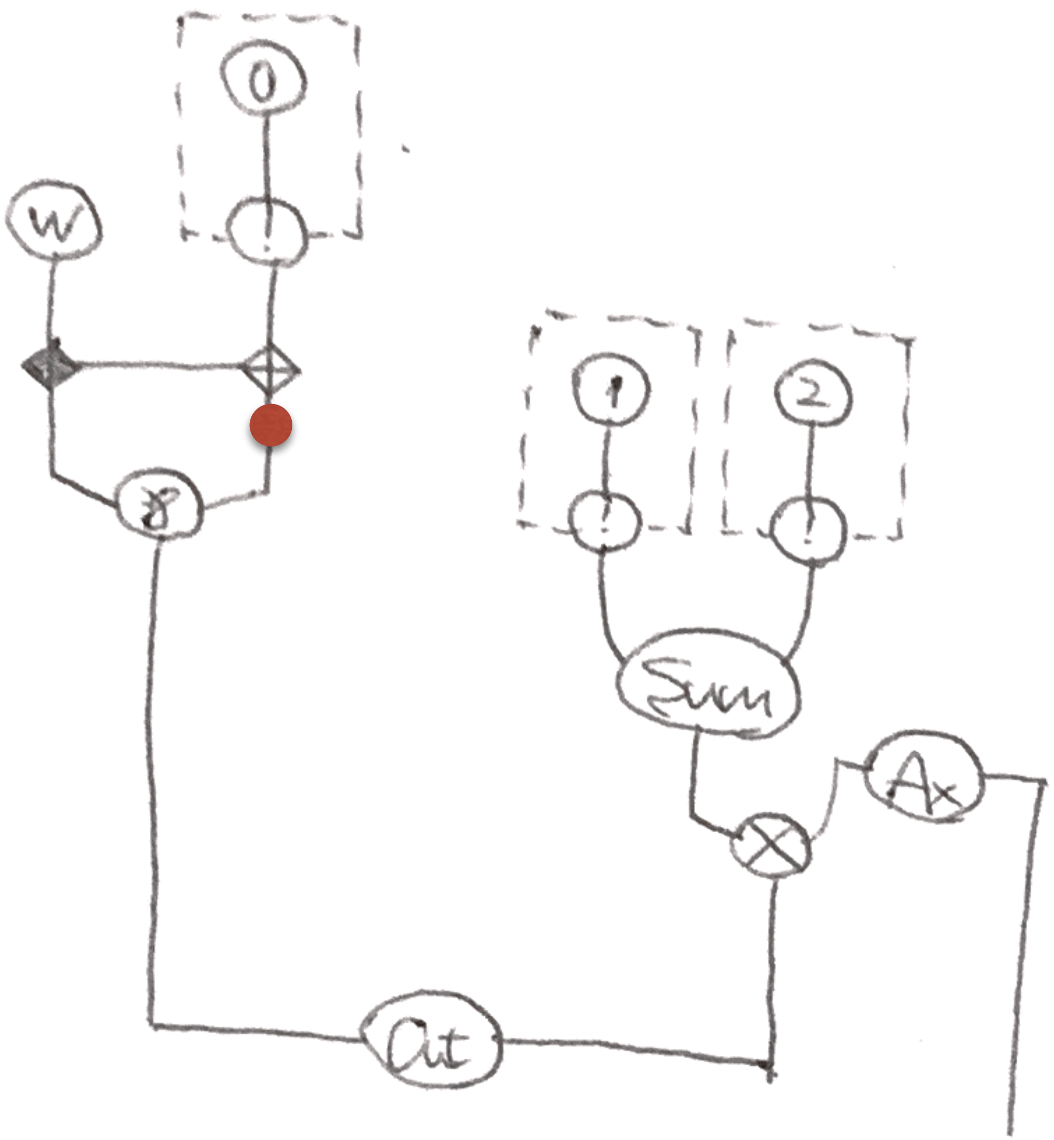
$$(\lambda x.\, 0)\,(1 + 2)$$



Muroya (U. Birmingham)

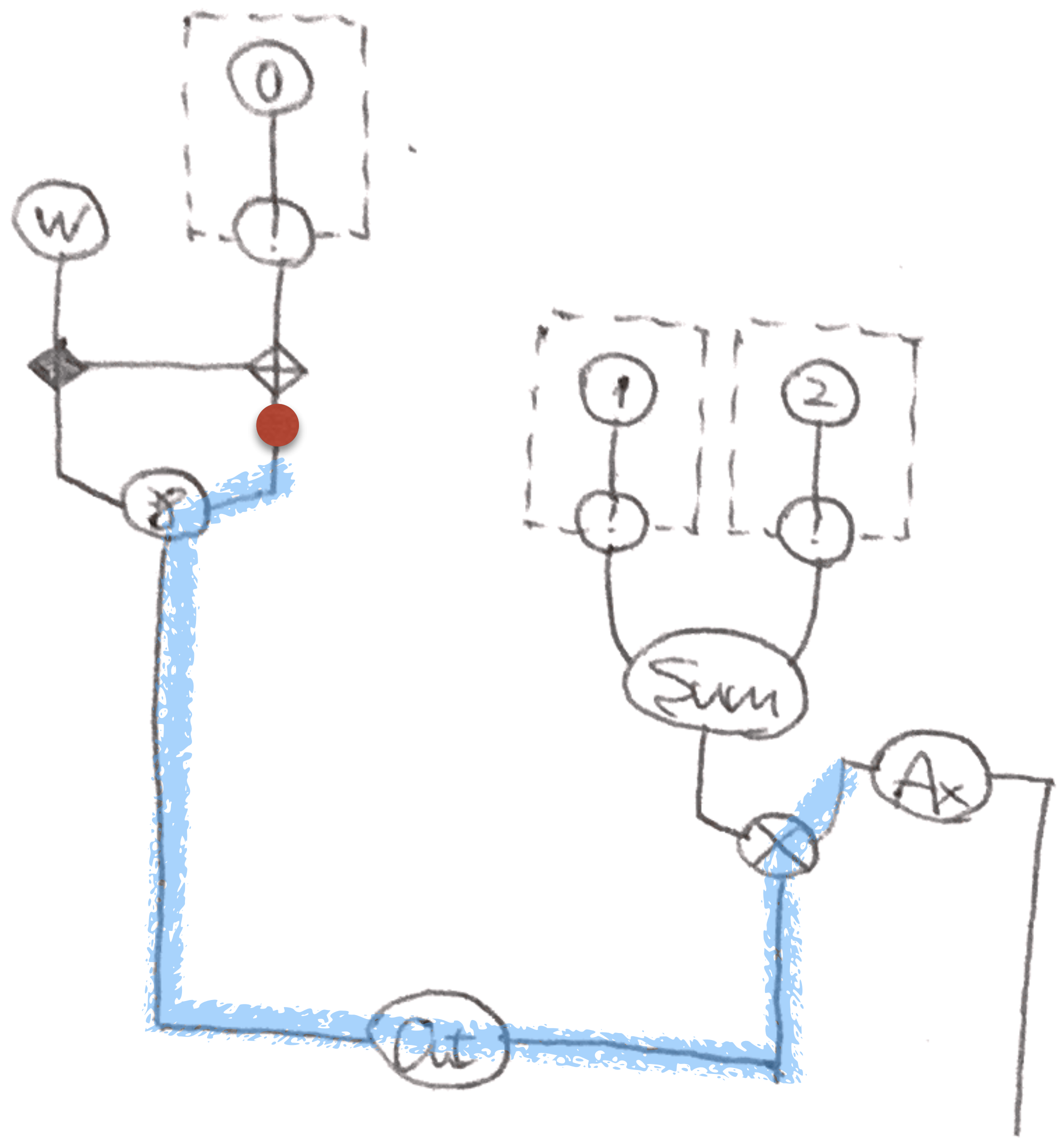
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\; (1 + 2)$$



Force evaluation of arguments: checkpoint

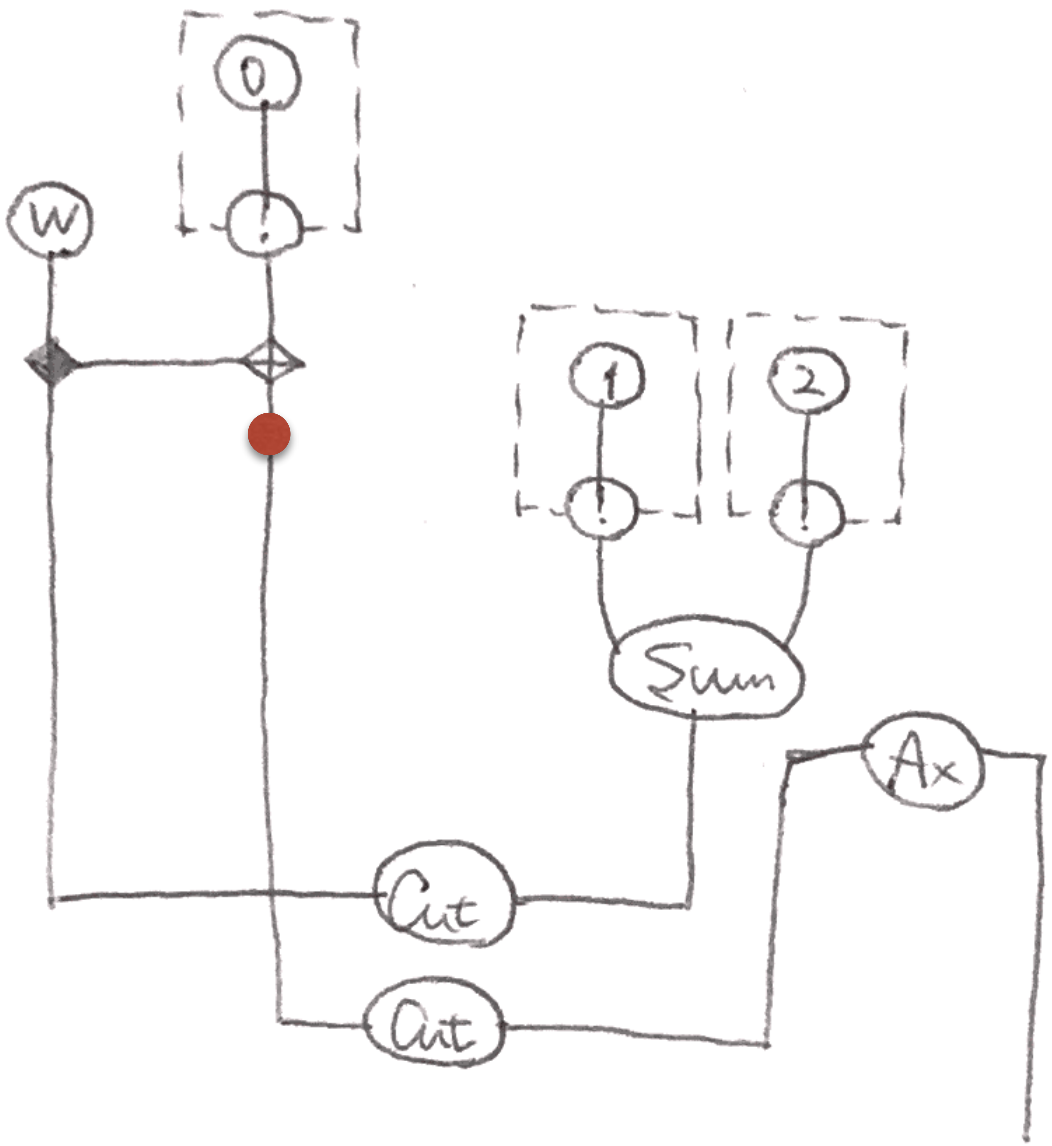
$$(\lambda x.\,0)\,(1+2)$$



Muroya (U. Birmingham)

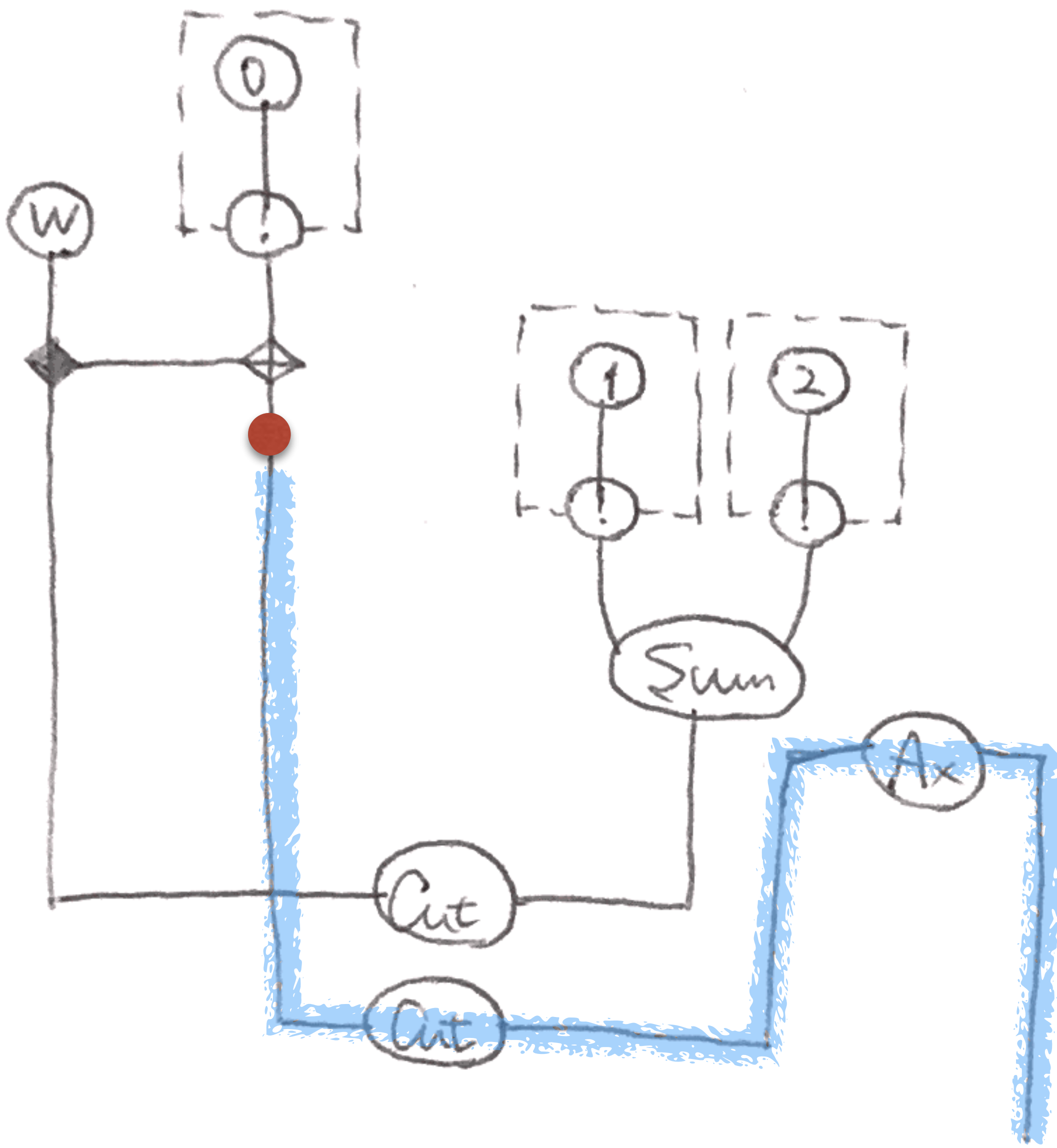
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1 + 2)$$



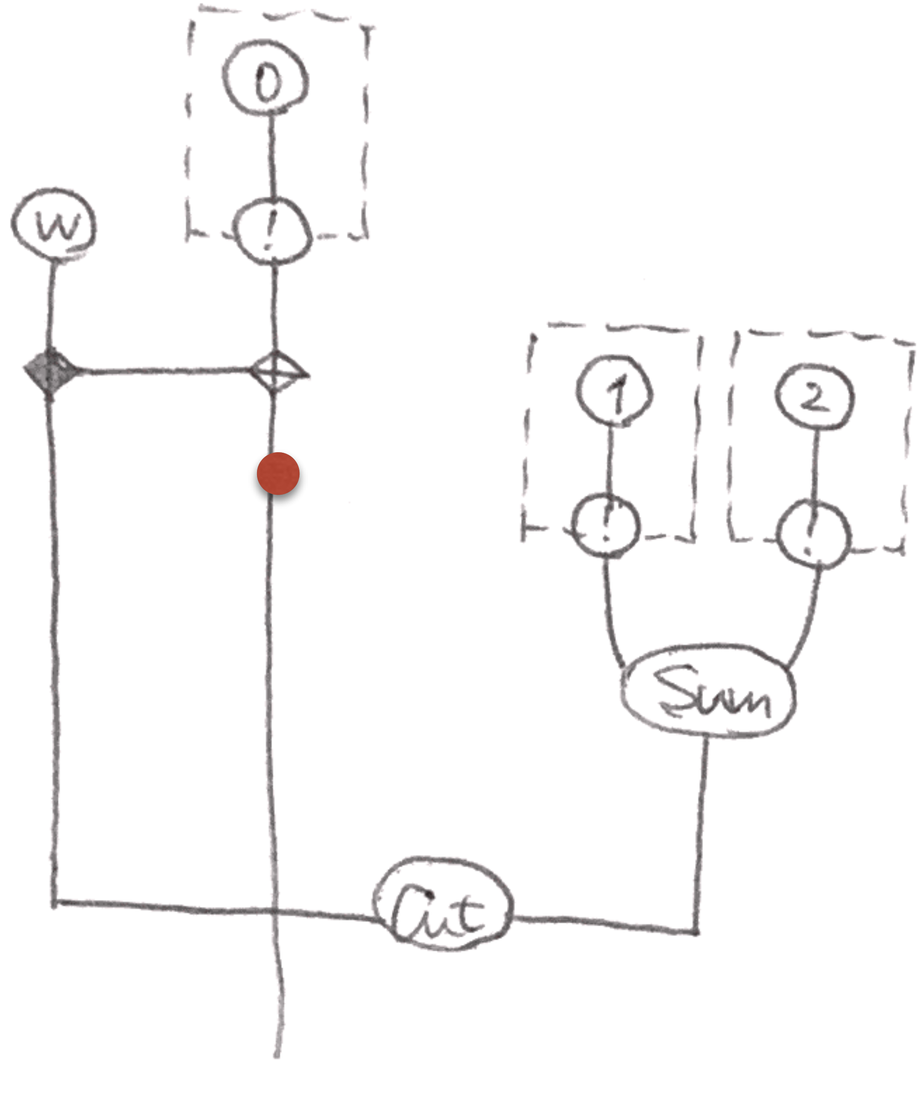
Force evaluation of arguments: checkpoint

$$(\lambda x\,.\,0)\,(1+2)$$



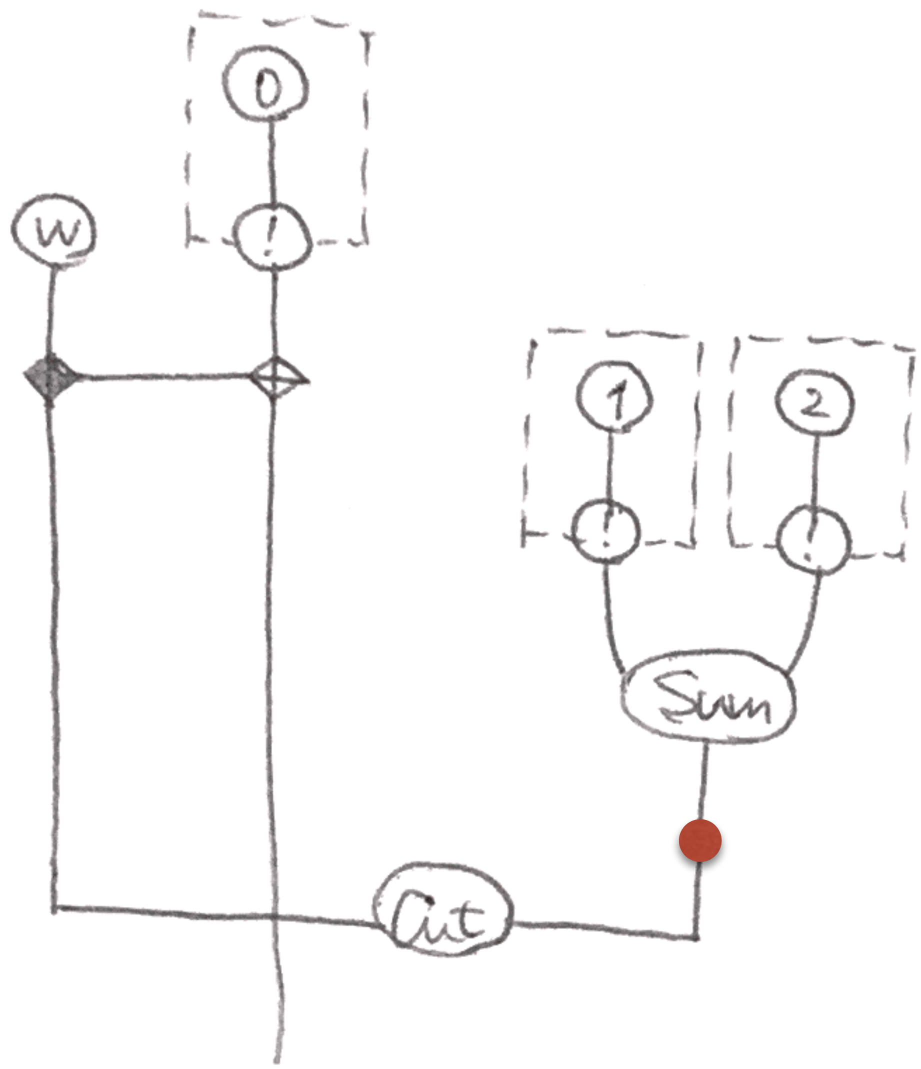
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\ (1 + 2)$$



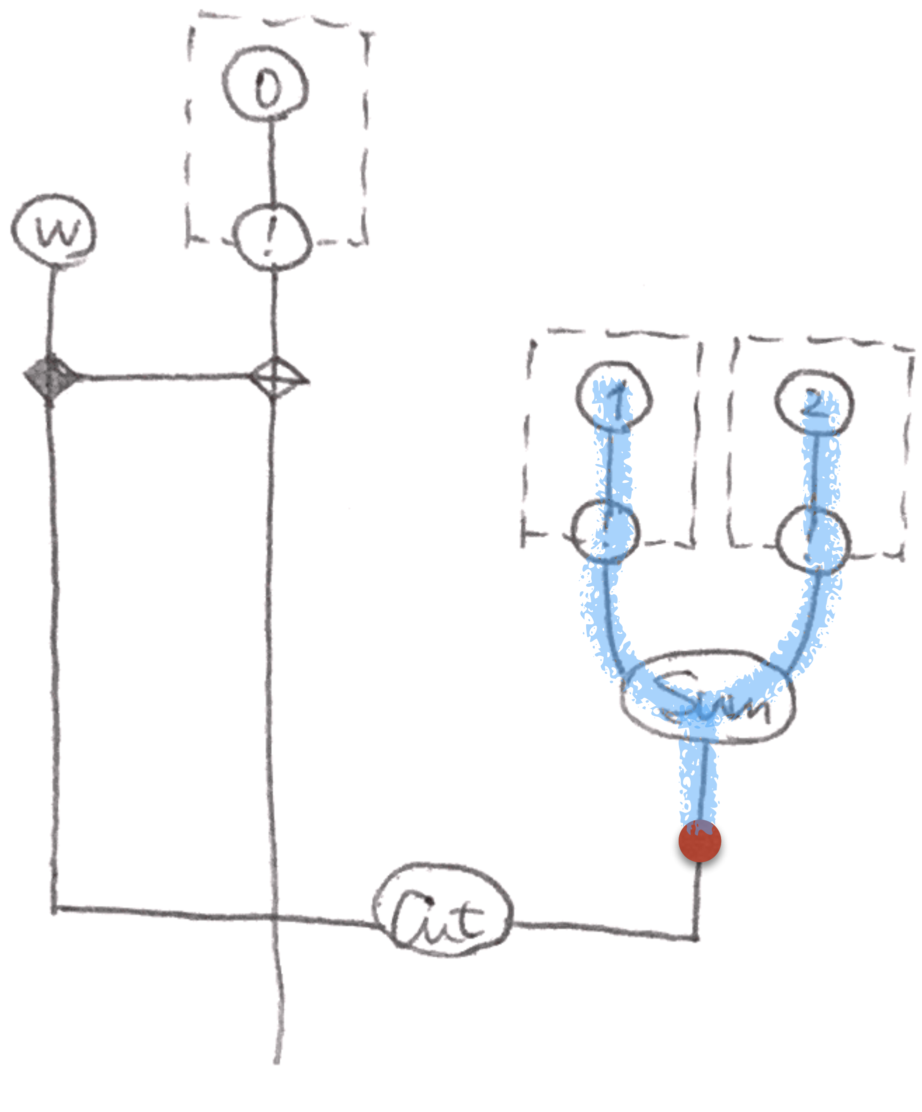
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1 + 2)$$



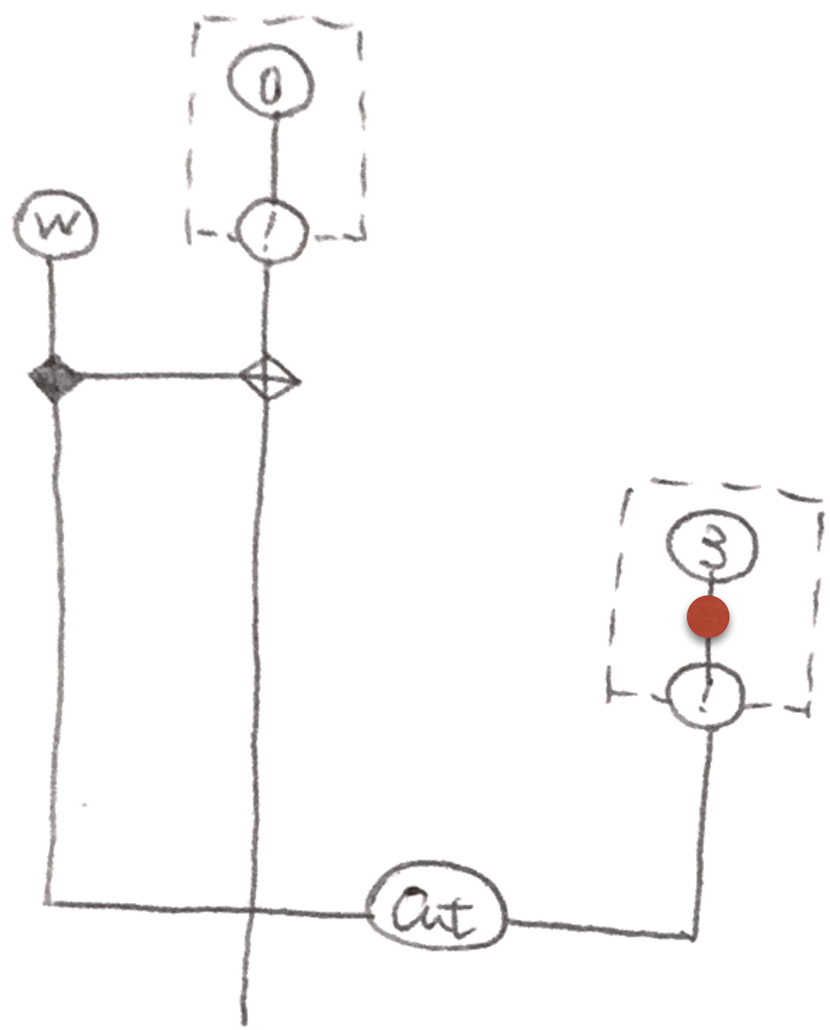
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1 + 2)$$



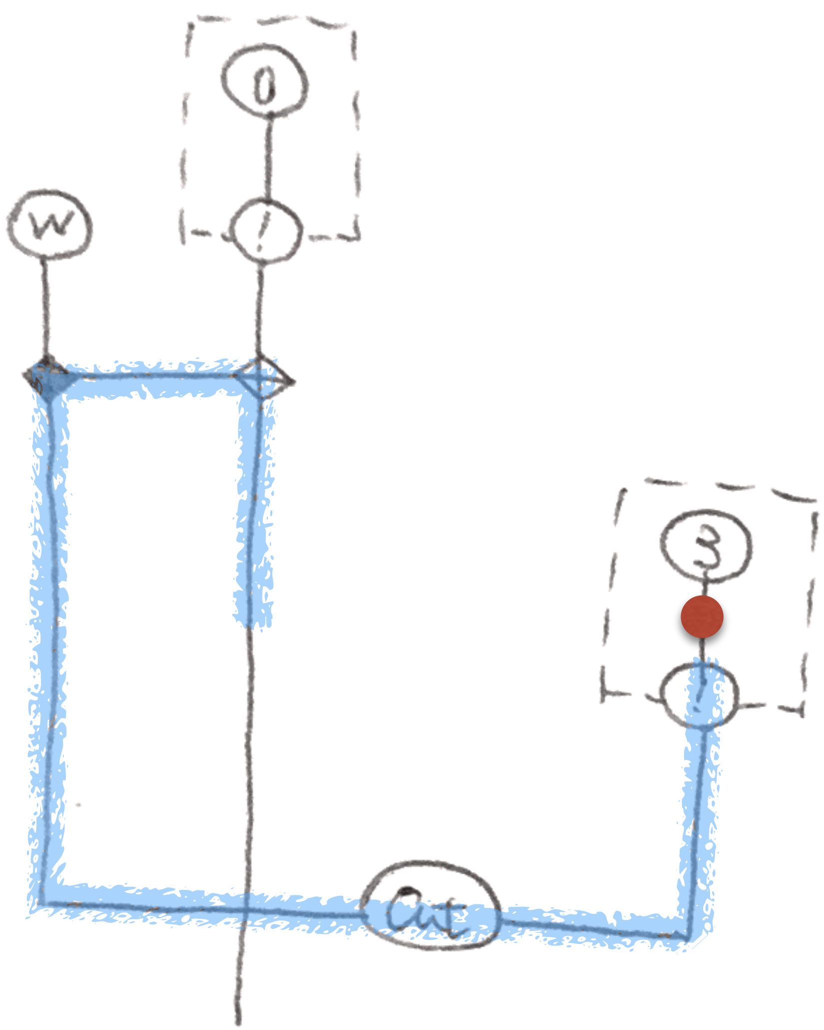
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1 + 2)$$



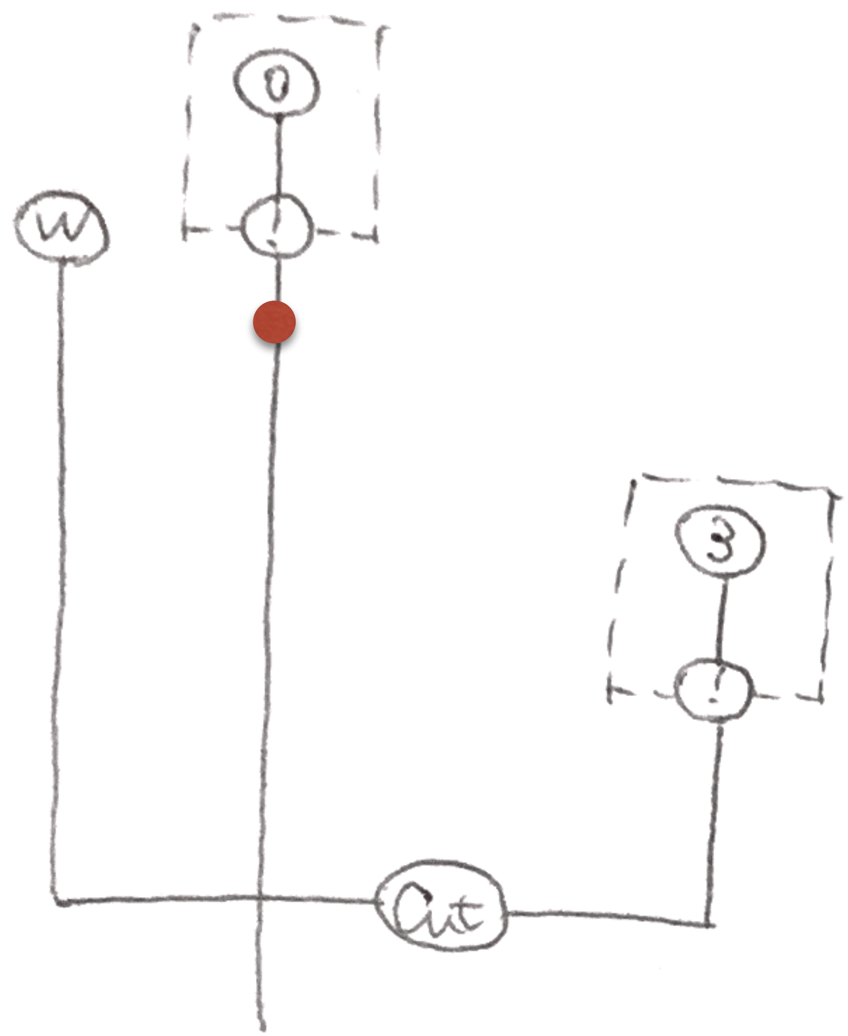
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\,(1 + 2)$$



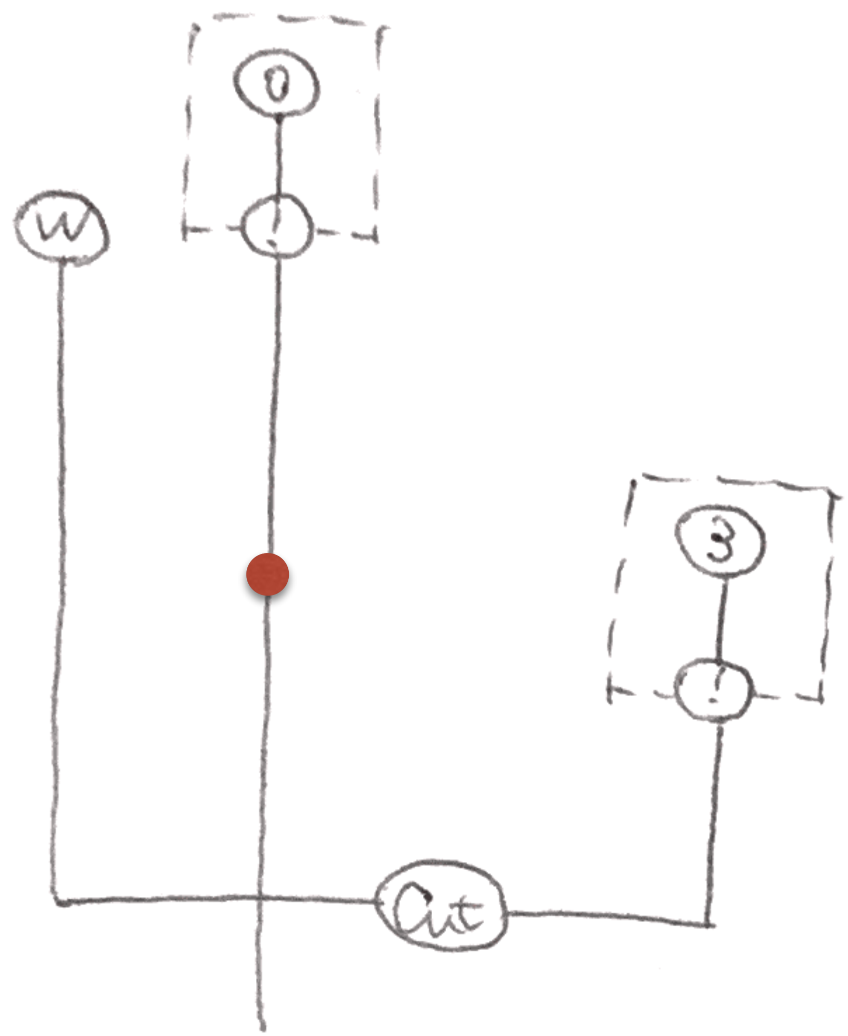
Force evaluation of arguments: checkpoint

$$(\lambda x.\, 0)\ (1 + 2)$$



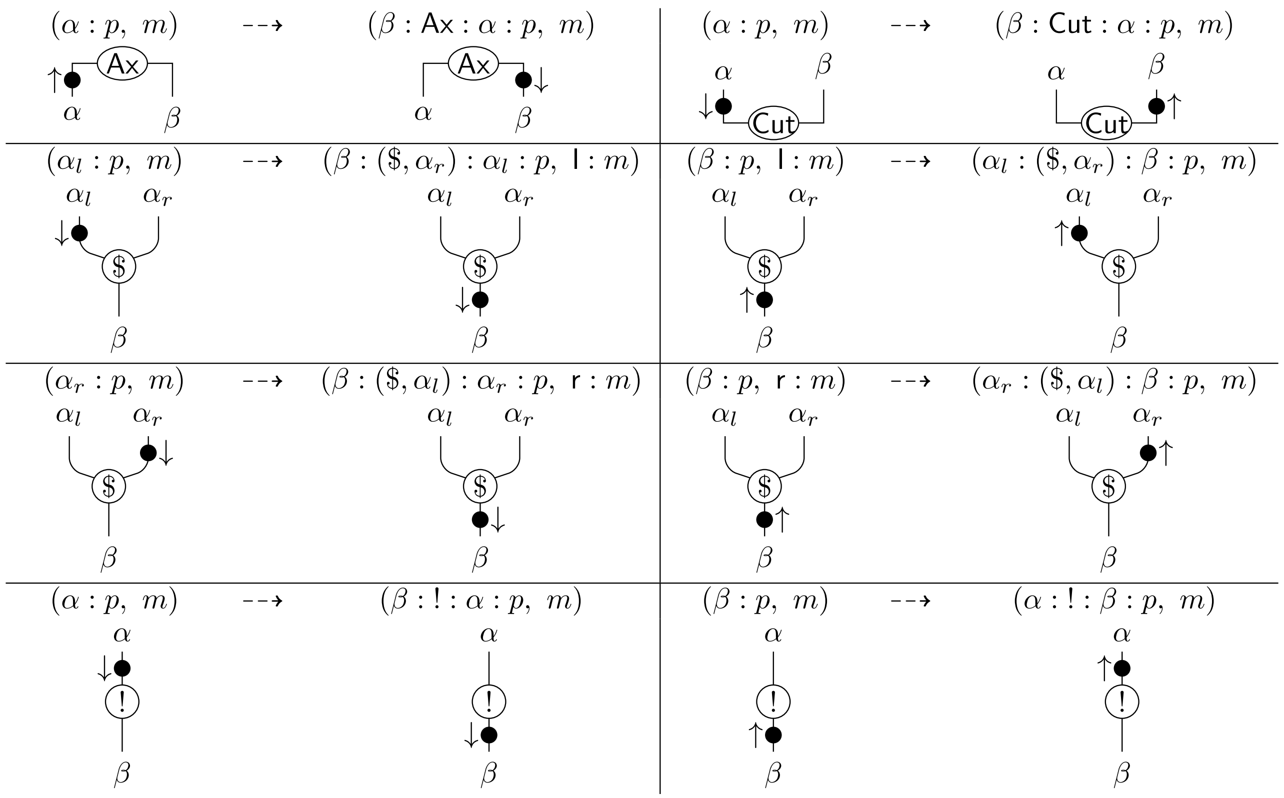
Force evaluation of arguments: checkpoint

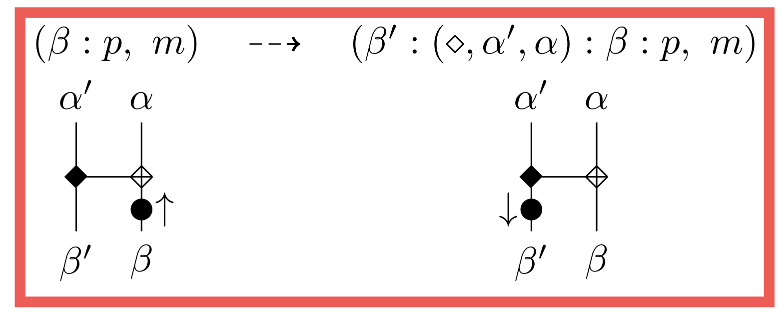
$$(\lambda x.\, 0)\ (1 + 2)$$

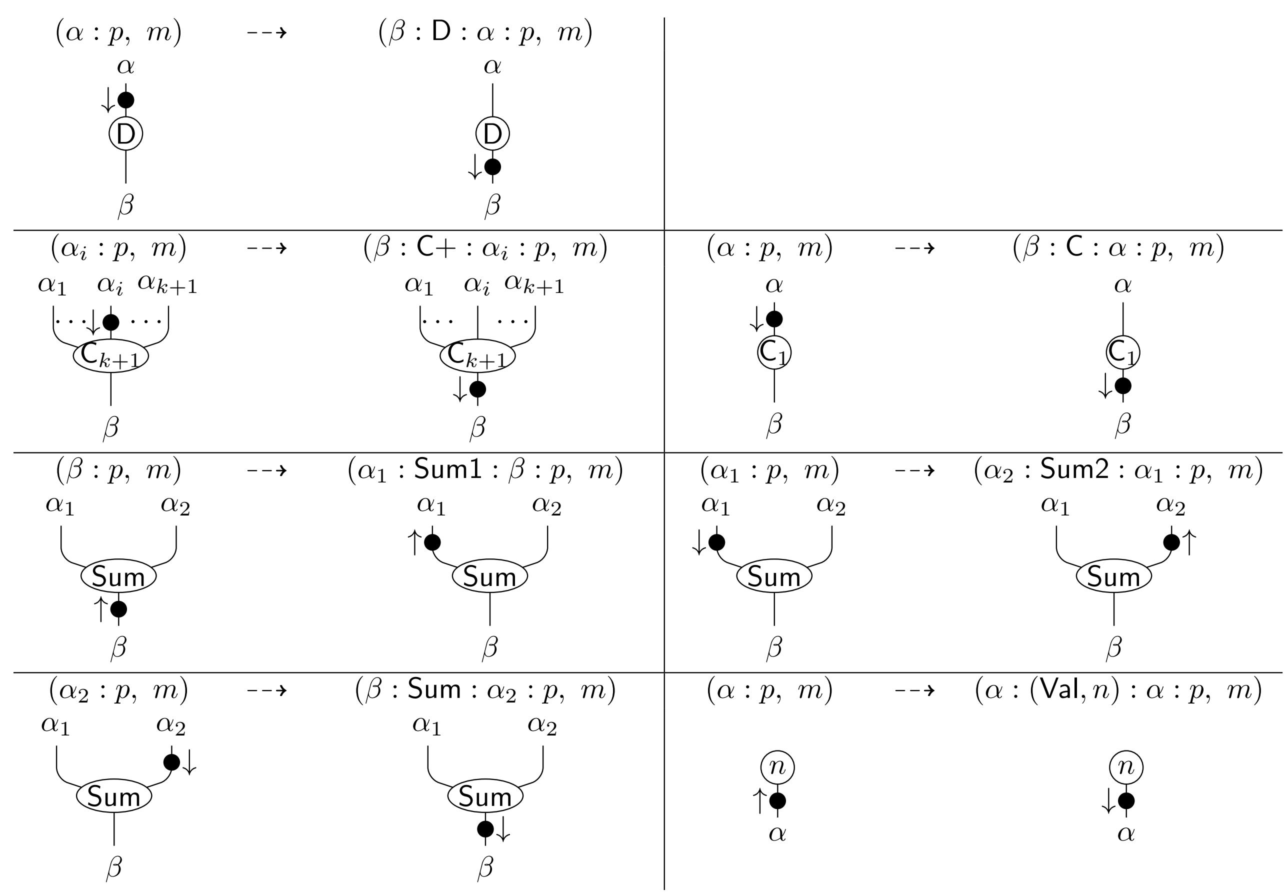


Dynamic GoI machine, extended

“move” transition $((G, \ell_e, \ell_b), p, d, m) \dashrightarrow ((G, \ell_e, \ell_b), p', d', m')$



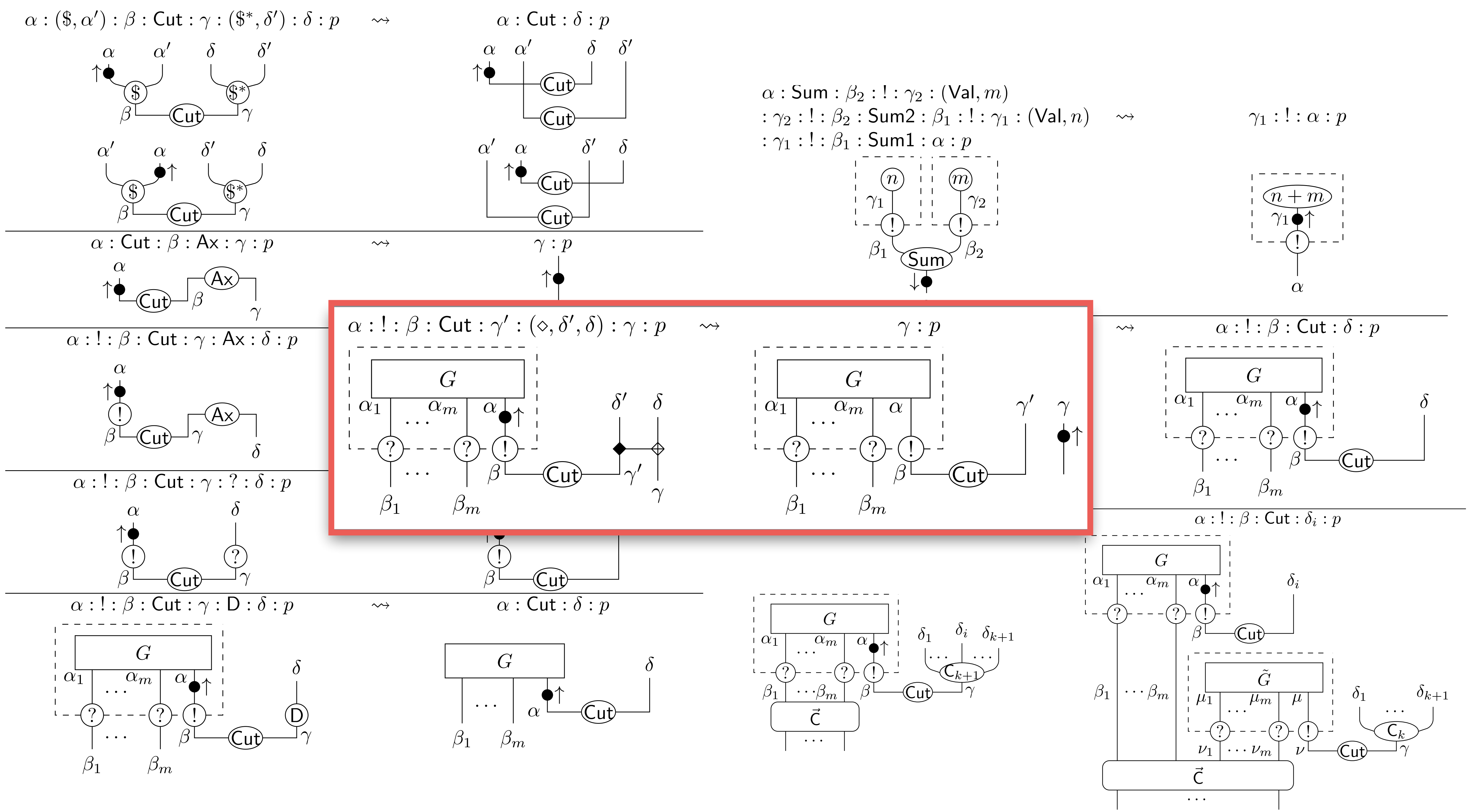




Muroya (U. Birmingham)

Dynamic GoI machine, extended

"rewrite" transition $((G, \ell_e, \ell_b), p, d, m) \rightsquigarrow ((G', \ell'_e, \ell'_b), p', d', m)$



Muroya (U. Birmingham)

“Linear-cost” simulation

- Dynamic GoI machine $\rightarrowtriangle \overset{\text{def.}}{\Longleftrightarrow} \begin{cases} \rightsquigarrow & \text{if a rewrite is possible} \\ \dashrightarrow & \text{if no rewrite but a move is possible} \end{cases}$

- Call-by-value storeless abstract machine

$$\begin{aligned}
\langle V, E\rangle_t &\to_{\text{val}} \langle E, V\rangle_c \\
\langle M\,N, E\rangle_t &\to_{\text{val}} \langle M, E[[\,]\,N]\rangle_t \\
\langle M + N, E\rangle_t &\to_{\text{val}} \langle M, E[[\,] + N]\rangle_t \\
\langle x, E_1[\texttt{let}\ x = V\ \texttt{in}\ E_2]\rangle_t &\to_{\text{val}} \langle E_1[\texttt{let}\ x = V\ \texttt{in}\ E_2], V\rangle_c
\end{aligned}$$

$$\begin{aligned}
\langle [\,], A[V]\rangle_c &\to_{\text{val}} \langle A[V]\rangle_a \\
\langle E[[\,]\,N], A[\lambda x.\,M]\rangle_c &\to_{\text{val}} \langle N, E[A[\texttt{let}\ x' := [\,]\ \texttt{in}\ M[x'/x]]]\rangle_t \\
\langle E[[\,] + N], A[\underline{n}]\rangle_c &\to_{\text{val}} \langle N, E[A[\underline{n} + [\,]]]\rangle_t \\
\langle E[\underline{n} + [\,]], A[\underline{m}]\rangle_c &\to_{\text{val}} \langle E, A[\underline{n+m}]\rangle_c \\
\langle E[\texttt{let}\ x = V'\ \texttt{in}\ [\,]], A[V]\rangle_c &\to_{\text{val}} \langle E, \texttt{let}\ x = V'\ \texttt{in}\ A[V]\rangle_c \\
\langle E[\texttt{let}\ x := [\,]\ \texttt{in}\ M], A[V]\rangle_c &\to_{\text{val}} \langle M, E[A[\texttt{let}\ x = V\ \texttt{in}\ [\,]]]\rangle_t
\end{aligned}$$

Theorem A.2. There exists a binary relation $\ddagger$ that satisfies

$$c \overset{k}{\to}_{\text{val}} c' \ \wedge\ c \ddagger (G, p, d, m)$$
$$\implies\ (G, p, d, m) \rightarrowtriangle \overset{\mathcal{O}(k)}{\cdots} \rightarrowtriangle (G', p', d', m')\ \wedge\ c' \ddagger (G', p', d', m')\ .$$

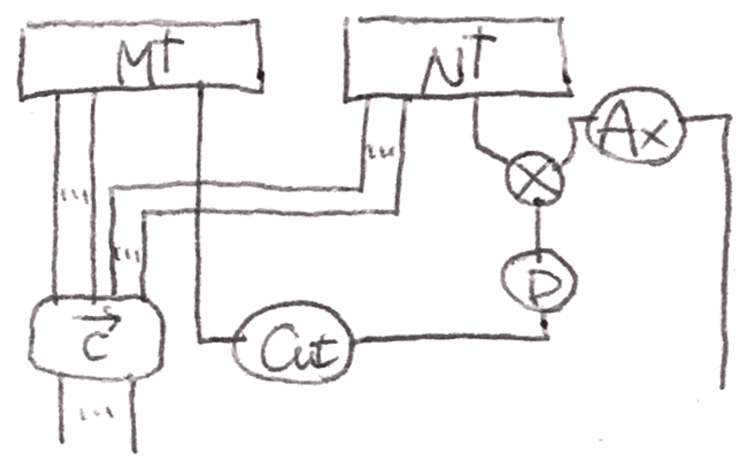
“Linear-cost” simulation

Theorem A.2. There exists a binary relation $\ddagger$ that satisfies

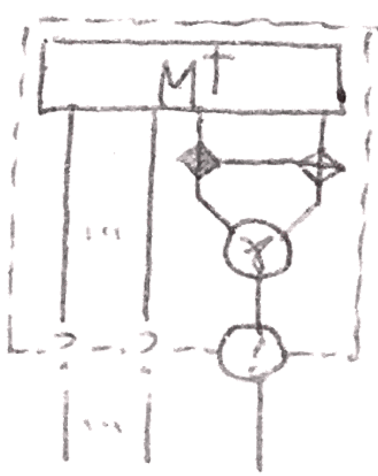
$$c \xrightarrow{k}_{\mathrm{val}} c' \;\wedge\; c \ddagger (G, p, d, m)$$
$$\Longrightarrow \; (G, p, d, m) \xrightarrow{\mathcal{O}(k)} (G', p', d', m') \;\wedge\; c' \ddagger (G', p', d', m') \;.$$

$x^{\dagger} :=$

$(M\,N)^{\dagger} :=$



$(\lambda x.\, M)^{\dagger} :=$



GoI machine [Danos & Regnier '99] [Mackie '95]

call-by-name call-by-need call-by-value effects

	avoid re-evaluation	force evaluation of arguments	track each copy of terms
CPS transform.	?	✔	?
memory	△		✔
parallelism & sync.	△	✔	
dynamic jump	✔	✔	
dynamic rewrite	✔		☺
checkpoint		✔	

Schöpp

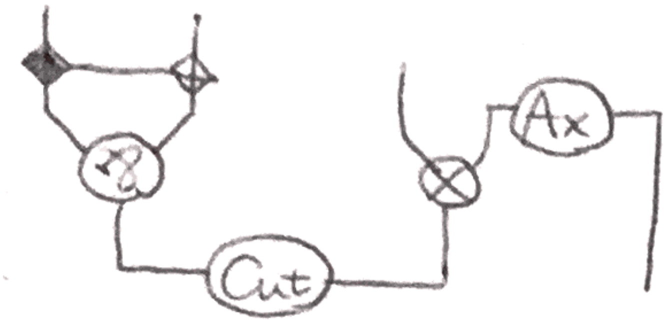
Hoshino+

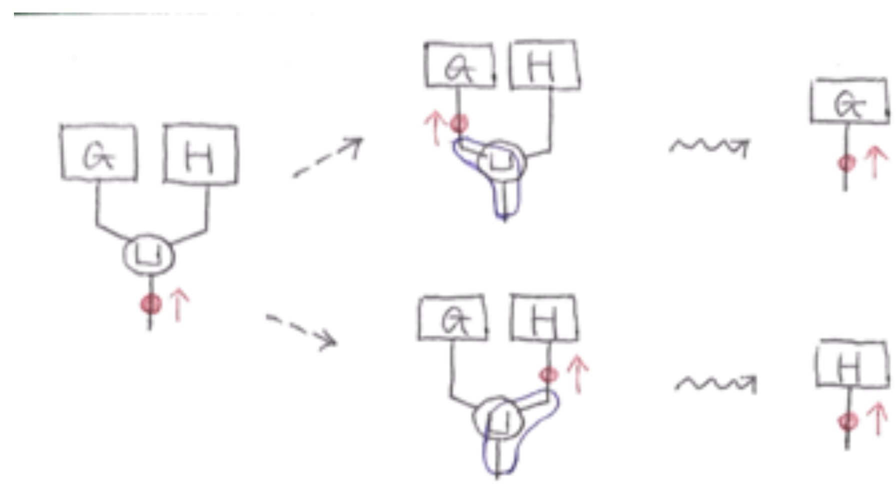
Dal Lago+

Fernández+

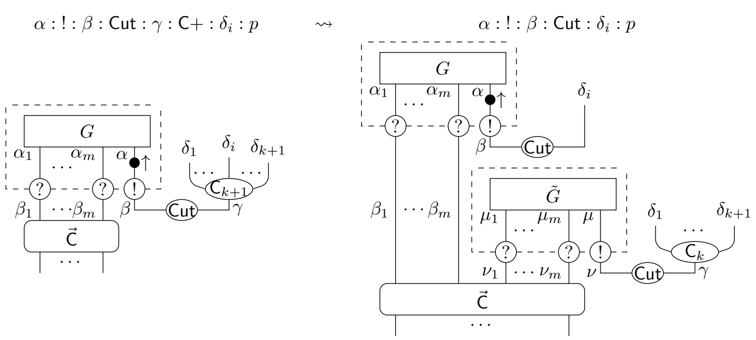
Dynamic rewrite & checkpoint

- jumping as rewriting

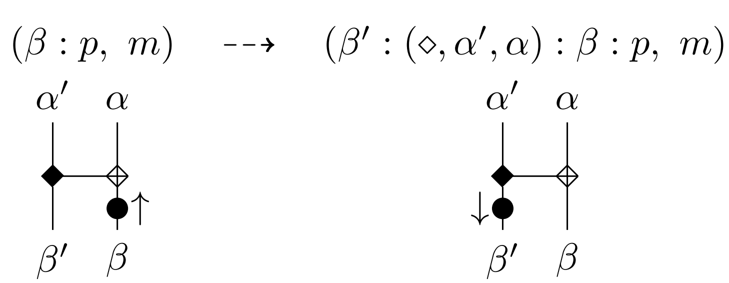


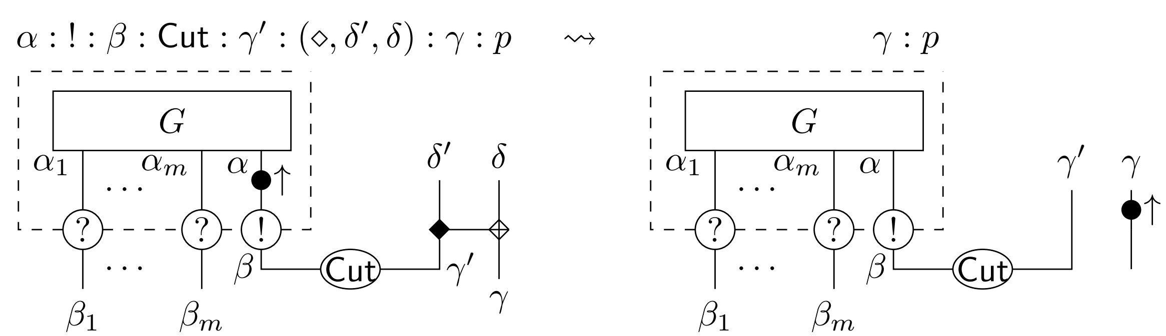


- honest copying



- potential parallelism





As an abstract machine

- efficiency
  - no cost for look-up $\langle x, E_1[\texttt{let}\ x = N\ \texttt{in}\ E_2]\rangle_t \to_{\mathrm{need}} \langle N, E_1[\texttt{let}\ x := [\,]\ \texttt{in}\ E_2[x]]\rangle_t$
  - garbage collection integrated, in a restricted way

$$\langle E[\texttt{let}\ x := [\,]\ \texttt{in}\ E'[x]], A[V]\rangle_c \to_{\mathrm{need}} \langle E[A[\texttt{let}\ x = V\ \texttt{in}\ E']], V\rangle_c$$

$$\langle E[\texttt{let}\ x := [\,]\ \texttt{in}\ E'[x]], A[V]\rangle_c \to_{\mathrm{need}} \langle E[A[E']], V\rangle_c \quad (\text{if } x \text{ does not appear in } E')$$

- compositional reasoning
- automated "graph rewriter" of specific strategies
  - rather than "graph rewriting simulator"