Rewriting dataflow diagrams

David Sprunger (NII Tokyo)

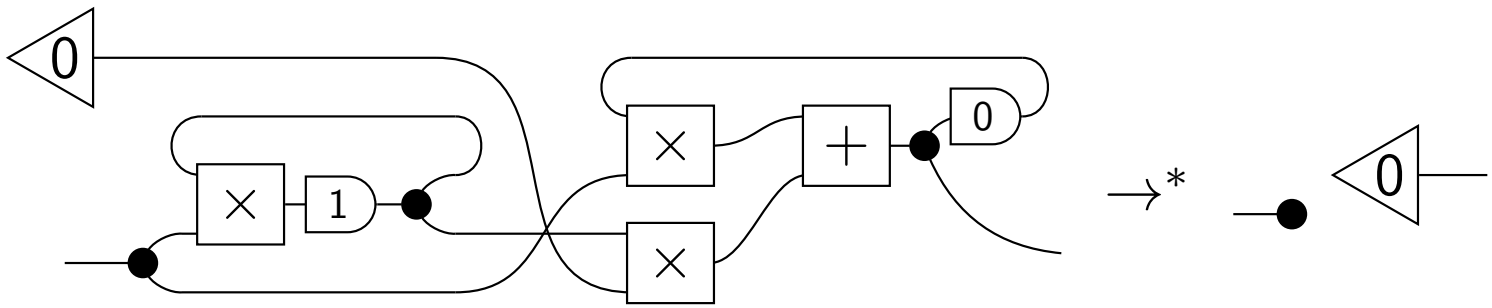
TRS 50^{th} meeting
February 29, 2019

Disclaimer

I don't know ~~anything~~ much about term rewriting.

Where we're heading

I would like to use rewriting tools to simplify diagrams representing certain kinds of functions on sequences:



Causal functions on sequences

A^ω is the set of A-*valued infinite sequences*. The entries of $\sigma \in A^\omega$ are $\sigma_k \in A$ for $k \in \mathbb{N}$, so $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_k, \ldots)$.

Causal functions on sequences

A^ω is the set of *A-valued infinite sequences*. The entries of $\sigma \in A^\omega$ are $\sigma_k \in A$ for $k \in \mathbb{N}$, so $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_k, \ldots)$.

Slicing extracts a finite list from an infinite sequence:

$$(\cdot)_{i:j} : \sigma \mapsto (\sigma_i, \sigma_{i+1}, \ldots, \sigma_j)$$

Causal functions on sequences

A^ω is the set of *A-valued infinite sequences*. The entries of $\sigma \in A^\omega$ are $\sigma_k \in A$ for $k \in \mathbb{N}$, so $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_k, \ldots)$.

Slicing extracts a finite list from an infinite sequence:

$$(\cdot)_{i:j} : \sigma \mapsto (\sigma_i, \sigma_{i+1}, \ldots, \sigma_j)$$

Definition

A *causal* function $f : A^\omega \to B^\omega$ satisfies

$$\forall \sigma, \tau \in A^\omega, \forall k \in \mathbb{N}, \sigma_{0:k} = \tau_{0:k} \to f(\sigma)_{0:k} = f(\tau)_{0:k}$$

Intuitively, the first k outputs of f only depend on the first k inputs.

Examples and non-examples

We specialize from arbitrary sets A to $\mathbb{R}$.

1. $\overline{r} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\overline{r}(\sigma)]_k = r$ is causal. The kth output depends on nothing.

Examples and non-examples

We specialize from arbitrary sets A to $\mathbb{R}$.

1. $\overline{r} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\overline{r}(\sigma)]_k = r$ is causal. The kth output depends on nothing.

2. If $g : \mathbb{R} \to \mathbb{R}$, then $\mathtt{map}(g) : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\mathtt{map}(g)(\sigma)]_k = g(\sigma_k)$ is causal. The kth output of this function depends on the kth input only.

Examples and non-examples

We specialize from arbitrary sets A to $\mathbb{R}$.

1. $\overline{r} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\overline{r}(\sigma)]_k = r$ is causal. The kth output depends on nothing.

2. If $g : \mathbb{R} \to \mathbb{R}$, then $\mathtt{map}(g) : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\mathtt{map}(g)(\sigma)]_k = g(\sigma_k)$ is causal. The kth output of this function depends on the kth input only.

3. $\mathtt{runProd} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\mathtt{runProd}(\sigma)]_k = \prod_{i=0}^{k} \sigma_i$ is causal. The kth output depends on all the first k inputs.

Examples and non-examples

We specialize from arbitrary sets A to $\mathbb{R}$.

1. $\overline{r} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\overline{r}(\sigma)]_k = r$ is causal. The kth output depends on nothing.

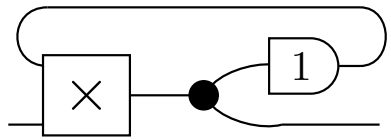
2. If $g : \mathbb{R} \to \mathbb{R}$, then $\mathtt{map}(g) : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\mathtt{map}(g)(\sigma)]_k = g(\sigma_k)$ is causal. The kth output of this function depends on the kth input only.

3. $\mathtt{runProd} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\mathtt{runProd}(\sigma)]_k = \prod_{i=0}^{k} \sigma_i$ is causal. The kth output depends on all the first k inputs.

4. $\mathtt{tl} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ defined by $[\mathtt{tl}(\sigma)]_k = \sigma_{k+1}$ is **not causal**, since output k is input $k + 1$.

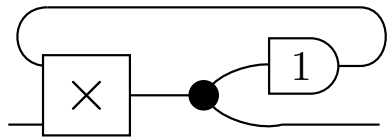
Dataflow diagrams

Dataflow diagrams are a common visual language for defining causal functions. For example, this is one of the example causal functions:

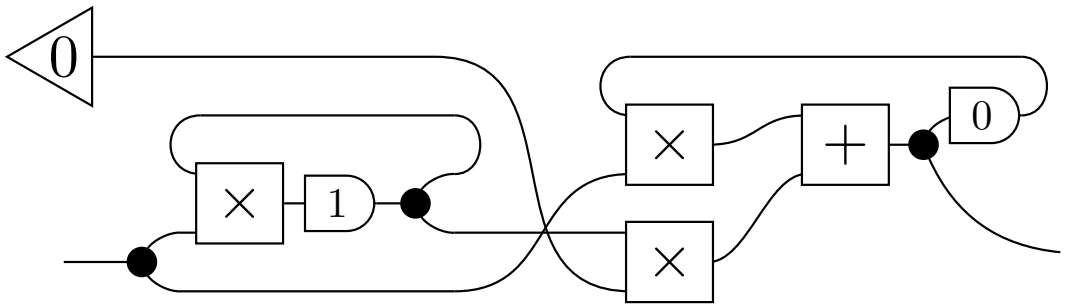


Dataflow diagrams

Dataflow diagrams are a common visual language for defining causal functions. For example, this is one of the example causal functions:

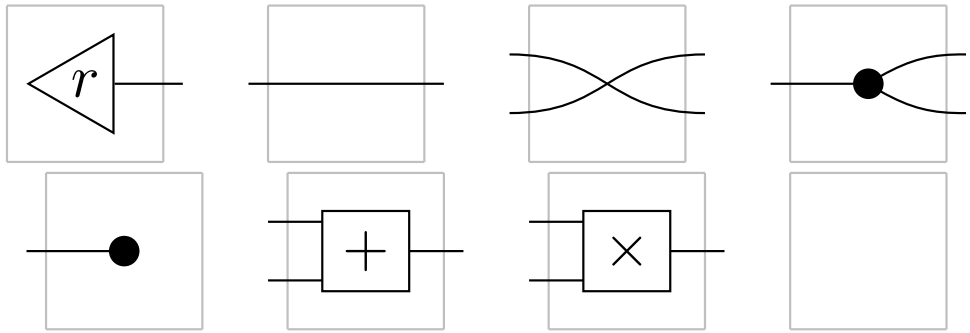


The first diagram we saw is another example, related to the derivative of the diagram above:

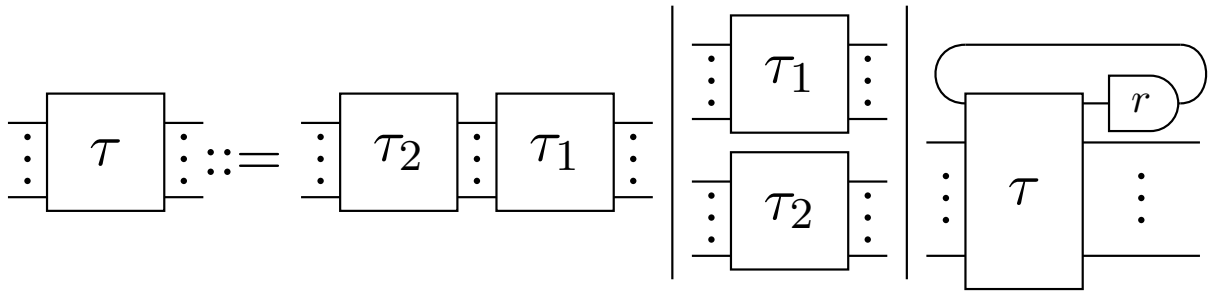


Dataflow diagrams

Dataflow diagrams are constructed from these basic components:



And closed under these operations:



In the above rules, $r \in \mathbb{R}$ is arbitrary.

Linearized language

We can use a 1-D syntax set up to parallel the 2-D syntax of diagrams. Types are $[n, m]$ (think of $[n, m] \leftrightsquigarrow \mathbb{R}^n \to \mathbb{R}^m$), and symbols are:

$$\texttt{const}_r : [0, 1] \mid \texttt{id}_1 : [1, 1] \mid \texttt{swap} : [2, 2] \mid \texttt{dup} : [1, 2] \mid$$
$$\texttt{del} : [1, 0] \mid + : [2, 1] \mid \times : [2, 1] \mid \texttt{id}_0 : [0, 0]$$

Operations take the following types (superscripts usually omitted):

$$\circ^{\ell,m,n} : [m, n] \times [\ell, m] \to [\ell, n]$$
$$\otimes^{n,p,\ell,m} : [n, p] \times [\ell, m] \to [n + \ell, p + m]$$
$$tr_r^{m,n} : [m + 1, n + 1] \to [m, n]$$

Linearized language

We can use a 1-D syntax set up to parallel the 2-D syntax of diagrams. Types are $[n, m]$ (think of $[n, m] \leftrightsquigarrow \mathbb{R}^n \to \mathbb{R}^m$), and symbols are:

$$\texttt{const}_r : [0, 1] \mid \texttt{id}_1 : [1, 1] \mid \texttt{swap} : [2, 2] \mid \texttt{dup} : [1, 2] \mid$$
$$\texttt{del} : [1, 0] \mid + : [2, 1] \mid \times : [2, 1] \mid \texttt{id}_0 : [0, 0]$$

Operations take the following types (superscripts usually omitted):

$$\circ^{\ell,m,n} : [m, n] \times [\ell, m] \to [\ell, n]$$
$$\otimes^{n,p,\ell,m} : [n, p] \times [\ell, m] \to [n + \ell, p + m]$$
$$tr_r^{m,n} : [m + 1, n + 1] \to [m, n]$$

In this language, the running product function is $tr_1(\texttt{dup} \circ \times)$.

Linearized language

We can use a 1-D syntax set up to parallel the 2-D syntax of diagrams. Types are $[n, m]$ (think of $[n, m] \leftrightsquigarrow \mathbb{R}^n \to \mathbb{R}^m$), and symbols are:

$$\mathtt{const}_r : [0, 1] \mid \mathtt{id}_1 : [1, 1] \mid \mathtt{swap} : [2, 2] \mid \mathtt{dup} : [1, 2] \mid$$
$$\mathtt{del} : [1, 0] \mid + : [2, 1] \mid \times : [2, 1] \mid \mathtt{id}_0 : [0, 0]$$

Operations take the following types (superscripts usually omitted):

$$\circ^{\ell,m,n} : [m, n] \times [\ell, m] \to [\ell, n]$$
$$\otimes^{n,p,\ell,m} : [n, p] \times [\ell, m] \to [n + \ell, p + m]$$
$$tr_r^{m,n} : [m + 1, n + 1] \to [m, n]$$

In this language, the running product function is $tr_1(\mathtt{dup} \circ \times)$.

We may also use variables in terms.

(Vague) problem statement

Given:

- a set of equations between terms in this language
- a size function $\|\cdot\|$ sending closed terms to non-negative real numbers

Produce:

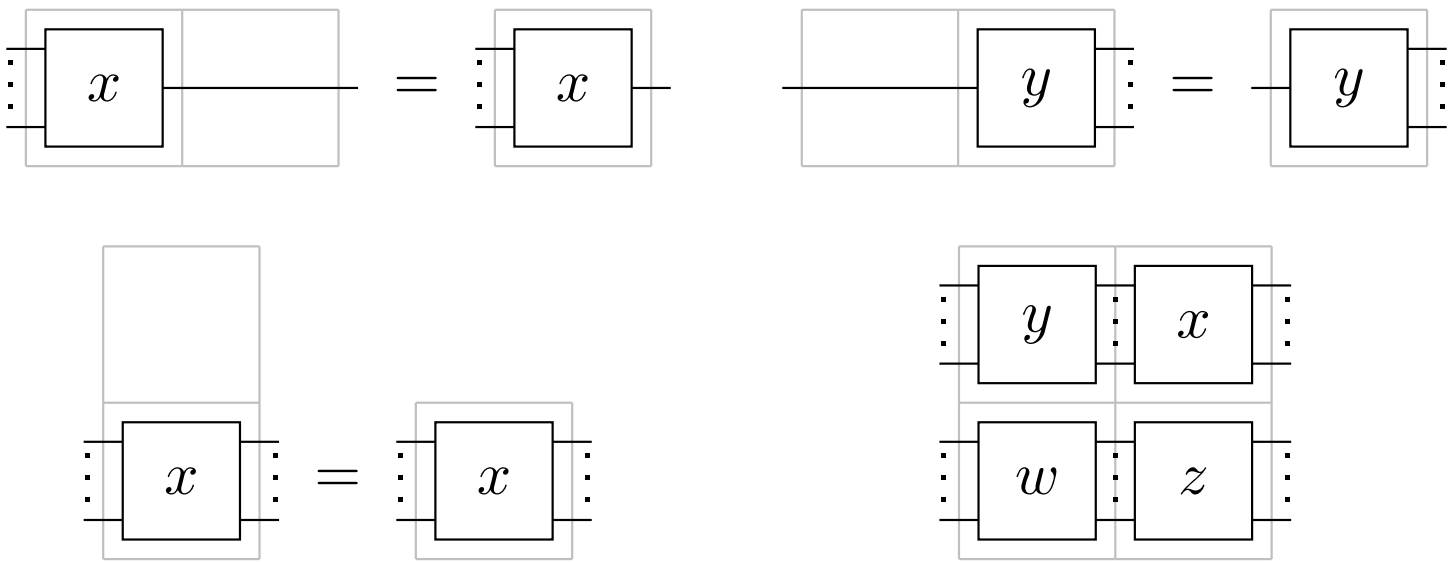
- a rewrite system taking terms to least-cost terms obtainable by applying equations

We may assume:

- equations are contextually closed under $\circ$, $\otimes$, and tr_r
- equations are closed under substitutions
- if τ_1 is a subterm of τ_2, then $\|\tau_1\| \leq \|\tau_2\|$

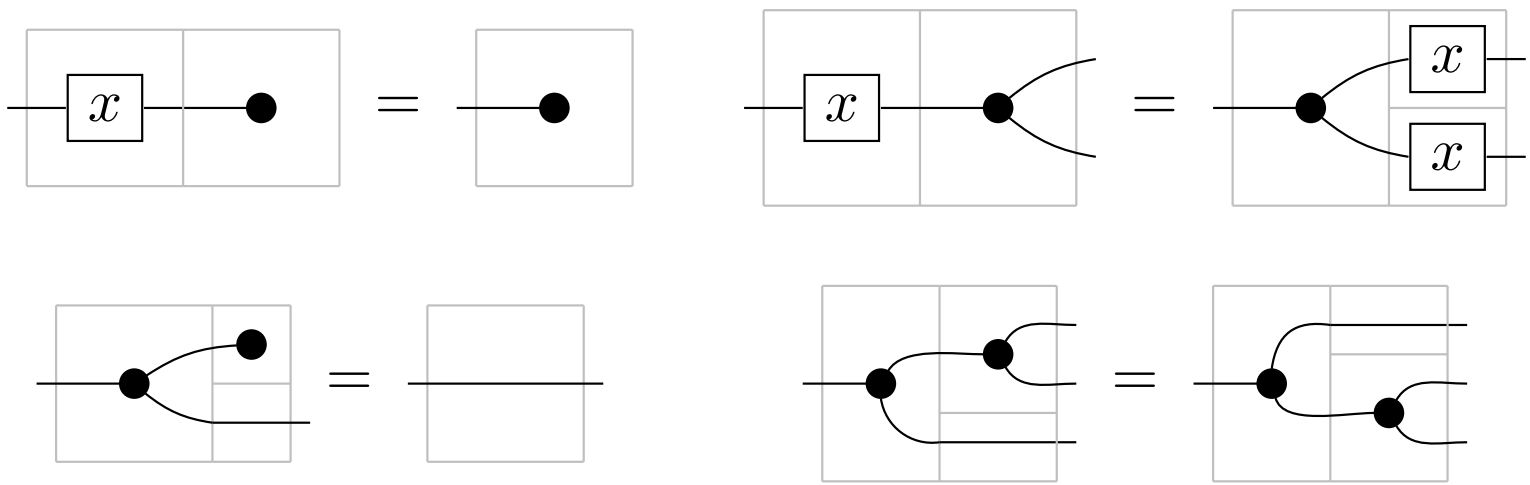
Equations I: composition

1. $\mathtt{id}_1 \circ x = x$ and $y = y \circ \mathtt{id}_1$ for all $x : [n, 1]$, $y : [1, n]$
2. $\mathtt{id}_0 \circ x = x$ and $y = y \circ \mathtt{id}_0$ for $x : [n, 0]$, $y : [0, n]$
3. $\circ$ and $\otimes$ are associative (not necessarily commutative)
4. $\mathtt{id}_0 \otimes x = x = x \otimes \mathtt{id}_0$ for all $x : [m, n]$
5. $(x \circ y) \otimes (z \circ w) = (x \otimes z) \circ (y \otimes w)$ for all ...



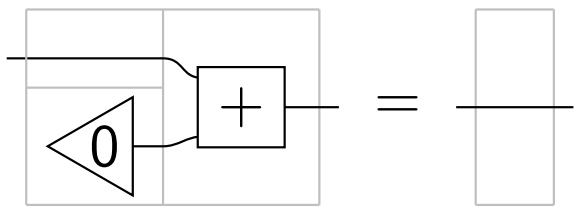
Equations II: resource management

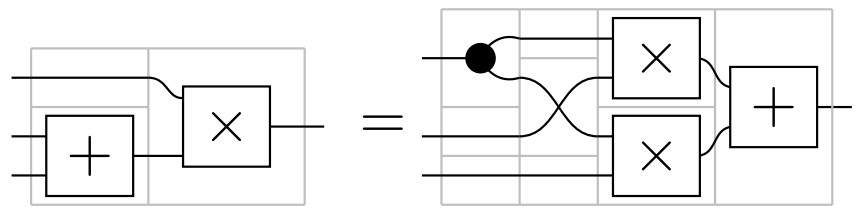
1. $\mathtt{swap} \circ \mathtt{swap} = \mathtt{id}_1 \otimes \mathtt{id}_1$
2. $\mathtt{swap} \circ (x \otimes y) = (y \otimes x) \circ \mathtt{swap}$ for all $x, y : [1, 1]$
3. $\mathtt{del} \circ x = \mathtt{del}$ for all $x : [1, 1]$
4. $\mathtt{dup} \circ x = (x \otimes x) \circ \mathtt{dup}$ for all $x : [1, 1]$
5. $(\mathtt{del} \otimes \mathtt{id}_1) \circ \mathtt{dup} = \mathtt{id}_1$
6. $\mathtt{swap} \circ \mathtt{dup} = \mathtt{dup}$
7. $(\mathtt{id}_1 \otimes \mathtt{dup}) \circ \mathtt{dup} = (\mathtt{dup} \otimes \mathtt{id}_1) \circ \mathtt{dup}$

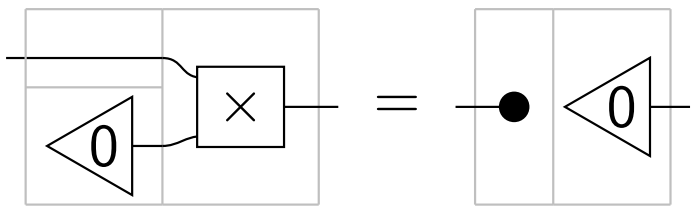


Equations III: semiring axioms

1. $+ \circ \mathtt{swap} = +$ (also for $\times$)
2. $+ \circ (\mathtt{id}_1 \otimes +) = + \circ (+ \otimes \mathtt{id}_1)$ (also for $\times$)
3. $+ \circ (\mathtt{id}_1 \otimes \mathtt{const}_0) = \mathtt{id}_1$ and $\times \circ (\mathtt{id}_1 \otimes \mathtt{const}_1) = \mathtt{id}_1$
4. $\times \circ (\mathtt{id}_1 \otimes \mathtt{const}_0) = \mathtt{const}_0 \circ \mathtt{del}$
5. $+ \circ (\mathtt{const}_r \otimes \mathtt{const}_s) = \mathtt{const}_{r+s}$ for all $r, s \in \mathbb{R}$
6. $\times \circ (\mathtt{const}_r \otimes \mathtt{const}_s) = \mathtt{const}_{rs}$ for all $r, s \in \mathbb{R}$
7. $\times \circ (\mathtt{id}_1 \otimes +) =$
   $+ \circ (\times \otimes \times) \circ (\mathtt{id}_1 \otimes \mathtt{swap} \otimes \mathtt{id}_1) \circ (\mathtt{dup} \otimes \mathtt{id}_1 \otimes \mathtt{id}_1)$

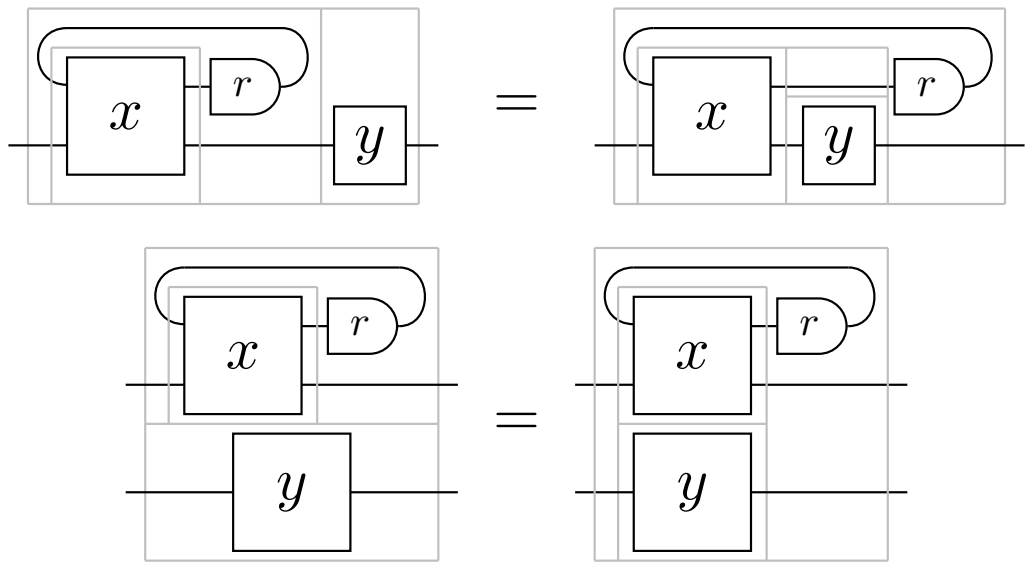






Equations IV: trace axioms

1. $tr_r(x) \circ y = tr_r(x \circ (\mathtt{id}_1 \otimes y))$ for all ...
2. $y \circ tr_r(x) = tr_r((\mathtt{id}_1 \otimes y) \circ x)$ for all ...
3. $tr_r(x) \otimes y = tr_r(x \otimes y)$ for all ...
4. $tr_r(\mathtt{dup}) = \mathtt{const}_r$



Symbol sizes

Nullary symbols represent machine instructions or simple components of circuitry. We are given some basic information about the resources they consume: $\| \cdot \| : \Sigma_0 \to [0, \infty)$. Think of this as measuring time or power.

Symbol sizes

Nullary symbols represent machine instructions or simple components of circuitry. We are given some basic information about the resources they consume: $\|\cdot\| : \Sigma_0 \to [0, \infty)$. Think of this as measuring time or power.

For example,

$$\|\mathtt{id}_0\| = 0 \quad \|\mathtt{id}_1\| = 1 \quad \|\mathtt{del}\| = 5 \quad \|\mathtt{const}_r\| = 7$$
$$\|+\| = 10 \quad \|\mathtt{dup}\| = 15 \quad \|\mathtt{swap}\| = 30 \quad \|\times\| = 40$$

Term sizes

There are various ways to extend these to terms:

$$\|\tau\|_{\mathsf{work}} \triangleq \begin{cases} \|\tau\| & \text{if } \tau \in \Sigma_0 \\ \|\tau_1\|_{\mathsf{work}} + \|\tau_2\|_{\mathsf{work}} & \text{if } \tau = \tau_2 \circ \tau_1 \\ \|\tau_1\|_{\mathsf{work}} + \|\tau_2\|_{\mathsf{work}} & \text{if } \tau = \tau_1 \otimes \tau_2 \\ \|\tau_1\|_{\mathsf{work}} + \epsilon & \text{if } \tau = tr_r(\tau_1) \end{cases}$$

Term sizes

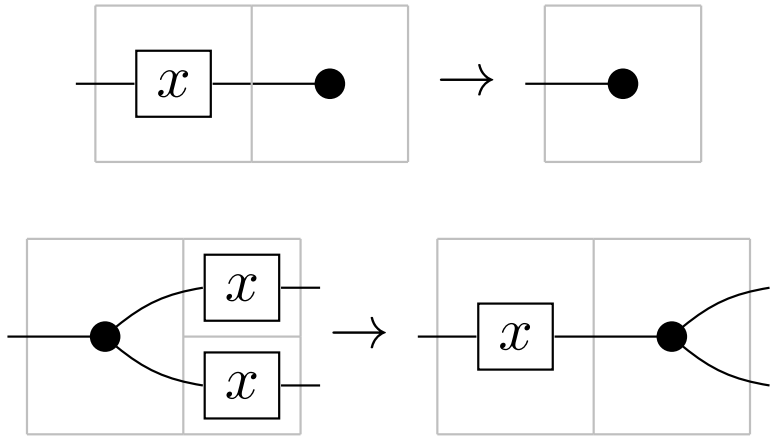
There are various ways to extend these to terms:

$$\|\tau\|_{\mathsf{work}} \triangleq \begin{cases} \|\tau\| & \text{if } \tau \in \Sigma_0 \\ \|\tau_1\|_{\mathsf{work}} + \|\tau_2\|_{\mathsf{work}} & \text{if } \tau = \tau_2 \circ \tau_1 \\ \|\tau_1\|_{\mathsf{work}} + \|\tau_2\|_{\mathsf{work}} & \text{if } \tau = \tau_1 \otimes \tau_2 \\ \|\tau_1\|_{\mathsf{work}} + \epsilon & \text{if } \tau = tr_r(\tau_1) \end{cases}$$

$$\|\tau\|_{\mathsf{span}} \triangleq \begin{cases} \|\tau\| & \text{if } \tau \in \Sigma_0 \\ \|\tau_1\|_{\mathsf{span}} + \|\tau_2\|_{\mathsf{span}} & \text{if } \tau = \tau_2 \circ \tau_1 \\ \max(\|\tau_1\|_{\mathsf{span}}, \|\tau_2\|_{\mathsf{span}}) & \text{if } \tau = \tau_1 \otimes \tau_2 \\ \|\tau_1\|_{\mathsf{span}} + \epsilon' & \text{if } \tau = tr_r(\tau_1) \end{cases}$$

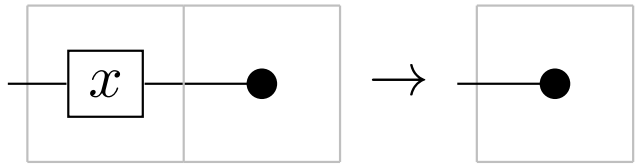
Orienting equations by size

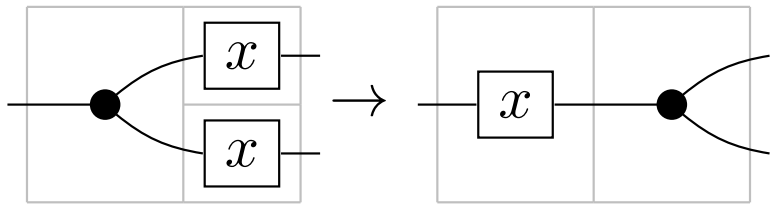
Two important orientable equations include:



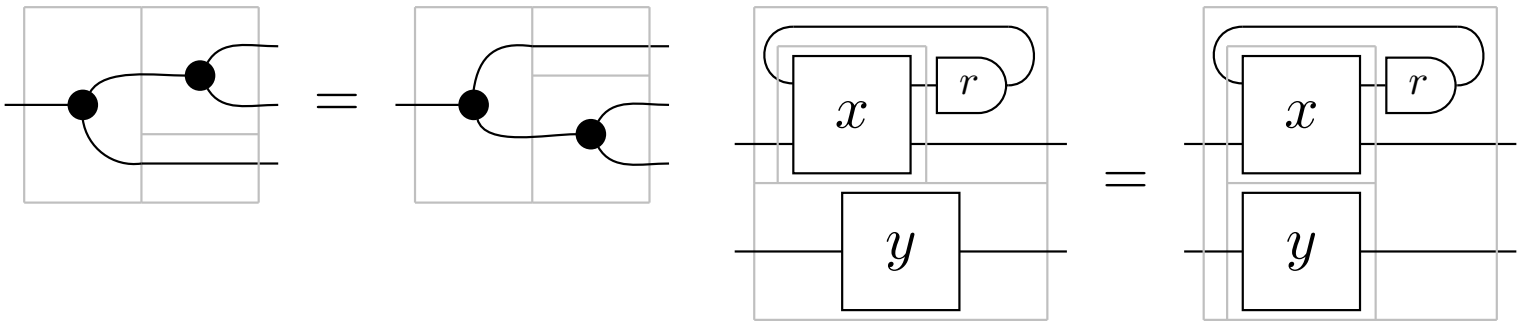
Orienting equations by size

Two important orientable equations include:



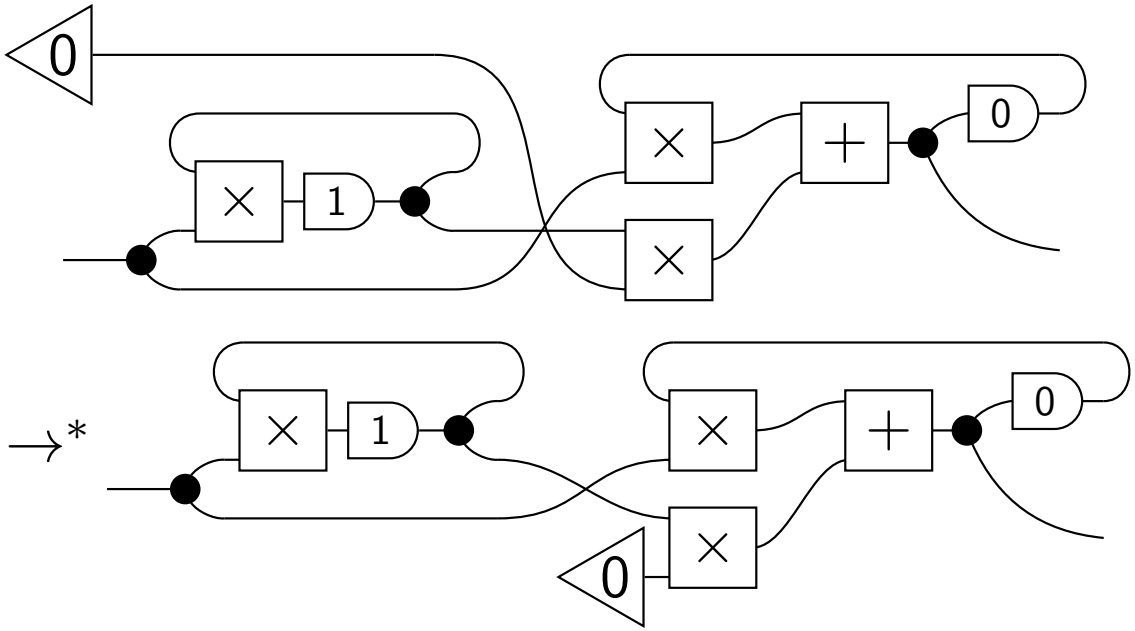


Many other equations don't have an obvious orientation, like:



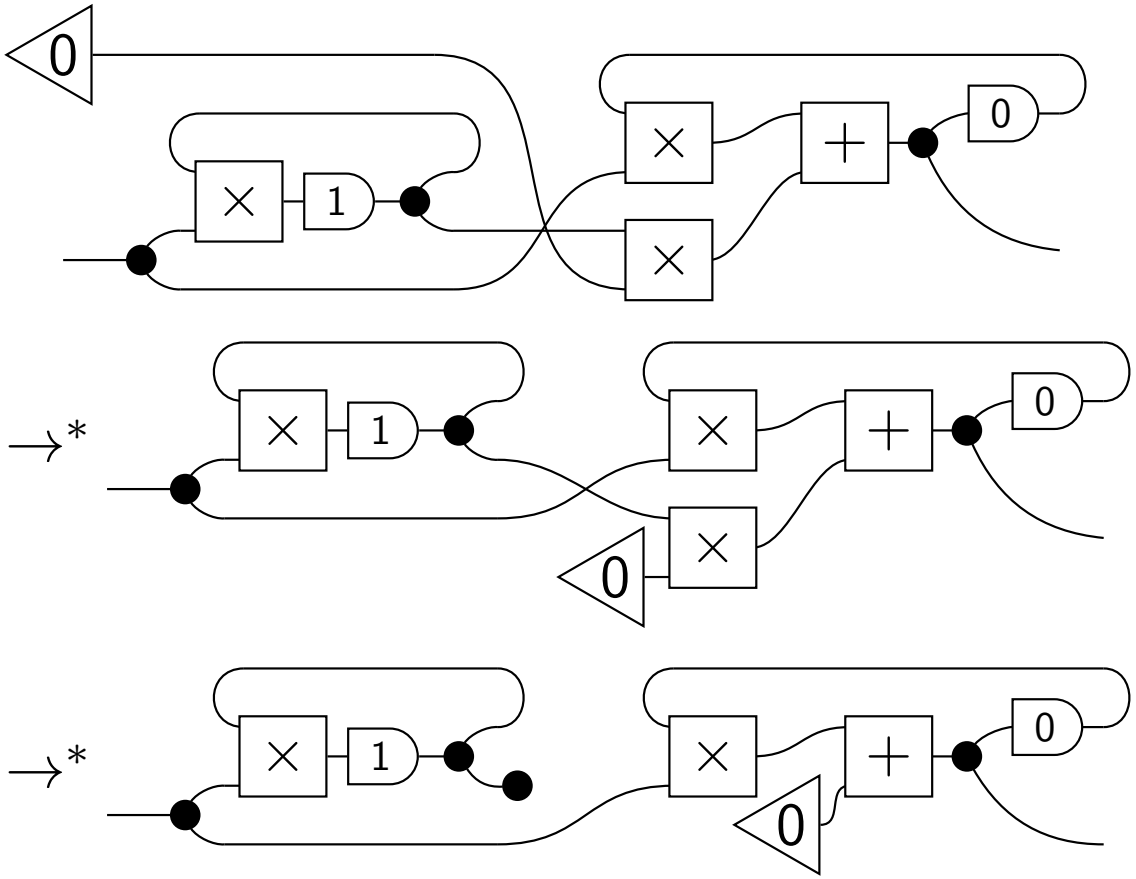
Rewriting to produce smaller diagrams

We would like a rewriting system that helps obtain the following reasoning.



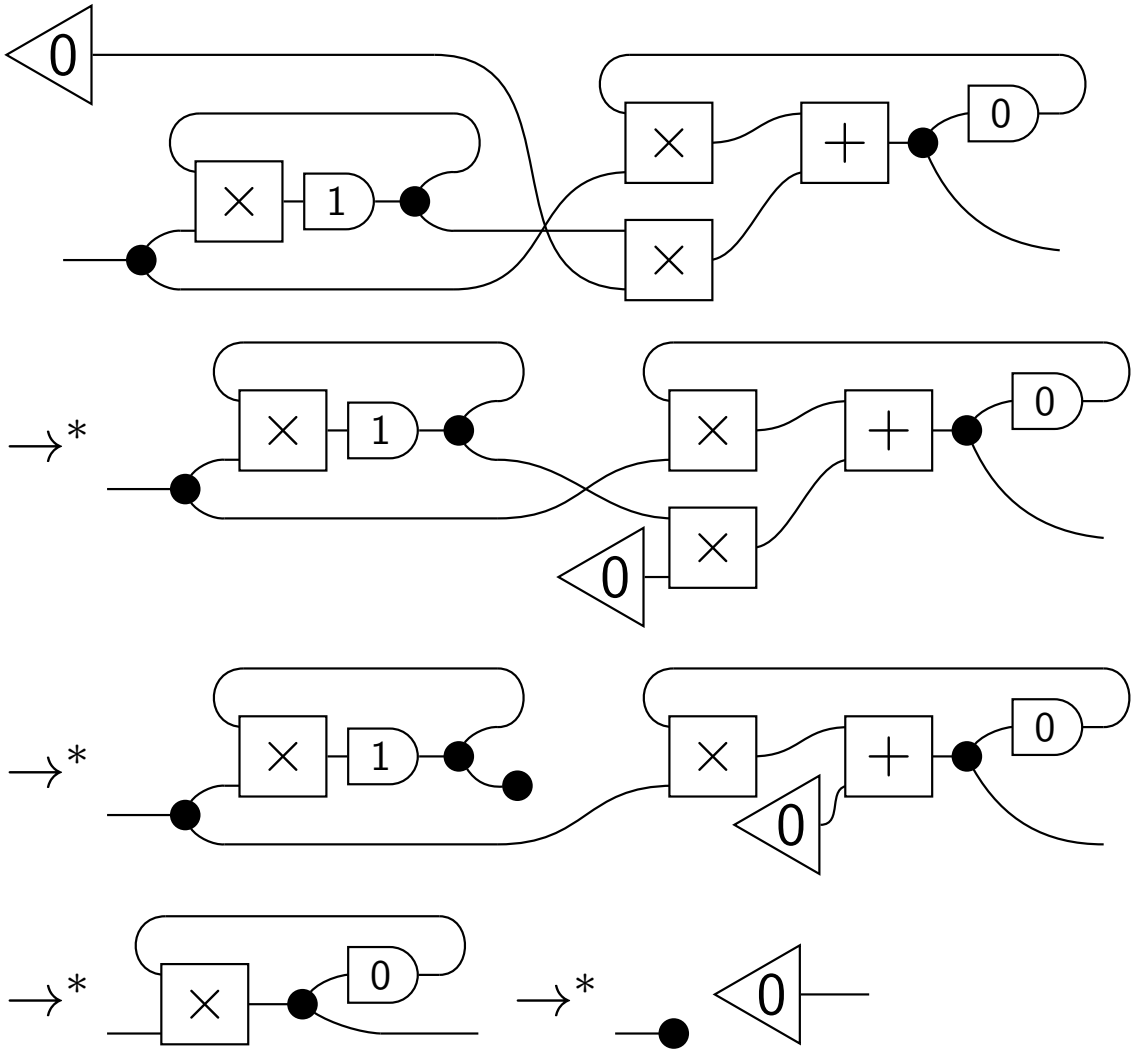
Rewriting to produce smaller diagrams

We would like a rewriting system that helps obtain the following reasoning.



Rewriting to produce smaller diagrams

We would like a rewriting system that helps obtain the following reasoning.



Problem extensions

1. Allow infinitely many symbols and equations, often defined by recursion. For example, $\mathtt{id}_n := \mathtt{id}_1 \otimes \mathtt{id}_{n-1}$ and $f = f \circ \mathtt{id}_n$ for all $f : [n, m]$
2. If we add finitely many symbols and equations, can we obtain a rewrite system for the new language from the rewrite system for the old language? (e.g. adding $\sin, \cos : [1, 1]$ and related equations)
3. Add further typing constraints to the wires in addition to co/arity, e.g. $\mathtt{if} : \mathbb{B} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$
4. Constrained optimizations, e.g. minimize $\|\tau\|_{\text{work}}$ subject to $\|\tau\|_{\text{span}} \leq T$
5. Multi-objective optimizations, e.g. find the Pareto front for $\|\tau\|_{\text{work}}$ and $\|\tau\|_{\text{span}}$

Thanks!