Formalizing the implementation of Streaming NESL

Dandan Xue

October 16, 2017

Abstract

Streaming NESL (SNESL) is a first-order functional, nested data-parallel language, employing a streaming execution model and integrating with a cost model that can predict both time and space complexity. The experimentation has demonstrated good performance of SNESL's implementation and positive empirical evidence of the validity of the code model. In this thesis, we first present non-trivial extension to SNESL's target language, SVCODE, which enables SNESL to support recursion at the same time preserve the cost. Then the formalization of the semantics of this low-level streaming language is given. Finally, we present the proof of the correctness of SNESL's implementation model.

Contents

Chapter 1

Formalization

In this chapter, we will present the formal proof of the correctness of the language SNESL_0 , a subset of SNESL_1 . First its definition and semantics will be given. Then SVCODE_0 , the target language of SNESL_0 , is defined, and proof of its determinism is given. The value representation and translation from SNESL_0 to SVCODE_0 are also formalized. Finally we show the proof of the main correctness theorem of this language.

1.1 SNESL_0

The language SNESL_0 we will proof in this chapter is a subset of SNESL_1 , with mainly the following simplifications:

- only one primitive type **int**
- no pairs
- selected built-in functions
- restricted comprehension is removed
- no user-defined functions

1.1.1 Syntax

(1) The types of SNESL_0 are:

$$\tau ::= \textbf{int} \mid \{\tau_1\}$$

(2) The synatx of SNESL_0 values :

$$n \in \mathbb{Z}$$

$$v ::= n \mid \{v_1, ..., v_k\}$$

(3) The syntax of SNESL_0 expressions and the built-in function are shown in Figure 1.1. Note that constants can be generated by calling the built-in function $\textbf{const}_n()$ for decreasing expression cases. Also, the arguments of built-in functions as well as the bound sequence in general comprehension are variables instead of expressions; we can simply convert them into their general forms by adding let-bindings of these variables.

$$
\begin{array}{lr}
e ::= x & \text{(variable)} \\
\quad \mid \textbf{let } x = e_1 \textbf{ in } e_2 & \text{(let-binding)} \\
\quad \mid \phi(x_1, ..., x_k) & \text{(built-in function call)} \\
\quad \mid \{e : x \textbf{ in } y \textbf{ using } x_1, ..., x_j\} & \text{(general comprehension)}
\end{array}
$$

$$\phi ::= \textbf{const}_n \mid \textbf{iota} \mid \textbf{plus}$$

Figure 1.1: SNESL_0 expressions and built-in functions

1.1.2 Typing rules

The typing environment Γ is a mapping from variables to types:

$$\Gamma = [x_1 \mapsto \tau_1, ..., x_i \mapsto \tau_i]$$

- Expression typing rules:

  Judgment $\boxed{\Gamma \vdash e : \tau}$

  $$\frac{}{\Gamma \vdash x : \tau}\,(\Gamma(x) = \tau) \qquad \frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma[x \mapsto \tau_1] \vdash e_2 : \tau}{\Gamma \vdash \textbf{let } x = e_1 \textbf{ in } e_2 : \tau}$$

  $$\frac{\phi : (\tau_1, ..., \tau_k) \to \tau}{\Gamma \vdash \phi(x_1, ..., x_k) : \tau}\,((\Gamma(x_i) = \tau_i)_{i=1}^k)$$

  $$\frac{[x \mapsto \tau_1, (x_i \mapsto \textbf{int})_{i=1}^j] \vdash e : \tau}{\Gamma \vdash \{e : x \textbf{ in } y \textbf{ using } x_1, ..., x_j\} : \{\tau\}}\,(\Gamma(y) = \{\tau_1\}, (\Gamma(x_i) = \textbf{int})_{i=1}^j)$$

- Built-in functions typing rules:

  Judgment $\boxed{\phi : (\tau_1, ..., \tau_k) \to \tau}$

  $$\frac{}{\textbf{const}_n : () \to \textbf{int}} \qquad \frac{}{\textbf{iota} : (\textbf{int}) \to \{\textbf{int}\}} \qquad \frac{}{\textbf{plus} : (\textbf{int}, \textbf{int}) \to \textbf{int}}$$

- Value typing rules:

  Judgment $\boxed{v : \tau}$

  $$\frac{}{n : \textbf{int}} \qquad \frac{(v_i : \tau)_{i=1}^k}{\{v_1, ..., v_k\} : \{\tau\}}$$

1.1.3 Semantics

The evaluation environment ρ is a mapping from variables to values:

$$\rho = [x_1 \mapsto v_1, ..., x_i \mapsto v_i]$$

- Expression evaluation rules:

  Judgment $\boxed{\rho \ \vdash \ e \downarrow v}$

  $$\frac{}{\rho \ \vdash \ x \downarrow v} \, (\rho(x) = v) \qquad \frac{\rho \ \vdash \ e_1 \downarrow v_1 \qquad \rho[x \mapsto v_1] \ \vdash \ e_2 \downarrow v}{\rho \ \vdash \ \textbf{let} \ e_1 = x \ \textbf{in} \ e_2 \downarrow v}$$

  $$\frac{\phi(v_1, ..., v_k) \downarrow v}{\rho \ \vdash \ \phi(x_1, ..., x_k) \downarrow v} \, ((\rho(x_i) = v_i)_{i=1}^{k})$$

  $$\frac{([x \mapsto v_i, (x_i \mapsto n_i)_{i=1}^{j}] \ \vdash \ e \downarrow v_i')_{i=1}^{k}}{\rho \ \vdash \ \{e : x \ \textbf{in} \ y \ \textbf{using} \ x_1, ..., x_j\} \downarrow \{v_1', ..., v_k'\}} \, (\rho(y) = \{v_1, ..., v_k\}, (\rho(x_i) = n_i)_{i=1}^{j})$$

- Built-in function evaluation rules:

  Judgment $\boxed{\phi(v_1, ..., v_k) \downarrow v}$

  $$\frac{}{\textbf{const}_n() \downarrow n} \qquad \frac{}{\textbf{iota}(n) \downarrow \{0, 1, ..., n-1\}} \, (n \geq 0)$$

  $$\frac{}{\textbf{plus}(n_1, n_2) \downarrow n_3} \, (n_3 = n_1 + n_2)$$

1.2 SVCODE$_0$

The target language of SNESL$_0$ is also a subset of SVCODE presented in the last chapter. We will call it SVCODE$_0$.

1.2.1 Syntax

In this minimal language, a primitive stream $\vec{a}$ can be a vector of booleans, integers or units, as the following grammar shows:

$$b \in \mathbb{B} = \{\mathtt{T}, \mathtt{F}\}$$
$$a ::= n \mid b \mid ()$$
$$\vec{b} = \langle b_1, ..., b_i \rangle$$
$$\vec{c} = \langle (), ..., () \rangle$$
$$\vec{a} = \langle a_1, ..., a_i \rangle$$

The syntax of SVCODE$_0$ is given in Figure 1.2.

$$
\begin{array}{lll}
p & ::= & \epsilon \\
& & \mid s := \psi(s_1, ..., s_k) \\
& & \mid S_{out} := \texttt{WithCtrl}(s, S_{in}, p_1) \\
& & \mid p_1; p_2 \\
\\
s & ::= & 0 \mid 1 \ldots \in \mathbf{SId} = \mathbb{N} \qquad \text{(stream ids)} \\
\\
\psi & ::= & \texttt{Const}_{\texttt{a}} \mid \texttt{ToFlags} \mid \texttt{Usum} \mid \texttt{MapTwo}_{\oplus} \mid \texttt{ScanPlus}_{n_0} \mid \texttt{Distr} \qquad \text{(Xducers)} \\
\\
S & ::= & \{s_1, ..., s_i\} \in \mathbb{S} \qquad \text{(a set of stream ids)}
\end{array}
$$

Figure 1.2: Abstract syntax of SVCODE_0

The function `dv` returns the set of defined variables of a given SVCODE_0 program.

$$
\begin{array}{ll}
\texttt{dv}(\epsilon) & = \emptyset \\
\texttt{dv}(s := \psi(s_1, ..., s_k)) & = \{s\} \\
\texttt{dv}(S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1)) & = S_{out} \\
\texttt{dv}(p_1; p_2) & = \texttt{dv}(p_1) \cup \texttt{dv}(p_2)
\end{array}
$$

Correspondingly, `fv` returns the free variables set.

$$
\begin{array}{ll}
\texttt{fv}(\epsilon) & = \emptyset \\
\texttt{fv}(s := \psi(s_1, ..., s_i)) & = \{s_1, ..., s_k\} \\
\texttt{fv}(S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1)) & = \{s_c\} \cup S_{in} \\
\texttt{fv}(p_1; p_2) & = \texttt{fv}(p_1) \cup (\texttt{fv}(p_2) - \texttt{dv}(p_1))
\end{array}
$$

An immediate property of this language is that the defined variables of a *well-formed* SVCODE_0 program are always fresh. In other words, there is no overlapping between the free variables and the newly generated ones.

Definition 1.1 (**Well-formedness**). *p is a well-formed SVCODE program, written as $\Vdash p$, if one of the following rules applies:*

Judgment $\boxed{\Vdash p}$

$$
\frac{}{\Vdash \epsilon} \qquad \frac{}{\Vdash s := \psi(s_1, ..., s_k)} \, (s \notin \{s_1, ..., s_k\})
$$

$$
\frac{\Vdash p_1}{\Vdash S_{out} := \texttt{WithCtrl}(s, S_{in}, p_1)} \begin{pmatrix} \texttt{fv}(p_1) \subseteq S_{in} \\ S_{out} \subseteq \texttt{dv}(p_1) \\ (\{s\} \cup S_{in}) \cap \texttt{dv}(p_1) = \emptyset \end{pmatrix}
$$

$$\frac{\Vdash p_1 \qquad \Vdash p_2}{\Vdash (p_1; p_2)} \left((\mathtt{fv}(p_1) \cup \mathtt{dv}(p_1)) \cap \mathtt{dv}(p_2) = \emptyset \right)$$

Lemma 1.2 (**Freshness**). *If p is well-formed, then* $\mathtt{fv}(p) \cap \mathtt{dv}(p) = \emptyset$.

The proof is straightward by induction on the derivation of $\Vdash p$.

1.2.2 Instruction semantics

Before showing the semantics, we first introduce some notations and operations about streams for convenience.

Notation 1.3. *Let $\langle a_0 | \vec{a} \rangle$ denote a non-empty stream $\langle a_0, a_1, ..., a_i \rangle$ for some $\vec{a} = \langle a_1, ..., a_i \rangle$.*

Notation 1.4 (**Stream concatenation**). $\langle a_1, ..., a_i \rangle ++ \langle a'_1, ..., a'_j \rangle = \langle a_1, ..., a_i, a'_1, ..., a'_j \rangle$

The operational semantics of SVCODE_0 is given in Figure 1.3. The runtime environment or store σ is a mapping from stream variables to vectors:

$$\sigma = [s_1 \mapsto \vec{a}_1, ..., s_i \mapsto \vec{a}_i]$$

The control stream $\vec{c}$, which is basically a vector of units representing an unary number, indicates the *parallel degree* of the computation. It is worth noting that only in the rule P-Wc-Nonemp the control stream has a chance to get changed.

Judgment $\boxed{\langle p, \sigma \rangle \Downarrow^{\vec{c}} \sigma'}$

P-Empty: $\dfrac{}{\langle \epsilon, \sigma \rangle \Downarrow^{\vec{c}} \sigma}$

P-Xducer : $\dfrac{\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\vec{c}} \vec{a}}{\langle s := \psi(s_1, ..., s_k), \sigma \rangle \Downarrow^{\vec{c}} \sigma[s \mapsto \vec{a}]} \left((\sigma(s_i) = \vec{a}_i)_{i=1}^k \right)$

P-Wc-Emp: $\dfrac{}{\langle S_{out} := \mathtt{WithCtrl}(s_c, S_{in}, p_1), \sigma \rangle \Downarrow^{\vec{c}} \sigma[(s_i \mapsto \langle\rangle)_{i=1}^l]} \begin{pmatrix} \forall s \in \{s_c\} \cup S_{in}.\sigma(s) = \langle\rangle \\ S_{out} = \{s_1, ..., s_l\} \end{pmatrix}$

P-Wc-Nonemp : $\dfrac{\langle p_1, \sigma \rangle \Downarrow^{\vec{c}_1} \sigma''}{\langle S_{out} := \mathtt{WithCtrl}(s_c, S_{in}, p_1), \sigma \rangle \Downarrow^{\vec{c}} \sigma[(s_i \mapsto \sigma''(s_i))_{i=1}^l]} \begin{pmatrix} \sigma(s_c) = \vec{c}_1 \neq \langle\rangle \\ S_{out} = \{s_1, ..., s_l\} \end{pmatrix}$

P-Seq : $\dfrac{\langle p_1, \sigma \rangle \Downarrow^{\vec{c}} \sigma'' \qquad \langle p_2, \sigma'' \rangle \Downarrow^{\vec{c}} \sigma'}{\langle p_1; p_2, \sigma \rangle \Downarrow^{\vec{c}} \sigma'}$

Figure 1.3: SVCODE_0 semantics

The rule P-Empty is trivial, empty program doing nothing on the store.

The rule P-Xducer adds the store a new stream binding where the bound vector is generated by a specific Xducer. The detailed semantics of Xducers are defined in the next subsection.

The rules P-Wc-Emp and P-Wc-Nonemp together show two possibilities for interpreting a `WithCtrl` instruction:

- if the new control stream s_c and the streams in S_{in}, which includes the free variables of p_1, are all empty, then just bind empty vectors to the stream ids in S_{out}, which are part of the defined streams of p_1.
- otherwise execute the code of p_1 as usual under the new control stream, ending in the store σ''; then copy the bindings of S_{out} from σ'' to the initial store.

1.2.3 Xducer semantics

The semantics of Xducers are abstracted into two levels: the *general* level and the *block* level. The general level summarizes the comman property that all Xducers share, and the block level describes the specific behavior of each Xducer.

Figure 1.4 shows the semantics at the general level.

Judgment $\boxed{\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\vec{c}} \vec{a}}$

$$\text{P-X-Loop}: \frac{\psi(\vec{a}_{11}, ..., \vec{a}_{k1}) \downarrow \vec{a}_{01} \qquad \psi(\vec{a}_{12}, ..., \vec{a}_{k2}) \Downarrow^{\vec{c}_0} \vec{a}_{02}}{\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\langle()|\vec{c}_0\rangle} \vec{a}_0} ((\vec{a}_{i1} +\!\!+ \vec{a}_{i2} = \vec{a}_i)_{i=0}^{k})$$

$$\text{P-X-Termi}: \frac{}{\psi(\langle\rangle_1, ..., \langle\rangle_k) \Downarrow^{\langle\rangle} \langle\rangle}\ ^{1}$$

Figure 1.4: General semantics of SVCODE_0 Xducers

There are only two rules for the general semantics. They together say that the output stream is computed in a "loop" fashion, where the iteration uses specific block semantics of the Xducer and the number of iteration is the unary number that the control stream represents, i.e., the length of the control stream. In the parallel setting, we prefer to call this iteration a *block*. Recall the control stream is a representation of the parallel degree of the computation, then a block consumes exact one degree. It it worth noting that all these blocks are data-independent, which means they can be executed in parallel. So the control stream indeed carries the theoretical maximum number of processors we need to execute the computation most efficiently, if the computation within the block cannot be parallelized further.

After abstracting the general semantics, the remaining work of formalizing the specific semantics of Xducers within a block becomes relatively clear and simple. The block semantics are defined in Figure 1.5.

[1] For notational convenience, in this thesis we add subscripts to a sequence of constants, such as $\langle\rangle, \mathtt{F}, 1$, to denote the total number of these constants.

Judgment $\boxed{\psi(\vec{a}_1, ..., \vec{a}_k) \downarrow \vec{a}}$

P-X-Const: $\dfrac{}{\texttt{Const}_\texttt{a}() \downarrow \langle a \rangle}$ P-X-ToFlags: $\dfrac{}{\texttt{ToFlags}(\langle n \rangle) \downarrow \langle \texttt{F}_1, ..., \texttt{F}_n, \texttt{T} \rangle}$

P-X-MapTwo : $\dfrac{}{\texttt{MapTwo}_\oplus(\langle n_1 \rangle, \langle n_2 \rangle) \downarrow \langle n_3 \rangle}\ (n_3 = n_1 \oplus n_2)$

P-X-UsumF : $\dfrac{\texttt{Usum}(\vec{b}) \downarrow \vec{a}}{\texttt{Usum}(\langle \texttt{F} | \vec{b} \rangle) \downarrow \langle () | \vec{a} \rangle}$ P-X-UsumT: $\dfrac{}{\texttt{Usum}(\langle \texttt{T} \rangle) \downarrow \langle \rangle}$

P-X-ScanF : $\dfrac{\texttt{ScanPlus}_{n_0+n}(\vec{b}, \vec{a}) \downarrow \vec{a}'}{\texttt{ScanPlus}_{n_0}(\langle \texttt{F} | \vec{b} \rangle, \langle n | \vec{a} \rangle) \downarrow \langle n_0 | \vec{a}' \rangle}$

P-ScanT: $\dfrac{}{\texttt{ScanPlus}_{n_0}(\langle \texttt{T} \rangle, \langle \rangle) \downarrow \langle \rangle}$

P-X-DistrF : $\dfrac{\texttt{Distr}(\vec{b}, \langle n \rangle) \downarrow \vec{a}}{\texttt{Distr}(\langle \texttt{F} | \vec{b} \rangle, \langle n \rangle) \downarrow \langle n | \vec{a} \rangle}$ P-X-DistrT: $\dfrac{}{\texttt{Distr}(\langle \texttt{T} \rangle, \langle n \rangle) \downarrow \langle \rangle}$

Figure 1.5: Block semantics of Xducers

As have discussed before, we consider a block as the minimum computing unit assigned to a single processor. This is reasonable for Xducers such as $\texttt{Const}_\texttt{a}$ and $\texttt{MapTwo}_\oplus$, because they are already sequential at the block level.

However, some other Xducers, such as `Usum`, can be parallelized further inside a block. As we extend the language with more Xducers, we could find that computations on unary numbers within blocks are common, which is mainly due to the value representation strategy we use, but also more difficult to be regularized. For the scope of this thesis, the block semantics we have shown are already relatively clear and simple enough to reason about, and the unary level parallelism can be investigated in future work.

1.2.4 SVCODE$_0$ determinism

We now present how we prove a well-formed SVCODE$_0$ program is deterministic.

Definition 1.5 (Stream prefix). *$\vec{a}$ is a prefix of $\vec{a}'$, written $\vec{a} \sqsubseteq \vec{a}'$, if one of the following rules applys:*

Judgment $\boxed{\vec{a} \sqsubseteq \vec{a}'}$

I-Emp: $\dfrac{}{\langle\rangle \sqsubseteq \vec{a}'}$ I-Nonemp: $\dfrac{\vec{a} \sqsubseteq \vec{a}'}{\langle a_0 \mid \vec{a} \rangle \sqsubseteq \langle a_0 \mid \vec{a}' \rangle}$

Lemma 1.6. *If $\vec{a}_1 +\!\!+ \vec{a}_2 = \vec{a}$, then $\vec{a}_1 \sqsubseteq \vec{a}$.*

Proof. The proof is straightforward by induction on $\vec{a}_1$: case $\vec{a}_1 = \langle\rangle$ and case $\vec{a}_1 = \langle a_0 \mid \vec{a}'_1\rangle$ for some $\vec{a}'_1$. ■

The following lemma says that a Xducer always knows how many elements it should consume and produce when it reads one element from the control stream, i.e., for one block, and these numbers are fixed in any block for the same Xducer.

Lemma 1.7 (**Blocks are self-delimiting**). *If*

(i) $(\vec{a}'_i \sqsubseteq \vec{a}_i$ *by some derivation* $\mathcal{I}_i)_{i=1}^k$ *and* $\psi(\vec{a}'_1, ..., \vec{a}'_k) \downarrow \vec{a}'$ *by some* $\mathcal{P}$,

(ii) $(\vec{a}''_i \sqsubseteq \vec{a}_i$ *by some derivation* $\mathcal{I}'_i)_{i=1}^k$ *and* $\psi(\vec{a}''_1, ..., \vec{a}''_k) \downarrow \vec{a}''$ *by some* $\mathcal{P}'$.

then

(i) $(\vec{a}'_i = \vec{a}''_i)_{i=1}^k$

(ii) $\vec{a}' = \vec{a}''$.

Proof. The proof can be done by induction on $\mathcal{P}$. We show three cases P–X-ToFlags, P–X-ScanT and P–X-ScanF here; the others are analogous.

- Case $\mathcal{P}$ uses P–X-ToFlags.
  Then
  $$\mathcal{P} = \frac{}{\texttt{ToFlags}(\langle n_1\rangle) \downarrow \langle \texttt{F}_1, ..., \texttt{F}_{n_1}, \texttt{T}\rangle}$$
  and
  $$\mathcal{P}' = \frac{}{\texttt{ToFlags}(\langle n_2\rangle) \downarrow \langle \texttt{F}_1, ..., \texttt{F}_{n_2}, \texttt{T}\rangle}$$
  so $k = 1, \vec{a}'_1 = \langle n_1\rangle$, $\vec{a}' = \langle \texttt{F}_1, ..., \texttt{F}_{n_1}, \texttt{T}\rangle$, and $\vec{a}''_1 = \langle n_2\rangle$, $\vec{a}'' = \langle \texttt{F}_1, ..., \texttt{F}_{n_2}, \texttt{T}\rangle$.
  Since both $\vec{a}'_1$ and $\vec{a}''_1$ are nonempty, $\mathcal{I}_1$ and $\mathcal{I}'_1$ must both use the rule I-Nonemp, which implies $n_1 = n_2$. Then it is clear that $\vec{a}'_1 = \vec{a}''_1$ and $\vec{a}' = \vec{a}''$, as required.

- Case $\mathcal{P}$ uses P–X-ScanT.
  Then
  $$\mathcal{P} = \frac{}{\texttt{ScanPlus}_{n_0}(\langle \texttt{T}\rangle, \langle\rangle) \downarrow \langle\rangle}$$
  so k=2, $\vec{a}'_1 = \langle \texttt{T}\rangle$. Since $\vec{a}'_1$ is nonempty, then $\mathcal{I}_1$ must use I-Nonemp, which implies the first element of $\vec{a}_1$ is T.
  There are two possibilities for $\mathcal{P}'$:

  – Subcase $\mathcal{P}'$ uses P–X-ScanF.
    This subcase is impossible, because it requires $\vec{a}_1$ starts with a F, which is contradictory to what we already know.

  – Subcase $\mathcal{P}'$ uses P–X-ScanT.
    Then
    $$\mathcal{P}' = \frac{}{\texttt{ScanPlus}_{n_0}(\langle \texttt{T}\rangle, \langle\rangle) \downarrow \langle\rangle}$$
    So $\vec{a}''_1 = \langle \texttt{T}\rangle = \vec{a}'_1$, $\vec{a}''_2 = \langle\rangle = \vec{a}'_2$, and $\vec{a}'' = \langle\rangle = \vec{a}''$, as required.

- Case $\mathcal{P}$ uses P–X-ScanF.
  Then
  $$\mathcal{P} = \frac{\begin{array}{c}\mathcal{P}_0\\ \texttt{ScanPlus}_{n_0+n}(\vec{a}'_{10}, \vec{a}'_{20}) \downarrow \vec{a}'_0\end{array}}{\texttt{ScanPlus}_{n_0}(\langle \texttt{F}|\vec{a}'_{10}\rangle, \langle n|\vec{a}'_{20}\rangle) \downarrow \langle n_0|\vec{a}'_0\rangle}$$
  So k=2, $\vec{a}'_1 = \langle \texttt{F}|\vec{a}'_{10}\rangle$, and $\vec{a}'_2 = \langle n|\vec{a}'_{20}\rangle$. $\mathcal{I}_1$ must use I-Nonemp, which implies the first element of $\vec{a}_1$ is F. So $\vec{a}_1 = \langle \texttt{F}|\vec{a}_{10}\rangle$ for some $\vec{a}_{10}$. By the rule I-Nonemp, we have
  $$\frac{\vec{a}'_{10} \sqsubseteq \vec{a}_{10}}{\langle \texttt{F}|\vec{a}'_{10}\rangle \sqsubseteq \langle \texttt{F}|\vec{a}_{10}\rangle}$$
  Similarly, we can assume $\vec{a}_2 = \langle n|\vec{a}_{20}\rangle$, and we must have $\vec{a}'_{20} \sqsubseteq \vec{a}_{20}$. Thus,
  $$(\vec{a}'_{i0} \sqsubseteq \vec{a}_{i0})_{i=1}^2 \tag{1.1}$$
  There are two possibilities for $\mathcal{P}'$:
  - Subcase $\mathcal{P}'$ uses P–X-ScanT.
    This subcase is impossible, because $\vec{a}_1$ does not start with a T.
  - Subcase $\mathcal{P}'$ uses P–X-ScanF.
    Then
    $$\mathcal{P}' = \frac{\begin{array}{c}\mathcal{P}'_0\\ \texttt{ScanPlus}_{n_0+n}(\vec{a}''_{10}, \vec{a}''_{20}) \downarrow \vec{a}''_0\end{array}}{\texttt{ScanPlus}_{n_0}(\langle \texttt{F}|\vec{a}''_{10}\rangle, \langle n|\vec{a}''_{20}\rangle) \downarrow \langle n_0|\vec{a}''_0\rangle}$$
    so $\vec{a}''_1 = \langle \texttt{F}|\vec{a}''_{10}\rangle, \vec{a}''_2 = \langle n|\vec{a}''_{20}\rangle$, $\vec{a}'' = \langle n_0|\vec{a}''_0\rangle$, and it is easy to show
    $$(\vec{a}''_{i0} \sqsubseteq \vec{a}_{i0})_{i=1}^2 \tag{1.2}$$
    Here we first prove the following inner lemma:
    if $(\vec{a}'_{i0} \sqsubseteq \vec{a}_{i0})_{i=1}^2$ and $\texttt{ScanPlus}_{n_0}(\vec{a}'_{10}, \vec{a}'_{20}) \downarrow \vec{a}'$ by some derivation $\mathcal{P}_0$, $(\vec{a}''_{i0} \sqsubseteq \vec{a}_{i0})_{i=1}^2$ and $\texttt{ScanPlus}_{n_0}(\vec{a}''_{10}, \vec{a}''_{20}) \downarrow \vec{a}''$, then $\vec{a}'_{10} = \vec{a}''_{10}, \vec{a}'_{20} = \vec{a}''_{20}$, and $\vec{a}' = \vec{a}''$.
    The proof is by induction on $\mathcal{P}_0$. There are two subcases: for the case $\mathcal{P}_0$ uses P–X-ScanT, the proof is analogous to the outer proof case $\mathcal{P}$ uses P–X-ScanT; for the case $\mathcal{P}_0$ uses P–X-ScanF, the proof can be done by the inner IH.
    Then, by this inner lemma on (1.1) with $\mathcal{P}_0$, (1.2), $\mathcal{P}'_0$, we get $(\vec{a}'_{i0} = \vec{a}''_{i0})_{i=1}^2$, and $\vec{a}'_0 = \vec{a}''_0$.
    Thus $\langle \texttt{F}|\vec{a}'_{10}\rangle = \langle \texttt{F}|\vec{a}''_{10}\rangle$, i.e., $\vec{a}'_1 = \vec{a}''_1$. Likewise, $\vec{a}'_2 = \langle n|\vec{a}'_{20}\rangle = \langle n|\vec{a}''_{20}\rangle = \vec{a}''_2$, and $\vec{a}' = \langle n_0|\vec{a}'_0\rangle = \langle n_0|\vec{a}''_0\rangle = \vec{a}''$, as required.

■

Lemma 1.8 (**Xducer determinism**). *If $\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\vec{c}} \vec{a}_0$ by some derivation $\mathcal{P}$, and $\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\vec{c}} \vec{a}'_0$ by some derivation $\mathcal{P}'$, then $\vec{a}_0 = \vec{a}'_0$.*

Proof. The proof is by induction on the structure of $\vec{c}$.

- Case $\vec{c} = \langle\rangle$
  Then both $\mathcal{P}$ and $\mathcal{P}'$ must use P-X-Termi:
  $$\mathcal{P} = \mathcal{P}' = \frac{}{\psi(\langle\rangle_1, ..., \langle\rangle_k) \Downarrow^{\langle\rangle} \langle\rangle}$$
  so $\vec{a}_0 = \vec{a}'_0 = \langle\rangle$, as required.

- Case $\vec{c} = \langle()|\vec{c}_0\rangle$.
  $\mathcal{P}$ must use P-X-Loop:

$$\mathcal{P} = \frac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}_2\end{array}}{\begin{array}{cc}\psi(\vec{a}_{11}, ..., \vec{a}_{k1}) \downarrow \vec{a}_{01} & \psi(\vec{a}_{12}, ..., \vec{a}_{k2}) \Downarrow^{\vec{c}_0} \vec{a}_{02}\end{array}}{\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\langle()|\vec{c}_0\rangle} \vec{a}_0}$$

  where

$$(\vec{a}_{i1} +\!\!+ \vec{a}_{i2} = \vec{a}_i)_{i=1}^k \tag{1.3}$$

$$\vec{a}_{01} +\!\!+ \vec{a}_{02} = \vec{a}_0 \tag{1.4}$$

  Similarly,

$$\mathcal{P}' = \frac{\begin{array}{cc}\mathcal{P}'_1 & \mathcal{P}'_2\end{array}}{\begin{array}{cc}\psi(\vec{a}'_{11}, ..., \vec{a}'_{k1}) \downarrow \vec{a}'_{01} & \psi(\vec{a}'_{12}, ..., \vec{a}'_{k2}) \Downarrow^{\vec{c}_0} \vec{a}'_{02}\end{array}}{\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\langle()|\vec{c}_0\rangle} \vec{a}'_0}$$

  where

$$(\vec{a}'_{i1} +\!\!+ \vec{a}'_{i2} = \vec{a}_i)_{i=1}^k \tag{1.5}$$

$$\vec{a}'_{01} +\!\!+ \vec{a}'_{02} = \vec{a}'_0 \tag{1.6}$$

  Using Lemma 1.6 on each of the k equations of (1.3), we have

$$(\vec{a}_{i1} \sqsubseteq \vec{a}_i)_{i=1}^k \tag{1.7}$$

  Analogously, from (1.5),

$$(\vec{a}'_{i1} \sqsubseteq \vec{a}_i)_{i=1}^k \tag{1.8}$$

  By Lemma 1.7 on (1.7) with $\mathcal{P}_1$, (1.8), $\mathcal{P}'_1$, we get

$$(\vec{a}_{i1} = \vec{a}'_{i1})_{i=1}^k \tag{1.9}$$

$$\vec{a}_{01} = \vec{a}'_{01} \tag{1.10}$$

  It is easy to show that from (1.3), (1.5) and (1.9) we can get

$$(\vec{a}_{i2} = \vec{a}'_{i2})_{i=1}^k \tag{1.11}$$

  Then by IH on $\mathcal{P}_2$ with $\mathcal{P}'_2$, we obtain $\vec{a}_{02} = \vec{a}'_{02}$.
  Therefore, with (1.4), (1.6), (1.10), we obtain $\vec{a}_0 = \vec{a}_{01} +\!\!+ \vec{a}_{02} = \vec{a}'_{01} +\!\!+ \vec{a}'_{02} = \vec{a}'_0$, as required.

■

Theorem 1.9 (SVCODE$_0$ determinism). *If $\langle p, \sigma\rangle \Downarrow^{\vec{c}} \sigma'$ (by some derivation $\mathcal{P}$) and $\langle p, \sigma\rangle \Downarrow^{\vec{c}} \sigma''$ (by some derivation $\mathcal{P}'$), then $\sigma' = \sigma''$.*

Proof. The proof is by induction on the syntax of p. There are four cases: the case for $p = \epsilon$ is trivial; with the help of Lemma 1.8, the case for $p = s := \psi(s_1, ..., s_k)$ is immediate; proof of $p = p_1; p_2$ can be done by IH; the only interesting case is $p = S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1)$.

- Case $p = S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1)$.
  Assume $S_{out} = \{s_1, ..., s_l\}$. There are two subcases by induction on $\sigma(s_c)$:

– Subcase $\sigma(s_c) = \langle\rangle$.
Then $\mathcal{P}$ and $\mathcal{P}'$ must both use P-WC-EMP, and they must be identical:

$$\mathcal{P} = \mathcal{P}' = \frac{}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1), \sigma\rangle \Downarrow^{\vec{c}} \sigma[(s_i \mapsto \langle\rangle)_{i=1}^{l}]}$$

with $\forall s \in S_{in}.\sigma(s) = \langle\rangle$. So $\sigma' = \sigma'' = \sigma[(s_i \mapsto \langle\rangle)_{i=1}^{l}]$, as required.

– Subcase $\sigma(s_c) \neq \langle\rangle$.
Then we must have

$$\mathcal{P} = \frac{\begin{array}{c}\mathcal{P}_1 \\ \langle p_1, \sigma\rangle \Downarrow^{\vec{c}_1} \sigma_1\end{array}}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1), \sigma\rangle \Downarrow^{\vec{c}} \sigma[(s_i \mapsto \sigma_1(s_i))_{i=1}^{l}]}$$

Also, we have

$$\mathcal{P}' = \frac{\begin{array}{c}\mathcal{P}'_1 \\ \langle p_1, \sigma\rangle \Downarrow^{\vec{c}_1} \sigma'_1\end{array}}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_1), \sigma\rangle \Downarrow^{\vec{c}} \sigma[(s_i \mapsto \sigma'_1(s_i))_{i=1}^{l}]}$$

So $\sigma' = \sigma[(s_i \mapsto \sigma_1(s_i))_{i=1}^{l}]$, and $\sigma'' = \sigma[(s_i \mapsto \sigma'_1(s_i))_{i=1}^{l}]$.
By IH on $\mathcal{P}_1$ and $\mathcal{P}'_1$, we obtain

$$\sigma_1 = \sigma'_1$$

Then it is clear that $\sigma' = \sigma''$, as required.

■

1.3 Translation

(1) Since we do not have pairs, the stream tree type will be :

$$\textbf{STree} \ni st ::= s \mid (st_1, s)$$

(2) Translation environment:

$$\delta = [x_1 \mapsto st_1, ..., x_i \mapsto st_i]$$

The translation judgments shown below can be read as: " in the environment δ, the expression e (or the function call $\phi(st_1, ..., st_k)$) will be translated to an SVCODE$_0$ program p with $\texttt{dv}(p) \subseteq \{s_0, ..., s_1 - 1\}$, and the evaluation result is represented as the streams in st".

- Expression translation rules:

Judgment $\boxed{\delta \ \vdash \ e \Rightarrow_{s_1}^{s_0} (p, st)}$

$$\frac{}{\delta \ \vdash \ x \Rightarrow_{s_0}^{s_0} (\epsilon, st)} \, (\delta(x) = st)$$

$$\frac{\delta \ \vdash \ e_1 \Rightarrow_{s_0'}^{s_0} (p_1, st_1) \qquad \delta[x \mapsto st_1] \ \vdash \ e_2 \Rightarrow_{s_1}^{s_0'} (p_2, st)}{\delta \ \vdash \ \textbf{let } x = e_1 \textbf{ in } e_2 \Rightarrow_{s_1}^{s_0} (p_1; p_2, st)}$$

$$\frac{\phi(st_1, ..., st_k) \Rightarrow_{s_1}^{s_0} (p, st)}{\delta \ \vdash \ \phi(x_1, ..., x_k) \Rightarrow_{s_1}^{s_0} (p, st)} \left((\delta(x_i) = st_i)_{i=1}^{k}\right)$$

$$\frac{[x \mapsto st_1, (x_i \mapsto s_i)_{i=1}^{j}] \ \vdash \ e \Rightarrow_{s_1}^{s_0+1+j} (p_1, st)}{\delta \ \vdash \ \{e : x \textbf{ in } y \textbf{ using } x_1, ..., x_j\} \Rightarrow_{s_1}^{s_0} (p, (st, s_b))} \begin{pmatrix} \delta(y) = (st_1, s_b) \\ (\delta(x_i) = s_i')_{i=1}^{j} \\ p = s_0 := \texttt{Usum}(s_b); \\ \quad (s_i := \texttt{Distr}(s_b, s_i');)_{i=1}^{j} \\ \quad S_{out} := \texttt{WithCtrl}(s_0, S_{in}, p_1) \\ S_{in} = \texttt{fv}(p_1) \\ S_{out} = \overline{st} \cap \texttt{dv}(p_1) \\ s_{i+1} = s_i + 1, \forall i \in \{0, ..., j-1\} \end{pmatrix}$$

- Built-in function translation rules:

  Judgment $\boxed{\phi(st_1, ..., st_k) \Rightarrow_{s_1}^{s_0} (p, st)}$

  $$\frac{}{\textbf{const}_n() \Rightarrow_{s_0+1}^{s_0} (s_0 := \texttt{Const}_\texttt{n}(), s_0)}$$

  $$\frac{}{\textbf{plus}(s_1, s_2) \Rightarrow_{s_0+1}^{s_0} (s_0 := \texttt{MapTwo}_{+}(s_1, s_2), s_0)}$$

  $$\frac{}{\textbf{iota}(s) \Rightarrow_{s_4}^{s_0} (p, (s_3, s_0))} \begin{pmatrix} s_{i+1} = s_i + 1, \forall i \in \{0, ..., 3\} \\ p = s_0 := \texttt{ToFlags}(s); \\ \quad s_1 := \texttt{Usum}(s_0); \\ \quad \overline{s_2} := \texttt{WithCtrl}(s_1, [\,], s_2 := \texttt{Const}_\texttt{1}()); \\ \quad s_3 := \texttt{ScanPlus}_0(s_0, s_2) \end{pmatrix}$$

1.4 Value representation

(1) SVCODE$_0$ values:

$$\textbf{SvVal} \ni w ::= \vec{a} \mid (w, \vec{b})$$

(2) We annotate the operation $+\!\!+$ with the high-level type to represent SVCODE$_0$ values' concatenation:

$$\begin{aligned} &+\!\!+_\tau : \textbf{SvVal} \to \textbf{SvVal} \to \textbf{SvVal} \\ &\vec{a}_1 \ +\!\!+_{\textbf{int}} \ \vec{a}_2 = \vec{a}_1 \ +\!\!+ \ \vec{a}_2 \\ &(w_1, \vec{b}_1) \ +\!\!+_{\{\tau\}} \ (w_2, \vec{b}_2) = (w_1 \ +\!\!+_\tau \ w_2, \vec{b}_1 \ +\!\!+ \ \vec{b}_2) \end{aligned}$$

(3) A function for SVCODE_0 value construction from a stream tree:

$$\begin{aligned}&\sigma^* : \textbf{STree} \to \textbf{SvVal}\\&\sigma^*(s) = \sigma(s)\\&\sigma^*((st, s)) = (\sigma^*(st), \sigma(s))\end{aligned}$$

(4) Value representation rules:

- **Judgment** $\boxed{v \rhd_\tau w}$

$$\frac{}{n \rhd_{\textbf{int}} \langle n \rangle} \qquad \frac{(v_i \rhd_\tau w_i)_{i=1}^k}{\{v_1, ..., v_k\} \rhd_{\{\tau\}} (w, \langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle)} (w = w_1 +\!\!+_\tau ... +\!\!+_\tau w_k)$$

Lemma 1.10 (**Value translation backwards determinism**). *If* $v \rhd_\tau w$, $v' \rhd_\tau w$, *then* $v = v'$.

1.5 Correctness

1.5.1 Definitions

We first define a binary relation $\overset{S}{\sim}$ on stores to denote that two stores are *similar*: they have identical domains, and their bound values by S are the same. We call this S an *overlap* of these two stores.

Definition 1.11 (**Store similarity**). $\sigma_1 \overset{S}{\sim} \sigma_2$ *iff*
(1) $dom(\sigma_1) = dom(\sigma_2)$
(2) $\forall s \in S. \sigma_1(s) = \sigma_2(s)$

According to this definition, it is only meaningful to have $S \subseteq dom(\sigma_1)$ $(= dom(\sigma_2))$. When $S = dom(\sigma_1) = dom(\sigma_2)$, σ_1 and σ_2 are identical. It is easy to show that this relation $\overset{S}{\sim}$ is symmetric and transitive.

- If $\sigma_1 \overset{S}{\sim} \sigma_2$, then $\sigma_2 \overset{S}{\sim} \sigma_1$.
- If $\sigma_1 \overset{S}{\sim} \sigma_2$ and $\sigma_2 \overset{S}{\sim} \sigma_3$, then $\sigma_1 \overset{S}{\sim} \sigma_3$.

We also define a binary operation $\overset{S}{\bowtie}$ on stores to denote a kind of specical concatenation of two similar stores: the *concatenation* of two similar stores is a new store, in which the bound values by S are from any of the parameter stores, and the others are the concatenation of the values from the two stores. In other words, a *concatenation* of two similar stores is only a concatenation of the bound values that *may* be different in these stores.

Definition 1.12 (**Store Concatenation**). *For* $\sigma_1 \overset{S}{\sim} \sigma_2$, $\sigma_1 \overset{S}{\bowtie} \sigma_2 = \sigma$ *where*

$$\sigma(s) = \begin{cases}\sigma_1(s)(= \sigma_2(s)), & s \in S\\ \sigma_1(s) +\!\!+ \sigma_2(s), & s \notin S\end{cases}$$

Clearly, if $\sigma_1 \overset{S}{\bowtie} \sigma_2 = \sigma$, then $\sigma_1 \overset{S}{\sim} \sigma$ and $\sigma_2 \overset{S}{\sim} \sigma$.

Lemma 1.13. *If*

(i) $\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\vec{c}} \vec{a}$ *by some derivation* $\mathcal{P}$

(ii) $\psi(\vec{a}'_1, ..., \vec{a}'_k) \Downarrow^{\vec{c}'} \vec{a}'$ *by some derivation* $\mathcal{P}'$,

then $\psi(\vec{a}_1{+\!+}\vec{a}'_1, ..., \vec{a}_k{+\!+}\vec{a}'_k) \Downarrow^{\vec{c}+\!+\vec{c}'} \vec{a}{+\!+}\vec{a}'$ *by some* $\mathcal{P}''$.

Proof. By induction on the structure of $\vec{c}$.

- Case $\vec{c} = \langle\rangle$.
  Then $\mathcal{P}$ must use P-X-Termi:
  $$\mathcal{P} = \frac{}{\psi(\langle\rangle, ..., \langle\rangle) \Downarrow^{\vec{c}} \langle\rangle}$$
  so $(\vec{a}_i = \langle\rangle)_{i=1}^k$ and $\vec{a} = \langle\rangle$. Then $(\vec{a}_i{+\!+}\vec{a}'_i = \vec{a}'_i)_{i=1}^k$, $\vec{c}{+\!+}\vec{c}' = \vec{c}'$ and $\vec{a}{+\!+}\vec{a}' = \vec{a}'$. Take $\mathcal{P}'' = \mathcal{P}'$ and we are done.

- Case $\vec{c} = \langle()|\vec{c}_0\rangle$.
  Then $\mathcal{P}$ must use P-X-Loop:
  $$\mathcal{P} = \frac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}_2\end{array}}{\begin{array}{cc}\psi(\vec{a}_{11}, ..., \vec{a}_{k1}) \downarrow \vec{a}_{01} & \psi(\vec{a}_{12}, ..., \vec{a}_{k2}) \Downarrow^{\vec{c}_0} \vec{a}_{02}\end{array}}{\psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\langle()|\vec{c}_0\rangle} \vec{a}}$$
  with $(\vec{a}_i = \vec{a}_{i1}{+\!+}\vec{a}_{i2})_{i=1}^k$, and $\vec{a} = \vec{a}_{01}{+\!+}\vec{a}_{02}$.

  By IH on $\vec{c}_0$ with $\mathcal{P}_2$ and $\mathcal{P}'$, we get a derivation $\mathcal{P}'_2$ of
  $$\psi(\vec{a}_{12}{+\!+}\vec{a}'_1, ..., \vec{a}_{k2}{+\!+}\vec{a}'_k) \Downarrow^{\vec{c}_0+\!+\vec{c}'} \vec{a}_{02}{+\!+}\vec{a}'$$
  Then using the rule P-X-Loop we can build a derivation $\mathcal{P}'''$ as follows:
  $$\frac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}'_2\end{array}}{\begin{array}{cc}\psi(\vec{a}_{11}, ..., \vec{a}_{k1}) \downarrow \vec{a}_{01} & \psi(\vec{a}_{12}{+\!+}\vec{a}'_1, ..., \vec{a}_{k2}{+\!+}\vec{a}'_k) \Downarrow^{\vec{c}_0+\!+\vec{c}'} \vec{a}_{02}{+\!+}\vec{a}'\end{array}}{\psi(\vec{a}_{11}{+\!+}(\vec{a}_{12}{+\!+}\vec{a}'_1), ..., \vec{a}_{k1}{+\!+}(\vec{a}_{k2}{+\!+}\vec{a}'_k)) \Downarrow^{\langle()|\vec{c}_0+\!+\vec{c}'\rangle} \vec{a}_{01}{+\!+}(\vec{a}_{02}{+\!+}\vec{a}')}$$
  Since it is clear that
  $$\forall i \in \{1, ..., k\}.\ \vec{a}_{i1}{+\!+}(\vec{a}_{i2}{+\!+}\vec{a}'_i) = (\vec{a}_{i1}{+\!+}\vec{a}_{i2}){+\!+}\vec{a}'_i = \vec{a}_i{+\!+}\vec{a}'_i$$
  $$\langle()|\vec{c}'_1{+\!+}\vec{c}_2\rangle = \langle()|\vec{c}'_1\rangle{+\!+}\vec{c}_2 = \vec{c}_1{+\!+}\vec{c}_2$$
  $$\vec{a}_{01}{+\!+}(\vec{a}_{02}{+\!+}\vec{a}') = (\vec{a}_{01}{+\!+}\vec{a}_{02}){+\!+}\vec{a}' = \vec{a}{+\!+}\vec{a}'$$
  so $\mathcal{P}'' = \mathcal{P}'''$, and we are done.

■

Lemma 1.14. *If* $\langle p, \sigma\rangle \Downarrow^{\vec{c}} \sigma'$, *then* $\forall s$ *in* $\mathtt{fv}(p).\sigma'(s) = \sigma(s)$.

Lemma 1.15. *If*

(i) $\langle p, \sigma_1\rangle \Downarrow^{\vec{c}} \sigma'_1$

(ii) $\forall s \in \mathtt{fv}(p).\sigma_2(s) = \sigma_1(s)$

then

(iv) $\langle p, \sigma_2 \rangle \Downarrow^{\vec{c}} \sigma_2'$

(v) $\forall s \in \mathtt{dv}(p).\sigma_2'(s) = \sigma_1'(s)$

Lemma 1.16 (**Store concatenation lemma**). *If*

(i) $\sigma_1 \overset{S}{\sim} \sigma_2$

(ii) $\langle p, \sigma_1 \rangle \Downarrow^{\vec{c}_1} \sigma_1'$ *(by some derivation* $\mathcal{P}_1$*)*

(iii) $\langle p, \sigma_2 \rangle \Downarrow^{\vec{c}_2} \sigma_2'$ *(by some derivation* $\mathcal{P}_2$*)*

(iv) $\mathtt{fv}(p) \cap S = \emptyset$

then $\sigma_1' \overset{S}{\sim} \sigma_2'$ *and* $\left\langle p, \sigma_1 \overset{S}{\bowtie} \sigma_2 \right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} \sigma_1' \overset{S}{\bowtie} \sigma_2'$ *(by* $\mathcal{P}$*).*

We will need this lemma to prove that the concatenation of the results of partial computations inside a comprehension body (i.e. p in the lemma) is equivalent to the result of the entire parallel computation. From the other direction, it can be considered as splitting the computation of p into subcomputations with smaller parallel degrees; all the supplier streams, i.e., $\mathtt{fv}(p)$, are split correspondingly to feed the Xducers of p. These smaller parallel degrees are specified by the control streams, i.e., $\vec{c}_1$ and $\vec{c}_2$ in the lemma. Other untouched streams (i.e., S) of σs have no change during this process.

Proof. By induction on the syntax of p.

- Case $p = \epsilon$. $\dfrac{}{\langle \epsilon, \sigma_1 \rangle \Downarrow^{\vec{c}_1} \sigma_1}$, and $\mathcal{P}_2$ must be $\dfrac{}{\langle \epsilon, \sigma_2 \rangle \Downarrow^{\vec{c}_2} \sigma_2}$.
  So $\sigma_1' = \sigma_1$, and $\sigma_2' = \sigma_2$, thus $\sigma_1' \overset{S}{\sim} \sigma_2'$, and $\sigma_1' \overset{S}{\bowtie} \sigma_2' = \sigma_1 \overset{S}{\bowtie} \sigma_2$.

  By P-Empty, we take $\mathcal{P} = \dfrac{}{\left\langle \epsilon, (\sigma_1 \overset{S}{\bowtie} \sigma_2) \right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} (\sigma_1 \overset{S}{\bowtie} \sigma_2)}$ and we are done.

- Case $p = s_l := \psi(s_1, ..., s_k)$.
  $\mathcal{P}_1$ must look like

$$\dfrac{\begin{array}{c}\mathcal{P}_1' \\ \psi(\vec{a}_1, ..., \vec{a}_k) \Downarrow^{\vec{c}_1} \vec{a}\end{array}}{\langle s_l := \psi(s_1, ..., s_k), \sigma_1 \rangle \Downarrow^{\vec{c}_1} \sigma_1[s_l \mapsto \vec{a}]}$$

  and we have

$$(\sigma_1(s_i) = \vec{a}_i)_{i=1}^{k} \tag{1.12}$$

  Similarly, $\mathcal{P}_2$ must look like

$$\dfrac{\begin{array}{c}\mathcal{P}_2' \\ \psi(\vec{a}_1', ..., \vec{a}_k') \Downarrow^{\vec{c}_2} \vec{a}'\end{array}}{\langle s_l := \psi(s_1, ..., s_k), \sigma_2 \rangle \Downarrow^{\vec{c}_2} \sigma_2[s_l \mapsto \vec{a}']}$$

and we have

$$(\sigma_2(s_i) = \vec{a}'_i)_{i=1}^{k} \tag{1.13}$$

So $\sigma'_1 = \sigma_1[s_l \mapsto \vec{a}], \sigma'_2 = \sigma_2[s_l \mapsto \vec{a}']$, and clearly, $\sigma'_1 \overset{S}{\sim} \sigma'_2$.

From assumption (iv) we have $\mathtt{fv}(s_l := \psi(s_1, ..., s_k)) \cap S = \emptyset$, that is,

$$\{s_1, ..., s_k\} \cap S = \emptyset \tag{1.14}$$

By Lemma 1.13 on $\mathcal{P}'_1$, $\mathcal{P}'_2$, we get a derivation $\mathcal{P}'$ of

$$\psi(\vec{a}_1{+\!\!+}\vec{a}'_1, ..., \vec{a}_k{+\!\!+}\vec{a}'_k) \Downarrow^{\vec{c}_1{+\!\!+}\vec{c}_2} \vec{a}{+\!\!+}\vec{a}'$$

Since $\sigma_1 \overset{S}{\sim} \sigma_2$, with (1.12),(1.13) and (1.14), by Definition 1.12 we have

$$\forall i \in \{1, ..., k\}.(\sigma_1 \overset{S}{\bowtie} \sigma_2)(s_i) = \sigma_1(s_i){+\!\!+}\sigma_2(s_i) = \vec{a}_i{+\!\!+}\vec{a}'_i \tag{1.15}$$

Also, it is easy to prove that $\sigma_1[s_l \mapsto \vec{a}] \overset{S}{\bowtie} \sigma_2[s_l \mapsto \vec{a}'] \overset{S}{\sim} (\sigma_1 \overset{S}{\bowtie} \sigma_2)[s_l \mapsto \vec{a}{+\!\!+}\vec{a}']$ and

$$\sigma_1[s_l \mapsto \vec{a}] \overset{S}{\bowtie} \sigma_2[s_l \mapsto \vec{a}'] = (\sigma_1 \overset{S}{\bowtie} \sigma_2)[s_l \mapsto \vec{a}{+\!\!+}\vec{a}'] \tag{1.16}$$

Using the rule P-XDUCER with (1.15), we can build $\mathcal{P}''$ as follows

$$\frac{\begin{array}{c}\mathcal{P}' \\ \psi(\vec{a}_1{+\!\!+}\vec{a}'_1, ..., \vec{a}_k{+\!\!+}\vec{a}'_k) \Downarrow^{\vec{c}_1{+\!\!+}\vec{c}_2} \vec{a}{+\!\!+}\vec{a}'\end{array}}{\left\langle s_l := \psi(s_1, ..., s_k), (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1{+\!\!+}\vec{c}_2} (\sigma_1 \overset{S}{\bowtie} \sigma_2)[s_l \mapsto \vec{a}{+\!\!+}\vec{a}']}$$

With (1.16), we take $\mathcal{P} = \mathcal{P}''$, as required.

- Case $p = S_{out} := \mathtt{WithCtrl}(s_c, S_{in}, p_0)$ where

$$\mathtt{fv}(p_0) \subseteq S_{in} \tag{1.17}$$

$$S_{out} \subseteq \mathtt{dv}(p_0) \tag{1.18}$$

From the assumption (iv), we have

$$\begin{aligned}\mathtt{fv}(S_{out} := \mathtt{WithCtrl}(s_c, S_{in}, p_0)) \cap S = \emptyset& \\ (\{s_c\} \cup S_{in}) \cap S = \emptyset& \qquad \text{(by definition of } \mathtt{fv}())\end{aligned}$$

thus

$$\{s_c\} \cap S = \emptyset \tag{1.19}$$

$$S_{in} \cap S = \emptyset \tag{1.20}$$

Since (1.17) with (1.20), we also have

$$\mathtt{fv}(p_0) \cap S = \emptyset \tag{1.21}$$

Assume $S_{out} = [s_1, ..., s_j]$.
There are four possibilities:

– Subcase both $\mathcal{P}_1$ and $\mathcal{P}_2$ use P-Wc-Emp.

So $\mathcal{P}_1$ must look like

$$\frac{}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), \sigma_1\rangle \Downarrow^{\vec{c}_1} \sigma_1[s_1 \mapsto \langle\rangle, ..., s_j \mapsto \langle\rangle]}$$

and we have

$$\forall s \in \{s_c\} \cup S_{in}.\sigma_1(s) = \langle\rangle \tag{1.22}$$

thus

$$\sigma_1(s_c) = \langle\rangle \tag{1.23}$$

$$\forall s \in \texttt{fv}(p_0).\sigma_1(s) = \langle\rangle \tag{1.24}$$

Similarly,$\mathcal{P}_2$ must look like

$$\frac{}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), \sigma_2\rangle \Downarrow^{\vec{c}_2} \sigma_2[s_1 \mapsto \langle\rangle, ..., s_j \mapsto \langle\rangle]}$$

and we have

$$\forall s \in \{s_c\} \cup S_{in}.\sigma_2(s) = \langle\rangle \tag{1.25}$$

thus

$$\sigma_2(s_c) = \langle\rangle \tag{1.26}$$

$$\forall s \in \texttt{fv}(p_0).\sigma_2(s) = \langle\rangle \tag{1.27}$$

So $\sigma_1' = \sigma_1[(s_i \mapsto \langle\rangle)_{i=1}^j]$, $\sigma_2' = \sigma_2[(s_i \mapsto \langle\rangle)_{i=1}^j]$.

Since $\sigma_1 \overset{S}{\sim} \sigma_2$, by Definition 1.12 with (1.19), (1.20), and (1.22), (1.25), we have

$$\forall s \in \{s_c\} \cup S_{in}.(\sigma_1 \overset{S}{\bowtie} \sigma_2)(s) = \sigma_1(s) +\!\!+ \sigma_2(s) = \langle\rangle \tag{1.28}$$

Also, it is easy to show that $\sigma_1[(s_i \mapsto \langle\rangle)_{i=1}^j] \overset{S}{\sim} \sigma_2[(s_i \mapsto \langle\rangle)_{i=1}^j]$ and

$$\sigma_1[(s_i \mapsto \langle\rangle)_{i=1}^j] \overset{S}{\bowtie} \sigma_2[(s_i \mapsto \langle\rangle)_{i=1}^j] = (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto \langle\rangle)_{i=1}^j] \tag{1.29}$$

Using P-Wc-Emp with (1.28), we can build a derivation $\mathcal{P}'$ as follows

$$\frac{}{\left\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1 +\!\!+ \vec{c}_2} (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto \langle\rangle)_{i=1}^j]}$$

With (1.29), we take $\mathcal{P} = \mathcal{P}'$, as requierd.

– Subcase $\mathcal{P}_1$ uses P-Wc-Nomemp, $\mathcal{P}_2$ uses P-Wc-Emp.
$\mathcal{P}_1$ must look like

$$\frac{\begin{array}{c}\mathcal{P}_1' \\ \langle p_0, \sigma_1\rangle \Downarrow^{\vec{c}_1'} \sigma_1''\end{array}}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), \sigma_1\rangle \Downarrow^{\vec{c}_1} \sigma_1[(s_i \mapsto \sigma_1''(s_i))_{i=1}^j]}$$

and we have

$$\sigma_1(s_c) = \vec{c}_1' \neq \langle\rangle \tag{1.30}$$

$\mathcal{P}_2$ must look like

$$\frac{}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), \sigma_2\rangle \Downarrow^{\vec{c}_2} \sigma_2[(s_i \mapsto \langle\rangle)_{i=1}^{j}]}$$

and we have $\forall s \in \{s_c\} \cup S_{in}.\sigma_2(s) = \langle\rangle$ thus

$$\sigma_2(s_c) = \langle\rangle \tag{1.31}$$

$$\forall s \in \texttt{fv}(p_0).\sigma_2(s) = \langle\rangle \tag{1.32}$$

So $\sigma_1' = \sigma_1[(s_i \mapsto \sigma_1''(s_i))_{i=1}^{j}]$, $\sigma_2' = \sigma_2[(s_i \mapsto \langle\rangle)_{i=1}^{j}]$.
Since $\sigma_1 \overset{S}{\sim} \sigma_2$, it is easy to show that

$$\forall s \in \texttt{fv}(p_0).(\sigma_1 \overset{S}{\bowtie} \sigma_2)(s) = \sigma_1(s) +\!\!+ \sigma_2(s) = \sigma_1(s) \tag{1.33}$$

Then by Lemma 1.15 on $\mathcal{P}_1'$ with (1.33), we obtain a derivation $\mathcal{P}_0$ of

$$\left\langle p_0, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1} \sigma_0$$

for some σ_0, and

$$\forall s \in \texttt{dv}(p_0).\sigma_0(s) = \sigma_1''(s),$$

Then, with (1.18), we have

$$(\sigma_0(s_i) = \sigma_1''(s_i))_{i=1}^{j} \tag{1.34}$$

Since $\sigma_1 \overset{S}{\sim} \sigma_2$, by Definition 1.12 with (1.30), (1.31), we have

$$(\sigma_1 \overset{S}{\bowtie} \sigma_2)(s_c) = \sigma_1(s_c) +\!\!+ \sigma_2(s_c) = \vec{c}_1' \neq \langle\rangle \tag{1.35}$$

and it is also easy to show $\sigma_1' \overset{S}{\sim} \sigma_2'$ and

$$(\sigma_1' \overset{S}{\bowtie} \sigma_2') = (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto \sigma_1''(s_1))_{i=1}^{j}] \tag{1.36}$$

With (1.34), we replace $\sigma_1''(s_i)$ with $\sigma_0(s_i)$ for $\forall i \in \{1, ..., j\}$ in (1.36), giving us

$$(\sigma_1' \overset{S}{\bowtie} \sigma_2') = (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto \sigma_0(s_1))_{i=1}^{j}] \tag{1.37}$$

Using the rule P-Wc-Nonemp with (1.35) we can build a derivation $\mathcal{P}'$ as follows

$$\frac{\begin{array}{c}\mathcal{P}_0 \\ \left\langle p_0, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1} \sigma_0\end{array}}{\left\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1 +\!\!+ \vec{c}_2} (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto \sigma_0(s_i))_{i=1}^{j}]}$$

Then with (1.37), we take $\mathcal{P} = \mathcal{P}'$, as required.

– Subcase $\mathcal{P}_1$ uses P-Wc-Emp and $\mathcal{P}_2$ uses P-Wc-Nonemp.
This subcase is analogous to the previous one.

– Subcase both $\mathcal{P}_1$ and $\mathcal{P}_2$ use P-Wc-Nonemp.

$\mathcal{P}_1$ must look like

$$\frac{\begin{array}{c}\mathcal{P}_1' \\ \langle p_0, \sigma_1\rangle \Downarrow^{\vec{c}_1} \sigma_1''\end{array}}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), \sigma_1\rangle \Downarrow^{\vec{c}_1} \sigma_1[(s_i \mapsto \sigma_1''(s_i))_{i=1}^{j}]}$$

and

$$\sigma_1(s_c) = \vec{c}_1' \neq \langle\rangle \tag{1.38}$$

Similarly, $\mathcal{P}_2$ must look like

$$\frac{\begin{array}{c}\mathcal{P}_2' \\ \langle p_0, \sigma_2\rangle \Downarrow^{\vec{c}_2} \sigma_2''\end{array}}{\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), \sigma_2\rangle \Downarrow^{\vec{c}_2} \sigma_2[(s_i \mapsto \sigma_2''(s_i))_{i=1}^{j}]}$$

and

$$\sigma_2(s_c) = \vec{c}_2' \neq \langle\rangle \tag{1.39}$$

So $\sigma_1' = \sigma_1[(s_i \mapsto \sigma_1''(s_i))_{i=1}^{j}], \sigma_2' = \sigma_2[(s_i \mapsto \sigma_2''(s_i))_{i=1}^{j}]$.
By IH on $\mathcal{P}_1'$, $\mathcal{P}_2'$ with (1.21), we get a derivation $\mathcal{P}_0$ of

$$\left\langle p_0, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1' +\!\!+ \vec{c}_2'} \sigma_1'' \overset{S}{\bowtie} \sigma_2''$$

Since $\forall i \in \{1, ..., j\}.s_i \notin S$, then by Definition 1.12, we know

$$(\sigma_1'' \overset{S}{\bowtie} \sigma_2'')(s_i) = \sigma_1''(s_i) +\!\!+ \sigma_2''(s_i) \tag{1.40}$$

Also, it is easy to show that $\sigma_1' \overset{S}{\sim} \sigma_2'$, and

$$(\sigma_1' \overset{S}{\bowtie} \sigma_2') = (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto \sigma_1''(s_i) +\!\!+ \sigma_2''(s_i))_{i=1}^{j}] \tag{1.41}$$

Then, with (1.40), we replace $\sigma_1''(s_i) +\!\!+ \sigma_2''(s_i)$ with $(\sigma_1'' \overset{S}{\bowtie} \sigma_2'')(s_i)$ for $\forall i \in \{1, ..., j\}$ in (1.41), giving us

$$(\sigma_1' \overset{S}{\bowtie} \sigma_2') = (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto (\sigma_1'' \overset{S}{\bowtie} \sigma_2'')(s_i))_{i=1}^{j}] \tag{1.42}$$

Since (1.19) with (1.38), (1.39), we know $(\sigma_1 \overset{S}{\bowtie} \sigma_2)(s_c) = \vec{c}_1' +\!\!+ \vec{c}_2' \neq \langle\rangle$, therefore we can use the rule P-Wc-Nonemp to build a derivation $\mathcal{P}'$ as follows:

$$\frac{\begin{array}{c}\mathcal{P}_0 \\ \left\langle p_0, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1' +\!\!+ \vec{c}_2'} \sigma_1'' \overset{S}{\bowtie} \sigma_2''\end{array}}{\left\langle S_{out} := \texttt{WithCtrl}(s_c, S_{in}, p_0), (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1 +\!\!+ \vec{c}_2} (\sigma_1 \overset{S}{\bowtie} \sigma_2)[(s_i \mapsto (\sigma_1'' \overset{S}{\bowtie} \sigma_2'')(s_i))_{i=1}^{j}]}$$

.
Therefore, with (1.42), we take $\mathcal{P} = \mathcal{P}'$, as required.

- Case $p = p_1; p_2$

  We must have

$$\mathcal{P}_1 = \frac{\begin{array}{cc}\mathcal{P}_1' & \mathcal{P}_1'' \\ \langle p_1, \sigma_1\rangle \Downarrow^{\vec{c}_1} \sigma_1'' & \langle p_2, \sigma_1''\rangle \Downarrow^{\vec{c}_1} \sigma_1'\end{array}}{\langle p_1; p_2, \sigma_1\rangle \Downarrow^{\vec{c}_1} \sigma_1'}$$

  and

$$\mathcal{P}_2 = \frac{\begin{array}{cc}\mathcal{P}_2' & \mathcal{P}_2'' \\ \langle p_1, \sigma_2\rangle \Downarrow^{\vec{c}_2} \sigma_2'' & \langle p_2, \sigma_2''\rangle \Downarrow^{\vec{c}_2} \sigma_2'\end{array}}{\langle p_1; p_2, \sigma_2\rangle \Downarrow^{\vec{c}_1} \sigma_2'}$$

  Since $\mathtt{fv}(p_1; p_2) \cap S = \emptyset$, we have $(\mathtt{fv}(p_1) \cup \mathtt{fv}(p_2) - \mathtt{dv}(p_1)) \cap S = \emptyset$, thus

$$\mathtt{fv}(p_1) \cap S = \emptyset \tag{1.43}$$

$$\mathtt{fv}(p_2) \cap S = \emptyset \tag{1.44}$$

  By IH on $\mathcal{P}_1'$, $\mathcal{P}_2'$, (1.43), we get

$$\sigma_1'' \overset{S}{\sim} \sigma_2'' \tag{1.45}$$

  and a derivation $\mathcal{P}'$ of

$$\left\langle p_1, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} \sigma_1'' \overset{S}{\bowtie} \sigma_2''$$

  Likewise, by IH on (1.45) with $\mathcal{P}_1''$, $\mathcal{P}_2''$ and (1.44), we get $\sigma_1' \overset{S}{\sim} \sigma_2'$, and a derivation $\mathcal{P}''$ of

$$\left\langle p_2, \sigma_1'' \overset{S}{\bowtie} \sigma_2''\right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} (\sigma_1' \overset{S}{\bowtie} \sigma_2')$$

  Therefore, we use the rule P-Seq to build $\mathcal{P}$ as follows:

$$\frac{\begin{array}{cc}\mathcal{P}' & \mathcal{P}'' \\ \left\langle p_1, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} \sigma_1'' \overset{S}{\bowtie} \sigma_2'' & \left\langle p_2, \sigma_1'' \overset{S}{\bowtie} \sigma_2''\right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} (\sigma_1' \overset{S}{\bowtie} \sigma_2')\end{array}}{\left\langle p_1; p_2, (\sigma_1 \overset{S}{\bowtie} \sigma_2)\right\rangle \Downarrow^{\vec{c}_1 +\!+ \vec{c}_2} (\sigma_1' \overset{S}{\bowtie} \sigma_2')}$$

  and we are done.

■

Let $\sigma_1 \overset{<s}{=\!=} \sigma_2$ denote $\forall s' < s. \sigma_1(s') = \sigma_2(s')$.

Lemma 1.17. *If* $\sigma_1 \overset{S_1}{\sim} \sigma_1'$, $\sigma_2 \overset{S_2}{\sim} \sigma_2'$, $\sigma_1 \overset{<s}{=\!=} \sigma_2$, *and* $\sigma_1' \overset{<s}{=\!=} \sigma_2'$ *then* $\sigma_1 \overset{S_1}{\bowtie} \sigma_1' \overset{<s}{=\!=} \sigma_2 \overset{S_2}{\bowtie} \sigma_2'$.

1.5.2 Correctness proof

Lemma 1.18. *If*

(i) $\phi : (\tau_1, ..., \tau_k) \to \tau$ *(by some derivation* $\mathcal{T}$*)*

(ii) $\phi(v_1, ..., v_k) \downarrow v$ *(by* $\mathcal{E}$*)*

(iii) $\phi(st_1, ..., st_k) \Rightarrow_{s_1}^{s_0} (p, st)$ *(by* $\mathcal{C}$*)*

(iv) $(v_i \rhd_{\tau_i} \sigma(st_i))_{i=1}^k$

(v) $\bigcup_{i=1}^k \mathtt{sids}(st_i) \lessdot s_0$

then

(vi) $\langle p, \sigma \rangle \Downarrow^{\langle () \rangle} \sigma'$ *(by* $\mathcal{P}$*)*

(vii) $v \rhd_\tau \sigma'(st)$ *(by* $\mathcal{R}$*)*

(viii) $\sigma' \overset{<s_0}{=\!=} \sigma$

(ix) $s_0 \leq s_1$

(x) $\mathtt{sids}(st) \lessdot s_1$

Proof. By inducntion on the syntax of ϕ.

- Case $\phi = \mathbf{const}_n$
  There is only one possibility for each of $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{C}$:

$$\mathcal{T} = \frac{}{\mathbf{const}_n : () \to \mathbf{int}}$$

$$\mathcal{E} = \frac{}{\vdash\ \mathbf{const}_n() \downarrow n}$$

$$\mathcal{C} = \frac{}{\mathbf{const}_n() \Rightarrow_{s_0+1}^{s_0} (s_0 := \mathtt{Const_n}(), s_0)}$$

  So $k = 0, \tau = \mathbf{int}, v = n, p = s_0 := \mathtt{Const_n}()$, $s_1 = s_0 + 1$, and $st = s_0$
  By P-Xducer, P-X-Loop, P-X-Termi and P-X-Const, we can construct $\mathcal{P}$ as follows:

$$\mathcal{P} = \frac{\dfrac{\dfrac{}{\mathtt{Const_n}() \downarrow \langle n \rangle} \quad \dfrac{}{\mathtt{Const_n}() \Downarrow^{\langle \rangle} \langle \rangle}}{\mathtt{Const_n}() \Downarrow^{\langle () \rangle} \langle n \rangle}}{\langle s_0 := \mathtt{Const_n}(), \sigma \rangle \Downarrow^{\langle () \rangle} \sigma[s_0 \mapsto \langle n \rangle]}$$

  So $\sigma' = \sigma[s_0 \mapsto \langle n \rangle]$.
  Then we take $\mathcal{R} = \dfrac{}{n \rhd_{\mathbf{int}} \sigma'(s_0)}$.
  Also clearly, $\sigma' \overset{<s_0}{=\!=} \sigma$, $s_0 \leq s_0 + 1$, $\mathtt{sids}(s_0) \lessdot s_0 + 1$, and we are done.

- Case $\phi = \mathbf{plus}$
  We must have

$$\mathcal{T} = \frac{}{\mathbf{plus} : (\mathbf{int}, \mathbf{int}) \to \mathbf{int}}$$

$$\mathcal{E} = \frac{}{\vdash\ \mathbf{plus}(n_1, n_2) \downarrow n_3}$$

where $n_3 = n_2 + n_1$, and

$$\mathcal{C} = \overline{\mathbf{plus}(s_1, s_2) \Rightarrow_{s_0+1}^{s_0} (s_0 := \mathtt{MapTwo}_+(s_1, s_2), s_0)}$$

So $k = 2, \tau_1 = \tau_2 = \tau = \mathbf{int}, v_1 = n_1, v_2 = n_2, v = n_3, st_1 = s_1, st_2 = s_2, st = s_0, s_1 = s_0 + 1$ and $p = s_0 := \mathtt{MapTwo}_+(s_1, s_2)$.

Assumption (iv) gives us $\overline{n_1 \triangleright_{\mathbf{int}} \sigma(s_1)}$ and $\overline{n_2 \triangleright_{\mathbf{int}} \sigma(s_2)}$, which implies $\sigma(s_1) = \langle n_1 \rangle$ and $\sigma(s_2) = \langle n_2 \rangle$ respectively.

For (v) we have $s_1 < s_0$ and $s_2 < s_0$.

Then using P-XDUCER with $\sigma(s_1) = \langle n_1 \rangle$ and $\sigma(s_2) = \langle n_2 \rangle$, and using P-X-LOOP and P-X-TERMI, we can build $\mathcal{P}$ as follows:

$$\dfrac{\dfrac{\overline{\mathtt{MapTwo}_+(\langle n_1 \rangle, \langle n_2 \rangle) \downarrow \langle n_3 \rangle} \quad \overline{\mathtt{MapTwo}_+(\langle\rangle, \langle\rangle) \Downarrow^{\langle\rangle} \langle\rangle}}{\mathtt{MapTwo}_+(\langle n_1 \rangle, \langle n_2 \rangle) \Downarrow^{\langle()\rangle} \langle n_3 \rangle}}{\left\langle s_0 := \mathtt{MapTwo}_+(s_1, s_2), \sigma \right\rangle \Downarrow^{\langle()\rangle} \sigma[s_0 \mapsto \langle n_3 \rangle]}$$

Therefore, $\sigma' = \sigma[s_0 \mapsto \langle n_3 \rangle]$.
Now we can take $\mathcal{R} = \overline{n_3 \triangleright_{\mathbf{int}} \sigma'(s_0)}$, and it is clear that $\sigma' \overset{<s_0}{=\!=} \sigma$, $s_0 \leq s_0 + 1$ and $\mathtt{sids}(s_0) \lessdot s_0 + 1$ as required.

- Case $\phi = \mathbf{iota}$

■

Theorem 1.19. ***If***

(i) $\Gamma \vdash e : \tau$ *(by some derivation* $\mathcal{T}$*)*

(ii) $\rho \vdash e \downarrow v$ *(by some* $\mathcal{E}$*)*

(iii) $\delta \vdash e \Rightarrow_{s_1}^{s_0} (p, st)$ *(by some* $\mathcal{C}$*)*

(iv) $\forall x \in dom(\Gamma).\ \vdash \rho(x) : \Gamma(x)$

(v) $\forall x \in dom(\Gamma).\overline{\delta(x)} \lessdot s_0$

(vi) $\forall x \in dom(\Gamma).\rho(x) \triangleright_{\Gamma(x)} \sigma(\delta(x))$

then

(vii) $\langle p, \sigma \rangle \Downarrow^{\langle()\rangle} \sigma'$ *(by some derivation* $\mathcal{P}$*)*

(viii) $v \triangleright_\tau \sigma'(st)$ *(by some* $\mathcal{R}$*)*

(ix) $\sigma' \overset{<s_0}{=\!=} \sigma$

(x) $s_0 \leq s_1$

(xi) $\overline{st} \lessdot s_1$

Proof. By induction on the syntax of e.

- Case $e = x$.
  We must have
  $$\mathcal{T} = \frac{}{\Gamma \ \vdash \ x : \tau}\,(\Gamma(x) = \tau)$$
  $$\mathcal{E} = \frac{}{\rho \ \vdash \ x \downarrow v}\,(\rho(x) = v)$$
  $$\mathcal{C} = \frac{}{\delta \ \vdash \ x \Rightarrow_{s_0}^{s_0} (\epsilon, st)}\,(\delta(x) = st)$$
  So $p = \epsilon$.
  Immediately we have $\mathcal{P} = \dfrac{}{\langle \epsilon, \sigma \rangle \Downarrow^{\langle () \rangle} \sigma}$
  So $\sigma' = \sigma$, which implies $\sigma' \overset{<s_0}{=\!=} \sigma$.
  From the assumptions (iv),(v) and(vi) we already have $v \triangleright_\tau \sigma(st)$, and $\overline{st} \lessdot s_0$.
  Finally it's clear that $s_0 \leq s_0$, and we are done.

- Case $e = \textbf{let } x = e_1 \textbf{ in } e_2$.
  We must have:
  $$\mathcal{T} = \frac{\begin{matrix}\mathcal{T}_1 & \mathcal{T}_2 \\ \Gamma \ \vdash \ e_1 : \tau_1 & \Gamma[x \mapsto \tau_1] \ \vdash \ e_2 : \tau\end{matrix}}{\Gamma \ \vdash \ \textbf{let } x = e_1 \textbf{ in } e_2 : \tau}$$
  $$\mathcal{E} = \frac{\begin{matrix}\mathcal{E}_1 & \mathcal{E}_2 \\ \rho \ \vdash \ e_1 \downarrow v_1 & \rho[x \mapsto v_1] \ \vdash \ e_2 \downarrow v\end{matrix}}{\rho \ \vdash \ \textbf{let } x = e_1 \textbf{ in } e_2 \downarrow v}$$
  $$\mathcal{C} = \frac{\begin{matrix}\mathcal{C}_1 & \mathcal{C}_2 \\ \delta \ \vdash \ e_1 \Rightarrow_{s_0'}^{s_0} (p_1, st_1) & \delta[x \mapsto st_1] \ \vdash \ e_2 \Rightarrow_{s_1}^{s_0'} (p_2, st)\end{matrix}}{\delta \ \vdash \ \textbf{let } x = e_1 \textbf{ in } e_2 \Rightarrow_{s_1}^{s_0} (p_1; p_2, st)}$$

  So $p = p_1; p_2$.

  By IH on $\mathcal{T}_1$ with $\mathcal{E}_1, \mathcal{C}_1$, we get

  (a) $\mathcal{P}_1$ of $\langle p_1, \sigma \rangle \Downarrow^{\langle () \rangle} \sigma_1$
  (b) $\mathcal{R}_1$ of $v_1 \triangleright_{\tau_1} \sigma_1(st_1)$
  (c) $\sigma_1 \overset{<s_0}{=\!=} \sigma$
  (d) $s_0 \leq s_0'$
  (e) $\overline{st_1} \lessdot s_0'$

  From (b), we know $\rho[x \mapsto v_1](x) : \Gamma[x \mapsto \tau_1](x)$ and $\rho[x \mapsto v_1](x) \triangleright_{\Gamma[x \mapsto \tau_1](x)} \sigma_1(\delta[x \mapsto st_1](x))$ must hold. From (e), we have $\overline{\delta[x \mapsto st_1](x)} \lessdot s_0'$.
  Then by IH on $\mathcal{T}_2$ with $\mathcal{E}_2, \mathcal{C}_2$, we get

  (f) $\mathcal{P}_2$ of $\langle p_2, \sigma_1 \rangle \Downarrow^{\langle () \rangle} \sigma_2$
  (g) $\mathcal{R}_2$ of $\sigma_2 \triangleright_\tau \sigma_2(st)$

(h) $\sigma_2 \overset{<s_0'}{=\!=\!=} \sigma_1$

(i) $s_0' \leq s_1$

(j) $\overline{st} \lessdot s_1$

So we can construct:

$$\mathcal{P} = \frac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}_2\end{array}}{\begin{array}{cc}\langle p_1, \sigma\rangle \Downarrow^{\langle()\rangle} \sigma_1 & \langle p_2, \sigma_1\rangle \Downarrow^{\langle()\rangle} \sigma_2\end{array}}$$
$$\langle p_1; p_2, \sigma\rangle \Downarrow^{\langle()\rangle} \sigma_2$$

From (c), (d) and (h), it is clear that $\sigma_2 \overset{<s_0}{=\!=\!=} \sigma_1 \overset{<s_0}{=\!=\!=} \sigma$. From (d) and (i), $s_0 \leq s_1$.

Take $\sigma' = \sigma_2$ (thus $\mathcal{R} = \mathcal{R}_2$) and we are done.

- Case $e = \phi(x_1, ..., x_k)$
  We must have

$$\mathcal{T} = \frac{\begin{array}{c}\mathcal{T}_1\\ \phi : (\tau_1, ..., \tau_k) \to \tau\end{array}}{\Gamma \vdash \phi(x_1, ..., x_k) : \tau} ((\Gamma(x_i) = \tau_i)_{i=1}^k)$$

$$\mathcal{E} = \frac{\begin{array}{c}\mathcal{E}_1\\ \vdash (v_1, ..., v_k)(\downarrow) \downarrow v\end{array}}{\rho \vdash \phi(x_1, ..., x_k) \downarrow v} ((\rho(x_i) = v_i)_{i=1}^k)$$

$$\mathcal{C} = \frac{\begin{array}{c}\mathcal{C}_1\\ \phi(st_1, ..., st_k) \Rightarrow_{s_1}^{s_0} (p, st)\end{array}}{\delta \vdash \phi(x_1, ..., x_k) \Rightarrow_{s_1}^{s_0} (p, st)} ((\delta(x_i) = st_i)_{i=1}^k)$$

From the assumptions (iv),(v) and (vi), for all $i \in \{1, ..., k\}$:

(iv) $\vdash \rho(x_i) : \Gamma(x_i)$, that is, $\vdash v_i : \tau_i$

(v) $\overline{\delta(x_i)} \lessdot s_0$, that is, $\overline{st_i} \lessdot s_0$

(vi) $\rho(x_i) \rhd_{\Gamma(x_i)} \sigma(st_i)$, that is, $v_i \rhd_{\tau_i} \sigma(st_i)$

So using Lemma 1.18 on $\mathcal{T}_1, \mathcal{E}_1, \mathcal{C}_1, (a), (b)$ and (c) gives us exactly what we shall show.

- Case $e = \{e_1 : x \textbf{ in } y \textbf{ using } x_1, ..., x_j\}$.

  We must have:

  (i)

$$\mathcal{T} = \frac{\begin{array}{c}\mathcal{T}_1\\ [x \mapsto \tau_1, x_1 \mapsto \textbf{int}, ..., x_j \mapsto \textbf{int}] \vdash e_1 : \tau_2\end{array}}{\Gamma \vdash \{e_1 : x \textbf{ in } y \textbf{ using } x_1, ..., x_j\} : \{\tau_2\}}$$

  with

$$\Gamma(y) = \{\tau_1\}$$
$$(\Gamma(x_i) = \textbf{int})_{i=1}^j$$

(ii)

$$\mathcal{E} = \frac{\begin{pmatrix} \mathcal{E}_i \\ [x \mapsto v_i, x_1 \mapsto n_1, ..., x_j \mapsto n_j] \ \vdash \ e_1 \downarrow v_i' \end{pmatrix}_{i=1}^{k}}{\rho \ \vdash \ \{e_1 : x \ \textbf{in} \ y \ \textbf{using} \ x_1, ..., x_j\} \downarrow \{v_1', ..., v_k'\}}$$

with

$$\rho(y) = \{v_1, ..., v_k\}$$
$$(\rho(x_i) = n_i)_{i=1}^{j}$$

(iii)

$$\mathcal{C} = \frac{\mathcal{C}_1}{[x \mapsto st_1, x_1 \mapsto s_1, ..., x_j \mapsto s_j] \ \vdash \ e_1 \Rightarrow_{s_1}^{s_0+1+j} (p_1, st_2)}{\delta \ \vdash \ \{e_1 : x \ \textbf{in} \ y \ \textbf{using} \ x_1, ..., x_j\} \Rightarrow_{s_1}^{s_0} (p, (st_2, s_b))}$$

with

$$\begin{aligned} \delta(y) &= (st_1, s_b) \\ (\delta(x_i) &= s_i')_{i=1}^{j} \\ p &= s_0 := \texttt{Usum}(s_b); \\ &\quad (s_i := \texttt{Distr}(s_b, s_i');)_{i=1}^{j} \\ &\quad S_{out} := \texttt{WithCtrl}(s_0, S_{in}, p_1) \\ S_{in} &= \texttt{fv}(p_1) \\ S_{out} &= \overline{st_2} \cap \texttt{dv}(p_1) \\ s_{i+1} &= \ s_i + 1, \forall i \in \{0, ..., j-1\} \end{aligned} \tag{1.46}$$

So $\tau = \{\tau_2\}, v = \{v_1', ..., v_k'\}, st = (st_2, s_b)$.

(iv) $\vdash \ \rho(y) : \Gamma(y)$ gives us $\vdash \ \{v_1, ..., v_k\} : \{\tau_1\}$, which must have the derivation:

$$\frac{(\vdash \ v_i : \tau_1)_{i=1}^{k}}{\vdash \ \{v_1, ..., v_k\} : \{\tau_1\}} \tag{1.47}$$

and clearly for $\forall i \in \{1, ..., j\}$, $\vdash \ \rho(x_i) : \Gamma(x_i)$, that is

$$(\overline{\vdash \ n_i : \textbf{int}})_{i=1}^{j} \tag{1.48}$$

.

(v) $\overline{\delta(y)} \lessdot s_0$ gives us

$$\overline{\delta(y)} = \overline{(st_1, s_b)} = \overline{st_1} +\!\!+ [s_b] \lessdot s_0 \tag{1.49}$$

and $(\overline{\delta(x_i)})_{i=1}^{j} \lessdot s_0$ implies $[s_1', ..., s_j'] \lessdot s_0$.

(vi) Since $\rho(y) \triangleright_{\Gamma(y)} \sigma(\delta(y)) = \{v_1, ..., v_k\} \triangleright_{\{\tau_1\}} \sigma((st_1, s_b))$, which must have the derivation:

$$\frac{\begin{pmatrix} \mathcal{R}_i \\ v_i \triangleright_{\tau_1} w_i \end{pmatrix}_{i=1}^{k}}{\{v_1, ..., v_k\} \triangleright_{\{\tau_1\}} (w, \langle \texttt{F}_1, ..., \texttt{F}_k, \texttt{T} \rangle)} \tag{1.50}$$

where $w = w_1 +\!\!+ ... +\!\!+ w_k$, therefore we have

$$\sigma(st_1) = w \tag{1.51}$$

$$\sigma(s_b) = \langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle. \tag{1.52}$$

Also, for $\forall i \in \{1, ..., j\}$, $\rho(x_i) \triangleright_{\Gamma(x_i)} \sigma(\delta(x_i)) = n_i \triangleright_{\mathbf{int}} \sigma(s'_i)$, which implies

$$(\sigma(s'_i) = \langle n_i \rangle)_{i=1}^{j} \tag{1.53}$$

First we shall show:

(vii) $\left\langle \begin{array}{l} s_0 := \mathtt{Usum}(s_b); \\ (s_i := \mathtt{Distr}(s_b, s'_i);)_{i=1}^{j} \\ S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1) \end{array}, \sigma \right\rangle \Downarrow^{\langle () \rangle} \sigma'$ by some $\mathcal{P}$

(viii) $\{v'_1, ..., v'_k\} \triangleright_{\{\tau_2\}} \sigma'((st_2, s_b))$ by some $\mathcal{R}$

Using P-Seq $(j+1)$ times, we can build $\mathcal{P}$ as follows:

$$\dfrac{\mathcal{P}_0 \quad \langle s_0 := \mathtt{Usum}(s_b), \sigma \rangle \Downarrow^{\langle () \rangle} \sigma_0 \qquad \dfrac{\mathcal{P}_1 \quad \langle s_1 := \mathtt{Distr}(s_b, s'_1), \sigma_0 \rangle \Downarrow^{\langle () \rangle} \sigma_1 \qquad \dfrac{\dfrac{\mathcal{P}_{j+1}}{\mathcal{P}_j \quad \langle S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1), \sigma_j \rangle \Downarrow^{\langle () \rangle} \sigma'} \vdots}{\left\langle \begin{array}{l} (s_i := \mathtt{Distr}(s_b, s'_i))_{i=2}^{j}; \\ S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1) \end{array}, \sigma_1 \right\rangle \Downarrow^{\langle () \rangle} \sigma'}}{\left\langle \begin{array}{l} (s_i := \mathtt{Distr}(s_b, s'_i);)_{i=1}^{j} \\ S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1) \end{array}, \sigma_0 \right\rangle \Downarrow^{\langle () \rangle} \sigma'}}{\left\langle \begin{array}{l} s_0 := \mathtt{Usum}(s_b); \\ (s_i := \mathtt{Distr}(s_b, s'_i);)_{i=1}^{j} \\ S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1) \end{array}, \sigma \right\rangle \Downarrow^{\langle () \rangle} \sigma'}$$

in which for $\forall i \in \{1, ..., j\}$, $\mathcal{P}_i$ is a derivation of $\langle s_i := \mathtt{Distr}(s_b, s'_i), \sigma_{i-1} \rangle \Downarrow^{\langle () \rangle} \sigma_i$.

For $\mathcal{P}_0$, with $\sigma(s_b) = \langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle$, we can build it as follows:

$$\text{by P-X-UsumF} \dfrac{\text{by P-X-Loop} \dfrac{\text{by P-XDucer} \dfrac{\text{by P-X-UsumT} \dfrac{}{\mathtt{Usum}(\langle \mathtt{T} \rangle) \downarrow \langle \rangle} \vdots \quad \mathtt{Usum}(\langle \mathtt{F}_2, ..., \mathtt{F}_k, \mathtt{T} \rangle) \downarrow \langle ()_2, ..., ()_k \rangle}{\mathtt{Usum}(\langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle) \downarrow \langle ()_1, ..., ()_k \rangle} \quad \text{by P-X-Termi} \dfrac{}{\mathtt{Usum}(\langle \rangle) \Downarrow^{\langle \rangle} \langle \rangle}}{\mathtt{Usum}(\langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle) \Downarrow^{\langle () \rangle} \langle ()_1, ..., ()_k \rangle}}{\langle s_0 := \mathtt{Usum}(s_b), \sigma \rangle \Downarrow^{\langle () \rangle} \sigma[s_0 \mapsto \langle ()_1, ..., ()_k \rangle]}$$

So $\sigma_0 = \sigma[s_0 \mapsto \langle ()_1, ..., ()_k \rangle]$.

Similarly, with $\sigma(s_b) = \langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle$ and $(\sigma(s'_i) = \langle n_i \rangle)_{i=1}^{j}$ from (1.53), we can build each $\mathcal{P}_i$ for $\forall i \in \{1, ..., j\}$ as follows:

$$\text{by P-X-DistrT } \frac{}{\texttt{Distr}(\langle \texttt{T} \rangle, \langle n_i \rangle) \downarrow \langle\rangle}$$
$$\vdots$$
$$\text{by P-X-DistrF } \frac{\texttt{Distr}(\langle \texttt{F}_2, ..., \texttt{F}_k, \texttt{T} \rangle, \langle n_i \rangle) \downarrow \langle \overbrace{n_i, ..., n_i}^{k-1} \rangle}{\texttt{Distr}(\langle \texttt{F}_1, ..., \texttt{F}_k, \texttt{T} \rangle, \langle n_i \rangle) \downarrow \langle \overbrace{n_i, ..., n_i}^{k} \rangle} \quad \text{by P-X-Termi } \frac{}{\texttt{Distr}(\langle\rangle, \langle\rangle) \Downarrow^{\langle\rangle} \langle\rangle}$$
$$\text{by P-X-Loop } \frac{\texttt{Distr}(\langle \texttt{F}_1, ..., \texttt{F}_k, \texttt{T} \rangle, \langle n_i \rangle) \Downarrow^{\langle()\rangle} \langle \overbrace{n_i, ..., n_i}^{k} \rangle}{\langle s_i := \texttt{Distr}(s_b, s_i'), \sigma_{i-1} \rangle \Downarrow^{\langle()\rangle} \sigma_{i-1}[s_i \mapsto \langle \overbrace{n_i, ..., n_i}^{k} \rangle]}$$

So $\forall i \in \{1, ..., j\}.\sigma_i = \sigma_{i-1}[s_i \mapsto \langle \overbrace{n_i, ..., n_i}^{k} \rangle]$.

Thus $\sigma_j = \sigma[s_0 \mapsto \langle ()_1, ..., ()_k \rangle, s_1 \mapsto \langle \overbrace{n_1, ..., n_1}^{k} \rangle, ..., s_j \mapsto \langle \overbrace{n_j, ..., n_j}^{k} \rangle]$.

Now it remains to build $\mathcal{P}_{j+1}$.

Since we have

$$\mathcal{T}_1 = \ [x \mapsto \tau_1, x_1 \mapsto \textbf{int}, ..., x_j \mapsto \textbf{int}] \ \vdash \ e_1 : \tau_2$$

$$(\mathcal{E}_i = \ [x \mapsto v_i, x_1 \mapsto n_1, ..., x_j \mapsto n_j] \ \vdash \ e_1 \downarrow v_i' \qquad)_{i=1}^{k}$$

$$\mathcal{C}_1 = \ [x \mapsto st_1, x_1 \mapsto s_1, ..., x_j \mapsto s_j] \ \vdash \ e_1 \Rightarrow_{s_1}^{s_0+1+j} (p_1, st_2)$$

Let $\Gamma_1 = [x \mapsto \tau_1, x_1 \mapsto \textbf{int}, ..., x_j \mapsto \textbf{int}], \rho_i = [x \mapsto v_i, x_1 \mapsto n_1, ..., x_j \mapsto n_j]$ and $\delta_1 = [x \mapsto st_1, x_1 \mapsto s_1, ..., x_j \mapsto s_j]$.
For $\forall i \in \{1, ..., k\}$, we show the following three conditions, which allows us to use IH with $\mathcal{T}_1$, $\mathcal{E}_i$, $\mathcal{C}_1$ later.

(a) $\forall x \in dom(\Gamma_1). \ \vdash \ \rho_i(x) : \Gamma_1(x)$

(b) $\forall x \in dom(\Gamma_1).\overline{\delta_1(x)} \lessdot s_0 + 1 + j$

(c) $\forall x \in dom(\Gamma_1).\rho_i(x) \rhd_{\Gamma_1(x)} \sigma_{ji}(\delta_1(x))$

TS: (a)
From (1.47) and (1.48) it is clear that

$$\forall x \in dom(\Gamma_1). \ \vdash \ \rho_i(x) : \Gamma_1(x)$$

TS: (b)
From (1.49), it is clear that $\overline{\delta_1(x)} = \overline{st_1} \lessdot s_0 + 1 + j$. From (1.46), for $\forall i \in \{1, .., j\}.\delta_1(x_i) = s_0 + i < s_0 + 1 + j$. Therefore,

$$\forall x \in dom(\Gamma_1).\overline{\delta_1(x)} \lessdot s_0 + 1 + j$$

TS: (c)
For $\forall i \in \{1, .., k\}$, we take $\sigma_{ji} \overset{S}{\sim} \sigma_j$ where $S = dom(\sigma_j) - (\overline{st_1} \cup \{s_1, ..., s_j\})$,

such that

$$\begin{aligned}\sigma_{ji}(st_1) &= w_i\\ \sigma_{ji}(s_1) &= \langle n_1\rangle\\ &\vdots\\ \sigma_{ji}(s_j) &= \langle n_j\rangle\end{aligned}$$

It is easy to show that

$$\sigma_{j1} \overset{S}{\sim} \sigma_{j2} \overset{S}{\sim} ... \overset{S}{\sim} \sigma_{jk} \overset{S}{\sim} \sigma_j \tag{1.54}$$

$$\sigma_{j1} \overset{S}{\bowtie} \sigma_{j2} \overset{S}{\bowtie} ... \overset{S}{\bowtie} \sigma_{jk} = \sigma_j \tag{1.55}$$

Also note that

$$S_{in} = \mathtt{fv}(p_1) \subseteq (\overline{st_1} \cup \{s_1, ..., s_j\}) \cap S = \emptyset \tag{1.56}$$

$$\overline{st_2} \subseteq (\overline{st_1} \cup \{s_1, ...s_j\} \cup \mathtt{dv}(p_1)) \cap S = \emptyset \tag{1.57}$$

From $\mathcal{R}_i$ in (1.50) we have $\rho_i(x) \triangleright_{\Gamma_1(x)} \sigma_{ji}(\delta_1(x))$
and it is clear that

$$\begin{aligned}&\rho_i(x_1)\triangleright_{\Gamma_1(x_1)}\sigma_{ji}(\delta_1(x_1))\\ &\vdots\\ &\rho_i(x_j)\triangleright_{\Gamma_1(x_j)}\sigma_{ji}(\delta_1(x_j))\end{aligned}$$

Therefore, $\forall x \in dom(\Gamma_1).\rho_i(x) \triangleright_{\Gamma_1(x)} \sigma_{ji}(\delta_1(x))$.
Then by IH (k times) on $\mathcal{T}_1$ with $\mathcal{E}_i$, $\mathcal{C}_1$ we obtain the following result:

$$(\langle p_1, \sigma_{ji}\rangle \Downarrow^{\langle()\rangle} \sigma'_{ji})_{i=1}^k \tag{1.58}$$

$$(v'_i \triangleright_{\tau_2} \sigma'_{ji}(st_2))_{i=1}^k \tag{1.59}$$

$$(\sigma'_{ji} \overset{<s_0+j+1}{=\!=\!=\!=\!=\!=\!=} \sigma_{ji})_{i=1}^k \tag{1.60}$$

$$s_0 + 1 + j \leq s_1 \tag{1.61}$$

$$\overline{st_2} \lessdot s_1 \tag{1.62}$$

Assume $S_{out} = \{s_{j+1}, ..., s_{j+l}\}$.(Note here s_{j+i} is not necessary equal to $s_j + i$, but must be $\geq s_j$).

There are two possibilities for $\mathcal{P}_{j+1}$:

– Subcase $\sigma_j(s_0) = \langle\rangle$, i.e., $k = 0$.
Then $(\sigma_j(s_i) = \langle\rangle)_{i=1}^j$. Also, with (1.50) and (1.51), we have $\forall s \in \overline{st_1}.\sigma_j(s) = \langle\rangle$; with (1.52), $\sigma_j(s_b) = \langle \mathtt{T}\rangle$. Thus

$$\forall s \in (\{s_0\} \cup S_{in}).\sigma_j(s) = \langle\rangle$$

Then we can use the rule P-Wc-Emp to build $\mathcal{P}$ as follows:

$$\frac{}{\langle S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1), \sigma_j\rangle \Downarrow^{\langle()\rangle} \sigma_j[(s_{j+i} \mapsto \langle\rangle)_{i=1}^l]}$$

So in this subcase, we take

$$\sigma' = \sigma_j[(s_{j+i} \mapsto \langle\rangle)_{i=1}^{l}] = \sigma[s_0 \mapsto \langle\rangle, s_1 \mapsto \langle\rangle, ..., s_{j+l} \mapsto \langle\rangle] \tag{1.63}$$

TS: (viii)
Since $k = 0$, then $v = \{\}$. Also, we have

$$\sigma'(s_b) = \sigma(s_b) = \langle \mathtt{T} \rangle$$
$$\forall s \in \overline{st_2}.\sigma'(s) = \langle\rangle$$

Therefore, $\sigma'((st_2, s_b)) = (\sigma'(st_2), \sigma'(s_b))$, with which we construct

$$\mathcal{R} = \frac{}{\{\} \rhd_{\{\tau_2\}} ((...(\langle\rangle), ...), \langle \mathtt{T} \rangle)}$$

as required.

– Subcase $\sigma_j(s_0) = \langle()|...\rangle$, i.e., $k > 0$.
Since we have (1.54), (1.58) and $\mathtt{fv}(p_1) \cap S = \emptyset$ from (1.56), it is easy to show that using Lemma 1.16 at most (k-1) times we can obtain

$$\left\langle p_1, (\overset{S}{\bowtie} \sigma_{ji})_{i=1}^{k} \right\rangle \Downarrow^{\langle()_1,...,()_k\rangle} (\overset{S}{\bowtie} \sigma'_{ji})_{i=1}^{k} \tag{1.64}$$

Let $\sigma'' = (\overset{S}{\bowtie} \sigma'_{ji})_{i=1}^{k}$. Also with (1.55), we replace both the start and ending stores in (1.64), giving us a derivation $\mathcal{P}'_{j+1}$ of

$$\langle p_1, \sigma_j \rangle \Downarrow^{\langle()_1,...,()_k\rangle} \sigma''$$

Now we build $\mathcal{P}_{j+1}$ using the rule P-Wc-Nonemp as follows:

$$\frac{\begin{array}{c}\mathcal{P}'_{j+1} \\ \langle p_1, \sigma_j \rangle \Downarrow^{\langle()_1,...,()_k\rangle} \sigma''\end{array}}{\langle S_{out} := \mathtt{WithCtrl}(s_0, S_{in}, p_1), \sigma_j \rangle \Downarrow^{\langle()\rangle} \sigma_j[(s_{j+i} \mapsto \sigma''(s_{j+i}))_{i=1}^{l}]}$$

So in this subcase we take

$$\begin{aligned}\sigma' &= \sigma_j[(s_{j+i} \mapsto \sigma''(s_{j+i}))_{i=1}^{l}] \\ &= \sigma[s_0 \mapsto \langle()_1, ..., ()_k\rangle, s_1 \mapsto \langle\overbrace{n_1, ..., n_1}^{k}\rangle, ..., s_j \mapsto \langle\overbrace{n_j, ..., n_j}^{k}\rangle, \\ &\quad s_{j+1} \mapsto \sigma''(s_{j+1}), ..., s_{j+l} \mapsto \sigma''(s_{j+l})]\end{aligned} \tag{1.65}$$

TS : (viii)
Let $\sigma'(st_2) = w'$, and $\sigma'_{ji}(st_2) = w'_i$.
For $\forall i \in \{1, ..., k\}$, by Definition 1.12 with (1.57), we get

$$w' = \sigma''(st_2) = w'_1 +\!+_{\tau_2} ... +\!+_{\tau_2} w'_k$$

Also, $\sigma'(s_b) = \sigma(s_b) = \langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle$, we now have $\sigma'((st_2, s_b)) = (\sigma'(st_2), \sigma'(s_b)) = (w', \langle \mathtt{F}_1, ..., \mathtt{F}_k, \mathtt{T} \rangle)$. With (1.59), we can construct $\mathcal{R}$ as follows:

$$\frac{(v'_i \rhd_{\tau_2} w'_i)_{i=1}^{k}}{\{v'_1, ..., v'_k\} \rhd_{\{\tau_2\}} \sigma'((st_2, s_b))}$$

as required.

(ix) TS: $\sigma' \overset{<s_0}{=\!=} \sigma$

Since $\forall s \in \{s_0\} \cup \{s_1, ..., s_j\} \cup \{s_{j+l}, ..., s_{j+l}\}.s \geq s_0$, with (1.63) and (1.65), it is clear $\forall s < s_0.\sigma'(s) = \sigma(s)$, i.e., $\sigma' \overset{<s_0}{=\!=} \sigma$ as required.

(x) TS: $s_0 \leq s_1$
From (1.61) we immediately get $s_0 \leq s_1 - 1 - j < s_1$.

(xi) TS: $\overline{(st_2, s_b)} \lessdot s_1$
From (1.49) we know $s_b < s_0$, thus $s_b < s_0 \leq s_1$. And we already have (1.62). Therefore,

$$\overline{(st_2, s_b)} = \overline{st_2} +\!+ [s_b] \lessdot s_1.$$

■

1.6 Scaling up

Bibliography

[Ble89] Guy E Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11):1526–1538, 1989.

[Ble95] Guy E Blelloch. Nesl: A nested data-parallel language.(version 3.1). Technical Report CMU-CS-95-170, School of Computer Science, Carnegie Mellon University, 1995.

[Ble96] Guy E Blelloch. Programming parallel algorithms. *Communications of the ACM*, 39(3):85–97, 1996.

[Mad13] Frederik M. Madsen. A streaming model for nested data parallelism. Master's thesis, Department of Computer Science (DIKU), University of Copenhagen, March 2013.

[Mad16] Frederik M. Madsen. *Streaming for Functional Data-Parallel Languages*. PhD thesis, Department of Computer Science (DIKU), University of Copenhagen, September 2016.

[PP93] Jan F Prins and Daniel W Palmer. Transforming high-level data-parallel programs into vector operations. In *Proceedings of the Fourth SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP'93, pages 119–128. ACM, 1993.

[PPCF95] Daniel W Palmer, Jan F Prins, Siddhartha Chatterjee, and Rickard E Faith. Piecewise execution of nested data-parallel programs. In *International Workshop on Languages and Compilers for Parallel Computing*, pages 346–361. Springer, 1995.