Inversion and Term Rewriting Systems

Maja H. Kirkeby[1] Robert Glück[2]

kirkebym@acm.org, Roskilde University, Denmark [1]

glueck@acm.org, University of Copenhagen, Denmark [2]

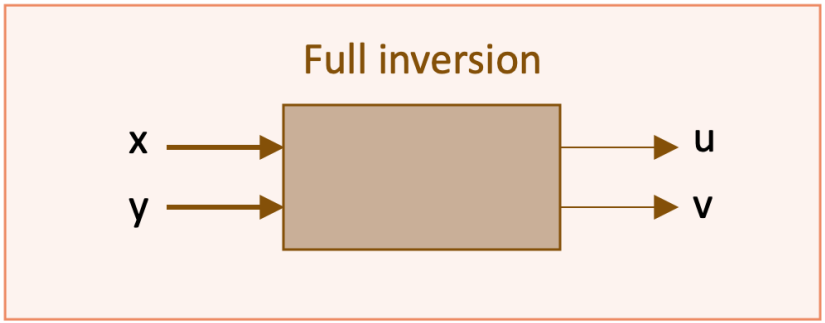
12th International School on Rewriting,
Virtual at Universidad Complutense de Madrid, Spain,
July 2021

Full Inversion, Partial Inversion, and Semi-inversion

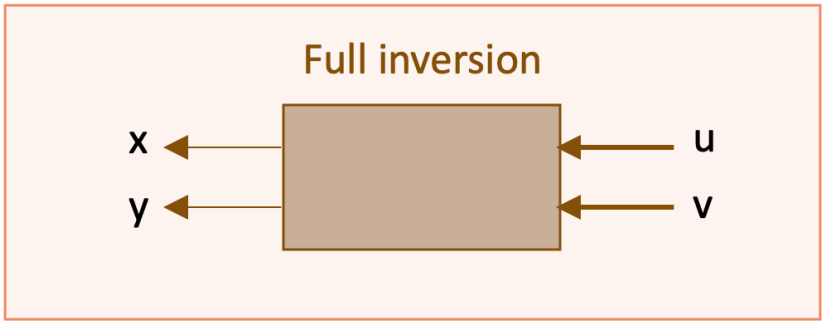
Full Inversion, Partial Inversion, and Semi-inversion

Full inversion

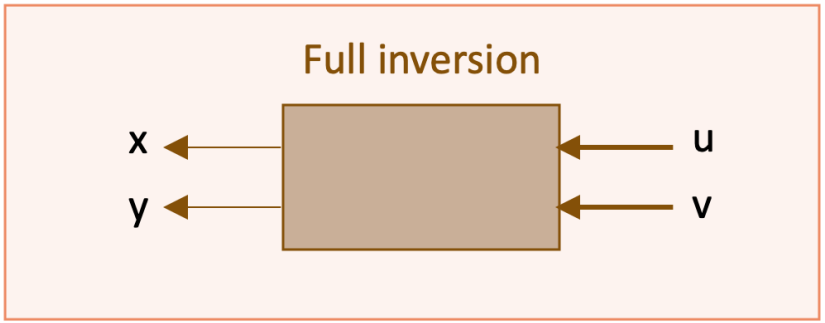
Full Inversion, Partial Inversion, and Semi-inversion

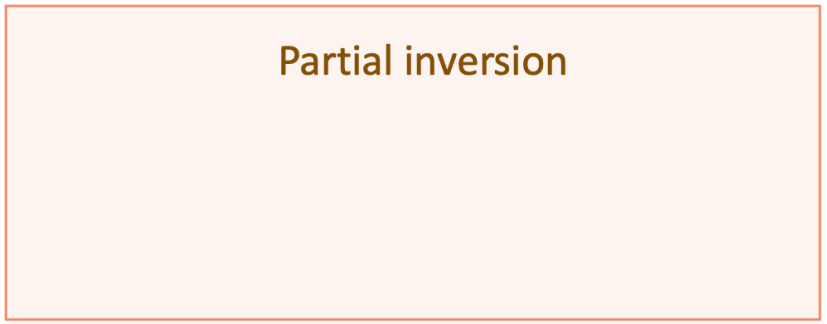


Full Inversion, Partial Inversion, and Semi-inversion

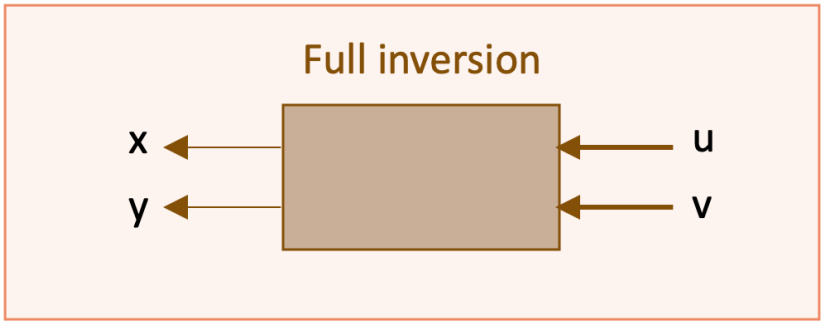


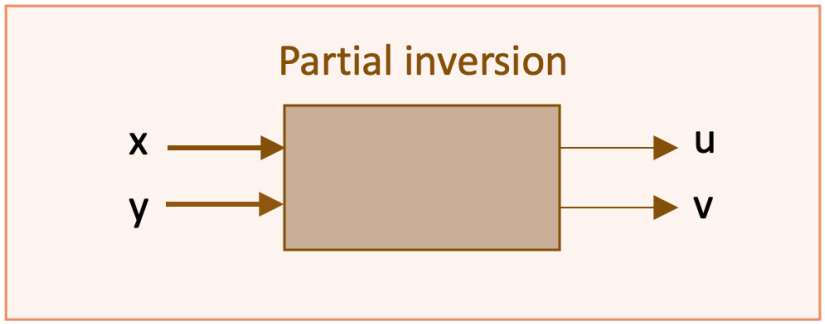
Full Inversion, Partial Inversion, and Semi-inversion



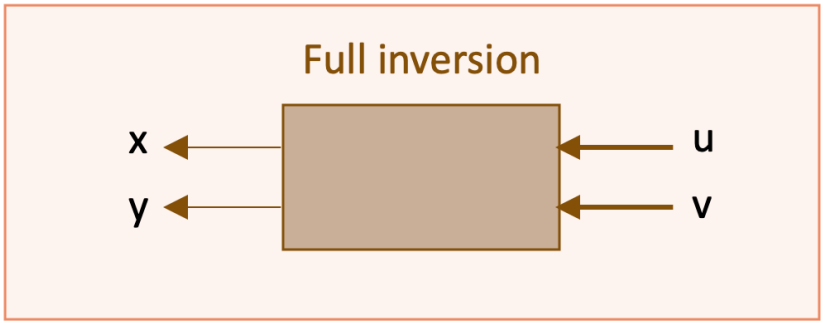


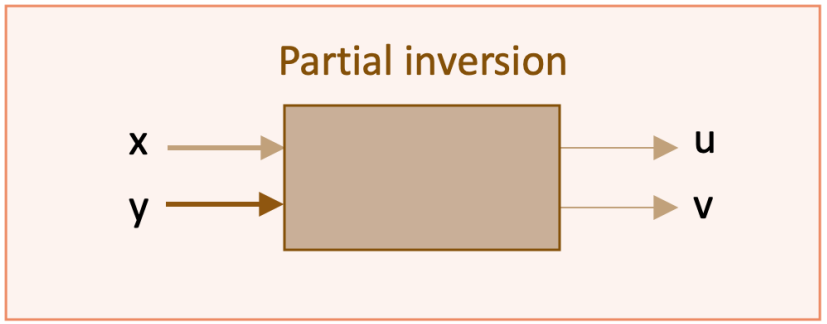
Full Inversion, Partial Inversion, and Semi-inversion



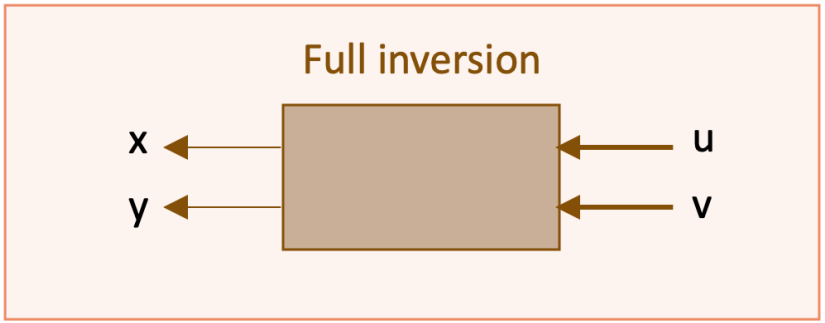


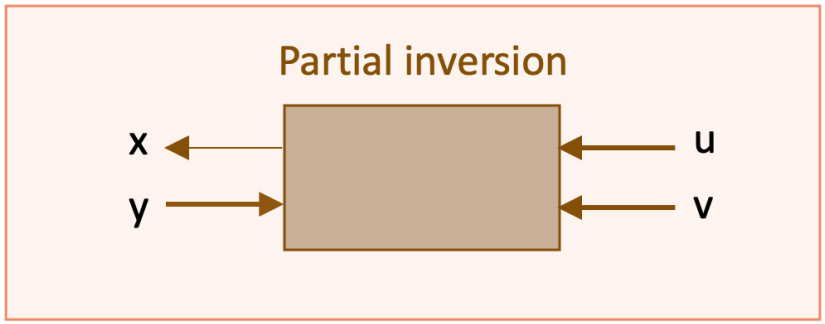
Full Inversion, Partial Inversion, and Semi-inversion



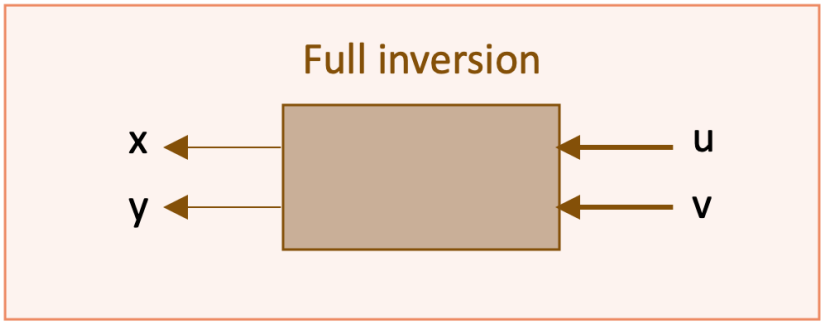


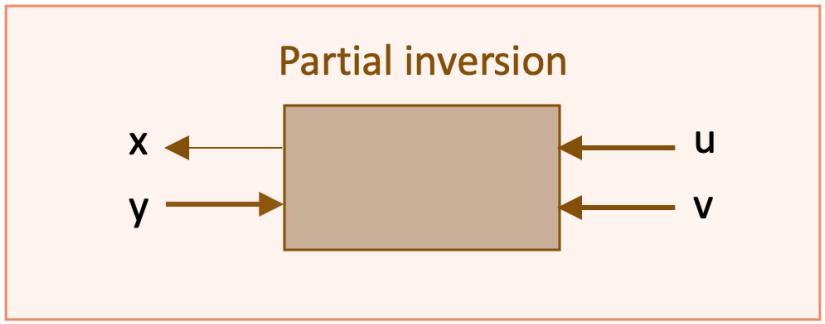
Full Inversion, Partial Inversion, and Semi-inversion

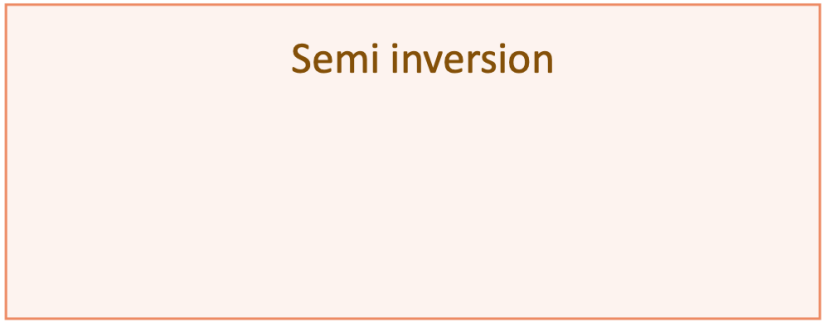




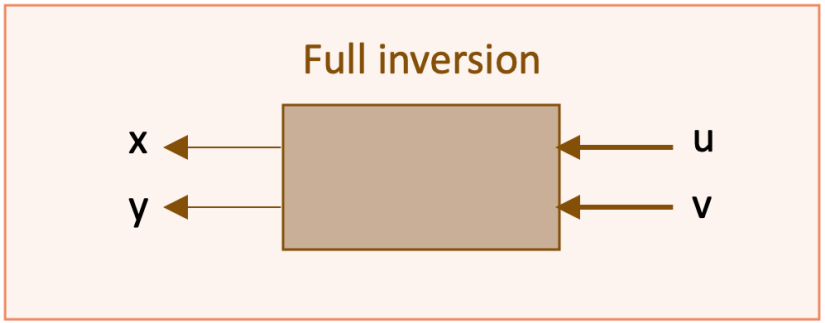
Full Inversion, Partial Inversion, and Semi-inversion

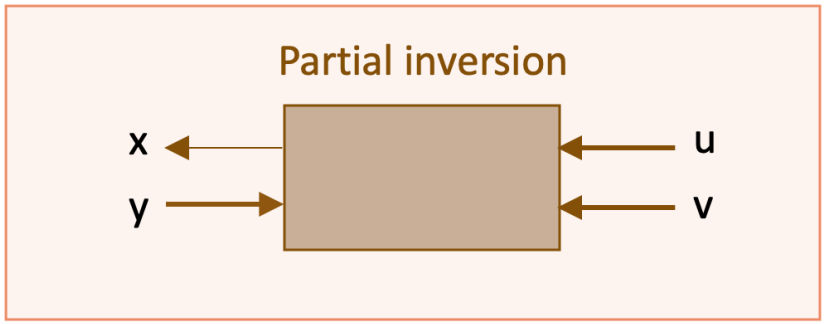


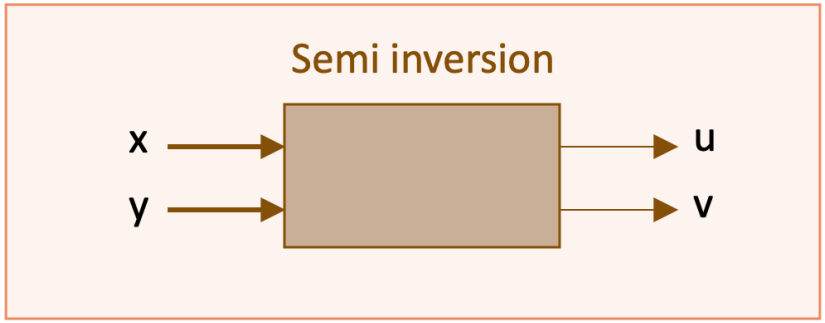




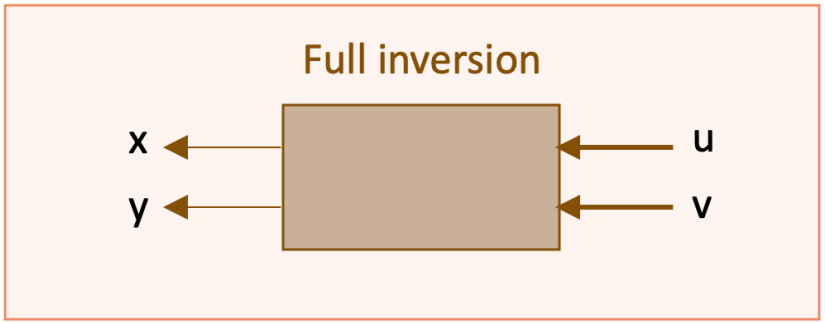
Full Inversion, Partial Inversion, and Semi-inversion

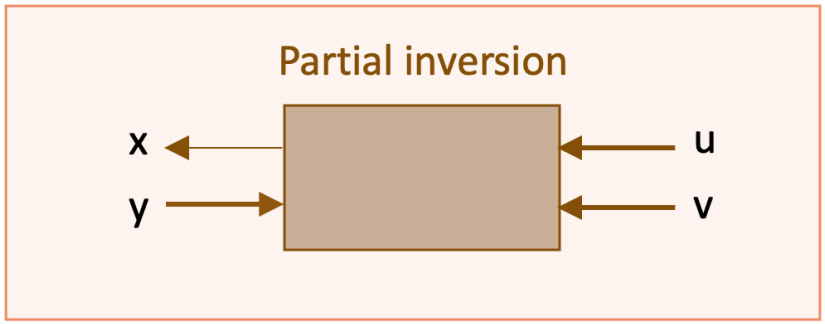


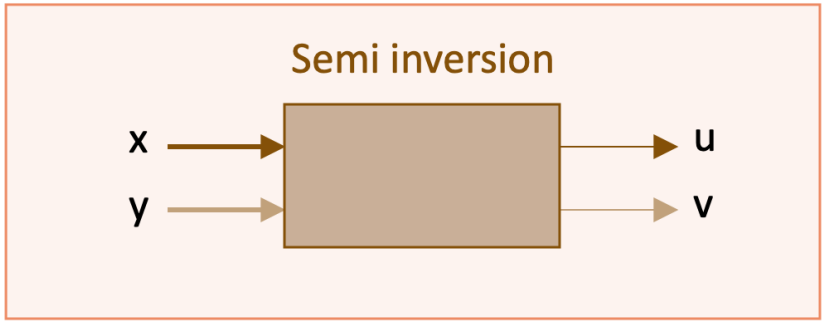




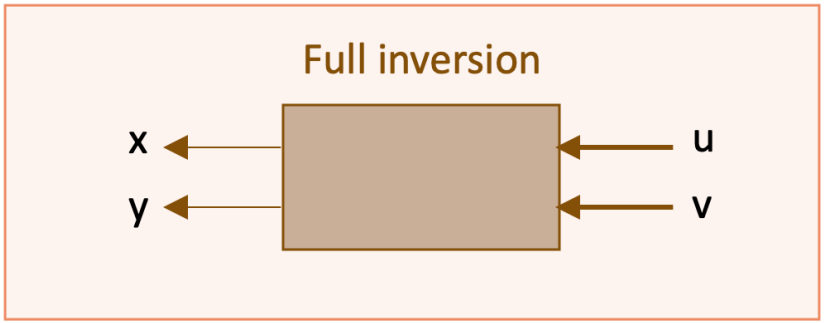
Full Inversion, Partial Inversion, and Semi-inversion

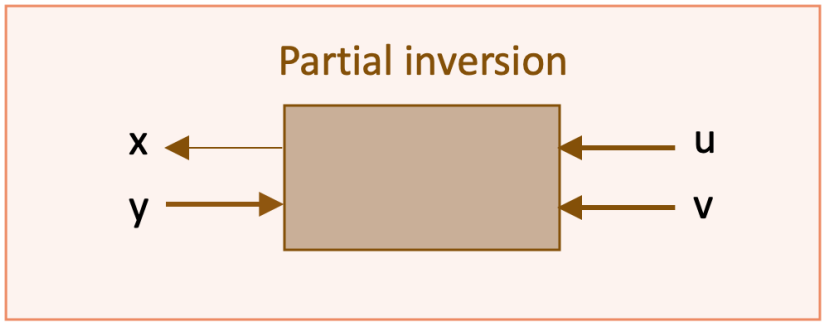


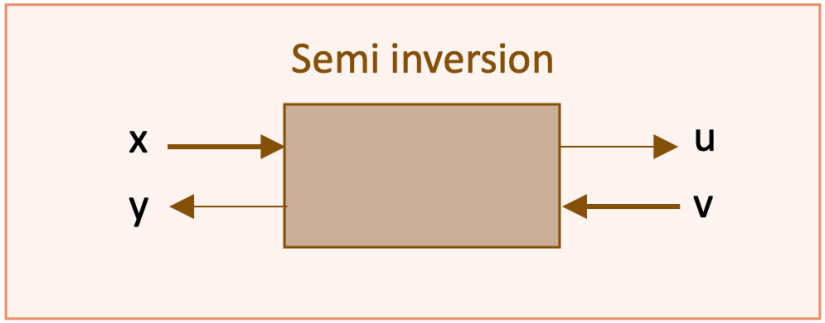




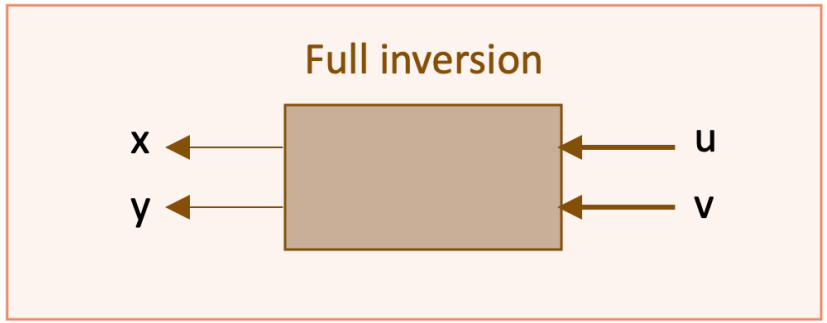
Full Inversion, Partial Inversion, and Semi-inversion



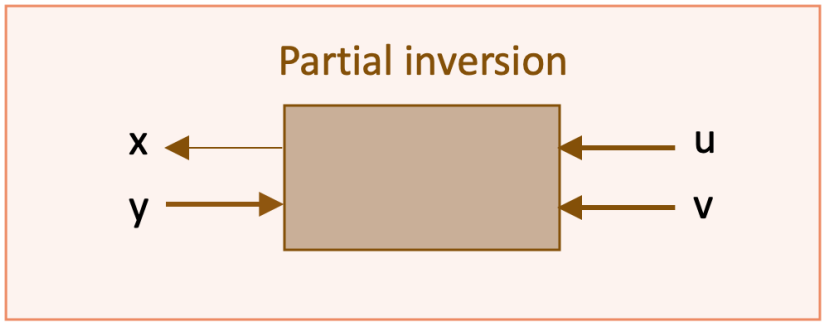




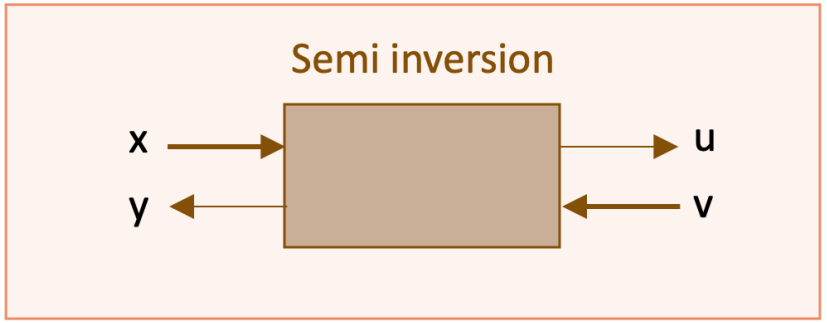
Full Inversion, Partial Inversion, and Semi-inversion



$\cap\!\!\mid$



$\cap\!\!\mid$



Inversions – Programs as Relations

Inversions – Programs as Relations

divide

$$[1,2,3] \to \langle [], [1,2,3] \rangle$$
$$[1,2,3] \to \langle [1], [2,3] \rangle$$
$$[1,2,3] \to \langle [1,2], [3] \rangle$$
$$[1,2,3] \to \langle [1,2,3], [] \rangle$$

Inversions – Programs as Relations

Full Inv.

append (function)

$$\begin{tikzcd}
([], [1,2,3]) \arrow[dr] & \\
([1], [2,3]) \arrow[r] & \langle [1,2,3] \rangle \\
([1,2], [3]) \arrow[ur] & \\
([1,2,3], []) \arrow[uur] &
\end{tikzcd}$$

divide

$$\begin{tikzcd}
 & \langle [], [1,2,3] \rangle \\
[1,2,3] \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & \langle [1], [2,3] \rangle \\
 & \langle [1,2], [3] \rangle \\
 & \langle [1,2,3], [] \rangle
\end{tikzcd}$$

Inversions – Programs as Relations

Full Inv.

append (function)

$$\begin{tikzcd}[column sep=small] ([],[1,2,3]) \arrow[dr] & \\ ([1],[2,3]) \arrow[r] & \langle[1,2,3]\rangle \\ ([1,2],[3]) \arrow[ur] & \\ ([1,2,3],[]) \arrow[uur] & \end{tikzcd}$$

divide

$$\begin{tikzcd}[column sep=small] & \langle[],[1,2,3]\rangle \\ [1,2,3] \arrow[ur] \arrow[r] \arrow[dr] & \langle[1],[2,3]\rangle \\ & \langle[1,2],[3]\rangle \\ & \langle[1,2,3],[]\rangle \end{tikzcd}$$

Part. Inv.

append (pred.)

$$\begin{tikzcd}[column sep=small] ([],[1,2,3],[1,2,3]) \arrow[dr] & \\ ([1],[2,3],[1,2,3]) \arrow[r] & \langle\rangle \\ ([1,2],[3],[1,2,3]) \arrow[ur] & \\ ([1,2,3],[],[1,2,3]) \arrow[uur] & \end{tikzcd}$$

Inversions – Programs as Relations

divide

$[1,2,3] \to \langle [], [1,2,3] \rangle$

$[1,2,3] \to \langle [1], [2,3] \rangle$

$[1,2,3] \to \langle [1,2], [3] \rangle$

$[1,2,3] \to \langle [1,2,3], [] \rangle$

Full Inv.

append (function)

$([], [1,2,3]) \to \langle [1,2,3] \rangle$

$([1], [2,3]) \to \langle [1,2,3] \rangle$

$([1,2], [3]) \to \langle [1,2,3] \rangle$

$([1,2,3], []) \to \langle [1,2,3] \rangle$

Part. Inv.

append (pred.)

$([], [1,2,3], [1,2,3]) \to \langle \rangle$

$([1], [2,3], [1,2,3]) \to \langle \rangle$

$([1,2], [3], [1,2,3]) \to \langle \rangle$

$([1,2,3], [], [1,2,3]) \to \langle \rangle$

Semi-inv.

drop prefix

$([1,2,3], []) \longrightarrow \langle [1,2,3] \rangle$

$([1,2,3], [1]) \longrightarrow \langle [2,3] \rangle$

$([1,2,3], [1,2]) \longrightarrow \langle [3] \rangle$

$([1,2,3], [1,2,3]) \longrightarrow \langle [] \rangle$

Related work (excerpt)

	Functions	**Relations**
Full Inversion	Glück and Kawabe (2003)	Nishida *et al.* (2004,2011)
Partial Inversion	Almendros-Jiménez *et al.* (2006)	Romanenko (1988) Nishida *et al.* (2005)
Semi-inversion	Mogensen (2005,2008)	Romanenko (1991) Kirkeby and Glück (2020)

Related work (excerpt)

	Functions	**Relations**
Full Inversion	Glück and Kawabe (2003)	Nishida *et al.* (2004,2011)
Partial Inversion	Almendros-Jiménez *et al.* (2006)	Romanenko (1988) Nishida *et al.* (2005)
Semi-inversion	Mogensen (2005,2008)	Romanenko (1991) Kirkeby and Glück (2020)
Comparison		Kirkeby and Glück (2020) Mikkelsen *et al.* (2021)

Related work (excerpt)

	Functions	**Relations**
Full Inversion	Glück and Kawabe (2003)	Nishida *et al.* (2004,2011)
Partial Inversion	Almendros-Jiménez *et al.* (2006)	Romanenko (1988) Nishida *et al.* (2005)
Semi-inversion	Mogensen (2005,2008)	Romanenko (1991) Kirkeby and Glück (2020)
Comparison		Kirkeby and Glück (2020) Mikkelsen *et al.* (2021)

Formalism – Conditional Constructor Term Rewriting

Definition

Let $\mathcal{F}$ be a finite signature that can be partitioned into two disjoint sets: a set of *defined function* symbols $\mathcal{D}$ and a set of *constructor symbols* $\mathcal{C}$; each $f/n/m \in \mathcal{D}$ has an arity n and a co-arity m, and each $a \in \mathcal{C}$ has an arity n.

Formalism – Conditional Constructor Term Rewriting

Definition

Let $\mathcal{F}$ be a finite signature that can be partitioned into two disjoint sets: a set of *defined function* symbols $\mathcal{D}$ and a set of *constructor symbols* $\mathcal{C}$; each $f/n/m \in \mathcal{D}$ has an arity n and a co-arity m, and each $a \in \mathcal{C}$ has an arity n.

Definition (CCS)

A *conditional constructor term rewriting system* $\mathcal{R}$ (*CCS*) over $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ is a finite set of rules of the form

$$l_0 \to r_0 \Leftarrow l_1 \to r_1 \wedge \ldots \wedge l_k \to r_k,$$

where each $l_i \to r_i$ is of the form $f^i(p_1^i, \ldots, p_{n_i}^i) \to \langle q_1^i, \ldots, q_{m_i}^i \rangle$ such that $f^i/n_i/m_i \in \mathcal{D}$, $\langle\,\rangle/m_i \in \mathcal{C}$, and $p_j^i, q_j^i \in T(\mathcal{C} \setminus \{\langle\rangle\})$.

Formalism – Conditional Constructor Term Rewriting

Definition

Let $\mathcal{F}$ be a finite signature that can be partitioned into two disjoint sets: a set of *defined function* symbols $\mathcal{D}$ and a set of *constructor symbols* $\mathcal{C}$; each $f/n/m \in \mathcal{D}$ has an arity n and a co-arity m, and each $a \in \mathcal{C}$ has an arity n.

Definition (CCS)

A *conditional constructor term rewriting system* $\mathcal{R}$ (*CCS*) over $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ is a finite set of rules of the form

$$l_0 \to r_0 \Leftarrow l_1 \to r_1 \wedge \ldots \wedge l_k \to r_k,$$

where each $l_i \to r_i$ is of the form $f^i(p^i_1, \ldots, p^i_{n_i}) \to \langle q^i_1, \ldots, q^i_{m_i} \rangle$ such that $f^i/n_i/m_i \in \mathcal{D}$, $\langle\,\rangle/m_i \in \mathcal{C}$, and $p^i_j, q^i_j \in T(\mathcal{C} \setminus \{\langle\rangle\})$.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \to \langle y \rangle, \quad add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle\}$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$add(s(0), 0)$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(x, y) \xrightarrow{?} \langle z \rangle$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(x, y) \xrightarrow{?} \langle z \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0\}$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0\}$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$add(s(0), 0) \overset{?}{\rightarrow} \langle s(z) \rangle \Leftarrow add(0, 0) \overset{?}{\rightarrow} \langle z \rangle \;\; \sigma = \{x \mapsto 0, y \mapsto 0\}$

$add(0, 0)$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0\}$$

$$add(0, 0) \xrightarrow{?} \langle y' \rangle$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \;\; \sigma = \{x \mapsto 0, y \mapsto 0\}$$

$$add(0, 0) \xrightarrow{?} \langle y' \rangle \;\; \sigma = \{y' \mapsto 0\}$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \;\; \sigma = \{x \mapsto 0, y \mapsto 0\}$$

$$add(0, 0) \xrightarrow{r1} \langle 0 \rangle \;\; \sigma = \{y' \mapsto 0\}$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \;\; \sigma = \{x \mapsto 0, y \mapsto 0\}$$

$$add(0, 0) \xrightarrow{r1} \langle 0 \rangle \;\; \sigma = \{y' \mapsto 0\}$$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \;\; \sigma = \{x \mapsto 0, y \mapsto 0\}$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0\}$$

$$add(0, 0) \xrightarrow{r1} \langle 0 \rangle \quad \sigma = \{y' \mapsto 0\}$$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{r1} \langle 0 \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0, z \mapsto 0\}$$

Rewriting CCS

Definition

A *ground constructor-based rewrite relation*[a] $\rightarrow_{\mathcal{R}}$ of a CTRS $\mathcal{R}$ is the smallest binary relation for all pairs of ground terms (s, t), where there is a ground constructor substitution σ and a rewrite rule $l \rightarrow r \Leftarrow c$ such that $(l \rightarrow r \Leftarrow c)\sigma$ is ground, $s = l\sigma$, $r\sigma = t$ and, for each condition $(l_i \rightarrow r_i) \in c$, $l_i\sigma \rightarrow^*_{\mathcal{R}} r_i\sigma$.

[a] The position is always the root position, and is implicit in the definition.

Example (Addition of unary numbers)

$\mathcal{C} = \{s/1, 0/0\}$ and $\mathcal{D} = \{add/2/1\}$

$\{add(0, y) \rightarrow \langle y \rangle, \quad add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle\}$

$$add(s(0), 0) \xrightarrow{?} \langle s(z) \rangle \Leftarrow add(0, 0) \xrightarrow{?} \langle z \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0\}$$

$$add(0, 0) \xrightarrow{r1} \langle 0 \rangle \quad \sigma = \{y' \mapsto 0\}$$

$$add(s(0), 0) \xrightarrow{r2} \langle s(0) \rangle \Leftarrow add(0, 0) \xrightarrow{r1} \langle 0 \rangle \quad \sigma = \{x \mapsto 0, y \mapsto 0, z \mapsto 0\}$$

Modeling Language Paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Modeling Language Paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Example (Declarative, Functional, and Reversible Programs)

Declarative :

$$\mathrm{upto}(x) \to \langle 0 \rangle$$
$$\mathrm{upto}(s(x)) \to \langle s(y) \rangle \Leftarrow \mathrm{upto}(x) \to \langle y \rangle$$

Functional :

$$\mathrm{add}(0, y) \to \langle y \rangle$$
$$\mathrm{add}(s(x), y) \to \langle s(z) \rangle \Leftarrow \mathrm{add}(x, y) \to \langle z \rangle$$

Reversible :

$$\mathrm{add}'(0, y) \to \langle y, 0 \rangle$$
$$\mathrm{add}'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow \mathrm{add}'(x, y) \to \langle z, x \rangle$$

Modeling Language Paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Example (Declarative, Functional, and Reversible Programs)

$$\textit{Declarative}: \quad \begin{array}{l} \mathsf{upto}(x) \to \langle 0 \rangle \\ \mathsf{upto}(s(x)) \to \langle s(y) \rangle \Leftarrow \mathsf{upto}(x) \to \langle y \rangle \end{array}$$

Definition

Two rules $l \to r \Leftarrow c$ and $l' \to r' \Leftarrow c'$ are

- *overlapping* if l and l' overlap and
- *output-overlapping* if $root(l) = root(l')$ and r and r' overlap.

Modeling Language Paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Example (Declarative, Functional, and Reversible Programs)

Functional :

$$\mathrm{add}(0, y) \to \langle y \rangle$$
$$\mathrm{add}(s(x), y) \to \langle s(z) \rangle \Leftarrow \mathrm{add}(x, y) \to \langle z \rangle$$

Definition

Two rules $l \to r \Leftarrow c$ and $l' \to r' \Leftarrow c'$ are

- *overlapping* if l and l' overlap and
- *output-overlapping* if $root(l) = root(l')$ and r and r' overlap.

Modeling Language Paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Example (Declarative, Functional, and Reversible Programs)

$$\textit{Reversible}: \quad \begin{aligned} &\mathrm{add}'(0, y) \to \langle y, 0 \rangle \\ &\mathrm{add}'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow \mathrm{add}'(x, y) \to \langle z, x \rangle \end{aligned}$$

Definition

Two rules $l \to r \Leftarrow c$ and $l' \to r' \Leftarrow c'$ are

- *overlapping* if l and l' overlap and
- *output-overlapping* if $root(l) = root(l')$ and r and r' overlap.

Modeling Language Paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Definition

A conditional rule $l \to r \Leftarrow c$ where $c = l_1 \to r_1 \wedge \ldots \wedge l_k \to r_k$ is

- *left-linear* if l is linear and
- *right-linear* if r is linear.

- *left-to-right deterministic* if $\mathcal{V}ar(l_i) \subseteq \mathcal{V}ar(l, r_1, \ldots, r_{i-1})$ for $1 \leq i \leq k$ and
- *weakly non-erasing* if $\mathcal{V}ar(r_i) \subseteq \mathcal{V}ar(r, l_{i+1}, \ldots, l_k)$

- *extra variables free* if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l, c)$ and
- *non-erasing* if $\mathcal{V}ar(l) \subseteq \mathcal{V}ar(r, c)$

Full Inversion

Definition (Full Inversion)

Let $\mathcal{R}$ and $\underline{\mathcal{R}}$ be CCSs over $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ and $\underline{\mathcal{F}} = \underline{\mathcal{C}} \uplus \underline{\mathcal{D}}$, respectively, with $f/n/m \in \mathcal{D}$ and $\underline{f}/m/n \in \underline{\mathcal{D}}$. Then, $\underline{\mathcal{R}}$ is a *full inverse of* $\mathcal{R}$ *w.r.t.* f if for all ground constructor terms $s_1, \ldots, s_n,\ t_1, \ldots, t_m \in T(\mathcal{C} \setminus \{\langle\rangle\}, \emptyset)$,

$$\begin{array}{lll} f(s_1, \ldots, s_n) & \to_{\mathcal{R}} & \langle t_1, \ldots, t_m \rangle \ \Leftrightarrow \\ \underline{f}(t_1, \ldots, t_m) & \to_{\underline{\mathcal{R}}} & \langle s_1, \ldots, s_n \rangle \end{array}$$

Full Inversion

Definition (Full Inversion)

Let $\mathcal{R}$ and $\underline{\mathcal{R}}$ be CCSs over $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ and $\underline{\mathcal{F}} = \underline{\mathcal{C}} \uplus \underline{\mathcal{D}}$, respectively, with $f/n/m \in \mathcal{D}$ and $\underline{f}/m/n \in \underline{\mathcal{D}}$. Then, $\underline{\mathcal{R}}$ is a *full inverse of* $\mathcal{R}$ *w.r.t.* f if for all ground constructor terms $s_1, \ldots, s_n,\ t_1, \ldots, t_m \in T(\mathcal{C} \setminus \{\langle\rangle\}, \emptyset)$,

$$\begin{array}{lll} f(s_1, \ldots, s_n) & \to_{\mathcal{R}} & \langle t_1, \ldots, t_m \rangle \ \Leftrightarrow \\ \underline{f}(t_1, \ldots, t_m) & \to_{\underline{\mathcal{R}}} & \langle s_1, \ldots, s_n \rangle \end{array}$$

Example (Addition)

$$\begin{array}{ll} \mathcal{R}_{add'} = & add(0, y) \to \langle y \rangle \\ & add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle \\ \underline{\mathcal{R}}_{\underline{add'}} = & \underline{add}(y) \to \langle 0, y \rangle \\ & \underline{add}(s(z)) \to \langle s(x), y \rangle \Leftarrow \underline{add}(z) \to \langle x, y \rangle \end{array}$$

Full Inversion

Definition (Full Inversion)

Let $\mathcal{R}$ and $\underline{\mathcal{R}}$ be CCSs over $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ and $\underline{\mathcal{F}} = \underline{\mathcal{C}} \uplus \underline{\mathcal{D}}$, respectively, with $f/n/m \in \mathcal{D}$ and $\underline{f}/m/n \in \underline{\mathcal{D}}$. Then, $\underline{\mathcal{R}}$ is a *full inverse of* $\mathcal{R}$ *w.r.t.* f if for all ground constructor terms $s_1, \ldots, s_n,\ t_1, \ldots, t_m \in T(\mathcal{C} \setminus \{\langle\rangle\}, \emptyset)$,

$$\begin{array}{lll} f(s_1, \ldots, s_n) & \to_{\mathcal{R}} & \langle t_1, \ldots, t_m \rangle \ \Leftrightarrow \\ \underline{f}(t_1, \ldots, t_m) & \to_{\underline{\mathcal{R}}} & \langle s_1, \ldots, s_n \rangle \end{array}$$

Example (Reversible addition)

$$\begin{array}{ll} \mathcal{R}_{add'} = & add'(0, y) \to \langle y, 0 \rangle \\ & add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \\ \underline{\mathcal{R}}_{\underline{add'}} = & \underline{add'}(y, 0) \to \langle 0, y \rangle \\ & \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\begin{aligned}\mathcal{R}_{add'} = \quad & add'(0, y) \to \langle y, 0\rangle \\ & add'(s(x), y) \to \langle s(z), s(x)\rangle \Leftarrow add'(x, y) \to \langle z, x\rangle\end{aligned}$$

Full Inversion of Rules

Example

$$\begin{aligned}\mathcal{R}_{add'} = \quad & add'(0, y) \to \langle y, 0\rangle \\ & add'(s(x), y) \to \langle s(z), s(x)\rangle \Leftarrow add'(x, y) \to \langle z, x\rangle\end{aligned}$$

Example (Inverse reverse addition)

$$\begin{aligned}\underline{\mathcal{R}_{add'}} = \quad & \underline{add'}(y, 0) \to \langle 0, y\rangle \\ & \underline{add'}(s(z), s(x)) \to \langle s(x), y\rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y\rangle\end{aligned}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}_{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}_{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}_{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$add'(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}}_{\underline{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}_{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}}_{\underline{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}_{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow add'(z, x) \to \langle x, y \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}}_{\underline{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$$

Example (Inverse reverse addition)

$$\underline{\mathcal{R}}_{\underline{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Full Inversion of Rules

Example

$$\mathcal{R}_{add'} = \begin{array}{l} add'(0, y) \to \langle y, 0 \rangle \\ add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle \end{array}$$

$$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$$

Example (Inverse reverse addition = Reverse subtraction)

$$\underline{\mathcal{R}}_{\underline{add'}} = \begin{array}{l} \underline{add'}(y, 0) \to \langle 0, y \rangle \\ \underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle \end{array}$$

Algorithm – Full Inversion of Rules

$\mathsf{ruleinv}_{\mathsf{full}}(\mathtt{f}(\mathtt{p}_1,\ldots,\mathtt{p}_n) \to \langle \mathtt{q}_1,\ldots,\mathtt{q}_m\rangle \Leftarrow \mathtt{c})$ =

```
  // full invert and label the rule head
  lhs → rhs := f̲(q1,...,qm) → ⟨p1,...,pn⟩
  // reorder & locally full invert the conditions
  c' := reverse(c)        // reorder
  c'' := localinvc_full(c')   // locally full invert
  // return the full inverted rule
  lhs → rhs ⇐ c''
```

$\mathsf{localinvc}_{\mathsf{full}}(\mathtt{c},\ \mathtt{Var})$ = case c of

```
  // if no conditions, return the empty conjunction
  ϵ  =>  ϵ
  // else full invert the left-most condition
  f(p1,...,pn) → ⟨q1,...,qm⟩ ∧ Restc  =>
    // locally full invert and label the condition
  lhs → rhs := f̲(q1,...,qm) → ⟨p1,...,pn⟩
    // return the inverted conditions
    lhs → rhs ∧ localinvc_full(Restc)
```

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Definition

Given a CCS $\mathcal{R}$, a defined function symbol $f/n/m \in \mathcal{D}$, the full inverter yields the CCS

$$\underline{\mathcal{R}}_{\underline{f}} = \mathsf{inv}_{\mathsf{full}}(\{\underline{f}\}, \emptyset)_{\mathcal{R}}.$$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$)

$add'(0, y) \to \langle y, 0 \rangle$
$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$add'(0, y) \to \langle y, 0 \rangle$
$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$add'(0, y) \to \langle y, 0 \rangle$
$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$add'(0, y) \to \langle y, 0 \rangle$
$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$add'(0, y) \to \langle y, 0 \rangle$
$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$add'(0, y) \to \langle y, 0 \rangle$
$add'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow add'(x, y) \to \langle z, x \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$\underline{add'}(y, 0) \to \langle 0, y \rangle$

$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$\underline{add'}(y, 0) \to \langle 0, y \rangle$

$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$

Full Inversion Algorithm

```
invfull(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rulesoriginal := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rulesinverted := { ruleinvfull(ρ) | ρ ∈ f-Rulesoriginal };
    // update the pending and done sets
    NewDep := getdep(f-Rulesinverted) \ Done ; = {add'}
    f-Rulesinverted ∪ invfull((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$\underline{add'}(y, 0) \to \langle 0, y \rangle$

$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ; = {add'}
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} = ?$)

$\underline{add'}(y, 0) \to \langle 0, y \rangle$

$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$

Full Inversion Algorithm

```
inv_full(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_full(ρ) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted) \ Done ;
    f-Rules_inverted ∪ inv_full((Pend ∪ NewDep) \ {f}, Done ∪ {f});

getdep(Rules) =
    { root(l_i) | l → r ⇐ c ∈ Rules, l_i → r_i ∈ c };
```

Example ($\mathcal{R}_{add'}$ $\mathsf{inv}_{\mathsf{full}}(\{\underline{add'}\}, \emptyset)_{\mathcal{R}_{\underline{add'}}} =$)

$\underline{add'}(y, 0) \to \langle 0, y \rangle$
$\underline{add'}(s(z), s(x)) \to \langle s(x), y \rangle \Leftarrow \underline{add'}(z, x) \to \langle x, y \rangle$

Challenge: Full Invert Divide

Full Invert Divide

Use the Full Inversion Algorithm to manually invert the divide program:

$$\text{divide}(ys) \rightarrow \langle [], ys \rangle$$
$$\text{divide}(:(x, zs)) \rightarrow \langle :(x, xs), ys \rangle \Leftarrow \text{divide}(zs) \rightarrow \langle xs, ys \rangle$$

Challenge: Full Invert Divide

Full Invert Divide

Use the Full Inversion Algorithm to manually invert the divide program:

$$\text{divide}(ys) \rightarrow \langle [], ys \rangle$$
$$\text{divide}(:(x, zs)) \rightarrow \langle :(x, xs), ys \rangle \Leftarrow \text{divide}(zs) \rightarrow \langle xs, ys \rangle$$

The full inversion of divide

$$\underline{\text{divide}}([], ys) \rightarrow \langle ys \rangle$$
$$\underline{\text{divide}}(:(x, xs), ys) \rightarrow \langle :(x, zs) \rangle \Leftarrow \underline{\text{divide}}(xs, ys) \rightarrow \langle zs \rangle$$

Full Inversion –Tool Demonstration

Code-along Challenge

Open your browser at `http://topps.diku.dk/pirc/inversion-tool/` and be ready to code-along.

Example

```
(VAR x xs y ys zs)
(RULES
 divide(ys) -> <[], ys>
 divide(:(x, zs)) -> <:(x, xs), ys> <= divide(zs) -> <xs, ys>
)
```

Inversion and Extra Variables

Definition (Extra variables)

A rule $l \to r \Leftarrow c$ has *extra variables*[a], if a variable occurs on the right-hand side r but neither in its left-hand side l nor in its conditions c, i.e.,

$$\mathcal{V}ar(r) \setminus (\mathcal{V}ar(l) \cup \mathcal{V}ar(c)).$$

In case a rule has no extra variables, we call it *extra-variable-free* (*EV-free*).

[a] We follow the terminology of [NSS05] — these are not to be confused with "extra variables" [Ohl02], i.e., $(\mathcal{V}ar(r) \cup \mathcal{V}ar(c)) \setminus \mathcal{V}ar(l)$.

Example (First projection)

```
(VAR x y test)
(RULES
   fst(x, y) -> <x>
 )
```

```
(VAR x y test)
(RULES
   fst{}{1}(x) -> <x, y>
 )
```

Extra Variables

- There are two interchangeable source for nondeterminism [AH06]:

Extra Variables

- There are two interchangeable source for nondeterminism [AH06]:
  - overlapping rules

Extra Variables

- There are two interchangeable source for nondeterminism [AH06]:
  - overlapping rules
  - extra variables

Extra Variables

- There are two interchangeable source for nondeterminism [AH06]:
  - overlapping rules
  - extra variables
- The ground constructor rewrite relation ensures that the reductions are ground.

Extra Variables

- There are two interchangeable source for nondeterminism [AH06]:
  - overlapping rules
  - extra variables
- The ground constructor rewrite relation ensures that the reductions are ground.
- Extra variables may cause infinitely branching.

Extra Variables

- There are two interchangeable source for nondeterminism [AH06]:
  - overlapping rules
  - extra variables
- The ground constructor rewrite relation ensures that the reductions are ground.
- Extra variables may cause infinitely branching.
- Instead, a more general evaluation mechanism *narrowing* is applicable; in narrowing, matching is replaced by unification, see, *e.g.*, [BN98] for further details and [Han94; Ant10] for a survey.

Permutation

Example (Permutations of a list)

$$\mathsf{perm}([]) \to \langle[]\rangle$$
$$\mathsf{perm}(x) \to \langle{:}(y,z)\rangle \Leftarrow \mathsf{del}(x) \to \langle y,u\rangle \ \wedge \mathsf{perm}(u) \to \langle z\rangle$$
$$\mathsf{del}({:}(x,y)) \to \langle x,y\rangle$$
$$\mathsf{del}({:}(x,y)) \to \langle z,{:}(x,u)\rangle \Leftarrow \mathsf{del}(y) \to \langle z,u\rangle$$

Example (The full inversion, another list permutation program)

$$\underline{\mathsf{perm}}([]) \to \langle[]\rangle$$
$$\underline{\mathsf{perm}}({:}(y,z)) \to \langle x\rangle \Leftarrow \underline{\mathsf{perm}}(z) \to \langle u\rangle \ \wedge \underline{\mathsf{del}}(y,u) \to \langle x\rangle$$
$$\underline{\mathsf{del}}(x,y) \to \langle{:}(x,y)\rangle$$
$$\underline{\mathsf{del}}(z,{:}(x,u)) \to \langle{:}(x,y)\rangle \Leftarrow \underline{\mathsf{del}}(z,u) \to \langle y\rangle$$

Janus Inverter

Janus-Inverter

$$\texttt{inv(proc(name,progr))} \rightarrow \langle \texttt{proc(u,v)} \rangle \Leftarrow \texttt{invName(name)} \rightarrow \langle \texttt{u} \rangle \wedge \texttt{inv(progr)} \rightarrow \langle \texttt{v} \rangle$$

$$\texttt{inv(+=(x,y))} \rightarrow \langle \texttt{-=(x,y)} \rangle$$
$$\texttt{inv(-=(x,y))} \rightarrow \langle \texttt{+=(x,y)} \rangle$$
$$\texttt{inv(<=>(x,y))} \rightarrow \langle \texttt{<=>(x,y)} \rangle$$

$$\texttt{inv(if(}x_1\texttt{,}y_1\texttt{,}y_2\texttt{,}x_2\texttt{))} \rightarrow \langle \texttt{if(}x_2\texttt{,}z_1\texttt{,}z_2\texttt{,}x_1\texttt{)} \rangle \Leftarrow \texttt{inv(}y_1\texttt{)} \rightarrow \langle z_1 \rangle \wedge \texttt{inv(}y_2\texttt{)} \rightarrow \langle z_2 \rangle$$
$$\texttt{inv(loop(}x_1\texttt{,}y_1\texttt{,}y_2\texttt{,}x_2\texttt{))} \rightarrow \langle \texttt{loop(}x_2\texttt{,}z_1\texttt{,}v_2\texttt{,}x_1\texttt{)} \rangle \Leftarrow \texttt{inv(}y_1\texttt{)} \rightarrow \langle z_1 \rangle \wedge \texttt{inv(}y_2\texttt{)} \rightarrow \langle z_2 \rangle$$

$$\texttt{inv(call(name))} \rightarrow \langle \texttt{call(u)} \rangle \Leftarrow \texttt{invName(name)} \rightarrow \langle \texttt{u} \rangle$$
$$\texttt{inv(uncall(name))} \rightarrow \langle \texttt{uncall(u)} \rangle \Leftarrow \texttt{invName(name)} \rightarrow \langle \texttt{u} \rangle$$

$$\texttt{inv(sequence(x,y))} \rightarrow \langle \texttt{sequence(u,v)} \rangle \Leftarrow \texttt{inv(y)} \rightarrow \langle \texttt{u} \rangle \wedge \texttt{inv(x)} \rightarrow \langle \texttt{v} \rangle$$
$$\texttt{inv(skip)} \rightarrow \langle \texttt{skip} \rangle$$

Fully inverted Janus-inverter

$$\underline{\texttt{inv}}\texttt{(proc(u,v))} \rightarrow \langle \texttt{proc(name,progr)} \rangle \Leftarrow \underline{\texttt{invName}}\texttt{(u)} \rightarrow \langle \texttt{name} \rangle \wedge \underline{\texttt{inv}}\texttt{(v)} \rightarrow \langle \texttt{progr} \rangle$$

$$\underline{\texttt{inv}}\texttt{(-=(x,y))} \rightarrow \langle \texttt{+=(x,y)} \rangle$$
$$\underline{\texttt{inv}}\texttt{(+=(x,y))} \rightarrow \langle \texttt{-=(x,y)} \rangle$$
$$\underline{\texttt{inv}}\texttt{(<=>(x,y))} \rightarrow \langle \texttt{<=>(x,y)} \rangle$$

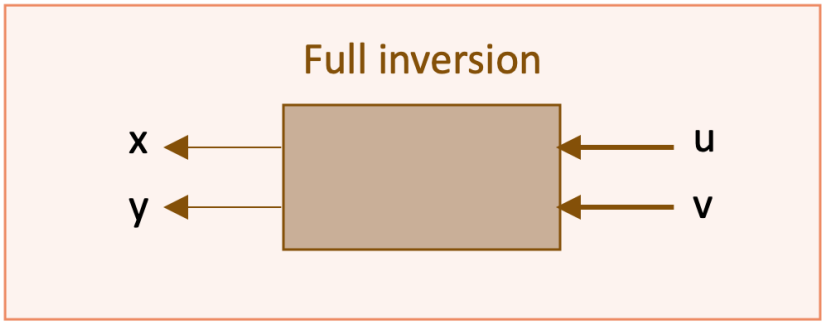
$$\underline{\texttt{inv}}\texttt{(if(}x_2\texttt{,}z_1\texttt{,}z_2\texttt{,}x_1\texttt{))} \rightarrow \langle \texttt{if(}x_1\texttt{,}y_1\texttt{,}y_2\texttt{,}x_2\texttt{)} \rangle \Leftarrow \underline{\texttt{inv}}\texttt{(}z_1\texttt{)} \rightarrow \langle y_1 \rangle \wedge \underline{\texttt{inv}}\texttt{(}z_2\texttt{)} \rightarrow \langle y_2 \rangle$$
$$\underline{\texttt{inv}}\texttt{(loop(}x_2\texttt{,}z_1\texttt{,}z_2\texttt{,}x_1\texttt{))} \rightarrow \langle \texttt{loop(}x_1\texttt{,}y_1\texttt{,}y_2\texttt{,}x_2\texttt{)} \rangle \Leftarrow \underline{\texttt{inv}}\texttt{(}z_1\texttt{)} \rightarrow \langle y_1 \rangle \wedge \underline{\texttt{inv}}\texttt{(}z_2\texttt{)} \rightarrow \langle y_2 \rangle$$

$$\underline{\texttt{inv}}\texttt{(call(u))} \rightarrow \langle \texttt{call(name)} \rangle \Leftarrow \underline{\texttt{invName}}\texttt{(u)} \rightarrow \langle \texttt{name} \rangle$$
$$\underline{\texttt{inv}}\texttt{(uncall(u))} \rightarrow \langle \texttt{uncall(name)} \rangle \Leftarrow \underline{\texttt{invName}}\texttt{(u)} \rightarrow \langle \texttt{name} \rangle$$

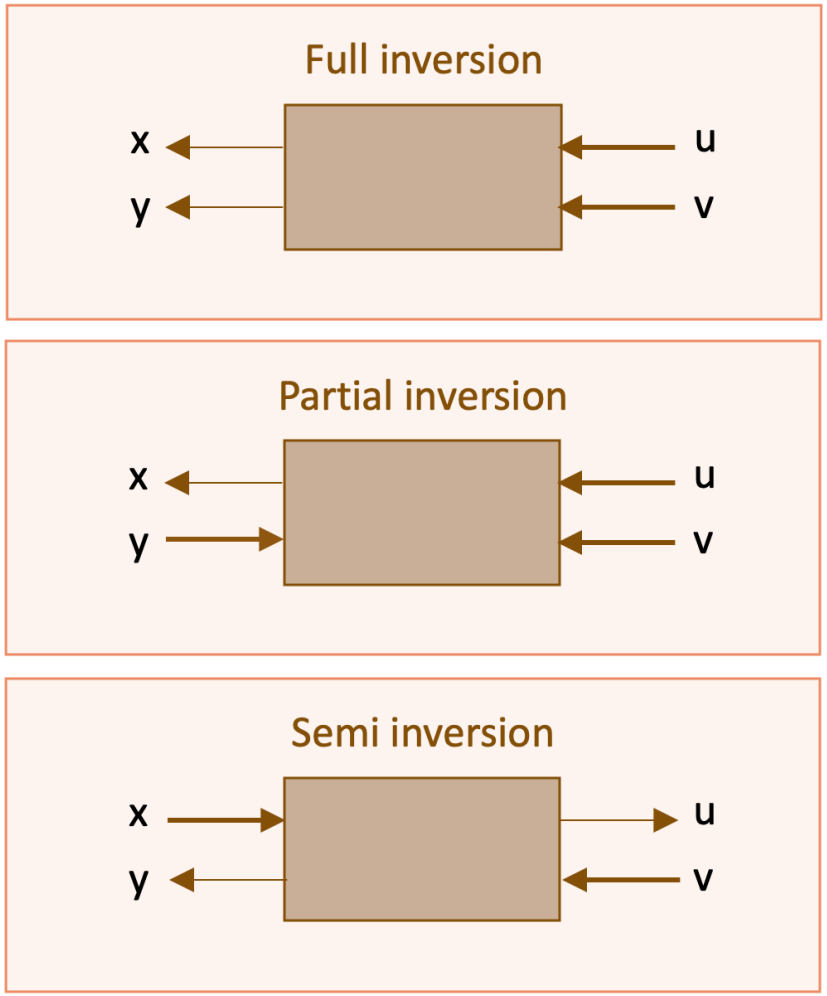
$$\underline{\texttt{inv}}\texttt{(sequence(u,v))} \rightarrow \langle \texttt{sequence(x,y)} \rangle \Leftarrow \underline{\texttt{inv}}\texttt{(u)} \rightarrow \langle y \rangle \wedge \underline{\texttt{inv}}\texttt{(v)} \rightarrow \langle x \rangle$$
$$\underline{\texttt{inv}}\texttt{(skip)} \rightarrow \langle \texttt{skip} \rangle$$

Full Inversion, Partial Inversion, and Semi Inversion

Full Inversion, Partial Inversion, and Semi Inversion



Full Inversion, Partial Inversion, and Semi Inversion



Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$

Which input-output relation does $\underline{add}_{\{1\}\{1\}}$ specify?

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$

Which input-output relation does $\underline{add}_{\{1\}\{1\}}$ specify? answer: *subtraction.*

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$

Which input-output relation does $\underline{add}_{\{1\}\{1\}}$ specify? answer: *subtraction.*

Other examples are:

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \rightarrow \langle y \rangle$
$add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \rightarrow \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \rightarrow \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \rightarrow \langle y \rangle$

Which input-output relation does $\underline{add}_{\{1\}\{1\}}$ specify? answer: *subtraction.*

Other examples are:

- multiplication and division, e.g.,
  $\mathtt{mul}(fac_1, fac_2) = prod$ $\qquad$ $\mathtt{div}(prod, fac_1) = fac_2$

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$

Which input-output relation does $\underline{add}_{\{1\}\{1\}}$ specify? answer: *subtraction.*

Other examples are:

- multiplication and division, e.g.,
  $\mathtt{mul}(fac_1, fac_2) = prod$ $\quad$ $\mathtt{div}(prod, fac_1) = fac_2$
- symmetric encoding/decoding, e.g.,
  $\mathtt{enc}(text_{clear}, key) = text_{enc}$ $\quad$ $\mathtt{dec}(text_{enc}, key) = text_{clear}$

Motivation for Partial and Semi Inversion

- better systems when using partial/semi inversion
- require partial/semi inversion

Partial inversion of addition

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$

Which input-output relation does $\underline{add}_{\{1\}\{1\}}$ specify? answer: *subtraction.*

Other examples are:

- multiplication and division, e.g.,
  $\mathtt{mul}(fac_1, fac_2) = prod$ $\quad$ $\mathtt{div}(prod, fac_1) = fac_2$
- symmetric encoding/decoding, e.g.,
  $\mathtt{enc}(text_{clear}, key) = text_{enc}$ $\quad$ $\mathtt{dec}(text_{enc}, key) = text_{clear}$
- discrete physical simulations, e.g., free fall

Semi-inversion (Partial Inversion)

Definition (Semi-inverse)

Let $\mathcal{R}$ and $\underline{\mathcal{R}}$ be CCSs over $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$, respectively, with $f/n/m, \underline{f}_{IO}/\underline{n}/\underline{m} \in \mathcal{D}$, where $I = \{i_1, \ldots, i_a\}$ and $O = \{o_1, \ldots, o_b\}$ are index sets such that $\underline{n} = a + b$ and $\underline{m} = m + n - \underline{n}$. Then, $\underline{\mathcal{R}}$ is a *semi-inverse of* $\mathcal{R}$ *w.r.t.* f, I, *and* O if for all ground constructor terms $s_1, \ldots, s_n,\ t_1, \ldots, t_m \in T(\mathcal{C} \backslash \{\langle\rangle\}, \emptyset)$,

$$\begin{array}{rcl} f(s_1, \ldots, s_n) & \to_{\mathcal{R}} & \langle t_1, \ldots, t_m \rangle \ \Leftrightarrow \\ \underline{f}_{IO}(s_{i_1}, \ldots, s_{i_a}, t_{o_1}, \ldots, t_{o_b}) & \to_{\underline{\mathcal{R}}} & \langle s_{i_{a+1}}, \ldots, s_{i_n}, t_{o_{b+1}}, \ldots, t_{o_m} \rangle \end{array}$$

where $\{i_1, \ldots, i_a\} \uplus \{i_{a+1}, \ldots, i_n\} = \{1, \ldots, n\}$ and $\{o_1, \ldots, o_b\} \uplus \{o_{b+1}, \ldots, o_m\} = \{1, \ldots, m\}$.

A Semi-inversion is:
a *Full Inversion* if $O = \{t_1, \ldots, t_m\}$ and $I = \emptyset$, and
a *Partial Inversion* if $O = \{t_1, \ldots, t_m\}$.

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$

$mul \quad (s(x), y) \to \langle s(z) \rangle \Leftarrow add \quad (y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add \quad (y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$

$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add \quad (y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$

$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add \quad (y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul \quad (x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$mul_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow add_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge mul_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Step 1: Label all function symbols – Index propagation $I = \{2\}, O = \{1\}$
$mul_{\{2\}\{1\}}(s(x), y) \to \langle s(z) \rangle \Leftarrow add_{\{1\}\{1\}}(y, w) \to \langle z \rangle \wedge mul_{\{2\}\{1\}}(x, y) \to \langle w \rangle$

Step 2: Local Inversion – according to labels
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Example (Part. inverse multiplication)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Semi-inversion of Rules

$\mathsf{ruleinv}_{\mathsf{semi}}(\mathtt{f}(\mathtt{p}_1,\ldots,\mathtt{p}_n) \to \langle \mathtt{q}_1,\ldots,\mathtt{q}_m\rangle \Leftarrow \mathtt{c}, \{\mathtt{i}_1,\ldots,\mathtt{i}_a\},\{\mathtt{o}_1,\ldots,\mathtt{o}_b\}) =$

// semi-invert the rule head and label the function symbol

$\{\mathtt{i}_{a+1},\ldots,\mathtt{i}_n\} := \{1,\ldots,n\} \setminus \{\mathtt{i}_1,\ldots,\mathtt{i}_a\}$

$\{\mathtt{o}_{b+1},\ldots,\mathtt{o}_m\} := \{1,\ldots,m\} \setminus \{\mathtt{o}_1,\ldots,\mathtt{o}_b\}$

$\mathtt{l} \to \mathtt{r} := \underline{\mathtt{f}}_{\{\mathtt{i}_1,\ldots,\mathtt{i}_a\}\{\mathtt{o}_1,\ldots,\mathtt{o}_b\}}(\mathtt{p}_{\mathtt{i}_1},\ldots,\mathtt{p}_{\mathtt{i}_a},\mathtt{q}_{\mathtt{o}_1},\ldots,\mathtt{q}_{\mathtt{o}_b}) \to \langle \mathtt{p}_{\mathtt{i}_{a+1}},\ldots,\mathtt{p}_{\mathtt{i}_n},\mathtt{q}_{\mathtt{o}_{b+1}},\ldots,\mathtt{q}_{\mathtt{o}_m}\rangle$

// locally invert the conditions after reordering

c' := $\mathsf{heuristic}$(c, $\mathcal{V}ar$(l)) // reorder conditions of rule

c'' := $\mathsf{localinvc}_{\mathsf{semi}}$(c',$\mathcal{V}ar$(l)) // reorder terms in conditions

// return the inverted rule

$\mathtt{l} \to \mathtt{r} \Leftarrow$ c''

$\mathsf{localinvc}_{\mathsf{semi}}$(c, Var) = case c of

// if no condition, then return the empty condition

ϵ => ϵ

// else invert the left-most condition

$\mathtt{f}(\mathtt{p}_1,\ldots,\mathtt{p}_n) \to \langle \mathtt{q}_1,\ldots,\mathtt{q}_m\rangle \wedge \mathtt{Restc}$ =>

// build the index sets

$\{\mathtt{i}_1, \ldots, \mathtt{i}_a\} := \{\mathtt{i} \mid \mathtt{i} \in \{1,\ldots,n\}, \mathcal{V}ar(\mathtt{p}_\mathtt{i}) \subseteq \mathtt{Var}\}$

$\{\mathtt{i}_{a+1},\ldots, \mathtt{i}_n\} := \{1,\ldots,n\} \setminus \{\mathtt{i}_1,\ldots,\mathtt{i}_a\}$

$\{\mathtt{o}_1, \ldots, \mathtt{o}_b\} := \{\mathtt{o} \mid \mathtt{o} \in \{1,\ldots,m\}, \mathcal{V}ar(\mathtt{q}_\mathtt{o}) \subseteq \mathtt{Var}\}$

$\{\mathtt{o}_{b+1},\ldots, \mathtt{o}_m\} := \{1,\ldots,m\} \setminus \{\mathtt{o}_1,\ldots,\mathtt{o}_b\}$

// locally invert and label the left-most condition

$\mathtt{l} \to \mathtt{r} := \underline{\mathtt{f}}_{\{\mathtt{i}_1,\ldots,\mathtt{i}_a\}\{\mathtt{o}_1,\ldots,\mathtt{o}_b\}}(\mathtt{p}_{\mathtt{i}_1},\ldots,\mathtt{p}_{\mathtt{i}_a},\mathtt{q}_{\mathtt{o}_1},\ldots,\mathtt{q}_{\mathtt{o}_b}) \to \langle \mathtt{p}_{\mathtt{i}_{a+1}},\ldots,\mathtt{p}_{\mathtt{i}_n},\mathtt{q}_{\mathtt{o}_{b+1}},\ldots,\mathtt{q}_{\mathtt{o}_m}\rangle$

// return the inverted conditions

$\mathtt{l} \to \mathtt{r} \wedge \mathsf{localinvc}_{\mathsf{semi}}(\mathtt{Restc}, \mathtt{Var} \cup \mathcal{V}ar(\mathtt{r}))$

Semi-inversion Algorithm

$\mathsf{inv}_{\mathsf{semi}}$(Pend, Done) =

if Pend = $\emptyset$ then $\emptyset$ else

// choose a pending task for semi-inversion

$\underline{\mathtt{f}}_{\mathtt{IO}} \in$ Pend;

// full invert all rules of f

f-Rules$_{\mathtt{original}}$:= { ρ | $\rho : l \to r \Leftarrow c \in \mathcal{R}$, $root(l) = \mathtt{f}$ };

f-Rules$_{\mathtt{inverted}}$:= { $\mathsf{ruleinv}_{\mathsf{semi}}(\rho$, I, O) | $\rho \in$ f-Rules$_{\mathtt{original}}$ };

// update the pending and done sets

NewDep := getdep(f-Rules$_{\mathtt{inverted}}$)\Done;

f-Rules$_{\mathtt{inverted}}$ $\cup$ $\mathsf{inv}_{\mathsf{semi}}$((Pend $\cup$ NewDep) $\setminus$ $\{\underline{\mathtt{f}}_{\mathtt{IO}}\}$, Done $\cup$ $\{\underline{\mathtt{f}}_{\mathtt{IO}}\}$);

Semi-inversion Algorithm

$\mathsf{inv}_{\mathsf{semi}}$(Pend, Done) =

if Pend = $\emptyset$ then $\emptyset$ else

// choose a pending task for semi-inversion

$\underline{\mathtt{f}}_{\mathtt{IO}} \in$ Pend;

// full invert all rules of f

f-Rules$_{\mathtt{original}}$:= { ρ | $\rho : l \to r \Leftarrow c \in \mathcal{R}$, $root(l) = \mathtt{f}$ };

f-Rules$_{\mathtt{inverted}}$:= { $\mathsf{ruleinv}_{\mathsf{semi}}(\rho$, I, O) | $\rho \in$ f-Rules$_{\mathtt{original}}$ };

// update the pending and done sets

NewDep := getdep(f-Rules$_{\mathtt{inverted}}$)\Done;

f-Rules$_{\mathtt{inverted}}$ $\cup$ $\mathsf{inv}_{\mathsf{semi}}$((Pend $\cup$ NewDep) $\setminus$ $\{\underline{\mathtt{f}}_{\mathtt{IO}}\}$, Done $\cup$ $\{\underline{\mathtt{f}}_{\mathtt{IO}}\}$);

Semi-inversion Algorithm

```
inv_semi(Pend, Done) =
    if Pend = ∅ then ∅ else
    // choose a pending task for semi-inversion
    f_IO ∈ Pend;
    // full invert all rules of f
    f-Rules_original := { ρ | ρ : l → r ⇐ c ∈ R, root(l) = f };
    f-Rules_inverted := { ruleinv_semi(ρ, I, O) | ρ ∈ f-Rules_original };
    // update the pending and done sets
    NewDep := getdep(f-Rules_inverted)\Done;
    f-Rules_inverted ∪ inv_semi((Pend ∪ NewDep) \ {f_IO}, Done ∪ {f_IO});
```

Definition

Given a CCS $\mathcal{R}$, a defined function symbol $f/n/m \in \mathcal{D}$ and io-sets $\mathtt{I} \subseteq \{1, \ldots, n\}$ and $\mathtt{O} \subseteq \{1, \ldots, m\}$, the semi-inverter yields the CCS

$$\underline{\mathcal{R}}_{\underline{f}} = \mathsf{inv}_{\mathsf{semi}}(\{\underline{f}_{\mathtt{IO}}\}, \emptyset)_{\mathcal{R}}.$$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \wedge mul(x, y) \to \langle w \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow mul(x, y) \to \langle w \rangle \wedge add(y, w) \to \langle z \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow mul(x, y) \to \langle w \rangle \wedge add(y, w) \to \langle z \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Reordering of conditions

What was the reordering used in full inversion?

Heuristic

Example (Multiplication of unary numbers $\mathcal{R}_{mul}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow mul(x, y) \to \langle w \rangle \wedge add(y, w) \to \langle z \rangle$

Example (Division $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$\underline{add}_{\{1\}\{1\}}(0, y) \to \langle y \rangle$
$\underline{add}_{\{1\}\{1\}}(s(x), s(z)) \to \langle y \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(x, z) \to \langle y \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, 0) \to \langle 0 \rangle$
$\underline{mul}_{\{2\}\{1\}}(y, s(z)) \to \langle s(x) \rangle \Leftarrow \underline{add}_{\{1\}\{1\}}(y, z) \to \langle w \rangle \wedge \underline{mul}_{\{2\}\{1\}}(y, w) \to \langle x \rangle$

Reordering of conditions

What was the reordering used in full inversion?
Can we use that again? why/why not?

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$add(0, y) \rightarrow \langle y \rangle$
$add(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow add(x, y) \rightarrow \langle z \rangle$
$mul(0, y) \rightarrow \langle 0 \rangle$
$mul(s(x), y) \rightarrow \langle s(z) \rangle \Leftarrow mul(x, y) \rightarrow \langle w \rangle \ \wedge \ add(y, w) \rightarrow \langle z \rangle$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow mul(x, y) \to \langle w \rangle \ \wedge \ add(y, w) \to \langle z \rangle$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow mul(x, y) \to \langle w \rangle \ \wedge \ add(y, w) \to \langle z \rangle$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$$add(0, y) \to \langle y \rangle$$
$$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$$
$$mul(0, y) \to \langle 0 \rangle$$
$$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow \underbrace{mul(x, y) \to \langle w \rangle}_{1/3} \wedge \underbrace{add(y, w) \to \langle z \rangle}_{2/3}$$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$$add(0, y) \to \langle y \rangle$$
$$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$$
$$mul(0, y) \to \langle 0 \rangle$$
$$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow \underbrace{mul(x, y) \to \langle w \rangle}_{1/3} \wedge \underbrace{add(y, w) \to \langle z \rangle}_{2/3}$$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \ \wedge \ mul(x, y) \to \langle w \rangle$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \ \wedge \ mul(x, y) \to \langle w \rangle$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$add(0, y) \to \langle y \rangle$
$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$
$mul(0, y) \to \langle 0 \rangle$
$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \ \wedge \ mul(x, y) \to \langle w \rangle$

Heuristic

Semi-inversion solution: a greedy heuristic.

Recursively choose the condition with highest "known terms"-%.

Example (Heuristic – to create $\underline{\mathcal{R}}_{\underline{mul}_{\{2\}\{1\}}}$)

$$add(0, y) \to \langle y \rangle$$
$$add(s(x), y) \to \langle s(z) \rangle \Leftarrow add(x, y) \to \langle z \rangle$$
$$mul(0, y) \to \langle 0 \rangle$$
$$mul(s(x), y) \to \langle s(z) \rangle \Leftarrow add(y, w) \to \langle z \rangle \ \wedge \ \underbrace{mul(x, y) \to \langle w \rangle}_{2/3}$$

Heuristic

heuristic(c, KnownVar) = case c of

// if no condition, then return the empty condition

ϵ => ϵ

// else find condition with highest percentage of known parameters

$\mathsf{l}_1 \to \mathsf{r}_1 \wedge \ldots \wedge \mathsf{l}_k \to \mathsf{r}_k$ =>

// determine percentages

$\mathsf{Percent}_1$:= percent($\mathsf{l}_1 \to \mathsf{r}_1$, KnownVar)

... ...

$\mathsf{Percent}_k$:= percent($\mathsf{l}_k \to \mathsf{r}_k$, KnownVar)

(P_i, i) := maxPercent(($\mathsf{Percent}_1$,1),...,($\mathsf{Percent}_k$,k))

// select condition, reorder remaining conditions in updated variable set

$\mathsf{l}_i \to \mathsf{r}_i \wedge$ heuristic(c $\setminus$ ($\mathsf{l}_i \to \mathsf{r}_i$), KnownVar $\cup$ $\mathcal{V}ar(\mathsf{l}_i, \mathsf{r}_i)$)

percent($\mathsf{f}(\mathsf{p}_1, \ldots, \mathsf{p}_\mathtt{n}) \to \langle \mathsf{q}_1, \ldots, \mathsf{q}_\mathtt{m} \rangle$, KnownVar) =

// determine index sets of known parameters

$\mathsf{I} := \{\mathsf{i} \mid \mathsf{i} \in \{1, \ldots, \mathsf{n}\}, \mathcal{V}ar(\mathsf{p}_i) \subseteq \mathsf{KnownVar}\}$

$\mathsf{O} := \{\mathsf{j} \mid \mathsf{j} \in \{1, \ldots, \mathsf{m}\}, \mathcal{V}ar(\mathsf{q}_j) \subseteq \mathsf{KnownVar}\}$

// return percentage of known parameters

$(|\mathtt{I}| + |\mathtt{O}|) / (\mathtt{m} + \mathtt{n})$ // known = $|\mathtt{I}| + |\mathtt{O}|$, total = $\mathtt{m} + \mathtt{n}$

Tool Demonstration

Code-along Challenge

Open your browser at

`http://topps.diku.dk/pirc/inversion-tool/`

and be ready to code-along.

Example (multiplication)

Partially invert the `mul2` using the Semi-inverter with io-set I={2}, O={1}

```
(VAR x y z w)
(RULES
  mul2(0,y) -> <0>
  mul2(s(x),y) -> <z> <= add(y,w) -> <z>,  mul2(x,y) -> <w>

  add(0,y) -> <y>
  add(s(x), y) -> <s(z)> <= add(x,y) -> <z>
)
```

Tool Demonstration

Tool Challenge

Use the Semi-inverter to invert `mul` and `mul2` with io-set $I=\{2\}, O=\{1\}$. Compare their inversions.

```
(VAR x y z w)
(RULES
  mul(0,y) -> <0>
  mul(s(x),y) -> <z> <= mul(x,y) -> <w>, add(y,w) -> <z>

  mul2(0,y) -> <0>
  mul2(s(x),y) -> <z> <= add(y,w) -> <z>,  mul2(x,y) -> <w>

  add(0,y) -> <y>
  add(s(x), y) -> <s(z)> <= add(x,y) -> <z>
)
```

Discrete Physical Simulation of Free Fall

$$v_t = v_{t-1} + g$$
$$h_t = h_{t-1} - v_t + g/2$$
where $g \approx 10\ m/s^2$

Discrete Physical Simulation of Free Fall

Example (Free fall)

$\mathtt{fall}(\mathtt{v}, \mathtt{h}, 0) \to \langle \mathtt{v}, \mathtt{h} \rangle$

$\mathtt{fall}(\mathtt{v}_0, \mathtt{h}_0, \mathtt{s}(\mathtt{t})) \to \langle \mathtt{v}, \mathtt{h} \rangle \Leftarrow \mathtt{add}(\mathtt{v}_0, \mathtt{s}^{10}(0)) \to \langle \mathtt{v}_\mathtt{n} \rangle \wedge \mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \wedge \mathtt{fall}(\mathtt{v}_\mathtt{n}, \mathtt{h}_\mathtt{n}, \mathtt{t}) \to \langle \mathtt{v}, \mathtt{h} \rangle$

$\mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \Leftarrow \mathtt{add}(\mathtt{h}_0, \mathtt{s}^{5}(0)) \to \langle \mathtt{h}_\mathtt{temp} \rangle \wedge \mathtt{sub}(\mathtt{h}_\mathtt{temp}, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle$

$v_t = v_{t-1} + g$
$h_t = h_{t-1} - v_t + g/2$
where $g \approx 10\ m/s^2$

Discrete Physical Simulation of Free Fall

Example (Free fall)

$\mathtt{fall}(\mathtt{v}, \mathtt{h}, 0) \to \langle \mathtt{v}, \mathtt{h} \rangle$

$\mathtt{fall}(\mathtt{v}_0, \mathtt{h}_0, \mathtt{s}(\mathtt{t})) \to \langle \mathtt{v}, \mathtt{h} \rangle \Leftarrow \mathtt{add}(\mathtt{v}_0, \mathtt{s}^{10}(0)) \to \langle \mathtt{v}_\mathtt{n} \rangle \wedge \mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \wedge \mathtt{fall}(\mathtt{v}_\mathtt{n}, \mathtt{h}_\mathtt{n}, \mathtt{t}) \to \langle \mathtt{v}, \mathtt{h} \rangle$

$\mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \Leftarrow \mathtt{add}(\mathtt{h}_0, \mathtt{s}^{5}(0)) \to \langle \mathtt{h}_\mathtt{temp} \rangle \wedge \mathtt{sub}(\mathtt{h}_\mathtt{temp}, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle$

Inversions

Original	$\mathtt{fall}:$	$(\mathtt{v}_0, \mathtt{h}_0, \mathtt{t}) \to (\mathtt{v}, \mathtt{h})$
Full inv.	$\underline{\mathtt{fall}}_{\emptyset\{1,2\}}:$	$(\mathtt{v}, \mathtt{h}) \to (\mathtt{v}_0, \mathtt{h}_0, \mathtt{t})$
Partial inv.	$\underline{\mathtt{fall}}_{\{3\}\{1,2\}}:$	$(\mathtt{t}, \mathtt{v}, \mathtt{h}) \to (\mathtt{v}_0, \mathtt{h}_0)$
Semi-inv.	$\underline{\mathtt{fall}}_{\{1,2\}\{2\}}:$	$(\mathtt{v}_0, \mathtt{h}_0, \mathtt{h}) \to (\mathtt{t}, \mathtt{v})$

$v_t = v_{t-1} + g$
$h_t = h_{t-1} - v_t + g/2$
where $g \approx 10\ m/s^2$

Discrete Physical Simulation of Free Fall

Example (Free fall)

$\mathtt{fall}(\mathtt{v}, \mathtt{h}, 0) \to \langle \mathtt{v}, \mathtt{h} \rangle$

$\mathtt{fall}(\mathtt{v}_0, \mathtt{h}_0, \mathtt{s}(\mathtt{t})) \to \langle \mathtt{v}, \mathtt{h} \rangle \Leftarrow \mathtt{add}(\mathtt{v}_0, \mathtt{s}^{10}(0)) \to \langle \mathtt{v}_\mathtt{n} \rangle \wedge \mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \wedge \mathtt{fall}(\mathtt{v}_\mathtt{n}, \mathtt{h}_\mathtt{n}, \mathtt{t}) \to \langle \mathtt{v}, \mathtt{h} \rangle$

$\mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \Leftarrow \mathtt{add}(\mathtt{h}_0, \mathtt{s}^5(0)) \to \langle \mathtt{h}_\mathtt{temp} \rangle \wedge \mathtt{sub}(\mathtt{h}_\mathtt{temp}, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle$

Inversions

Original	$\mathtt{fall}:$	$(\mathtt{v}_0, \mathtt{h}_0, \mathtt{t}) \to (\mathtt{v}, \mathtt{h})$
Full inv.	$\underline{\mathtt{fall}}_{\emptyset\{1,2\}}:$	$(\mathtt{v}, \mathtt{h}) \to (\mathtt{v}_0, \mathtt{h}_0, \mathtt{t})$
Partial inv.	$\underline{\mathtt{fall}}_{\{3\}\{1,2\}}:$	$(\mathtt{t}, \mathtt{v}, \mathtt{h}) \to (\mathtt{v}_0, \mathtt{h}_0)$
Semi-inv.	$\underline{\mathtt{fall}}_{\{1,2\}\{2\}}:$	$(\mathtt{v}_0, \mathtt{h}_0, \mathtt{h}) \to (\mathtt{t}, \mathtt{v})$

$v_t = v_{t-1} + g$
$h_t = h_{t-1} - v_t + g/2$
where $g \approx 10\ m/s^2$

Example (Semi-inversion)

$\underline{\mathtt{fall}}_{\{1,2\}\{2\}}(\mathtt{v}, \mathtt{h}, \mathtt{h}) \to \langle 0, \mathtt{v} \rangle$

$\underline{\mathtt{fall}}_{\{1,2\}\{2\}}(\mathtt{v}_0, \mathtt{h}_0, \mathtt{h}) \to \langle \mathtt{s}(\mathtt{t}), \mathtt{v} \rangle \Leftarrow$
$\mathtt{add}(\mathtt{v}_0, \mathtt{s}^{10}(0)) \to \langle \mathtt{v}_\mathtt{n} \rangle \wedge \mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \wedge \underline{\mathtt{fall}}_{\{1,2\}\{2\}}(\mathtt{v}_\mathtt{n}, \mathtt{h}_\mathtt{n}, \mathtt{h}) \to \langle \mathtt{t}, \mathtt{v} \rangle$

$\mathtt{height}(\mathtt{h}_0, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle \Leftarrow \mathtt{add}(\mathtt{h}_0, \mathtt{s}^5(0)) \to \langle \mathtt{h}_\mathtt{temp} \rangle \wedge \mathtt{sub}(\mathtt{h}_\mathtt{temp}, \mathtt{v}_\mathtt{n}) \to \langle \mathtt{h}_\mathtt{n} \rangle$

Reasoning Program Inversion Algorithms

A step towards comparing and reasoning about program inverters

Reasoning Program Inversion Algorithms

A step towards comparing and reasoning about program inverters

- studied existing algorithms: Trivial Inverter [KG20a], Full Inverter [Nis04], Partial Inverter [NSS05] & Semi Inverter [KG20b]

Reasoning Program Inversion Algorithms

A step towards comparing and reasoning about program inverters

- studied existing algorithms: Trivial Inverter [KG20a], Full Inverter [Nis04], Partial Inverter [NSS05] & Semi Inverter [KG20b]
- created an inversion framework capturing these (and more) inverters and shown its correctness [KG20a],

Reasoning Program Inversion Algorithms

A step towards comparing and reasoning about program inverters

- studied existing algorithms: Trivial Inverter [KG20a], Full Inverter [Nis04], Partial Inverter [NSS05] & Semi Inverter [KG20b]
- created an inversion framework capturing these (and more) inverters and shown its correctness [KG20a],
- identified program properties which holds for all full, all partial or for all semi-inverters covered by this framework [KG20a], and

Reasoning Program Inversion Algorithms

A step towards comparing and reasoning about program inverters

- studied existing algorithms: Trivial Inverter [KG20a], Full Inverter [Nis04], Partial Inverter [NSS05] & Semi Inverter [KG20b]
- created an inversion framework capturing these (and more) inverters and shown its correctness [KG20a],
- identified program properties which holds for all full, all partial or for all semi-inverters covered by this framework [KG20a], and
- implemented the algorithm [MGK21] http://topps.diku.dk/pirc/inversion-tool/

Generic Inversion Algorithm

```
L1  inv(Pend, R) =
L2     Done := ∅;    // done tasks
L3     R' := ∅;      // inverted rules
L4     while(Pend ≠ ∅){
L5        // choose a pending task for semi-inversion
L6        (f, I, O) ∈ Pend;
L7        // semi-invert all rules of f in R with index sets I and O
L8        R'' := { ruleinv(ρ, I, O) | ρ ∈ (R|_f) };
L9        // update the inverted rules, pending and done sets
L10       R' := R' ∪ R'';
L11       Done := Done ∪ {(f, I, O)};
L12       Pend := (Pend ∪ getdep(R'')) \ Done;
L13    }
L14    return R';

L15 getdep(Rules) =
L16    return { (f, I, O) | l→r⇐c ∈ Rules, l_i→r_i ∈ c, root(l_i) = f_IO };
```

Generic Inversion Algorithm

```
L1  inv(Pend, R) =
L2     Done := ∅;    // done tasks
L3     R' := ∅;      // inverted rules
L4     while(Pend ≠ ∅){
L5        // choose a pending task for semi-inversion
L6        (f, I, O) ∈ Pend;
L7        // semi-invert all rules of f in R with index sets I and O
L8        R'' := { ruleinv(ρ, I, O) | ρ ∈ (R|_f) };
L9        // update the inverted rules, pending and done sets
L10       R' := R' ∪ R'';
L11       Done := Done ∪ {(f, I, O)};
L12       Pend := (Pend ∪ getdep(R'')) \ Done;
L13    }
L14    return R';

L15 getdep(Rules) =
L16    return { (f, I, O) | l→r⇐c ∈ Rules, l_i→r_i ∈ c, root(l_i) = f_IO };
```

Generic Inversion Algorithm

Invoke the inverter by
$\mathsf{inv}(\{(f,I,O)\}, R)$,

e.g., full inversion:
$\mathsf{inv}(\{(\mathsf{mul},\emptyset,\{1\})\}, R)$

```
L1  inv(Pend, R) =
L2     Done := ∅;    // done tasks
L3     R' := ∅;      // inverted rules
L4     while(Pend ≠ ∅){
L5        // choose a pending task for semi-inversion
L6        (f, I, O) ∈ Pend;
L7        // semi-invert all rules of f in R with index sets I and O
L8        R'' := { ruleinv(ρ, I, O) | ρ ∈ (R|_f) };
L9        // update the inverted rules, pending and done sets
L10       R' := R' ∪ R'';
L11       Done := Done ∪ {(f, I, O)};
L12       Pend := (Pend ∪ getdep(R'')) \ Done;
L13    }
L14    return R';

L15 getdep(Rules) =
L16    return { (f, I, O) | l→r⇐c ∈ Rules, l_i→r_i ∈ c, root(l_i) = f_IO };
```

Parameterized Rule Inverter & Instantiations

L1 $\mathsf{ruleinv}(f(p_{\bar{n}}) \to \langle q_{\bar{m}} \rangle \Leftarrow c, I, O) =$

L2 $l' \to r' := \underline{f}_{IO}(p_I, q_O) \to \langle p_{\bar{n} \setminus I}, q_{\bar{m} \setminus O} \rangle;$ // semi-invert rule head

L3 $c' \quad := \mathsf{reorder}(c, l' \to r');$ // reorder conditions

L4 $c'' \quad := \mathsf{localinv}(c', l' \to r');$ // invert conditions

L5 $\mathtt{return}(l' \to r' \Leftarrow c'');$ // the inverted rule

Parameterized Rule Inverter & Instantiations

L1 $\mathsf{ruleinv}(f(p_{\bar{n}}) \to \langle q_{\bar{m}} \rangle \Leftarrow c, I, O) =$

L2 $l' \to r' := \underline{f}_{IO}(p_I, q_O) \to \langle p_{\bar{n} \setminus I}, q_{\bar{m} \setminus O} \rangle;$ // semi-invert rule head

L3 $c' \quad := \mathsf{reorder}(c, l' \to r');$ // reorder conditions

L4 $c'' \quad := \mathsf{localinv}(c', l' \to r');$ // invert conditions

L5 $\mathtt{return}(l' \to r' \Leftarrow c'');$ // the inverted rule

Parameterized Rule Inverter & Instantiations

L1 $\mathsf{ruleinv}(f(p_{\bar{n}}) \to \langle q_{\bar{m}} \rangle \Leftarrow c, I, O) =$

L2 $l' \to r' := \underline{f}_{IO}(p_I, q_O) \to \langle p_{\bar{n} \setminus I}, q_{\bar{m} \setminus O} \rangle;$ // semi-invert rule head

L3 $c' \quad := \mathsf{reorder}(c, l' \to r');$ // reorder conditions

L4 $c'' \quad := \mathsf{localinv}(c', l' \to r');$ // invert conditions

L5 $\mathtt{return}(l' \to r' \Leftarrow c'');$ // the inverted rule

Parameterized Rule Inverter & Instantiations

L1 $\text{ruleinv}(f(p_{\bar{n}}) \to \langle q_{\bar{m}} \rangle \Leftarrow c, I, O) =$

L2 $l' \to r' := \underline{f}_{IO}(p_I, q_O) \to \langle p_{\bar{n}\setminus I}, q_{\bar{m}\setminus O} \rangle;$ // semi-invert rule head

L3 $c' \;\; := \text{reorder}(c, l' \to r');$ // reorder conditions

L4 $c'' \;\; := \text{localinv}(c', l' \to r');$ // invert conditions

L5 $\texttt{return}(l' \to r' \Leftarrow c'');$ // the inverted rule

trivial inversion [KG20a]:

$\text{reorder}(c, l \to r) = c;$

$\text{localinv}(c, l \to r) = c;$

pure partial inversion [NSS05]:

$\text{reorder}(c, l \to r) = \text{reverse}(c);$

$\text{localinv}(c, l \to r) = \text{localinv}_{\text{part}}(c, \text{Var}(l'));$

pure full inversion [Nis04]:

$\text{reorder}(c, l \to r) = \text{reverse}(c);$

$\text{localinv}(c, l \to r) = \text{map swap}\, c;$

semi-inversion [KG20b]:

$\text{reorder}(c, l \to r) = \text{heuristic}(c, \text{Var}(l'));$

$\text{localinv}(c, l \to r) = \text{localinv}_{\text{semi}}(c, \text{Var}(l'));$

More than one Algorithm

Challenge: Compare the Partial and Semi Inversion Algorithms

Partially invert the following `mul` with io-set $I = \{2\}$ and $O = \{1\}$ with both the semi-inverter and the partial inverter (check both).

```
(VAR x y z w)
(RULES
  mul2(0,y) -> <0>
  mul2(s(x),y) -> <z> <= add(y,w) -> <z>, mul2(x,y) -> <w>
  add(0,y) -> <y>
  add(s(x), y) -> <s(z)> <= add(x,y) -> <z>
)
```

More than one Algorithm

Challenge: Compare the Partial and Semi Inversion Algorithms

Partially invert the following `mul` with io-set $I = \{2\}$ and $O = \{1\}$ with both the semi-inverter and the partial inverter (check both).

```
(VAR x y z w)
(RULES
  mul2(0,y) -> <0>
  mul2(s(x),y) -> <z> <= add(y,w) -> <z>, mul2(x,y) -> <w>
  add(0,y) -> <y>
  add(s(x), y) -> <s(z)> <= add(x,y) -> <z>
)
```

1. what are the differences between the output systems?

More than one Algorithm

Challenge: Compare the Partial and Semi Inversion Algorithms

Partially invert the following `mul` with io-set $I = \{2\}$ and $O = \{1\}$ with both the semi-inverter and the partial inverter (check both).

```
(VAR x y z w)
(RULES
  mul2(0,y) -> <0>
  mul2(s(x),y) -> <z> <= add(y,w) -> <z>, mul2(x,y) -> <w>
  add(0,y) -> <y>
  add(s(x), y) -> <s(z)> <= add(x,y) -> <z>
)
```

1. what are the differences between the output systems?
2. which do you prefer and why?

More than one Algorithm

Challenge: Compare the Partial and Semi Inversion Algorithms

Partially invert the following `mul` with io-set $I = \{2\}$ and $O = \{1\}$ with both the semi-inverter and the partial inverter (check both).

```
(VAR x y z w)
(RULES
  mul2(0,y) -> <0>
  mul2(s(x),y) -> <z> <= add(y,w) -> <z>, mul2(x,y) -> <w>
  add(0,y) -> <y>
  add(s(x), y) -> <s(z)> <= add(x,y) -> <z>
)
```

1. what are the differences between the output systems?
2. which do you prefer and why?
3. what do you think causes the differences?

Well-behaved Rule Inverters & Correctness [KG20a]

Definition (Well-behaved rule inverters)

A *well-behaved rule inverter* ruleinv

- locally inverts the left and the right side of a rule according to the given io-sets,
- reorders the condition conjunction, and
- locally inverts each of the conditions in that conjunction.

Well-behaved Rule Inverters & Correctness [KG20a]

Definition (Well-behaved rule inverters)

A *well-behaved rule inverter* ruleinv

- locally inverts the left and the right side of a rule according to the given io-sets,
- reorders the condition conjunction, and
- locally inverts each of the conditions in that conjunction.

Theorem (correctness of well-behaved inverters)

Let $\mathcal{R}$ be a CCS and inv be a well-behaved inverter defining $\underline{\mathcal{R}}_{all}$ and $\underline{\mathcal{R}}_{\underline{f}_{IO}}$ for a defined symbol $f/n/m \in \mathcal{D}_{\mathcal{R}}$ and io-sets I and O. Then, $\underline{\mathcal{R}}_{\underline{f}_{IO}}$ is an inverse of $\mathcal{R}$ with respect to f, I, and O.

Inversion in different language paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Inversion in different language paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Full Inversions

output-orthogonal $\Leftrightarrow$ orthogonal

non-erasing $\Leftrightarrow$ EV-free

Inversion in different language paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Full Inversions

output-orthogonal $\Leftrightarrow$ orthogonal

non-erasing $\Leftrightarrow$ EV-free

Inversion in different language paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Full Inversions

output-orthogonal $\Leftrightarrow$ orthogonal

non-erasing $\Leftrightarrow$ EV-free

Inversion in different language paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Full Inversions

output-orthogonal $\Leftrightarrow$ orthogonal

non-erasing $\Leftrightarrow$ EV-free

Inversion in different language paradigms

Declarative

Functional

Reversible

- output-orthogonal:
  - non-output-overlapping
  - right-linear
- weakly non-erasing
- non-erasing

- orthogonal:
  - non-overlapping
  - left-linear
- left-to-right deterministic
- EV-free

- all CCSs

Full Inversions

output-orthogonal $\Leftrightarrow$ orthogonal

non-erasing $\Leftrightarrow$ EV-free

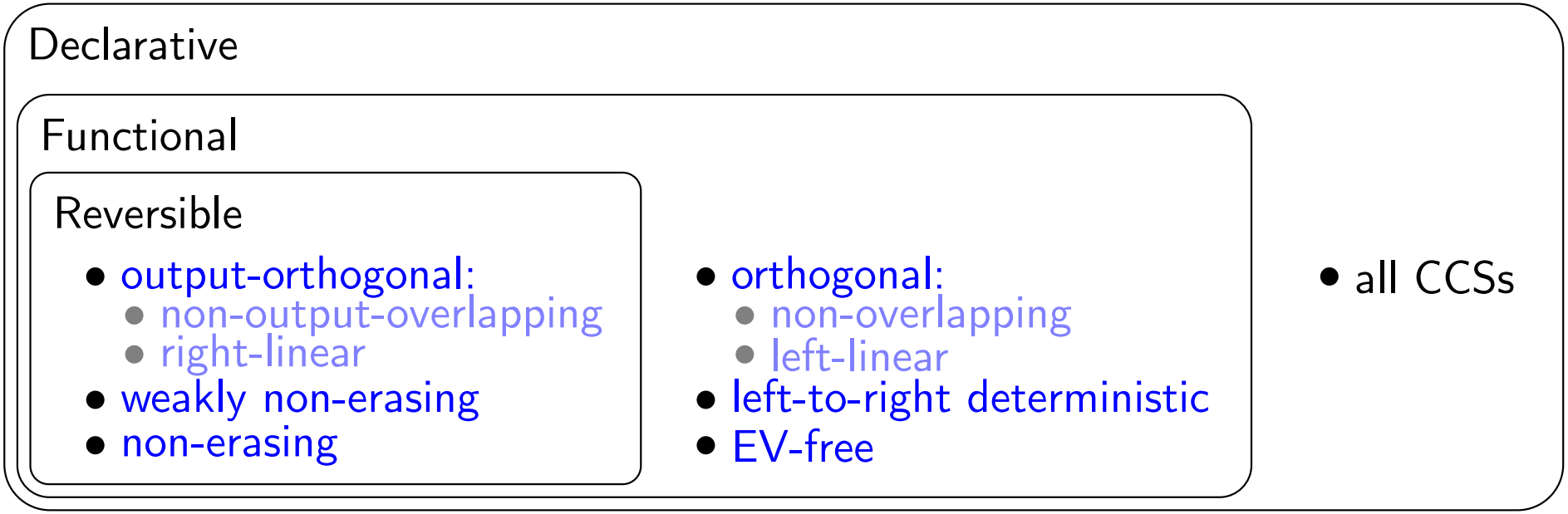
Partial Inversions

non-output-overlapping $\Rightarrow$ nonoverlapping

right-linear $\Leftarrow$ left-linear

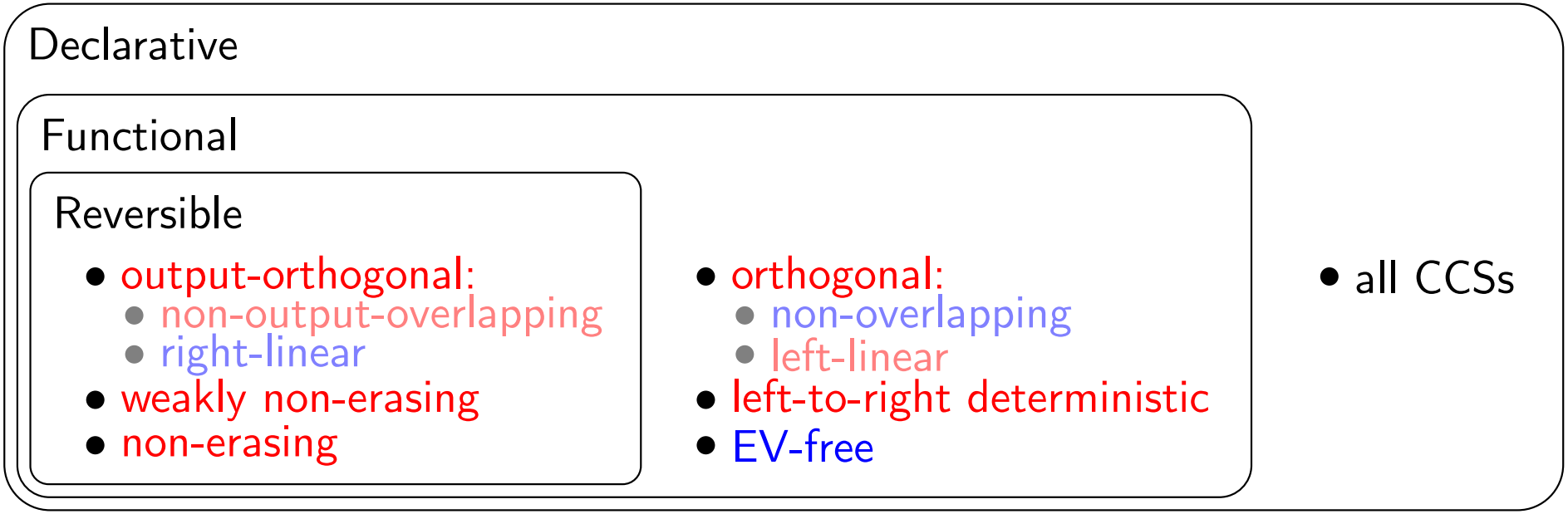
non-erasing $\Rightarrow$ EV-free

Inversion in different language paradigms



Full Inversions			**Partial Inversions**		
output-orthogonal	$\Leftrightarrow$	orthogonal	non-output-overlapping	$\Rightarrow$	nonoverlapping
			right-linear	$\Leftarrow$	left-linear
non-erasing	$\Leftrightarrow$	EV-free	non-erasing	$\Rightarrow$	EV-free

Inversion in different language paradigms



Full Inversions			**Partial Inversions**		
output-orthogonal	$\Leftrightarrow$	orthogonal	non-output-overlapping	$\Rightarrow$	nonoverlapping
			right-linear	$\Leftarrow$	left-linear
non-erasing	$\Leftrightarrow$	EV-free	non-erasing	$\Rightarrow$	EV-free

Properties of Inverted Reversible Systems

Challenge

Use the tool to explore the properties of the different inversions of reversible addition.

$$\mathsf{add}'(0, y) \to \langle y, 0 \rangle$$
$$\mathsf{add}'(s(x), y) \to \langle s(z), s(x) \rangle \Leftarrow \mathsf{add}'(x, y) \to \langle z, x \rangle$$

Summary

- Introduction to the concept of full, partial and semi-inversion, i.e., given one program, we access a handfull of programs.

The tool's source code: `https://github.com/pirc-src/inversion-tool`

Summary

- Introduction to the concept of full, partial and semi-inversion, i.e., given one program, we access a handfull of programs.
- Introduction to a mathematical formalism CCS which models logic, functional and logic and functional programs

The tool's source code: `https://github.com/pirc-src/inversion-tool`

Summary

- Introduction to the concept of full, partial and semi-inversion, i.e., given one program, we access a handfull of programs.
- Introduction to a mathematical formalism CCS which models logic, functional and logic and functional programs
- The core algorithm for full inversion including step-by-step challenges, code-along and applications.

The tool's source code: `https://github.com/pirc-src/inversion-tool`

Summary

- Introduction to the concept of full, partial and semi-inversion, i.e., given one program, we access a handfull of programs.
- Introduction to a mathematical formalism CCS which models logic, functional and logic and functional programs
- The core algorithm for full inversion including step-by-step challenges, code-along and applications.
- The algorithm for semi-inversion including challenges, code-along and applications.

The tool's source code: `https://github.com/pirc-src/inversion-tool`

Summary

- Introduction to the concept of full, partial and semi-inversion, i.e., given one program, we access a handfull of programs.
- Introduction to a mathematical formalism CCS which models logic, functional and logic and functional programs
- The core algorithm for full inversion including step-by-step challenges, code-along and applications.
- The algorithm for semi-inversion including challenges, code-along and applications.
- Seen how the generic inversion algorithm can be used for comparing existing algorithms and discussing the properties of the resulting systems.

The tool's source code: `https://github.com/pirc-src/inversion-tool`

References I

[AH06] Sergio Antoy and Michael Hanus. "Overlapping Rules and Logic Variables in Functional Logic Programs". In: *ICLP* 4079 (2006), pp. 87–101. DOI: 10.1007/11799573_9.

[Ant10] Sergio Antoy. "Programming with narrowing: A tutorial". In: *Journal of Symbolic Computation* 45.5 (2010). (version: June 2017, update), pp. 501–522. ISSN: 07477171. DOI: 10.1016/j.jsc.2010.01.006.

[AV07] Jesús Manuel Almendros-Jiménez and Germán Vidal. "Automatic partial inversion of inductively sequential functions". In: *Implementation and Application of Functional Languages. Proceedings*. Ed. by Zoltán Horváth et al. LNCS 4449. Springer, 2007, pp. 253–270. DOI: 10.1007/978-3-540-74130-5_15.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. United Kingdom: Cambridge University Press, 1998.

[GK03] Robert Glück and Masahiko Kawabe. "A program inverter for a functional language with equality and constructors". In: *Programming Languages and Systems. Proceedings*. Ed. by Atsushi Ohori. LNCS 2895. Springer, 2003, pp. 246–264. DOI: 10.1007/978-3-540-40018-9_17.

[Han94] Michael Hanus. "The integration of functions into logic programming: From theory to practice". In: *The Journal of Logic Programming* 19-20.Suppl. 1 (1994), pp. 583–628. ISSN: 07431066. DOI: 10.1016/0743-1066(94)90034-5.

References II

[KG20a] Maja H. Kirkeby and Robert Glück. "Inversion framework: reasoning about inversion by conditional term rewriting systems". In: *Principles and Practice of Declarative Programming. Proceedings*. ACM, 2020, Article 9. DOI: 10.1145/3414080.3414089.

[KG20b] Maja H. Kirkeby and Robert Glück. "Semi-inversion of conditional constructor term rewriting systems". In: *Logic-based Program Synthesis and Transformation. Proceedings*. Ed. by Maurizio Gabbrielli. LNCS 12042. Springer, 2020, pp. 243–259. DOI: 10.1007/978-3-030-45260-5_15.

[MGK21] Maria Bendix Mikkelsen, Robert Glück, and Maja H. Kirkeby. "An Inversion Tool for Conditional Term Rewriting Systems". In: *ETAPS Workshop on Verification and Program Transformation (VPT 2021)*. 2021. URL: http://refal.botik.ru/vpt/vpt2021/VPT2021_paper_2.pdf.

[Nis04] Naoki Nishida. "Transformational Approach to Inverse Computation in Term Rewriting". PhD thesis. Japan: Graduate School of Engineering, Nagoya University, 2004.

[NSS05] Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe. "Partial inversion of constructor term rewriting systems". In: *Rewriting Techniques and Applications. Proceedings*. Ed. by Jürgen Giesl. LNCS 3467. Springer, 2005, pp. 264–278. DOI: 10.1007/978-3-540-32033-3_20.

References III

[NV11] Naoki Nishida and Germán Vidal. "Program inversion for tail recursive functions". In: *Rewriting Techniques and Applications. Proceedings*. Ed. by Manfred Schmidt-Schauß. Vol. 10. LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011, pp. 283–298. ISBN: 978-3-939897-30-9. DOI: 10.4230/LIPIcs.RTA.2011.283.

[Ohl02] Enno Ohlebusch. *Advanced Topics in Term Rewriting*. New York: Springer, 2002.

[Rom88] Alexander Y. Romanenko. "The generation of inverse functions in Refal". In: *Partial Evaluation and Mixed Computation*. Ed. by Dines Bjørner, Andrei P. Ershov, and Neil D. Jones. North-Holland, 1988, pp. 427–444.

[Rom91] Alexander Y. Romanenko. "Inversion and metacomputation". In: *Partial Evaluation and Semantics-Based Program Manipulation. Proceedings*. ACM, 1991, pp. 12–22. DOI: 10.1145/115865.115868.