Vincent van Oostrom
Kees-Jan van de Looij
Marijn Zwitserlood
Universiteit Utrecht
Department of Philosophy

Contents

1. Goal

Extend optimality theory and practice to HRS

1. Goal

Extend optimality theory and practice to HRS

Subgoals:

- TRS optimality
  theory done (Terese book)
- λ-calculus optimality
  here

1. Goal

Extend optimality theory and practice to HRS

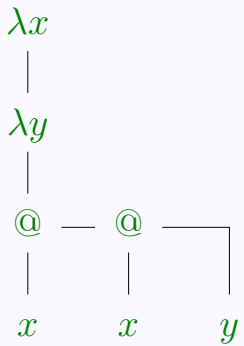
Subgoals:

- TRS optimality
  theory done (Terese book)
- λ-calculus optimality
  here

Literature: Lévy, Lamping, Gonthier, Abadi, Danos, Mackie, Regnier, Asperti, Laneve, Guerrini, Mairson, Lawall, ...
Here: analysis of De Bruijn's calculational approach

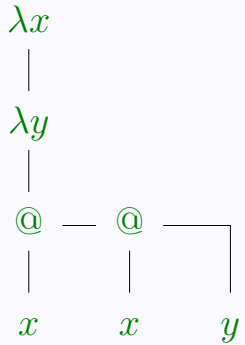
1.1. Named

Church numeral $\underline{2} = \lambda x.\lambda y.x(xy)$



1.1. Named

Church numeral $\underline{2} = \lambda x.\lambda y.x(xy)$

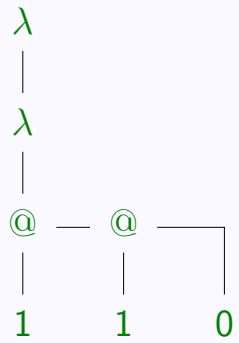


Application is exponentiation

$$\begin{aligned}
\underline{2}\,\underline{2} &\to_\beta \lambda y.\underline{2}(\underline{2}y) \\
&\to_\beta \lambda y.\underline{2}\lambda x.y(yx) \\
&\to_\beta \lambda y.\lambda z.(\lambda x.y(yx))((\lambda x.y(yx))z) \\
&\to_\beta \lambda y.\lambda z.(\lambda x.y(yx))(y(yz)) \\
&\to_\beta \lambda y.\lambda z.y(y(y(yz))) = \underline{4}
\end{aligned}$$

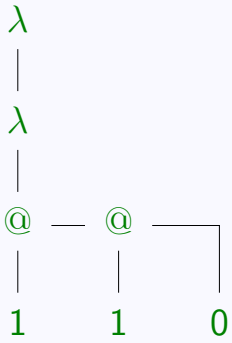
1.2. De Bruijn indexed

De Bruijn indexed Church numeral $\underline{2} = \lambda\lambda 1(10)$



1.2. De Bruijn indexed

De Bruijn indexed Church numeral $\underline{2} = \lambda\lambda 1(10)$

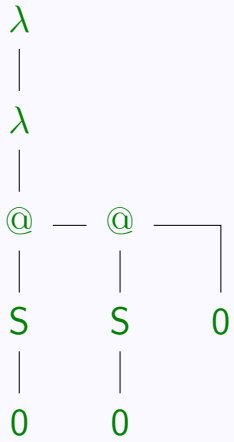


De Bruijn indexed exponentiation

$$
\begin{aligned}
\underline{2}\,\underline{2} &\rightarrow_\beta \lambda\underline{2}(\underline{2}0) \\
&\rightarrow_\beta \lambda\underline{2}\lambda 1(10) \\
&\rightarrow_\beta \lambda\lambda(\lambda 2(20))((\lambda 2(20))0) \\
&\rightarrow_\beta \lambda\lambda(\lambda 2(20))(1(10)) \\
&\rightarrow_\beta \lambda\lambda 1(1(1(10)))
\end{aligned}
$$

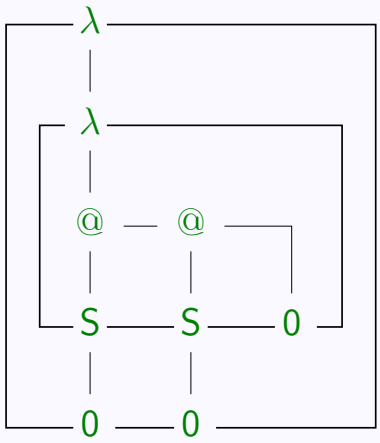
1.3. Unary indexed

Unary indexed $\underline{2} = \lambda\lambda(\mathsf{S}0)((\mathsf{S}0)0)$



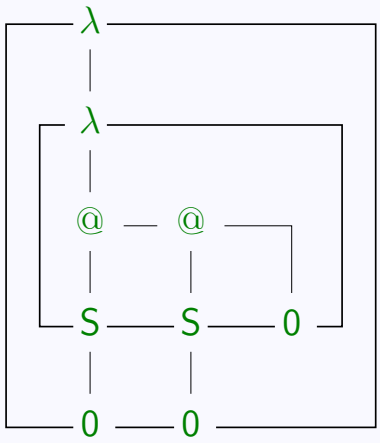
1.4. Scoped

Scoped $\underline{2} = \lambda\lambda(\mathsf{S}0)((\mathsf{S}0)0)$



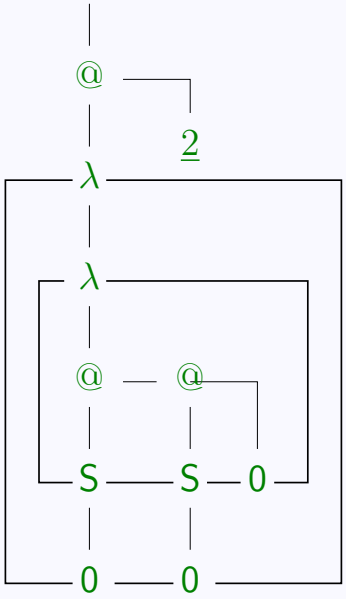
1.4. Scoped

Scoped $\underline{2} = \lambda\lambda(\mathsf{S}0)((\mathsf{S}0)0)$



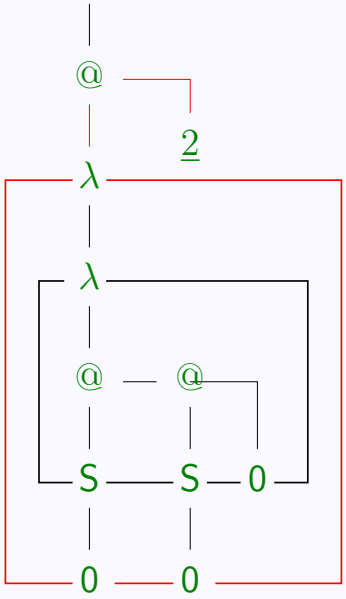
scoping is matching
binding is match with 0
end-of-scope is match with S
λ-terms are context-free trees

1.5. β-reduction



how to reduce?

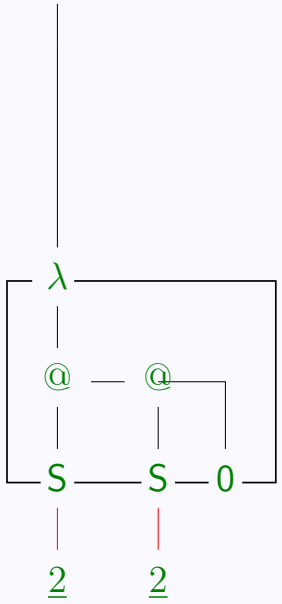
1.6. β-reduction: replication



put argument if matching 0

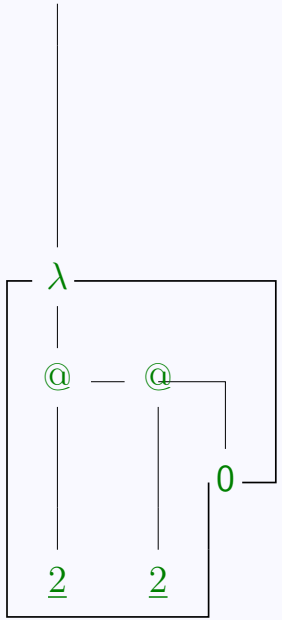
erase argument if matching S

1.7. β-reduction: yields generalised term



successors of subterms!

1.8. β-reduction: extrusion



extrude scopes to yield unary indexed terms

1.9. Generalised terms inductively

$$t \in G\Lambda \ ::= \ 0 \mid \mathsf{S}t \mid \lambda t \mid tt$$

De Bruijn indexed Λ-terms: repeated successors of 0

1.9. Generalised terms inductively

$$t \in G\Lambda \ ::= \ 0 \mid \mathsf{S}t \mid \lambda t \mid tt$$

De Bruijn indexed Λ-terms: repeated successors of 0

$$\frac{\mathsf{S}i \vdash 0}{}\,0 \quad \frac{\mathsf{S}i \vdash \mathsf{S}t}{i \vdash t}\,\mathsf{S} \quad \frac{i \vdash \lambda t}{\mathsf{S}i \vdash t}\,\lambda \quad \frac{i \vdash t_1 t_2}{i \vdash t_1 \quad i \vdash t_2}\,@$$

t well-formed under stack i

1.9. Generalised terms inductively

$$t \in G\Lambda \ ::= \ 0 \mid \mathsf{S}t \mid \lambda t \mid tt$$

De Bruijn indexed Λ-terms: repeated successors of 0

$$\frac{}{\mathsf{S}i \vdash 0}\,0 \quad \frac{\mathsf{S}i \vdash \mathsf{S}t}{i \vdash t}\,\mathsf{S} \quad \frac{i \vdash \lambda t}{\mathsf{S}i \vdash t}\,\lambda \quad \frac{i \vdash t_1 t_2}{i \vdash t_1 \quad i \vdash t_2}\,@$$

t well-formed under stack i

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{0 \vdash \lambda\lambda(\mathsf{S}0)((\mathsf{S}0)0)}\,\lambda}{\mathsf{S}0 \vdash \lambda(\mathsf{S}0)((\mathsf{S}0)0)}\,\lambda}{\mathsf{SS}0 \vdash (\mathsf{S}0)((\mathsf{S}0)0)}\,@}{\dfrac{\mathsf{SS}0 \vdash \mathsf{S}0}{\mathsf{S}0 \vdash 0}\,0}\,\mathsf{S} \quad \dfrac{\mathsf{SS}0 \vdash (\mathsf{S}0)0}{\dfrac{\mathsf{SS}0 \vdash \mathsf{S}0}{\mathsf{S}0 \vdash 0}\,0}\,\mathsf{S} \quad \dfrac{\mathsf{SS}0 \vdash 0}{}\,0}\,@$$

1.10. Replication recursively

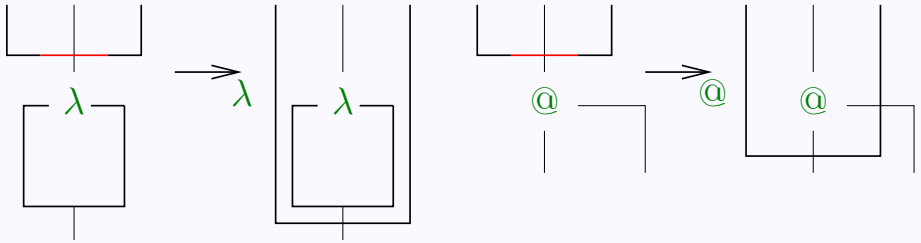
$$(\lambda t)s \to t[s]^0$$

replication $t[s]^i$ of argument s in t at depth i

$$\begin{aligned}
0[s]^0 &= s \quad \text{put argument} \\
0[s]^{Si} &= 0 \\
(St)[s]^0 &= t \quad \text{erase argument} \\
(St)[s]^{Si} &= St[s]^i \\
(\lambda t)[s]^i &= \lambda t[s]^{Si} \\
(t_1 t_2)[s]^i &= t_1[s]^i t_2[s]^i \quad \text{duplicate argument}
\end{aligned}$$

$$\underline{2}\,\underline{2} = (\lambda\lambda(S0)((S0)0))\underline{2} \to \lambda(S\underline{2})((S\underline{2})0)$$

1.11. Extrusion iteratively/recursively



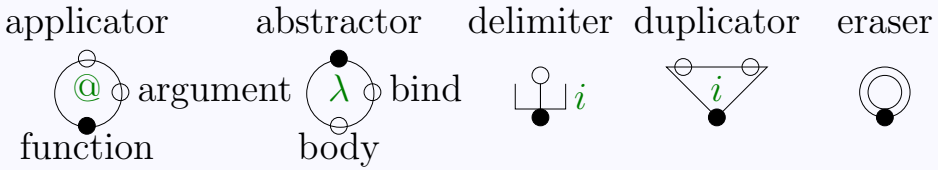
$\mathsf{S}\lambda t \to_\lambda \lambda t^{\mathsf{S}0}$ $\qquad$ $\mathsf{S}(t_1 t_2) \to_@ \mathsf{S}t_1 \mathsf{S}t_2$

minimal lifting t^i

$$
\begin{aligned}
t^0 &= \mathsf{S}t \\
0^{\mathsf{S}i} &= 0 \\
(\mathsf{S}t)^{\mathsf{S}i} &= \mathsf{S}t^i \\
(\lambda t)^{\mathsf{S}i} &= \lambda t^{\mathsf{SS}i} \\
(t_1 t_2)^{\mathsf{S}i} &= t_1^{\mathsf{S}i} t_2^{\mathsf{S}i}
\end{aligned}
$$

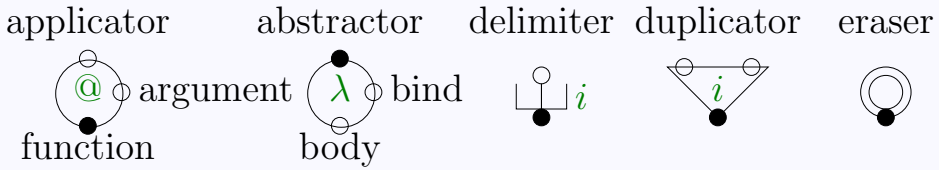
$$\lambda(\mathsf{S}\underline{2})((\mathsf{S}\underline{2})0) \to \lambda\underline{2}(\underline{2}0)$$

2. Interaction net implementation



- ○s are ports
- ●s are principal ports
- i index (default 0)

2. Interaction net implementation



○s are ports

●s are principal ports

i index (default 0)

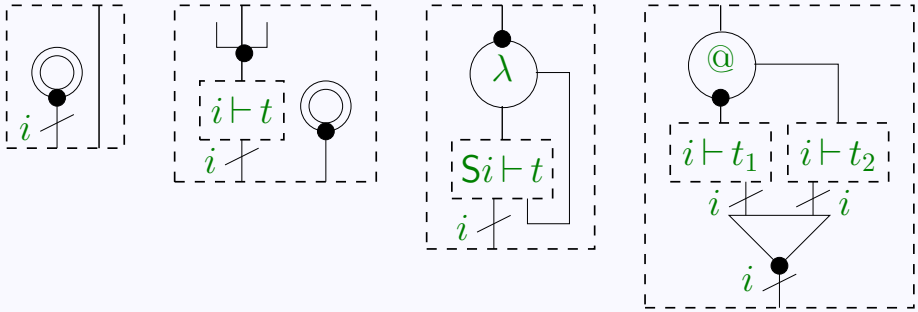
@ applicator, λ abstractor for local β

∇_i duplicator, ◎ eraser for explicit local replication

$\sqcup_i$ scope delimiter for explicit local extrusion

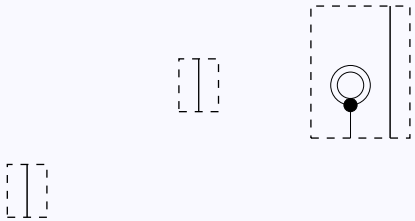
2.1. From terms to nets

$\square : \Lambda \longrightarrow \mathsf{IN}$ recursively maps closed terms to 'closed' nets

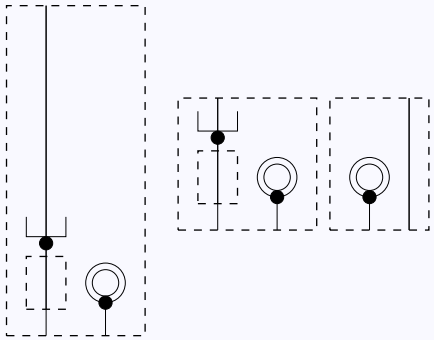


$i \vdash t$ mapped to net with $i + 1$ free ports

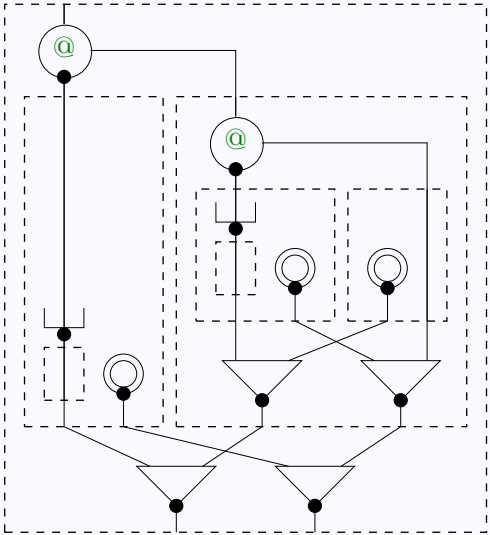
2.2. From terms to nets: translating zero



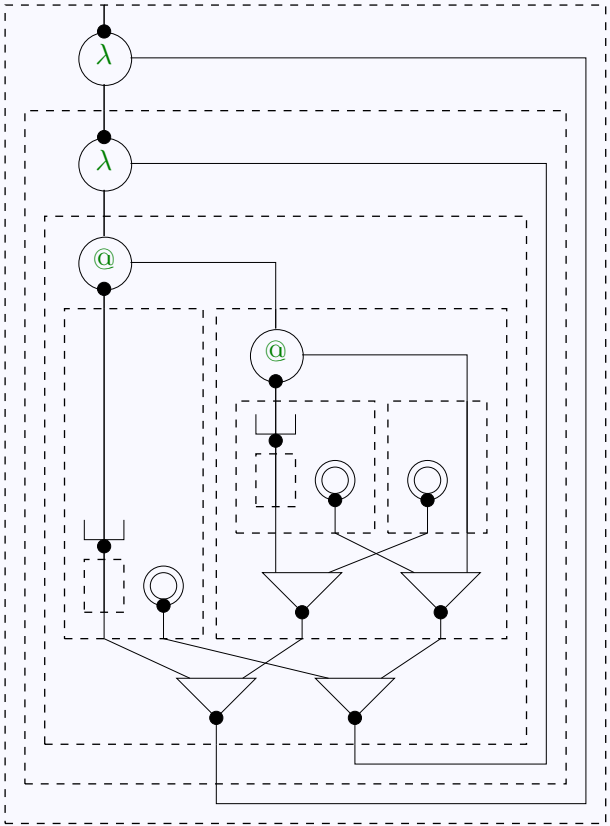
2.3. From terms to nets: translating successor



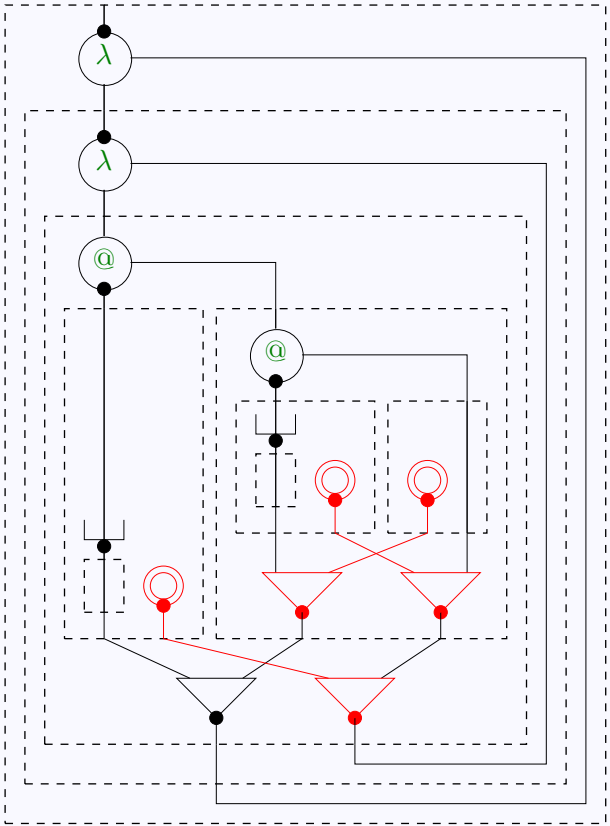
2.4. From terms to nets: translating application



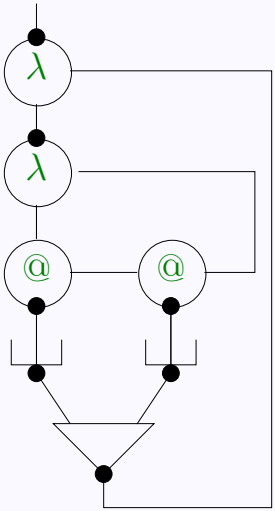
2.5. From terms to nets: translating abstraction



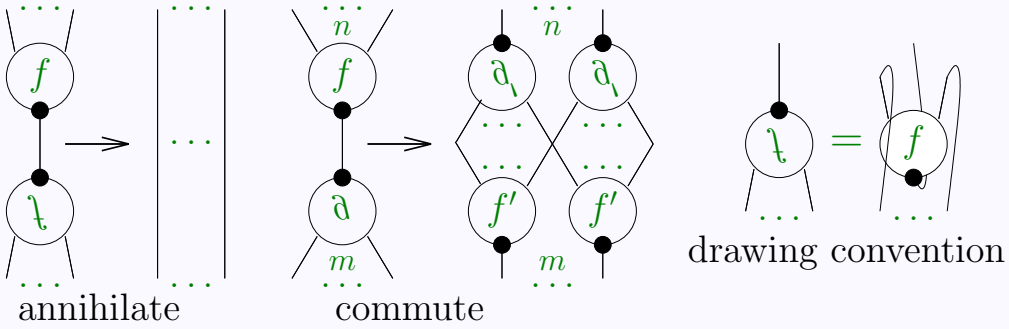
2.6. From terms to nets: useless parts



2.7. From terms to nets: useful part



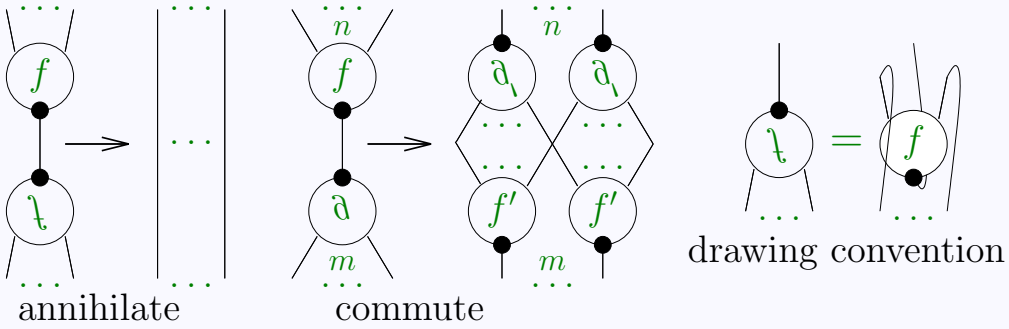
2.8. Net reduction: x-rules



f',g' identical to or updates of f,g (distinct)

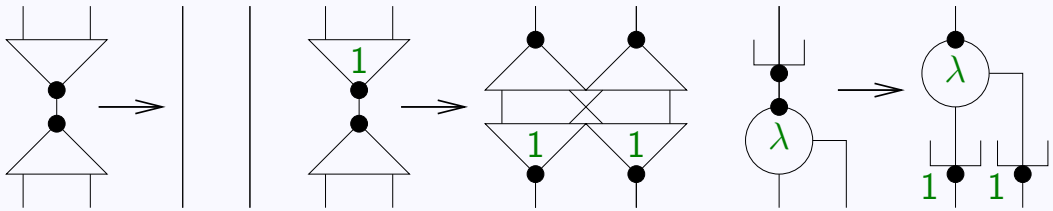
Update: increment of i iff other λ or $\sqcup_j$ $(i \geq j)$

2.8. Net reduction: x-rules

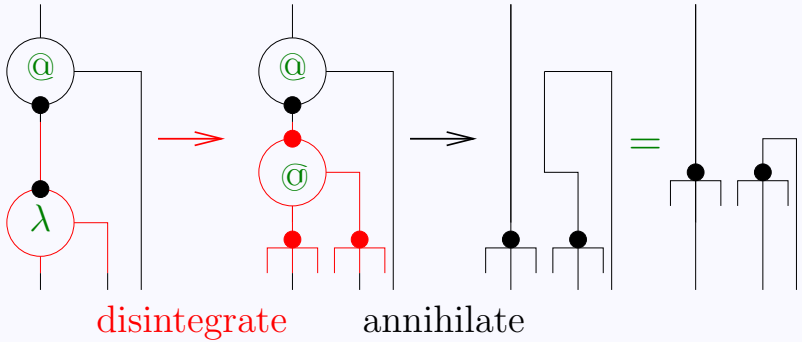


f',g' identical to or updates of f,g (distinct)

Update: increment of i iff other λ or $\sqcup_j$ $(i \geq j)$

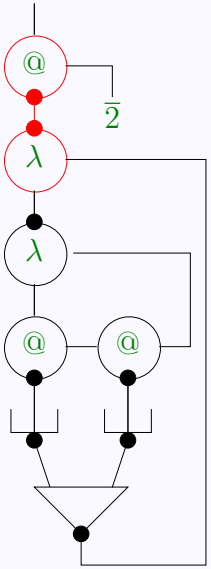


2.9. Net reduction: Beta

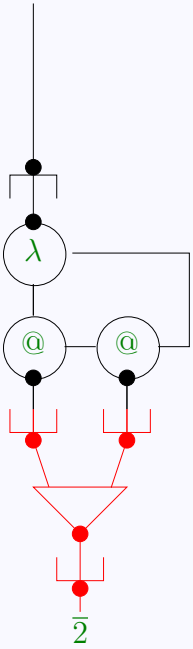


$\mathsf{B} = \mathsf{Beta} + \mathbf{x}$

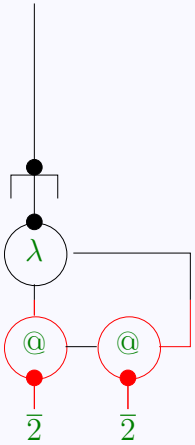
2.10. Net reduction: example Beta-step



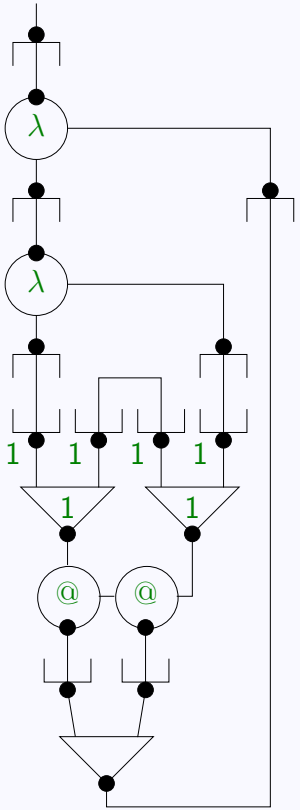
2.11. Net reduction: example x-normalisation



2.12. Net reduction: example B-normalisation



2.13. Net reduction: example normal form



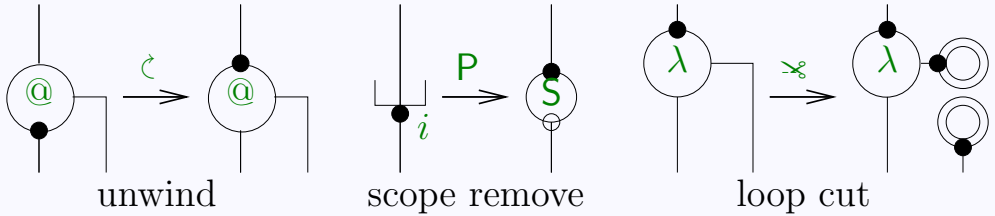
2.14. From nets to terms

Why does this net represent $\underline{4}$?

2.14. From nets to terms

Why does this net represent $\underline{4}$?

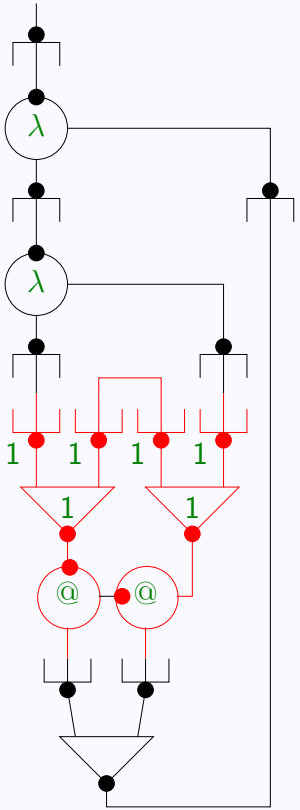
read-back map $\triangle : \mathsf{IN} \rightarrow \Lambda$ in three phases:



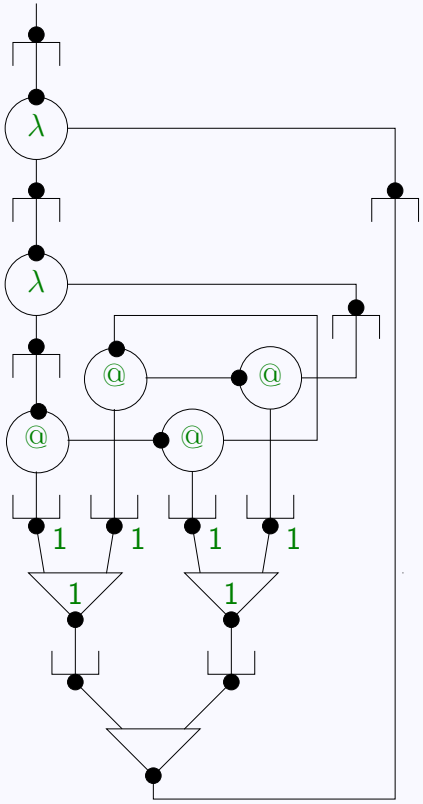
each followed by $\mathbf{x}$-normalisation

1. unwind: replicate,extrude applications
2. scope remove: remove redundant scopes
3. cut loop: elide duplicators yielding tree

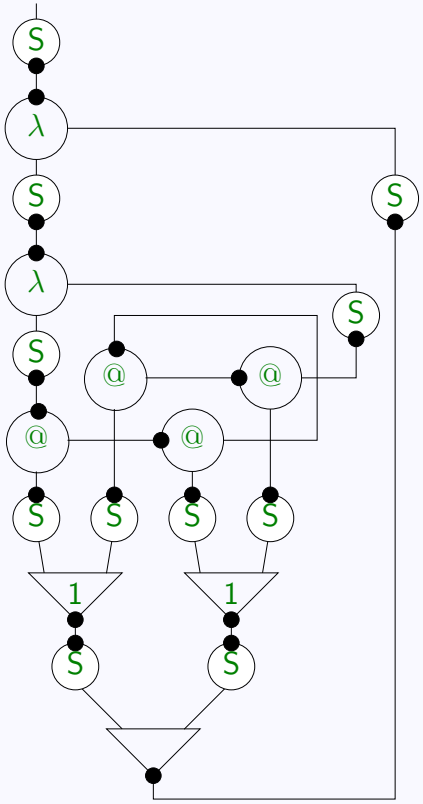
2.15. Read-back: rotating port of application



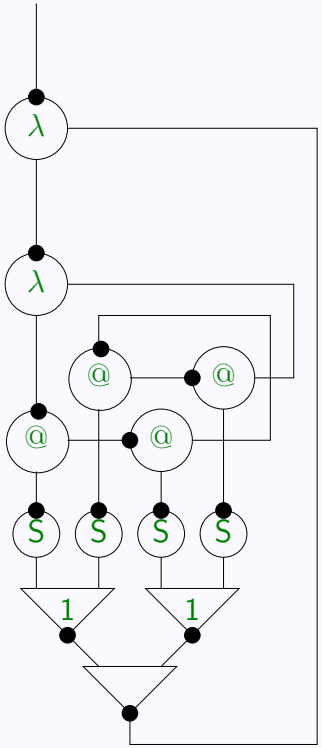
2.16. Read-back: replicate, extrude applications



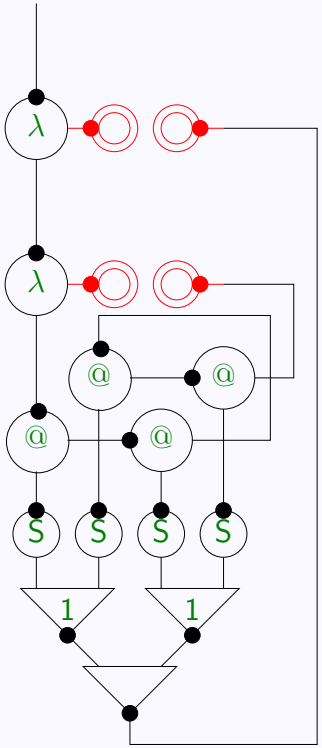
2.17. Read-back: rotating port of delimiter



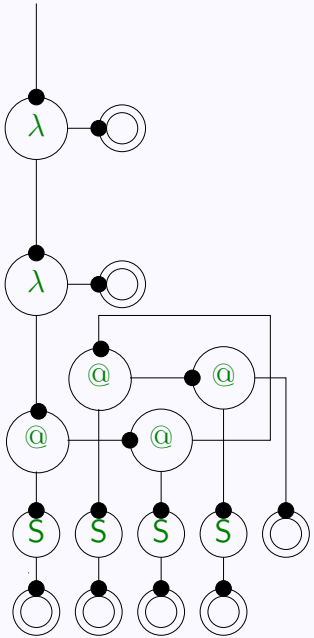
2.18. Read-back: remove redundant scopes



2.19. Read-back: cut loops



2.20. Read-back: elide duplicators



3. Correctness

No syntactic proof yet . . .

3. Correctness

No syntactic proof yet ...

Semantic proof via stack-based read-back

- Invariant under reduction
- Coincides with normal form (tree)

3.1. Push down automaton

Transitions between (ports on) edges of net

$$b \in B \ ::= \ i\vec{\delta} \quad \text{blocks with replication info}$$
$$\ell \in L \ ::= \ b\vec{\ell} \quad \text{levels with scoping info}$$
$$\sigma \in S \ ::= \ i\vec{\ell} \quad \text{stack with read-back info}$$

δ ranges over directors $\{L, R\}$

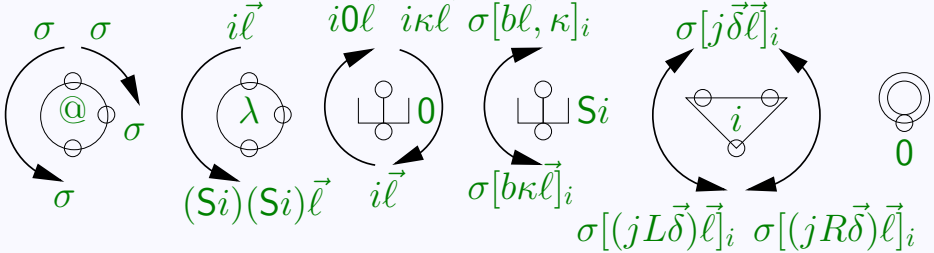
3.1. Push down automaton

Transitions between (ports on) edges of net

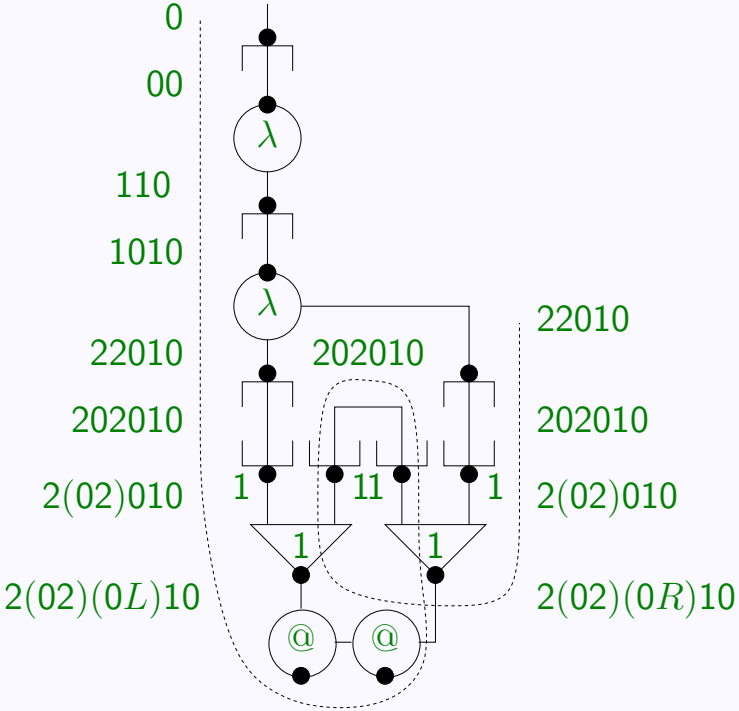
$$b \in B \ ::= \ i\vec{\delta} \quad \text{blocks with replication info}$$
$$\ell \in L \ ::= \ b\vec{\ell} \quad \text{levels with scoping info}$$
$$\sigma \in S \ ::= \ i\vec{\ell} \quad \text{stack with read-back info}$$

δ ranges over directors $\{L, R\}$

Constraints on PDA transitions:



3.2. Example path read-back



So term of shape $\lambda\lambda?(?(?(?0)))$ (in fact $?$s are 1s)

4. Optimality

Implementation is optimal in sense of Lévy

- No needless work
  via outermost strategy
- No double work
  because of local replication

Why?

4. Optimality

Implementation is optimal in sense of Lévy

- No needless work
  via outermost strategy
- No double work
  because of local replication

Why?

Same abstract algorithm as extant implementations

5. Efficiency

- Prototype implementation lambdascope
- Trivial: 1 day of programming (in Java)
- As efficient as (optimised version of) BOHM (outperforms functional languages on same examples)
- Solves oracle problems (brackets/croissants)
- Many possibilities for optimisation
  closedness, types etc.

6. Future work

- Fast implementaton
- Ideal explicit substitution calculus
  (orthogonal, preservation of termination etc.)
- Type system for graphs
- Extension to higher-order rewriting

6. Future work

- Fast implementaton
- Ideal explicit substitution calculus
  (orthogonal, preservation of termination etc.)
- Type system for graphs
- Extension to higher-order rewriting

Attend RTA 2004

Registration is open now