Efficient implementation of evaluation strategies

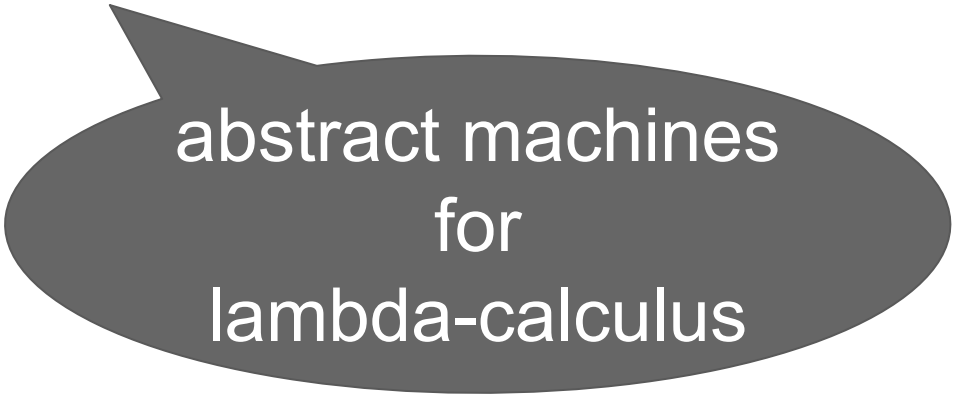
via token-guided graph rewriting

Koko Muroya & Dan R. Ghica
(University of Birmingham)

WPTE (Oxford), 8 Sep. 2017

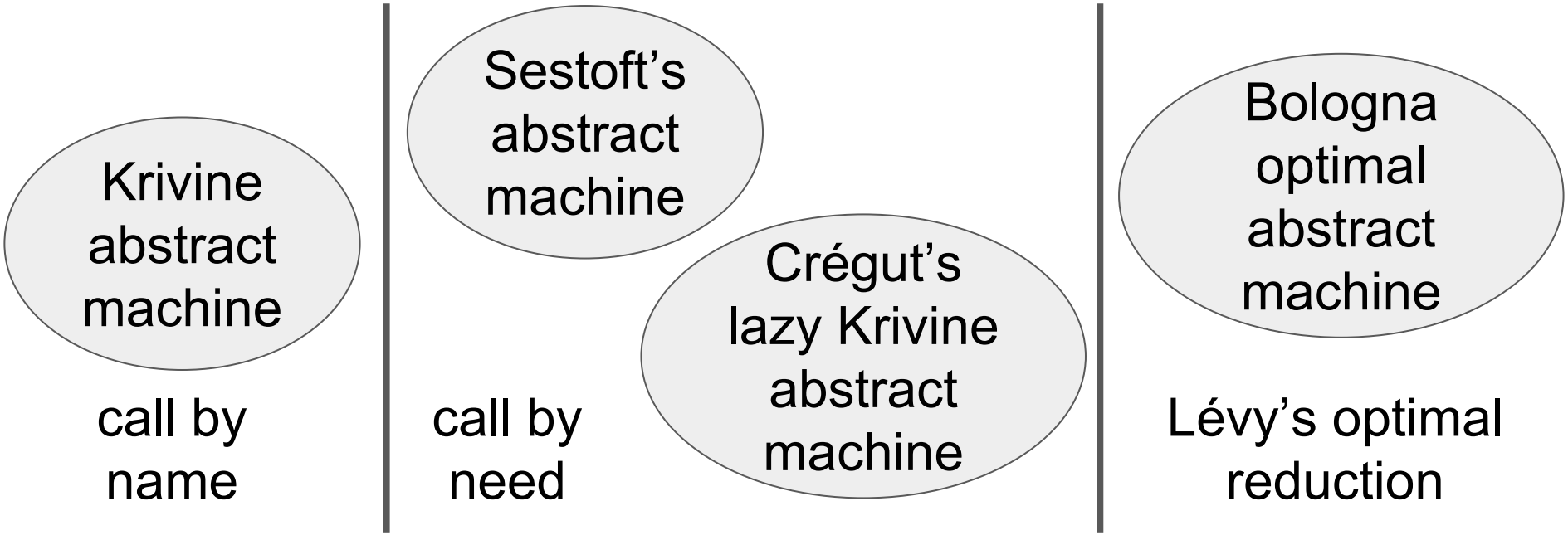
TARGET

balancing space cost & time cost
of program execution



Motivation: series of non-strict evaluation

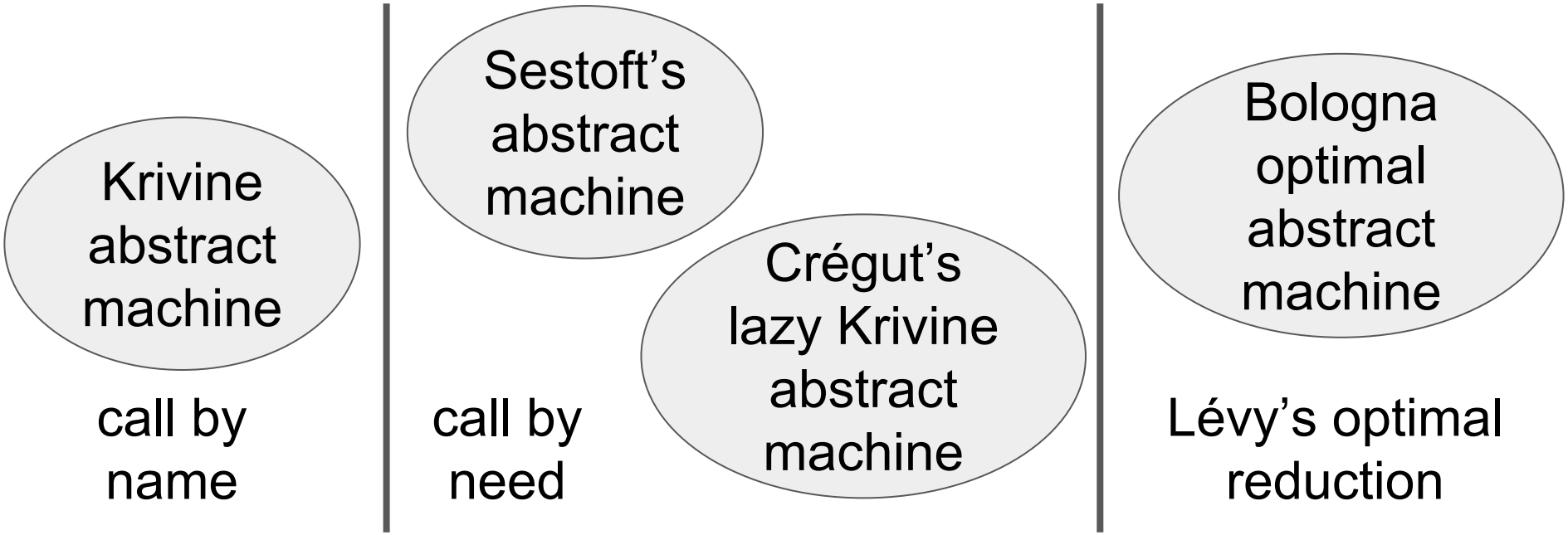
- abstract machines of same end result
  - number of beta-reduction



Motivation: series of non-strict evaluation

- abstract machines of same end result
  - number of beta-reduction
  - time cost

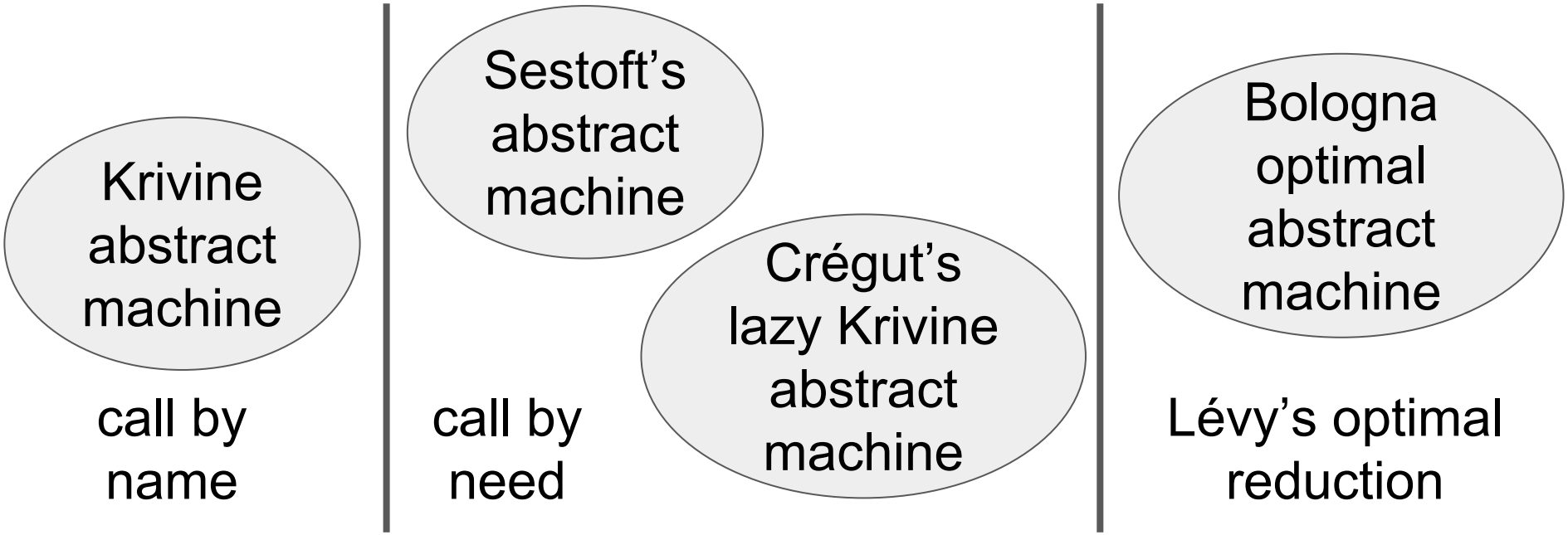
Accattoli et al.



Motivation: series of non-strict evaluation

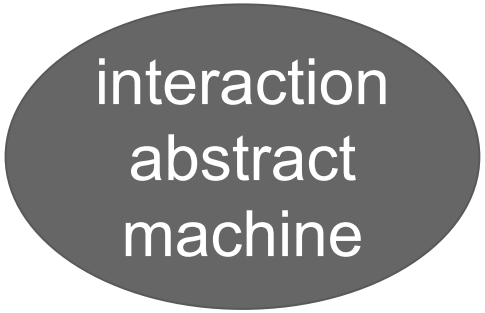
- abstract machines of same end result
  - number of beta-reduction
  - time cost
  - space cost

Accattoli et al.



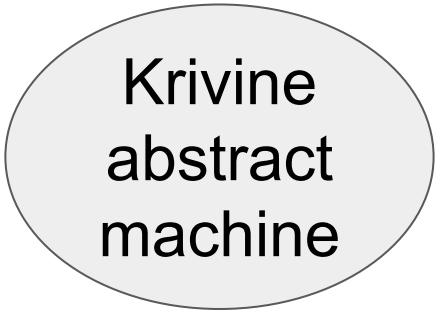
Motivation: series of non-strict evaluation

- abstract machines of same end result
  - space cost



[Danos & Regnier '99]

- GoI-style token passing
- fixed graph

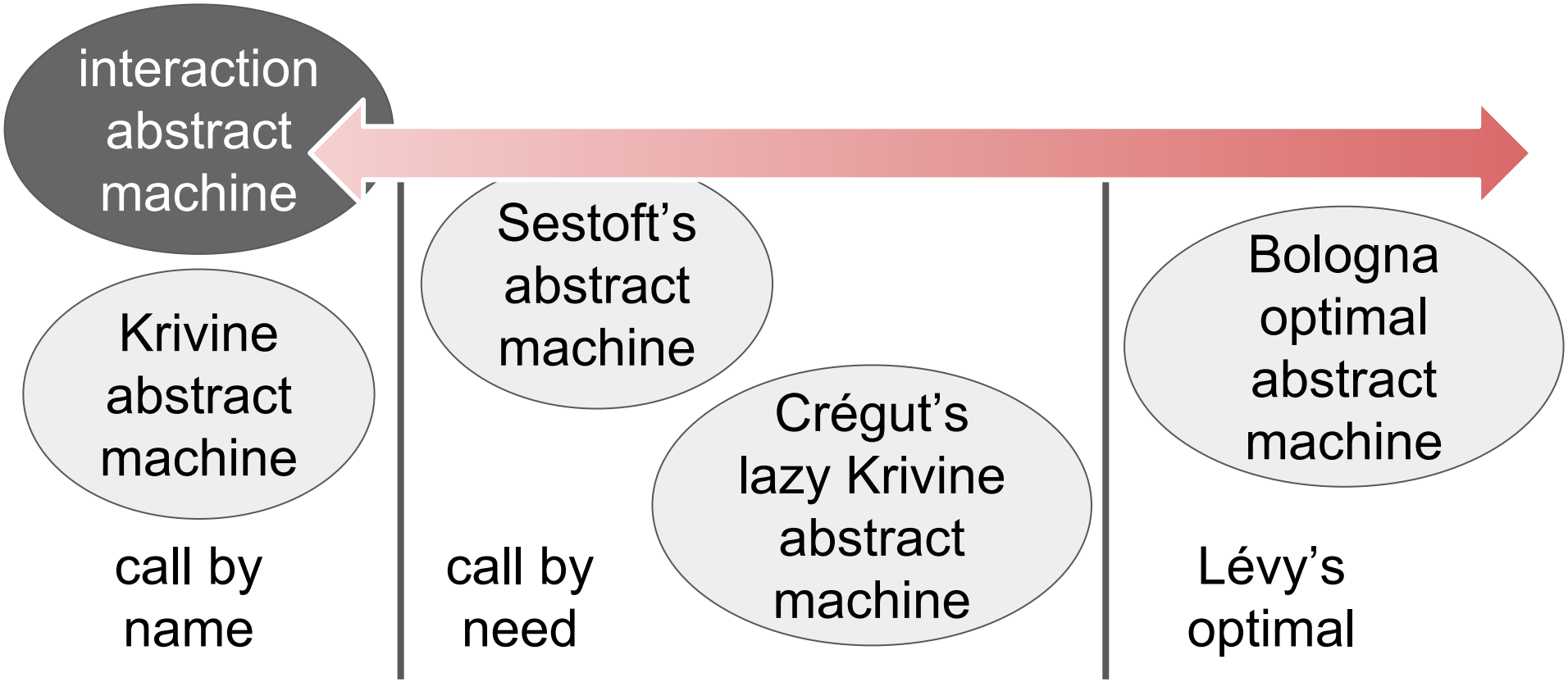


- term rewriting

call by
name

Question

- abstract machines of same end result
  - space cost vs time cost… trade-off?



GOAL

unified framework
that can balance space cost & time cost
of program execution

GOAL

unified framework
that can balance space cost & time cost
of program execution

token-guided graph-rewriting abstract machine
for lambda-calculus

Token-guided graph rewriting

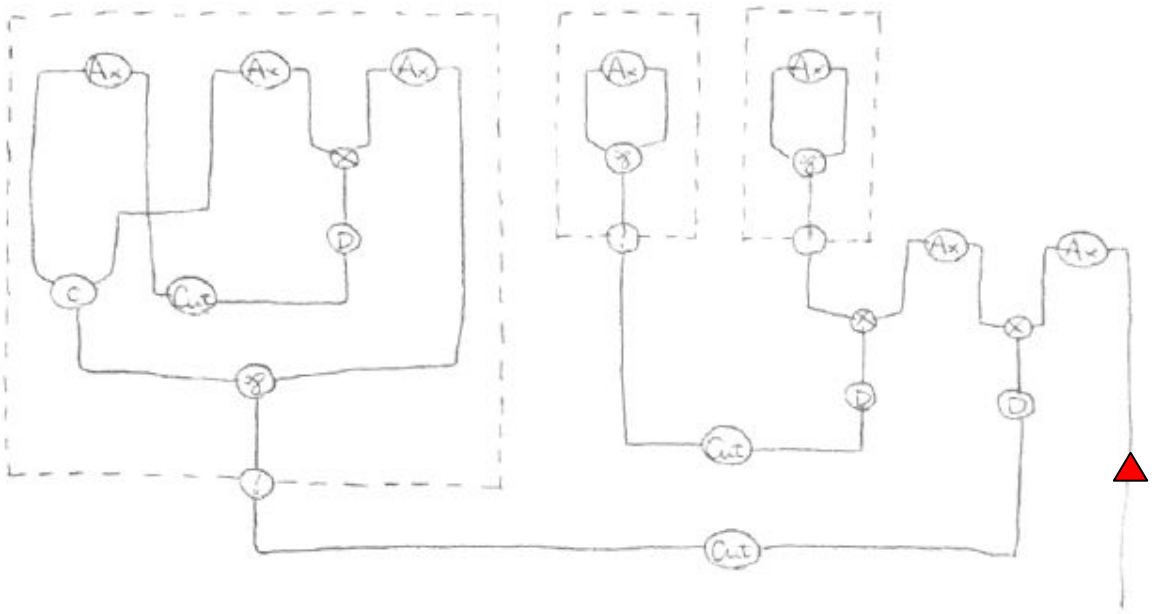
GoI-style token passing,

interleaved with graph rewriting

Token-guided graph rewriting

GoI-style token passing,

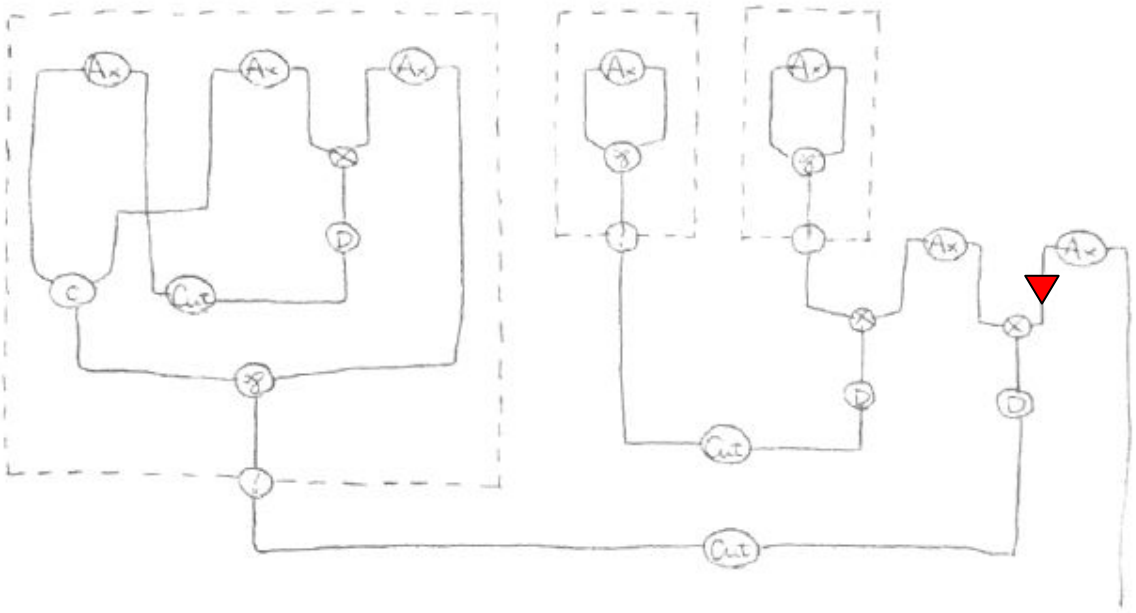
interleaved with graph rewriting



Token-guided graph rewriting

GoI-style token passing,

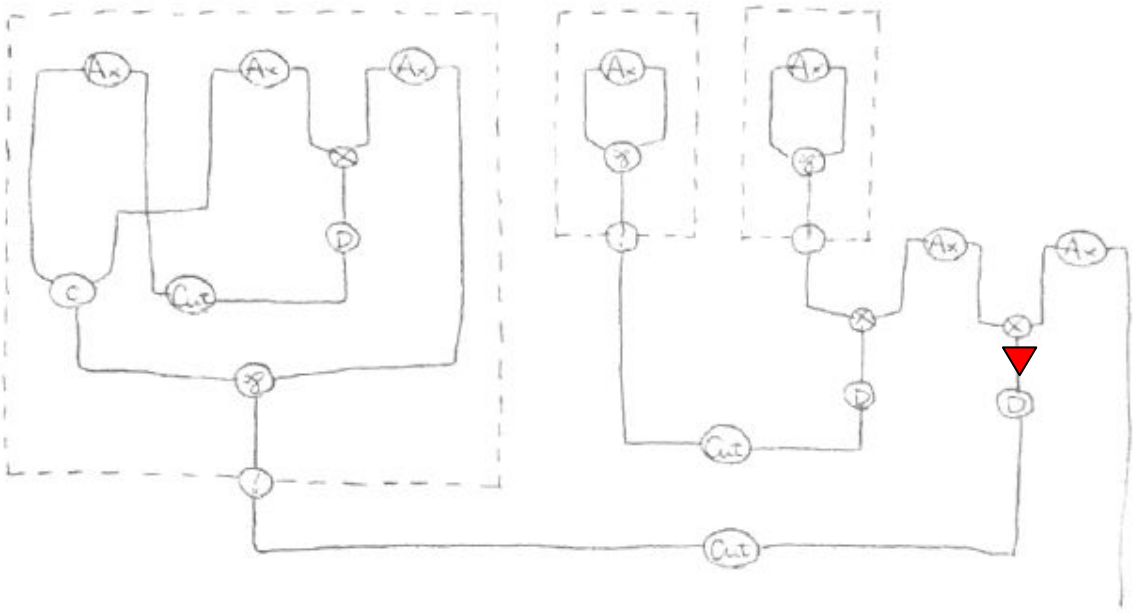
interleaved with graph rewriting



Token-guided graph rewriting

GoI-style token passing,

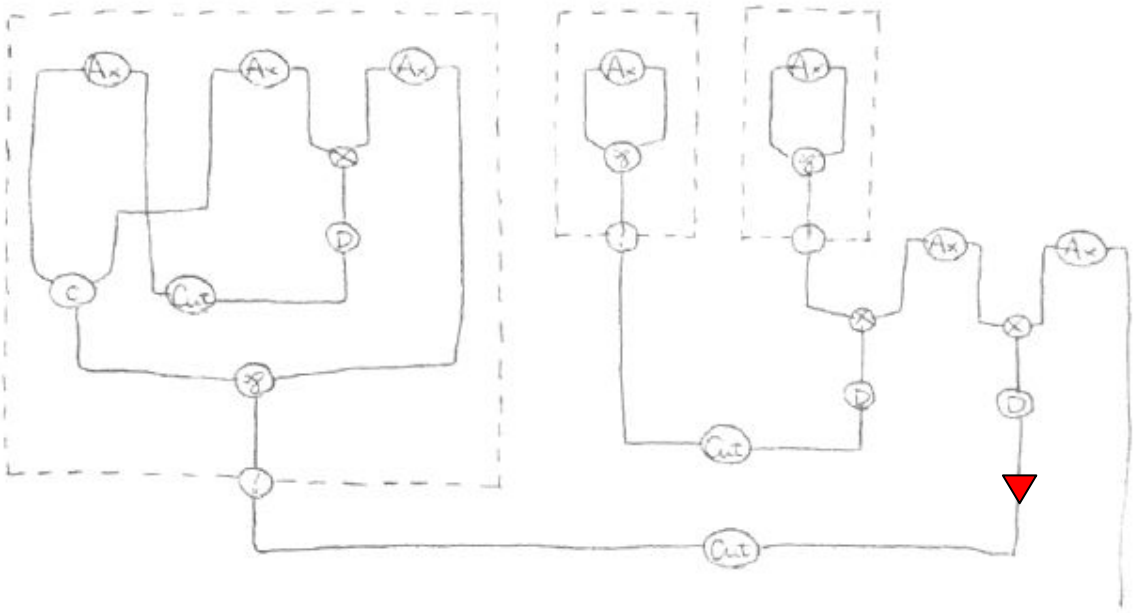
interleaved with graph rewriting



Token-guided graph rewriting

GoI-style token passing,

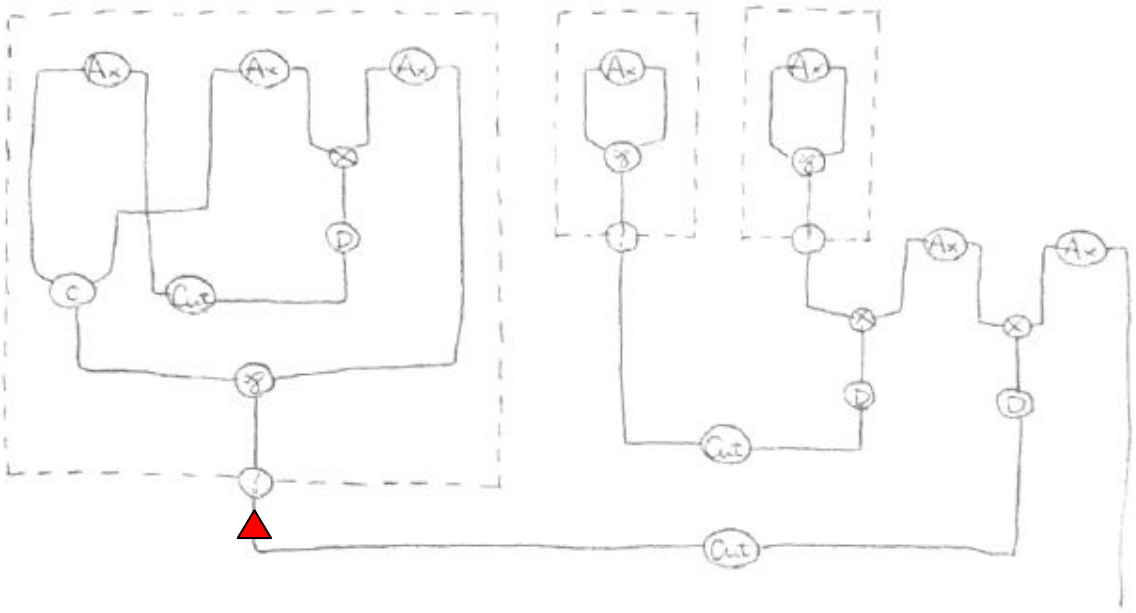
interleaved with graph rewriting



Token-guided graph rewriting

GoI-style token passing,

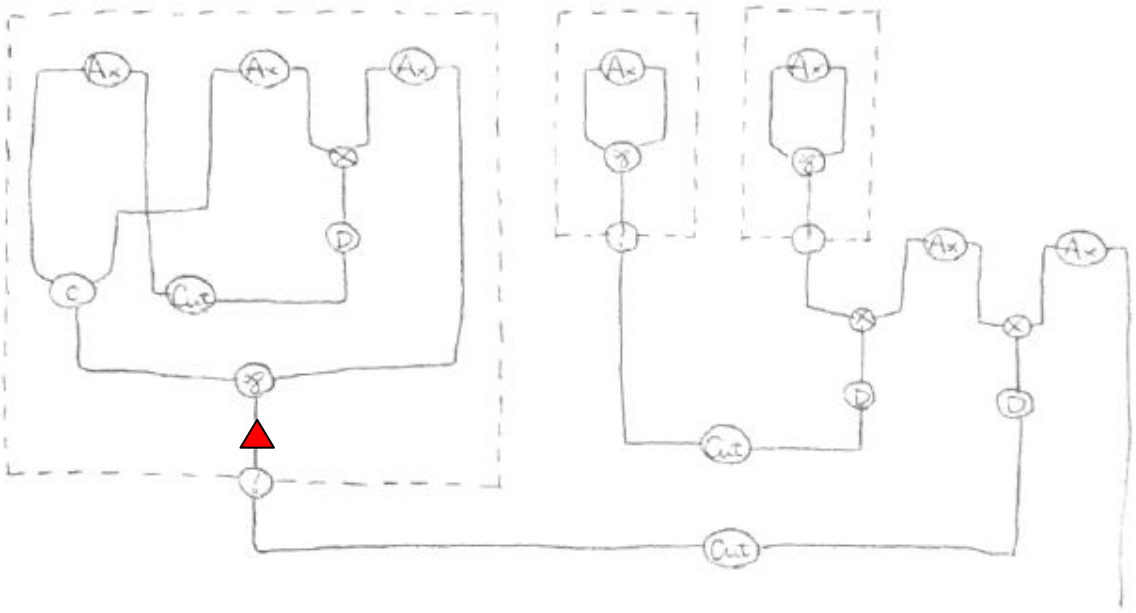
interleaved with graph rewriting



Token-guided graph rewriting

GoI-style token passing,

interleaved with graph rewriting

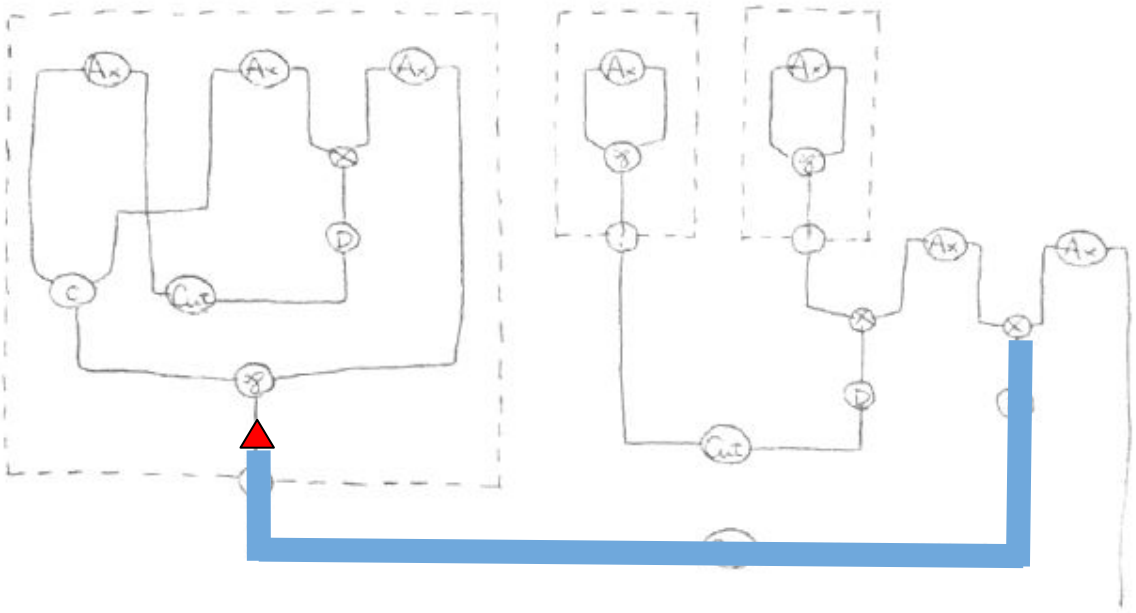


Token-guided graph rewriting

GoI-style token passing,

interleaved with graph rewriting

redex detected

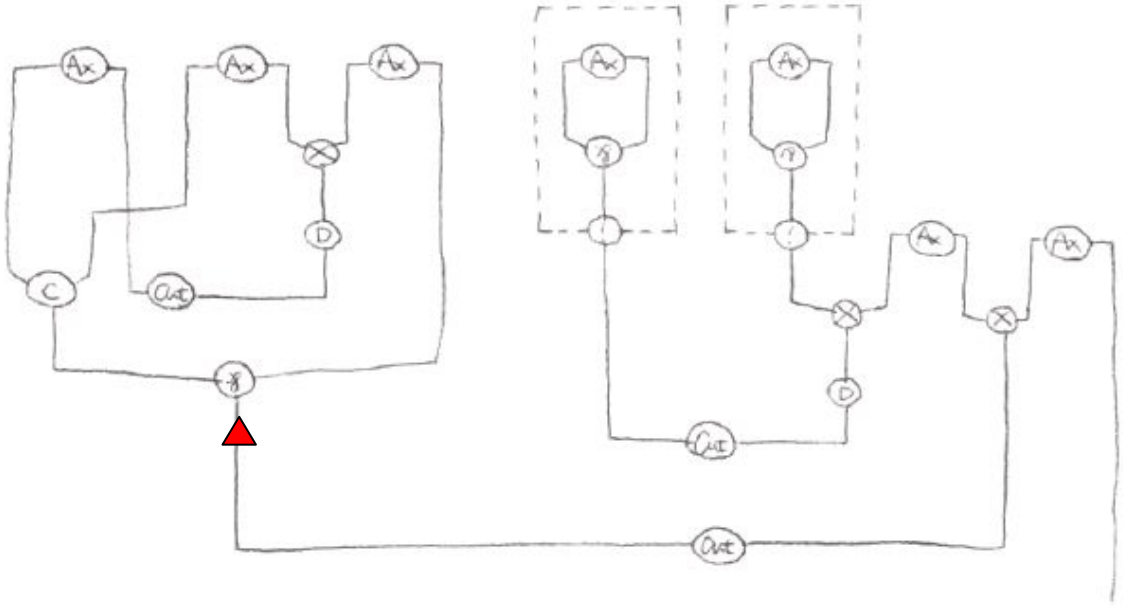


Token-guided graph rewriting

GoI-style token passing,

interleaved with graph rewriting

(1) trigger rewriting

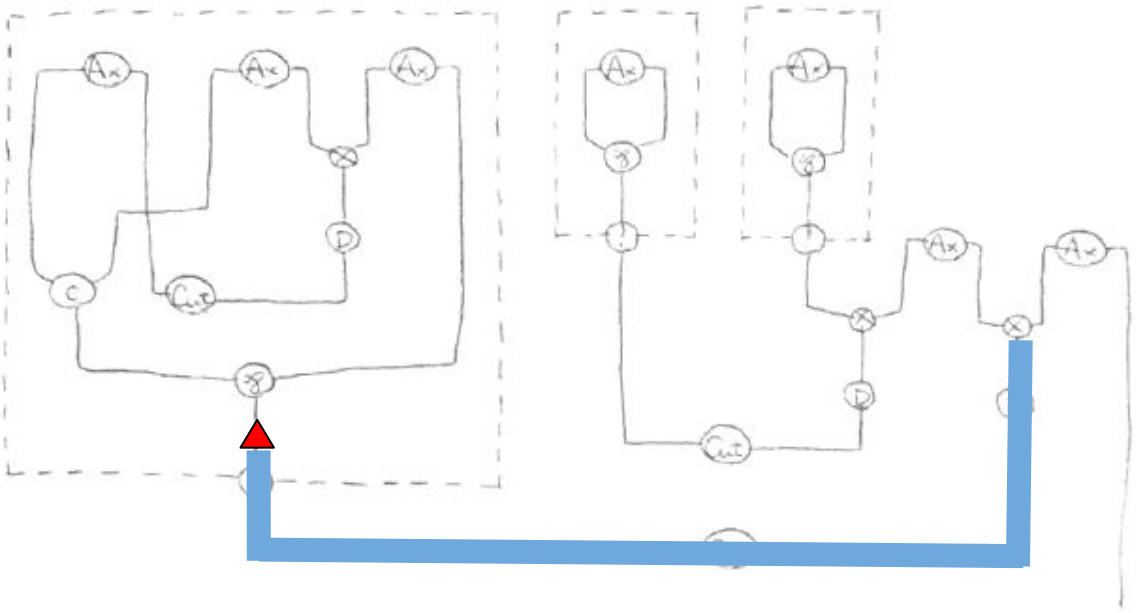


Token-guided graph rewriting

GoI-style token passing,

interleaved with graph rewriting

redex detected

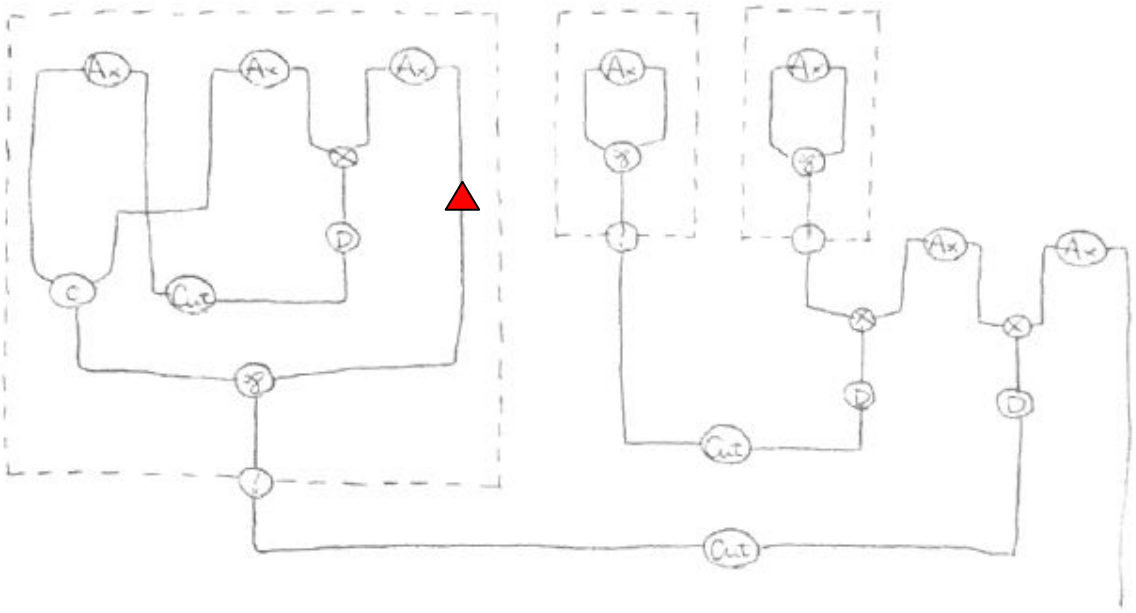


Token-guided graph rewriting

GoI-style token passing,

interleaved with graph rewriting

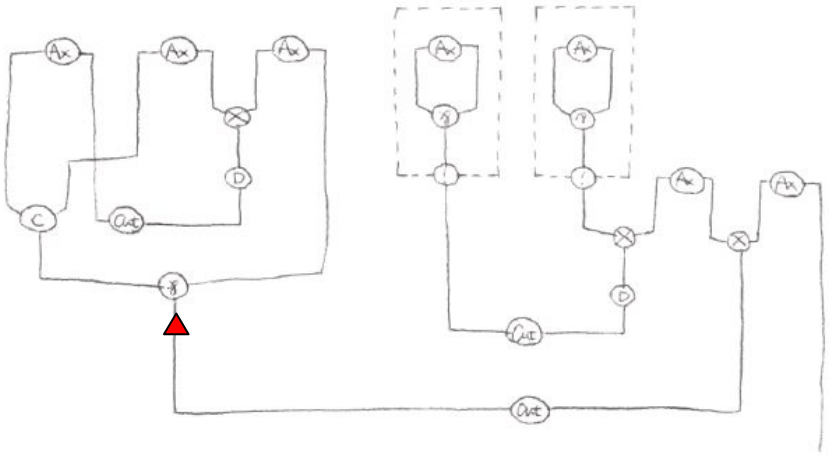
(2) keep passing



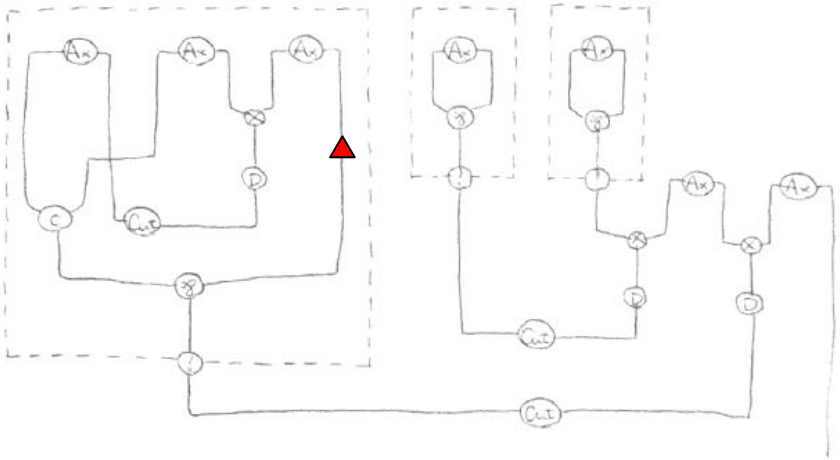
Token-guided graph rewriting

GoI-style token passing,

interleaved with graph rewriting



(1) trigger rewriting



(2) keep passing

Token-guided graph rewriting for lambda-calculus

flexibility, by choices of:

- graph rewriting system, with token passing
  - proof nets
- interleaving strategy
  - trigger rewriting vs. keep passing
- translation of lambda-terms
  - $!(A \multimap B)$, $(!A) \multimap B$

Token-guided graph rewriting for lambda-calculus

flexibility, by choices of:

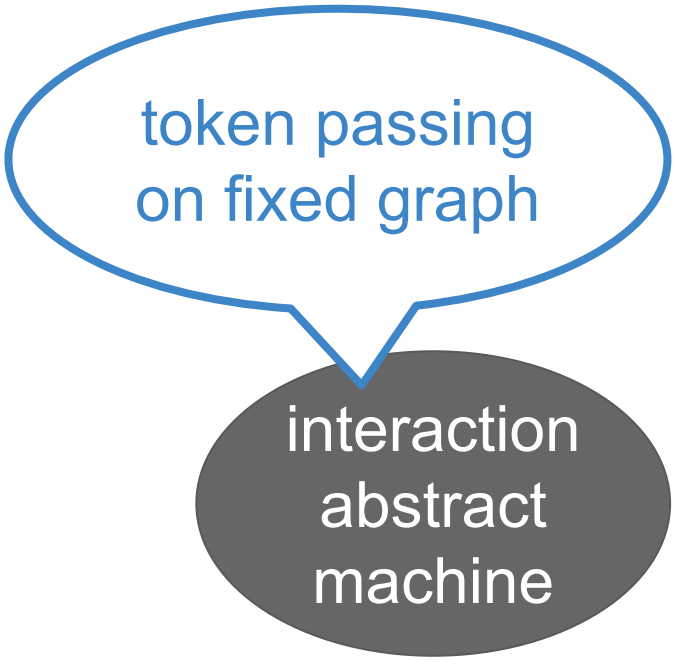
- graph rewriting system, with token passing
- interleaving strategy
- translation of lambda-terms

to…

- balance space cost & time cost

Non-strict evaluation: time cost improvement

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms



call by name

call by need

Non-strict evaluation: time cost improvement

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms

token passing on fixed graph

- proof nets
- **passes-only**
- !(A ⊸ B)

call-by-name time cost

call by need

Non-strict evaluation: time cost improvement

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms

token passing on fixed graph

- proof nets
- **passes-only**
- !(A ⊸ B)

call-by-name time cost

[- & Ghica, CSL ‘17]

- proof nets
- **rewrites-first**
- !(A ⊸ B)

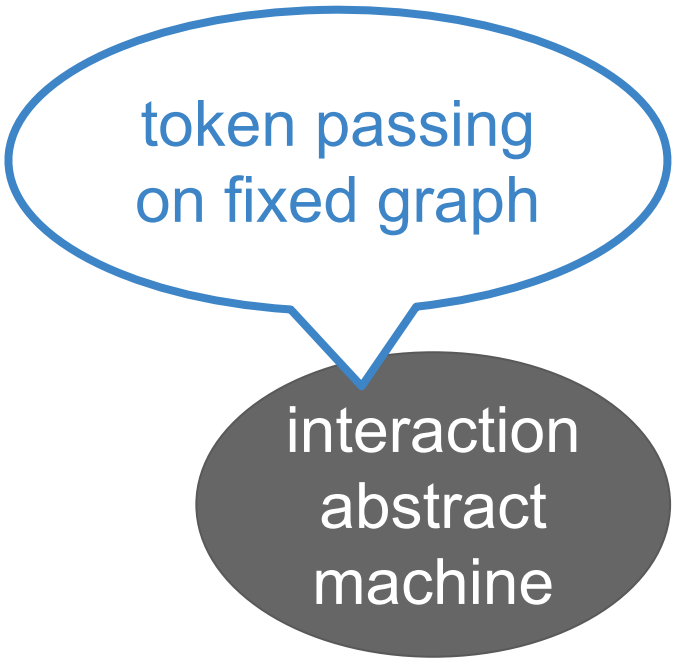
call-by-need time cost

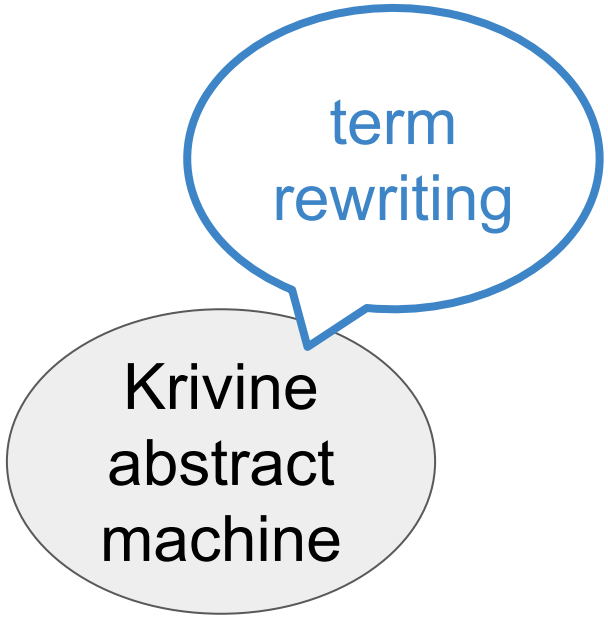
graph rewriting

time cost analysis à la [Accattoli ’16]

Non-strict evaluation: space cost improvement?

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms





call by name

Non-strict evaluation: space cost improvement?

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms

token passing on fixed graph

- proof nets
- **passes-only**
- !(A ⊸ B), (!A) ⊸ B

call-by-name time cost

term rewriting

Krivine abstract machine

call by name

Non-strict evaluation: space cost improvement?

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms

token passing on fixed graph

- proof nets
- **passes-only**
- !(A ⊸ B), (!A) ⊸ B

call-by-name time cost

graph rewriting

- proof nets
- **rewrites-first**
- (!A) ⊸ B

call-by-name time cost

call by name

Strict evaluation

- graph rewriting system with token passing
- interleaving strategy
- translation of lambda-terms

[- & Ghica, WPTE ‘17]

- $\lambda_!^{[\overrightarrow{@}, \overleftarrow{@}]}$**-graphs**
- rewrites-first
- !(A ⊸ B)

call-by-value time cost

graph rewriting

time cost analysis
à la [Accattoli ’16]

Token-guided graph rewriting for lambda-calculus

series of
- non-strict evaluation
- strict evaluation

flexibility, by choices of:

- graph rewriting system, with token passing
- interleaving strategy
- translation of lambda-terms

to…

- balance space cost & time cost

Analyse token-guided graph rewriting

via term rewriting + explicit redex searching [Sinot '05]

$$
\begin{aligned}
&(\lambda x.x\ x)\ ((\lambda y.y)\ (\lambda z.z))\\
&(\lambda x.x\ x)\ ((\lambda y.y)\ (\lambda z.z))\\
&(\lambda x.x\ x)\ ((\lambda y.y)\ (\lambda z.z))\\
&(\lambda x.x\ x)\ ((\lambda y.y)\ (\lambda z.z))\\
&(\lambda x.x\ x)\ ((\lambda y.y)\ (\lambda z.z))\\
&(\lambda x.x\ x)\ (y[y \leftarrow \lambda z.z])\\
&(\lambda x.x\ x)\ ((\lambda z.z)[y \leftarrow \lambda z.z])\\
&(\lambda x.x\ x)\ ((\lambda z.z)[y \leftarrow \lambda z.z])\\
&(x\ x)[x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]\\
&(x\ x)[x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]\\
&(x\ (\lambda z.z))[x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]\\
&(x\ (\lambda z.z))[x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]\\
&((\lambda z'.z')\ (\lambda z.z))[x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]\\
&z'[z' \leftarrow \lambda z.z][x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]\\
&\lambda z.z[z' \leftarrow \lambda z.z][x \leftarrow \lambda z.z][y \leftarrow \lambda z.z]
\end{aligned}
$$

https://cwtsteven.github.io/GoI-Visualiser/CBV-with-CBV-embedding/index.html

Token-guided graph rewriting for lambda-calculus

series of
- non-strict evaluation
- strict evaluation

flexibility, by choices of:

- graph rewriting system, with token passing
- interleaving strategy
- translation of lambda-terms

to…

- balance space cost & time cost

analysis via term rewriting