

Home Page

Title Page

Contents



Page 1 of 44

Go Back

Full Screen

Close

Quit

]

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



Contents

1	Goal	4
1.1	Named	5
1.2	De Bruijn indexed	7
1.3	Unary indexed	9
1.4	Scoped	10
1.5	β -reduction	11
1.6	β -reduction: replication	12
1.7	β -reduction: yields generalised term	13
1.8	β -reduction: extrusion	14
1.9	Generalised terms inductively	15
1.10	Replication recursively	16
1.11	Extrusion iteratively/recursively	17
2	Interaction net implementation	18
2.1	From terms to nets	19
2.2	From terms to nets: translating zero	20
2.3	From terms to nets: translating successor	21
2.4	From terms to nets: translating application	22
2.5	From terms to nets: translating abstraction	23
2.6	From terms to nets: useless parts	24
2.7	From terms to nets: useful part	25
2.8	Net reduction: x-rules	26

Home Page

Title Page

Contents



Page 3 of 44

Go Back

Full Screen

Close

Quit

2.9	Net reduction: Beta	27
2.10	Net reduction: example Beta-step	28
2.11	Net reduction: example α -normalisation	29
2.12	Net reduction: example B-normalisation	30
2.13	Net reduction: example normal form	31
2.14	From nets to terms	32
2.15	Read-back: rotating port of application	33
2.16	Read-back: replicate, extrude applications	34
2.17	Read-back: rotating port of delimiter	35
2.18	Read-back: remove redundant scopes	36
2.19	Read-back: cut loops	37
2.20	Read-back: elide duplicators	38
3	Correctness	39
3.1	Push down automaton	40
3.2	Example path read-back	41
4	Optimality	42
5	Efficiency	43
6	Future work	44

Home Page

Title Page

Contents



Page 4 of 44

Go Back

Full Screen

Close

Quit

1. Goal

Extend optimality theory and practice to HRS

Home Page

Title Page

Contents



Page 4 of 44

Go Back

Full Screen

Close

Quit

1. Goal

Extend optimality theory and practice to HRS

Subgoals:

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

Home Page

Title Page

Contents



Page 4 of 44

Go Back

Full Screen

Close

Quit

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

Home Page

Title Page

Contents



Page 5 of 44

Go Back

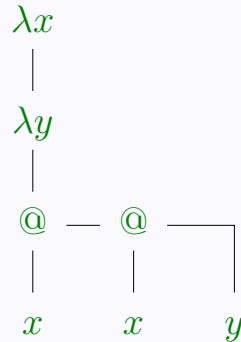
Full Screen

Close

Quit

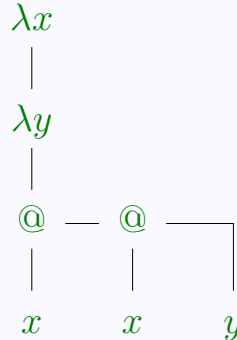
1.1. Named

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

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

Home Page

Title Page

Contents



Page 6 of 44

Go Back

Full Screen

Close

Quit

Home Page

Title Page

Contents



Page 7 of 44

Go Back

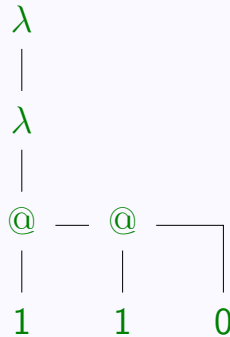
Full Screen

Close

Quit

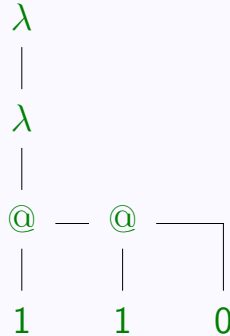
1.2. De Bruijn indexed

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

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

Home Page

Title Page

Contents



Page 8 of 44

Go Back

Full Screen

Close

Quit

Home Page

Title Page

Contents



Page 9 of 44

Go Back

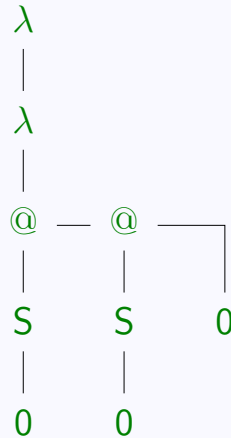
Full Screen

Close

Quit

1.3. Unary indexed

Unary indexed $\underline{2} = \lambda\lambda(S0)((S0)0)$



Home Page

Title Page

Contents



Page 10 of 44

Go Back

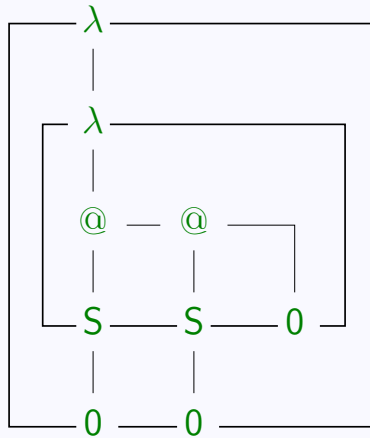
Full Screen

Close

Quit

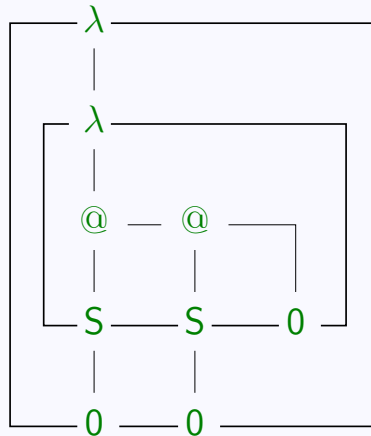
1.4. Scoped

Scoped $\underline{\lambda} = \lambda\lambda(S0)((S0)0)$



1.4. Scoped

Scoped $\underline{\lambda} = \lambda\lambda(S0)((S0)0)$



scoping is matching

binding is match with 0

end-of-scope is match with S

λ -terms are context-free trees

Home Page

Title Page

Contents



Page 11 of 44

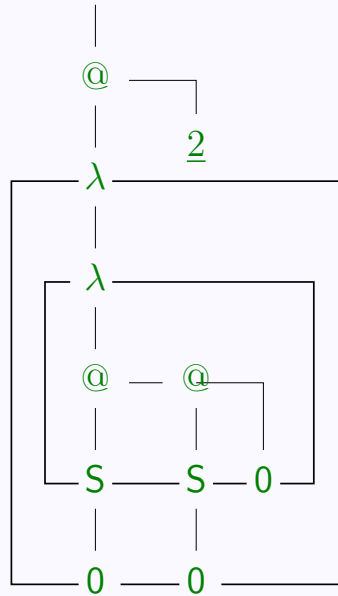
Go Back

Full Screen

Close

Quit

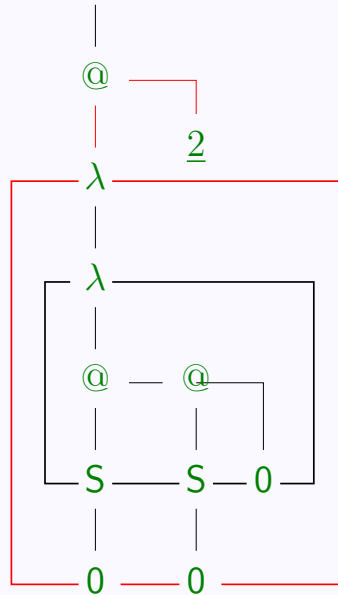
1.5. β -reduction



how to reduce?



1.6. β -reduction: replication



put argument if matching 0

erase argument if matching S

Home Page

Title Page

Contents



Page 13 of 44

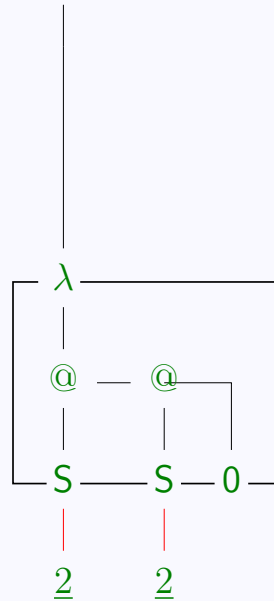
Go Back

Full Screen

Close

Quit

1.7. β -reduction: yields generalised term



successors of subterms!

Home Page

Title Page

Contents



Page 14 of 44

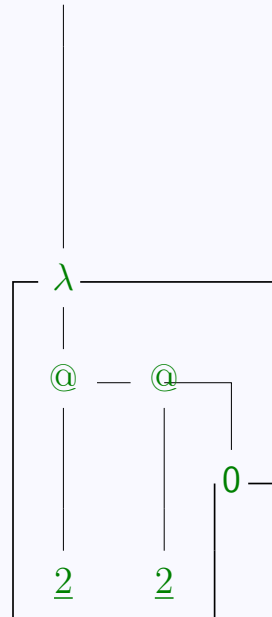
Go Back

Full Screen

Close

Quit

1.8. β -reduction: extrusion



extrude scopes to yield unary indexed terms

[Home Page](#)

[Title Page](#)

[Contents](#)



Page 15 of 44

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

1.9. Generalised terms inductively

$$t \in G\Lambda ::= 0 \mid St \mid \lambda t \mid tt$$

De Bruijn indexed Λ -terms: repeated successors of 0



1.9. Generalised terms inductively

$$t \in G\Lambda ::= 0 \mid St \mid \lambda t \mid tt$$

De Bruijn indexed Λ -terms: repeated successors of 0

$$\frac{Si \vdash 0}{i \vdash 0} 0 \quad \frac{Si \vdash St}{i \vdash t} S \quad \frac{i \vdash \lambda t}{Si \vdash t} \lambda \quad \frac{i \vdash t_1 t_2}{i \vdash t_1 \quad i \vdash t_2} @$$

t well-formed under stack i



1.9. Generalised terms inductively

$$t \in G\Lambda ::= 0 \mid St \mid \lambda t \mid tt$$

De Bruijn indexed Λ -terms: repeated successors of 0

$$\frac{Si \vdash 0}{0} \quad \frac{Si \vdash St}{i \vdash t} S \quad \frac{i \vdash \lambda t}{Si \vdash t} \lambda \quad \frac{i \vdash t_1 t_2}{i \vdash t_1 \quad i \vdash t_2} @$$

t well-formed under stack i

$$\frac{\frac{\frac{0 \vdash \lambda \lambda (S0) ((S0)0)}{S0 \vdash \lambda (S0) ((S0)0)} \lambda}{SS0 \vdash (S0) ((S0)0)} @}{\frac{\frac{SS0 \vdash S0}{S0 \vdash 0} S \quad \frac{\frac{SS0 \vdash (S0)0}{SS0 \vdash S0} @}{\frac{SS0 \vdash S0}{S0 \vdash 0} S} S} S} S \quad \frac{\frac{SS0 \vdash (S0)0}{SS0 \vdash S0} @}{\frac{SS0 \vdash S0}{S0 \vdash 0} S} S} S \quad \frac{SS0 \vdash 0}{0} @$$



1.10. Replication recursively

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

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

$$0[s]^0 = s \quad \text{put argument}$$

$$0[s]^{Si} = 0$$

$$(St)[s]^0 = t \quad \text{erase argument}$$

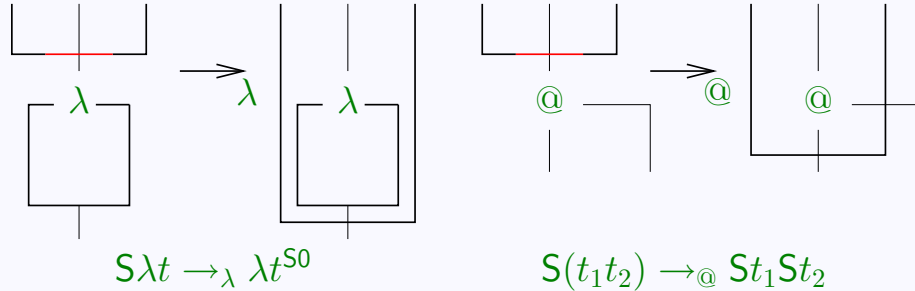
$$(St)[s]^{Si} = St[s]^i$$

$$(\lambda t)[s]^i = \lambda t[s]^{Si}$$

$$(t_1 t_2)[s]^i = t_1[s]^i t_2[s]^i \quad \text{duplicate argument}$$

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

1.11. Extrusion iteratively/recursively



minimal lifting t^i

$$t^0 = St$$

$$0^{Si} = 0$$

$$(St)^{Si} = St^i$$

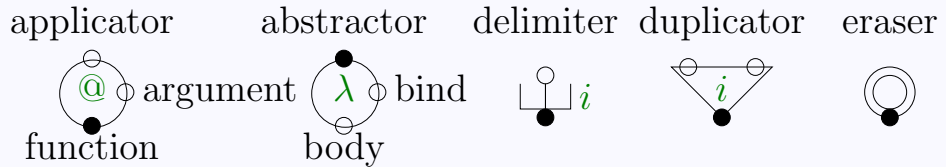
$$(\lambda t)^{Si} = \lambda t^{SSi}$$

$$(t_1 t_2)^{Si} = t_1^{Si} t_2^{Si}$$

$$\lambda(S\underline{2})((S\underline{2})0) \rightarrow \lambda\underline{2}(\underline{2}0)$$



2. Interaction net implementation



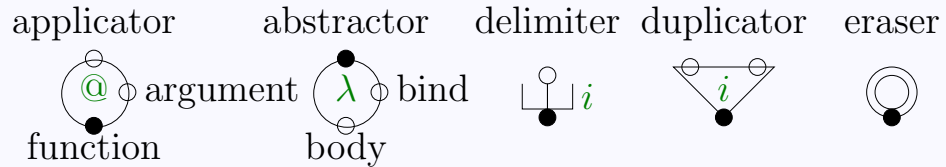
○s are ports

●s are **principal** ports

i index (default 0)



2. Interaction net implementation



○s are ports

●s are **principal** ports

i index (default 0)

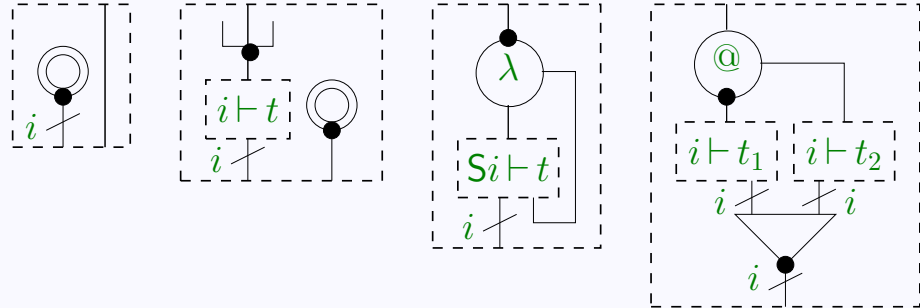
@ **applicator**, λ **abstractor** for local β

∇_i **duplicator**, \odot **eraser** for explicit local replication

\sqcup_i **scope delimiter** for explicit local extrusion

2.1. From terms to nets

$\square : \Lambda \rightarrow \mathbb{N}$ recursively maps closed terms to 'closed' nets



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

Home Page

Title Page

Contents



Page 20 of 44

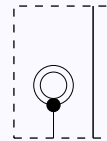
Go Back

Full Screen

Close

Quit

2.2. From terms to nets: translating zero



Home Page

Title Page

Contents



Page 21 of 44

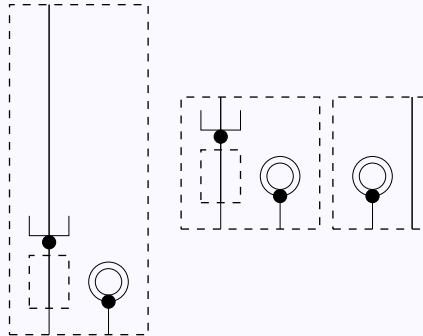
Go Back

Full Screen

Close

Quit

2.3. From terms to nets: translating successor



Home Page

Title Page

Contents



Page 22 of 44

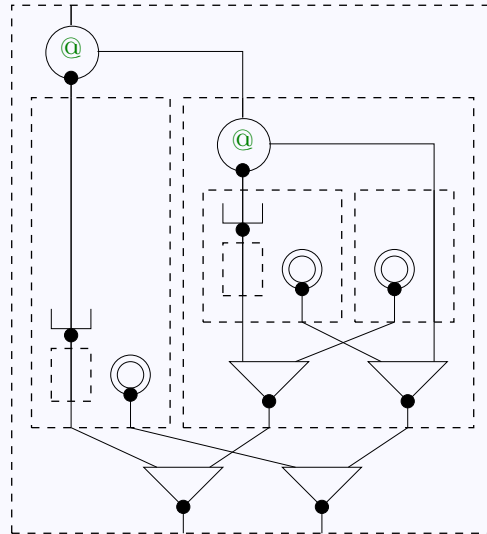
Go Back

Full Screen

Close

Quit

2.4. From terms to nets: translating application



Home Page

Title Page

Contents



Page 23 of 44

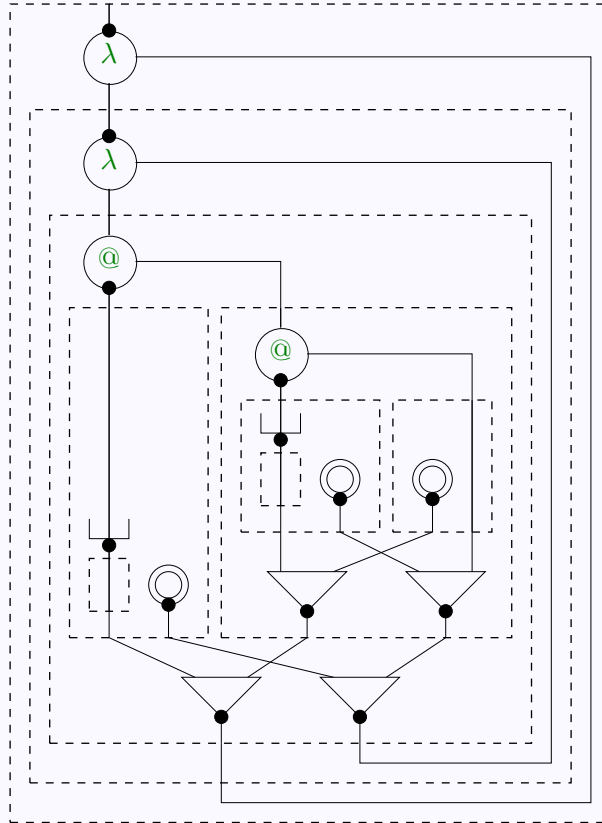
Go Back

Full Screen

Close

Quit

2.5. From terms to nets: translating abstraction



Home Page

Title Page

Contents



Page 24 of 44

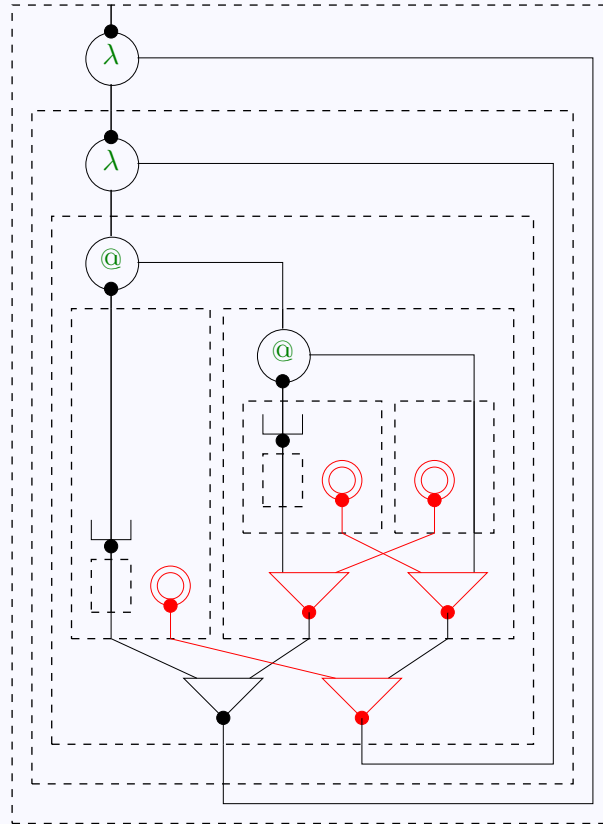
Go Back

Full Screen

Close

Quit

2.6. From terms to nets: useless parts



Home Page

Title Page

Contents



Page 25 of 44

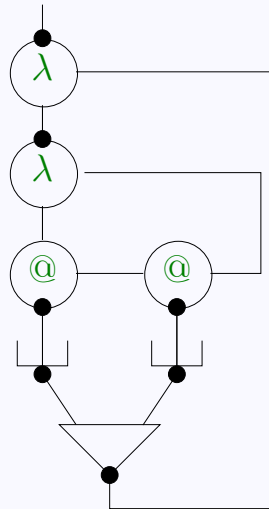
Go Back

Full Screen

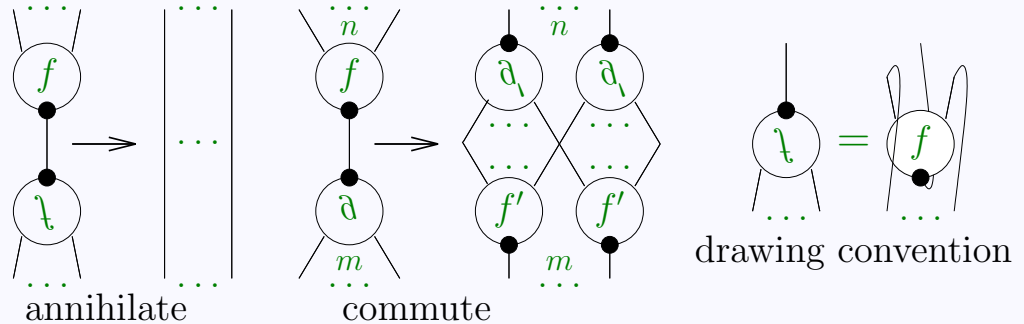
Close

Quit

2.7. From terms to nets: useful part



2.8. Net reduction: x-rules

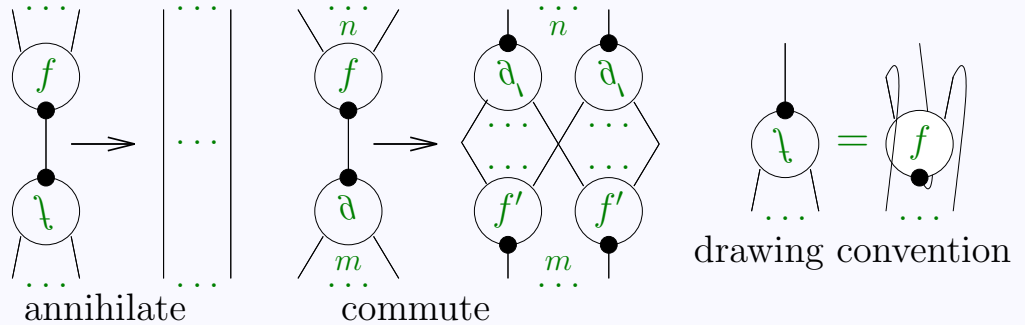


f', g' identical to or updates of f, g (distinct)

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

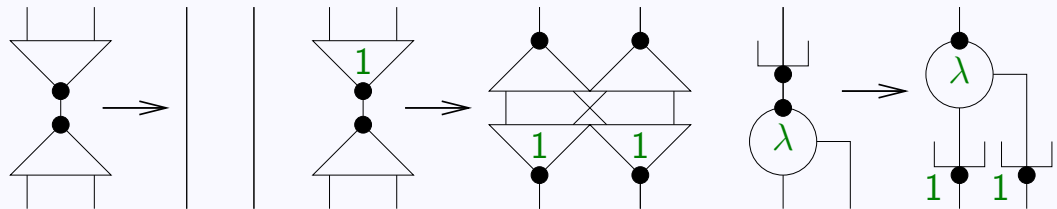


2.8. Net reduction: x-rules



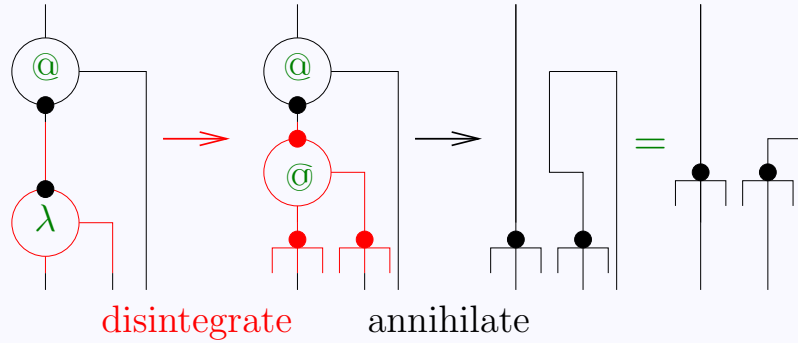
f', g' identical to or updates of f, g (distinct)

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





2.9. Net reduction: Beta



$$B = \text{Beta} + x$$

Home Page

Title Page

Contents



Page 28 of 44

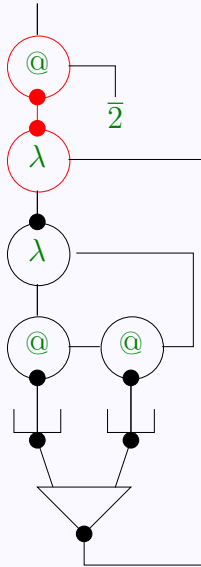
Go Back

Full Screen

Close

Quit

2.10. Net reduction: example Beta-step



Home Page

Title Page

Contents



Page 29 of 44

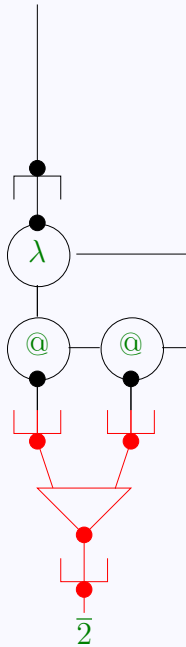
Go Back

Full Screen

Close

Quit

2.11. Net reduction: example x-normalisation



Home Page

Title Page

Contents



Page 30 of 44

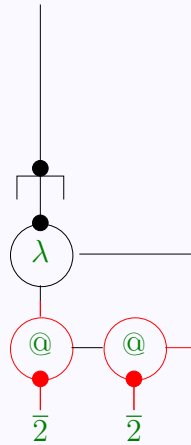
Go Back

Full Screen

Close

Quit

2.12. Net reduction: example B-normalisation



Home Page

Title Page

Contents



Page 31 of 44

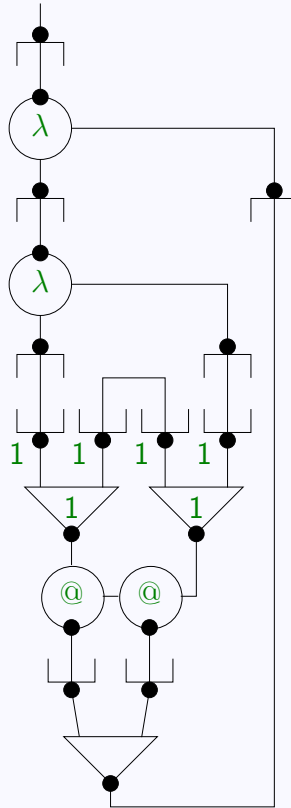
Go Back

Full Screen

Close

Quit

2.13. Net reduction: example normal form



Home Page

Title Page

Contents



Page 32 of 44

Go Back

Full Screen

Close

Quit

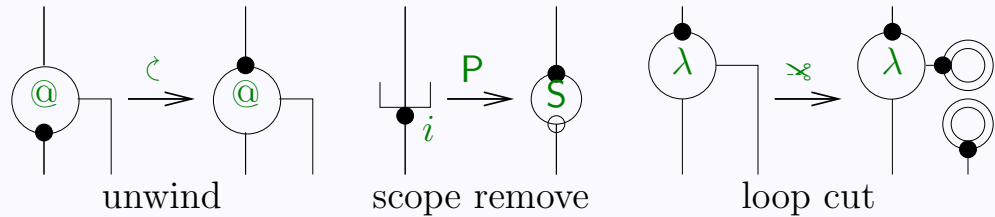
2.14. From nets to terms

Why does this net represent 4?

2.14. From nets to terms

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

read-back map $\Delta : \mathbb{IN} \rightarrow \Lambda$ in three phases:



each followed by **x**-normalisation

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

Home Page

Title Page

Contents



Page 33 of 44

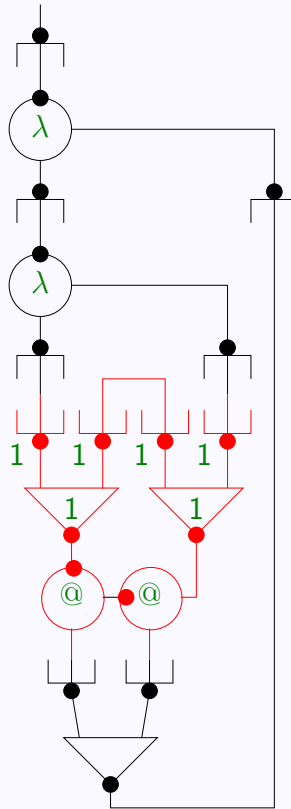
Go Back

Full Screen

Close

Quit

2.15. Read-back: rotating port of application



Home Page

Title Page

Contents



Page 34 of 44

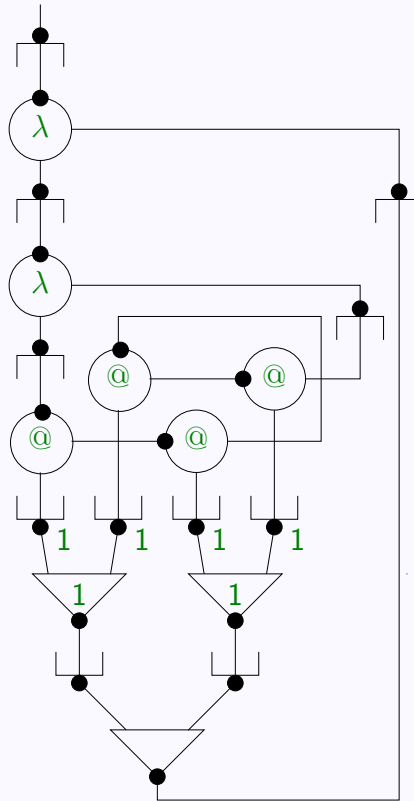
Go Back

Full Screen

Close

Quit

2.16. Read-back: replicate, extrude applications



Home Page

Title Page

Contents



Page 35 of 44

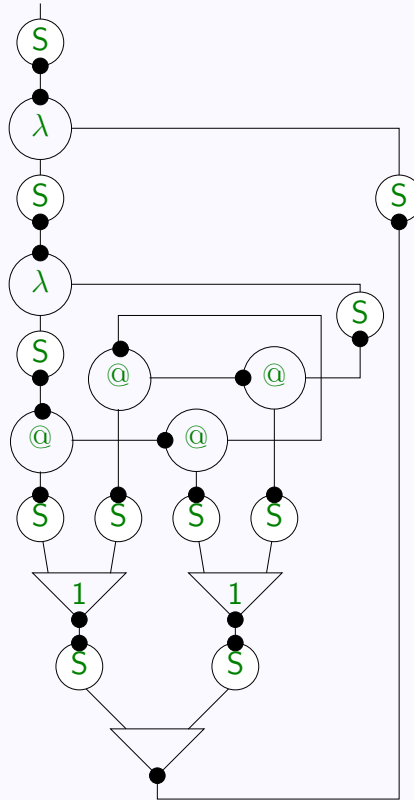
Go Back

Full Screen

Close

Quit

2.17. Read-back: rotating port of delimiter



Home Page

Title Page

Contents



Page 36 of 44

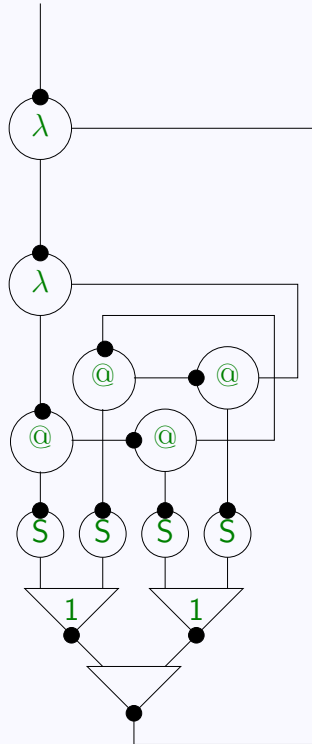
Go Back

Full Screen

Close

Quit

2.18. Read-back: remove redundant scopes



Home Page

Title Page

Contents



Page 37 of 44

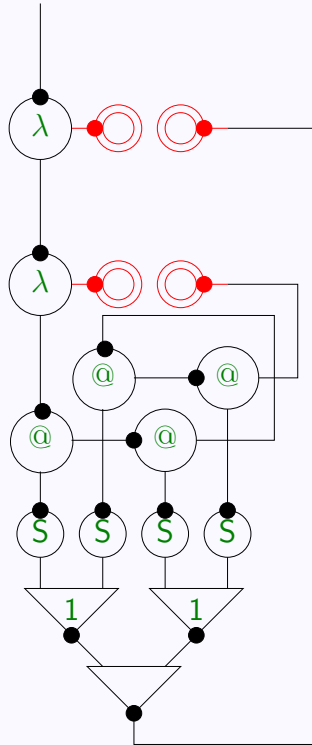
Go Back

Full Screen

Close

Quit

2.19. Read-back: cut loops



Home Page

Title Page

Contents



Page 38 of 44

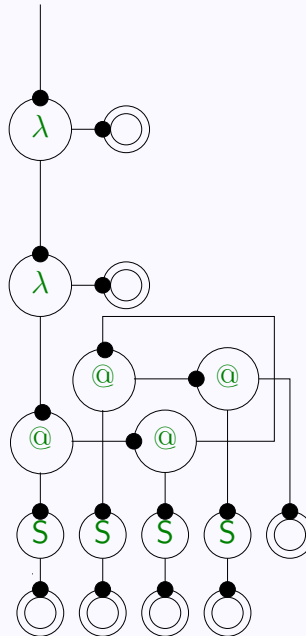
Go Back

Full Screen

Close

Quit

2.20. Read-back: elide duplicators



Home Page

Title Page

Contents



Page 39 of 44

Go Back

Full Screen

Close

Quit

3. Correctness

No syntactic proof yet ...

Home Page

Title Page

Contents



Page 39 of 44

Go Back

Full Screen

Close

Quit

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

$b \in B ::= i\vec{\delta}$ blocks with replication info

$l \in L ::= b\vec{l}$ levels with scoping info

$\sigma \in S ::= i\vec{l}$ stack with read-back info

δ ranges over directors $\{L, R\}$

3.1. Push down automaton

Transitions between (ports on) edges of net

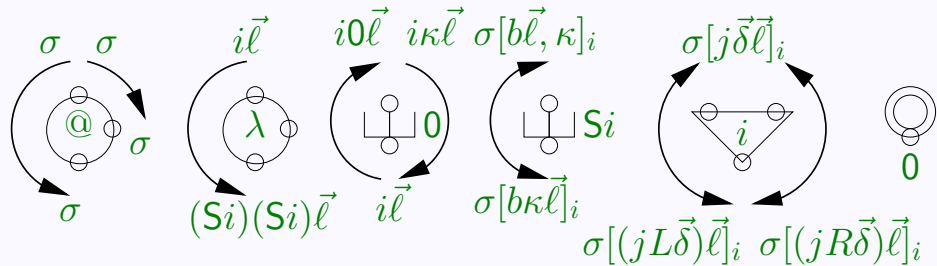
$b \in B ::= i\vec{\delta}$ blocks with replication info

$\ell \in L ::= b\vec{\ell}$ levels with scoping info

$\sigma \in S ::= i\vec{\ell}$ stack with read-back info

δ ranges over directors $\{L, R\}$

Constraints on PDA transitions:



Home Page

Title Page

Contents



Page 41 of 44

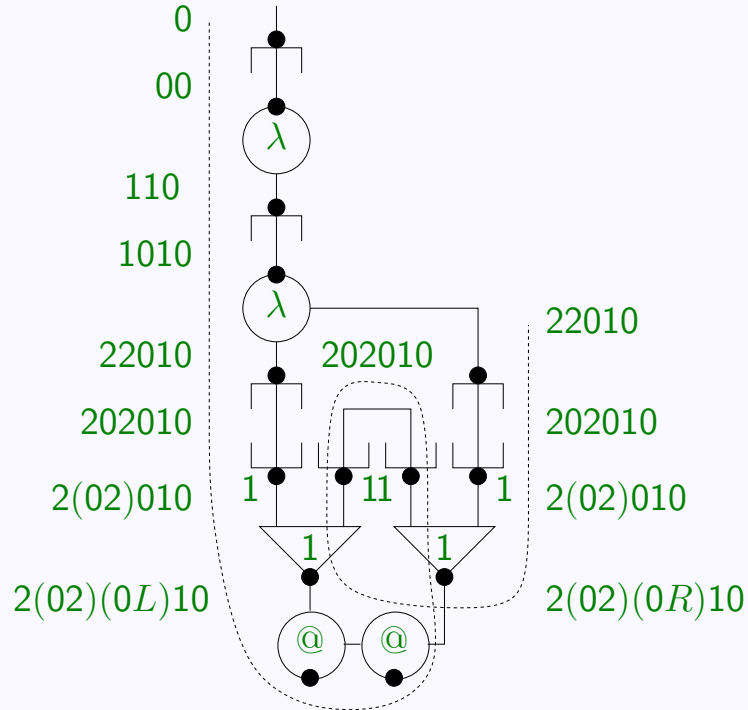
Go Back

Full Screen

Close

Quit

3.2. Example path read-back



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

Home Page

Title Page

Contents



Page 42 of 44

Go Back

Full Screen

Close

Quit

4. Optimality

Implementation is **optimal** in sense of Lévy

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

Why?

Home Page

Title Page

Contents



Page 42 of 44

Go Back

Full Screen

Close

Quit

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

Home Page

Title Page

Contents



Page 43 of 44

Go Back

Full Screen

Close

Quit

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.

[Home Page](#)

[Title Page](#)

[Contents](#)



Page 44 of 44

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

6. Future work

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

Home Page

Title Page

Contents



Page 44 of 44

Go Back

Full Screen

Close

Quit

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