# Properties of Needed Strategies

Vincent van Oostrom

Theoretical Philosophy
Utrecht University
The Netherlands

July 5, 2006

## Neededness Intuition

## Four formalizations of neededness

- Permutation
- Standardisation
- Labelling
- Tracing

# Motivation for Neededness

Do not contract redexes not needed to reach result

# Typical Result

### Theorem

*The needed strategy is normalising for combinatory logic.*

$$
\begin{aligned}
(K \cdot x) \cdot y &\longrightarrow x \\
((S \cdot x) \cdot y) \cdot z &\longrightarrow (x \cdot z) \cdot (y \cdot z)
\end{aligned}
$$

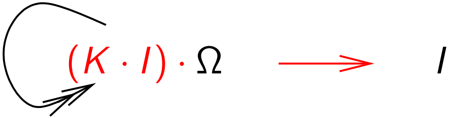
# Typical Result

### Theorem

*The* *needed* *strategy is normalising for combinatory logic.*

$$
\begin{array}{rcl}
(K \cdot x) \cdot y & \rightarrow & x \\
((S \cdot x) \cdot y) \cdot z & \rightarrow & (x \cdot z) \cdot (y \cdot z)
\end{array}
$$

### Example




$$
\begin{array}{rcl}
I & = & (S \cdot K) \cdot K \\
\Omega & = & ((S \cdot I) \cdot I) \cdot ((S \cdot I) \cdot I)
\end{array}
$$

# Needed redexes need not exist

# Needed redexes need not exist

$$\begin{aligned} a &\rightarrow b \\ f(b,x) &\rightarrow c \\ f(x,b) &\rightarrow c \end{aligned}$$

$f(a, a)$ not needed: $f(a, a) \rightarrow f(a, b) \rightarrow c$

# Needed redexes need not exist

$$\begin{array}{rcl} a & \to & b \\ f(b,x) & \to & c \\ f(x,b) & \to & c \end{array}$$

$f(a, a)$ not needed: $f(a, a) \to f(b, a) \to c$

# Needed redexes need not exist

$$
\begin{array}{rcl}
a & \to & b \\
f(b, x) & \to & c \\
f(x, b) & \to & c
\end{array}
$$

$f(a, a)$ no needed redex!

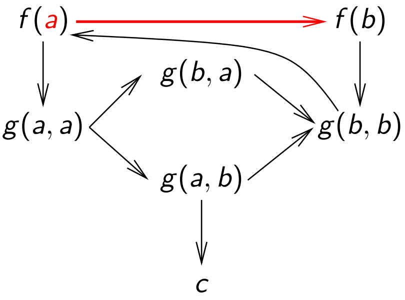
# Needed reduction may be counterproductive

# Needed reduction may be counterproductive

$$\begin{aligned} a &\rightarrow b \\ f(x) &\rightarrow g(x, x) \\ g(a, b) &\rightarrow c \\ g(b, b) &\rightarrow f(a) \end{aligned}$$

# Needed reduction may be counterproductive

$$
\begin{array}{rcl}
a & \to & b \\
f(x) & \to & g(x,x) \\
g(a,b) & \to & c \\
g(b,b) & \to & f(a)
\end{array}
$$

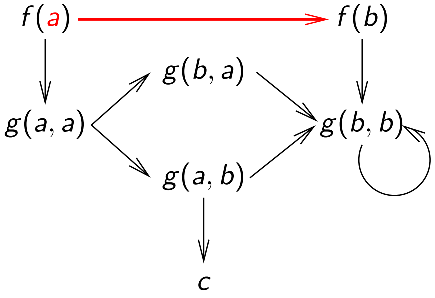


# Needed reduction may lose normalisation

$$\begin{aligned} a &\longrightarrow b \\ f(x) &\longrightarrow g(x,x) \\ g(a,b) &\longrightarrow c \\ g(b,b) &\longrightarrow g(b,b) \end{aligned}$$

# Needed reduction may lose normalisation

$$\begin{aligned} a &\to b \\ f(x) &\to g(x,x) \\ g(a,b) &\to c \\ g(b,b) &\to g(b,b) \end{aligned}$$



# When is needed reduction interesting?

# When is needed reduction interesting?

for orthogonal term rewriting systems

- combinatory logic

# When is needed reduction interesting?

for orthogonal term rewriting systems

- combinatory logic
- $\lambda$-calculus

# When is needed reduction interesting?

for orthogonal term rewriting systems

- combinatory logic
- $\lambda$-calculus
- primitive recursion

# When is needed reduction interesting?

for orthogonal term rewriting systems

- combinatory logic
- $\lambda$-calculus
- primitive recursion
- functional programming

# When is needed reduction interesting?

for orthogonal term rewriting systems

- combinatory logic
- $\lambda$-calculus
- primitive recursion
- functional programming
- ...

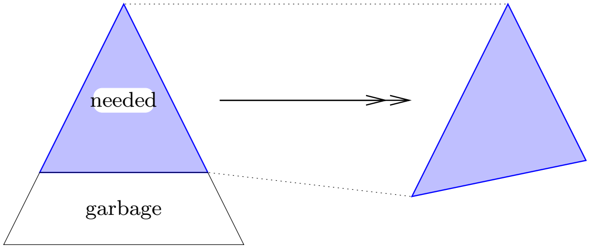
# When is needed reduction interesting?

for orthogonal term rewriting systems

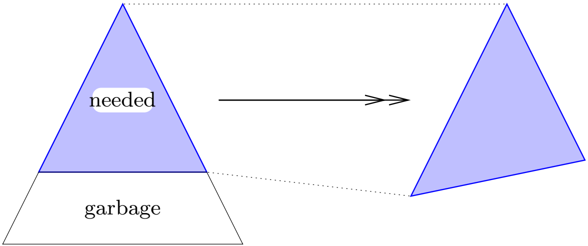
- combinatory logic
- $\lambda$-calculus
- primitive recursion
- functional programming
- ...

Intuition: reductions to normal form contract same redexes
(only order of contraction differs)

# Neededness intuition



# Neededness intuition




Reduction to normal form splits term into
needed top and garbage bottom

# Formalizing intuition

# Formalizing intuition

1. Permutation

# Formalizing intuition

1. Permutation
2. Standardisation

# Formalizing intuition

1. Permutation
2. Standardisation
3. Labelling

# Formalizing intuition

1. Permutation
2. Standardisation
3. Labelling
4. Tracing

# Neededness via permutation

### Definition

Redex not needed if permutation equivalent to reduction where redex is not contracted.

# Neededness via permutation

### Definition

Redex not needed if permutation equivalent to reduction where redex is not contracted.

$$(K \cdot I) \cdot \Omega \to^* (K \cdot I) \cdot \Omega \to I$$

permutation equivalent to

$$(K \cdot I) \cdot \Omega \to^* I$$

# Neededness via permutation

## Definition

Redex not needed if permutation equivalent to reduction where redex is not contracted.

$$(K \cdot I) \cdot \Omega \to^* (K \cdot I) \cdot \Omega \to I$$

permutation equivalent to

$$(K \cdot I) \cdot \Omega \to^* I$$

- $\vartheta(x, y) : (K \cdot x) \cdot y \to x$
- $\phi : \Omega \to^* \Omega$

# Neededness via permutation

### Definition

Redex not needed if permutation equivalent to reduction where redex is not contracted.

$$(K \cdot I) \cdot \Omega \to^* (K \cdot I) \cdot \Omega \to I$$

permutation equivalent to

$$(K \cdot I) \cdot \Omega \to^* I$$

- $\vartheta(x, y) : (K \cdot x) \cdot y \to x$
- $\phi : \Omega \to^* \Omega$

$((K \cdot I) \cdot \phi) \circ \vartheta(I, \Omega)$ permutation equivalent to $\vartheta(I, \Omega)$

# Permutation equivalence on proof terms

$$
\begin{array}{rcl}
1 \circ \phi & \approx & \phi \\
\phi \circ 1 & \approx & \phi \\
(\phi \circ \psi) \circ \chi & \approx & \phi \circ (\psi \circ \chi) \\
f(\phi_1, \ldots, \phi_n) \circ f(\psi_1, \ldots, \psi_n) & \approx & f(\phi_1 \circ \psi_1, \ldots, \phi_n \circ \psi_n) \\
\varrho(\phi_1, \ldots, \phi_n) & \approx & l(\phi_1, \ldots, \phi_n) \circ \varrho(t_1, \ldots, t_n) \\
\varrho(\phi_1, \ldots, \phi_n) & \approx & \varrho(s_1, \ldots, s_n) \circ r(\phi_1, \ldots, \phi_n)
\end{array}
$$

where $\varrho : l \to r$ and $\phi_i : s_i \to^* t_i$

# Permutation equivalence on proof terms

$$
\begin{array}{rcl}
1 \circ \phi & \approx & \phi \\
\phi \circ 1 & \approx & \phi \\
(\phi \circ \psi) \circ \chi & \approx & \phi \circ (\psi \circ \chi) \\
f(\phi_1, \ldots, \phi_n) \circ f(\psi_1, \ldots, \psi_n) & \approx & f(\phi_1 \circ \psi_1, \ldots, \phi_n \circ \psi_n) \\
\varrho(\phi_1, \ldots, \phi_n) & \approx & l(\phi_1, \ldots, \phi_n) \circ \varrho(t_1, \ldots, t_n) \\
\varrho(\phi_1, \ldots, \phi_n) & \approx & \varrho(s_1, \ldots, s_n) \circ r(\phi_1, \ldots, \phi_n)
\end{array}
$$

where $\varrho : \ l \to r$ and $\phi_i : \ s_i \to^* t_i$

Example

$((K \cdot I) \cdot \phi) \circ \vartheta(I, \Omega) \approx \vartheta(I, \phi) \approx \vartheta(I, \Omega) \circ I \approx \vartheta(I, \Omega)$

# Neededness via standardisation

### Definition

Redex not needed if redex not contracted
in reduction obtained by removing anti-standard pairs

# Neededness via standardisation

### Definition

Redex not needed if redex not contracted
in reduction obtained by removing anti-standard pairs

Removal of anti-standard pair = oriented permutation equivalence:

$$l(\phi_1, \ldots, \phi_n) \circ \varrho(t_1, \ldots, t_n) \quad \Rightarrow \quad \varrho(s_1, \ldots, s_n) \circ r(\phi_1, \ldots, \phi_n)$$

# Neededness via standardisation

**Definition**

Redex not needed if redex not contracted
in reduction obtained by removing anti-standard pairs

Removal of anti-standard pair = oriented permutation equivalence:

$$l(\phi_1, \ldots, \phi_n) \circ \varrho(t_1, \ldots, t_n) \quad \Rightarrow \quad \varrho(s_1, \ldots, s_n) \circ r(\phi_1, \ldots, \phi_n)$$

**Example**

$((K \cdot I) \cdot \phi) \circ \vartheta(I, \Omega) \Rightarrow \vartheta(I, \Omega) \circ I \approx \vartheta(I, \Omega)$

**Theorem**

*Removing anti-standard pairs terminates*

**Proof.**

Like sorting by inversions but more difficult (duplication) □

# Neededness via labelling

**Definition**

Redex not needed if labelling it yields no labels in target

**Example**

labelling $\Omega$ by underlining in:

$$(K \cdot I) \cdot \Omega \rightarrow^* (K \cdot I) \cdot \Omega \rightarrow I$$

does not give rise to labels in the target:

$$(K \cdot I) \cdot \underline{\Omega} \rightarrow^* (K \cdot I) \cdot \underline{\Omega} \rightarrow I$$

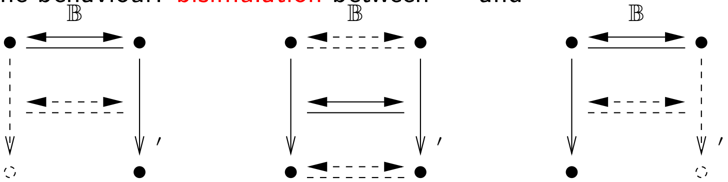
# Labelling ARSs

Attach information; behaviour should be the same

# Labelling ARSs

Attach information; behaviour should be the same

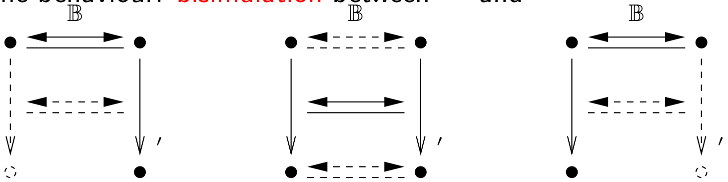
Same behaviour: bisimulation between $\rightarrow$ and $\rightarrow'$



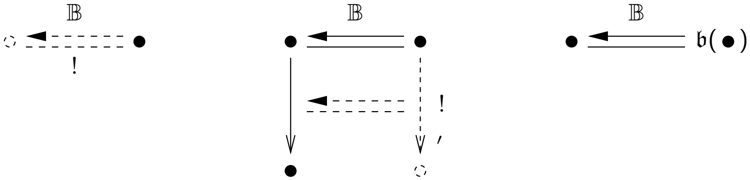
# Labelling ARSs

Attach information; behaviour should be the same

Same behaviour: bisimulation between $\to$ and $\to'$




Attach information: labelling of $\to$ to $\to'$



# Labelled ARSs

Example

- Semantic labelling of TRSs

# Labelled ARSs

**Example**

- Semantic labelling of TRSs
- Typing untyped $\lambda$-terms

# Labelled ARSs

### Example

- Semantic labelling of TRSs
- Typing untyped $\lambda$-terms
- Labelling atoms of molecules in chemical reactions

# Labelled ARSs

### Example

- Semantic labelling of TRSs
- Typing untyped $\lambda$-terms
- Labelling atoms of molecules in chemical reactions
- ...

### Lemma

# Labelled ARSs

## Example

- Semantic labelling of TRSs
- Typing untyped $\lambda$-terms
- Labelling atoms of molecules in chemical reactions
- ...

## Lemma

- *termination preserved and reflected by bisimulation*

# Labelled ARSs

## Example

- Semantic labelling of TRSs
- Typing untyped $\lambda$-terms
- Labelling atoms of molecules in chemical reactions
- ...

## Lemma

- *termination preserved and reflected by bisimulation*
- *normalisation preserved and reflected by bisimulation*

# Labelled ARSs

## Example

- Semantic labelling of TRSs
- Typing untyped $\lambda$-terms
- Labelling atoms of molecules in chemical reactions
- ...

## Lemma

- *termination preserved and reflected by bisimulation*
- *normalisation preserved and reflected by bisimulation*
- *confluence reflected by labelling*

# Labelling TRSs

# Labelling TRSs

Labelling of signature, lifting this to rules, steps

# Labelling TRSs

Labelling of signature, lifting this to rules, steps

Example

Semantic labelling

# Hyland–Wadsworth labelling

# Hyland–Wadsworth labelling

Label symbol with its creation level

Example

$\varrho : \; a \longrightarrow f(a)$

# Hyland–Wadsworth labelling

Label symbol with its creation level

Example

$\varrho : \ a \rightarrow f(a)$

$\varrho_{a^i} : \ a^i \rightarrow f^{i+1}(a^{i+1})$, for all $i \in \mathbb{N}$

# Hyland–Wadsworth labelling

Label symbol with its creation level

Example

$\varrho : \ a \rightarrow f(a)$

$\varrho_{a^i} : \ a^i \rightarrow f^{i+1}(a^{i+1})$, for all $i \in \mathbb{N}$

$$\mathcal{R} : \ a \rightarrow f(a) \rightarrow f(f(a)) \rightarrow f(f(f(a))) \rightarrow \cdots$$

# Hyland–Wadsworth labelling

Label symbol with its creation level

Example

$\varrho : \ a \to f(a)$
$\varrho_{a^i} : \ a^i \to f^{i+1}(a^{i+1})$, for all $i \in \mathbb{N}$

$$\mathcal{R} : \ a \to f(a) \to f(f(a)) \to f(f(f(a))) \to \cdots$$

$$\mathfrak{HW}(\mathcal{R}) : \ a^0 \to f^1(a^1) \to f^1(f^2(a^2)) \to f^1(f^2(f^3(a^3))) \to \cdots$$

Theorem

*Upper bounding creation level implies termination*

Proof.

use recursive path orders □

# Hyland–Wadsworth vs. Arts–Giesl

**Lemma**

*If non-terminating, there's an infinite creation chain*

**Proof.**

zoom-in □

**Corollary**

*If non-terminating, then no upper bound on creation level*

**Corollary**

*If non-terminating, then infinite dependency chain*

# Neededness via Tracing

**Definition**

Redex not needed if its positions do not trace to target

# Neededness via Tracing

### Definition

Redex not needed if its positions do not trace to target

### Example

$\Omega$ does not trace to target in

$$(K \cdot I) \cdot \Omega \to^* (K \cdot I) \cdot \Omega \to I$$

# Neededness via Tracing

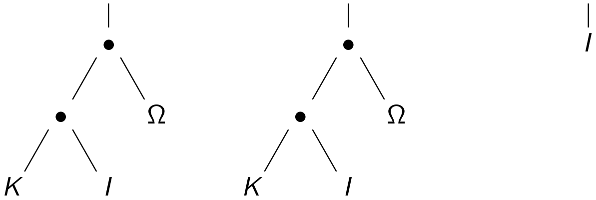
## Definition

Redex not needed if its positions do not trace to target

## Example

$\Omega$ does not trace to target in

$$(K \cdot I) \cdot \Omega \rightarrow^* (K \cdot I) \cdot \Omega \rightarrow I$$



# Neededness via Tracing

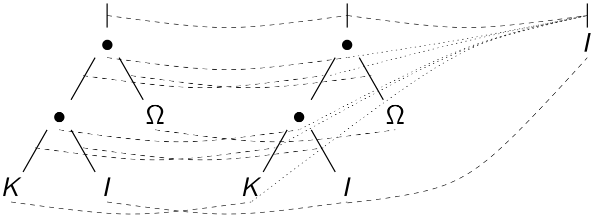
### Definition

Redex not needed if its positions do not trace to target

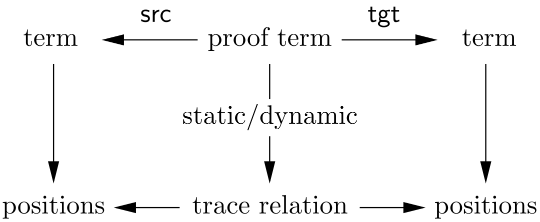
### Example

$\Omega$ does not trace to target in

$$(K \cdot I) \cdot \Omega \to^* (K \cdot I) \cdot \Omega \to I$$



# Tracing as proof term algebra



# Equivalence of notions of neededness

# Equivalence of notions of neededness

**Theorem**

*Permutation non-needed* $\Leftarrow$ *Standardisation non-needed*

**Proof.**

Standardisation is permutation □

# Equivalence of notions of neededness

**Theorem**

*Standardisation non-needed* $\Leftarrow$ *Labelling non-needed*

**Proof.**

Permutation does not change labelling of target

Contracting any labelled redex would label target □

# Equivalence of notions of neededness

**Theorem**

*Labelling non-needed* $\Leftarrow$ *Tracing non-needed*

**Proof.**

Labels trace (internal vs. external) □

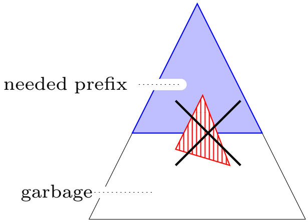
# Equivalence of notions of neededness

### Theorem

*Tracing non-needed* $\Leftarrow$ *Permutation non-needed*

### Proof.

Tracing invariant wrt. permutation (redex-pattern closed)




□

# Can needed redexes be computed?

# Can needed redexes be computed?

No

# Can needed redexes be computed?

No

How to evaluate $g(s_1, s_2, s_3)$ in CL + Gustave's TRS?

$$
\begin{aligned}
g(a, b, x) &\rightarrow c \\
g(x, a, b) &\rightarrow c \\
g(b, x, a) &\rightarrow c
\end{aligned}
$$

(recall: word problem for CL is undecidable)

# Neededness via externality

### Definition

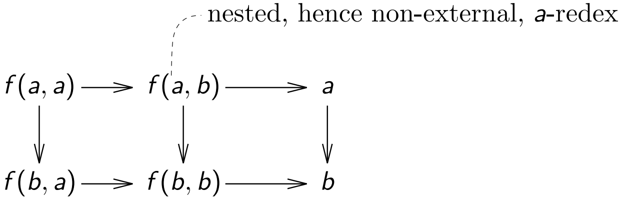
Redex is external if outermost until contracted

# Neededness via externality

## Definition

Redex is external if outermost until contracted

TRS: $a \to b$, $f(x, b) \to x$

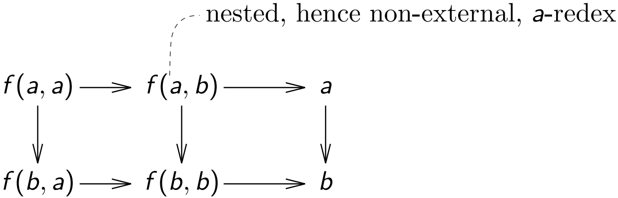


# Neededness via externality

## Definition

Redex is external if outermost until contracted

TRS: $a \to b$, $f(x, b) \to x$




## Lemma

*For orthogonal fully-extended HRSs*
*every reducible term contains an external redex*

## Proof.

Idea: cannot mutually erase each other □

# Computable cases of neededness

**Lemma**

*External redexes are needed (needed strategy is strategy!)*

**Proof.**

Trivial □

- combinatory logic: leftmost-outermost redex external

# Computable cases of neededness

**Lemma**
*External redexes are needed (needed strategy is strategy!)*

**Proof.**
Trivial □

- combinatory logic: leftmost-outermost redex external
- $\lambda$-calculus: idem

# Computable cases of neededness

**Lemma**
*External redexes are needed (needed strategy is strategy!)*

**Proof.**
Trivial □

- combinatory logic: leftmost-outermost redex external
- $\lambda$-calculus: idem
- left-normal orthogonal fully-extended HRSs: idem

# Computable cases of neededness

**Lemma**

*External redexes are needed (needed strategy is strategy!)*

**Proof.**

Trivial □

- combinatory logic: leftmost-outermost redex external
- $\lambda$-calculus: idem
- left-normal orthogonal fully-extended HRSs: idem
- sequential TRSs (but sequentiality not decidable)

# Computable cases of neededness

Lemma

*External redexes are needed (needed strategy is strategy!)*

Proof.

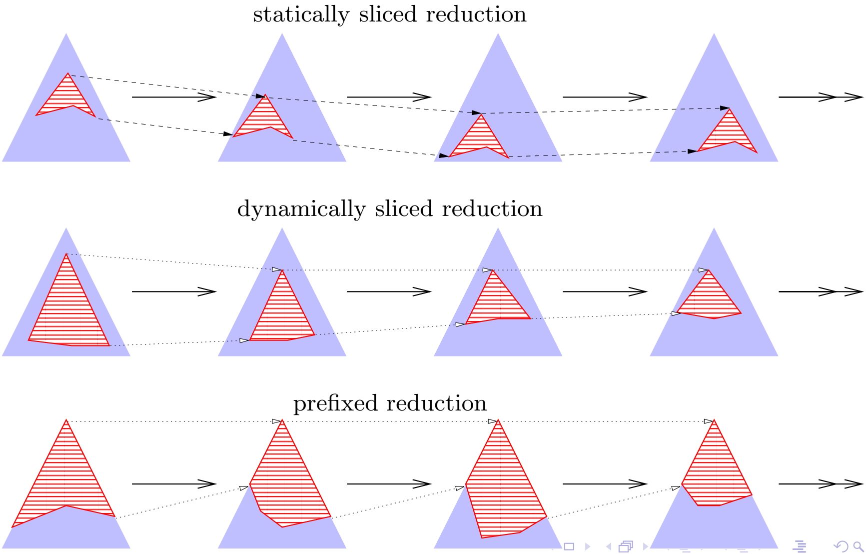
Trivial ☐

- combinatory logic: leftmost-outermost redex external
- $\lambda$-calculus: idem
- left-normal orthogonal fully-extended HRSs: idem
- sequential TRSs (but sequentiality not decidable)
- strongly sequential TRSs (both decidable).

# Prefix property

## Theorem

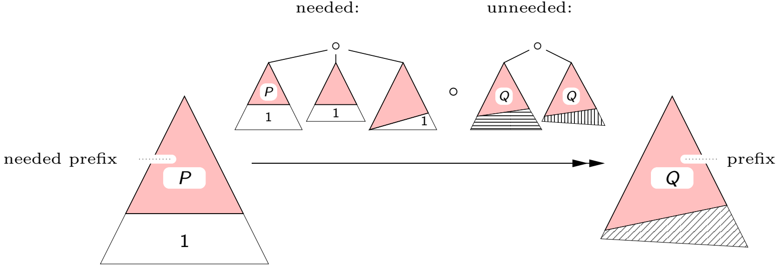
*Ancestor of prefix is prefix*



# Factorisation (semi-standardisation)

## Theorem

*Reductions factorise into needed/garbage for given prefix of target*



# Neededness summary

Formalized four neededness intuitions

- Permutation (inverting steps)
- Standardisation (sorting steps outside-in)
- Labelling (attach info, preserving behaviour)
- Tracing (see what happens)

Formalizations are equivalent