Programming with Time Warps

Adrien Guatto

IRIF & Université Paris 7

SYNCHRON 2019 Workshop

Programming with Time Warps

(More like: Trying to Implement Time Warp Polymorphism. . .)

Adrien Guatto

IRIF & Université Paris 7

SYNCHRON 2019 Workshop

Problems I've Been Interested In

Type-checking generalized guarded recursion	$\Leftrightarrow$	Deciding an extension of Lambek calculus (NC ILL)

Problems I've Been Interested In

Type-checking generalized guarded recursion $\Leftrightarrow$ Deciding an extension of Lambek calculus (NC ILL)

Problems I've Been Interested In

Type-checking generalized guarded recursion	$\Leftrightarrow$	Deciding an extension of Lambek calculus (NC ILL)

Problems I've Been Interested In

Type-checking generalized guarded recursion $\Leftrightarrow$ Deciding an extension of Lambek calculus (NC ILL)

Guarded Recursion in a Nutshell

- The usual typing rule for recursive definitions:

$$\textsc{Rec}\;\frac{\Gamma, x : A \vdash t : A}{\Gamma \vdash \mathbf{rec}\,(x : A).t : A}$$

In its presence, every type A classifies partial terms, e.g., $\mathbf{rec}\,(x : A).x$.

Guarded Recursion in a Nutshell

- The usual typing rule for recursive definitions:

$$\textsc{Rec}\quad \frac{\Gamma, x : A \vdash t : A}{\Gamma \vdash \mathbf{rec}\,(x : A).t : A}$$

In its presence, every type A classifies partial terms, e.g., $\mathbf{rec}\,(x : A).x$.

- A reasonably simple alternative is guarded recursion:

$$\textsc{GuardedRec}\quad \frac{\Gamma, x : \circledast_{0\overline{1}} A \vdash t : A}{\Gamma \vdash \mathbf{rec}\,(x : A).t : A}$$

Intuitively, this forces t's output to be strictly more defined than x.

Guarded Recursion in a Nutshell

- The usual typing rule for recursive definitions:

$$\frac{\Gamma, x : A \vdash t : A}{\Gamma \vdash \mathbf{rec}\,(x : A).t : A}\ \textsc{Rec}$$

  In its presence, every type A classifies partial terms, e.g., $\mathbf{rec}\,(x : A).x$.

- A reasonably simple alternative is guarded recursion:

$$\frac{\Gamma, x : \ast_{0\,\bar{1}} A \vdash t : A}{\Gamma \vdash \mathbf{rec}\,(x : A).t : A}\ \textsc{GuardedRec}$$

  Intuitively, this forces t's output to be strictly more defined than x.

- This leads to a language with recursive types and a family of modalities $\ast_{\_}$ graded by time warps, with attenant term formers.

Time Warps: Definition and Basic Structure

So-called "time warps" are generalized synchronous clocks.

Definition

A *time warp* p is a sup-preserving map from $\omega + 1$ to itself, i.e., it is a monotonic map such that:

$$p(0) = 0 \qquad \text{and} \qquad p(\omega) = \bigvee_{n<\omega} p(n).$$

Time Warps: Definition and Basic Structure

So-called "time warps" are generalized synchronous clocks.

Definition

A *time warp* p is a sup-preserving map from $\omega + 1$ to itself, i.e., it is a monotonic map such that:

$$p(0) = 0 \qquad \text{and} \qquad p(\omega) = \bigvee_{n<\omega} p(n).$$

Programming applications rely on the following structure:

- the lattice structure obtained by ordering time warps pointwise,
- the monoidal structure given by composition $p * q \triangleq q \circ p$,
- the residuals $\multimap$ and $\multimapinv$ are right adjoints to composition.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

Time Warps: Definition and Basic Structure

So-called "time warps" are generalized synchronous clocks.

Definition

A *time warp* p is a sup-preserving map from $\omega + 1$ to itself, i.e., it is a monotonic map such that:

$$p(0) = 0 \qquad \text{and} \qquad p(\omega) = \bigvee_{n<\omega} p(n).$$

Programming applications rely on the following structure:

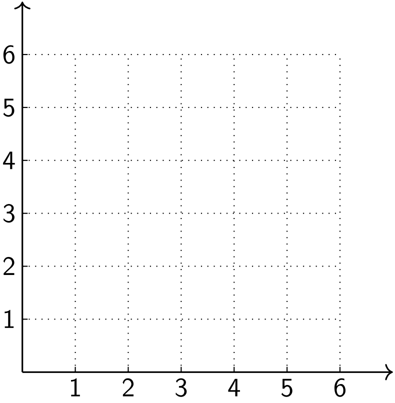
- the lattice structure obtained by ordering time warps pointwise,
- the monoidal structure given by composition $p * q \triangleq q \circ p$,
- the residuals $\multimap$ and $\multimapinv$ are right adjoints to composition.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

More on this structure, and in particular residuals, later.

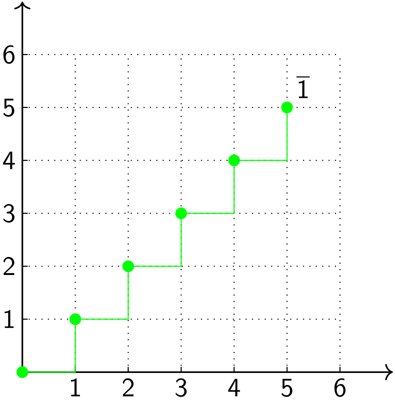
Representing Time Warps as Eventually Periodic Sequences

$\overline{1} = n \mapsto n$



Representing Time Warps as Eventually Periodic Sequences

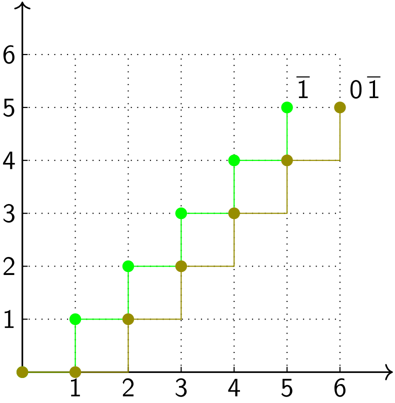
$\overline{1} = n \mapsto n$



Representing Time Warps as Eventually Periodic Sequences

$$\overline{1} = n \mapsto n$$

$$0\,\overline{1} = n \mapsto n - 1$$



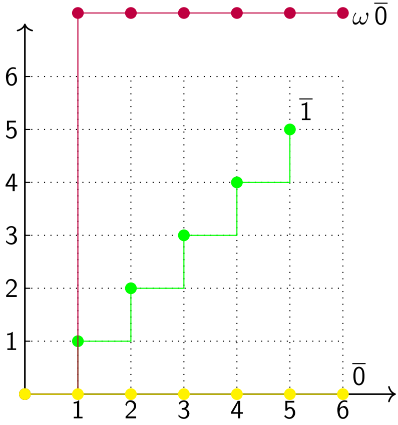
Representing Time Warps as Eventually Periodic Sequences

$$\overline{1} = n \mapsto n$$

$$0\,\overline{1} = n \mapsto n - 1$$

$$\overline{0} = n \mapsto 0$$

$$\omega\,\overline{0} = n \mapsto \omega \text{ for } n > 0$$



Representing Time Warps as Eventually Periodic Sequences

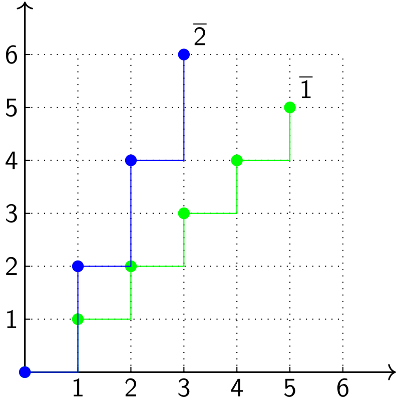
$$\overline{1} = n \mapsto n$$

$$0\,\overline{1} = n \mapsto n - 1$$

$$\overline{0} = n \mapsto 0$$

$$\omega\,\overline{0} = n \mapsto \omega \text{ for } n > 0$$

$$\overline{2} = n \mapsto 2n$$



Representing Time Warps as Eventually Periodic Sequences

$$\overline{1} = n \mapsto n$$

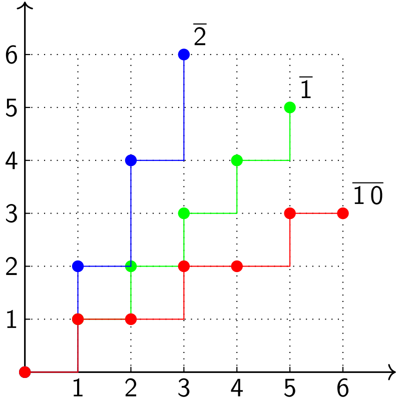
$$0\,\overline{1} = n \mapsto n - 1$$

$$\overline{0} = n \mapsto 0$$

$$\omega\,\overline{0} = n \mapsto \omega \text{ for } n > 0$$

$$\overline{2} = n \mapsto 2n$$

$$\overline{1\,0} = n \mapsto (n+1)/2$$

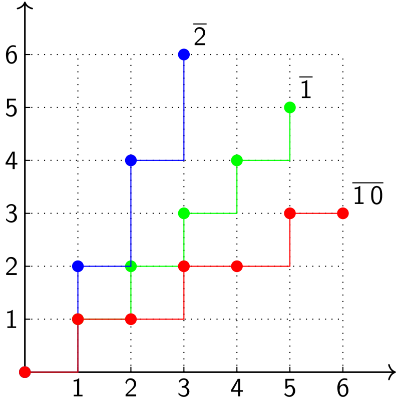


Representing Time Warps as Eventually Periodic Sequences

$$\begin{aligned}
\overline{1} &= n \mapsto n \\
0\,\overline{1} &= n \mapsto n - 1 \\
\overline{0} &= n \mapsto 0 \\
\omega\,\overline{0} &= n \mapsto \omega \text{ for } n > 0 \\
\overline{2} &= n \mapsto 2n \\
\overline{1\,0} &= n \mapsto (n+1)/2
\end{aligned}$$

They are closed under $*$, $\vee$, $\wedge$, $\multimap$, and $\multimapinv$.
For example, $\overline{2} * \overline{1\,0} = \overline{1}$.

$$\begin{array}{r|cccccc}
\overline{2} & 2 & 2 & 2 & 2 & 2 & \dots \\
\overline{1\,0} & 1\,0 & 1\,0 & 1\,0 & 1\,0 & 1\,0 & \dots \\
\overline{2} * \overline{1\,0} & 1 & 1 & 1 & 1 & 1 & \dots
\end{array}$$



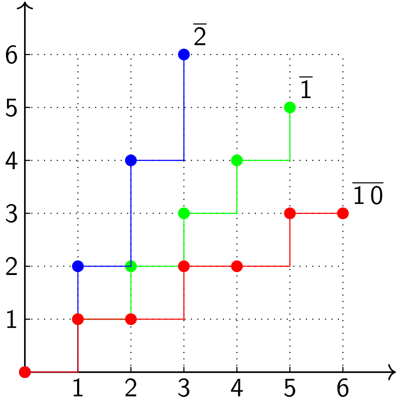
Representing Time Warps as Eventually Periodic Sequences

$$
\begin{aligned}
\overline{1} &= n \mapsto n \\
0\,\overline{1} &= n \mapsto n - 1 \\
\overline{0} &= n \mapsto 0 \\
\omega\,\overline{0} &= n \mapsto \omega \text{ for } n > 0 \\
\overline{2} &= n \mapsto 2n \\
\overline{1\,0} &= n \mapsto (n+1)/2
\end{aligned}
$$

They are closed under $*$, $\vee$, $\wedge$, $\multimap$, and $\multimapinv$.
For example, $\overline{2} * \overline{1\,0} = \overline{1}$.

$\overline{2}$	2	2	2	2	2	...
$\overline{1\,0}$	1 0	1 0	1 0	1 0	1 0	...
$\overline{2} * \overline{1\,0}$	1	1	1	1	1	...

Everything is computable/decidable.



Programming with Time Warps: Main Aspects

- Formal time warps P appear in gradings and denote time warps $[\![P]\!]$.

$$P, Q, R \coloneqq \underline{p} \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \mid P \wedge Q \mid P \vee Q$$
$$A \coloneqq \cdots \mid \ast_P A$$

Programming with Time Warps: Main Aspects

- Formal time warps P appear in gradings and denote time warps $[\![P]\!]$.

$$P, Q, R \coloneqq \underline{p} \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \mid P \wedge Q \mid P \vee Q$$
$$A \coloneqq \cdots \mid \ast_P A$$

- Delays provide unbounded buffering.

$$\frac{\Gamma \vdash t : \ast_Q A \qquad P \vdash Q}{\Gamma \vdash \mathbf{delay}_P\ t : \ast_P A}\ \textsc{Delay}$$

Programming with Time Warps: Main Aspects

- Formal time warps P appear in gradings and denote time warps $[\![P]\!]$.

$$P, Q, R \coloneqq \underline{p} \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \mid P \wedge Q \mid P \vee Q$$
$$A \coloneqq \cdots \mid \ast_P A$$

- Delays provide unbounded buffering.

$$\textsc{Delay}\quad \frac{\Gamma \vdash t : \ast_Q A \qquad P \vdash Q}{\Gamma \vdash \mathbf{delay}_P\ t : \ast_P A}$$

- Rescaling by P lets a subterm run "P-times faster" than its context.

$$\textsc{BadScale}\quad \frac{\Gamma \vdash t : A}{\ast_P \Gamma \vdash \mathbf{shut}_P\ t : \ast_P A}$$

(The above rule is not quite satisfactory but sufficient for this talk.)

Programming with Time Warps: Main Aspects

- Formal time warps P appear in gradings and denote time warps $[\![P]\!]$.

$$P, Q, R \coloneqq \underline{p} \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \mid P \wedge Q \mid P \vee Q$$
$$A \coloneqq \cdots \mid *_P A$$

- Delays provide unbounded buffering.

$$\frac{\Gamma \vdash t : *_Q A \qquad P \vdash Q}{\Gamma \vdash \mathbf{delay}_P\ t : *_P A}\ \textsc{Delay}$$

- Rescaling by P lets a subterm run "P-times faster" than its context.

$$\frac{\Gamma \vdash t : A}{*_P \Gamma \vdash \mathbf{shut}_P\ t : *_P A}\ \textsc{BadScale}$$

(The above rule is not quite satisfactory but sufficient for this talk.)

The entailment $P \vdash Q$ stands for $[\![P]\!] \leq [\![Q]\!]$, which is easy to decide.

Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

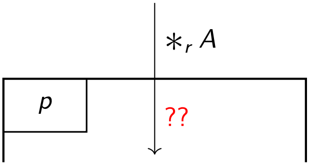
$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \mathrel{\circ\!\!-} q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \circ\!\!- q$$

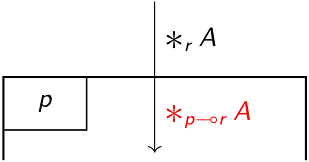
(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)



The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \circ\!\!- q$$

(The boxes below represent **shut**$_P$ $(-)$ term formers.)

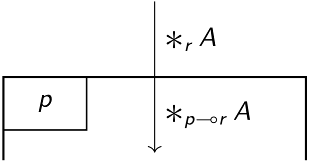


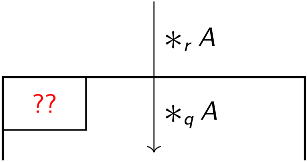
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \circ\!\!- q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)



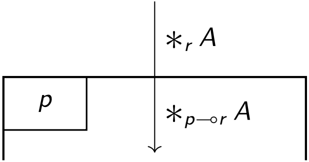


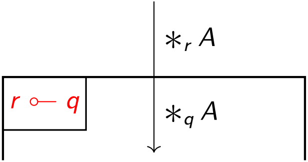
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\ (-)$ term formers.)



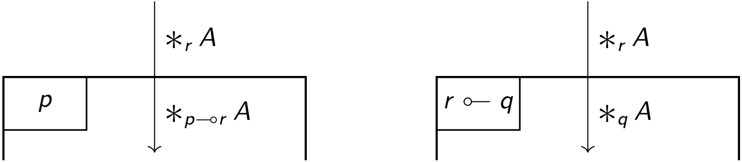


Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \circ\!\!- q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)



Exercise: let's try to find the values below.

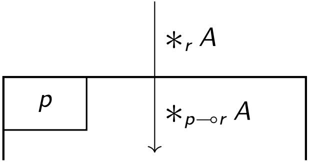
$\overline{1} \multimap \overline{2} =$	$0\,\overline{1} \multimap 0\,\overline{1} =$	$\overline{2} \multimap \overline{1} =$	$\overline{2} \multimap \overline{2} =$	$\omega\,\overline{0} \multimap \overline{1} =$
$\overline{2} \circ\!\!- \overline{1} =$	$0\,\overline{1} \circ\!\!- 0\,\overline{1} =$	$\overline{1} \circ\!\!- \overline{2} =$	$\overline{1\,0} \circ\!\!- \overline{1\,0} =$	$\overline{1} \circ\!\!- \overline{0} =$

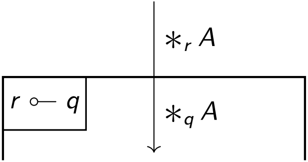
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





Exercise: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\quad 0\,\overline{1} \multimap 0\,\overline{1} =$ $\quad \overline{2} \multimap \overline{1} =$ $\quad \overline{2} \multimap \overline{2} =$ $\quad \omega\,\overline{0} \multimap \overline{1} =$

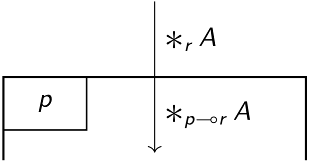
$\overline{2} \multimapinv \overline{1} =$ $\quad 0\,\overline{1} \multimapinv 0\,\overline{1} =$ $\quad \overline{1} \multimapinv \overline{2} =$ $\quad \overline{1\,0} \multimapinv \overline{1\,0} =$ $\quad \overline{1} \multimapinv \overline{0} =$

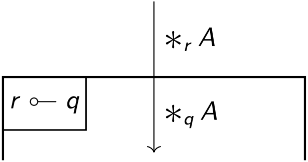
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \le p \multimap r \qquad \Leftrightarrow \qquad p * q \le r \qquad \Leftrightarrow \qquad p \le r \multimapinv q$$

(The boxes below represent $\textbf{shut}_P\,(-)$ term formers.)





Exercise: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\qquad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\qquad \overline{2} \multimap \overline{1} =$ $\qquad \overline{2} \multimap \overline{2} =$ $\qquad \omega\,\overline{0} \multimap \overline{1} =$

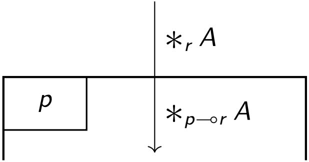
$\overline{2} \multimapinv \overline{1} =$ $\qquad 0\,\overline{1} \multimapinv 0\,\overline{1} =$ $\qquad \overline{1} \multimapinv \overline{2} =$ $\qquad \overline{1\,0} \multimapinv \overline{1\,0} =$ $\qquad \overline{1} \multimapinv \overline{0} =$

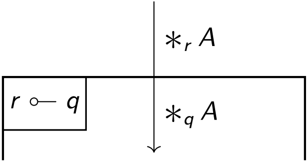
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





Exercise: let's try to find the values below.

$$\overline{1} \multimap \overline{2} = \overline{2} \qquad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1} \qquad \overline{2} \multimap \overline{1} = \overline{1\,0} \qquad \overline{2} \multimap \overline{2} = \qquad \omega\,\overline{0} \multimap \overline{1} =$$

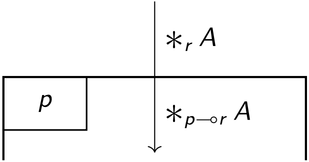
$$\overline{2} \multimapinv \overline{1} = \qquad 0\,\overline{1} \multimapinv 0\,\overline{1} = \qquad \overline{1} \multimapinv \overline{2} = \qquad \overline{1\,0} \multimapinv \overline{1\,0} = \qquad \overline{1} \multimapinv \overline{0} =$$

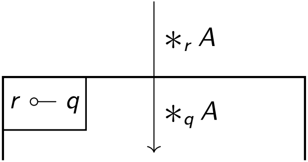
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \circ\!\!- q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





<u>Exercise</u>: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2} \qquad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1} \qquad \overline{2} \multimap \overline{1} = \overline{1\,0} \qquad \overline{2} \multimap \overline{2} = \overline{2\,0} \qquad \omega\,\overline{0} \multimap \overline{1} =$

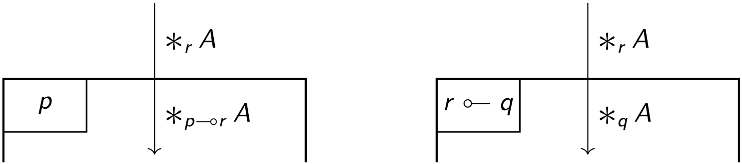
$\overline{2} \circ\!\!- \overline{1} = \qquad 0\,\overline{1} \circ\!\!- 0\,\overline{1} = \qquad \overline{1} \circ\!\!- \overline{2} = \qquad \overline{1\,0} \circ\!\!- \overline{1\,0} = \qquad \overline{1} \circ\!\!- \overline{0} =$

Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)



Exercise: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\quad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\quad \overline{2} \multimap \overline{1} = \overline{1\,0}$ $\quad \overline{2} \multimap \overline{2} = \overline{2\,0}$ $\quad \omega\,\overline{0} \multimap \overline{1} = 1\,\overline{0}$

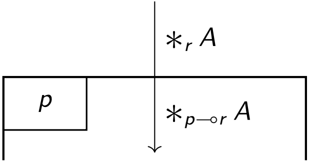
$\overline{2} \multimapinv \overline{1} =$ $\quad 0\,\overline{1} \multimapinv 0\,\overline{1} =$ $\quad \overline{1} \multimapinv \overline{2} =$ $\quad \overline{1\,0} \multimapinv \overline{1\,0} =$ $\quad \overline{1} \multimapinv \overline{0} =$

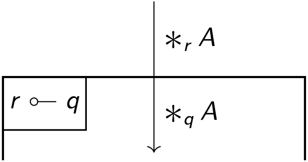
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \quad \Leftrightarrow \quad p * q \leq r \quad \Leftrightarrow \quad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





Exercise: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\quad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\quad \overline{2} \multimap \overline{1} = \overline{1\,0}$ $\quad \overline{2} \multimap \overline{2} = \overline{2\,0}$ $\quad \omega\,\overline{0} \multimap \overline{1} = 1\,\overline{0}$

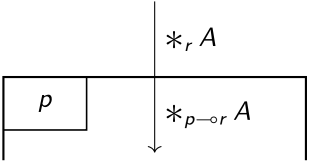
$\overline{2} \multimapinv \overline{1} = \overline{2}$ $\quad 0\,\overline{1} \multimapinv 0\,\overline{1} =$ $\quad \overline{1} \multimapinv \overline{2} =$ $\quad \overline{1\,0} \multimapinv \overline{1\,0} =$ $\quad \overline{1} \multimapinv \overline{0} =$

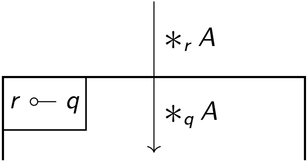
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





<u>Exercise</u>: let's try to find the values below.

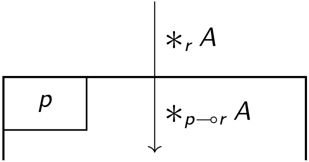
$\overline{1} \multimap \overline{2} = \overline{2}$ $\qquad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\qquad \overline{2} \multimap \overline{1} = \overline{1\,0}$ $\qquad \overline{2} \multimap \overline{2} = \overline{2\,0}$ $\qquad \omega\,\overline{0} \multimap \overline{1} = 1\,\overline{0}$

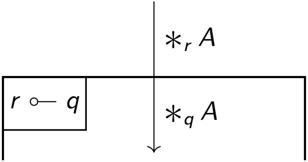
$\overline{2} \multimapinv \overline{1} = \overline{2}$ $\qquad 0\,\overline{1} \multimapinv 0\,\overline{1} = \overline{1}$ $\qquad \overline{1} \multimapinv \overline{2} =$ $\qquad \overline{1\,0} \multimapinv \overline{1\,0} =$ $\qquad \overline{1} \multimapinv \overline{0} =$

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \mathrel{\circ\!\!-} q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





<u>Exercise</u>: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\quad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\quad \overline{2} \multimap \overline{1} = \overline{1\,0}$ $\quad \overline{2} \multimap \overline{2} = \overline{2\,0}$ $\quad \omega\,\overline{0} \multimap \overline{1} = 1\,\overline{0}$

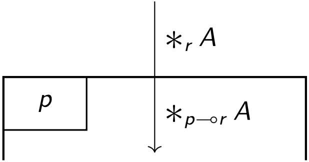
$\overline{2} \mathrel{\circ\!\!-} \overline{1} = \overline{2}$ $\quad 0\,\overline{1} \mathrel{\circ\!\!-} 0\,\overline{1} = \overline{1}$ $\quad \overline{1} \mathrel{\circ\!\!-} \overline{2} = \overline{0\,1}$ $\quad \overline{1\,0} \mathrel{\circ\!\!-} \overline{1\,0} =$ $\quad \overline{1} \mathrel{\circ\!\!-} \overline{0} =$

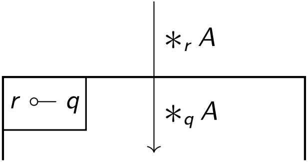
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\textbf{shut}_P\,(-)$ term formers.)





<u>Exercise</u>: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\quad$ $0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\quad$ $\overline{2} \multimap \overline{1} = \overline{1\,0}$ $\quad$ $\overline{2} \multimap \overline{2} = \overline{2\,0}$ $\quad$ $\omega\,\overline{0} \multimap \overline{1} = 1\,\overline{0}$

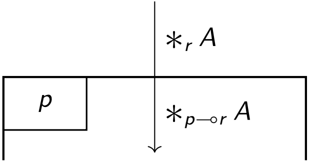
$\overline{2} \multimapinv \overline{1} = \overline{2}$ $\quad$ $0\,\overline{1} \multimapinv 0\,\overline{1} = \overline{1}$ $\quad$ $\overline{1} \multimapinv \overline{2} = \overline{0\,1}$ $\quad$ $\overline{1\,0} \multimapinv \overline{1\,0} = \overline{2\,0}$ $\quad$ $\overline{1} \multimapinv \overline{0} =$

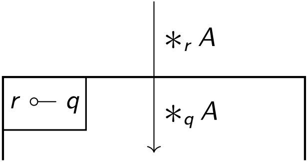
Interlude: Residuals from a Programming Perspective

The universal property of residuals makes them useful when programming.

$$q \leq p \multimap r \qquad \Leftrightarrow \qquad p * q \leq r \qquad \Leftrightarrow \qquad p \leq r \multimapinv q$$

(The boxes below represent $\mathbf{shut}_P\,(-)$ term formers.)





<u>Exercise</u>: let's try to find the values below.

$\overline{1} \multimap \overline{2} = \overline{2}$ $\quad 0\,\overline{1} \multimap 0\,\overline{1} = \overline{1}$ $\quad \overline{2} \multimap \overline{1} = \overline{1\,0}$ $\quad \overline{2} \multimap \overline{2} = \overline{2\,0}$ $\quad \omega\,\overline{0} \multimap \overline{1} = 1\,\overline{0}$

$\overline{2} \multimapinv \overline{1} = \overline{2}$ $\quad 0\,\overline{1} \multimapinv 0\,\overline{1} = \overline{1}$ $\quad \overline{1} \multimapinv \overline{2} = \overline{0\,1}$ $\quad \overline{1\,0} \multimapinv \overline{1\,0} = \overline{2\,0}$ $\quad \overline{1} \multimapinv \overline{0} = \omega\,\overline{0}$

Interlude: What About Left Adjoints To Composition?

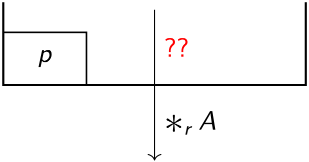
What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

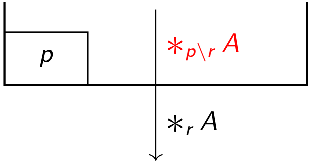
$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$



Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

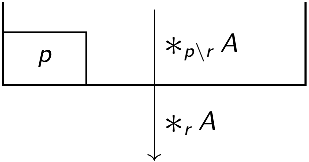
$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r / q \leq p$$

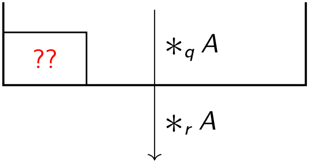


Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$

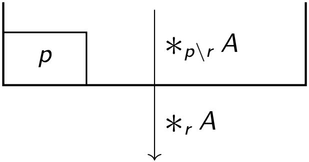


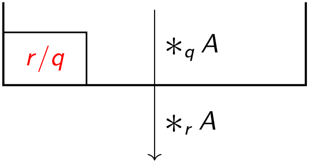


Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$

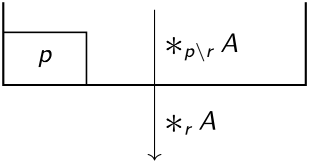


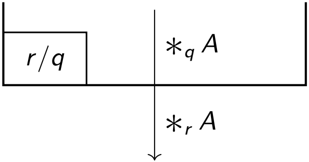


Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





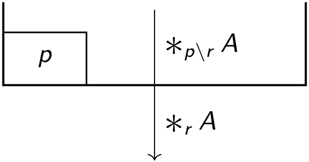
<u>Exercise</u>: let's try to find the values below.

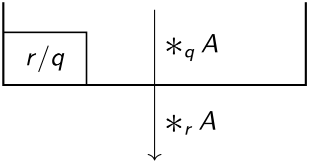
$\overline{1}/\overline{2} =$	$0\,\overline{1}/0\,\overline{1} =$	$\overline{2}/\overline{1} =$	$\overline{1\,0}/\overline{1\,0} =$	$\overline{1}/\overline{0} =$
$\overline{1} \setminus \overline{2} =$	$\overline{2} \setminus \overline{1} =$	$\overline{0\,1} \setminus \overline{0\,1} =$	$0\,\overline{1} \setminus \overline{1} =$	$\omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





<u>Exercise</u>: let's try to find the values below.

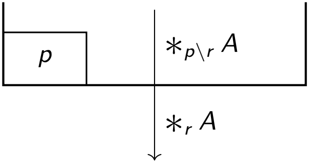
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} =$ $\quad \overline{2}/\overline{1} =$ $\quad \overline{1\,0}/\overline{1\,0} =$ $\quad \overline{1}/\overline{0} =$

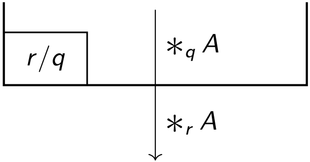
$\overline{1} \setminus \overline{2} =$ $\quad \overline{2} \setminus \overline{1} =$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





Exercise: let's try to find the values below.

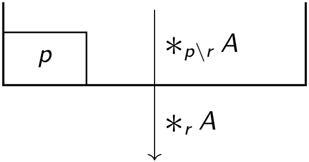
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} =$ $\quad \overline{1\,0}/\overline{1\,0} =$ $\quad \overline{1}/\overline{0} =$

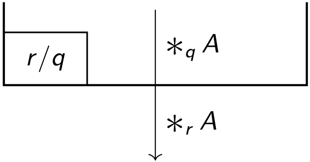
$\overline{1} \setminus \overline{2} =$ $\quad \overline{2} \setminus \overline{1} =$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





<u>Exercise</u>: let's try to find the values below.

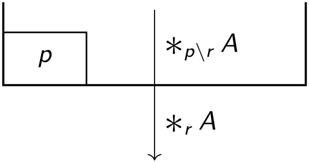
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} =$ $\quad \overline{1}/\overline{0} =$

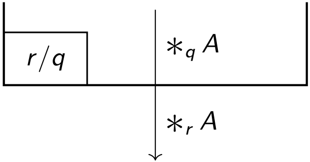
$\overline{1} \setminus \overline{2} =$ $\quad \overline{2} \setminus \overline{1} =$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





<u>Exercise</u>: let's try to find the values below.

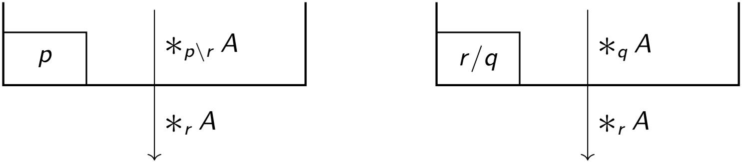
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad \overline{1}/\overline{0} =$

$\overline{1} \setminus \overline{2} =$ $\quad \overline{2} \setminus \overline{1} =$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$



<u>Exercise</u>: let's try to find the values below.

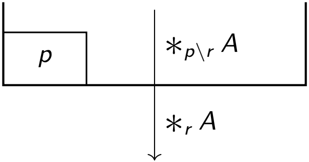
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad \overline{1}/\overline{0} =$ *undef*

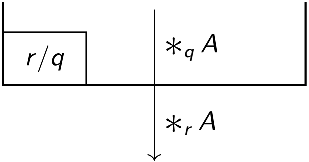
$\overline{1} \setminus \overline{2} =$ $\quad \overline{2} \setminus \overline{1} =$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





<ins>Exercise</ins>: let's try to find the values below.

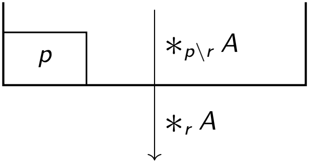
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad \overline{1}/\overline{0} = \mathit{undef}$

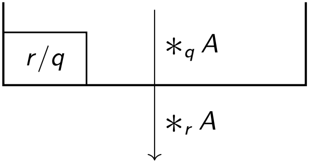
$\overline{1} \setminus \overline{2} = \overline{2}$ $\quad \overline{2} \setminus \overline{1} =$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \iff r \leq p * q \iff r/q \leq p$$





<u>Exercise</u>: let's try to find the values below.

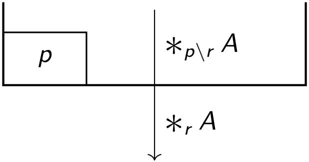
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad \overline{1}/\overline{0} = \mathit{undef}$

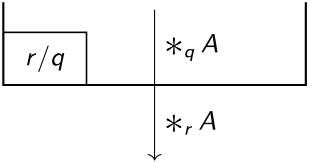
$\overline{1} \setminus \overline{2} = \overline{2}$ $\quad \overline{2} \setminus \overline{1} = \overline{0\,1}$ $\quad \overline{0\,1} \setminus \overline{0\,1} =$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





Exercise: let's try to find the values below.

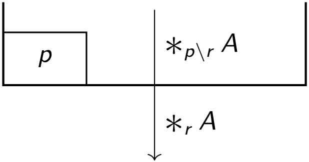
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad \overline{1}/\overline{0} = \mathit{undef}$

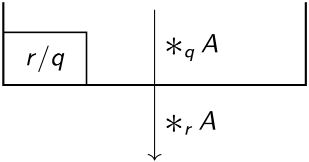
$\overline{1} \setminus \overline{2} = \overline{2}$ $\quad \overline{2} \setminus \overline{1} = \overline{0\,1}$ $\quad \overline{0\,1} \setminus \overline{0\,1} = \overline{0\,2}$ $\quad 0\,\overline{1} \setminus \overline{1} =$ $\quad \omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





<u>Exercise</u>: let's try to find the values below.

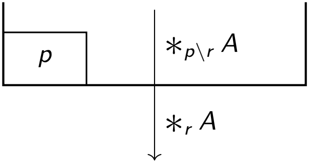
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad$ $0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad$ $\overline{2}/\overline{1} = \overline{2}$ $\quad$ $\overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad$ $\overline{1}/\overline{0} = \mathit{undef}$

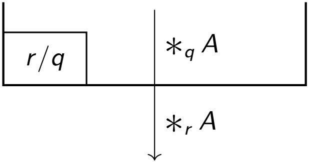
$\overline{1} \setminus \overline{2} = \overline{2}$ $\quad$ $\overline{2} \setminus \overline{1} = \overline{0\,1}$ $\quad$ $\overline{0\,1} \setminus \overline{0\,1} = \overline{0\,2}$ $\quad$ $0\,\overline{1} \setminus \overline{1} = \mathit{undef}$ $\quad$ $\omega\,\overline{0} \setminus \overline{1} =$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





Exercise: let's try to find the values below.

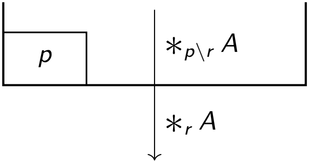
$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad 0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad \overline{2}/\overline{1} = \overline{2}$ $\quad \overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad \overline{1}/\overline{0} = \mathit{undef}$

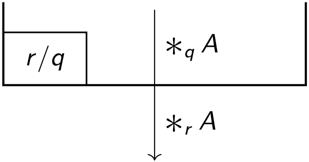
$\overline{1} \setminus \overline{2} = \overline{2}$ $\quad \overline{2} \setminus \overline{1} = \overline{0\,1}$ $\quad \overline{0\,1} \setminus \overline{0\,1} = \overline{0\,2}$ $\quad 0\,\overline{1} \setminus \overline{1} = \mathit{undef}$ $\quad \omega\,\overline{0} \setminus \overline{1} = \mathit{undef}$

Interlude: What About Left Adjoints To Composition?

What about *left* adjoints to pre/post-composition?

$$p \setminus r \leq q \;\Leftrightarrow\; r \leq p * q \;\Leftrightarrow\; r/q \leq p$$





<u>Exercise</u>: let's try to find the values below.

$\overline{1}/\overline{2} = \overline{1\,0}$ $\quad$ $0\,\overline{1}/0\,\overline{1} = 0\,2\,\overline{1}$ $\quad$ $\overline{2}/\overline{1} = \overline{2}$ $\quad$ $\overline{1\,0}/\overline{1\,0} = 1\,\overline{0\,2}$ $\quad$ $\overline{1}/\overline{0} = \mathit{undef}$

$\overline{1} \setminus \overline{2} = \overline{2}$ $\quad$ $\overline{2} \setminus \overline{1} = \overline{0\,1}$ $\quad$ $\overline{0\,1} \setminus \overline{0\,1} = \overline{0\,2}$ $\quad$ $0\,\overline{1} \setminus \overline{1} = \mathit{undef}$ $\quad$ $\omega\,\overline{0} \setminus \overline{1} = \mathit{undef}$

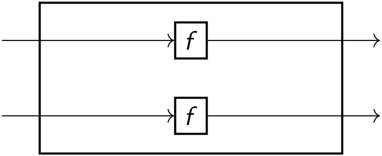
They are *partial*: r/q is defined when $r(\omega) \leq q(\omega)$, $p \setminus r$ is more complicated.

The Need for Time Warp Polymorphism

What type do we want for the function g? (Adapted from Gonthier.)

$f : \mathbf{S}\,\mathbf{B} \to \mathbf{S}\,\mathbf{B}$

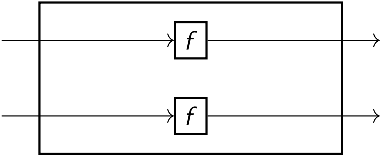
$\mathrm{g} \triangleq \lambda \mathrm{x}.(f\ (\mathbf{fst}\ \mathtt{x}), f\ (\mathbf{snd}\ \mathtt{x}))$



The Need for Time Warp Polymorphism

What type do we want for the function g? (Adapted from Gonthier.)

$$f : \mathbf{S}\,\mathbf{B} \to \mathbf{S}\,\mathbf{B}$$
$$\mathrm{g} \triangleq \lambda \mathtt{x}.(f\ (\mathbf{fst}\ \mathtt{x}), f\ (\mathbf{snd}\ \mathtt{x}))$$

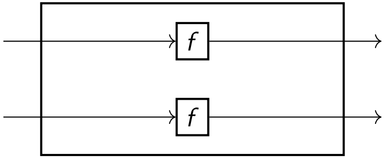


$$\mathrm{g} : \forall (X\ Y : \mathbf{warp}). *_X (\mathbf{S}\,\mathbf{B}) \times *_Y (\mathbf{S}\,\mathbf{B}) \to *_X (\mathbf{S}\,\mathbf{B}) \times *_Y (\mathbf{S}\,\mathbf{B})$$

The Need for Time Warp Polymorphism

What type do we want for the function g? (Adapted from Gonthier.)

$$f : \mathbf{S}\,\mathbf{B} \to \mathbf{S}\,\mathbf{B}$$
$$\mathrm{g} \triangleq \lambda \mathrm{x}.(f\ (\mathbf{fst}\ \mathrm{x}), f\ (\mathbf{snd}\ \mathrm{x}))$$



$$\mathrm{g} : \forall (X\ Y : \mathbf{warp}). *_X (\mathbf{S}\,\mathbf{B}) \times *_Y (\mathbf{S}\,\mathbf{B}) \to *_X (\mathbf{S}\,\mathbf{B}) \times *_Y (\mathbf{S}\,\mathbf{B})$$

This requires universal quantification over time warps.

$$A ::= \cdots \mid *_P A \mid \forall (X : \mathbf{warp}).A$$
$$P, Q, R ::= X \mid \underline{p} \mid P * Q \mid P \multimap Q \mid P \circ\!\!- Q \mid P \wedge Q \mid P \vee Q$$

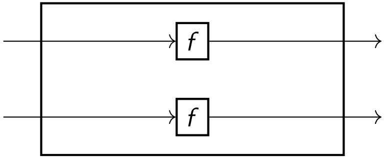
How should $P \vdash Q$ be extended to deal with formal variables?

The Need for Time Warp Polymorphism

What type do we want for the function g? (Adapted from Gonthier.)

$f : \mathbf{S}\,\mathbf{B} \to \mathbf{S}\,\mathbf{B}$

$\mathrm{g} \triangleq \lambda \mathrm{x}.(f\ (\mathbf{fst}\ \mathrm{x}), f\ (\mathbf{snd}\ \mathrm{x}))$



$$\mathrm{g} : \forall (X\ Y : \mathbf{warp}).*_X (\mathbf{S}\,\mathbf{B}) \times *_Y (\mathbf{S}\,\mathbf{B}) \to *_X (\mathbf{S}\,\mathbf{B}) \times *_Y (\mathbf{S}\,\mathbf{B})$$

This requires universal quantification over time warps.

$$A ::= \cdots \mid *_P A \mid \forall (X : \mathbf{warp}).A$$
$$P, Q, R ::= X \mid \underline{p} \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \mid P \wedge Q \mid P \vee Q$$

How should $P \vdash Q$ be extended to deal with formal variables?

Time Warp Entailment: Desiderata

Formal time warps now denote maps from $\varphi \in \mathit{Var} \to \mathcal{W}$ to $\mathcal{W}$.

$$[\![X]\!]_\varphi = \varphi(X) \qquad [\![p]\!]_\varphi = p \qquad [\![P * Q]\!]_\varphi = [\![P]\!]_\varphi * [\![Q]\!]_\varphi \qquad \ldots$$

Time Warp Entailment: Desiderata

Formal time warps now denote maps from $\varphi \in Var \to \mathcal{W}$ to $\mathcal{W}$.

$$[\![X]\!]_\varphi = \varphi(X) \qquad [\![p]\!]_\varphi = p \qquad [\![P * Q]\!]_\varphi = [\![P]\!]_\varphi * [\![Q]\!]_\varphi \qquad \ldots$$

What should the entailment predicate $P \vdash Q$ look like?

- It should (at least) be sound.

$$P \vdash Q \qquad \Rightarrow \qquad \forall \varphi : Var \to \mathcal{W}, [\![P]\!]_\varphi \leq [\![Q]\!]_\varphi$$

- It does not necessarily have to be complete.

$$\forall \varphi : Var \to \mathcal{W}, [\![P]\!]_\varphi \leq [\![Q]\!]_\varphi \qquad \overset{?}{\Rightarrow} \qquad P \vdash Q$$

- It should be decidable, perhaps defined as an inductive judgment?

Time Warp Entailment: Desiderata

Formal time warps now denote maps from $\varphi \in \mathit{Var} \to \mathcal{W}$ to $\mathcal{W}$.

$$[\![X]\!]_\varphi = \varphi(X) \qquad [\![p]\!]_\varphi = p \qquad [\![P * Q]\!]_\varphi = [\![P]\!]_\varphi * [\![Q]\!]_\varphi \qquad \ldots$$

What should the entailment predicate $P \vdash Q$ look like?

- It should (at least) be sound.

$$P \vdash Q \qquad \Rightarrow \qquad \forall \varphi : \mathit{Var} \to \mathcal{W}, [\![P]\!]_\varphi \leq [\![Q]\!]_\varphi$$

- It does not necessarily have to be complete.

$$\forall \varphi : \mathit{Var} \to \mathcal{W}, [\![P]\!]_\varphi \leq [\![Q]\!]_\varphi \qquad \overset{?}{\Rightarrow} \qquad P \vdash Q$$

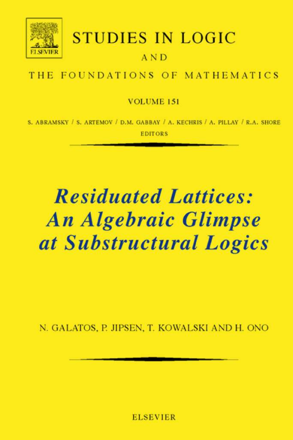
- It should be decidable, perhaps defined as an inductive judgment?

I struggled for a while with ad-hoc proof rules, until…

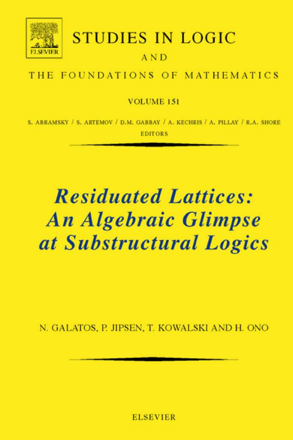
Heard on a Bus in Copenhagen, May 2019

Niccolò Veltri: “This seems related to Lambek calculus!”

Residuated Lattices and Lambek Calculi



Residuated Lattices and Lambek Calculi



After some reading, I learned how time warps:

- form a *bounded residuated lattice*,
- and thus model *full Lambek calculus* (FL).

FL is decidable, and its formulas belong to the following subset of formal time warps:

$$\begin{aligned} P, Q, R ::= {} & X \mid \overline{0} \mid \overline{1} \mid \omega\,\overline{0} \\ & \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \\ & \mid P \wedge Q \mid P \vee Q \end{aligned}$$

All the connectives are there, but most time warps are obviously missing. Can we add them while preserving decidability?

Full Lambek Calculus with Time Warps: Syntax

Formulas are formal time warps, contexts are lists of formulas.

$$
\begin{aligned}
P, Q, R &::= X \mid \underline{p} \mid P * Q \mid P \multimap Q \mid P \multimapinv Q \mid P \wedge Q \mid P \vee Q \\
\Theta &::= \cdot \mid \Theta, P
\end{aligned}
$$

The decidable congruence $\equiv$ captures equivalence of closed formulas.

$$
\frac{}{X \equiv X} \qquad \frac{}{\underline{p} * \underline{q} \equiv \underline{p * q}} \qquad \frac{}{\underline{p} \multimap \underline{q} \equiv \underline{p \multimap q}} \qquad \cdots
$$

We extend it to contexts.

$$
\frac{}{\Theta \equiv \Theta} \qquad \frac{\Theta_1 \equiv \Theta_2 \qquad \Theta_2 \equiv \Theta_3}{\Theta_1 \equiv \Theta_3} \qquad \frac{P \equiv P'}{\Theta_1, P, \Theta_2 \equiv \Theta_1, P', \Theta_2}
$$

$$
\frac{}{\Theta_1, \underline{p}, \underline{q}, \Theta_2 \equiv \Theta_1, \underline{p * q}, \Theta_2} \qquad \frac{}{\Theta_1, \underline{\overline{1}}, \Theta_2 \equiv \Theta_1, \Theta_2}
$$

Full Lambek Calculus: Existing Rules (1/2)

FL corresponds to non-commutative ILL with units.

$$\textsc{Axiom}\ \frac{}{P \vdash P} \qquad \textsc{Cut}\ \frac{\Theta_2 \vdash Q \qquad \Theta_1, Q, \Theta_3 \vdash P}{\Theta_1, \Theta_2, \Theta_3 \vdash P}$$

$$*\mathrm{L}\ \frac{\Theta_1, P, Q, \Theta_2 \vdash R}{\Theta_1, P * Q, \Theta_2 \vdash R} \qquad *\mathrm{R}\ \frac{\Theta_1 \vdash P \qquad \Theta_2 \vdash Q}{\Theta_1, \Theta_2 \vdash P * Q} \qquad \multimap\mathrm{L}\ \frac{\Theta_2 \vdash P \qquad \Theta_1, Q, \Theta_3 \vdash R}{\Theta_1, \Theta_2, P \multimap Q, \Theta_3 \vdash R}$$

$$\multimap\mathrm{R}\ \frac{P, \Theta \vdash Q}{\Theta \vdash P \multimap Q} \qquad \multimapinv\mathrm{L}\ \frac{\Theta_2 \vdash P \qquad \Theta_1, Q, \Theta_3 \vdash R}{\Theta_1, Q \multimapinv P, \Theta_2, \Theta_3 \vdash R} \qquad \multimapinv\mathrm{R}\ \frac{\Theta, P \vdash Q}{\Theta \vdash Q \multimapinv P}$$

Full Lambek Calculus: Existing Rules (2/2)

$$\wedge\mathrm{L1}\;\frac{\Theta_1, P, \Theta_2 \vdash R}{\Theta_1, P \wedge Q, \Theta_2 \vdash R} \qquad \wedge\mathrm{L2}\;\frac{\Theta_1, Q, \Theta_2 \vdash R}{\Theta_1, P \wedge Q, \Theta_2 \vdash R} \qquad \wedge\mathrm{R}\;\frac{\Theta \vdash P \qquad \Theta \vdash Q}{\Theta \vdash P \wedge Q}$$

$$\vee\mathrm{L}\;\frac{\Theta_1, P, \Theta_2 \vdash R \qquad \Theta_1, Q, \Theta_2 \vdash R}{\Theta_1, P \vee Q, \Theta_2 \vdash R} \qquad \vee\mathrm{R1}\;\frac{\Theta \vdash P}{\Theta \vdash P \vee Q} \qquad \vee\mathrm{R2}\;\frac{\Theta \vdash P}{\Theta \vdash P \vee Q}$$

$$\overline{0}\mathrm{L}\;\frac{}{\Theta_1, \overline{0}, \Theta_2 \vdash P} \qquad \omega\,\overline{0}\mathrm{R}\;\frac{}{\Theta \vdash \omega\,\overline{0}} \qquad \overline{1}\mathrm{L}\;\frac{\Theta_1, \Theta_2 \vdash P}{\Theta_1, \overline{1}, \Theta_2 \vdash P} \qquad \overline{1}\mathrm{R}\;\frac{}{\cdot \vdash \overline{1}}$$

Full Lambek Calculus: Adding General Time Warps

We want to enrich the preceding rules in order to:

- include the ordering relation between time warps,
- reason up to the equations coming from the model.

This leads to the following rules:

$$\textsc{WeakWarp}\ \frac{p \leq q \qquad \Theta_1, \underline{q}, \Theta_2 \vdash P}{\Theta_1, \underline{p}, \Theta_2 \vdash P} \qquad \textsc{EqL}\ \frac{\Theta \equiv \Theta' \qquad \Theta' \vdash P}{\Theta \vdash P} \qquad \textsc{EqR}\ \frac{P \equiv P' \qquad \Theta \vdash P'}{\Theta \vdash P}$$

The decidability of FL follows from cut elimination and subformula property. However, in the extended calculus:

- cut elimination is not obvious, and
- even if it held, the calculus does not enjoy the subformula property.

Failure of the Subformula Property

All the following rules are derivable even in the absence of cut:

$$\frac{\Theta_1, \overline{\underline{1}}, \Theta_2 \vdash P}{\Theta_1, \Theta_2 \vdash P} \qquad \frac{\Theta_1, \overline{\underline{2}}, \overline{\underline{1\,0}}, \Theta_2 \vdash P}{\Theta_1, \Theta_2 \vdash P} \qquad \frac{\Theta_1, \overline{\underline{2}}, \overline{\underline{0\,1}}, \Theta_2 \vdash P}{\Theta_1, \Theta_2 \vdash P}$$

$$\frac{\Theta_1, \overline{\underline{3}}, \overline{\underline{1\,0\,0}}, \Theta_2 \vdash P}{\Theta_1, \Theta_2 \vdash P} \qquad \frac{\Theta_1, \overline{\underline{3}}, \overline{\underline{0\,1\,0}}, \Theta_2 \vdash P}{\Theta_1, \Theta_2 \vdash P} \qquad \frac{\Theta_1, \overline{\underline{3}}, \overline{\underline{0\,0\,1}}, \Theta_2 \vdash P}{\Theta_1, \Theta_2 \vdash P}$$

$$\ldots$$

An infinite number of instances of the $\textsc{EqL}$ rule can always be applied.

An Idea of the Difficulties

Why is the following sequent derivable?

$$X \multimapinv \overline{\underline{1\,0}}, \overline{\underline{2}} \multimap Y \vdash X * Y$$

An Idea of the Difficulties

Why is the following sequent derivable?

$$X \multimapinv \underline{\overline{1\,0}}, \underline{\overline{2}} \multimap Y \vdash X * Y$$

The proof below is, I think, the only one up to nonessential differences.

$$\dfrac{\overline{1} \mathrel{\underline{\leq}} \overline{1\,0} * \overline{2} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\underline{\overline{1\,0}} \vdash \underline{\overline{1\,0}} \qquad X \vdash X}{X \multimapinv \underline{\overline{1\,0}}, \underline{\overline{1\,0}} \vdash X} \qquad \dfrac{\underline{\overline{2}} \vdash \underline{\overline{2}} \qquad Y \vdash Y}{\underline{\overline{2}}, \underline{\overline{2}} \multimap Y \vdash Y}}{X \multimapinv \overline{1\,0}, \underline{\overline{1\,0}}, \underline{\overline{2}}, \overline{2} \multimap Y \vdash X * Y}}{X \multimapinv \overline{1\,0}, \underline{\overline{1\,0}} * \underline{\overline{2}}, \overline{2} \multimap Y \vdash X * Y}}{X \multimapinv \overline{1\,0}, \underline{\overline{1\,0} * \overline{2}}, \overline{2} \multimap Y \vdash X * Y}}{\dfrac{X \multimapinv \overline{1\,0}, \underline{\overline{1}}, \overline{2} \multimap Y \vdash X * Y}{X \multimapinv \overline{1\,0}, \overline{2} \multimap Y \vdash X * Y}}$$

An Idea of the Difficulties

Why is the following sequent derivable?

$$X \multimapinv \underline{\overline{1\,0}}, \underline{\overline{2}} \multimap Y \vdash X * Y$$

The proof below is, I think, the only one up to nonessential differences.

$$\dfrac{\overline{1} \leq \overline{1\,0} * \overline{2} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\underline{\overline{1\,0}} \vdash \underline{\overline{1\,0}} \qquad X \vdash X}{X \multimapinv \underline{\overline{1\,0}}, \underline{\overline{1\,0}} \vdash X} \qquad \dfrac{\underline{\overline{2}} \vdash \underline{\overline{2}} \qquad Y \vdash Y}{\underline{\overline{2}}, \underline{\overline{2}} \multimap Y \vdash Y}}{X \multimapinv \overline{1\,0}, \underline{\overline{1\,0}}, \underline{\overline{2}}, \overline{2} \multimap Y \vdash X * Y}}{X \multimapinv \overline{1\,0}, \underline{\overline{1\,0}} * \underline{\overline{2}}, \overline{2} \multimap Y \vdash X * Y}}{X \multimapinv \overline{1\,0}, \underline{\overline{1\,0} * \overline{2}}, \overline{2} \multimap Y \vdash X * Y}}{\dfrac{X \multimapinv \overline{1\,0}, \underline{\overline{1}}, \overline{2} \multimap Y \vdash X * Y}{X \multimapinv \overline{1\,0}, \overline{2} \multimap Y \vdash X * Y}}$$

This seems discouraging: how do we know that $\overline{1}$ has to be split into $\overline{1\,0} * \overline{2}$?

Perspectives

Dear off-line readers…

… at this point the talk was cut short.

In this talk, I tried to make the following points:

- time warps are a generalization of synchronous clocks that allow an arbitrary countable number of activations (or data) per time step,
- they form a model of Lambek calculus, a non-commutative and intuitionnistic variant of linear logic originating from linguistics,
- Lambek calculus can be extended to talk directly about concrete time warps by adding them as atomic formulas related by axioms,
- the decidability of this extension is an interesting and unsettled problem.